West Virginia Wetland Rapid Assessment Method (WVWRAM) Appendix:

ArcGIS Procedures and Python 2.7

April 10, 2019 Version 0.5

Watershed Assessment Branch
Division of Water and Wastewater Management
WV Department of Environmental Protection

Contents

5.5 ArcGIS Methods for Creating Input Layers

5.5.1 BRankInput: Site Biodiversity Rank Input Layer

Version date: 5 November 2017

Strategy: 3/16/2017 EAB
GIS method: 11/5/2017 Elizabeth Byers
Python code: N/A

Purpose:

Habitat Function: Input to Site Biodiversity Rank Override (Brank)

Description:

This layer is derived from sensitive WVDNR data on rare, threatened, or endangered species, high quality natural communities, and species in greatest need of conservation statewide. The layer includes spatial occurrence data attributed with unique taxa identifiers, Conservation Status Rank at the state (SRank) and global (GRank) level, and Occurrence Quality Ranks (OQRank).

The strategy to create this layer is to:

- Combine the SGCN data with the Biotics EO's, being sure to retain the Elcode, TaxonName, Grank, Srank, OQrank, and TaxGroup.
- QC the Elcode and Ranks for correct updated values.
- Round the ranks using the rounding rules for Site Biodiversity Rank.
- Delete records that have large spatial uncertainty, that are historic (last observation date before 1980), that are extirpated, and records with any other problems in their data quality.
- Add fields for Disjunct occurrences and for species/community differentiation.
- Export and add random ID tied to Elcode. Delete all fields with taxonomic information except whether a record is a species or a community occurrence.

Source Data:

Note that the first three datasets below are sensitive and maintained by WVDNR. Processing of these data must be done by authorized WVDNR staff or their designees. Data may not be shared without explicit permission from WVDNR.

- Elements Occurrence Data
    - SGCN_EOS2016
    - EO_SHAPE_Statewide_2017_10_10
- Element Ranking Data
    - Communities: PlotData_20171004.mdb (Table: WV Associations 2014)
    - Plants: TrackedPlantsHandout_10Mar2017.doc
    - Animals: WVNHP_RTE_Animals with ranks_2016Nov.pdf (public data from WVDNR website)

Method:

STEP 1: COMPILE DATA

Fields of Interest

SGCN fields	**EO_SHAPE fields**
Elcode	ELCODE
TaxonName	SNAME
Srank	S_RANK
Grank	G_RANK
OQrank	BASIC_EO_R
TaxGroup	NAME_CAT_1
	LAST_OBS_D (last observation date)
	EST_REP_AC (estimated representational accuracy)
	"EO rank comment" (not sure what this is called in Biotics)

plus spatial fields Shape, OBJECTID, Shape_Length, Shape_Area

Export SGCN_EOS2016 and EO_SHAPE_Statewide_2017_10_10 to create working copies SGCN and EO_SHAPE

In SGCN, delete the old Biotics records prior to importing the new ones
"Dataset" = 'EO_SHAPE_2017_01_04'

Add fields to SGCN in preparation for import

LAST_OBS_D (text, length = 50)
EST_REP_AC (text, length= 30)

In EO_SHAPE, add the fields of interest with the SGCN names shown above and field calculate to populate them.
Elcode (text, 254) = ELCODE
TaxonName (text, 254) = SNAME
Srank (text, 254) = S_RANK
Grank (text, 254) = G_RANK
OQrank (text, 254) = BASIC_EO_R (when field calculating, populate this with letter codes, not strings)
TaxGroup (text, 254) = NAME_CAT_1

Load data from EO_SHAPE into SGCN

STEP 2: ADD RANKS AND QC DATA

Notes on interpreting ranks for the purpose of assigning biodiversity significance ratings to sites:

- Elements with range ranks spanning two levels (e.g., G2G3) should be treated as if they had the higher (e.g., G2) rank;
- Elements with range ranks spanning three levels (e.g., G3G5) should be treated as the middle rank (e.g., G4);
- Elements with ranks such as G3? should be treated as if there were no question marks;
- Elements with a GU or G? rank should be treated as if they were G4;
- Elements with "Q"s attached to their global ranks (i.e., questionable taxa) should be treated at the next lower G rank (e.g., treat a G3Q as if it was a G4); however, this is open to the discretion of the senior Natural Heritage scientist and is dependent upon the rationale for the "Q" qualifier. For example, if there is taxonomic uncertainty as to whether a G5T1Q subspecies should be elevated to species status, the Q "penalty" should not be applied.
- Subspecies (elements with T-ranks): The rationale behind this process is accounting for the relative size of a subspecies' distribution to its overall species distribution.
    - If the T-rank is 3 or 4 units away from its G-rank (e.g., G5T1, G5T2, G4T1), treat the functional Grank as one unit lower than the T-rank (e.g., G5T1 = G2; G5T2 = G3).
    - If the T-rank is <3 units away from its G-rank (e.g., G4T3, G5T3), treat the functional G-rank as the T-rank (e.g., G4T2 = G2, G5T4 = G4).
- Element Occurrences with range ranks (e.g., AB) should be treated as if they were ranked at the lower of the two levels (e.g., B);
- Element Occurrences that are not yet ranked should be treated as if they were C-ranked.

Table Cleanup

- Populate empty fields as needed
    - blank OQrank = C
    - add missing Elcodes
- I made one change: Biotics record "Undefined Wetlands" (6) at Willis Ridge were renamed to "Summit Sinkhole" and given Elcode = CEWVSumSin and Ranks = S1 G5.

- Update all ranks against updated RTE lists from DNR. There are three RTE lists: animals, plants, communities.
    - Combine the three lists into one Excel table, load to ArcGIS and Join on Elcode; export.
        - If the ranking lists do not have the Elcode, you can join on TaxonName to add most of the Elcodes to the ranking table, then populate the non-matches in the ranking table by hand. The final ranking table can be joined to the SGCN layer.
- Round the SRanks and GRanks using the rounding rules, i.e., final SRanks are S1, S2, S3, S4, S5

- Round the OQranks using the rounding rules, i.e., A, B, C, D
- Some SGCN are not currently tracked by WVNHP. Assign these S4 & S5 species a rank of S3 for purposes of analysis. Do not change their GRank. Select "Elcode" NOT LIKE 'CE%' AND("Srank" = 'S4' OR "Srank" = 'S5')
- Check for hidden characters in Elcodes, which will disrupt calculations. There are lots in the snail records – invisible until you try to work with the data. Create a new field Elcode10 (text, 10 characters allowed) and Field Calculate = [Elcode]. The fields with hidden characters will be truncated and marked with an asterisk. Note that some of the longer non-standard Elcode fields will be starred also – these should be changed to 10-character Elcodes.
- Check for unique relationships: Every Elcode should have exactly one TaxonName, and the inverse should also be true. Also, every Elcode should have only one SRank. There are lots of problems in the birds (common names mixed in – I did not fix all of these) and snails (mishmash of new and old Elcodes and SRanks). Here's a quick way to find the problems (do this for Elcode | TaxonName, TaxonName| Elcode, and Elcode | SRank):

First compute a field that concatenates Elcode and TaxonName. A delimiter, "|", is optionally added.

```
Elcode          TaxonName       Concatenate

PGPIN01020      Abies balsamea   PGPIN01020|Abies balsamea
```

R-Click the `Concatenation` field and Select "Summarize", retaining the first occurrence of `Elcode`:

```
Concatenate                     Elcode          Count

PGPIN01020|Abies balsamea       PGPIN01020      15
```

On this new table, R-click `Elcode` and select "Summarize". In the table that is generated, sort the "Count" field in descending order. All of the Elcodes should have a count of "1". If any have a count higher than "1", then there are two taxon names for that Elcode.

```
Elcode          Count

PGPIN01020      1
```

Based on method at: https://gis.stackexchange.com/questions/71452/how-to-count-occurrences-of-one-field-grouped-by-values-of-another

Save this cleaned file, then export to another file for deleting records.

STEP 3: DELETE OLD OR UNCERTAIN RECORDS

Multipart to Singlepart: explode multipart features so that you don't mistakenly delete a small accurate source feature that is linked to a large low-res source feature.

Delete historic records and records with approximate locations

- Delete historic, failed to find, extirpated, or "unrankable" occurrences: "OQrank" IN ('F', 'F?', 'H', 'H?', 'X', 'X?', 'U')

- Delete records with LAST_OBS_D before 1980
- Delete records with last observation date prior to a known year that have low, very low or null representational accuracy: ("LAST_OBS_D" LIKE '%pre%' OR "LAST_OBS_D" LIKE '%PRE%' OR "LAST_OBS_D" LIKE '%Pre%') AND "EST_REP_AC" NOT IN ('Very High', 'High', 'Medium') (78 records)
- Delete records that have large spatial uncertainty, i.e., circular features with a radius of more than a kilometer, or non-circular features with a similar area (> 3,150,000 m^2)
    - Select "Shape_Area" > 40000000 and delete these 54 highly uncertain locational records.
    - Select "Shape_Area" > 3150000 (780 records). Delete these unless they are a wetland species that overlies a single wetland or a nearly contiguous wetland complex (I did not see any obvious ones).

STEP 4: ADD FIELDS NEEDED TO CALCULATE SITE BIODIVERSITY RANK

Add fields needed to calculate Site Biodiversity Rank

- SpecComm (text, length = 2): S = species, C = natural community
- Flag (text, length = 20). Ideally this field will come from the "EO rank comment" in Biotics. I consulted with WVNHP to populate it. It is currently not complete but good enough to use, with many of the plant and community disjuncts identified. In the process, we identified a few other elements with Granks in need of revision: flatrock riverscour = G1, summit sinkhole = G3, Meadow River oak-ash swamp = G1.
    - OnlyRange: only known occurrence of an element rangewide
    - BestState: best available occurrence in state (for G1, G2, S1 elements)
    - Best5Range: among 5 best occurrences rangewide (for G3 elements)
    - Best5Ecoregion: among 5 best community occurrences in an ecoregion (for G4-G5 communities)
    - Disjunct: disjunct occurrence
    - OnlyState (or EOCount = 1): only known occurrence in state
- EO_Count (short integer): count of Elcode10 (summarize field and join output table on Elcode10)
- Wetland (short integer): to calculate B-ranks for wetland sites, add this modifier.
    - 0 = Exclude from analysis: element would not occur in a wetland, i.e., taxon would not include wetlands in any part of its habitat, foraging area, or life cycle; vegetation community would not be a wetland nor typically contain wetland inclusions below minimum mapping size.
    - 1 = Include in analysis: element could occur in a wetland.
    - *Note that screening has been done for plants, communities, fish, snails, cave invertebrates, and mussels as of 11/03/2017. Other invertebrates and vertebrates have not yet been screened. Wide ranging species including mammals and birds probably do not need to be screened.*

STEP 5: ENCRYPT SENSITIVE FIELDS

Encrypt the data so that no taxa are identifiable

Export all records to another file, e.g., SGCN_trimmed_randomID

Generate random number ID

Summarize Elcode10 to determine the number of unique Elcodes. Export this table to Excel and generate unique random numbers: =RANDBETWEEN (1,9999). Fill about 4000 rows with random numbers, then filter for unique values. Paste the values next to the Elcodes. Re-load the table back to ArcGIS and Join on Elcode10.

Attribute table has the following fields, which will be used to create Site Biodiversity Ranks:
RandID: Randomly assigned unique element identification number
Srank: State Conservation Status Rank (S1-S5)
Grank: Global Conservation Status Rank (G1-G5)
OQrank: Occurrence Quality Rank (A-D)
SpecComm: S = species, C = natural community
Flag: Disjunct = disjunct, BestRange = among 5 best occurrences rangewide, OnlyRange = only known occurrence rangewide
EO_Count: count of the number of times each Elcode10 occurs in the dataset.

The rest of the fields can be deleted, creating an encrypted version of the data.

Export the resulting feature class and name it "BRankInputAll".

Select only the wetland elements (Wetland = 1) and re-export data with filename "BRankInput".

5.5.2 Calcareous Soils and Karst Layer: SSURGO exports

Version date: 27 February 2017

Strategy: completed 3/1/2017 EAB
GIS method: 3/1/2017 EAB
Python coding: not needed
Final verification by EAB: 3/1/2017

Purpose:

Habitat Function / Intrinsic Potential / Soils

Description:

Rationale: Calcareous soils and karst areas support unique, rich, and often rare biodiversity. They are particularly vulnerable to hydrologic damage and to invasive species encroachments.
Summary of strategy: Select calcareous soil series or soils with “karst” in the geomorphic description, respecitively.

Source Data:

M:\basemap\ssurgo\ssurgo.gdb
    Feature Class: ssurgo_wv

Output:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb
    Feature Class: CalcareousSoils
    Feature Class: Karst

Method:

Calcareous Soils
SSURGO query based on advice from Jared Beard, State Soil Scientist:

SSURGO table: ssurgo_wv
Select "muname" IN ('Fairplay (marl) silt loam', 'Lappans (marl) loam', 'Lappans (marl) silt loam', 'Massanetta loam', 'Massanetta silt loam')
108 out of 413438 selected

This selects soil polygons in Jefferson and Berkeley Counties, with a few additional polygons in Pendleton and Hardy Counties.

My original thought was to query the chorizon table for “claysizedcarb_r” > 0, but this returns only the polygons in Jefferson County.

Soils with geomorphic description of “karst”

SSURGO table: component_all
"geomdesc" LIKE '%karst%'
241 out of 9533 selected
Related tables – back to ssurgo_wv
5575 out of 413438 selected

Note:

This includes some but not all of the limestone areas of the state, stopping at certain county boundaries. So I think the geology layer is better for getting at karst soils statewide. However, the SSURGO data is much finer-scale and gets smaller patches nicely, which are sometimes missed by the geology layer. So, use both geology and soils layers to delineate karst areas.

5.5.3 DisturbedLand: Disturbed Land Cover Layer

Version date: 8 March 2017

Strategy: completed 4/6/2016 EAB
GIS method: 4/8/2016 EAB; verified 4/18/2016 EAB; revised 3/8/2017 to remove Godzilla polygons
Python coding: not needed
Final verification by EAB: 4/18/2016; 3/8/2017 EAB
Note: this layer can be re-created with each new NLCD update, i.e., every few years.

Purpose:

Water Quality Function / Opportunity aspect / LandUse50m & LandUseWshd factors

Description:

Rationale: Farming, grazing, golf courses, residential areas, commercial land uses, urban areas, and developed areas in general, are major sources of pollutants (Sheldon et al. 2005). Tilled fields are a source of nutrients, pesticides, and sediment. Pastures are a source of nutrients and pathogenic bacteria, and clearcut areas are a source of sediment (Sheldon et al. 2005).
Summary of strategy: Vectorize NLCD2011 and export disturbed land cover classes. Union with recent recent timber harvests, urbanized areas, and grazed pastures. For this layer, NLCD disturbed land cover classes are defined as

- 21 Developed, open space
- 22 Developed, low intensity
- 23 Developed, medium intensity
- 24 Developed, high intensity
- 31 Barren land, non-natural
- 81 Pasture/hay
- 82 Cultivated crops

Source Data:

- M:\basemap\landcover_grids\ Raster: landcover_2011_NLCD.img
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: PasturesNotHayfields
    - Feature Class: TimberHarvest
- M:\LayerFiles\arcsde_backup.gdb
    - Feature Dataset: tiger2010 Feature Class: urbanized_areas

Output:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: DisturbedLand

Method:

## Vectorize NLCD pixels

Conversion tools / From Raster / Raster to Polygon
Field: Value
Input: landcover_2011_NLCD.img
Output Feature Class: NLCDpolys
Note that I had to turn off Geoprocessing/Background processing for this to work. It takes some time to run.

## Select disturbed land cover types from NLCD.

Select by attributes.
Method: Create a new selection.
SELECT * FROM NLCDpolys WHERE: "grid_code" IN (21, 22, 23, 24, 31, 81, 82)
(2072074 out of 3540450 selected)

## Export disturbed land cover types.

R-click NLCDpolys / Data / Export Data
Export: Selected features
Output feature class: DisturbedNLCD

## Union disturbed land use selections.

ArcToolbox/Analysis Tools/Overlay/Union
Input Features: urbanized_areas, PasturesNotHayfields, TimberHarvest, DisturbedNLCD
Output Feature Class: DisturbedLand in WaterQualityDatasets.gdb
Join Attributes: ONLY_FID
Check box “Gaps Allowed” (default)

Note that with ArcEditor license, Union can only accept two layers at a time. The unions take some time to process. Dissolve the TimberHarvest layer prior to Union-ing.

## Dissolve DisturbedLand to reduce file size and processing time.

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: DisturbedLand

## Clip the layer to the state boundary

ArcToolbox / Clip
Input feature: DisturbedLand
Output: DisturbedLand_clip

## Check the layer for Godzilla polygons with more than 100,000 vertices

```
##       (Calculate field for VertexCount: !shape!.pointcount)
## Cut Godzilla polygons to decrease number of vertices
## Explode large multipart polygons as you see them
## Check geometry to be sure there aren’t any errors
```

5.5.4 DrainageArea: Drainage Area of Wetland Basic Geometry Layer

Version date: 25 January 2018

Strategy: 1/25/2018 EAB with advice from Mike Shank
GIS method: 1/25/2018 EAB
Python code: Yibing Han, WV GIS Tech Center
Final review by EAB: 2/15/2018

Purpose:

This layer is a basic spatial input to multiple wetland function metrics within Water Quality, Flood Attenuation, and Habitat/Ecological Integrity.

Description:

Drainage Area or Contributing Watershed of a Wetland Unit, with 27-meter resolution.

This method creates a feature class containing the polygonal boundary of the watershed that contributes flow to a Wetland Unit, and the area of that watershed. This layer contains a linking field (WUKey) to relate it to the Wetland Unit layer (WU_20150514 in WetlandUnits.gdb).

The DrainageArea layer was first developed and run statewide at 27-meter resolution by Mike Shank in 2016. Finer resolutions may be possible in the future but are extremely time-consuming (several weeks or more of server time).

For each new set of Wetland Unit polygons, the drainage area must be calculated. DrainageArea forms one of the basic input geometries for wetland functional assessment.

Source Data:

Flow direction model
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Watershed.gdb
  Raster: hydrogrid_16U_flowdir_27m

Wetland Units
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  Feature Class: WU_20150514

##Note that this method takes a very long time to run on large datasets.

Strategy:

1. Create DrainageArea feature class to hold watershed polygon output in WetlandUnits.gdb
2. Begin LOOP: Note that Wetland Units must be processed one at a time because of overlapping watershed boundaries. For each Wetland Unit polygon:
   a. Copy Wetland Unit polygon to a temporary feature class
      i. Select by attributes where WUKey = 1
      ii. Export polygon to feature class "temp"

        iii. Go through all steps below
        iv. Next time through loop select WUKey = 2, and so on until all Wetland Units have been processed
    b. Convert temporary feature class to a temporary grid
        i. Use the same cell size as the flow direction grid: 27m
        ii. Use the WUKey of the Wetland Unit polygon as the grid value, which will carry over to the output feature class. This allows you to associate the wetland polygon with its drainage area.
        iii. Snap the Wetland Unit grid to the flow direction grid.
    c. Run watershed command using temporary grid as input, and the processing extent set to the flow direction grid. Note that you can calculate drainage for areas. Points are often tougher to match up exactly to the stream channels on imperfect elevation models.
    d. Convert watershed grid to a polygon
    e. Append polygon to DrainageArea feature class
3. Return to beginning of LOOP until all Wetland Units have been processed
4. Add field to DrainageArea feature class to store WUKey

Method:

## Create feature class to hold watershed polygon output in WetlandUnits.gdb

Create polygon feature class in WetlandUnits.gdb: DrainageArea
Add field: grid_code (long integer)

## BEGIN LOOP; start with WUKey = 1 and repeat until all Wetland Units are processed

## Copy Wetland Unit polygon to a temporary feature class

In WetlandUnits.gdb, open attribute table of feature class WU_20150514
Select by attributes where WUKey = 1 *(then 2,3,4,…n)*
Export selected features to feature class "temp"

## Convert temporary feature class to a temporary grid

Conversion Tools / To Raster / Polygon to Raster
Input features: temp
Value Field: WUKey
Output Raster Dataset: tempras
Cell assignment type: CELL_CENTER
Priority Field: NONE
Cellsize: 27
Environment Settings / Processing Extent
    Extent: Default
    Snap raster: hydrogrid_16U_flowdir_27m

## In order to capture very thin or small Wetland Units, use Feature to Raster instead of Polygon to Raster; or Set WUKey field also as Priority field in Polygon to Raster.

## Run watershed command using temporary grid as input.

Spatial Analyst Tools / Hydrology / Watershed
Input flow direction raster: hydrogrid_16U_flowdir_27m
Input raster or feature pour point data: tempras
Pour point field: VALUE
Output raster: temprasout
Environment Settings / Processing extent
    Extent: Same as layer hydrogrid_16U_flowdir_27m
    Snap raster: blank

## Convert watershed grid to a polygon

Conversion Tools / From Raster / Raster to Polygon
Input raster: temprasout
Field: WUKey
Output polygon features: temppolyout
Do not check "simplify polygons"

## Append polygon to DrainageArea feature class

DrainageArea / Load data / Input data = temppolyout

## RETURN TO BEGINNING OF LOOP
## END LOOP when all Wetland Units are processed

## All Wetland Units have now been added to DrainageArea feature class

## Add field to DrainageArea feature class to store WUKey

Add field WUKey (long integer)
Field calculate WUKey = grid_code

## DrainageArea is ready to be used for analysis

## End of procedure

5.5.5 FloodplainARAFEMA: Composite Floodplain Layer

Version date: 8 March 2017

Strategy: Completed 4/13/2016 EAB
GIS method: 4/20/2016 EAB
Python code: not needed but layer needs to be created. Started 6/13/2016 MCA; Completed 11//28/2016 JCC
Final review by EAB: 11/28/2016; godzillas fixed and reviewed again 3/8/2017 EAB

Purpose:

Input to Flood Attenuation / Opportunity / Floodplain Ratio

Description:

This layer is a composite of the FEMA 100-year floodplain layer and the TNC Active River Area base zone data.

Source Data:

- Active River Layer floodplain subset
    - M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\ActiveRiverArea_gdb\ara_wv_514
        - DESC_SHORT = (Base Zone, wetflat OR Base Zone, non-wetflat OR input water cells)
- FEMA 100-year floodplain
    - M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\wvFloodHazardFeatures_WVGISTC_20130410\wvFloodHazardFeatures20130207.gdb\wvFldZone_20130410_wgs84wmA

Method:

## Create composite ARA / FEMA floodplain layer

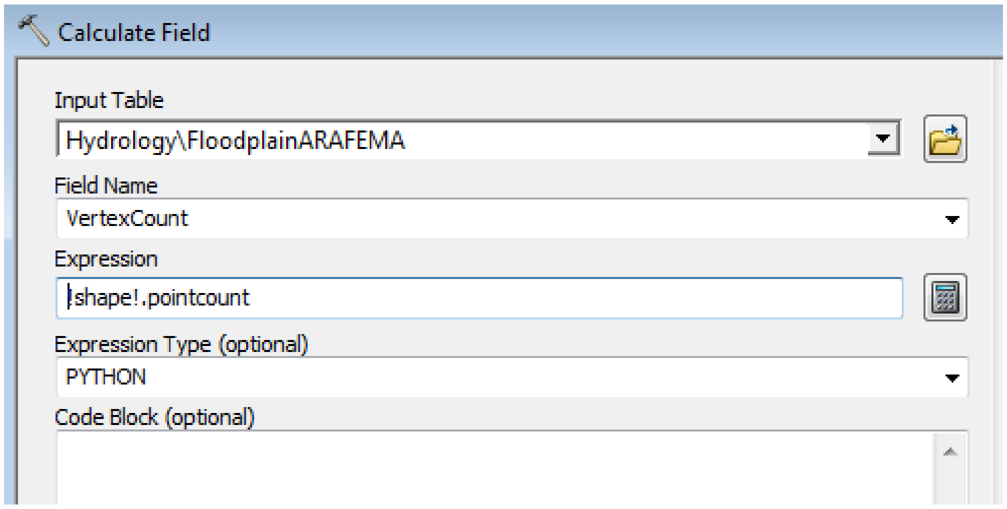
ArcToolbox / Data Management Tools / Projections and Transformations / Raster / Project Raster
Input Raster: ara_wv_514
Output raster: ara_wv_514_WGS1984_AS
Output coordinate system: WGS_1984_Mercator_Auxilary_Sphere (Same as wvFldZone_20130410_wgs84wmA
ArcToolbox / Conversion Tools / From Raster / Raster to Polygon
Input raster: ara_wv_514_WGS1984_AS (base zone pixels only)
Output polygon features: ARA_BaseStreamDissolve
Feature Class: ARA_BaseStreamDissolve) *
## Note: The same projection is need for this to work.

ArcToolbox / Analysis Tools / Overlay / Union
Input features: ARA_BaseStreamDissolve
wvFldZone_20130410_wgs84wmA
Output feature class: FloodplainARAFEMA1 (in Floodplain/FloodplainData.gdb)
Join Attributes: ONLY_FID

Check box “Gaps allowed” (default)

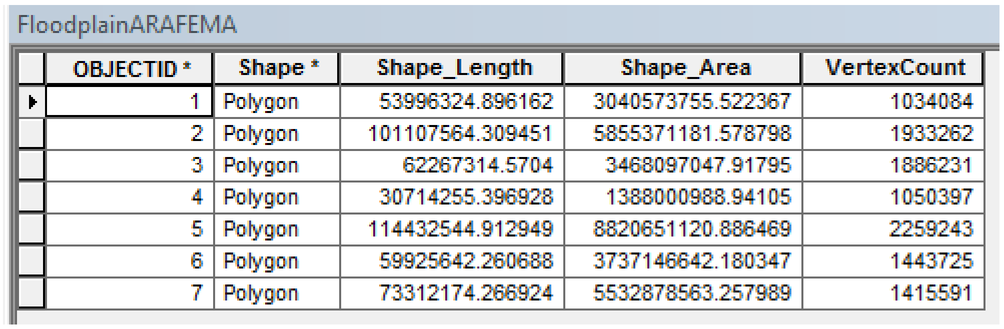
ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: FloodplainARAFEMA1
Output feature: FloodplainARAFEMA

This procedure creates “Godzilla” polygons with excessive numbers of vertices. Calculate the number of vertices:



Input Table
Hydrology\FloodplainARAFEMA
Field Name
VertexCount
Expression
!shape!.pointcount
Expression Type (optional)
PYTHON
Code Block (optional)

The result shows that the 7 polygons have over 1 million vertices each.


FloodplainARAFEMA

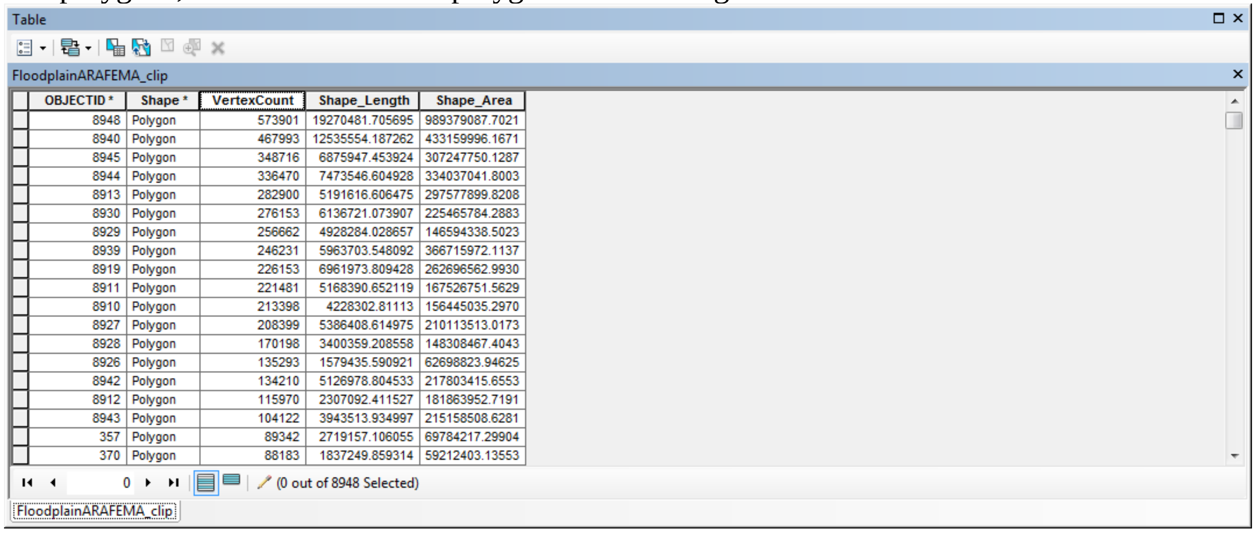
OBJECTID *	Shape *	Shape_Length	Shape_Area	VertexCount
1	Polygon	53996324.896162	3040573755.522367	1034084
2	Polygon	101107564.309451	5855371181.578798	1933262
3	Polygon	62267314.5704	3468097047.91795	1886231
4	Polygon	30714255.396928	1388000988.94105	1050397
5	Polygon	114432544.912949	8820651120.886469	2259243
6	Polygon	59925642.260688	3737146642.180347	1443725
7	Polygon	73312174.266924	5532878563.257989	1415591

We want to reduce the number of vertices, and it is okay to greatly increase the number of polygons. Use the Advanced Editing menu. Select the first polygon and Explode Multipart Feature. Repeat for all the polygons with excessive numbers of vertices. A reasonable target is no more than 200,000 vertices, or to run on machines with less computing power, aim for 50,000 max.

There are now 38,273 polygons.

Re-calculate the Vertex Count and sort descending to see how large the remaining polygons are. There are still 8 godzilla polygons.
Select the largest polygon and display it. In this case, much of the polygon is outside the state boundary.

Clip to the state boundary.
Re-calculate the Vertex Count. Clipping to the state boundary reduced the number of vertices and polygons, but we still have 13 polygons that are large.



Table

FloodplainARAFEMA_clip

OBJECTID *	Shape *	VertexCount	Shape_Length	Shape_Area
8948	Polygon	573901	19270481.705695	989379087.7021
8940	Polygon	467993	12535554.187262	433159996.1671
8945	Polygon	348716	6875947.453924	307247750.1287
8944	Polygon	336470	7473546.604928	334037041.8003
8913	Polygon	282900	5191616.606475	297577899.8208
8930	Polygon	276153	6136721.073907	225465784.2883
8929	Polygon	256662	4928284.028657	146594338.5023
8939	Polygon	246231	5963703.548092	366715972.1137
8919	Polygon	226153	6961973.809428	262696562.9930
8911	Polygon	221481	5168390.652119	167526751.5629
8910	Polygon	213398	4228302.81113	156445035.2970
8927	Polygon	208399	5386408.614975	210113513.0173
8928	Polygon	170198	3400359.208558	148308467.4043
8926	Polygon	135293	1579435.590921	62698823.94625
8942	Polygon	134210	5126978.804533	217803415.6553
8912	Polygon	115970	2307092.411527	181863952.7191
8943	Polygon	104122	3943513.934997	215158508.6281
357	Polygon	89342	2719157.106055	69784217.29904
370	Polygon	88183	1837249.859314	59212403.13553

0 (0 out of 8948 Selected)

FloodplainARAFEMA_clip

Explode Multi-part Features again for polygons with more than 100,000 vertices.
Re-calculate the Vertex Count. 9 polygons still have counts > 200,000.
Select the largest polygon and zoom to it. Find good places to cut it (subwatersheds along major rivers require only a single cut) and cut it into smaller chunks. Repeat for all the other large polygons.
Re-calculate the Vertex Count to verify that no “Godzillas” are left.

5.5.6 HUCWetlandSizeUniq: Wetland Size/Uniqueness by 12-digit HUC Layer

Version date: 5 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/5/2017 EAB
Python code: not needed
Final review by EAB: 10/5/2017

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)

Update schedule:

This layer should be re-created when there are significant updates to the National Wetlands Inventory for West Virginia.

Description:

Rationale:

GIS Method (no field method):
Make heat map of 12-digit HUC watershed layer with the following fields, highlighting HUCs in the top 5% of the state.

- Type diversity: number of unique NWI codes in the watershed (e.g., PEMA, PEMC, PEMCx) including types that have no vegetation component (e.g., PUBH, R3US2). Do not include spoil wetlands (%s%)
- Density: number of vegetated NWI polygons; many of these polygons may be contiguous with each other, forming a single wetland.
- Proportional Area: proportion of the watershed's total area occupied by vegetated wetlands as mapped by NWI.
- Area of largest vegetated Wetland Unit in the HUC12.

Source Data:

- M:\LayerFiles\arcsde_backup.gdb
  - Feature Dataset: basemap_physical_non_replica
    - Feature Class: watersheds_12digit
  - Feature Dataset: basemap_cultural_non_replica
    - Feature Class: SDE_wvbound
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
  - Feature Dataset: CONUS_WVWetlandsProj
    - Feature Class: EnhWVWetland
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandResults_EAB.gdb
  - Feature Class: WU_VegAll

Method:

## STEP 1: Prepare the HUC watershed layer

## Clip 12-digit HUC to state boundary

ArcToolbox / Analysis / Extract / Clip
Input features: watersheds_12digit
Clip features: SDE_wvbound
Output Feature Class:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Hydrology.gdb
    - Feature Class: HUC_12digit1

## Reduce number of attributes to decrease processing time
## Note that HUC_12 is a unique identifier for the watershed but NU_12_NAME is not unique

ArcToolbox / Data Management / Fields / Delete Field
Input Table: HUC_12digit1
Drop field: Select all except HUC_12 and HU_12_NAME

## STEP 2: Determine the largest vegetated wetland in each 12-digit HUC (MaxVegArea)

## Spatial Join HUC watershed with vegetation-attributed Wetland Units

ArcToolbox / Analysis / Overlay / Spatial Join
Target Features: HUC_12digit1
Join Features: WU_VegAll
Output feature class: Hydrology.gdb / HUC_12digit2
Join operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features: retain the following
HUC_12
HUC_12_NAME
SHAPE_Length
SHAPE_Area
VegArea (R-click, select MERGE RULE / Maximum)
Match option: INTERSECT

## Add field to store the largest vegetated wetland area and set NULL intersections to zero

Open attribute table of HUC_12digit2
Add field "MaxVegArea" and Field Calculate MaxVegArea = VegArea
SELECT * FROM HUC_12digit2 WHERE: MaxVegArea IS NULL
Field Calculate MaxVegArea = 0

## Find the wetlands that intersect more than one 12-digit HUC.

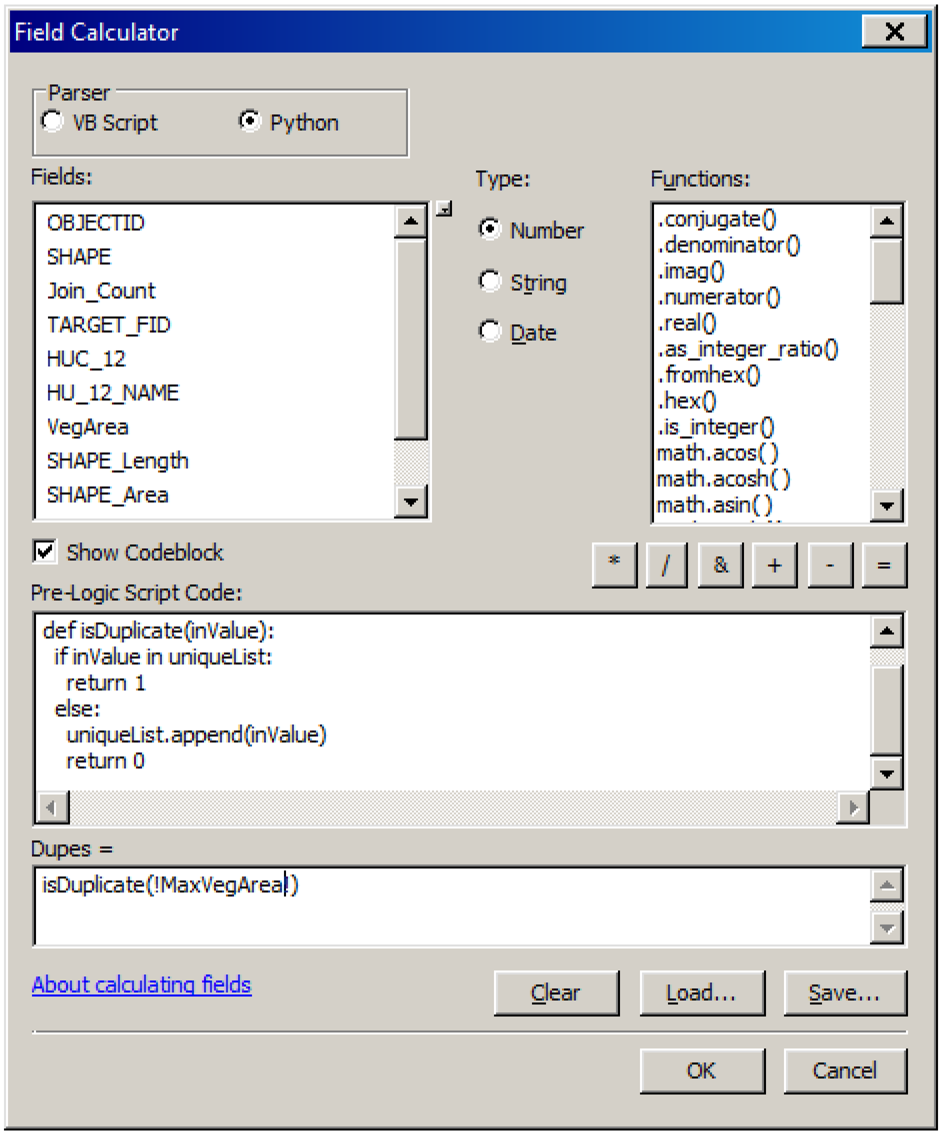
Instructions below from https://gis.stackexchange.com/questions/183472/identify-duplicate-attributes

Instructions provided demonstrate how to use the Field Calculator to identify duplicate field values. Single occurrences and the first occurrence of multiple values are flagged with 0. Duplicates are flagged with 1.

Create a new field. Set the type as short or long integer and accept the other defaults. Right-click the newly created field and select Field Calculator. Select the Python parser. Ensure that the 'Show Codeblock' option is checked. Paste the following code into the Pre-Logic Script Code box:

```
uniqueList = []

def isDuplicate(inValue):

  if inValue in uniqueList:

    return 1

  else:

    uniqueList.append(inValue)

    return 0
```

Type '`isDuplicate(!Field!)`' in the lower expression box and replace the word 'Field' with the name of the field that contains the duplicated values. Click OK. All duplicate records are designated with a value of 1 and non-duplicate records are designated with a value of 0 in the new field.



Parser: VB Script [✓ Python]

Type: [✓ Number] String Date

[✓] Show Codeblock

Pre-Logic Script Code:

```
def isDuplicate(inValue):
  if inValue in uniqueList:
    return 1
  else:
    uniqueList.append(inValue)
    return 0
```

Dupes =

```
isDuplicate(!MaxVegArea!)
```

## Fourteen wetlands cross HUC boundaries. Examine these manually to put the correct area in each watershed. Some are just touching a boundary and don't need replacement. Below are the manual revisions to MaxVegArea:

Buffalo Creek-Ohio River (050901011007): 0

    Horseshoe Run: 102938

Brush Run-Greenbrier River: 65827

Bull Run-Cheat River: 7903

Devilhole Creek-North Fork Hughes River: 6180

Middle Grave Creek: 328183

    Grave Creek: 140738

    Lee Creek: 9387 (largest vegetated wetland has been converted to golf course pond)

    Little Creek-Monongahela River: 3553

    Little Otter Creek-Elk River: 1462

    Mill Creek-Meadow River: 442947

    Little Clear Creek: 578758

Guyan Creek: 54196
Rich Creek-Gauley River: 46555
Laurel Creek-Gauley River: 13356
Scary Creek-Kanawha River: 122268
Buffalo Creek-Kanawha River: 253100
Shields Run-North Branch Potomac River: 196589

## STEP 3: Number of unique NWI codes in the watershed (DiversNWI)

## Intersect Enhanced NWI and HUC watershed

ArcToolbox / Analysis / Overlay / Intersect
Input Features: EnhWVWetland
                        HUC_12digit1
Output feature class: NWIExports.gdb / NWI_HUC12digit_join
Join attributes: ALL
Output type: INPUT

## Delete the spoil wetlands

R-click NWI_HUC12_join and select Edit Features / Start Editing
Open attribute table of NWI_HUC12_join
SELECT * FROM NWI_HUC12_join WHERE: "ATTRIBUTE" LIKE '%s%'
Delete selected features
Editor / Save Edits
Editor / Stop Editing

## Add field and concatenate NWI attribute with HUC_12

Open attribute table of NWI_HUC12_join
Add field Concatenate (text, length = 50)
Field Calculate Concatenate = [HUC_12] & " | " & [ATTRIBUTE]

## Summarize by HUC watershed

Open attribute table of NWI_HUC12_join
R-click the header for the field “Concatenate” and select “Summarize”
Select a field to summarize: Concatenate
Choose one or more summary statistics to be included in the output table: HUC_12 (first)
Specify output table: NWI_HUC12_summary1

## Summarize again for the count of unique attributes by HUC watershed

Open attribute table of NWI_HUC12_summary1
R-click the header for the field “First_HUC_12” and select “Summarize”
Select a field to summarize: First_HUC_12

Choose one or more summary statistics to be included in the output table: Concatenate (first)
Specify output table: NWI_HUC12_summary2

## Join results back to HUC watershed

R-click HUC_12digit2 and select Joins and Relates / Joins
Join attributes from a table
Choose the field: HUC_12
Choose the table: NWI_HUC12_summary2
Choose the field in the table: First_HUC12
Join options: Keep all records

## Export joined layer

R-click HUC_12digit2 and select Export Data / All features
Output feature class: Hydrology.gdb / HUC_12digit3

## Add field to store number of unique NWI codes in the watershed (DiversNWI)

Open attribute table of HUC_12digit3
Add field DiverseNWI (short integer) and Field Calculate DiversNWI = [Cnt_First_HUC_12]

## Set null intersections to zero

Open attribute table of HUC_12digit3
SELECT * FROM HUC_12digit3 WHERE: "DiverseNWI" IS NULL
Field Calculate DiverseNWI = 0

## STEP 4: Number of vegetated NWI polygons in the watershed (DensVegNWI)

## Select the forest, shrubland, emergent, moss, and aquatic bed vegetation.
## (Note that the VegAll layer created in WU_VegByLP and used in WU_VegAll includes only the Palustrine polygons. The layer below includes an additional 3 vegetated lake polygons and 7 vegetated riverine polygons)

Clear all selections
SELECT * FROM EnhWVWetland WHERE:
"ATTRIBUTE" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PML%' OR "ATTRIBUTE" LIKE 'L%AB%' OR"ATTRIBUTE" LIKE 'L%EM%' OR "ATTRIBUTE" LIKE 'R%AB%' OR"ATTRIBUTE" LIKE 'R%EM%'

## Create layer of all vegetation from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features

Output feature class: \NWIExports.gdb \ NWI_VegPalLacRiv

## Determine the number of vegetated NWI polygons in each HUC

ArcToolbox / Analysis / Overlay / Spatial Join
Target Features: HUC_12digit3
Join Features: NWI_VegPalLacRiv
Output feature class: Hydrology.gdb / HUC_12digit4
Join operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features: retain the following
HUC_12
HUC_12_NAME
SHAPE_Length
SHAPE_Area
MaxVegArea
DiversNWI
Attribute (R-click, select MERGE RULE / Count)
Match option: INTERSECT

## Add field to store the number of vegetated NWI polygons and set NULL intersections to zero

Open attribute table of HUC_12digit4
Add field "DensVegNWI" (short integer) and Field Calculate DensVegNWI = [ATTRIBUTE]
SELECT * FROM HUC_12digit4 WHERE: DensVegNWI IS NULL
Field Calculate DensVegNWI = 0

## Check the zero values of MaxVegArea and compare to zeros for DensVegNWI. If they are not the same (three were different on 10/5/2017) then check the HUCs manually and correct them.

## STEP 5: Proportion of the watershed's total area occupied by vegetated wetlands (RatioVeg)

## Spatial join vegetated NWI wetlands to HUC

ArcToolbox / Analysis / Overlay / Spatial Join
Target Features: HUC_12digit4
Join Features: WU_VegPalLacRiv
Output feature class: Hydrology.gdb / HUC_12digit
Join operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features: retain the following
HUC_12
HUC_12_NAME
SHAPE_Length

SHAPE_Area
MaxVegArea
DiversNWI
DensVegNWI
SHAPE_Area_1 (R-click, select MERGE RULE / Sum)
Match option: INTERSECT

## Add field to store the proportion of vegetated wetlands and set NULL intersections to zero

Open attribute table of HUC_12digit
Add field “RatioVeg” (float) and Field Calculate RatioVeg = [SHAPE_Area_1] / [SHAPE_Area]
SELECT * FROM HUC_12digit WHERE: RatioVeg IS NULL
Field Calculate RatioVeg = 0

5.5.7 ILF_banks: In-Lieu Fee Mitigation Sites and Mitigation Banks Layer

Version date: 24 January 2018

Strategy: 10/19/2017 EAB
GIS method: 10/19/2017 EAB
Python code: not needed
Final review by EAB: 10/19/2017

Purpose:

Input to Habitat / Value to Society / HInvest

Update schedule:

This layer should be updated at least every 5 years, or more often if time permits.

Description:

Rationale: Wetlands that have been restored, enhanced or created represent investments of time and money, and are of high value to society. In-lieu fee sites and mitigation banks are the most common type of project in West Virginia, although voluntary restoration also occurs.

Method:

- Check web for updates to RIBITS: https://ribits.usace.army.mil/.
- Check with WVDEP DWWM/WIB for updates to in-lieu fee sites, mitigation banks, or other mitigation sites. WVDNR Coordination Unit may also have information.
- Note that this is a point file; as polygons become available, sites should be moved to the "RestoredWetland" layer.
- Note that most project wetlands have not been mapped in the National Wetlands Inventory as of 2017.

Merge points from all sources and write to ILF_banks point file.

Output:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
Feature Class: ILF_banks

5.5.8 IEIUMa2010v32: Index of Ecological Integrity U Mass Layer

Version date: 28 November 2018

Strategy: 11/15/2018 EAB
GIS method: 11/28/2018 JSH
Python coding: not needed
Final review by EAB: 11/28/2018

Purpose:

Index of Ecological Integrity is an input to the Landscape Integrity metric (LandInteg).

Update schedule for layer:

Layer should be updated (re-created from UMass source data) when U Mass releases major updates. The next update is expected in 2019. We should receive notification from UMass when it is released.

Description:

U Mass developed this layer as part of their Conservation and Prioritization System (CAPS) for the northeastern states. The layer shows scores for an Index of Ecological Integrity (high scores = higher integrity), normalized for the state of West Virginia. Version 3.2 (2010). 30m pixels.

Scott Jackson (sjackson@umass.edu,11/19/2018) writes: *The DSL data sources are from around 2010. Some are a little older; some are newer. The landcover is based on 2011 NLCD and TNC's Ecological Systems Map. I believe that there will be an updated version coming out once the new NLCD data are released.*

Elizabeth Byers (WVDEP) asks: *Pipelines and powerlines are mapped as "shrubland & grassland" and are ranked higher than the adjacent wetlands or forests that they fragment. I wonder if this is an unintended artifact of your analysis?*

Scott Jackson replies: *We have no way to score ecological integrity in absolute terms (e.g. x units of integrity). So, we created a relative scoring system that compares shrub swamps to shrub swamps, salt marshes to salt marshes. Powerlines and pipelines are generally categorized as shrublands. Because much of the northeast lacks shrublands of any size, powerline/pipeline shrublands often score pretty high. It is not meant to suggest that a powerline shrubland has more integrity than an adjacent forested wetland, it means that the powerline shrubland is better than many other shrublands in that state (assuming that you used the IEI that was scaled to state boundaries).*

Source Data:

Project webpage: http://www.umasscaps.org/

Data download page: http://www.umasscaps.org/data_maps/dsl.html#WV

Source file: DSL_IEI_state_2010_v3.2_WV.tif

Method:

Download data and process as described below, from Jack Hopkins' notes on file processing (November 2018):

# Converting pixel values from (0.1 - 1) to (1 - 100)

Use the Spatial Analyst tool "Raster Calculator"

Double click input layer from the available layers and variables box

Multiple the layer by 100 using the expression *100 example expression.. ("IEIUMa2010v32" * 100)

Click Ok to save as a new output raster file

# Creating fixed integer values instead of floating values

Use the Spatial Analyst tool "Int"

Load the previously created raster as the input raster

Click Ok to save as a new output raster file

# Creating an attribute table

Use the Data Management tool "Build Raster Attribute Table"

Load the previously created raster as the input raster

Click Ok to create an attribute table with an object Id field, pixel value field, and count field

# Assigning the No Data pixels a value of zero

Use the Spatial Analyst tool Reclassify

Load the previously created rater as the input raster

In the reclassification table window type in the desired new values

Click Ok to save as a new output raster file

# Clipping the Image to the WV state boundary

Make sure that both the input raster and the WV State Boundary layer are in the table of contents

Open the Attribute table for the WV State Boundary and select the attribute so that it is highlighted

Open the Image Analysis function located in the Windows tab at the top toolbar in Arc Map

You'll see a list of all available layers in the top most box

Make sure to add a check mark next to the raster layer (if this isn't already) as well as click on the name of the layer causing it to highlight itself

Click the now available “Clip tool” in the processing window

This creates a new raster that has not been saved but should now have been clipped to the wv state boundary. This new clip should also be located in the table of contents

Right click this new layer, hover over the word data, and then click export data

# You might have to create the attribute table again for the new raster layer

Use the Data Management tool “Build Raster Attribute Table”

Load the previously created raster as the input raster

Click Ok to create an attribute table with an object Id field, pixel value field, and count field

Output Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb\IEIUMa2010v32

5.5.9 InfrasturctureWetlands: Wetlands with Public Use Infrastructure Layer

Version date: 12 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/12/2017 EAB
Python code: not needed
Final review by EAB: 10/12/2017

Purpose:

Input to Habitat / Value to Society / PublicUse

Update schedule:

This layer should be updated at least every 5 years, or whenever staff become aware of new boardwalks or educational/research programs at wetlands.

Description:

Rationale: Wetlands with boardwalks, nature trails, on-going educational programs, or sustained scientific research programs are of high value to society.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
  - Feature Dataset: CONUS_WVWetlandsProj
    - Feature Class: EnhWVWetland

Method:

Create named wetland polygons where there are known boardwalks, nature trails, educations programs, or sustained scientific research programs. Educational and research programs should continue for more than 2 years. Priority is given to adding wetland polygons that are not already included in a named wetland database (e.g. ExemplaryWetlands).

Often wetland polygons can be copied or traced from the National Wetlands Inventory (NWI). However, if they are not mapped in the NWI or another source, they must be heads-up digitized. The boundaries can be approximate, e.g. +- 100m, and do not need to be attributed to NWI codes. In wetland functional assessment, this layer will be intersected with Wetland Units, and therefore the boundaries do not need to be exact.

Data Fields:

Name [text 50 characters]
Give the wetland a name based on, in this order: (1) name on USGS topographic map, (2) name on signage at the wetland or local maps including the wetland, (3) name used informally by users of the wetland, or (4) name based on nearest stream or other natural landmark.

Boardwalk [short integer]

0	No boardwalk present
1	Boardwalk present

NatureTrail [short integer]

0	Wetland not accessible via maintained trail
1	Wetland is accessible via maintained trail

PublicOutreach [text 50 characters]
Briefly describe the educational or outreach activity that takes place at the wetland

ResearchSite [text 50 characters]
Briefly describe the sustained scientific research that takes place at the wetland

MapSource [text 50 characters]
Briefly describe the data source used to create the wetland polygon, e.g., "copied from NWI" or "heads-up digitized".

Comment [text 50 characters]
Add brief comments, such as name of educational organization or date of boardwalk construction

5.5.10 Karst Composite Layer

Version date: 18 September 2017

Strategy: completed 9/16/2017 EAB
GIS method: 9/18/2017 EAB; verified 9/18/2017 EAB
Python coding: not needed
Final verification by EAB: 9/18/2017 EAB
Note: this layer should be re-created as significant updates are made to the SSURGO soils database, perhaps every 10 years.

Purpose:

Habitat and Ecological Integrity Function / Potential aspect / Karst factor

Description:

<u>Rationale</u>: A rich and distinctive flora and fauna are characteristic of calcareous wetlands.

<u>Summary of strategy</u>: Union of Karst Geology, Karst Soils and Calcareous Soils.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: KarstGeology
        - Note: this feature class is an export from the state geologic map: In "geology_250k/Map Units", SELECT * FROM ALL_polygons WHERE: "TYPE" IN ('dolostone', 'limestone') (250 out of 5175 selected).
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb
    - Feature Class: KarstSoils
        - Note: this feature class is an export of statewide karst soils from SSURGO 2016:
            - SSURGO table: component_all
            - "geomdesc" LIKE '%karst%'
            - 241 out of 9533 selected
            - Related tables – back to ssurgo_wv
    - Feature Class: CalcareousSoils
        - Note: this feature class is an export of statewide calcareous soil map units (Fairplay, Lappans, Massanetta) from SSURGO 2016, based on consultation with Jared Beard, NRCS state soil scientist.

Output:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: KarstComposite

Method:

## Union KarstGeology, KarstSoils, and CalcareousSoils

ArcToolbox/Analysis Tools/Overlay/Union
Input Features: KarstSoils, CalcareousSoils

Output Feature Class:
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb\KarstComposite1
Join Attributes: ONLY_FID
Check box "Gaps Allowed" (default)

ArcToolbox/Analysis Tools/Overlay/Union
Input Features: KarstGeology, KarstComposite1
Output Feature Class:
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb\KarstComposite
Join Attributes: ONLY_FID
Check box "Gaps Allowed" (default)

Note: if the second Union doesn't work because of difficulty opening the "KarstComposite1" feature class, then export the feature class to a shapefile and re-do the Union.

5.5.11 NSPA: Wetland is in a Natural Streams Preservation Act Watershed Layer

Version date: 16 March 2016

Strategy: 2/14/2016
GIS method: completed 3/16/2016
Python code: not needed – this is the procedure to create a spatial dataset
Final review by EAB: 3/16/2016

Purpose:

Water Quality Function
This procedure creates a layer showing the watersheds drained by the streams in the Natural Streams Preservation Act.

Update Schedule for Layer:

None, except in the unlikely event that the legislation is updated

Description:

NSPA. Wetland is in the contributing watershed of a stream reach protected by the Natural Streams Preservation Act (2 points). These include (a) Greenbrier River from its confluence with Knapps Creek to its confluence with the New River, (b) Anthony Creek from its headwaters to its confluence with the Greenbrier River, (c) Cranberry River from its headwaters to its confluence with the Gauley River, (d) Birch River from the Cora Brown bridge in Nicholas county to the confluence of the river with the Elk River, and (e) New River from its confluence with the Gauley River to its confluence with the Greenbrier River.
http://www.legis.state.wv.us/WVcode/Code.cfm?chap=22&art=13

- GIS layer showing stream reaches in NSPA is at: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\DNRWetlandData.gdb, Feature Class: ntrlStrmPresrvationActStrms. This version of the NSPA reaches, from the DNR server, is better-registered and more complete than the version on the DEP server (M:\wr\WTRSHD_BRANCH_INTERNAL\STREAM FILES\Natural Stream Preservation\nat_strms_preservation.shp)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\DNRWetlandData.gdb
    - Feature Class: ntrlStrmPresrvationActStrms.
- M:\basemap\watersheds_10digit.shp

Method:

Create layer showing watersheds drained by NSPA streams

Open 10-digit watershed layer and NSPA layers. Select 10-digit watersheds drained by NSPA streams and export to a new feature class “WaterQualityDatasets.gdb / NatStrProAct_HUC10.shp”.

Final data layer is stored in:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Watershed.gdb
    - Feature Class: NatStrPreACt_HUC10

5.5.12 NWIExports.gdb: NWI Exports Layer

Version date: 18 February 2018

Strategy: 2/15/2018 EAB
GIS method: completed 2/15/2018 EAB
Python code: not needed – this is the procedure to create a spatial dataset
Final review by EAB: 2/18/2018

Purpose:

Input to HydSW, IrrEdge, StreamEdge, VegByLP, WFlowPath metrics

Update Schedule for Layers:

Every 5 years, or whenever the National Wetlands Inventory is significantly updated for the state. Note that the NWI Data Verification Tools must be run on the statewide dataset, and any errors fixed, prior to creating exports.

Description:

The National Wetlands Inventory contains mapped polygons of rivers, lakes, and ponds, in additional to vegetated wetlands. These aquatic resources are of importance in understanding the function of adjacent or nearby wetlands. Exports used in functional assessment include: NWIOpenWater, Rivers, Lakes, and RiversLakes.

Source Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
Feature Dataset: CONUS_WVWetlandsProj
Feature Class: EnhWVWetland

Output Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIExpo
rts.gdb
Feature Class: NWIOpenWater
Feature Class: Rivers
Feature Class: Lakes
Feature Class: RiversLakes

Method:

## Select NWI Open Water polygons. These are attributed as lakes, rivers, and open
## water palustrine (aquatic bed, unconsolidated bottom, unconsolidated shore) AND have a
## hydrologic regime that is (permanently flooded, semipermanently flooded, intermittently exposed)
## AND are not spoil.

Clear all selections.
Select * FROM EnhWVWetland WHERE: ("ATTRIBUTE" LIKE 'L%' OR"ATTRIBUTE" LIKE 'R%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PUB%' OR "ATTRIBUTE" LIKE 'PUS%') AND ("ATTRIBUTE" LIKE '%H%' OR "ATTRIBUTE" LIKE '%G%' OR "ATTRIBUTE" LIKE '%F%') AND "ATTRIBUTE" NOT LIKE '%s%'

## Create open water layer from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\NWIExports.gdb
    Feature Class: NWIOpenWater

## Select NWI rivers and create Rivers layer

Clear all selections
SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'R%'

R-click EnhWVWetland and select Data / Export Data
Export: Selected features
Output feature class: Rivers

## Select NWI lakes and create Lakes layer

Clear all selections
SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'L%'

R-click EnhWVWetland and select Data / Export Data
Export: Selected features
Output feature class: Lakes

## Select the NWI rivers and lakes and create RiversLakes layer

Clear all selections.
Select * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'R%' OR "ATTRIBUTE" LIKE 'L%'

R-click EnhWVWetland and select Data / Export Data
Export: Selected features
Output feature class: RiversLakes

5.5.13 RunoffLand: Lands that Produce Runoff Layer

Version date: 28 February 2017

Strategy: completed 4/21/2016 EAB
GIS method: 4/21/2016 EAB; completed 11/28/2016 JCC; revised 2/28/2017 MCA to reduce the size of Godzilla polygons, which were causing calculations to hang
Python coding: coding not needed, but need to create layer
Final verification by EAB: 11/28/2016; revisions verified 2/28/2017
Note: this layer can be re-created with each new NLCD/ssurgo/timber update, i.e., every few years.

Purpose:

Flood Attentuation Function / Opportunity aspect / Runoff50m & RunoffWshd factors

Description:

Rationale: Impervious surfaces, urban areas, agricultural areas, mining, industrial and commercial land uses, and recently logged areas contribute to increased runoff. Soils with low infiltration and high runoff characteristics also contribute to runoff.
Summary of strategy: Vectorize NLCD2011 and export land cover classes that produce runoff (impervious surfaces plus cultivated crops). Union with recent timber harvests and SSURGO soils that have high runoff/low infiltration characteristics.

NLCD Values selected
21 Developed, Open Space
22 Developed, Low Intensity
23 Developed, Medium Intensity
24 Developed, High Intensity
31 Barren Land
82 Cultivated Crops

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: TimberHarvest
    - Feature Class: NLCDpolys (this was created for the DisturbedLand layer)
- M:\basemap\ssurgo\ssurgo.gdb
    - Feature Class: ssurgo_wv

Output:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: RunoffLand

Method:

## Select disturbed land cover types from NLCD.

Select by attributes.
Method: Create a new selection.
SELECT * FROM NLCDpolys WHERE: "grid_code" IN (21, 22, 23, 24, 31, 82)

## Export disturbed land cover types.

R-click NLCDpolys / Data / Export Data
Export: Selected features
Output feature class: RunoffNLCD

## Dissolve to reduce file size and processing time.

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: RunoffNLCD
Output feature class: RunoffNLCD_diss
Check box "Create multipart features" (default)

##Note: disable background processing for this to run.

## Union land use selections.
For the union to work, the Z and M values must be removed from the TimerHarvest feature class. I union can't be performed when one feature class has these values and the other (RunoffNLCD_diss) does not.

ArcToolbox/Conversion Tools/To Geodatabase/Feature Class to Geodatabase
Input features: TimberHarvest
Output Geodatabase: RunoffLand.gdb
Environments… (Button): Disable the Z and M values

ArcToolbox/Analysis Tools/Overlay/Union
Input Features: TimberHarvest, RunoffNLCD_diss
Output Feature Class: RunoffLand1
Join Attributes: ONLY_FID
Check box "Gaps Allowed" (default)

## Dissolve to reduce file size and processing time.

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: RunoffLand1
Output feature class: RunoffLand1_diss
Check box "Create multipart features" (default)

## Select SSURGO soils with high runoff/low infiltration characteristics.

Select by attributes.
Method: Create a new selection.

SELECT * FROM ssurgo_wv WHERE: "hydgrpdcd" = 'D'

## **Export SSURGO soils with high runoff/low infiltration characteristics.**
R-click ssurgo_wv / Data / Export Data
Export: Selected features
Output feature class: RunoffSoil

## Dissolve to reduce file size and processing time.

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: RunoffSoil)
Output feature class: RunoffSoil_diss
Check box “Create multipart features” (default)

## Union land use and soil selections.

ArcToolbox/Analysis Tools/Overlay/Union
Input Features: RunoffSoil_diss, RunoffLand1_diss
Output Feature Class: RunoffLand2
Join Attributes: ONLY_FID
Check box “Gaps Allowed” (default)

## Dissolve to reduce file size and processing time.

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: RunoffLand2
Output feature class: RunoffLand in WaterQualityDatasets.gdb
Check box “Create multipart features” (default)

## Simplify Godzilla polygons with large numbers of vertices, in other words,
## use Simplify Polygons tool on each overly-complex polygon

5.5.14 Septic: Septic System Failure Risk Layer

Version date: 17 March 2016

Strategy: completed 3/11/2016 EAB
GIS method: completed 3/11/2016 EAB; verified 3/11/2016 EAB
Python coding: started & completed 3/17/2016 MCA
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function, Opportunity aspect, input to Discharges (discharges to the wetland within 100 meter buffer)
This procedure creates a data layer. No points at this level, maximum of 1 point can be assigned in Discharges variable

Update schedule:

Layer should be updated (re-created from source data) approximately every 5 years, or with major updates to SAMB or Tiger sources.

Description:

Rationale: Septic systems can pollute groundwater because nitrogen is not removed underground. Plumes of nitrogen from septic systems can be traced at least 250ft in the groundwater (Aravena and others 1993). Use an aerial photograph of the unit to determine if there are any residences within 250ft of the unit. Septic systems are still in common use in many areas outside of city boundaries. If you are outside city limits in areas with lots of 1/2 acre or larger you can assume the houses are on septic systems unless another type of water supply system is indicated on the water supply GIS layer.
<u>Summary of strategy</u>: Calculate as known septic systems (NPDES permit) or presence of structures. Exclude sewered areas, urbanized areas, and areas with low or very low risk of septic failure.

Source Data:

- M:\wr\owrnpdes_.shp
- M:\basemap\WVSAMB\structures_SAMB_points_UTM83.shp
  - Geometry Type: Point
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\WaterQualityDatasets.gdb
  - Feature Class: SeweredAreas
  - Feature Class: SepticFailureRiskStatsgo
- M:\LayerFiles\arcsde_backup.gdb
  - Feature Dataset: tiger2010 Feature Class: urbanized_areas

Method:

## PART 1: Select structures that intersect sewered areas or urbanized areas.

Select by Location
Selection method: select features from
Target layer: structures_SAMB_points_UTM83
Source layer: SeweredAreas
Spatial selection method: intersect the source layer feature

Select by Location
Selection method: add to the currently selected features in
Target layer: structures_SAMB_points_UTM83
Source layer: urbanized_areas
Spatial selection method: intersect the source layer feature

## Reverse selection.

Open attribute table of structures_SAMB_points_UTM83 and click “Switch Selection”

## Select only those structures that intersect moderate and high risk septic failure areas.

Select by attributes
SELECT * FROM SepticFailureRiskStatsgo WHERE: "SepticZone" = 'high' OR "SepticZone" = 'moderate'

Select by Location
Selection method: select from the currently selected features in
Target layer: structures_SAMB_points_UTM83
Source layer: SepticFailureRiskStatsgo
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

## Export points.

R-click structures_SAMB_points_UTM83 and Data/Export Data/Export selected features
Output feature class: Septic

## PART 2: Select septic permits from owrnpdes_ and export to point feature class.

Select by attributes from owrnpdes_
SELECT * FROM owrnpdes_ WHERE "perm_type" = 'Septic Tank'

R-click owrnpdes_ and Data/Export Data/Export selected features
Output feature class: SepticTanks

## Select septic permits that intersect sewered areas or urbanized areas.

Select by Location

Selection method: select features from
Target layer: SepticTanks
Source layer: SeweredAreas
Spatial selection method: intersect the source layer feature

Select by Location
Selection method: add to the currently selected features in
Target layer: SepticTanks
Source layer: urbanized_areas
Spatial selection method: intersect the source layer feature

## Reverse selection.

Open attribute table of SepticTanks and click “Switch Selection”

## Select only those permits that intersect moderate and high risk septic failure areas.

Select by attributes
SELECT * FROM SepticFailureRiskStatsgo WHERE: "SepticZone" = 'high' OR "SepticZone" = 'moderate'

Select by Location
Selection method: select from the currently selected features in
Target layer: SepticTanks
Source layer: SepticFailureRiskStatsgo
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

## Export points.

R-click structures_SAMB_points_UTM83 and Data/Export Data/Export selected features
Output file: SepticTankRisk

## PART 3: Append SepticTankRisk to Septic to include both septic tanks and structures.

In ArcCatalog, R-click Septic and Load/Load Data
Input data: SepticTankRisk
Add
Check radio button “I do not want to load all features into a subtype” (default)
No need to link any target fields with matching source fields
Check radio button “Load all of the source data”

5.5.15 SepticFailureRiskStatsgo: Septic Failure Risk from NRCS Statsgo Layer

Version date: 24 January 2018

Strategy & GIS method: 3/16/2016 EAB
Python coding: not needed
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function, Opportunity aspect, Discharges, input to Septic System Failure Risk (Septic) Layer.

Update schedule:

Layer should be updated (re-created from NRCS source data) approximately every 5 years, or with major updates to NRCS data portal.

Description:

Rationale: Soils with high septic failure risk are more likely to produce groundwater pollution.

NRCS has mapped soils with high septic failure risk in its Statsgo dataset. When I checked back in 2018, Statsgo appears to have been replaced with the Web Soil Survey. I found the Septic data in:
Soil Data Explorer / Suitabilities and Limitations Ratings / Sanitary Facilities / Septic Tank Absorption Fields. It may be worth looking at other NRCS products like the gridded SSURGO, which may be better or more easily downloaded data. Also, check with Steve Stutler or whomever is managing TMDL mapping for WVDEP / WAB, as they may have more knowledge of the best update source.

Source Data:

Download from NRCS.

Method:

Download data and save to location below.

Output Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\WaterQualityDatasets.gdb\SeweredAreas

5.5.16 Sewered Areas: Sewered Areas Layer

Version date: 2 Febraury 2018

Strategy & GIS method: 3/16/2016 Steve Stutler, WVDEP / WAB
Python coding: not needed
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function, Opportunity aspect, Discharges, input to Septic System Failure Risk (Septic) Layer.

Update schedule:

Layer should be updated (re-created from source data) approximately every 5 years, or with major updates to TMDLs and associated sewered areas mapping.

Description:

Rationale: Sewered areas are unlikely to receive groundwater pollution from septic systems.

Source Data and Method:

Sewered areas have been mapped within TMDL watersheds. This coverage is currently maintained by Steve Stutler, WVDEP / Watershed Assessment Branch / TMDL / Listing & Reporting.
Steve's process is to check the NPDES permits and outlets and create a shapefile of records that are likely to have sewage in them, e.g., package plants, publicly owned treatment works, and collection systems. He then figures out the areas covered by those sewered outlets, often by going to the facility and having someone familiar with the system draw a sketch map on an air photo or other map. Steve then converts this field map to GIS polygons.

Output Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\WaterQualityDatasets.gdb\SeweredAreas

5.5.17 Swimming Areas: Swimming Areas Layer

Version date: 22 January 2018

Strategy: 2/14/2016
GIS method: completed 3/17/2016
Python code: not needed – this is the procedure to create a spatial dataset
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function / Value to Society / WQ Use
This procedure creates a layer showing known public swimming areas in the state.

Update Schedule:

Every 5 years.

Description:

Water quality is of particular importance in public swimming areas. Wetlands upstream of public swimming areas contribute to better water quality by filtering out pollutants, nutrients, and sediment.

Source Data:

Internet sources, especially:
http://www.swimmingholes.org/wv.html (last updated 2.2016)
and interviews with state staff about known swimming areas

Output Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: SwimmingAreas2016

Method:

As new swimming areas are identified or closed, update layer as appropriate.

5.5.18 TimberHarvest: Recent Timber Harvests Statewide Layer

Version date: 20 October 2017

Strategy: completed 3/14/2016 EAB
GIS method: completed 3/14/2016 EAB
Updates: 10/20/2017 EAB
Python coding: not needed

Purpose:

Water Quality Function, Opportunity aspect
This method outputs a "TimberHarvest" feature class for use in computing the Landuse50m variable

Description:

Compilation of recent logging permits from WVDOF (Steve Harouff) and timber sales from WVDNR (Jeremy Rowan).
<u>Summary of strategy</u>: Select recent harvests, i.e., the last five years or 2010 to the present. Buffer points to create 80-acre polygons (average size of cut), which corresponds to a buffer distance of 321 meters. Union the layers.

Update schedule:

Every two years

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Timber Harvest
  - LSCA_HARVEST_NOTIFS_021716.shp
  - LSCA_HARVEST_POLYS_021716.shp
  - TimberSalesFeb2016WVDNR.shp
  - *And updates to the layers above*
    - *LSCA_Harvest_Polys_031017.shp*

Input Variables:

None

Method:

## Select recent sales (2010 to present) from WVDNR data.

Select by attributes
Layer: TimberSalesFeb2016WVDNR
Method: Create a new selection
SELECT * FROM TimberSalesFeb2016WVDNR WHERE: "YearContra" > 2010 AND "FEATURE" NOT IN ('Leave Area', 'No Cut area') AND "TYPE" NOT IN ('Leave', 'No Sale', 'Removed')

## Export selection to new feature class

R-click TimberSalesFeb2016WVDNR / Data / Export Data / Export selected features
Output feature class: TimberHarvest_DNR

## Select active permits from WVDOF polygon data, which dates back to 2012.
## Note that starting in 2018, these data must be filtered for the last 5 years.

Select by attributes
Layer: LSCA_Harvest_Polys_031017.shp
Method: Create a new selection
SELECT * FROM LSCA_Harvest_Polys_031017.shp WHERE: "Active_Sta" = 'Active'

## Export selection to new feature class

R-click LSCA_Harvest_Polys_031017.shp / Data / Export Data / Export selected features
Output feature class:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\WaterQualityDatasets.gdb
    - Feature Class: TimberHarvest_DOF_031017

## Select recent permits (2010 to present) from WVDOF point data.

Select by attributes
Layer: LSCA_HARVEST_NOTIFS_021716.shp
Method: Create a new selection
SELECT * FROM LSCA_HARVEST_NOTIFS_021716.shp WHERE: "START" > date '2010-01-01'

## Export selection to new feature class

R-click LSCA_HARVEST_POLYS_021716.shp / Data / Export Data / Export selected features
Output feature class: TimberHarvest_DOFpt

## Buffer points by 321 meters to create 80 acre polygons

Analysis Tools / Proximity / Buffer
Input Features: TimberHarvest_DOFpt
Output Feature Class: TimberHarvest_DOFpt_Buffer
Distance (Linear Unit): 321 meters
Dissolve Type: NONE

## Union the three timber layers into a single polygon feature class.

Analysis Tools / Overlay / Union

Input Features: TimberHarvest_DOFpt_Buffer
TimberHarvest_DOF_031017
TimberHarvest_DNR
Output Feature Class: TimberHarvest
Join Attributes: ONLY_FID
Check box “Gaps Allowed”

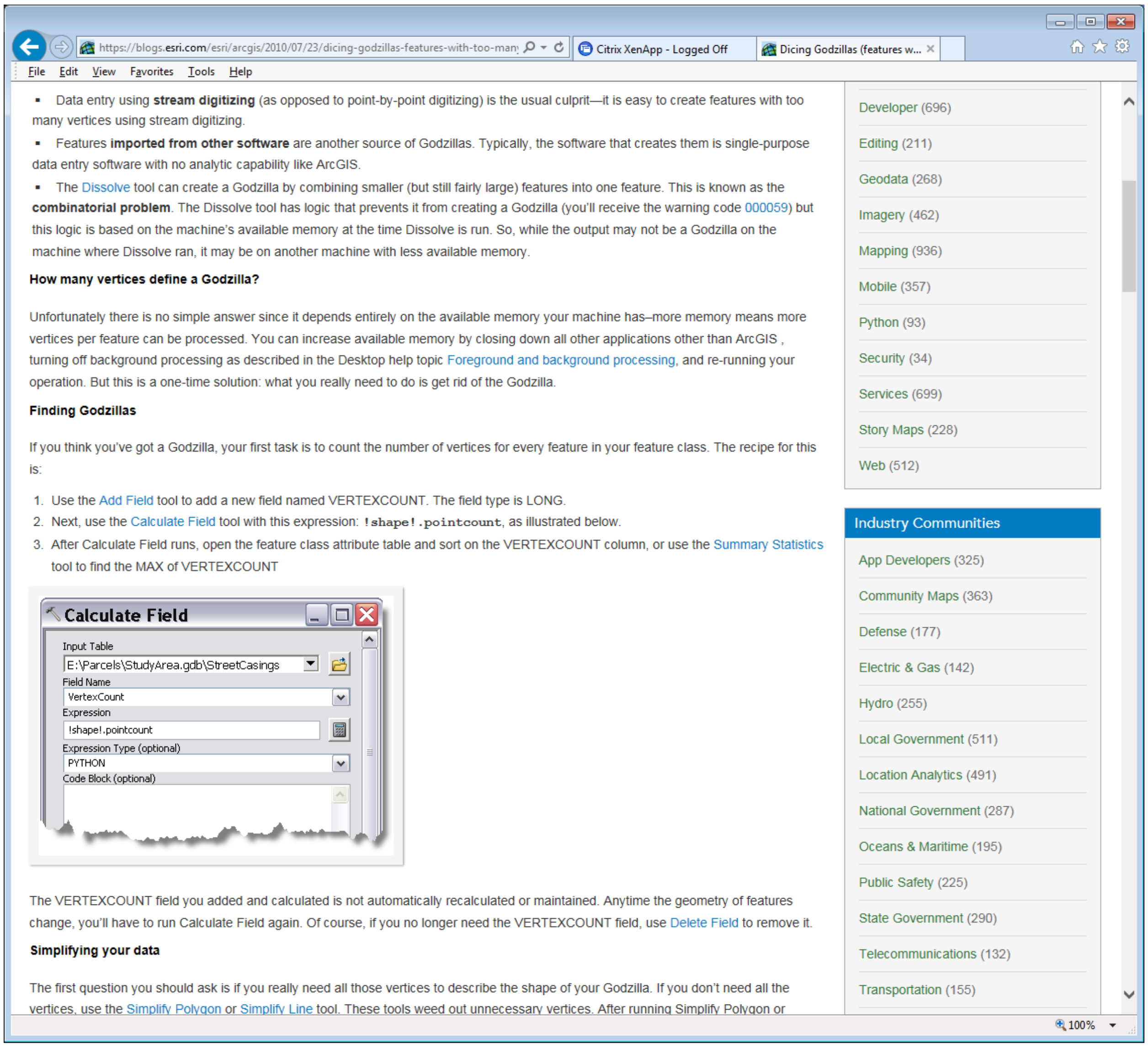
## Dissolve TimberHarvest

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: TimberHarvest
Do not allow multi-part features

## Check the layer for Godzilla polygons with more than 100,000 vertices
##   (Calculate field for VertexCount: !shape!.pointcount)
## Cut Godzilla polygons to decrease number of vertices
## Explode large multipart polygons as you see them
## Check geometry to be sure there aren’t any errors

## Simplify Godzilla polygons with large numbers of vertices, in other words,
## use Simplify Polygons tool on each overly-complex polygon

## Check the layer for Godzilla polygons with more than 100,000 vertices
## Add long integer field “VertexCount” to TimberHarvest
##   ArcToobox/Fields/Calculate field
for VertexCount: !shape!.pointcount)


https://blogs.esri.com/esri/arcgis/2010/07/23/dicing-godzillas-features-with-too-many

Citrix XenApp - Logged Off

Dicing Godzillas (features w...

File Edit View Favorites Tools Help

- Data entry using **stream digitizing** (as opposed to point-by-point digitizing) is the usual culprit—it is easy to create features with too many vertices using stream digitizing.
- Features **imported from other software** are another source of Godzillas. Typically, the software that creates them is single-purpose data entry software with no analytic capability like ArcGIS.
- The Dissolve tool can create a Godzilla by combining smaller (but still fairly large) features into one feature. This is known as the **combinatorial problem**. The Dissolve tool has logic that prevents it from creating a Godzilla (you'll receive the warning code 000059) but this logic is based on the machine's available memory at the time Dissolve is run. So, while the output may not be a Godzilla on the machine where Dissolve ran, it may be on another machine with less available memory.

How many vertices define a Godzilla?

Unfortunately there is no simple answer since it depends entirely on the available memory your machine has–more memory means more vertices per feature can be processed. You can increase available memory by closing down all other applications other than ArcGIS , turning off background processing as described in the Desktop help topic Foreground and background processing, and re-running your operation. But this is a one-time solution: what you really need to do is get rid of the Godzilla.

Finding Godzillas

If you think you've got a Godzilla, your first task is to count the number of vertices for every feature in your feature class. The recipe for this is:

1. Use the Add Field tool to add a new field named VERTEXCOUNT. The field type is LONG.
2. Next, use the Calculate Field tool with this expression: `!shape!.pointcount`, as illustrated below.
3. After Calculate Field runs, open the feature class attribute table and sort on the VERTEXCOUNT column, or use the Summary Statistics tool to find the MAX of VERTEXCOUNT

Calculate Field

Input Table
E:\Parcels\StudyArea.gdb\StreetCasings

Field Name
VertexCount

Expression
!shape!.pointcount

Expression Type (optional)
PYTHON

Code Block (optional)

The VERTEXCOUNT field you added and calculated is not automatically recalculated or maintained. Anytime the geometry of features change, you'll have to run Calculate Field again. Of course, if you no longer need the VERTEXCOUNT field, use Delete Field to remove it.

Simplifying your data

The first question you should ask is if you really need all those vertices to describe the shape of your Godzilla. If you don't need all the vertices, use the Simplify Polygon or Simplify Line tool. These tools weed out unnecessary vertices. After running Simplify Polygon or

Developer (696)
Editing (211)
Geodata (268)
Imagery (462)
Mapping (936)
Mobile (357)
Python (93)
Security (34)
Services (699)
Story Maps (228)
Web (512)

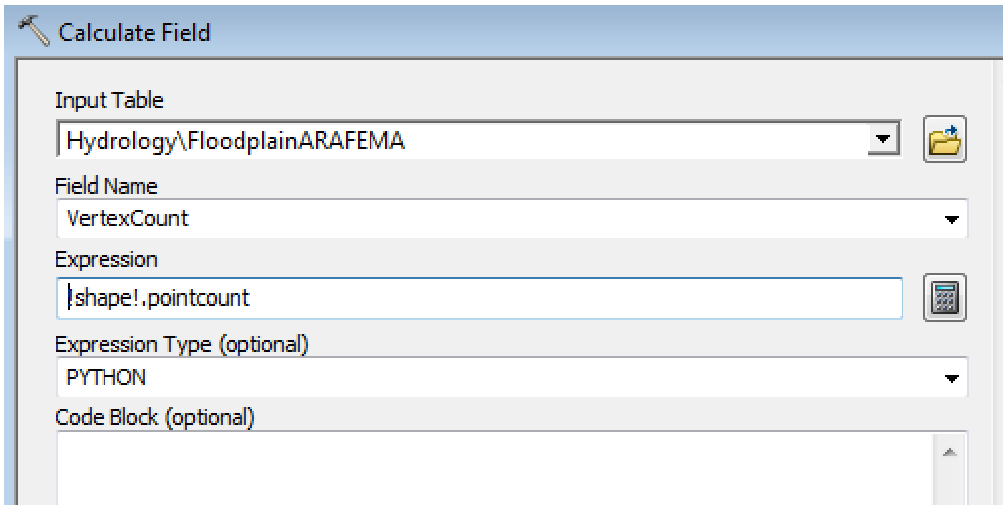
Industry Communities

App Developers (325)
Community Maps (363)
Defense (177)
Electric & Gas (142)
Hydro (255)
Local Government (511)
Location Analytics (491)
National Government (287)
Oceans & Maritime (195)
Public Safety (225)
State Government (290)
Telecommunications (132)
Transportation (155)

100%

## Cut Godzilla polygons to decrease number of vertices
## Explode large multipart polygons as you see them
## Check geometry to be sure there aren't any errors

This procedure creates "Godzilla" polygons with excessive numbers of vertices. Calculate the number of vertices:



Input Table: Hydrology\FloodplainARAFEMA

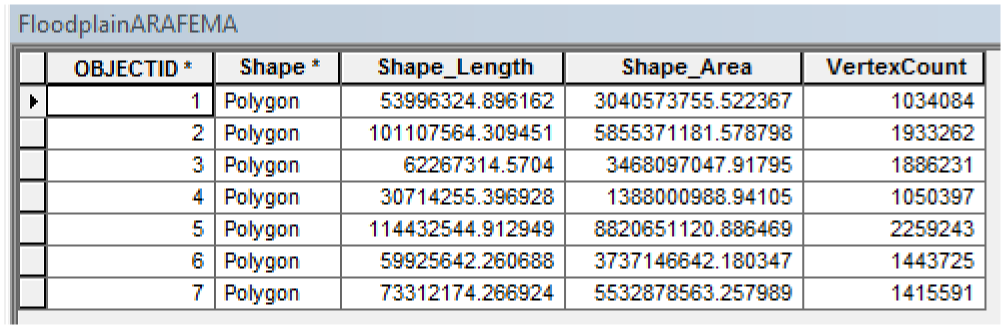
Field Name: VertexCount

Expression: !shape!.pointcount

Expression Type (optional): PYTHON

Code Block (optional):

The result shows that the 7 polygons have over 1 million vertices each.

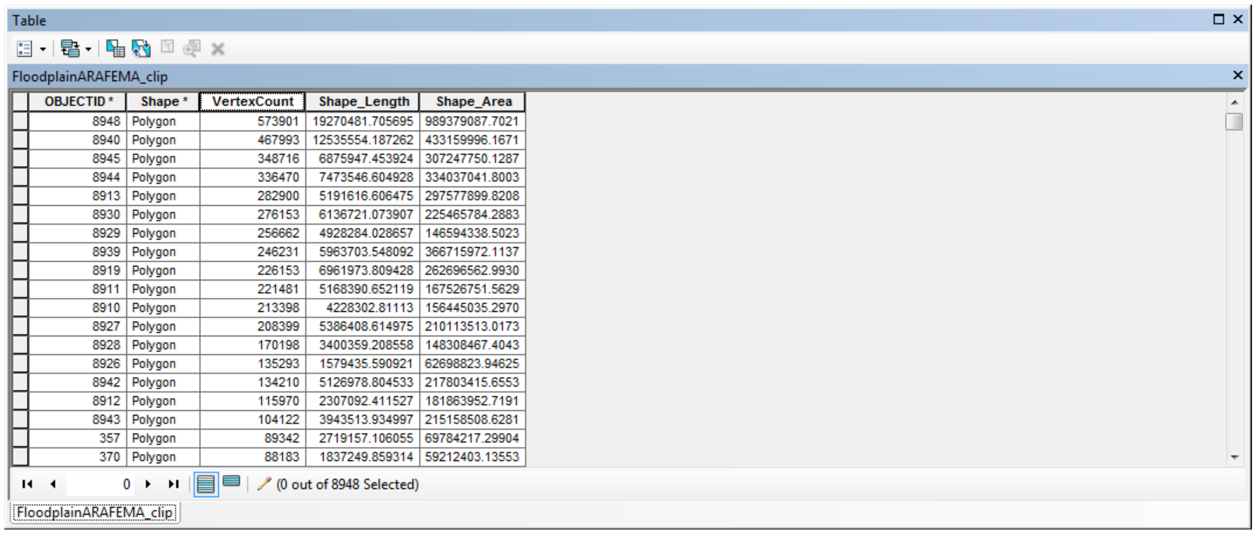


OBJECTID *	Shape *	Shape_Length	Shape_Area	VertexCount
1	Polygon	53996324.896162	3040573755.522367	1034084
2	Polygon	101107564.309451	5855371181.578798	1933262
3	Polygon	62267314.5704	3468097047.91795	1886231
4	Polygon	30714255.396928	1388000988.94105	1050397
5	Polygon	114432544.912949	8820651120.886469	2259243
6	Polygon	59925642.260688	3737146642.180347	1443725
7	Polygon	73312174.266924	5532878563.257989	1415591

We want to reduce the number of vertices, and it is okay to greatly increase the number of polygons. Use the Advanced Editing menu. Select the first polygon and Explode Multipart Feature. Repeat for all the polygons with excessive numbers of vertices. A reasonable target is no more than 200,000 vertices, or to run on machines with less computing power, aim for 50,000 max.

There are now have 38,273 polygons.

Re-calculate the Vertex Count and sort descending to see how large the remaining polygons are.
There are still 8 godzilla polygons.
Select the largest polygon and display it. In this case, much of the polygon is outside the state boundary.
Clip to the state boundary.
Re-calculate the Vertex Count. Clipping to the state boundary reduced the number of vertices and polygons, but we still have 13 polygons that are large.



Table

FloodplainARAFEMA_clip

OBJECTID *	Shape *	VertexCount	Shape_Length	Shape_Area
8948	Polygon	573901	19270481.705695	989379087.7021
8940	Polygon	467993	12535554.187262	433159996.1671
8945	Polygon	348716	6875947.453924	307247750.1287
8944	Polygon	336470	7473546.604928	334037041.8003
8913	Polygon	282900	5191616.606475	297577899.8208
8930	Polygon	276153	6136721.073907	225465784.2883
8929	Polygon	256662	4928284.028657	146594338.5023
8939	Polygon	246231	5963703.548092	366715972.1137
8919	Polygon	226153	6961973.809428	262696562.9930
8911	Polygon	221481	5168390.652119	167526751.5629
8910	Polygon	213398	4228302.81113	156445035.2970
8927	Polygon	208399	5386408.614975	210113513.0173
8928	Polygon	170198	3400359.208558	148308467.4043
8926	Polygon	135293	1579435.590921	62698823.94625
8942	Polygon	134210	5126978.804533	217803415.6553
8912	Polygon	115970	2307092.411527	181863952.7191
8943	Polygon	104122	3943513.934997	215158508.6281
357	Polygon	89342	2719157.106055	69784217.29904
370	Polygon	88183	1837249.859314	59212403.13553

0 (0 out of 8948 Selected)

FloodplainARAFEMA_clip

Explode Multi-part Features again for polygons with more than 100,000 vertices.
Re-calculate the Vertex Count. 9 polygons still have counts > 200,000.
Select the largest polygon and zoom to it. Find good places to cut it (subwatersheds along major rivers require only a single cut) and cut it into smaller chunks. Repeat for all the other large polygons.
Re-calculate the Vertex Count to verify that no godzillas are left.

5.5.19 TMDL: Wetland is in a Watershed with a TMDL Plan Layer

Version date: 16 March 2016

Strategy: 3/16/2016 EAB
GIS method: drafted 3/16/2016 EAB
Python code: not needed
Final review by EAB: 3/16/2016

Purpose:

Water Quality Function, Value to Society aspect, WQPlan factor
This procedure creates a layer showing the watersheds with a TMDL plan.

Description:

TMDL (Total Maximum Daily Load). A TMDL exists for the drainage in which the wetland is found (2 points). A Total Maximum Daily Load (TMDL) plan is a plan of action used to clean up streams that are not meeting water quality standards. The TMDL program is part of the Watershed Branch of the WVDEP. TMDLs have been completed for 32 watersheds in West Virginia, listed at: http://www.dep.wv.gov/wwe/watershed/tmdl/Pages/default.aspx

Update schedule:

Update every 5 years or with major releases of new TMDL plans.

Source Data:

- M:\wr\WTRSHD_BRANCH\TMDL FINAL GIS DATA\TMDL Subsheds (SWS) alphabetical order.lyr

Method:

Create layer showing watersheds with a TMDL plan

Open TMDL Subsheds layers. Export the first watershed to a new feature class "TMDL" and load data from all subsequent watersheds into this polygon feature class.

Output Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: TMDL

5.5.20 Watershed Plan: Watershed Plan Layer

Version date: 24 January 2018

GIS method: completed 2/12/2016 MCA & EAB
Python code: not needed
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function / Value to Society / WQPlan

Update Schedule:

Every 5 years

Description:

Wetland is within a watershed or local water quality planning area, outside the state TMDL plans. Local and watershed planning efforts are important in maintaining existing water quality. These wetlands provide a value to society at the local level that needs to be replaced if they are impacted.

Watershed plans are assembled on WVDEP's website.
Pollutants Field Data Definition: B - Bacteria, M - Metals, P - pH, S - Sediment, NA - None

Source Data:

Watershed plans located on WVDEP's website:

- http://www.dep.wv.gov/WWE/Programs/nonptsource/WBP/Pages/WPs.aspx.

Output Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb (WatershedPlan)

Method:

Identify/assemble local watershed plans or other local plans that address water quality. Digitize boundaries of plan as needed.

Note that fields referencing the plan location and plan name should be added to this layer.

5.5.21 Wetland Birds: Wetland Breeding Bird Occupancy Layer

Version date: 20 February 2017

Strategy: 2/20/2017 EAB
GIS method: 2/20/2017 EAB
Final review by EAB: 2/20/2017

Wetland Birds: WV Birds that benefit from wetland protection

Layer created from WV Breeding Bird Atlas data on 2/17/2017 by Elizabeth Byers, Senior Wetland Scientist at WVDEP, in consultation with Rich Bailey, Ornithologist at WVDNR.

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity) / WetldBird

Update schedule:

This layer does not need to updated until the next reiteration of the state Breeding Bird Atlas, which is on an approximately 20-year cycle, i.e. circa 2038.

Description:

Rationale: Landscapes with high occupancy and density of wetland-dependent breeding birds provide high biodiversity opportunity (maintenance and dispersal of native species richness, rare species, and natural communities).

Distributions of the following species were rolled up into a "wetland bird heat map" for WV. Full weight is given to wetland-dependent birds, and half-weight to non-wetland-dependent species that benefit from wetland protection as part of their habitat requirements. Species with no data were not included. GIS method is described below.

**Block Occupancy Model* (preceded by *)

+Raw Block Data Only

Wetland-dependent

- **+ABDU *Anas rubripes* American Black Duck**
- ***ALFL *Empidonax alnorum* Alder Flycatcher**
- **+AMBI *Botaurus lentiginosus* American Bittern**
- ***AMWO *Scolopax minor* American Woodcock**
- **COMO *Gallinula chloropus* Common Moorhen** (not detected during atlas)
- ***COYE *Geothlypis trichas* Common Yellowthroat** (not SGCN but declining)
- ***GBHE *Ardea herodias* Great Blue Heron**
- ***GRHE *Butorides virescens* Green Heron**
- **+HOME *Lophodytes cucullatus* Hooded Merganser**
- **KIRA *Rallus elegans* King Rail** (considered extirpated)

- **+LEBI *Ixobrychus exilis* Least Bittern**
- **MAWR *Cistothorus palustris* Marsh Wren** (not detected during atlas)
- ***NOWA *Seiurus noveboracensis* Northern Waterthrush**
- **+OSFL *Contopus cooperi* Olive-sided Flycatcher**
- **+PROW *Protonotaria citrea* Prothonotary Warbler**
- **RUBL *Euphagus carolinus* Rusty Blackbird** (not a breeding bird in WV)
- **SEWR *Cistothorus platensis* Sedge Wren**
- **+SORA *Porzana carolina* Sora**
- ***SWWA *Limnothlypis swainsonii* Swainson's Warbler**
- **+VIRA *Rallus limicola* Virginia Rail**
- ***WIFL *Empidonax traillii* Willow Flycatcher** (not SGCN but declining)
- **+WISN *Gallinago delicata* Wilson's Snipe**
- **YBFL *Empidonax flaviventris* Yellow-bellied Flycatcher** (not detected during atlas)
- ***YEWA Setophaga *petechia* Yellow Warbler** (not SGCN but declining)

<u>Not wetland-dependent, but benefit from wetland protection as part of their habitat requirements</u>

- *BAEA *Haliaeetus leucocephalus* Bald Eagle
- *BANS Bank Swallow *Riparia riparia*
- *BBCU Black-billed Cuckoo
- *BTBW Black-throated Blue Warbler
- *BWWA Blue-winged Warbler
- *CAWA *Cardellina canadensis* Canada Warbler
- +CCSP *Spizella pallida* Clay-colored Sparrow
- *CLSW Cliff Swallow
- *COME *Mergus merganser* Common Merganser
- *EAME *Sturnella magna* Eastern Meadowlark
- *EWPW *Antrostomus vociferus* Eastern Whip-poor-will
- *GWWA Golden-winged Warbler
- +LEOW *Asio otus* Long-eared Owl
- *LOWA *Seiurus motacilla* Louisiana Waterthrush
- +NAWA *Vermivora ruficapilla* Nashville Warbler
- +NOGO *Accipiter gentilis* Northern Goshawk
- +NOHA *Circus cyaneus* Northern Harrier
- +NSWO *Aegolius acadicus* Northern Saw-whet Owl
- +OSPR *Pandion haliaetus* Osprey
- +PBGR *Podilymbus podiceps* Pied-billed Grebe
- *RHWO Red-headed Woodpecker
- SEOW *Asio flammeus* Short-Eared Owl (winter/migration only)
- +SPSA *Actitis macularius* Spotted Sandpiper
- +UPSA *Bartramia longicauda* Upland Sandpiper
- *YBSA *Sphyrapicus varius* Yellow-bellied Sapsucker

Source Data:

Raw data from the WVDNR Breeding bird atlas, including block occupancy maps and distirbution maps by species

- \Breeding Bird Atlas\WVBBA2 Occupancy Maps.gdb
    - Feature Class: BlockOccupancy
- \Breeding Bird Atlas\Distribution_Maps.gdb
    - Feature Class: Blocks_2653_BBA2_PriorityNonpriority

Ouput Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb
Feature Class: WetlandBirds_WetBirdColumnOnly

Method:

Block Occupancy Models:
Scores are presented as occupancy probabilities from 0 to 1
Create new fields (float) for:

- WetDepSum: sum of occupancy scores for wetland dependent birds
- NDepSum: sum of occupancy scores for non-wetland dependent birds
- WetOccAvg: sum of the averaged scores, with full weight to WetDepSum and half-weight to NDepSum

WetDepSum = [ALFL]+ [AMWO]+ [COYE]+ [GBHE]+ [GRHE]+ [NOWA]+ [SWWA]+ [WIFL]+ [YEWA]
Count = 9
Weight = 1

NDepSum = [BAEA]+ [BANS]+ [BBCU]+ [BTBW]+ [BWWA]+ [CAWA]+ [CLSW]+ [COME]+ [EAME]+ [EWPW]+ [GWWA]+ [LOWA]+ [RHWO]+ [YBSA]
Count = 14
Weight = 0.5

WetOccAvg = WetDepSum/9 + NDepSum/(14*0.5)
Calculating the sum rather than the average of the weighted scores allows the heavier-weighted wetland-dependent birds to retain more of their value, with the non-wetland dependent birds added as an increment. This may not be the most elegant solution but it does appear to preserve the overall patterns of distribution pretty well.

Raw Block Data:
Scores are codes (see below) that must be converted to occupancy scores.

Codes:

0	Block note surveyed	11	Not reported in priority block
1	Not reported	12	Also detected (other datasets)
2	Also detected (other datasets)	13	Observed in priority block
3	Observed	14	Possible in priority block
4	Possible	15	Probable in priority block
5	Probable	16	Confirmed in priority block
6	Confirmed		

Create new field for each species and populate with approximate occupancy scores based on original codes.

Select values 2,3,12,13 and populate new species field with 1
    e.g., Select "ABDU_2" IN (2,3,12,13) and field calculate “ABDU” = 1
Select value 4,14. New species field= 2
Select value 5,15. new species field = 3
Select value 6,16. New species field = 4

Create fields to store the summations and average values:

- Wetland: average of occupancy scores for wetland dependent birds
- WetPartial: weighted (0.5) average of occupancy scores for non-wetland dependent birds
- WetSum: sum of the two averages

Sum values in “Wetland”, then divide by (4 x 9 spp) so that confirmed occupants have a score of “1”, and others less.
“Wetland” = ([ABDU] + [AMBI] + [HOME] + [LEBI] + [OSFL] + [PROW] + [SORA] + [VIRA] + [WISN])/(4*9)

For non-wetland-dependent birds, do the same calculations, but the final summation will be divided by (8 x 10 spp) for half the weight of the wetland birds.
"WetPartial"= ([CCSP]+ [LEOW]+ [NAWA]+ [NOGO]+ [NOHA]+ [NSWO]+ [OSPR]+ [PBGR]+ [SPSA]+ [UPSA])/(8*10)

These scores are then added together.
“WetSum” = [Wetland] + [WetPartial]
Calculating the sum rather than the average of the weighted scores allows the heavier-weighted wetland-dependent birds to retain more of their value, with the non-wetland dependent birds added as an increment. This may not be the most elegant solution but it does appear to preserve the overall patterns of distribution pretty well.

<u>Combine the two source layers</u>
Join the two source layers based on the Block_ID. Add the Raw Block “WetSum” to the Occupancy Model “WetOccAvg” for a total block occupancy.
Adding rather than averaging gives about twice as much weight to the block-only species (since there are fewer species) in the few blocks where they occur. It also minimizes downgrading the blocks that were not surveyed, since the occupancy models retain their weight in those blocks, i.e. they are not averaged with zero. This is not a perfect solution but it does a decent job of retaining the overall pattern of wetland bird occupancy, without much complexity in the calculation.
“WetBird” = [WetOccAvg] + [WetSum]

<u>Set thresholds for display</u>
    Very High Occupancy: atlas block in upper 10% of values (WetBird > 0.493)

High Occupancy: atlas block in upper 10-50% of values (WetBird > 0.408)
Moderate Occupancy: atlas block in upper 50-75% of values (WetBird > 0.354)
Low Occupancy: atlas block in lower 50% of values (WetBird =< 0.354)

5.6 ArcGIS Methods for Calculating Metrics

5.6.1 AllResults: Results of All GIS Metrics

Version date: 2 April 2018

Purpose:

Store results for each run of the wetland functional assessment tool, including identifiers and metrics.

Description:

<u>Rationale</u>: Storing the results in a single feature class allows for easy archival, sharing, report generation, and analysis. GIS results are exported to a table for import into the WVWRAM MS-Access database.

<u>Strategy</u>: Copy the values for the metrics below into a single feature class. This can be done by a Spatial Join, a Join on WUKey, or in ArcCatalog through Load Data.

Source Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction_[date_time].gdb\

Output:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\Runs\WetlandFunction_[date_time].gdb

- Feature Class: WU_AllResults

Method:

## STEP 1: Create feature class to store the results
## Note that WUKey is included and will be an important linking field

Open WetlandFunction_[date_time].gdb\WU_[date] and Data / Export Data (all features)
Output feature class: WU_AllResults1

## STEP 2: Add the identifier fields (link to original input polygons)
## Copy the values for the fields below to the AllResults feature class. This can be done using a Spatial Join, ## Join on WUKey, or in ArcCatalog through Load Data.

SiteCode (if available in original file)
WetlandName (required)
SurveyDate (required)

## STEP 3: Add scoring fields
## Link to WU_Function and add metrics, with the four roll-up scores listed first
## Copy the values for the fields below to the AllResults feature class. This can be done using a Spatial Join, ## Join on WUKey, or in ArcCatalog through Load Data.

Function
RegFunction
Condition
DNRLandAcq
BRank
FAFunction
FAOpportun
FAPotential
FASociety
HCondition
HFuncNoBR
HFunction
HOpportun
HPotential
HSociety
WQFunction
WQOpportun
WQPotential
WQSociety

## STEP 4: Add metrics
## Link to feature classes in WetlandFunction_[date_time].gdb\ WU_[fieldname]
## Copy the values for the metrics below into a single feature class. This can be done using a Spatial Join, ## Join on WUKey, or in ArcCatalog through Load Data.

AquaAbund	Fisheries	Karst
BRankHUC	FloodArea	LandEco
BufferContig	FloodIn	LandHydro
BufferLand	Floodplain	LandInteg
BufferPerim	Floodway	LandPos
ChemTime	Headwater	LandResil
Clay	HInvest	LowSlope
ClayOrganic	Histosol	MarlPEM
ConnectFL (in Connect)	HUC12WQ	Microtopo
ConsFocus	HUse	Organic
Depressions	HydIntact (in HydroH)	OwnerAccess
Discharges	HydroH	PublicUse
Dist50mFQ (in VegFQ)	HydSW (in HydroH)	RoadRail
Disturb50m	ImpairedIn	Runoff
DisturbWshd	ImpairedOut	Runoff50m
EconRisk	IrrEdge	RunoffWshd

SeaPondRat (in SeasonPond)	SWOutflow2	VegWoody
	VegAll	VegWoody4
SeasonPond	VegByLP	VegWoodyFor
Slope	VegFA	VegWQ
SlopeWshd	VegFQ	WaterSupply
SoilH	VegH	WetldBird
SoilIntact (in Disturb50m)	VegHorInt (in Microtopo)	WFlowPath
SoilOrgCalc	VegPerUng	WQPlan
StreamEdge	VegPerUng1	WQUse
StrucPatch	VegPerUng4	WshdPos
SWOutflow	VegVerStr	WshdUniq

## END OF PROCEDURE

5.6.2 AquaAbund: Aquatic Area Abundance

Version date: 3 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/3/2017 EAB; results verified 10/3/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Landscape Opportunity (LandHydro: Landscape Hydrologic Connectivity)

Description:

Maximum 2 points
Rationale: "Definition: The aquatic area abundance of an Assessment Area is assessed in terms of its spatial association with other areas of aquatic resources, such as other wetlands, lakes, streams, etc. It is assumed that wetlands close to each other have a greater potential to interact ecologically and hydrologically, and that such interactions are generally beneficial."

"Rationale: …The functional capacity of a wetland is determined not only by its intrinsic properties, but by its relationship to other habitats across the landscape. Several researchers have concluded that landscape-scale variables are often better predictors of stream and wetland integrity than localized variables (Roth et al. 1996; Scott et al. 2002). Wetlands that are close together without hydrological or ecological barriers between them are better able to provide refuge and alternative habitat patches for metapopulations of wildlife, to support transient or migratory wildlife species, and to function as sources of colonists for primary or secondary succession of newly created or restored wetlands. In general, good landscape connectivity exists only where neighboring wetlands or other habitats do not have intervening obstructions that could inhibit the movements of wildlife." (CWMW 2013)

GIS Method (no field method):
Calculate % cover of wetlands, ponds, lakes, and rivers in a 1 km buffer. Use the Enhanced National Wetlands Inventory as the source data for wetlands, ponds, lakes, and rivers. Note that the Enhanced NWI includes pond features copied over from the NHD. The Enhanced NWI includes wide (polygonal) rivers but it does not include line features for small streams. This part of the metric is size-neutral.

Calculate total length stream reaches in a 1 km buffer. Note that this calculation gives an advantage to large wetlands, which have more area in the 1 km buffer. This advantage is realistic because large wetlands also serve as aquatic resource areas to themselves.

Merge and set appropriate thresholds based on distribution of values. Assign points as follows:

2 points: at least 5% cover of NWI aquatic resources OR at least 8 km of NHD stream reaches within 1 kilometer buffer

1 point: 1-5% cover of NWI aquatic resources OR 6-8 km of NHD stream reaches within 1 kilometer buffer

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - FeatureClass: Buffer1km
        - Field: Buf1kArea
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
    - Feature Dataset: CONUS_WVWetlandsProj
        - Feature Class: EnhWVWetland
- M:\ \LayerFiles\arcsde_backup.gdb
    - Feature Dataset: wr
        - Feature Class: NHD_with_stream_codes

Method:

## STEP 1 Calculate percent of 1 km buffer that contains aquatic features from NWI
## Note that this method is very similar to STEP 2 of BufferPerim

## Create feature class to store intermediate results

R-click WU_20150514 and select Data / Export / All Features
Output feature class: WetlandFunction.gdb / WU_AquaAbund1

## Intersect the 1 km buffers and the Enhanced National Wetlands Inventory.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: Buffer1km
EnhWVWetland
Output feature class: Buffer1kAqua
Join attributes: ALL
Output type: INPUT

## Dissolve aquatic portion of wetland buffer by WUKey

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer1kAqua
Output Feature Class: Buffer1kAqua_diss
Dissolve Fields: WUKey
Statistics Fields: Buf1kArea (Statistic Type = First)
Check box “Create multipart features” (default)

## Add field and calculate ratio of aquatic area to total buffer area.

Open attribute table of Buffer1kAqua_diss
Add field “Aqua1kRat” (float)

Field calculate Aqua1kRat = [Shape_Area] / [FIRST_Buf1kArea]

## Join ratio of aquatic buffer to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_AquaAbund1
Input Join Field: WUKey
Join Table: Buffer1kAqua_diss
Output Join Field: WUKey

## Export joined data

R-click WU_AquaAbund1 and select Data / Export Data
Output feature class: WU_AquaAbund2

## Remove Join

R-click WU_ AquaAbund1 and select Joins and Relates / Remove All Joins

## Set value of Aqua1kRat to zero for null intersections

Open attribute table of WU_ AquaAbund2
SELECT * FROM WU_ AquaAbund2 WHERE: "Aqua1kRat" IS NULL
Field Calculate Aqua1kRat = 0

## STEP 2 Calculate total length of NHD streams in 1 km buffer

## Intersect the 1 km buffers and the NHD stream reaches
## Note that this is a very large intersection and takes about 30 minutes

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: Buffer1km
NHD_with_stream_codes
Output feature class: Buffer1kStrm
Join attributes: ALL
Output type: INPUT

## Add field to store Stream Reach Length

Open attribute table of Buffer1kStrm
Add field StrmLength (float)
Field Calculate StrmLength = [Shape_Length]

## Delete unnecessary fields in this large table to reduce processing time in the Dissolve below

## Note that this takes a long time

ArcToolbox > Data Management Tools > Fields > Delete Field
Input table: Buffer1kStrm
Drop Field: Check the boxes for all fields EXCEPT WUKey and StrmLength

## Dissolve stream reach lengths in the wetland buffer by WUKey

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer1kStrm
Output Feature Class: Buffer1kStrm_diss
Dissolve Fields: WUKey
Statistics Fields: StrmLength (Statistic Type = Sum)
Check box "Create multipart features" (default)

## Join sum of stream lengths to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_AquaAbund2
Input Join Field: WUKey (first occurrence in list of attributes – the 2nd occurrence has nulls)
Join Table: Buffer1kStrm_diss
Output Join Field: WUKey

## Export joined data

R-click WU_AquaAbund2 and select Data / Export Data / All Features
Output feature class: WU_AquaAbund

## Remove Join

R-click WU_ AquaAbund2 and select Joins and Relates / Remove All Joins

## Set value of FINALCODE to zero for null intersections

Open attribute table of WU_ AquaAbund
SELECT * FROM WU_ AquaAbund2 WHERE: "SUM_StrmLength" IS NULL
Field Calculate SUM_StrmLength = 0

## STEP 3: Assign points

## Add field AquaAbund, set initial value to zero

Open the attribute table to WU_AquaAbund
Add field "AquaAbund" (short integer) to attribute table
Field Calculate "AquaAbund" = 0

## Assign points
## StrmLength one SD below mean = 0, within 1 SD of mean = 1, >1 SD above mean = 2 points

SELECT * FROM WU_AquaAbund WHERE: "Aqua1kRat" > 0.01 OR "SUM_StrmLength" > 6000
Field Calculate AquaAbund = 1

SELECT * FROM WU_AquaAbund WHERE: "Aqua1kRat" > 0.05 OR "SUM_StrmLength" > 8000
Field Calculate AquaAbund = 2

5.6.3 Brank: Site Biodiversity Rank of Wetland

Version date: 8 November 2017

Strategy: 3/16/2017 EAB
GIS method: 10/19/2017 EAB; results verified 10/20/2017 EAB; revisions to GIS method on 11/5/2017 highlighted in yellow; results re-verified 11/5/2017
Python code: 10/30/2017, revised and packaged for stand-alone use 11/7/2017 YH
Final review by EAB: 11/7/2017

Notes on using the stand-alone packaged arcpy script to calculate Site Biodiversity Rank for all sites (not limited to wetlands)

1. Two layers are required:
    a. Input polygons to be assessed (e.g. "ManagedLands")
        i. Add field (long integer) name "WUKey" and populate it with a unique ID.
    b. Source data file of ranked element occurrences named "BRankInput".
        i. See the instructions for BRankInput to create that file. "BRankInput" can be screened to include only wetland species or other subsets of the full biodiversity database; however, the source data used in the python tool must be named "BRankInput".
2. Open the script BRank_packaged \ BRank.pyt \ Site Biodiversity Rank
3. Specify the location of the input polygon layer.
4. Specify the location of "BRankInput".
5. Specify the target geodatabase for output.
6. Run the tool.
7. The output file will be named "WU_BRank". Re-name it something meaningful like "BRank_ManagedLands_20171108". The code will not automatically overwrite the output file, so you must rename it or delete it before running the code again.

****
******

Purpose:

Input to/Override of Habitat Function

Description:

<u>Rationale</u>: Wetland supports rare, threatened, or endangered species or high quality natural communities. Certain wetlands are recognized as being exemplary in the state for their outstanding habitat value. Note that if a rare species or natural wetland community is documented at the site, or is found during rapid field assessment, then the wetland should be assigned a B-rank. Surveying for rare species and natural wetland communities is not a required part of the rapid assessment protocol; however, if a B-rank has already been documented by the state, then it will be included in the GIS assessment.

<u>GIS Strategy:</u>

The presence, quality, and concentration of rare species and natural vegetation communities is the basis for Site Biodiversity Ranks. B-rank methodology is implemented by WVDNR's Natural Heritage Program. Scoring for B-ranked wetlands is as follows:

B1	Outstanding Global Biodiversity Significance
B2	High Global Biodiversity Significance
B3	Global Biodiversity Significance
B4	Outstanding State Biodiversity Significance
B5	State Biodiversity Significance
B6	Local Biodiversity Significance

Screen out elements that would not occur in a wetland, i.e., would not include wetlands in any part of their habitat, foraging areas, or life cycle. This has already been done when creating the BRankInput file.

Site Biodiversity Ranks (B1-B6) are assigned at the highest rank for which any one of the criteria of that rank are met.

B1 – Outstanding global biodiversity significance
• Only known occurrence of an element rangewide, or
• A-ranked occurrence of a G1 element (or at least C-ranked if best available in state), or
• Concentration of A or B-ranked occurrences of G1 or G2 elements (4 or more)

B2 – High global biodiversity significance
• B or C-ranked occurrence of a G1 element, or
• A or B-ranked occurrence of a G2 element, or
• One of the most outstanding (e.g., among 5 best) occurrences rangewide (at least A or B-ranked) of a G3 element, or
• Concentration of C-ranked G2 and A- or B-ranked G3 element occurrences (4 or more)

B3 – Global biodiversity significance
• D-ranked occurrence of a G1 element (if best available in state), or
• C-ranked occurrence of a G2 element, or
• A or B- ranked occurrence of a G3 element, or
• A or B- ranked occurrence of a disjunct S1 element, or
• One of the most outstanding (e.g., among 5 best) occurrences of a G4 or G5 community (at least A or B-ranked) in an ecoregion. Of these, the 1 or 2 best in an ecoregion could be elevated to B2 status based on consultation with the Natural Heritage ecologist; or
• Concentration (4+) of C-ranked G3 and A- or B-ranked S1 element occurrences.

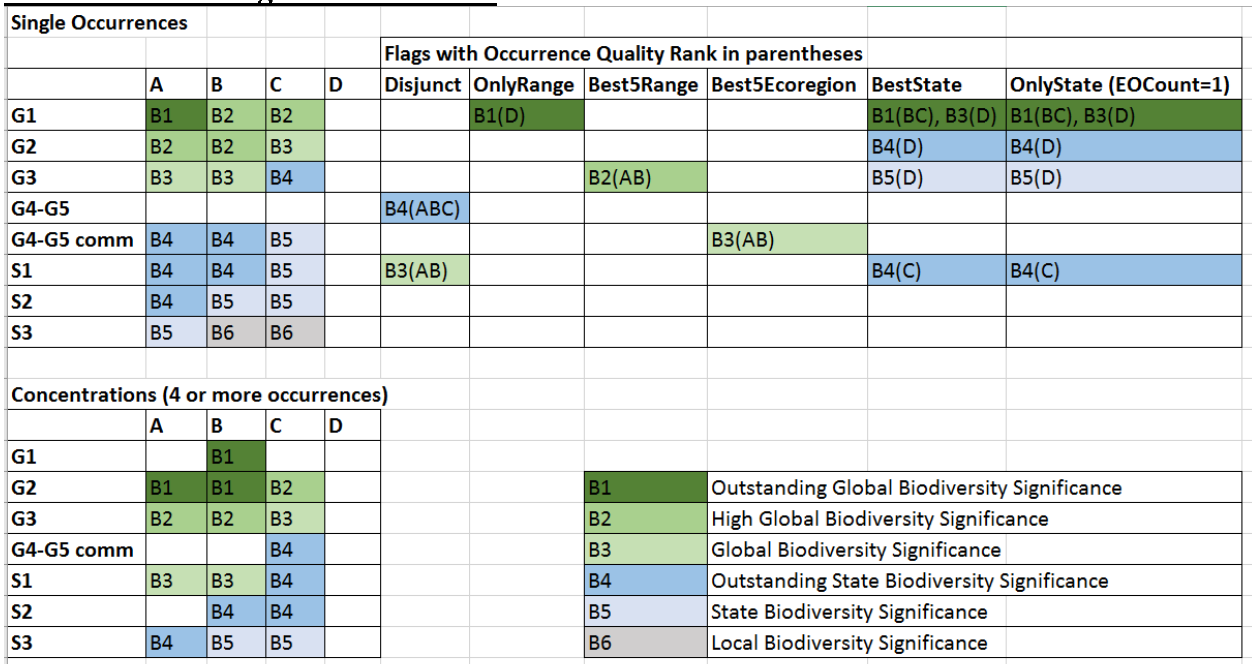
B4 – Outstanding state biodiversity significance
• D-ranked occurrence of a G2 element (if best available in state), or
• C-ranked occurrence of a G3 element, or
• A or B-ranked occurrence of S1 element (or at least C-ranked if best available in state), or
• A-ranked occurrence of an S2 element, or
• At least C-ranked occurrence of a disjunct G4 or G5 element, or
• A or B-ranked occurrence of a G4 or G5 community, or

• Concentration (4+) of C-ranked S1, B- or C-ranked S2, and A-ranked S3 element occurrences and C-ranked G4 or G5 communities.

B5 – State biodiversity significance
• D-ranked occurrence of G3 element (if best available in state), or
• C-ranked occurrence of S1 element, or
• B or C-ranked occurrence of S2 element, or
• A-ranked occurrence of an S3 element, or
• C-ranked occurrence of a G4 or G5 community, or
• Concentration (4+) of B- or C-ranked S3 element occurrences.

B6 – Local biodiversity significance
• B or C-ranked occurrence of S3 element

Chart summarizing B-Rank criteria



Single Occurrences

					Flags with Occurrence Quality Rank in parentheses					
	A	B	C	D	Disjunct	OnlyRange	Best5Range	Best5Ecoregion	BestState	OnlyState (EOCount=1)
G1	B1	B2	B2			B1(D)			B1(BC), B3(D)	B1(BC), B3(D)
G2	B2	B2	B3						B4(D)	B4(D)
G3	B3	B3	B4				B2(AB)		B5(D)	B5(D)
G4-G5					B4(ABC)					
G4-G5 comm	B4	B4	B5					B3(AB)		
S1	B4	B4	B5		B3(AB)				B4(C)	B4(C)
S2	B4	B5	B5							
S3	B5	B6	B6							

Concentrations (4 or more occurrences)

	A	B	C	D
G1		B1		
G2	B1	B1	B2	
G3	B2	B2	B3	
G4-G5 comm			B4	
S1	B3	B3	B4	
S2		B4	B4	
S3	B4	B5	B5	

B1	Outstanding Global Biodiversity Significance
B2	High Global Biodiversity Significance
B3	Global Biodiversity Significance
B4	Outstanding State Biodiversity Significance
B5	State Biodiversity Significance
B6	Local Biodiversity Significance

Notes on geographic extent of site and element occurrence *(not incorporated – simple intersections are used)*
• If the site includes >50% of the area of the element occurrence, or if the element occurrence is a point contained within the site, then the site is awarded full credit for that element.
• If the site includes <50% of the area of the element occurrence, then it is ranked at one level lower, e.g. if the full element occurrence would confer a site rank of B3, then the partial occurrence qualifies the site for a rank of B4. This decision can be manually overridden by a Natural Heritage biologist, if there is evidence that the site supports the element at population numbers and quality of habitat meriting the higher rank.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gd
    - Feature class: BRankInput

Method:

## STEP 1: Create feature class and add fields to store BRank values; screen input

R-click WU_20150514 and select Data / Export Data / All Features
Output feature class: WetlandFunction: WU_BRank

## Add fields to store partial B-Rank based on single elements, concentrations, and final BRank

Open attribute table of WU_BRank
Add field BSing (text, length = 10)
Add field BConc (text, length = 10)
Add field BRank (text, length = 10)

## STEP 2: Assign Site Biodiversity Ranks based on single element occurrences
## Rank B6 selection

Open attribute table of BRankInput
SELECT * FROM BRankInput WHERE:

```
        "Srank" = 'S3' AND "OQrank" IN ( 'B', 'C')
```

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features
Spatial selection method for target layer feature(s): intersect the source layer feature

R-click BSing and Field Calculate BSing = "B6"

## Rank B5 selection

Open attribute table of BRankInput
SELECT * FROM BRankInput WHERE:

```
        ("Grank" = 'G3' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" = 'BestState') ) OR
        ("Srank" = 'S1' AND "OQrank" = 'C') OR
        ("Srank" = 'S2' AND "OQrank" IN ('B','C') ) OR
        ("Srank" = 'S3' AND "OQrank" = 'A') OR
        ("SpecComm" = 'C' AND "Grank" IN ('G4', 'G5') AND "OQrank" = 'C' )
```

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features
Spatial selection method for target layer feature(s): intersect the source layer feature

R-click BSing and Field Calculate BSing = "B5"

## Rank B4 selection

Open attribute table of BRankInput

```
SELECT * FROM BRankInput WHERE:
        ("Grank" = 'G2' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" = 'BestState') )
        OR
        ("Grank" = 'G3' AND "OQrank" = 'C') OR
        ("Srank" = 'S1' AND "OQrank" IN ('A', 'B')) OR
        ("Srank" = 'S1' AND "OQrank" = 'C' AND ("EO_Count" = 1 OR "Flag" = 'BestState') )
        OR
        ("Srank" = 'S2' AND "OQrank" = 'A') OR
        ("Grank" IN ('G4', 'G5') AND "OQrank" IN ('A', 'B', 'C') AND "Flag" = 'Disjunct') OR
        ("SpecComm" = 'C' AND "Grank" IN ('G4', 'G5') AND "OQrank" IN ('A', 'B'))
```

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features
Spatial selection method for target layer feature(s): intersect the source layer feature

R-click BSing and Field Calculate BSing = "B4"

## Rank B3 selection

Open attribute table of BRankInput

```
SELECT * FROM BRankInput WHERE:
        ("Grank" = 'G1' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" = 'BestState') )
        OR
        ("Grank" = 'G2' AND "OQrank" = 'C') OR
        ("Grank" = 'G3' AND "OQrank" IN ('A', 'B')) OR
        ("Srank" = 'S1' AND "OQrank" IN ('A', 'B') AND "Flag" = 'Disjunct') OR
        (("SpecComm" = 'C' AND "Grank" IN ('G4', 'G5')) AND "OQrank" IN ('A', 'B') AND
        "Flag" = 'Best5Ecoregion')
```

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features: (2783 features selected)
Spatial selection method for target layer feature(s): intersect the source layer feature
(178 out of 43124 selected)

R-click BSing and Field Calculate BSing = "B3"

## Rank B2 selection

Open attribute table of BRankInput
SELECT * FROM BRankInput WHERE:
    ("Grank" = 'G1' AND "OQrank" IN ('B', 'C')) OR
    ("Grank" = 'G2' AND "OQrank" IN ('A','B')) OR
    ("Grank" = 'G3' AND "OQrank" IN ('A', 'B') AND "Flag" = 'Best5Range')

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features
Spatial selection method for target layer feature(s): intersect the source layer feature

R-click BSing and Field Calculate BSing = "B2"

## Rank B1 selection

Open attribute table of BRankInput
SELECT * FROM BRankInput WHERE:
    ("Flag" = 'OnlyRange') OR
    ("Grank" = 'G1' AND "OQrank" = 'A') OR
    ("Grank" = 'G1' AND "OQrank" IN ('B', 'C') AND ("EO_Count" = 1 OR "Flag" =
    'BestState'))

Select by Location
Selection method: Select features from
Target layer: WU_BRank
Source layer: BRankInput
Use selected features
Spatial selection method for target layer feature(s): intersect the source layer feature

R-click BSing and Field Calculate BSing = "B1"

Clear all selections

## STEP 3: Join Wetland Units with BRankInput

ArcToolbox / Spatial Join
Target features: WU_20150514
Join features: BRankInput
Output feature class: WU_BRank_join
Join Operation: JOIN_ONE_TO_MANY
Keep all target features
Field Map of Join Features: retain basically all (Shape_Length1 and Shape_Area1 are not needed)
Match option: Intersect

## STEP 4: Find the highest occurrence quality rank for each element in each Wetland Unit
## Set null values of RandID to zero to allow summarize function to work

SELECT * FROM WU_BRank_join WHERE: "RandID" IS NULL
R-click RandID and Field Calculate RandID = 0

## Add field to hold integer transformation of OQRank

Add new field: OQrankInt (short integer)
SELECT * FROM WU_BRank_join WHERE: "OQrank" IS NULL
Field Calculate OQrankInt = 0

SELECT * FROM WU_BRank_join WHERE: "OQrank" = 'D'
Field Calculate OQrankInt = 1

SELECT * FROM WU_BRank_join WHERE: "OQrank" = 'C'
Field Calculate OQrankInt = 2

SELECT * FROM WU_BRank_join WHERE: "OQrank" = 'B'
Field Calculate OQrankInt = 3

SELECT * FROM WU_BRank_join WHERE: "OQrank" = 'A'
Field Calculate OQrankInt = 4

Clear all selections

## Find the highest OQrank for each WUKey |RandID pair

Open attribute table WU_BRank_join
Add new field: Concat (text, length = 20)
Field Calculate Concat = [WUKey]&" | "& [RandID]

R-click the Concat field and Select “Summarize”
Select a field to summarize: Concat
Choose one or more summary statistics to be included in the output table:
    WUKey: Minimum
    RandID: Minimum
    Srank: First
    Grank: First
    SpecComm: First
    OQrankInt: Maximum
Specify output table: WU_BRank_summ

## STEP 5: Assign Site Biodiversity Rank based on concentrations of elements

## Rank B5 selection (Rank B6 does not have a criterion for concentrations)
## Select B- or C-ranked occurrences of S3 elements

Open table WU_BRank_summ
SELECT * FROM WU_BRank_summ WHERE:
    "First_Srank" = 'S3' AND "Max_OQrankInt" IN (2,3)

Open table WU_BRank_summ
R-click “Minimum_WUKey” and select Summarize
Select a field to summarize: Min_WUKey
Choose one or more summary statistics to be included in the output table: none
Specify output table: WU_BRank_summB5

## Join table back to WU_BRank

R-click WU_BRank and select Joins and Relates / Joins
What do you want to join to this layer: Join attributes from a table
Choose the field in this layer that the join will be based on: WUKey
Choose the table to join to this layer: WU_BRank_summB5
Choose the field in the table to base the join on: Min_WUKey
Keep all records

## Select records with 4 or more elements and assign value to BConc

Open attribute table of WU_BRank with Join displayed
SELECT * FROM WU_BRank_WU_BRank_summB5 WHERE:
    WU_BRank_summB5.Cnt_Min_WUKey > 3
Field calculate BConc = “B5”

## Remove Join

R-click WU_BRank and select Joins and Relates / Remove Joins / Remove All Joins

## Rank B4 selection
## Select C-ranked S1, B- or C-ranked S2, and A-ranked S3 element occurrences and
## C-ranked G4 or G5 communities

Open table WU_BRank_summ
SELECT * FROM WU_BRank_summ WHERE:
```
        ( "First_Srank" = 'S1' AND "Max_OQrankInt" = 2) OR
        ( "First_Srank" = 'S2' AND "Max_OQrankInt" IN (2,3)) OR
        ( "First_Srank" = 'S3' AND "Max_OQrankInt" = 4) OR
        ("First_SpecComm" = 'C' AND  "First_Grank" IN ('G4', 'G5') AND "Max_OQrankInt" =
        2)
```

Open table WU_BRank_summ
R-click "Minimum_WUKey" and select Summarize
Select a field to summarize: Min_WUKey
Choose one or more summary statistics to be included in the output table: none
Specify output table: WU_BRank_summB4
Summarize on the selected records only

## Join table back to WU_BRank

R-click WU_BRank and select Joins and Relates / Joins
What do you want to join to this layer: Join attributes from a table
Choose the field in this layer that the join will be based on: WUKey
Choose the table to join to this layer: WU_BRank_summB4
Choose the field in the table to base the join on: Min_WUKey
Keep all records

## Select records with 4 or more elements and assign value to BConc

Open attribute table of WU_BRank with Join displayed
SELECT * FROM WU_BRank_WU_BRank_summB4 WHERE:
```
        WU_BRank_summB4.Cnt_Min_WUKey > 3
```
Field calculate BConc = "B4"

## Remove Join

R-click WU_BRank and select Joins and Relates / Remove Joins / Remove All Joins

## Rank B3 selection
## Select C-ranked occurrences of G3 elements and A- or B-ranked occurrences of S1 elements

Open table WU_BRank_summ
SELECT * FROM WU_BRank_summ WHERE:
    ("First_Grank" = 'G3' AND "Max_OQrankInt" = 2) OR
    ("First_Srank" = 'S1' AND "Max_OQrankInt" IN (3,4))

Open table WU_BRank_summ
R-click "Minimum_WUKey" and select Summarize
Select a field to summarize: Min_WUKey
Choose one or more summary statistics to be included in the output table: none
Specify output table: WU_BRank_summB3
Summarize on the selected records only

## Join table back to WU_BRank

R-click WU_BRank and select Joins and Relates / Joins
What do you want to join to this layer: Join attributes from a table
Choose the field in this layer that the join will be based on: WUKey
Choose the table to join to this layer: WU_BRank_summB3
Choose the field in the table to base the join on: Min_WUKey
Keep all records

## Select records with 4 or more elements and assign value to BConc

Open attribute table of WU_BRank with Join displayed
SELECT * FROM WU_BRank_WU_BRank_summB3 WHERE:
    WU_BRank_summB3.Cnt_Min_WUKey > 3
Field calculate BConc = "B3"

## Remove Join

R-click WU_BRank and select Joins and Relates / Remove Joins / Remove All Joins

## Rank B2 selection
## Select C-ranked occurrences of G2 elements and A- or B-ranked occurrences of G3 elements

Open table WU_BRank_summ
SELECT * FROM WU_BRank_summ WHERE:
    ("First_Grank" = 'G2' AND "Max_OQrankInt" = 2) OR
    ("First_Grank" = 'G3' AND "Max_OQrankInt" IN (3,4))

Open table WU_BRank_summ
R-click "Minimum_WUKey" and select Summarize
Select a field to summarize: Min_WUKey
Choose one or more summary statistics to be included in the output table: none

Specify output table: WU_BRank_summB2
Summarize on the selected records only

## Join table back to WU_BRank

R-click WU_BRank and select Joins and Relates / Joins
What do you want to join to this layer: Join attributes from a table
Choose the field in this layer that the join will be based on: WUKey
Choose the table to join to this layer: WU_BRank_summB2
Choose the field in the table to base the join on: Min_WUKey
Keep all records

## Select records with 4 or more elements and assign value to BConc

Open attribute table of WU_BRank with Join displayed
```
SELECT * FROM WU_BRank_WU_BRank_summB2 WHERE:
        WU_BRank_summB2.Cnt_Min_WUKey > 3
```
Field calculate BConc = "B2"

## Remove Join

R-click WU_BRank and select Joins and Relates / Remove Joins / Remove All Joins

## Rank B1 selection
## Select B-ranked occurrences of G1 elements and A- or B-ranked occurrences of G2 elements

Open table WU_BRank_summ
```
SELECT * FROM WU_BRank_summ WHERE:
        ("First_Grank" = 'G1' AND "Max_OQrankInt" = 3) OR
        ("First_Grank" = 'G2' AND "Max_OQrankInt" IN (3,4))
```

Open table WU_BRank_summ
R-click “Minimum_WUKey” and select Summarize
Select a field to summarize: Min_WUKey
Choose one or more summary statistics to be included in the output table: none
Specify output table: WU_BRank_summB1
Summarize on the selected records only

## Join table back to WU_BRank

R-click WU_BRank and select Joins and Relates / Joins
What do you want to join to this layer: Join attributes from a table
Choose the field in this layer that the join will be based on: WUKey
Choose the table to join to this layer: WU_BRank_summB1
Choose the field in the table to base the join on: Min_WUKey

Keep all records

## Select records with 4 or more elements and assign value to BConc

Open attribute table of WU_BRank with Join displayed
SELECT * FROM WU_BRank_WU_BRank_summB1 WHERE:
    WU_BRank_summB1.Cnt_Min_WUKey > 3
Field calculate BConc = "B1"

## Remove Join

R-click WU_BRank and select Joins and Relates / Remove Joins / Remove All Joins

## STEP 6: Calculate final Site Biodiversity Rank

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B6'
R-click BRank and Field Calculate BRank = 'B6'

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B5' OR "BConc" = 'B5'
R-click BRank and Field Calculate BRank = 'B5'

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B4' OR "BConc" = 'B4'
R-click BRank and Field Calculate BRank = 'B4'

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B3' OR "BConc" = 'B3'
R-click BRank and Field Calculate BRank = 'B3'

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B2' OR "BConc" = 'B2'
R-click BRank and Field Calculate BRank = 'B2'

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BSing" = 'B1' OR "BConc" = 'B1'
R-click BRank and Field Calculate BRank = 'B1'

## Populate the “Null” BRank records with “none”

Open attribute table of WU_BRank
SELECT * FROM WU_BRank WHERE: "BRank" IS NULL

R-click BRank and Field Calculate BRank = "none"
Clear all selections

5.6.4 BRankHUC: Watershed Biodiversity Rank

Version date: 6 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/6/2017 EAB; results verified 10/6/2017 EAB
Python code: 10/10/2017 YH
Final review by EAB: 10/10/2017

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)

Description:

Maximum 4 points
Rationale: Landscape provides high biodiversity opportunity (maintenance and dispersal of native species richness, rare species, and natural communities)

GIS Method (no field method):
Select wetlands that intersect ranked HUC12 watersheds.

4 points: wetland intersects a B1-ranked 12-digit watershed (watershed provides habitat for good populations of critically globally imperiled species or natural communities (___out of 764 watersheds)

3 points: wetland intersects a B2-ranked 12-digit watershed (watershed provides habitat for good populations of globally imperiled species or natural communities) (___<172 out of 764 watersheds)

2 points: wetland intersects a B3-ranked 12-digit watershed (watershed provides habitat for good populations of globally vulnerable or disjunct state critically imperiled species or natural habitats) (376 out of 764 watersheds)

1 point: wetland intersects a B4- or B5-ranked 12-digit watershed (watershed provides habitat for good populations of state imperiled species or natural habitats) (376 out of 764 watersheds)

0 points: none of the above criteria are met

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83.shp

Method:

## Create feature class to store results for BRankHUC and set initial value to zero

R-click WU_20150514 and select Data / Export / All Features

Output feature class: WetlandFunction.gdb / WU_BRankHUC

Open attribute table of WU_BRankHUC
Add field “BRankHUC” (short integer)
Field calculate BRankHUC = 0

## Select B4- or B5 ranked watersheds

SELECT * FROM watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
 WHERE: "Brank" = 'B4' OR "Brank" = 'B5'

## Select Wetland Units that intersect B3- or B4-ranked watershed

Select by Location
Selection method: select features from
Target layer(s): WU_BRankHUC
Source layer: watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 1 point

Open attribute table of WU_BRankHUC
Field Calculate BRankHUC = 1

## Select B3- ~~or B4~~-ranked watersheds

SELECT * FROM watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
 WHERE: "Brank" = 'B3'

## Select Wetland Units that intersect B3- or B4-ranked watershed

Select by Location
Selection method: select features from
Target layer(s): WU_BRankHUC
Source layer: watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 2 points

Open attribute table of WU_BRankHUC
Field Calculate BRankHUC = 2

## Select B2-ranked watersheds

SELECT * FROM watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
WHERE: "Brank" = 'B2'

## Select Wetland Units that intersect B2-ranked watershed

Select by Location
Selection method: select features from
Target layer(s): WU_BRankHUC
Source layer: watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 3 points

Open attribute table of WU_BRankHUC
Field Calculate BRankHUC = 3

## Select B1-ranked watersheds

SELECT * FROM watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
WHERE: "Brank" = 'B1'

## Select Wetland Units that intersect B1-ranked watershed

Select by Location
Selection method: select features from
Target layer(s): WU_BRankHUC
Source layer: watershedBiodiversityRankHUC_WVDNR_12Nov2014_utm83
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 4 points and clear selections

Open attribute table of WU_BRankHUC
Field Calculate BRankHUC = 4

Clear all selections

5.6.5 BufferContig: Contiguous 300m Buffer for Wildlife

Version date: 29 Septmeber 2017

Strategy: 3/16/2017 EAB
GIS method: 9/29/2017 EAB; results verified 9/29/2017 EAB
Python code: 10/3/2017 YH
Final review by EAB: 10/3/2017

Purpose:

Input to Habitat / Opportunity / BufferLand (Buffer and Landscape Integrity)

Description:

Maximum 2 points
Rationale: "A wider buffer has a greater capacity to serve as habitat for wetland edge-dependent species, to reduce the inputs of non-point source contaminants, to control erosion, and to generally protect the wetland from human activities.
The condition of a buffer is assessed according to the extent and quality of its vegetation cover, the overall condition of its substrate, and the amount of human visitation.
The condition or composition of the buffer, in addition to its width and extent around a wetland, determines the overall capacity of the buffer to perform its critical functions." CWMW 2013

Strategy: *Note that the integrity of the 300m wetland buffer must be verified in the field, since GIS data may be out-of-date. This GIS metric is overwritten by field assessment.*
Calculate percent of contiguous 300m buffer not overlapping DisturbLand. Erase the DisturbedLand from the 300m buffer, then select the remaining buffer polygons that are contiguous (approximated as "share a line segment with") with their corresponding Wetland Unit.
Assign points as follows:

- 2 points: >90% of buffer is undisturbed AND is contiguous with Wetland Unit
- 1 point: 60-90% of buffer is undisturbed AND is contiguous with Wetland Unit
- 0 points: neither of the above criteria are met

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
  - Feature Class: WU_20150514
  - Feature Class: Buffer300m
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: DisturbedLand

## Note that the procedure in this box should be performed when Wetland Units are first created
## Therefore, please do not include this procedure in the code for this metric

## Create 300m buffer around Wetland Unit

ArcToolbox / Proximity / Buffer
Input features WU_20150514
Output feature class: WetlandUnits.gdb\Buffer300m
Distance [value or field]: Linear Unit: 300 Meters
Side Type: OUTSIDE_ONLY
Dissolve Type: NONE

## Add field to store buffer area

Open attribute table of Buffer300m
In attribute table of Buffer300m, add field “Buf300Area” (float)
Field calculate Buf300Area = Shape_Area

Method:

## Erase the portions of the 300m buffer that overlap DisturbedLand
ArcToolbox / Analysis Tools / Overlay / Erase
Input features: Buffer300m
Erase features: DisturbedLand
Output feature class: Buffer300mUndist

## The Erase tool produces multipart polygons. Change these to singlepart polygons.

ArcToolbox / Data Management / Multipart to Singlepart
Input: Buffer300mUndist
Output: Buffer300mUndist_sing

## Add field to Buffer300mUndist_sing to store area of contiguous singlepart polygons

Open attribute field of Buffer300mUndist_sing
Add field “ContigSingArea” (float)

## Select undisturbed buffer polygons that share a line segment with Wetland Units.
## This is a proxy for being contiguous to a Wetland Unit. When buffers of nearby wetland buffers
## do not overlap, it works perfectly. In the case of overlapping buffers, it is very unlikely that an
## overlapping buffer polygon from a nearby wetland would have the exact geometry to share a line
## segment with any Wetland Unit. Spot checks of the statewide output confirm this.

Select by location
Selection method: select features from
Target layers(s): Buffer300mUndist_sing

Source layer: WU_20150514
Spatial selection method for target layer feature(s): share a line segment with the source layer feature

## Calculate area of contiguous singlepart polygons

Open attribute field of Buffer300mUndist_sing
Field Calculate "ContigSingArea" = [Shape_Area]

Clear selections
SELECT * FROM Buffer300mUndist_sing WHERE: "ContigSingArea" IS NULL
Field Calculate "ContigSingArea" = 0

## Dissolve undisturbed portion of wetland buffer by WUKey

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer300mUndist_sing
Output Feature Class: Buffer300mUndist_diss
Dissolve Fields: WUKey
Statistics Fields: Buf300Area (Statistic Type = First)
ContigSingArea (Statistic Type = Sum)
Check box "Create multipart features" (default)

## Add field and calculate ratio of contiguous undisturbed area to total buffer area.

Open attribute table of Buffer300mUndist_diss
Add field "ContigUndRat" (float)
Field calculate ContigUndRat = [SUM_ContigSingArea] / [FIRST_Buf300Area]

## Create feature class to store intermediate results for BufferContig

R-click "WU_20150514" and Export Data
Export: All features
Output feature class: WetlandFunction.gdb\WU_BufferContig1

## Join ratio of contiguous undisturbed buffer to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_BufferContig1
Input Join Field: WUKey
Join Table: Buffer300mUndist_diss
Output Join Field: WUKey

## Export joined data

R-click WU_BufferContig1 and select Data / Export Data

Output feature class: WU_BufferContig

## Remove Join

R-click WU_BufferContig1 and select Joins and Relates / Remove All Joins

## Set value of ContigUndRat to zero for null intersections

Open attribute table of WU_BufferContig
SELECT * FROM WU_BufferContig WHERE: "ContigUndRat" IS NULL
Field Calculate ContigUndRat = 0

## Add field BufferContig, set initial value to zero

Open the attribute table to WU_BufferContig
Add field “BufferContig” (short integer) to attribute table
Field Calculate “BufferContig” = 0

## Assign points

SELECT * FROM WU_BufferContig WHERE: "ContigUndRat" > 0.6
Field Calculate BufferContig = 1

SELECT * FROM WU_BufferContig WHERE: "ContigUndRat" > 0.9
Field Calculate BufferContig = 2

5.6.6 BufferLand: Buffer and Landscape Integrity

Version date: 2 October 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 10/2/2017 EAB; results verified 10/2/2017 EAB
Python coding: 10/3/2017 YH
Final review by EAB: 10/3/2017

Purpose:

Input to Habitat & Ecological Integrity / Landscape Opportunity
Max 7 points
Rationale: see rationale for each of the three component metrics.
Strategy: Sum the points for BufferPerim, BufferContig, and LandInteg.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_BufferPerim
        - Field: BufferPerim
    - Feature Class: WU_BufferContig
        - Field: BufferContig
    - Feature Class: WU_LandInteg
        - Field: LandInteg

Method:

## Spatial join to merge BufferPerim and BufferContig into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_BufferPerim
Join Feature: WU_BufferContig
Output Feature Class: WU_BufferLand1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
BufferPerim
BufferContig
Match option: CONTAINS

## Spatial join to merge LandInteg

ArcToolbox / Analysis Tools / Overlay / Spatial Join

Target Feature: WU_BufferLand1
Join Feature: WU_LandInteg
Output Feature Class: WU_BufferLand
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
BufferPerim
BufferContig
LandInteg
Match option: CONTAINS

## Add BufferLand field to Wetland Units and set initial point value to zero.

Open attribute table of WU_BufferLand
Add field “BufferLand” (short integer)
R-click BufferLand and Field Calculate BufferLand = 0

## Sum the points for BufferPerim, BufferContig, LandInteg

R-click BufferLand and Field Calculate “BufferLand” = [BufferPerim] + [BufferContig] + [LandInteg]

5.6.7 BufferPerim: Wetland Perimeter with Natural Buffer

Version date: 24 January 2018

Strategy: 3/16/2017 EAB
GIS method: 9/28/2017 EAB; results verified 9/29/2017 EAB. 1/24/2018 EAB: Method updated to include mapped trails. Note that selection numbers are outdated as a result of this addition.
Python code: 10/3/2017 YH
Final review by EAB: 10/3/2017

Purpose:

Input to Habitat / Opportunity / BufferLand (Buffer and Landscape Integrity)

Description:

Maximum 2 points
Rationale: An intact perimeter, even with a narrow natural buffer, offers protection to the wetland habitat. The ability of buffers to protect a wetland increases with buffer extent along the wetland perimeter. For some kinds of stress, such as predation by feral pets or disruption of plant communities by cattle, small breaks in buffers may cause significant degradation of a wetland. However, for many stressors including trails and small unpaved roadways, small breaks in buffers probably do not significantly disrupt the buffer functions (CWMW 2013).

Strategy: *Note that the integrity of the wetland perimeter is best observed in the field. This GIS metric is overwritten by field assessment.*
Calculate the percent of the Wetland Unit perimeter (10m buffer around Wetland Unit) that intersects DisturbedLand. Identify the Wetland Units that are within 10 meters of linear disturbances (road, rail). Tiger line files have route type (RTTYP) codes as follows:

C = County (=DEP primary, local, other)
I = Interstate (=DEP interstate)
M = Common Name (=DEP interstate, primary, local, other)
O = Other (=DEP primary, local, other)
S = State recognized (=DEP primary, local, other)
U = U.S. (=DEP interstate, primary, local, other)

The DEP basemap has more accurate road attributes than the raw Tiger line files. Also the DEP railways layer appears to be more accurately and completely mapped than the tiger_2013 rail layer. We do not have good GIS coverage of pipelines or transmission lines. Assign points as follows:

- 2 points: 10m buffer does not intersect DisturbedLand AND Wetland Unit is not within 10 meters of trails, roads or railways.
- Wetland Unit is within 10 meters of trail, Local Road, Other Road or Trail, but is not within 10 meters of railways or larger roads OR 10m buffer intersects trace-25% of DisturbedLand
- 0 points: Wetland Unit is within 10 meters of Interstate Highway, Primary Road, or Railway OR 10m buffer intersects >25% of DisturbedLand

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\trails_Sep_27_2017_webmercator.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\basemap\tiger_2013\WV_Transportation_UTM.gdb
    - Feature Class: Interstates
    - Feature Class: Primary_Roads
    - Feature Class: Local_Roads
    - Feature Class: Other Roads & Trails
- M:\LayerFiles\arcsde_backup.gdb
    - Feature Dataset: basemap_cultural_non_replica
    - Feature Class: SDE_railway_tiger
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: DisturbedLand

## Create 10m buffer around Wetland Unit
## Note that this procedure should be performed when Wetland Units are first created
## Therefore, please do not include this procedure in the code for this metric

ArcToolbox / Proximity / Buffer
Input features WU_20150514
Output feature class: WetlandUnits.gdb\Buffer10m
Distance [value or field]: Linear Unit: 10 Meters
Side Type: OUTSIDE_ONLY
Dissolve Type: NONE

## Add field to store buffer area

Open attribute table of Buffer10m
In attribute table of Buffer10m, add field "Buf10Area" (float)
Field calculate Buf10Area = Shape_Area

Method:

## STEP 1 Roads and Railways
## Create feature class to store intermediate results for BufferPerim

R-click **"**WU_20150514" and Export Data
Export: All features
Output feature class: WetlandFunction.gdb\WU_BufferPerim1

## Add field to store road and rail type

Open attribute table of WU_BufferPerim1

Add field RoadRailType (text, 10 characters)

## Select Wetland Units that intersect or are within 10 meters of mapped trails

Select by Location
Selection method: Select features from: WU_BufferPerim1
Source layer: trails_Sep_27_2017_webmercator.shp
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters
(____ Wetland Units selected)

R-click RoadRailType and Field Calculate RoadRailType: "Trail"

## Select Wetland Units that intersect or are within 10 meters of other roads & trails

Select by Location
Selection method: Select features from: WU_BufferPerim1
Source layer: other roads & trails
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters

R-click RoadRailType and Field Calculate RoadRailType: "Other"

## Select Wetland Units that intersect or are within 10 meters of local roads

Select by Location
Selection method: Select features from: WU_BufferPerim1
Source layer: local roads
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters

R-click RoadRailType and Field Calculate RoadRailType: "Local"

## Select Wetland Units that intersect or are within 10 meters of railways

Select by Location
Selection method: Select features from: WU_BufferPerim1
Source layer: railways
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters

R-click RoadRailType and Field Calculate RoadRailType: "Rail"

## Select Wetland Units that intersect or are within 10 meters of primary roads

Select by Location

Selection method: Select features from: WU_BufferPerim1
Source layer: primary roads
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters

R-click RoadRailType and Field Calculate RoadRailType: "Primary"

## Select Wetland Units that intersect or are within 10 meters of interstate highways

Select by Location
Selection method: Select features from: WU_BufferPerim1
Source layer: interstates
Spatial selection method for target layer feature(s): Intersect the source layer feature
Apply a search distance: 10 meters

R-click RoadRailType and Field Calculate RoadRailType: "Interstate"

## STEP 2 DisturbedLand
## Note that much of this step is the same as the method for Disturb50m

## Intersect the 10m buffers and the disturbed land uses.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: Buffer10m
DisturbedLand
Output feature class: Buffer10mDist
Join attributes: ALL
Output type: INPUT

## Dissolve disturbed portion of wetland buffer by WUKey

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer10mDist
Output Feature Class: Buffer10mDist_diss
Dissolve Fields: WUKey
Statistics Fields: Buf10Area (Statistic Type = First)
Check box “Create multipart features” (default)

## Add field and calculate ratio of disturbed area to total buffer area.

Open attribute table of Buffer10mDist_diss
Add field “Dist10mRat” (float)
Field calculate Dist10mRat = [Shape_Area] / [FIRST_Buf10Area]

## Join ratio of disturbed buffer to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_BufferPerim1
Input Join Field: WUKey
Join Table: Buffer10mDist_diss
Output Join Field: WUKey

## Export joined data

R-click WU_BufferPerim1 and select Data / Export Data
Output feature class: WU_BufferPerim

## Remove Join

R-click WU_BufferPerim1 and select Joins and Relates / Remove All Joins

## Set value of Dist10mRat to zero for null intersections

Open attribute table of WU_BufferPerim
SELECT * FROM WU_BufferPerim WHERE: "Dist10mRat" IS NULL
Field Calculate Dist10mRat = 0

## STEP 3
## Assign points

## Add field BufferPerim, set initial value to zero

Open the attribute table to WU_BufferPerim
Add field “BufferPerim” (short integer) to attribute table
Field Calculate “BufferPerim” = 0

## Assign points

SELECT * FROM WU_BufferPerim WHERE: "Dist10mRat" = 0 AND "RoadRailType" IS NULL
Field Calculate BufferPerim = 2

SELECT * FROM WU_BufferPerim WHERE: "Dist10mRat" > 0 OR "RoadRailType" IN ('Trail', 'Local', 'Other')
Field Calculate BufferPerim = 1

SELECT * FROM WU_BufferPerim WHERE: "Dist10mRat" > 0.25 OR "RoadRailType" IN ('Rail', 'Primary', 'Interstate')
Field Calculate BufferPerim = 0

5.6.8 Chem Time: Time and space for Chemical Reactions to Occur

i.e., Seasonal Ponding, Slope, and Wetland Upland Interface

Version date: 14 March 2016

Strategy: completed 2/27/2016 EAB
GIS method: completed & verified 3/14/2016 EAB
Python code: started & completed 3/21/2016 MCA
Final review by EAB: 3/21/2016

Purpose:

Input to Water Quality
Maximum 3 points; groundwater wetlands only

Description:

Rationale: The area of the wetland that is seasonally ponded is an important characteristic in understanding how well it will remove nutrients, specifically nitrogen. The highest levels of nitrogen transformation occur in areas of the wetland that undergo a cyclic change between oxic (oxygen present) and anoxic (oxygen absent) conditions. The oxic regime (oxygen present) is needed so certain types of bacteria will change nitrogen that is in the form of ammonium ion (NH4+) to nitrate, and the anoxic regime is needed for denitrification (changing nitrate to nitrogen gas) (Mitsch and Gosselink 1993). The area that is seasonally ponded is used as an indicator of the area in the wetland that undergoes this seasonal cycling. The soils are oxygenated when dry but become anoxic during the time they are flooded.
Water velocity increases with increasing slope. This decreases the retention time of surface water in the wetland and the potential for retaining sediments and associated toxic pollutants. The potential for sediment deposition and retention of toxics by burial decreases as the slope increases (Adamus et al. 1991).

Summary of strategy: Select wetlands that are not in a floodplain and have seasonal ponding. Then, filter the points so that wetlands with slope >5% do not receive any points, and wetlands with slopes 2-5% get a maximum of 2 points.
Finally, wetlands can gain 1 additional point (regardless of slope) if they have a highly irregular upland/wetland boundary.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
    - Feature Class: EnhWVWetland

Input Variables:

- Floodplain (location in floodplain)
- SeasonPond (seasonal ponding)

- Slope (median percent slope)
- IrrEdge (irregular upland/wetland boundary edge)

Method:

## Create feature class to store ChemTime factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_ChemTime

## Spatial join to add input variables to attribute table

Spatial join (contains) to add the following to the WU_ChemTime: Floodplain, SeasonPond, Slope, IrrEdge

## Add field ChemTime to Wetland Units attribute table and set initial value to SeasonPond.

Add field ChemTime (ShortInteger) to WU_ChemTime attribute table.
R-click ChemTime and Field Calculate “ChemTime” = [SeasonPond]

## Filter Seasonal Ponding (SeasonPond) points based on Slope and Floodplain

SELECT * FROM WU_ChemTime WHERE: "SLOPE" > 5 OR "Floodplain" = 'Y'
R-click ChemTime and Field Calculate “ChemTime” = 0

SELECT * FROM WU_ChemTime WHERE: "SLOPE" > 2 AND "SLOPE" < 6 AND "ChemTime" > 2
R-click ChemTime and Field Calculate “ChemTime” = 2

## Add point for irregular edge (IrrEdge)

SELECT * FROM WU_ChemTime WHERE: "IrrEdge" = 1 AND "Floodplain" = 'N' AND "ChemTime" < 3
R-click ChemTime and Field Calculate “ChemTime” = [ChemTime] + 1

5.6.9 Clay: Clay near Surface

Version date: 4 October 2016

Strategy: completed 2/27/2016 EAB
GIS method: completed & verified 3/3/2016 EAB
Python code: Started 3/2/2016 MCA, Completed 3/3/2016 MCA
Final verification of result by EAB: 3/3/2016; 10/4/2016 EAB revised SsurgoClay layer to include clay < 8 cm instead of < 6 cm (better reading of Hruby 2012). This does not affect Python coding.

Purpose:

Input to Water Quality / Clay and Organic Soils Factor
Y/N

Description:

Clay near surface. Select Wetland Units that intersect with clay in the top 8 cm of the soil profile in Palustrine Plot or SSURGO datasets.
Note that SSURGO mapping is very uneven, with some counties heavily mapped in A-horizon clay (Wetzel, Tyler, Pleasants, Ritchie, Putnam, Cabell) and others with little or no A-horizon clay mapped (Hardy, Grant, Tucker, Preston, Webster, Roane, Calhoun, Lewis, Doddridge, Nicholas, and much of southwestern WV).

Definitions:

SSURGO soils data from NRCS has multiple non-spatial tables, which have one-to-many relationships with the ssurgo_wv table. We will access the component horizon table (chorizon_all) to extract the clay content, horizon, and top depth of the horizon.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: PalustrinePlotsMarch2015
- M:\basemap\ssurgo\ssurgo.gdb
    - Feature Class: ssurgo_wv
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb
    - Feature Class: SsurgoClay

NOTE: ArcGIS related tables cannot be programmed in Python, so before this procedure is run, the input data layer must be created in ArcGIS, as shown below. Note that this layer should be re-exported as SSURGO is updated by NRCS.

## Open the related one-to-many SSURGO chorizon_all table.

Open attribute table for ssurgo_wv
Click "Related Tables" (second icon from left).
Click "component to surgo: component_all" to open the component_all table. Note that a tab will appear at the bottom of the attribute table showing the tables that are open.

Click “Related Tables” (second icon from left).
Click “component to chorizon: chorizon_all” to open the chorizon_all table.

## Select soils with clay > 27% in the top 8 cm of the soil profile.

With the tab at the bottom of the attribute labelled “chorizon_all” highlighted:
SELECT * FROM chorizon_all WHERE: "claytotal_r" > 27 AND "hzdept_r" < 8 (516 out of 28520 selected)

## Relate this selection to the spatial data in the ssurgo_wv tab.

Click “Related Tables” again to get back to component_all and then back to ssurgo_wv (65525 out of 413438 selected). Note that the last step takes some time while all of the related tables open up.

## Export data to new feature class

R-click ssurgo_wv / Data / Export Data
Export Selected Features
Output feature class: SsurgoExports\SsurgoClay

Input Variables:

None

Method:

## Add field Clay to Wetland Units attribute table and set initial value to “no clay”.

Add field Clay (Text, Length 2) to Wetland Units attribute table.
R-click Clay and Field Calculate “Clay” = ‘N’

## PART 1: PALUSTRINE PLOTS
## Select Palustrine plots with clay near surface.

SELECT * FROM PalustrinePlotsMarch2015 WHERE: ("Soil_Textu" LIKE '%clay%' OR "Profile__1" LIKE '%clay%') AND "Depth_of_o" IN (' ', '0', '1', '2', '3', '4') (230 out of 1702 selected)

## Select Wetland Units that intersect palustrine plots selection.

Selected by location
Selection method: select features from
Target layer: WU_20150514
Source layer: PalustrinePlotsMarch2015

Check “Use selected features”
Spatial selection method: intersect the source feature layer

## Update the value for “Clay” based on Palustrine plots.

Open Wetland Units attribute table.
R-click “Clay” and Field Calculate “Clay” = “Y”
Clear all selections.

## Part 2: SSURGO

## Select Wetland Units that intersect with the SSURGO selection.

Select by Location
Selection method: select features from
Target layer(s): WU_21050514
Source layer: SsurgoClay
Spatial selection method: intersect the source layer feature

## Update the value for “Clay” based on SSURGO.

Open Wetland Units attribute table
R-click “Clay” and Field Calculate “Clay” = “Y”

5.6.10 Clay Organic: Clay and Organic Soils

Version date: 14 March 2016

Strategy: completed 2/27/2016 EAB
GIS method: completed 3/10/2016 EAB; verified 3/14/3016 EAB

Python code: started & completed 4/5/2016 MCA
Final review by EAB: 4/5/2016

Purpose:

Input to Water Quality
Maximum 3 points; groundwater wetlands only

Description:

Must be outside the area of permanent ponding.
Rationale: Clay soils and organic soils are good indicators that a wetland can remove a wide range of pollutants from surface water. The uptake of dissolved phosphorus and toxic compounds through adsorption to soil particles is highest when soils are high in clay or organic content (Mitsch and Gosselink 1993). Denitrification is also high in soils with high organic content (Fisher and Acreman 2004). We only consider the type of soil near the surface because this is where the soil actually has contact with the surface waters carrying the pollutants. This is where most of the chemical and biological reactions occur. We only consider the organic or clay soil horizon in areas that are not permanently ponded.
Summary of strategy: Select wetland units that have clay or organic soils AND are not in a floodplain. Assign points according to the area of seasonal ponding, as follows:

- SeaPondRatio = 70-100% cover: 3 points
- SeaPondRatio = 40-70% cover: 2 points
- SeaPondRatio = 10-40% cover: 1 point
- SeaPondRatio < 10% cover: 0 point

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Clay (clay soil near surface)
- Organic (organic matter near surface)
- SeaPondRatio (ratio of seasonally ponded area to total Wetland Unit area)
- Floodplain (location in floodplain)

Method:

## Create feature class to store ClayOrganic factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_ClayOrganic

## Add ClayOrganic field to Wetland Units and set initial point value to zero.

Open attribute table of WU_ClayOrganic
Add field “ClayOrganic” (short integer)

R-click ClayOrganic and Field Calculate ClayOrganic = 0

## Spatial join to add input variables to attribute table

Spatial join (contains) to add the following to the WU_ClayOrganic: Floodplain, Clay, Organic, SeaPondRatio

## Assign points to Wetland Units not in floodplain with clay or organic soils and seasonal ponding

SELECT * FROM WU_ClayOrganic WHERE: "Floodplain" = 'N' AND ("Clay" = 'Y' OR "Organic" = 'Y') AND "SeaPondRatio" > 0.1

Field Calculate ClayOrganic = 1

SELECT * FROM WU_ClayOrganic WHERE: "Floodplain" = 'N' AND ("Clay" = 'Y' OR "Organic" = 'Y') AND "SeaPondRatio" > 0.5

Field Calculate ClayOrganic = 2

SELECT * FROM WU_ClayOrganic WHERE: "Floodplain" = 'N' AND ("Clay" = 'Y' OR "Organic" = 'Y') AND "SeaPondRatio" > 0.9

Field Calculate ClayOrganic = 3

5.6.11 Connect FL: Connectivity to Historic Floodplain

Version date: 20 November 2016

Strategy: Completed 11/16/2016 EAB
GIS method: completed 11/20/2016 EAB, verified 12/19/2016 EAB

Python coding: Completed 12/22/2016 MCA
Final review by EAB: 12/30/2016

Purpose:

Input to Flood Attenuation / Opportunity
Max 2 points.
Rationale: Wetlands are more likely to receive flood waters if they are well-connected to their historic floodplain (Acreman and Holden 2013).
Summary of strategy: Sum the points for FloodArea and StreamEdge. Reduce to a maximum of 2 points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- FloodArea (2 points)
- StreamEdge (2 points)

Method:

## Spatial joins to add input variables to Wetland Units attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_20150514
Join Feature: WU_FloodArea
Output Feature Class: WU_Connect1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
FloodArea
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Connect1
Join Feature: WU_StreamEdge
Output Feature Class: WU_Connect
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area

FloodArea
StreamEdge
Match option: CONTAINS

## Add ConnectFL field and set initial point value to zero.

Open attribute table of WU_Connect
Add field “ConnectFL” (short integer)
R-click ConnectFL and Field Calculate ConnectFL = 0

## Sum the points for FloodArea and StreamEdge.

Open attribute table of WU_Connect
R-click ConnectFL and Field Calculate “ConnectFL” = [FloodArea] + [StreamEdge]

## Reduce the total points to a maximum of 2.

Open attribute table of WU_Connect
SELECT * FROM WU_Connect WHERE: "ConnectFL" > 2
Field Calculate (selection only) “ConnectFL” = 2

5.6.12 ConsFocus: WVDNR Conservation Focus Areas with Wetland Focus

Version date: 11 December 2018

Strategy: 3/16/2017 EAB

GIS method: 10/4/2017 EAB; results verified 10/4/2017 EAB; Updated to include amphibian and reptile focus area information from DNR on 12/10/2018 EAB; 12/11/2018 method verified by JSH
Python code: 10/9/2017 YH
Final review by EAB: 10/9/2017

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)

Description:

Maximum 2 points
Rationale: WVDNR has identified Conservation Focus Areas (CFAs) as part of the State Wildlife Action Plan. Several of the CFAs have a specific wetland focus. In addition, Partners in Reptile and Amphibian Conservation (PARCA) is working with WVDNR, USFS, USFWS, Dr. Tom Pauley of Marshall U., and others to develop focus areas, some of which have a specific wetland focus. CFAs that include a wetland focus are:

1. Cacapon River and Patterson Creek (wetland odonates, Short Mtn wetland); note that this shares a boundary with much of the PARCA "Eastern Panhandle" focus area for amphibians and reptiles, including wetland species.
2. Central Reservoirs (wetland birds and odonates),
3. High Alleghenies (High Allegheny Wetlands, all taxa groups, largest and most intact wetland complex in WV),
4. Little Kanawha and Middle Island Creek (wetland odonates),
5. Lower Elk River (wetland odonates),
6. Meadow River Wetlands (oak-ash swamps in 2nd largest wetland complex in WV, birds, crayfish, plants),
7. Ohio River Corridor (wetland birds, amphibians, plants, Greenbottom Swamp, Ohio River Islands); note that the southern half of this area shares boundaries with the PARCA "Moth Man" focus area for amphibians and reptiles, including wetland species.
8. Shenandoah Valley (marl wetlands, Virginia Rail, spotted turtle); note that this include the PARCA "Altona" focus area for amphibians and reptiles, including wetland species.
9. Sleepy Creek and Back Creek (wetland turtles, amphibians, plants); note that this shares a boundary with the eastern portion of the PARCA "Eastern Panhandle" focus area for amphibians and reptiles, including wetland species.

PARCAs with at least a partial wetland focus:

1. Snot Otter: includes riverine wetlands
2. Gorges: includes riverine wetlands
3. General Davis: includes riverine wetlands
4. Cranberry: includes riverine wetlands
5. Eastern Panhandle: mostly focused on the 3 major wood turtle streams (Cacapon, Sleepy Creek, and Back Creek) and associated palustrine wetlands and tributaries to those waterways

6. Altona: incorporates the bulk of TNC property and easements along Mill Creek, west of Charles Town, so includes the stream and associated palustrine wetlands.
7. Moth Man: incorporates many of the wetlands, mainly palustrine, serving as habitat for amphibians, including rare species such as small-mouthed salamander, streamside salamander, Jefferson salamander, and Blanchard's cricket frog (if still extant in the state).

PARCAs that do not have a wetland focus are Panther, Wayne, Cow Knob, and Pauley's Plethodon. Pauley's Plethodon includes a large wetland acreage with significant overall biodiversity, but the wetland-affiliated amphibians and reptiles in that PARCA don't happen to be priority species. Pauley's Plethodon has priority species Cheat Mountain salamander, green salamander, and timber rattlesnake.

<u>GIS Method (no field method):</u>
Select wetlands that intersect CFAs and/or PARCAs. Assign points as follows:

2 points: Wetland Unit intersects CFA/PARCA with specific wetland focus
1 point: Wetland Unit intersects any CFA except the "General CFA" or any PARCA
0 points: None of the above criteria are met

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb
    - Feature Class: ConsFocusArea

Method:

## Create feature class to store results for ConsFocus and set initial value to zero

R-click WU_20150514 and select Data / Export / All Features
Output feature class: WetlandFunction.gdb / WU_ConsFocus

Open attribute table of WU_ConsFocus
Add field "ConsFocus" (short integer)
Field calculate ConsFocus = 0

## Select Wetland Units that intersect the selected CFAs

Select by Location
Selection method: select features from
Target layer(s): WU_ConsFocus
Source layer: ConsFocusArea
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 1 point

Open attribute table of WU_ConsFocus
Field Calculate ConsFocus = 1

## Select CFAs with wetland focus

SELECT * FROM ConsFocusArea WHERE: WetlFocus = 'yes'

## Select Wetland Units that intersect CFAs with wetland focus

Select by Location
Selection method: select features from
Target layer(s): WU_ConsFocus
Source layer: ConsFocusArea
Check "Use selected features"
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 2 points

Open attribute table of WU_ConsFocus
Field Calculate ConsFocus = 2

5.6.13 Creating Wetland Units from NWI Polygons

Version date: 29 March 2018

Purpose:

Wetland Units are contiguous, hydrologically connected wetland polygons. They are the basic units used for functional assessment in West Virginia.

Methods:

1. Run the NWI data verification tests on the input polygons and correct any errors.

2. Create a new feature class that removes permanently flooded and unvegetated rivers and lakes, and unvegetated industrial and waste disposal ponds. Leave in riverine and lacustrine polygons that are temporarily flooded, seasonally flooded, or intermittently flooded and which often show vegetation on air photos.
   a. Select by attributes where: "ATTRIBUTE" NOT LIKE 'R%BH' AND "ATTRIBUTE" NOT LIKE 'L%Hh' AND "ATTRIBUTE" NOT LIKE 'L%Hx' AND "ATTRIBUTE" NOT LIKE 'L%s%' AND "ATTRIBUTE" NOT LIKE 'P%r%' AND "ATTRIBUTE" NOT LIKE 'P%K%' AND "ATTRIBUTE" NOT LIKE 'PU%s%' AND "ATTRIBUTE" NOT LIKE 'PR%s%'.
   b. Create a new feature class with the selected data called WUWorkingDS.

3. Dissolve NWI polygons. ArcGIS Geoprocessing menu/Dissolve. Remember to uncheck “Create multipart features (optional)”. We do not want any multipart features.
4. Add unique ID field (Long Integer: WUKey) for Wetland Units
5. Run “Check Geometry” on the final output to check for errors.
6. Create the basic geometries used for assessment of Wetland Units
   a. WUpoint
   b. Buffer10m
   c. Buffer50m
   d. Buffer300m
   e. Buffer1km
   f. DrainageArea

Buffer creation method
## Create 300m buffer around Wetland Unit

ArcToolbox / Proximity / Buffer
Input features WU_20150514
Output feature class: WetlandUnits.gdb\Buffer300m

Distance [value or field]: Linear Unit: 300 Meters
Side Type: OUTSIDE_ONLY
Dissolve Type: NONE

## Add field to store buffer area

Open attribute table of Buffer300m
In attribute table of Buffer300m, add field “Buf300Area” (float)
Field calculate Buf300Area = Shape_Area

5.6.14 CSFunction: Carbon Sequestration

Version date: 3 February 2017

Strategy: completed 4/13/2016 EAB
GIS method: completed 4/13/2016 EAB; verified 4/15/2016 EAB
Python code: Please wait to do this roll-up of the function. Since there are just 4 points in this function, we will combine it with Habitat/Ecological Integrity.

Final review by EAB:

Purpose:

Carbon Sequestration
4 points maximum – all wetlands

Description:

Rationale**:** The intrinsic potential of a wetland to sequester carbon depends on the amount and stability of above-ground biomass (vegetation) and below-ground carbon (organic soils). The carbon sequestration function includes only the potential aspect, since the opportunity and value to society aspects are the same for all wetlands.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the "Potential" aspect. If the total exceeds 4 points, reduce to 4.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Histosol (3 points – all types)
- VegCS (3 points – all types)

Method:

## Spatial join to add input variables to Wetland Units attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Histosol
Join Feature: WU_VegCS
Output Feature Class: WU_CSFunction
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
Histosol
VegCS
Match option: CONTAINS

## Add CSFunction field to Wetland Units and set initial point value to zero.

Open attribute table of WU_CSFunction
Add field "CSFunction" (short integer)
R-click CSFunction and Field Calculate CSFunction = 0

## Sum the factor points

R-click CSFunction and Field Calculate CSFunction = [Histosol] + [VegCS]

## Reduce any excess point scores to the maximum allowed.

Clear all selections.
SELECT * FROM WU_CSFunction WHERE: "CSFunction" > 4
(235 out of 43214 selected)
Field Calculate (selection only) “CSFunction” = 4

5.6.15 Depressions: Surface Depressions

Version date: 22 March 2016

Strategy: Completed 2/27/2016 EAB
GIS method: Drafted 2/27/2016 EAB, verified 3/14/2016 EAB
Python code: Started 3/21/2016, completed 3/22/2016 MCA
Final review by EAB: 3/22/2016

Purpose:

Input to Water Quality
Max 5 points, floodplain wetlands only.

Description:

Rationale: Surface depressions in a wetland that receives overland flow can trap sediments during a flood event. Depressions in riverine wetlands will tend to accumulate sediment and the pollutants associated with sediment (phosphorus and some toxics) because they reduce water velocities (Fennessey and others 1994) when the river floods. Wetlands where a larger part of the total area has depressions are relatively better at removing pollutants associated with sediments than those that have no such depressions. We cannot calculate surface depressions directly with the DEMs we have, so we estimate it from interspersion of Cowardin polygons, low slope, and irregularity of the upland/wetland edge. This is a proxy for complex microtopography. During field assessment, surface depressions will be estimated directly.
Summary of strategy: Select floodplain wetlands only. Sum the points for Microtopo (2 max), LowSlope (2 max), and IrrEdge (1 max).

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Floodplain (Floodplain Location)
- Microtopo (Microtopographic Complexity)
- LowSlope (Low Slope)
- IrrEdge (Irregular Edge of the Upland/Wetland Boundary)

Method:

## Create feature class to store Depressions factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_Depressions

## Spatial join to add input variables to attribute table

Spatial join (contains) to add the following to the WU_Depressions: Floodplain, Microtopo, LowSlope, IrrEdge

## Add Depressions field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Depressions
Add field “Depressions” (short integer)
R-click Depressions and Field Calculate Depressions = 0

# Sum points for surface depressions (Depressions) in floodplain Wetland Units.

Clear all selections.
SELECT * FROM WU_Depressions WHERE: "Floodplain" = 'Y'
Field Calculate “Depressions” = [Microtopo] + [LowSlope] + [IrrEdge]

5.6.16 Discharges: Discharges to wetland within 100 m of boundary

Version date: 6 April 2016

Strategy: completed 3/11/2016 EAB
GIS method: completed 4/6/2016 EAB; verified 4/6/2016 EAB
Python coding: completed 6/9/2016 MCA
Final review by EAB: 10/3/2016

Purpose:

Water Quality Function
Max 2 points

Description:

Rationale: Wetlands can receive polluted waters even if they have well-vegetated and large buffers. For example, a pipe can discharge directly into a wetland, or a stream that drains areas where pollutants are released far from the unit can pass through the wetland. Also, silt fences often do not prevent all the sediment from reaching the wetland during construction. Other sources of pollutants may be septic tanks, NPDES discharges, Hydrological Protection Units (mining impacts), Acid Mine Lands, Acid Mine Drainage sites, and other sources that we are not currently able to include such as pesticide spraying on golf courses, particulates in exhausts from airplanes or motor vehicles, pesticides used in mosquito or gypsy moth control, and atmospheric deposition of mercury or other contaminants.
Summary of strategy: Assign 1 point to Wetland Units within 100 meters a septic risk or low-certainty NPDES permit location (excluding deep injection sites). Assign 2 points to Wetland Units within 100 meters of NPDES outlets (excluding deep injection sites), Well pads permitted within the last 5 years, Hydrologic Protection Units, Acid Mine Lands, Acid Mine Drainage sites, Superfund sites, and National Priority List sites.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\owrnpdes_.shp
- M:\wr\owrnpdes_outlets.shp
- M:\mr\hpu.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: AMLAMDFeb2016 (data provided by James Summers, DEP)
    - Feature Class: WellPads_20160325 (data provided by Laura Adkins, DEP)
    - Feature Class: NPL_point_20160406 (data provided by Peter Costello, DEP)
    - Feature Class: NPL_Bndry_20160406 (data provided by Peter Costello, DEP)

Input Variables:

Septic

Method:

## Create feature class to store Discharges variable

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_Discharges

## Add Discharges field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Discharges
Add field “Discharges” (short integer)
R-click Discharges and Field Calculate Discharges = 0

## PART 1: Select the owrnpdes_ records that are not deep injection points and not septic tanks.

Select by Attributes from owrnpdes_
Method: Create a new selection
SELECT * FROM owrnpdes_ WHERE: "perm_type" = '401 Certification' OR "perm_type" = 'Industrial' OR "perm_type" = 'Sewage' OR "perm_type" = 'UIC Sewage' OR "perm_type" = 'UIC Stormwater Industrial'

## Select Wetland Units within 100 m of Septic or selected NPDES records and assign 1 point.

Select by location
Selection method: select features from
Target Layer: WU_Discharges
Source layer: Septic
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target Layer: WU_Discharges
Source layer: owrnpdes_
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

In WU_Discharges, R-click Discharges and Field Calculate Discharges = 1

## PART 2: Select relevant records from owrnpdes_outlets that are not deep injection.

Select by Attributes from owrnpdes_outlets
Method: Create a new selection

SELECT * FROM owrnpdes_outlets WHERE: "perm_type" = 'Industrial' OR "perm_type" = 'Sewage' OR "perm_type" = 'UIC Sewage' OR "perm_type" = 'UIC Stormwater Industrial'

## Select relevant records (outlets with open status) from Hydrologic Protection Units.

Select by Attributes from hpu
Method: Create a new selection
SELECT * FROM hpu WHERE: "stat_flag" = 'O' AND "insp_type" = 'OUTLT' - The field names have changed. The below where clause is correct as of 3/15/2017 MCA
"STATUS_FLA" = 'O' AND "INSPECTA_1" = 'OUTLT'

## Select Wetland Units within 100 m of potential discharges and assign 2 points.

Select by location
Selection method: select features from
Target layer: WU_Discharges
Source layer: owrnpdes_outlets
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Discharges
Source layer: hpu
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Discharges
Source layer: AMLAMDFeb2016
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Discharges
Source layer: WellPads_20160325
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Discharges

Source layer: WV_NPL_point
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Discharges
Source layer: WV_NPL_Bndry
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 100 meters

In WU_Discharges, R-click Discharges and Field Calculate Discharges = 2

5.6.17 Disturb50m: Land use disturbance within 50 meters of wetland boundary

Version date: 26 October 2016

Strategy: completed 3/12/2016 EAB
GIS method: 10/26/2016 EAB; verified 10/26/2016
Python coding: Started 6/16/2016 & completed 1/25/2017 MCA. Note: This code needs to run on a server that has adequate memory and processing power.
Final review by EAB: 1/25/2017

Purpose:

Water Quality Function / Opportunity aspect
Max 3 points

Description:

Rationale: Farming, grazing, golf courses, residential areas, commercial land uses, urban areas, and developed areas in general, are major sources of pollutants (Sheldon et al. 2005). Tilled fields are a source of nutrients, pesticides, and sediment. Pastures are a source of nutrients and pathogenic bacteria, and clearcut areas are a source of sediment (Sheldon et al. 2005). A well-vegetated buffer of 50 meters will only remove 60-80% of some pollutants from surface runoff into a wetland. Thus, pollutants from such land uses will probably reach the wetland unit if they are within 50 meters of the wetland.
Summary of strategy: Calculate the ratio of disturbed area to total area within 50 meters of the Wetland Unit. Disturbed land uses include agricultural, pasture, golf course, residential, commercial, urban, or area that have been timbered within the last 5 years. Create 50 meter buffer around Wetland Unit. Merge the disturbed land use selections and assign points as follows:

- 1/10 to ¼ of buffer is covered by disturbed land uses → 1 point
- ¼ to ½ of buffer is covered by disturbed land uses → 2 points
- > ½ of buffer is covered by disturbed land uses → 3 points

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: DisturbedLand

Input Variables:

None

Method:

## Add WUKey field to Wetland Units

In attribute table of WU_20150514, add field “WUKey” (long integer)
Field calculate WUKey = OBJECTID

## Buffer Wetland Units by 50 meters.

Analysis Tools / Proximity / Buffer
Input Features: WU_20150514
Output Feature Class: Buffer50m
Distance (Linear unit): 50 meters
Side Type: OUTSIDE_ONLY
Dissolve Type: NONE

## Add field to store buffer area

In attribute table of Buffer50m, add field “BufferArea” (float)
Field calculate BufferArea = Shape_Area

## Intersect the 50m buffers and the disturbed land uses.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: Buffer50m
DisturbedLand
Output feature class: Buffer50mDist
Join attributes: ALL
Output type: INPUT

## Dissolve disturbed lands by wetland buffer

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer50mDist
Output Feature Class: Buffer50mDist_diss
Dissolve Fields: WUKey
Statistics Fields: BufferArea (Statistic Type = First)
Check box “Create multipart features” (default)

## Add field and calculate ratio of disturbed area to total drainage area.

Open attribute table of Buffer50mDist_diss
Add field “Dist50mRat” (float)
Field calculate Dist50mRat = [Shape_Area] / [FIRST_BufferArea]

## Join ratio of disturbed land to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_20150514

Input Join Field: WUKey
Join Table: Buffer50mDist_diss
Output Join Field: WUKey

## Export joined data

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_Disturb50m

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Disturb50m
Add field “Disturb50m” (short integer)
R-click Disturb50m and Field Calculate Disturb50m = 0

## Assign points.

SELECT * FROM WU_Disturb50m WHERE: "Dist50mRat" > 0.1
R-click Disturb50m and Field Calculate Disturb50m = 1

SELECT * FROM WU_Disturb50m WHERE: "Dist50mRat" > 0.25
R-click Disturb50m and Field Calculate Disturb50m = 2

SELECT * FROM WU_Disturb50m WHERE: "Dist50mRat" > 0.5
R-click Disturb50m and Field Calculate Disturb50m = 3

5.6.18 DisturbWshd: Land use disturbance within contributing watershed

Version date: 25 October 2016

Strategy: completed 3/12/2016 EAB
GIS method: 4/20/2016 EAB; verified 10/25/2016
Python coding: Started 5/27/2016 MCA & Completed 11/22/2016
Final review by EAB: 11/28/2016

Purpose:

Water Quality Function / Opportunity aspect
Max 1 point

Description:

Rationale: Farming, grazing, golf courses, residential areas, commercial land uses, urban areas, and developed areas in general, are major sources of pollutants (Sheldon et al. 2005). Tilled fields are a source of nutrients, pesticides, and sediment. Pastures are a source of nutrients and pathogenic bacteria, and clearcut areas are a source of sediment (Sheldon et al. 2005). The presence of these sources in the contributing watershed of a wetland is a good indicator that pollutants may be reaching the wetland.
Summary of strategy: Calculate the ratio of disturbed area to total area within the drainage areas of each Wetland Unit. Disturbed land uses include agricultural, pasture, golf course, residential, commercial, urban, or area that have been timbered within the last 5 years. Merge the disturbed land use selections and assign 1 point if more than 10% of the contributing watershed is disturbed.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
  - Feature Class: DrainageArea27m
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: DisturbedLand

Input Variables:

None

Method:

## Intersect the drainage areas and the disturbed land uses.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: DrainageArea27m
DisturbedLand
Output feature class: DrainAreaDist

Join attributes: ALL
Output type: INPUT

## Dissolve disturbed lands by drainage area

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: DrainAreaDist
Output Feature Class: DrainAreaDist_diss
Dissolve Fields: WUKey
Statistics Fields: CntrWshd (Statistic Type = First)
Check box "Create multipart features" (default)

## Add field to DrainAreaDist_diss and calculate ratio of disturbed area to total drainage area.

Open attribute table of DrainAreaDist_diss
Add field "DistWshdRat" (float)
Field calculate DistWshdRat = [SHAPE_Area] / [FIRST_CntrWshd]

## Join ratio of disturbed land to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_20150514
Input Join Field: WUKey
Join Table: DrainAreaDist_diss
Output Join Field: WUKey

## Export joined data

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_DisturbWshd

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_DisturbWshd
Add field "DisturbWshd" (short integer)
R-click DisturbWshd and Field Calculate DisturbWshd = 0

## Assign points.

SELECT * FROM WU_DisturbWshd WHERE: DistWshdRat > 0.1
R-click DisturbWshd and Field Calculate DisturbWshd = 1

5.6.19 EconRisk: Economically Valuable Flood Risk Area Downstream of Wetland

Version date: 9 November 2016

Strategy: 4/21/2016 EAB
GIS method: 4/22/2016 EAB: verified for Barbour and Berkeley counties, need TotalLossRP100 layer to set quintile thresholds. Layer created by MCA 11/9/16. Quintile thresholds set and method revised by EAB 11/9/16. Method verified EAB 11/9/16.
Python coding: Started 11/9/2016 MCA & Completed 11/10/2016 MCA
Final verification by EAB:

Purpose:

Flood Attenuation / Value to Society
Max 4 points (all wetland types)

Description:

Rationale: Wetlands upstream of economically valuable flood-prone infrastructure (structures, roads, developed lands, cropland) can reduce the costs and negative impacts of flood damages on society (WI GIS-RAM).
Strategy: Wetland Unit is located in or near a census block with significant predicted total losses during a 100-year flood (Hazus census block data). In West Virginia, wetlands tend to be in headwaters, while infrastructure is generally in the bottomlands. Therefore we will approximate "wetland upgradient of risk area" by using three levels of increasing distance: (a) co-location within a census block with predicted losses, (b) 1 km distance from a census block with predicted losses, and (c) co-location within a HUC12 watershed that contains predicted losses. This will approximately capture wetlands that are upgradient of census blocks with predicted flood losses. Assign points as follows:

- 4 points: WU intersects TotalLossRP100 > 1204 (top quintile or $1,204,000-$246,103,000)
- 3 points: WU intersects TotalLossRP100 = 200 – 1204 OR is within 1 km of top quintile
- 2 points: WU intersects TotalLossRP100 = 42 - 200 OR is within 1 km of fourth quintile
- 1 point: WU intersects TotalLossRP100 > 0 – 42 OR WU is in a HUC12 watershed with TotalLossRP100 > 0
- 0 point: WU is in a HUC12 watershed with TotalLossRP100 = 0 (bottom quintile)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\FloodplainData.gdb
  - Feature Class: TotalLossRP100
- M:\basemap\watersheds_12digit.shp

Method:

## Create new feature class to store EconRisk data.

R-click WU_20150514 and select Data / Export Data

Output feature class: WU_EconRisk
Open attribute table and add field EconRisk (short integer)
Field calculate EconRisk = 0

## Select Census blocks and assign points to Wetland Units.
## Assign 1 point to wetlands within HUC12 with loss areas > 0.

Select by attributes from TotalLossRP100
Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 0

Select by location
Select Method: select features from
Target layer: watersheds_12digit
Source layer: TotalLossRP100
Check box "Use selected features"
Spatial selection method: intersect the source layer feature

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: watersheds_12digit
Check box "Use selected features"
Spatial selection method: intersect the source layer feature

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 1

## Assign 2 points to wetlands in middle quintile or within 1 km of second highest quintile.

Select by attributes from TotalLossRP100
Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 42

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: TotalLossRP100
Check box "Use selected features"
Spatial selection method: intersect the source layer feature

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 2

Select by attributes from TotalLossRP100

Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 200

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: TotalLossRP100
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 1000 meters

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 2

## Assign 3 points to wetlands in second highest quintile or within 1 km of highest quintile.

Select by attributes from TotalLossRP100
Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 200

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: TotalLossRP100
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 3

Select by attributes from TotalLossRP100
Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 1204

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: TotalLossRP100
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 1000 meters

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 3

## Assign 4 points to wetlands in highest quintile.

Select by attributes from TotalLossRP100
Method: Create a new selection
SELECT * FROM TotalLossRP100 WHERE: "TotalLossRP100" > 1204

Select by location
Select Method: select features from
Target layer: WU_EconRisk
Source layer: TotalLossRP100
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

R-click WU_EconRisk and open attribute table
In WU_EconRisk, Field Calculate EconRisk = 4

5.6.20 FAFunction: Flood Attenuation

Version date: 7 March 2017

Strategy: completed 4/21/2016 EAB
GIS method: completed 4/22/2016 EAB; verified 3/7/2017 EAB
Python code: 3/13/2017 MCA
Final review by EAB: 6/20/2017 EAB

Purpose:

Flood Attenuation Function
Maximum 24 points (floodplain wetlands); 20 points (groundwater wetlands)

Description:

Rationale: The flood attenuation function is a measure of the effectiveness of a wetland in storing water or delaying the downgradient movement of water, thus potentially influencing the height, timing, duration, and frequency of flooding in downstream areas. Many wetlands are capable of slowing the downslope movement of water, regardless of whether they have significant storage capacity. Water that is slowed, or stored, in a wetland becomes potentially available for recharging baseflow of streams or aquifers, and supporting local food webs.
Strategy: For each Wetland Unit, sum the points for the three aspects (wetland potential to provide function, landscape offers opportunity to carry out function, and value to society)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514

Input Variables:

- FAPotential (16 points max for floodplain wetlands; 14 points max for groundwater wetlands)
- FAOpportun (4 points max for floodplain wetlands; 2 points max for groundwater wetlands)
- FASociety (4 points max for all wetland types)

Method:

## Create feature class to store FAFunction

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_FAFunction

## Spatial join to bring in aspect values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_FAPotential
Join Feature: WU_Opportun

Output feature class: WU_FAFunction1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
FAPotential
FAOpportun
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_FAFunction1
Join Feature: WU_FASociety
Output feature class: WU_FAFunction
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
FAPotential
FAOpportun
FASociety
Match Option: CONTAINS

## Add FAFunction field to Wetland Units and set initial point value to zero.

Open attribute table of WU_FAFunction
Add field "FAFunction" (short integer)
R-click FAFunction and Field Calculate FAFunction = 0

## Sum the points for each aspect of Water Quality Function for Wetland Units

R-click FAFunction and Field Calculate FAFunction = [FAPotential] + [FAOpportun] + [FASociety]

5.6.21 FAOpportun: Flood Attenuation Opportunity

Version date: 7 March 2017

Strategy: completed 4/22/2016 EAB
GIS method: completed 11/20/2016 EAB; revised 12/19/2016 EAB; verified 3/17/2017
Python code:
Final check by EAB:

Purpose:

Flood Attenuation Function
Maximum 4 points (floodplain wetlands); Maximum 2 points (groundwater wetlands)

Description:

Rationale: Wetlands that are well-connected to their historic floodplains, are surrounded by runoff-producing areas, or have catchments with steep slopes, all tend to receive flood waters and have high opportunity to attenuate floods.
Strategy: For each Wetland Unit, sum the points for all factors within the "Opportunity" aspect. Restrict groundwater wetlands to a maximum of two points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514

Input Variables:

- FloodIn (2 points max) (all wetlands) from WU_FloodIn
- ConnectFL (2 points max) (all wetlands) from WU_Connect
- Floodplain from WU_Connect (or from WU_Floodplain)

Method:

## Bring together factor values and output feature class to store FAOpportun.

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_20150514
Join Feature: WU_FloodIn
Output feature class: WU_FAOpportun1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
FloodIn
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_FAOpportun1
Join Feature: WU_Connect
Output feature class: WU_FAOpportun2
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
FloodIn
ConnectFL
Floodplain
Match Option: CONTAINS

## Add FAOpportun field to Wetland Units and set initial point value to zero.

Open attribute table of WU_FAOpportun
Add field "FAOpportun" (short integer)
R-click FAOpportun and Field Calculate FAOpportun = 0

## Sum the factor points.

R-click FAOpportun and Field Calculate FAOpportun = [FloodIn] + [ConnectFL]

## Reduce points for groundwater wetlands to a maximum of 2.

Open attribute table of WU_FAOpportun
SELECT * FROM WU_FAOpportun WHERE: "FAOpportun" > 2 AND "Floodplain" = 'N'
(1369 out of 43124 selected)
Field Calculate (selection only) "FAOpportun" = 2

5.6.22 FAPotential: Flood Attenuation Potential

Version date: 1 September 2017

Strategy: completed 3/24/2016 EAB
GIS method: completed 3/24/2016 EAB; verified 12/20/2016 EAB; re-verified with revised Runoff values 1/9/2017 EAB
Python code: 12/22/2016 MCA; 1/10/2017 MCA (Re-ran Code)
Final review by EAB: 1/11/2017

Purpose:

Flood Attenuation Function
17 points maximum FL
14 points maximum GW

Description:

Rationale**:** The intrinsic potential of a wetland to attenuate floods depends on a number of factors, including its location in the watershed, slope, the structure and density of vegetation, soil infiltration capacity, microtopography, and the type of surface water outlet.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the "Potential" aspect. Note that different point values for floodplain vs. groundwater wetlands are assigned at the factor level.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Headwater (1 point – all types)
- LowSlope (2 points – all types)
- VegFA (9 points FL, 5 points GW)
- Runoff (5 points FL, 4 points GW)
- SWOutflow2 (2 points GW)

Method:

## Create feature class to store FAPotential

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_FAPotential

## Spatial join to bring in factor values

Spatial join (contains) to add the following to the WU_FAPotential table: Headwater, LowSlope, VegFA, Runoff, SWOutflow2

## Add FAPotential field to Wetland Units and set initial point value to zero.

Open attribute table of WU_FAPotential
Add field “FAPotential” (short integer)
R-click FAPotential and Field Calculate FAPotential = 0

## Sum the factor points

R-click FAPotential and Field Calculate FAPotential = [Headwater] + [LowSlope] + [VegFA] +[Runoff] + [SWOutflow2]

5.6.23 FASociety: Flood Attenuation Value to Society

Version date: 10 November 2016

Strategy: completed 4/22/2016 EAB
GIS method: completed 4/22/2016 EAB; verified 11/10/2016 EAB
Python code: Started & completed 11/10/2016 MCA
Final check by EAB: 11/10/2016

Purpose:

Flood Attenuation Function
Maximum 4 points (all wetlands)

Description:

Rationale: Wetlands in regulatory floodways or upstream of economically valuable flood-prone areas can reduce the costs and negative impacts of flood damages to society.
Strategy: For each Wetland Unit, sum the points for all factors within the "Society" aspect. Reduce values that exceed the maximum allowable points for this aspect of flood attenuation function. Note that floodplain and groundwater wetlands are treated the same for this aspect.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514

Input Variables:

- Floodway (4 points max)
- EconRisk (4 points max)

Method:

## Create feature class to store FASociety

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_FASociety1

## Spatial joins to bring in factor values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_FASociety1
Join Feature: WU_Floodway
Output feature class: WU_FASociety2
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length

Shape_Area
Floodway
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_FASociety2
Join Feature: WU_EconRisk
Output feature class: WU_FASociety
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
Floodway
EconRisk
Match Option: CONTAINS

## Add FASociety field to Wetland Units and set initial point value to zero.

Open attribute table of WU_FASociety
Add field "FASociety" (short integer)
R-click FASociety and Field Calculate FASociety = 0

## Sum the factor points

R-click FASociety and Field Calculate FASociety = [Floodway] + [EconRisk]

## Reduce values that exceed the maximum allowable points

Select by attributes
Layer: WU_Society
Method: Create a new selection
SELECT * FROM WU_FASociety WHERE: "FASociety" > 4
(1184 out of 43124 selected)

R-click FASociety and Field Calculate FASociety = 4

5.6.24 Fisheries: Wetland discharges to economically important fisheries

Version date: 17 July 2016

Strategy: 2/14/2016 EAB
GIS method: 3/16/2016 EAB
Python coding: started and completed 3/16/2016
Final review by EAB:

Purpose:

Water Quality Function
Max 2 points

Description:

Rationale: Wetlands filter sediments and contaminants, and buffer pH. Wetlands in the contributing basin of an economically important fishery are of high economic and social value.
Summary of strategy: Approximated as Wetland Unit located within 1 km of an economically important fishery.

- Fishery. Wetland is in the contributing watershed of a high quality fishery, warmwater fishery stream, stocked trout stream, or a stream with year-round trout populations (1 point). Note that warmwater fisheries are generally included in the "high quality fishery" layer (they can also perhaps be approximated by all polygonal streams below 2000 feet elevation, according to Mike Shingleton (WVDNR), pers. comm. March 2015, but we will assume they are covered by the high quality fisheries data).
- Special Fishery. Wetland is in the contributing watershed of a catch-and-release area, children/Class Q fishing area, or fly-fishing-only stream (2 points).

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
    - M:\wr\WTRSHD_BRANCH\TROUT\Trout_Streams.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_Fishing_20Aug2015\HighQualityStreamFisheriesWVDNR20150820.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_Fishing_20Aug2015\TrStStreams.shp

Input Variables:

None

Method:

## Create feature class to store Fisheries variable

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_Fisheries

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Fisheries
Add field “Fisheries” (short integer)
R-click Fisheries and Field Calculate Fisheries = 0

## Select Wetland Units within 1 km of perennial trout streams.

Select by location
Selection method: select features from
Target layer: WU_Fisheries
Source layer: Trout_Streams.shp
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 1000 meters

## Select Wetland Units within 1 km of high quality stream fisheries.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Fisheries
Source layer: HighQualityStreamFisheriesWVDNR20150820.shp
Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Select Wetland Units within 1 km of stocked trout streams.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_Fisheries
Source layer: TrStStreams.shp
Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Assign 1 point to Wetland Units that discharge to economic fisheries.

R-click “Fisheries” in WU_Fisheries and Field Calculate Fisheries = 1

## Select special fisheries

Select by Attributes
Layer: TrStStreams
Method: Create a new selection
SELECT * FROM TrStStreams WHERE: "StockCode" NOT LIKE 'NS'

Select by Attributes
Layer: TrStStreams

Method: Add to current selection
SELECT * FROM TrStStreams WHERE: "RegType" = 'Catch-and-Release' OR "RegType" = 'Children and Class Q' OR "RegType" = 'Fly-fishing-Only'

## Select Wetland Units within 1 km of special fisheries.

Select by location
Selection method: select features from
Target layer: WU_Fisheries
Source layer: TrStStreams
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 1000 meters

## Assign 2 points to Wetland Units within 1 km of special fisheries.

R-click “Fisheries” in WU_Fisheries and Field Calculate Fisheries = 2

5.6.25 FloodArea: Proportion of wetland area in floodplain, including the intermediate variable FloodRatio, and Floodplain (Y/N)

Version date: 8 March 2017

Strategy: Completed 4/13/2016 EAB
GIS method: Drafted 4/20/2016 EAB; Verified 12/5/2016; Revised 3/8/2017 EAB to include Floodplain
Python code: 12/21/2016 MCA; need revision
Final review by EAB: 12/22/2016; need re-check

Purpose:

Floodplain (Y/N): Input to numerous Water Quality and Flood Attenuation metrics
FloodArea: Input to Flood Attenuation / Opportunity
    Max 2 points (all wetlands, but only Floodplain wetlands will score high enough to get points)

Description:

Floodplain (Y/N)
Rationale: Wetlands that receive overland flood flows sometimes have differing functions from those that are primarily groundwater-fed. “Floodplain” and “Groundwater” wetlands are therefore assessed using different variables in some cases. For example, surface depressions are important in holding floodwaters in a wetland, and woody vegetation physically slows flood flows and associated debris. Groundwater wetlands, with slower-moving subsurface flows, have water quality functions that are more dependent on the presence of organic soils, clay soils, or the irregularity of the upland-wetland edge.
Summary of strategy: Wetland Units with 10% or greater of their area in the FEMA floodplain or Active River Area Base Zone are considered floodplain wetlands unless they have known peat deposits, in which case they are put in the groundwater wetland group. This technique probably over-estimates the actual number of floodplain wetlands, but it is the best method we currently have.

Proportion of wetland area in floodplain (FloodArea). Max 2 points.
Rationale: Floodplain wetlands store and slow water movement during floods and storms. The amount of flood attenuation in a wetland is related to the amount of overbank flooding it receives, which in turn is related to its position in the floodplain. Wetlands that are entirely within the floodplain have more opportunity to attenuate floods than wetlands that are only partially in a floodplain.
Strategy: Calculate ratio of Wetland Unit area that lies in the floodplain (either FEMA or ARA). Peatlands are set to zero since they are primarily groundwater, not floodplain, wetlands.
    Ratio > 0.5 = 2 points
    Ratio 0.1-0.5 = 1 point
    Ratio < 0.1 = 0 points.
Note that only Floodplain wetlands will receive these points, since by definition they have a ratio > 0.1.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\FloodplainData.gdb
    - Feature Class: FloodplainARAFEMA
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: Peatlands_20160228

Method:

## Intersect floodplain and Wetland Units

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: FloodplainARAFEMA
WU_20150514
Output feature class: WU_FloodArea1
Join attributes: ONLY_FID
Output type: INPUT

## Add field to store floodplain area.

Open attribute table of WU_FloodArea1
Add field "FloodAreaAF" (float)
R-click "FloodAreaAF" and Calculate Geometry.
Property: Area
Coordinate System: Use coordinate system of the data source
Units: Square Meters [sq m]

## Spatial Join floodplain selection to Wetland Units and sum floodplain area.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: WU_FloodArea1
Output Feature Class: WU_FloodArea
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape_Length
Shape_Area
WUKey
FloodAreaAF (R-click and select "Merge Rule", "Sum")
Match Option: INTERSECT

## Add fields to store Flood Area Ratio and Flood Area points.

Open attribute table of WU_FloodArea
Add field FloodRatio (float) to WU_FloodArea attribute table.
Add field FloodArea (short integer) to WU_FloodArea attribute table.
Field calculate FloodArea = 0

## Calculate the ratio of floodplain area to total Wetland Units area.

Field calculate FloodRatio = [FloodAreaAF] / [Shape_Area]

## Assign points

SELECT * FROM WU_FloodArea WHERE: FloodRatio > 0.1
Field Calculate FloodArea = 1

SELECT * FROM WU_FloodArea WHERE: FloodRatio > 0.5
Field Calculate FloodArea = 2

## Select Wetland Units that contain peat deposits

Clear selection.
Select by Location
Selection Method: Select features from
Target Layer: WU_FloodArea
Source Layer: Peatlands_20160228
Spatial selection method: Intersect the source feature
(277 out of 43124 selected)

## Set FloodArea to zero for Wetland Units that contain peat deposits.

R-click FloodArea and Field Calculate “FloodArea” = 0

Add the following to compute “Floodplain” (Y/N)

## Add field Floodplain to Wetland Units attribute table and set initial value to “N”.

Add field Floodplain (Text, 2 characters) to WU_Floodplain attribute table.
R-click Floodplain and Field Calculate “Floodplain” = “N”

## Select Wetland Units that have at least 10% of their area in a FEMA floodplain or Active River Area base zone.

SELECT * FROM WU_FloodArea WHERE: FloodArea > 0

Set Floodplain = “Yes” for selected Wetland Units.

R-click Floodplain and Field Calculate “Floodplain” = “Y”

5.6.26 FloodIn: Floodwaters Delivered to Wetland

Version date: 7 March 2017

Strategy: Completed 11/20/2016 EAB
GIS method: completed 11/20/2016 EAB, verified 3/7/2017
Python coding: 3/7/2017 MCA
Final review by EAB: 3/7/2017 EAB

Purpose:

Input to Flood Attenuation / Potential
Max 2 points (all wetlands).
Rationale: Wetlands are more likely to receive flood waters if there are steep slopes in their contributing watershed, and if the land surrounding the wetland has high runoff potential.
Summary of strategy: Sum the points for SlopeWshd, Runoff50m, and RunoffWshd. Assign points as follows:

    0 points: sum is (0,1,2).
    1 point: sum is (3,4).
    2 points: sum is (5,6).

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- SlopeWshd (2 points)
- Runoff50m (2 points)
- RunoffWshd (2 points)

Method:

## Spatial joins to add input variables to Wetland Units attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_20150514
Join Feature: WU_SlopeWshd
Output Feature Class: WU_FloodIn1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
SlopeWshd
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_FloodIn1
Join Feature: WU_Runoff50m
Output Feature Class: WU_FloodIn2
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
SlopeWshd
Runoff50m
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_FloodIn2
Join Feature: WU_RunoffWshd
Output Feature Class: WU_FloodIn
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
SlopeWshd
Runoff50m
RunoffWshd
Match option: CONTAINS

## Add FloodIn field to Wetland Units and set initial point value to zero.

Open attribute table of WU_FloodIn
Add field "FloodIn" (short integer)
R-click FloodIn and Field Calculate FloodIn = 0

## Calculate sum of SlopeWshd, Runoff50m, RunoffWshd and assign points to FloodIn.

Open attribute table of WU_FloodIn
Select * from WU_FloodIn where: ("RunoffWshd"+ "SlopeWshd"+ "Runoff50m") > 2
R-click FloodIn and Field Calculate "FloodIn" = 1
Select * from WU_FloodIn where: ("RunoffWshd"+ "SlopeWshd"+ "Runoff50m") > 4
R-click FloodIn and Field Calculate "FloodIn" = 2

5.6.27 Floodway: Wetland is in a FEMA Floodway

Version date: 21 April 2016

Strategy: 3/28/2016 EAB
GIS method: completed 4/21/2016 EAB; verified 4/21/EAB
Python code: started & completed 5/27/2016 MCA
Final review by EAB: 10/3/2016

Purpose:

Flood Attenuation Function / Value to Society
Maximum 4 points

Description:

Rationale: Regulatory floodways have been identified by FEMA as high priorities for flood control, with strict limits on development. Wetlands occurring in a regulatory floodway have a high value to society. FEMA: *A "Regulatory Floodway" means the channel of a river or other watercourse and the adjacent land areas that must be reserved in order to discharge the base flood without cumulatively increasing the water surface elevation more than a designated height. Communities must regulate development in these floodways to ensure that there are no increases in upstream flood elevations.*
Strategy: Wetland intersects a mapped FEMA Regulatory Floodway. 4 points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\wvFloodHazardFeatures_WVGISTC_20130410\wvFloodHazardFeatures20130207.gdb
    - Feature Class: WV_Floodway_20130205_wgs84wmA

Method:

## Create feature class to store Floodway points and set initial value

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_Floodway
Add field to attribute table of WU_Floodway: Floodway (short integer)
Field calculate Floodway = 0

## Select Wetland Units that intersect a Floodway

Select by Location
Selection method: select features from
Target layer: WU_Floodway
Source layer: WV_Floodway-20130205_wgs84wmA

Spatial selection method: intersect the source feature

## Assign points

In WU_Floodway, Field calculate Floodway = 4

5.6.28 Function: Total Wetland Function

Version date: 5 November 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/20/2017 EAB; results verified 10/20/2017 EAB; re-verified with revised BRank 11/5/17
Python code: 10/30/2017 YH
Final review by EAB: 11/5/2017

Purpose:

Maximum 200 points

Description:

Rationale:
Strategy: Sum the scores for WQFunction, FAFunction, and HFunction

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_WQFunction
    - Feature Class: WU_FAFunction
    - Feature Class: WU_HFunction

Method:

## Spatial Join to merge metrics and create feature class to store Function

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQFunction
Join Feature: WU_FAFunction
Output feature class: WetlandFunction.gdb \ WU_Function1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
WQFunction
FAFunction
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_Function1
Join Feature: WU_HFunction
Output feature class: WetlandFunction.gdb \ WU_Function
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following

WUKey
    Shape-Length
Shape_Area
WQFunction
FAFunction
HFunction
Match Option: CONTAINS

## Add Function and assign values

Open attribute table of WU_Function
Add field “Function” (short integer)

R-click Function and Field Calculate Function = [WQFunction]+ [FAFunction]+ [HFunction]

5.6.29 Headwater: Headwater Location

Version date: 18 April 2016

Strategy: Completed 2/27/2016 EAB
GIS method: Completed 4/12/2016 EAB, verified 4/18/2016 EAB
Python coding: Started & Completed 6/15/2016 MCA
    EAB 10/3/2016: re-run after WFlowPath and LandPos are updated; MCA 10/18/16 completed
Final review by EAB: 10/25/2016

Purpose:

Input to Water Quality (Potential) and Flood Attenuation (Potential) functions
Max 1 point.

Description:

Rationale: Headwater wetlands provide water quality and hydrologic stability benefits to waters downstream. Wetlands found in the headwaters of streams often do not store surface water to any great depth. They can, however, be important in reducing peak flows because they slow down and "desynchronize" the initial peak flows from a storm (Brassard and others 2000). Their importance in hydrologic functions is often under-rated. In the words of Michael Davis, Deputy Assistant of the Army, to the U.S. Senate: "The most recent data and scientific literature indicate that isolated and headwater wetlands often play an ecological role that is as important as other types of wetlands in protecting water quality, reducing flood flows, and providing habitat for many species of fish and wildlife" (Davis 1997).
Summary of strategy: Assign one point if the Wetland Unit has a Tiner Landscape Position (LandPos) with a headwater modifier.

Definitions:

Tiner (2011) defines lotic headwater wetlands as "wetlands along first- and second-order perennial streams in hilly terrain including all intermittent streams above these perennial streams". He defines terrene headwater wetlands as "wetland is the source of a river or stream but this watercourse does not extend through the wetland". These are coded in our database as "LandPos" = 'LSh' (lotic stream headwater wetlands and "LandPos" = 'TEh' (terrene headwater wetlands).

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- LandPos (Landscape Position)

Method:

## Spatial join to add input variable to attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_20150514
Join Feature: WU_LandPos
Output feature class: WU_Headwater
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape_Length
Shape_Area
LandPos
Match Option: CONTAINS

## Add Headwater field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Headwater
Add field “Headwater” (short integer)
R-click Headwater and Field Calculate Headwater = 0

## Assign 1 point to Wetland Units with Landscape Position headwater modifier.

Clear all selections.
SELECT * FROM WU_Headwater WHERE: "LandPos" LIKE '%h'
Field Calculate (selection only) “Headwater” = 1

5.6.30 HFuncNoBR: Habitat and Ecological Integrity Function without Site Biodiversity Rank

Version date: 19 October 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/16/2017 EAB; results verified 10/16/2017 EAB
Python code: 10/19/2017 YH
Final review by EAB: 10/19/2017

Purpose:

Input to Habitat and Ecological Integrity Function
Maximum 50 points

Description:

Rationale:
Strategy: For each Wetland Unit, sum the points for the three aspects (wetland potential to provide function, landscape offers opportunity to carry out function, and value to society).

- HPotential (30 points max)
- HOpportun (13 points max)
- HSociety (7 points max)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_HPotential
    - Feature Class: WU_HOpportun
    - Feature Class: WU_HSociety

Method:

## Spatial Join to merge metrics and create feature class to store HFuncNoBR

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_HPotential
Join Feature: WU_HOpportun
Output feature class: WetlandFunction.gdb \ WU_HFuncNoBR1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
HPotential
HOpportun
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join

Target Feature: WU_HFuncNoBR1
Join Feature: WU_HSociety
Output feature class: WetlandFunction.gdb \ WU_HFuncNoBR
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
        Shape-Length
Shape_Area
HPotential
HOpportun
HSociety
Match Option: CONTAINS

## Add HFuncNoBR field to Wetland Units and set initial point value to zero.

Open attribute table of WU_HFuncNoBR
Add field “HFuncNoBRn” (short integer)
R-click HFuncNoBR and Field Calculate HFuncNoBR = 0

## Sum the points

R-click HFuncNoBR and Field Calculate HFuncNoBR = [HPotential] + [HOpportun] + [HSociety]

5.6.31 HFunction: Habitat and Ecological Integrity Function

Version date: 5 November 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/26/2017 EAB; results verified 10/26/2017 EAB; re-verified with new BRank 11/5/2017
Python code: 10/26/2017 YH
Final review by EAB: 10/30/2017 EAB

Purpose:

Maximum 150 points

Description:

Rationale:
Strategy: Incorporate the Site Biodiversity Rank (BRank) and HFuncNoBR into HFunction as follows:

B1	Outstanding Global Biodiversity Significance: automatically assigned maximum points for habitat function x 3 (HFunction = 150 points)
B2	High Global Biodiversity Significance: automatically assigned maximum points for habitat function x 2 (HFunction = 100 points)
B3	Global Biodiversity Significance: automatically assigned maximum points for habitat function x 1.5 (HFunction = 75 points)
B4	Outstanding State Biodiversity Significance: automatically assigned maximum points for habitat function (HFunction = 50 points)
B5	State Biodiversity Significance: automatically assigned maximum points for "intrinsic potential" part of habitat function (HPotential = 30 points)
B6	Local Biodiversity Significance: automatically awarded an additional 5 points toward the maximum of 30 for the "intrinsic potential" part of habitat function (HPotential)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_BRank
  - Feature Class: WU_HFuncNoBR

Method:

## Spatial Join to merge metrics and create feature class to store HFunction

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_BRank
Join Feature: WU_HFuncNoBR
Output feature class: WetlandFunction.gdb \ WU_HFunction
Join Operation: JOIN_ONE_TO_ONE
Keep all target features

Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
HPotential
    HOpportun
HSociety
HFuncNoBR
BRank
Match Option: CONTAINS

## Add fields to store intermediate value for B6 sites and results and set initial values

Open attribute table of WU_HFunction
Add field “HPotB6” (short integer)
Field calculate HPotB6 = 0
Add field “HFunction” (short integer)
Field Calculate HFunction = [HFuncNoBR]

## Select B6 wetlands

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B6'

## Add 5 points to HPotential for B6 wetlands, up to a maximum of 30 points

Open attribute table of WU_HFunction
Field calculate HPotB6 = [HPotential] + 5
SELECT * FROM WU_HFunction WHERE: "HPotB6" > 30

Field calculate HPotB6 = 30

## Select B6 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B6'

Field Calculate HFunction = [HPotB6] + [HOpportun]+ [HSociety]

## Select B5 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B5'

Field Calculate HFunction = [HOpportun]+ [HSociety] + 30

## Select B4 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B4'

Field Calculate HFunction = 50

## Select B3 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B3'

Field Calculate HFunction = 75

## Select B2 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B2'

Field Calculate HFunction = 100

## Select B1 wetlands and calculate HFunction

Open attribute table of WU_HFunction
SELECT * FROM WU_HFunction WHERE: "BRank" = 'B1'

Field Calculate HFunction = 150

5.6.32 HInvest: Societal Investment in Habitat and Ecological Integrity

Version date: 20 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/16/2017 EAB; results verified 10/19/2017 EAB
Python code: 10/19/2017 YH
Final review by EAB: 10/19/2017 EAB

Purpose:

Input to Habitat / Value to Society

Description:

Maximum 3 points

Rationale: Society values wetland habitats by investing in them through management and conservation actions.

GIS and Field Method: Note that field data can overwrite the GIS determination of this metric. Assign points according to the criteria met:

3 points: high investment. Award 3 points if any of the following criteria are met.

- Mitigation investment: wetland is all or part of a mitigation site used explicitly to offset impacts elsewhere.
- Conservation investment: wetland is part of or contiguous to lands which public or private organizational funds were spent to preserve, create, restore, or enhance habitat and not used explicitly to offset impacts elsewhere.
    - Conservation easement managed for biodiversity, i.e., Gap Status Code = 1 (TNC, some land trust holdings) *Check with DNR (or Michael Schwartz of Freshwater Institute) every two years to see if there are updates to: WV_Protected_Lands_2015_PUBLIC*
    - USDA
        - FSA Conservation Reserve Program (CRP)
        - NRCS Wetland Reserve Program
        - NRCS Emergency Watershed Protection easement
        - USFS Forest Legacy easement
        - USFS Special Botanical Area
        - USFS Wilderness Area
    - USFWS National Wildlife Refuge
    - USNPS National Park, Monument, or Scenic River
    - WVDNR State Natural Area

2 points: moderate investment in a general area including the wetland, with some focus on habitat and ecological integrity along with other functions. Assign 2 points if any of the following contain the Wetland Unit.

- USFS National Forest outside wilderness or special botanical areas

1 point: low investment: wetlands that are on public land where the primary focus is on functions (e.g., recreation, military operations) other than ecological conservation or restoration, but where wetlands are unlikely to be destroyed or severely adversely impacted.

- US Department of Defense lands
- WVDNR State Parks
- WVDNR Wildlife Management Areas (open access, Gap status 2)
- WVDOF State Forests (GAP status: managed for multiple uses, subject to extractive, e.g. mining or logging, or OHV use)
- City and County Parks (all are open access, GAP status code = “no known mandate”)
- Natural Streams Preservation Act (NSPA). Wetland is in the contributing watershed of a stream reach protected by the Natural Streams Preservation Act. These include (a) Greenbrier River from its confluence with Knapps Creek to its confluence with the New River, (b) Anthony Creek from its headwaters to its confluence with the Greenbrier River, (c) Cranberry River from its headwaters to its confluence with the Gauley River, (d) Birch River from the Cora Brown bridge in Nicholas county to the confluence of the river with the Elk River, and (e) New River from its confluence with the Gauley River to its confluence with the Greenbrier River.

0 points: no known investment

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived
  - \Boundaries2017\countyCityParkBoundaries_20107731_utm83.gdb
    - Feature Class: countyCityParkBoundaries_20170731_utm83
  - \Boundaries2017\nationalForestOwnership_USFWS_20170803_utm83.gdb
    - Feature Class: nationalForestOwnership_USFWS_20170803_utm83
      - Field: Forest Service
  - \Boundaries2017\nationalParkBoundaries_nationalParkService_20170802.gdb
    - Feature Class: nationalParkBoundaries_nationalParkService_20170802
  - \Boundaries2017\nationalWildifeRefuge_USFWS_20170803_utm83.gdb
    - Feature Class: nationalWildifeRefuge_USFWS_20170803
  - \Boundaries2017\wvdnrManagedLands_wvdnr_20170731_utm83.gdb
    - Feature Class: wvdnrManagedLands_wvdnr_20170731_utm83
  - \Boundaries2017\wvStateForestBoundaries_wvdof_20171003_utm83.gdb
    - Feature Class: wvStateForestBoundaries_wvdof_20171003_utm83
  - \Boundaries2017\stateParkBoundaries_WVDNR_20170927_utm83
    - Feature Class: stateParkBoundaries_WVDNR_20170927_utm83
  - \USFS\botanical_areas_MNF.shp
    - ***Note: do not distribute this layer outside WVDEP - it is sensitive data!***
  - \WV_Protected_Lands_v2013c_Public\WV_Protected_Lands_2015_PUBLIC.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Watershed.gdb
  - Feature Class: NatStrPreAct_HUC10
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: ILF_banks
    - Feature Class: RestoredWetlands

Method:

## Create feature class to store results for HInvest and set initial value to zero

R-click WU_20150514 and select Export Data / All features
Output feature class: WetlandFunction.gdb / WU_HInvest

Open attribute table of WU_HInvest
Add field “HInvest” (short integer)
Field calculate HInvest = 0

## Low investment
## Select wetlands that intersect state or local public lands

Select by Location
Select features from: WU_HInvest
Source layer: stateParkBoundaries_WVDNR_20170927_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: wvdnrManagedLands_wvdnr_20170731_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: wvStateForestBoundaries_wvdof_20171003_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: countyCityParkBoundaries_20170731_utm83
Spatial selection method for target layer feature(s): intersect the source feature

## Add wetlands that intersect Natural Streams Preservation Act watersheds to selection

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: NatStrPreAct_HUC10
Spatial selection method for target layer feature(s): intersect the source feature

## Add wetlands that intersect Department of Defense Lands to selection

Open attribute table of WV_Protected_Lands_2015_PUBLIC
SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: "OwnName" = 'Department of Defense (DOD)'

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: WV_Protected_Lands_2015_PUBLIC
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Assign point and clear selections

Open attribute table of WU_HInvest
R-click HInvest and Field Calculate HInvest = 1

Clear all selections

## Moderate investment
## Select lands owned by USFS

Open attribute table of nationalForestOwnership_USFWS_20170803_utm83
SELECT * FROM nationalForestOwnership_USFWS_20170803_utm83 WHERE:
"Ownership" = 'Forest Service'

## Select wetlands that intersect selected areas

Select by Location
Select features from: WU_HInvest
Source layer: nationalForestOwnership_USFWS_20170803_utm83
Use Selected Features (1 Feature selected)
Spatial selection method for target layer feature(s): intersect the source feature

## Assign points and clear selections

Open attribute table of WU_HInvest
R-click HInvest and Field Calculate HInvest = 2

Clear all selections

## High investment
## Select Private Conservation Lands with Gap Status = 1 and USFS Wilderness Areas

Open attribute table of WV_Protected_Lands_2015_PUBLIC
SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: ("Mang_Name" IN ('EWPP-FPE', 'WRP', 'CLRLT', 'Potomac Conservancy', 'The Nature Conservancy (TNC)', 'WVLT', 'Forest Legacy') OR "GAP_Sts" = '1' OR "PdesTp" = 'Wilderness Area') AND ("PdesTp" <> 'Wild and Scenic River' AND "PdesTp" <> 'National Wildlife Refuge')

## Select wetlands that intersect selected areas

Select by Location
Select features from: WU_HInvest
Source layer: WV_Protected_Lands_2015_PUBLIC
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands that intersect National Wildlife Refuges or National Parks

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: nationalWildifeRefuge_USFWS_20170803
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: nationalParkBoundaries_nationalParkService_20170802
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands that intersect special botanical areas

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: botanical_areas_MNF.shp
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands that intersect Mitigation Banks and In-Lieu Fee sites

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: ILF_banks
Spatial selection method for target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 100 meters

## Add to Selection wetlands that intersect other restored, enhanced or created wetland sites

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: RestoredWetlands
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands in WVDNR State Natural Areas

SELECT * FROM stateParkBoundaries_WVDNR_20170927_utm83 WHERE: "Unit_Nm" IN ('Canaan Valley Resort State Park', 'Cathedral State Park', 'Beartown State Park')
(3 out of 36 selected)

## Hungry Beech State Natural Area is already selected in "Private Conservation Lands" above.
## There are currently (2017) no mapped wetlands in Hungry Beech.
## In case we ever need to generate State Natural Areas as a layer.
*## SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: "P_Des_Nm" = 'Hungry Beech'*

## Select wetlands that intersect selected areas

Select by Location
Add to the currently selected features in: WU_HInvest
Source layer: stateParkBoundaries_WVDNR_20170927_utm83
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Assign points and clear selections

Open attribute table of WU_HInvest
R-click HInvest and Field Calculate HInvest = 3

5.6.33 Histosol: Deep Organic Soils

Version date: 14 February 2017

Strategy: completed 4/13/2016 EAB
GIS method: completed 4/14/2016 EAB; verified 4/14/2016 EAB
Python code: Started & Completed 2/7/2017 MCA
Final review by EAB: 2/7/2017

Purpose:

Input to Habitat and Ecological Integrity / Intrinsic Potential

Description:

<u>Rationale</u>: Deep organic soils provide important habitat to specialist plants and animals, including bog and fen species. Wetlands with deep organic soils store large amounts of carbon. It is important to keep carbon locked in the soil where it will not contribute to anthropogenic climate change.

<u>Strategy:</u>
(Adapted from "Organic" variable – some code can be shared).
<u>Strategy</u>: Determine presence of histosols and/or histic epipedons. *Note that this criterion can be measured much more accurately during rapid field assessment. For GIS assessment, we will note only the presence, not the cover, of histosols and histic epipedons.*

> 3 points: Histosols occur in Wetland Unit (NWI polygons with "g" modifier OR SSURGO/Palustrine plots organic depth OR peatlands layer).
> 2 points: Histic epipedons occur in Wetland Unit (SSURGO/palustrine plots organic depth).
> 0 points: Histosols and histic epipedons are not present.

Select Wetland Units that intersect with any of the following: Peatlands, NWI polygons with organic modifier, Palustrine Plots with muck or peat soils, or SSURGO chorizon with organic soils in the upper 5 cm.
<u>SSURGO soils data</u> from NRCS has multiple non-spatial tables, which have one-to-many relationships with the ssurgo_wv table. We will access the component horizon table (chorizon_all) to extract the organic content, horizon, and top depth of the horizon.
<u>Note that SSURGO</u> mapping is very uneven, with some counties heavily mapped with organic soils and others with few or no organic soils mapped.

Definitions:

For the purposes of this assessment, histosols and histic epipedons are defined as follows, simplified from NRCS 2014 Keys to Soil Taxonomy.

> <u>Histosol</u>: Peat, mucky peat, or muck soil with at least 12-18% organic matter by weight and >= 40 cm deep within the upper 80 cm of soil profile.
> <u>Histic epipedon</u>: Peat, mucky peat, or muck soil with at least 12-18% organic matter by weight and >= 20 cm thick (but < 40 cm thick) as a surface horizon. Aquic conditions or artificial drainage is required.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
    - Feature Class: EnhWVWetland
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: Peatlands_20160228 (update if more recent file is available)
    - Feature Class: PalustrinePlotsMarch2015
- M:\basemap\ssurgo\ssurgo.gdb
    - Feature Class: ssurgo_wv
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb (see box below)
    - Feature Class: Histosol (created 4/14/2016)
    - Feature Class: HisticEpipedon (created 4/14/2016)

NOTE: ArcGIS related tables cannot be programmed in Python, so before this procedure is run, the input data layers “Histosol” and “HisticEpipedon” must be created in ArcGIS, as shown below. Note that these layers should be re-exported annually or as SSURGO is updated by NRCS. These 2 layers (Histosol, HisticEpipedon) were created by EAB on 4/14/2016.

## Open the related one-to-many SSURGO chorizon_all table.

Open attribute table for ssurgo_wv
Click “Related Tables” (second icon from left).
Click “component to surgo: component_all” to open the component_all table. Note that a tab will appear at the bottom of the attribute table showing the tables that are open.

Click “Related Tables” (second icon from left).
Click “component to chorizon: chorizon_all” to open the chorizon_all table.

## Select histisols: organic material > 15% by weight with a thickness of at least 40 cm in the upper 80 centimeters of the soil profile.

With the tab at the bottom of the attribute labelled “chorizon_all” highlighted:
Select by attributes
SELECT * FROM chorizon_all WHERE: ("hzname" LIKE 'O%' OR "om_r" > 15) AND "hzdept_r" < 40 AND "hzthk_r" > 39

## Relate this selection to the spatial data in the ssurgo_wv tab.

Click “Related Tables” again to get back to component_all and then back to ssurgo_wv
Note that the last step takes some time while all of the related tables open up.

## Export data to new feature class

R-click ssurgo_wv / Data / Export Data

Export Selected Features
Output feature class: SSurgoExports.gdb\Histosol

## Select histic epipedons: organic material > 15% by weight with a thickness of at least 20 cm in the upper 80 centimeters of the soil profile.

Clear selections.
With the tab at the bottom of the attribute labelled “chorizon_all” highlighted:
Select by Attributes.
SELECT * FROM chorizon_all WHERE: ("hzname" LIKE 'O%' OR "om_r" > 15) AND "hzdept_r" < 40 AND "hzthk_r" > 19 AND "hzthk_r" < 40

## Relate this selection to the spatial data in the ssurgo_wv tab.

Click “Related Tables” again to get back to component_all and then back to ssurgo_wv
Note that the last step takes some time while all of the related tables open up.

## Export data to new feature class

R-click ssurgo_wv / Data / Export Data
Export Selected Features
Output feature class: SSurgoExports.gdb\HisticEpipedon

Method:

## Add field for Histosol to Wetland Units attribute table and set initial value to zero.

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_Histosol
Add field Histosol (Short Integer) to WU_Histosol.
R-click Histosol and Field Calculate “Histosol” = 0

## PART 1: Histic Epipedon
## Select Palustrine plots that have peat or muck soils (conservative assumption that these are histic epipedons) or organic soils 20-39 cm thick.

SELECT * FROM PalustrinePlotsMarch2015 WHERE: "Soil_Textu" LIKE '%peat%' OR "Soil_Textu" LIKE '%muck%' OR "Profile__1" LIKE '%peat%' OR "Profile__1" LIKE '%muck%' OR ("DepOrgSoil" >19 AND "DepOrgSoil" < 40)

## Select Wetland Units that intersect palustrine plots with histic epipedons.

Select by Location
Selection method: select features from
Target layer: WU_Histosol
Source layer: PalustrinePlotsMarch2015

Check “Use selected features”
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on palustrine plots.

Open WU_Histosol attribute table
R-click “Histosol” and Field Calculate “Histosol” = 2
Clear all selections.

## Select Wetland Units that intersect with the SSURGO histic epipedon selection.

Select by Location
Selection method: select features from
Target layer(s): WU_Histosol
Source layer: HisticEpipedon
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on ssurgo data.

Open WU_Histosol attribute table.
R-click “Histosol” and Field Calculate “Histosol” = 2

## PART 2: Histosol

## Select Palustrine plots that have organic soils at least 40 cm thick.

SELECT * FROM PalustrinePlotsMarch2015 WHERE: "DepOrgSoil" >39

## Select Wetland Units that intersect palustrine plots with histosols.

Select by Location
Selection method: select features from
Target layer: WU_Histosol
Source layer: PalustrinePlotsMarch2015
Check “Use selected features”
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on palustrine plots.

Open WU_Histosol attribute table.
R-click “Histosol” and Field Calculate “Histosol” = 3
Clear all selections.

## Select Wetland Units that intersect with the SSURGO histosol selection.

Select by Location
Selection method: select features from
Target layer(s): WU_Histosol
Source layer: Histosol
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on ssurgo data.

Open WU_Histosol attribute table.
R-click “Histosol” and Field Calculate “Histosol” = 3

## PART 3: PEATLANDS
## Select Wetland Units that are peatlands.
Select by Location
Selection method: select features from
Target layer: WU_Histosol
Source layer: Peatlands_20160228 (or most recent version of Peatlands)
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on peatlands.

Open WU_Histosol attribute table (277 out of 43124 selected).
R-click “Histosol” and Field Calculate “Histosol” = 3
Clear all selections.

## PART 4: NWI ORGANIC MODIFIER
## Select polygons that have an organic modifier in the National Wetland Inventory.
SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE '%g'

## Select Wetland Units that intersect organic NWI polygons.

Select by Location
Selection method: select features from
Target layer: WU_HIstosol
Source layer: EnhWVWetland
Check “Use selected features”
Spatial selection method: intersect the source layer feature

## Update value for “Histosol” based on NWI.

Open WU_Histosol attribute table.
R-click “Histosol” and Field Calculate “Histosol” = “3”
Clear all selections.

5.6.34 HOpportunity: Habitat and Ecological Integrity Landscape Opportunity

Version date: 6 October 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/6/2017 EAB; results verified 10/6/2017 EAB
Python code: 10/10/2017 YH
Final review by EAB: 10/10/2017

Purpose:

Habitat Function
Maximum 13 points (all wetlands)

Description:

Rationale**:** The landscape around a wetland, including its perimeter, buffer, and the connectivity of the hydrologic and ecologic setting, are important influences on habitat value and ecological integrity of the wetland.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the "Opportunity" aspect.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunction.gdb
  - Feature Class: WU_BufferLand
    - Field: BufferLand
  - Feature Class: WU_LandHydro
    - Field: LandHydro
  - Feature Class: WU_LandEco
    - Field: LandEco

Method:

## Spatial joins to bring together factor values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_BufferLand
Join Feature: WU_LandHydro
Output feature class: WU_HOpportun1
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
BufferLand
LandHydro
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_HOpportun1
Join Feature: WU_LandEco
Output feature class: WU_HOpportun
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
BufferLand
LandHydro
LandEco
Match Option: CONTAINS

## Add HOpportun field and set initial point value to zero.

Open attribute table of WU_HOpportun
Add field “HOpportun” (short integer)
R-click HOpportun and Field Calculate HOpportun = 0

## Sum the factor points

R-click HOpportun and Field Calculate HOpportun = [BufferLand]+ [LandHydro]+ [LandEco]

5.6.35 HPotential: Habitat and Ecological Integrity Potential

Version date: 28 September 2017

Strategy: completed 3/16/2017 EAB
GIS method: 9/28/2017 EAB; results verified 9/28/2017 EAB
Python code: 9/28/2017 YH
Final review by EAB: 10/2/2017

Purpose:

Habitat Function
Maximum 30 points (all wetlands)

Description:

Rationale**:** Wetlands have an intrinsic potential to prove habitat for species, and wetlands benefit from high ecological integrity. This intrinsic capability is related to their vegetation, hydrology, soils, and physical structure.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the "Potential" aspect.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunction.gdb
    - Feature Class: WU_VegH
        - Field: VegH
    - Feature Class: WU_HydroH
        - Field: HydroH
    - Feature Class: WU_SoilH
        - Field: SoilH

Method:

## Spatial joins to bring together factor values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegH
Join Feature: WU_HydroH
Output feature class: WU_HPotential1
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
VegH
HydroH
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_HPotential1
Join Feature: WU_SoilH
Output feature class: WU_HPotential
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
VegH
HydroH
SoilH
Match Option: CONTAINS

## Add HPotential field and set initial point value to zero.

Open attribute table of WU_HPotential
Add field "HPotential" (short integer)
R-click HPotential and Field Calculate HPotential = 0

## Sum the factor points

R-click HPotential and Field Calculate HPotential = [VegH]+ [HydroH]+ [SoilH]

5.6.36 HSociety: Value to Society of Habitat and Ecological Integrity

Version date: 19 October 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/13/2017 EAB; results verified 10/19/2017 EAB; re-verified 11/6/2017 with
    new public lands layers for HInvest
Python code: 10/19/2017 YH
Final review by EAB: 10/19/2017

Purpose:

Habitat Function
Maximum 7 points (all wetland types)

Description:

<u>Rationale</u>**:** Societal investments in habitat and ecological integrity, along with public use and accessibility infrastructure, reflect the value to society of specific wetlands.

<u>Strategy</u>: Sum the points for HInvest and HUse.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunction.gdb
    - Feature Class: WU_HInvest
        - Field: HInvest
    - Feature Class: WU_HUse
        - Field: HUse

Method:

## Spatial join to bring together metric values
## Note that this could also be done with a "Join" on the field "WUKey"

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_HInvest
Join Feature: WU_HUse
Output feature class: WU_HSociety
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
HInvest
HUse
Match Option: CONTAINS

## Add HSociety field and set initial point value to zero.

Open attribute table of WU_HSociety
Add field “HSociety” (short integer)
R-click HSociety and Field Calculate HSociety = 0

## Sum the factor points

R-click HSociety and Field Calculate HSociety = [HInvest]+ [HUse]

5.6.37 HUC12WQ: Water quality issues in HUC12 watershed

Version date: 8 April 2019

Strategy: completed 3/16/2016 EAB
GIS method: completed and verified 3/17/2016 EAB
Python coding: started & completed 3/18/2016; EAB 4/8/2019 I'm not sure if the EPAOverlist layer was used for this metric. If yes, then please replace the ImpairedStreams and EPAOverlist shapefiles with the single updated ImpairedStreams, which combines both datasets.
Final review by EAB: 3/18/2016

Purpose:

Water Quality Function, Value to Society
Max 1 point

Description:

Rationale: Wetland Unit is in a basin or sub-basin where water quality is an issue in some aquatic resource (303d list, eutrophic lakes, problems with nuisance and toxic algae, karst). Wetlands can mitigate the impacts of pollution even if they do not discharge directly to a polluted body of water. Wetlands can remove nitrogen from groundwater as well as surface water. They can also trap airborne pollutants. Thus wetlands can provide an ecosystem service and value to our society in any basin and sub-basin that has pollution problems. The removal of pollutants by wetlands is judged to be more valuable in basins where other aquatic resources are already polluted or have problems with eutrophication. Any further degradation of these resources by destroying the wetland could result in irreparable damage to the ecosystem. Karst systems lack natural filtering capacity and are vulnerable to pollution wherever they occur in the state; therefore wetlands occurring in karst areas get this point whether or not degradation has been documented.
Summary of strategy: Select HUC12 (12-digit) watersheds that contain an impaired stream reach, algal lake, algal stream, lake with power boat use, or karst. Assign 1 point to Wetland Units that intersect with these watersheds.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\basemap\watersheds_12digit.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_Fishing_20Aug2015\PublicFishingLakesWVDNR20150820.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: AlgalStreams
    - Feature Class: AlgalLakes
- M:\wr\WTRSHD_BRANCH\303D_TMDL_IMPAIRED\WV2016_ImpairedStreams_24KNHD.shp
- M:\basemap\geology_shapefiles\type\geology-TYPE-limestone.shp
- M:\basemap\geology_shapefiles\type\geology-TYPE-dolostone.shp

Input Variables:

None

Method:

## Create feature class to store HUC12WQ variables

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_HUC12WQ

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_HUC12WQ
Add field "HUC12WQ" (short integer)
R-click HUC12WQ and Field Calculate HUC12WQ = 0

## Select lakes with power boat use

Select * FROM PublicFishingLakesWVDNR20150820 WHERE: "BoatType" NOT LIKE 'No%'
(86 out of 127 selected)

## Select HUC12s that contain lakes with power boat use.

Select by location
Selection method: select features from
Target layer: watersheds_12digit.shp
Source layer: PublicFishingLakesWVDNR20150820.shp
Check box "Use selected features" (86 features selected)
Spatial selection method: Intersect the source layer feature
(102 out of 1205 selected)

## Select HUC12s that contain algal lakes.

Select by location
Selection method: add to the currently selected features in
Target layer: watersheds_12digit.shp
Source layer: AlgalLakes
Spatial selection method: intersect the source layer feature
(107 out of 1205 selected)

## Select HUC12s that contain algal streams.

Select by location
Selection method: add to the currently selected features in
Target layer: watersheds_12digit.shp

Source layer: AlgalStreams
Spatial selection method: intersect the source layer feature
(138 out of 1205 selected)

## Select HUC12s that contain impaired stream reaches.

Select by location
Selection method: add to the currently selected features in
Target layer: watersheds_12digit.shp
Source layer: WV2016_ImpairedStreams_24KNHD.shp
Spatial selection method: intersect the source layer feature

## Select HUC12s that contain karst.

Select by location
Selection method: add to the currently selected features in
Target layer: watersheds_12digit.shp
Source layer: geology-TYPE-limestone.shp
Spatial selection method: intersect the source layer feature

Select by location
Selection method: add to the currently selected features in
Target layer: watersheds_12digit.shp
Source layer: geology-TYPE-dolostone.shp
Spatial selection method: intersect

## Select Wetland Units in HUC12s with water quality issues.

Select by location
Selection method: select features from
Target layer: WU_HUC12WQ
Source layer: watersheds_12digit.shp
Check box "Use selected features"
Spatial selection method: intersect the source feature

## Assign 1 point to Wetland Units in HUC12s with water quality issues.

R-click "HUC12WQ" in WU_HUC12WQ and Field Calculate HUC12WQ = 1

5.6.38 HUse: Public Use of Habitat and Ecological Integrity

Version date: 19 October 2017

Strategy: completed 3/16/2017 EAB
GIS method: 10/13/2017 EAB; results verified 10/19/2017 EAB
Python code: 10/18/2017 YH
Final review by EAB: 10/19/2017

Purpose:

Habitat Function / Value to Society
Maximum 4 points (all wetland types)

Description:

Rationale**:** Access, infrastructure, and habitat quality all impact public use of wetlands.

Strategy: Sum the points for OwnerAccess and PublicUse.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunction.gdb
  - Feature Class: WU_OwnerAccess
    - Field: OwnerAccess
  - Feature Class: WU_PublicUse
    - Field: PublicUse

Method:

## Spatial join to bring together metric values
## Note that this could also be done with a "Join" on the field "WUKey"

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_OwnerAccess
Join Feature: WU_PublicUse
Output feature class: WU_HUse
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
OwnerAccess
PublicUse
Match Option: CONTAINS

## Add HUse field and set initial point value to zero.

Open attribute table of WU_HUse
Add field “HUse” (short integer)
R-click HUse and Field Calculate HUse = 0

## Sum the factor points

R-click HUse and Field Calculate HUse = [OwnerAccess]+ [PublicUse]

5.6.39 HydIntact: Intactness of hydrologic regime

Version date: 24 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/24/2017 EAB; results verified 9/24/2017 EAB
Python coding: 9/25/2017 YH
Final review by EAB: 9/26/2017

Purpose:

Input to Habitat & Ecological Integrity / Potential
Max 6 points

Description:

Rationale: Natural hydrologic processes vary greatly among different types of wetlands. Peatlands rely on precipitation and very slow groundwater movement to create deep organic soils and unique plant communities. Seepage swamps rely primarily on groundwater movement. Floodplain wetlands receive water from overland flooding in addition to groundwater and precipitation. Rather than specifying a particular hydrologic regime, this metric is rated based on the dominance of natural hydrologic processes and deviations from natural conditions. Hydroperiod is the characteristic frequency and duration of inundation or saturation of a wetland during a typical year. In most wetlands, plant recruitment and maintenance are dependent on hydroperiod. The interactions of hydroperiod and topography are major determinants of the distribution and abundance of native wetland plants and animals (Mitsch and Gosselink 2000, National Research Council 2001, CWMW 2013).

Strategy: Note that this metric is best assessed in the field. The field value will overwrite the GIS value. For GIS assessment, award points for low score for disturbances and discharges to the wetland (inverse of WQOpportun-5 pts- based on Discharges, ImpairedIn, RoadRail, Disturb50m, DisturbWshd); add an additional point for wetlands with higher scores (2,3) for Landscape Integrity.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\WQOpportun.gdb
    - Feature Class: WU_WQOpportun
        - Field: WQOpportun
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\LandInteg.gdb
    - Feature Class: WU_LandInteg
        - Field: LandInteg

Method:

## Note that the Spatial Join could be replaced by a Join on WUKey if that is easier.

## Spatial join to merge WQOpportun and LandInteg into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_WQOpportun
Join Feature: WU_LandInteg
Output Feature Class: WU_HydroH
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
WQOpportun
LandInteg
Match option: CONTAINS

## Add HydIntact field to Wetland Units and set initial point value to zero.

Open attribute table of WU_HydroH
Add field "HydIntact" (short integer)
R-click HydIntact and Field Calculate HydIntact = 0

## Assign points to HydIntact

Open attribute table of WU_HydroH
SELECT * FROM WU_HydroH WHERE: ("WQOpportun" = 5 AND "LandInteg" IN (2,3))
OR ("WQOpportun" = 4 AND "LandInteg" IN (0,1))
Field Calculate HydIntact = 1

SELECT * FROM WU_HydroH WHERE: ("WQOpportun" = 4 AND "LandInteg" IN (2,3))
OR ("WQOpportun" = 3 AND "LandInteg" IN (0,1))
Field Calculate HydIntact = 2

SELECT * FROM WU_HydroH WHERE: ("WQOpportun" = 3 AND "LandInteg" IN (2,3))
OR ("WQOpportun" = 2 AND "LandInteg" IN (0,1))
Field Calculate HydIntact = 3

SELECT * FROM WU_HydroH WHERE: ("WQOpportun" = 2 AND "LandInteg" IN (2,3))
OR ("WQOpportun" = 1 AND "LandInteg" IN (0,1))
Field Calculate HydIntact = 4

SELECT * FROM WU_HydroH WHERE: ("WQOpportun" = 1 AND "LandInteg" IN (2,3))
OR ("WQOpportun" = 0 AND "LandInteg" IN (0,1))
Field Calculate HydIntact = 5

SELECT * FROM WU_HydroH WHERE: "WQOpportun" = 0 AND "LandInteg" IN (2,3)
Field Calculate HydIntact = 6

5.6.40 HydroH: Hydrologic regime for habitat

Version date: 27 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/27/2017 EAB; results verified 9/27/2017 EAB
Python coding: 9/27/2017 YH
Final review by EAB: 9/28/2017

Purpose:

Input to Habitat & Ecological Integrity / Potential
Max 9 points

Description:

Rationale: Hydrology is the most important direct determinant of wetland functions (Mitsch and Gosselink 2007); however, it is not easy to accurately assess hydroperiod remotely or during a single field visit (Stein et al 2009, Mack 2001). Wetland hydrology varies greatly under natural conditions, from rainfed bogs to groundwater wetlands to wetlands fed by overbank flooding. All of these natural hydrologic regimes create conditions under which wetland plants and animals can thrive and wetlands can perform their intrinsic ecological, hydrological, and societal functions and services (CWMW 2013). Disturbances to hydrology are one of the main sources of degradation to wetlands (Mack 2001).

Strategy: Sum the points for the metrics HydIntact, ConnectFL, and HydSW.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunction[date & time].gdb
  - Feature Class: WU_HydroH
    - Field: HydIntact
    - Field: HydSW
  - Feature Class: WU_ConnectFL
    - Field: ConnectFL

Method:

## Note that the step below could be done with a Spatial Join instead of a Join if that is easier to code.

## Add Join to merge fields into one attribute table

ArcToolbox / Data Management Tools / Joins / Add Join
Layer Name or Table View: WU_HydroH
Input Join Field: WUKey
Join Table: WU_ConnectFL
Output Join Field: WUKey

Check “Keep All Target Features”

## Add field to store ConnectFL

Open attribute table of WU_HydroH (Join is still active)
Add fields “ConnectFL1” (short integer)
Field Calculate ConnectFL1 = [WU_ConnectFL.ConnectFL]

## Remove Join

R-click “WU_HydroH” and select Joins and Relates / Remove All Joins

## Add field to store HydroH and set initial value to zero

Open attribute table of WU_HydroH (Join is gone)
Add Field “HydroH” (short integer)
R-click HydroH and Field Calculate HydroH = 0

## Assign points to HydroH

Open attribute table of WU_HydroH
Field Calculate HydroH = [HydIntact] + [HydSW] + [ConnectFL1]

5.6.41 HydSW: Available Surface Water

Version date: 24 Septmeber 2017

Strategy: completed 3/16/2017 EAB
GIS method: 9/24/2017 EAB; results verified 9/24/2017 EAB
Python code: 9/25/2017 YH
Final review by EAB: 9/26/2017

Purpose:

Input to Habitat & Ecological Integrity / Intrinsic Potential / Hydrology
Max 1 point.

Description:

Rationale: Seasonally or permanently available open water typically supports submerged macrophytes and provides important foraging and breeding habitat for birds, bats, and amphibians. The structural complexity provided by aquatic bed species increases habitat niches for a number of invertebrate and vertebrate species (Hruby 2012, Berglund and McEldowney 2008).
Summary of strategy: Select wetland polygons from the NWI that are attributed as palustrine aquatic bed, unconsolidated bottom, or unconsolidated shore AND are permanently flooded or intermittently exposed AND are not spoil. Also select Wetland Units that are contiguous to a non-impaired lake or stream. Assign points as follows:

- 1 point for PUB or PAB present OR NWI modifier [G,H] (Intermittently exposed/Permanently flooded) OR intersects/contiguous with non-impaired lake or stream (do not include impaired stream reaches or algal lakes/streams). Do not include polygons with the special modifier for spoil.
- 0 points: none of the above

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\WU_HydroH (this is created in HydIntact method)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Method:

## Select the wetland polygons that are attributed as open water, including lakes, rivers, and open water palustrine (aquatic bed, unconsolidated bottom, unconsolidated shore) AND have a hydrologic regime that is (permanently flooded, semipermanently flooded, intermittently exposed) AND are not spoil.

Clear all selections.
Select * FROM EnhWVWetland WHERE: ("ATTRIBUTE" LIKE 'L%' OR"ATTRIBUTE" LIKE 'R%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PUB%' OR "ATTRIBUTE" LIKE 'PUS%') AND ("ATTRIBUTE" LIKE '%H%' OR "ATTRIBUTE" LIKE '%G%' OR "ATTRIBUTE" LIKE '%F%') AND "ATTRIBUTE" NOT LIKE '%s%'

## Create open water layer from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\NWIExports.gdb
    Feature Class: NWIOpenWater

## Add field to store HydSW and set initial value to zero

In WU_HydroH, add field: HydSW (short integer)
R-click HydSW and Field Calculate HydSW = 0

## Select the Wetland Units that intersect or touch NWIOpenWater

Select By Location
Selection Method: select features from
Target Layer: WU_HydroH
Source layer: NWIOpenWater
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign points to HydSW

Open attribute table of WU_HydroH
R-click HydSW, Field Calculate HydSW = 1
Clear all selections

5.6.42 ImpairedIn: Impaired waters impacting wetland

Version date: 8 April 2019

Strategy: completed 3/12/2016 EAB
GIS method: 4/18/2016 EAB; verified 4/18/2016 EAB; 2017 update included 2/22/2017 EAB
Python coding: started 6/9/2016 MCA, completed 6/10/2016 MCA; EAB 2/15/2017 see update to impaired streams layer below – please change code to use these 2 new layers. EAB 4/8/2019 please replace the ImpairedStreams and EPAOverlist shapefiles with the single updated ImpairedStreams, which combines both datasets.
Final review by EAB: 10/3/2016; need to review after 2017 update is completed

Purpose:

Water Quality Function, Opportunity Aspect
Max 2 points

Description:

Impaired waters, algal blooms, or powerboat use impacting wetland (2 points max).
Rationale: Impaired waters are adjacent to wetland or are in the contributing watershed of a floodplain wetland. Impaired waters include the following:

- Power boat use. The presence of power boats in adjacent reservoirs or water bodies will increase the pollutants entering a fringe wetland. Toxic chemicals, oils, cleaners, and paint scrapings from boat maintenance can make their way into the water (Asplund 2000). In addition, older two stroke engines still found on many recreational boats and jet skis were purposely designed to discharge the exhaust that contains unburned gasoline and oil into the water. The landscape potential to improve water quality for a wetland unit along a shore is higher if the water body itself is directly receiving pollutants from power boats.
- Algal blooms and blooms or larger plants such as milfoil in open water are an indication of excessive nutrients (Schindler and Fee 1974, Smith et al. 1999). The increased levels of nutrients in the water body increase the amount of nutrients that the wetland plants absorb (Venterink and others 2002) and thus also increase the level of function within the wetland unit.
- Impaired streams.

Summary of Strategy: Select floodplain Wetland Units whose contributing watershed contains impaired waters, algal blooms, or powerboat use (1 point). Select all (floodplain and groundwater) Wetland Units within 5 meters' distance of impaired waters, algal blooms, or powerboat use (2 points).

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_Fishing_20Aug2015\PublicFishingLakesWVDNR20150820.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: AlgalStreams
  - Feature Class: AlgalLakes
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb

    - Feature Class: DrainageArea27m
- M:\wr\WTRSHD_BRANCH\303D_TMDL_IMPAIRED\WV2016_ImpairedStreams_24KNHD.shp

Input Variables:

- Floodplain

Method:

## Create feature class to store ImpairedIn variables

R-click WU_Floodplain and select Data/Export Data
Output feature class: WU_ImpairedIn

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_ImpairedIn
Add field “ImpairedIn” (short integer)
R-click ImpairedIn and Field Calculate ImpairedIn = 0

## Add field to DrainageArea27m and set initial value to “No”

Open attribute table of DrainageArea27m
Add field “ImpairSrc” (text, 2 characters)
R-click ImpairSrc and Field Calculate ImpairSrc = “N”

## Select lakes with power boat use

Select * FROM PublicFishingLakesWVDNR20150820 WHERE: "BoatType" NOT LIKE 'No%'

## Select Drainage Areas with power boat use.

Select by Location
Selection method: select features from
Target layers: DrainageArea27m
Source layer: PublicFishingLakesWVDNR20150820
Check box “Use Selected Features”
Spatial selection method: intersect the source layer feature

## Select (add) Drainage Areas with algal lakes.

Select by Location
Selection method: add to the currently selected features in
Target layers: DrainageArea27m
Source layer: AlgalLakes

Spatial selection method: intersect the source layer feature

## Select (add) Drainage Areas with algal streams.

Select by Location
Selection method: add to the currently selected features in
Target layers: DrainageArea27m
Source layer: AlgalStreams
Spatial selection method: intersect the source layer feature

## Select (add) Drainage Areas with impaired streams.

Select by Location
Selection method: add to the currently selected features in
Target layers: DrainageArea27m
Source layer: WV2016_ImpairedStreams_24KNHD
Spatial selection method: intersect the source layer feature

## Set ImpairSrc to yes for Drainage Areas with an impaired water source.

In DrainageArea27m feature class, Field calculate ImpairSrc = “Y”
Clear selections.

## Join Drainage Area attribute to Wetland Units.

ArcToolbox / Data Management Tools / Joins / Join Field
Input table: WU_ImpairedIn
Input Join Field: OBJECTID_1
Join Table: DrainageArea27m
Output Join Field: WUKey
Join Fields: WUKey, ImpairSrc

## Assign 1 point to floodplain Wetland Units with impaired waters in their Drainage Area.

SELECT * FROM WU_ImpairedIn WHERE: "Floodplain" = 'Y' AND "ImpairSrc" = 'Y'
R-click “ImpairedIn” in WU_ImpairedIn and Field Calculate ImpairedIn = 1
Clear selections.

## Select lakes with power boat use

Select * FROM PublicFishingLakesWVDNR20150820 WHERE: "BoatType" NOT LIKE 'No%'

## Select Wetland Units adjacent to lakes with power boat use

Select by location
Selection method: select features from
Target layers: WU_ImpairedIn
Source layer: PublicFishingLakesWVDNR20150820
Check box “Use selected features”
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

## Select Wetland Units adjacent to algal lakes

Select by location
Selection method: add to the currently selected features in
Target layers: WU_ImpairedIn
Source layer: AlgalLakes
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

## Select Wetland Units adjacent to algal streams

Select by location
Selection method: add to the currently selected features in
Target layers: WU_ImpairedIn
Source layer: AlgalStreams
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

## Select Wetland Units adjacent to impaired streams

Select by location
Selection method: add to the currently selected features in
Target layers: WU_ImpairedIn
Source layer: WV2016_ImpairedStreams_24KNHD
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

## Assign 2 points to Wetland Units adjacent to impaired waters.

R-click “ImpairedIn” in WU_ImpairedIn and Field Calculate ImpairedIn = 2

5.6.43 ImpairedOut: Wetland Discharges to Impaired Waters

Version date: 8 April 2019

Strategy: completed 3/12/2016 EAB
GIS method: completed and verified 3/17/2016 EAB
Python coding: started 3/16/2016 MCA, completed 3/18/2016 MCA; EAB 2/15/2017 please
replace old "impaired streams" layer with 2 new ones, as highlighted below
Changes finished 3/13/2017 MCA; EAB 4/8/2019 please replace the ImpairedStreams and EPAOverlist shapefiles with the single updated ImpairedStreams, which combines both datasets.
Final check by EAB: 3/18/2016

Purpose:

Water Quality Function
Max 1 point

Description:

Rationale: Wetland discharges to (<1 km above) a stream, river, or lake that is on the 303d list, or a water body that is impacted by chronic algal blooms or power boat use. The term, "303(d) list," is short for the list of impaired waters (stream segments, lakes) that the Clean Water Act requires all states to submit to the Environmental Protection Agency (EPA) every two years. Wetlands that discharge directly to these polluted waters are judged to be more valuable than those that discharge to unpolluted bodies of water because their role at cleaning up the pollution is critical for reducing further degradation of water quality. The WVDEP list is at:
http://www.dep.wv.gov/WWE/watershed/IR/Pages/303d_305b.aspx
Karst systems lack natural filtering capacity and are vulnerable to pollution wherever they occur in the state; therefore wetlands discharging to karst areas get this point whether or not degradation has been documented.
Summary of strategy: Select Wetland Units within 1 km of an impaired reach, algal lake, algal stream, or lake with power boat use AND Wetland Units that occur on karst. This is a coarse approximation (since it does not follow flowlines) that selects slightly over half of the state's wetlands.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_Fishing_20Aug2015\PublicFishingLakesWVDNR20150820.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: AlgalStreams
    - Feature Class: AlgalLakes
- M:\wr\WTRSHD_BRANCH\303D_TMDL_IMPAIRED\WV2016_ImpairedStreams_24KNHD.shp
- M:\basemap\geology_shapefiles\type\geology-TYPE-limestone.shp
- M:\basemap\geology_shapefiles\type\geology-TYPE-dolostone.shp

Input Variables:

None

Method:

## Create feature class to store ImpairedOut variables

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_ImpairedOut

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_ImpairedOut
Add field “ImpairedOut” (short integer)
R-click ImpairedOut and Field Calculate ImpairedOut = 0

## Select lakes with power boat use

Select * FROM PublicFishingLakesWVDNR20150820 WHERE: "BoatType" NOT LIKE 'No%'

## Select Wetland Units < 1 km from lakes with power boat use.

Select by location
Selection method: select features from
Target layer: WU_ImpairedOut
Source layer: PublicFishingLakesWVDNR20150820.shp
Check box “Use selected features”
Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Select Wetland Units < 1 km from algal lakes.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_ImpairedOut
Source layer: AlgalLakes
Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Select Wetland Units < 1 km from algal streams.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_ImpairedOut
Source layer: AlgalStreams

Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Select Wetland Units < 1 km from impaired stream reaches.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_ImpairedOut
Source layer: WV2016_ImpairedStreams_24KNHD
Spatial selection method: are within a distance of the source feature
Apply a search distance: 1000 meters

## Select Wetland Units on karst.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_ImpairedOut
Source layer: geology-TYPE-limestone.shp
Spatial selection method: intersect the source layer feature
(27668 out of 43124 selected)

Select by location
Selection method: add to the currently selected features in
Target layer: WU_ImpairedOut
Source layer: geology-TYPE-dolostone.shp
Spatial selection method: intersect
(27683 out of 43124 selected)

## Assign 1 point to Wetland Units that discharge to impaired waters.

R-click "ImpairedOut" in WU_ImpairedOut and Field Calculate ImpairedOut = 1

5.6.44 IrrEdge: Irregularity of the Upland/Wetland Edge

Version date: 8 March 2016

Strategy: Completed 2/17/2016 EAB
GIS method: Completed 3/8/2016 EAB JCC, Verified 3/8/2016 EAB
Python code: Completed 3/14/2016 MCA
Final result verified by EAB: 3/14/2016

Purpose:

Input to Water Quality/Surface Depressions
Max 1 point.
Nitrite removal is aided by upland/wetland contact (Adamus et al. 2010).
Note that GIS determination of surface depressions and the detailed perimeter (edge) of the wetland is not currently within our analysis reach. We are instead using the proxy of mapped perimeter/sqrt(area) to estimate this value. The perimeter will be much more accurately measured during rapid field assessment.
Summary of strategy: Calculate the perimeter of wetland unit that is NOT adjacent to open water divided by the square root of the area of wetland unit. Add 1 point if > 6.

Definitions:

N/A

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Input Variables:

- Shape_Length (perimeter of Wetland Unit in meters)
- Shape_Area (Area of Wetland Unit in m^2)

Method:

## Add field to the Wetland Units feature class.

Add numeric field to WU_20150514 attribute table: IrrEdgeRat (irregular edge ratio, float)

## Calculate initial value of IrrEdgeRat for all Wetland Units.
R-click IrrEdgeRat and Field Calculate = [Shape_Length] / ([Shape_Area] ^ 0.5)

## Select the rivers and lakes from the National Wetlands Inventory.
Clear all selections.
Select * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'R%' OR "ATTRIBUTE" LIKE 'L%'

## Export the rivers and lakes.

R-click EnhWVWetland
Data / Export Data
Export: Selected features
Output feature class: RiversLakes

## Select the Wetland Units that border a river or lake.

Select by Location
Selection method: select features from
Target layer: WU20150514
Source layer: RiversLakes
Spatial selection method: intersect the source layer feature

## Export the intersecting Wetland Units

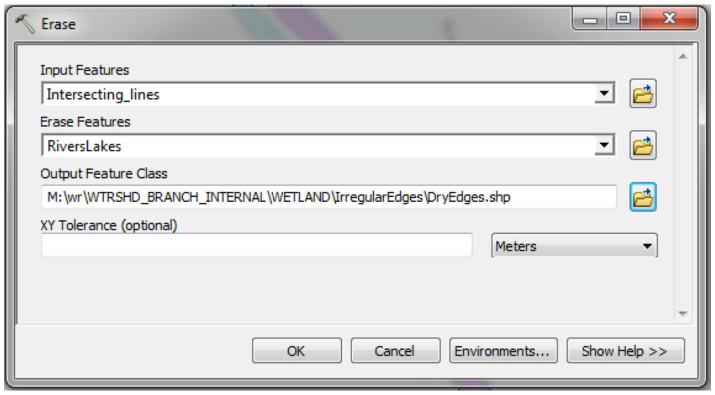
R-click WU_20150514
Data / Export Data
Export: Selected features
Output feature class: WURiversLakes

## Convert Wetland polygons to lines

Data Management Tools / Features / Polygon to Line
Input Features: WU_RiversLakes
Output Feature Class: Intersecting_lines
Do not check box "Identify and store polygon…"

## Erase the wet perimeter lines

Analysis Tools / Overlay / Erase
Input features: Intersecting_lines
Erase features: RiversLakes
Output feature class: DryEdges

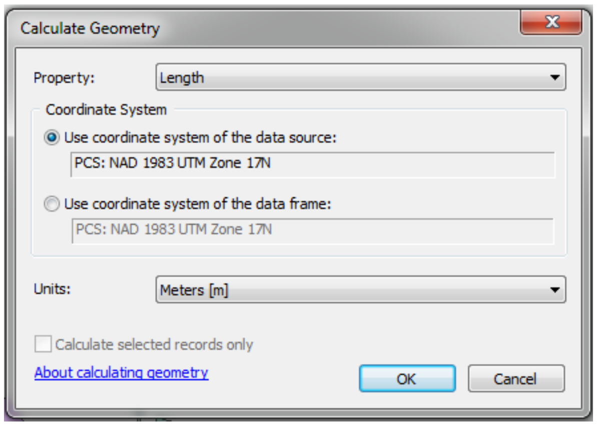

Erase

Input Features
Intersecting_lines
Erase Features
RiversLakes
Output Feature Class
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\IrregularEdges\DryEdges.shp
XY Tolerance (optional)
Meters

## Add field and calculate dry perimeter in DryEdges

In DryEdges, add field “DryPerim” (float)

Right click DryPerim attribute field and Field Calculate “DryPerim” = [Shape_Length]



Calculate Geometry

Property: Length

Coordinate System

(●) Use coordinate system of the data source:
PCS: NAD 1983 UTM Zone 17N

( ) Use coordinate system of the data frame:
PCS: NAD 1983 UTM Zone 17N

Units: Meters [m]

☐ Calculate selected records only

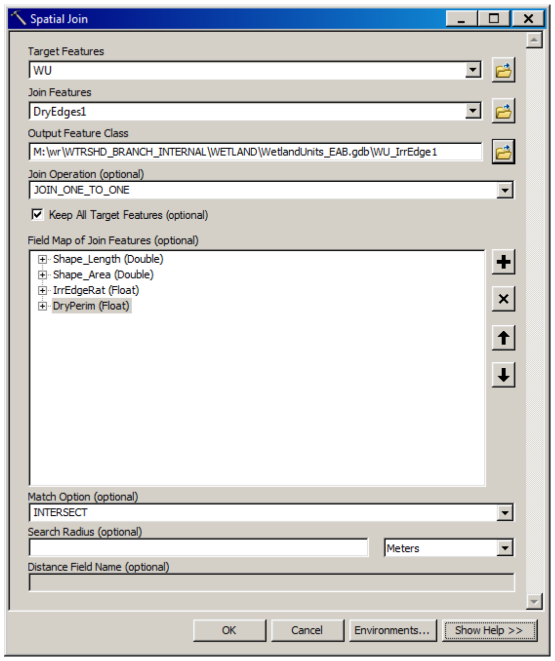
## Reset to NULL records from DryEdges with length < 26. These are mapping or computational errors.

Clear all selections.
SELECT * FROM DryEdges WHERE: “DryPerim” < 26
R-click DryPerim and Field Calculate “DryPerim” = NULL

## Join Wetland Units to DryEdges

Clear all selections.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: DryEdges
Output Feature Class: WU_IrrEdge
Join operation: Join_one_to_one
Check box “Keep all target features”
Field map of join features: (retain the following)
Shape_Length
Shape_Area
IrrEdgeRat
DryPerim
Match Option: Intersect



Spatial Join

Target Features: WU

Join Features: DryEdges1

Output Feature Class: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits_EAB.gdb\WU_IrrEdge1

Join Operation (optional): JOIN_ONE_TO_ONE

☑ Keep All Target Features (optional)

Field Map of Join Features (optional)
Shape_Length (Double)
Shape_Area (Double)
IrrEdgeRat (Float)
DryPerim (Float)

Match Option (optional): INTERSECT

Search Radius (optional): Meters

Distance Field Name (optional)

## Select the Wetland Units fronting rivers and lakes

SELECT * FROM WU_IrrEdge WHERE: “DryPerim” > 0

## Update the value of IrrEdgeRat for the selected records.

Open attribute table of WU_IrrEdge
R-click IrrEdgeRat and Field Calculate: [DryPerim] / ([Shape_Area] ^ 0.5)
Clear all Selections.

## Add IrrEdge point field and set initial value to zero.

In WU_IrrEdge, add field IrrEdge (short integer)
Field Calculate IrrEdge = 0

## Assign 1 point if IrrEdgeRat > 6.

SELECT * FROM WU_IrrEdge WHERE: IrrEdgeRat > 6
Field Calculate IrrEdge = 1

5.6.45 Karst: limestone/dolomite bedrock or calcareous soil

Version date: 18 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/15/2017 EAB; results verified 9/18/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Potential / Vegetation / Floristic Quality
Max 3 points

Description:

Rationale: A rich and distinctive flora and fauna are characteristic of calcareous wetlands.

Summary of strategy: Karst area (limestone/dolomite bedrock geology or SSURGO karst/calcareous soils) > 0.67 of total wetland area = 3 points; karst 0.33-0.67 = 2 points; > 0.1 = 1 point

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: KarstComposite

Method:

## Intersect karst and Wetland Units

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: KarstComposite
WU_20150514
Output feature class: WU_Karst1
Join attributes: ONLY_FID
Output type: INPUT

## Add field to store karst area.

Open attribute table of WU_Karst1
Add field "KarstArea" (float)
R-click "Karst" and Field Calculate: KarstArea = [Shape_Area]

## Spatial Join karst selection to Wetland Units and sum karst area.

Analysis Tools / Overlay / Spatial Join

Target Features: WU_20150514
Join Features: WU_Karst1
Output Feature Class: WU_Karst
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following
Shape_Length
Shape_Area
WUKey
    KarstArea (R-click and select “Merge Rule”, “Sum”)
Match Option: CONTAINS

Note: if the Spatial Join doesn’t work because of difficulties opening the “WU_Karst1” feature class, then export “WU_Karst1” to a shapefile and re-run the Spatial Join.

## Add fields to store KarstRatio and Karst.

Open attribute table of WU_Karst
Add field KarstRatio (float) to WU_Karst attribute table.
Add field Karst (short integer) to WU_Karst attribute table.
Field calculate Karst = 0

## Calculate the ratio of karst area to total Wetland Unit area.

Field calculate KarstRatio = [KarstArea] / [Shape_Area]

## Assign points

SELECT * FROM WU_Karst WHERE: KarstRatio > 0.1
Field Calculate Karst = 1

SELECT * FROM WU_Karst WHERE: KarstRatio > 0.33
Field Calculate Karst = 2

SELECT * FROM WU_Karst WHERE: KarstRatio > 0.67
Field Calculate Karst = 3

5.6.46 LandEco: Landscape-level Ecological Connectivity

Version date: 6 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/6/2017 EAB; results verified 10/6/2017 EAB
Python code: 10/10/2017 YH
Final review by EAB: 10/10/2017

Purpose:

Input to Habitat / Landscape Opportunity

Description:

Maximum 3 points

<u>Rationale</u>: Landscape-level ecological connectivity provides high opportunities for maintenance and dispersal of native species, rare species, and natural communities.

<u>GIS Method (no field method):</u>
Sum the values of the metrics below.

BRankHUC (4 max)
WshdUniq (2 max)
ConsFocus (2 max)
WetldBird (3 max)

Assign points as follows:

3 points: Sum of metrics = 8-11
2 points: Sum of metrics = 5-7
1 point: Sum of metrics = 2-4
0 points: Sum of metrics = 0-1

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_BRankHUC
        - Field: BRankHuc
    - Feature Class: WU_WshdUniq
        - Field: WshdUniq
    - Feature Class: WU_ConsFocus
        - Field: ConsFocus
    - Feature Class: WU_WetlandBird
        - Field: WetldBird

Method:

## Spatial Join to merge attributes into one table
## Note that this could also be done with a Join on WUKey if that is easier

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_BRankHUC
Join Feature: WU_WshdUniq
Output Feature Class: WU_LandEco1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:

 WUKey
 Shape_Length
 Shape_Area
 BRankHUC
 WshdUniq

Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_LandEco1
Join Feature: WU_ConsFocus
Output Feature Class: WU_LandEco2
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:

 WUKey
 Shape_Length
 Shape_Area
 BRankHUC
 WshdUniq
 ConsFocus

Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_LandEco2
Join Feature: WU_WetlandBird
Output Feature Class: WU_LandEco
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:

 WUKey
 Shape_Length
 Shape_Area
 BRankHUC
 WshdUniq
 ConsFocus
 WetldBird

Match option: CONTAINS

## Create feature class to store results for LandEco and set initial value to zero

Open attribute table of WU_LandEco
Add field “LandEco” (short integer)
Field calculate LandEco = 0

## Sum the metrics and assign points:

Open attribute table of WU_LandEco
SELECT * FROM WU_LandEco WHERE: ("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 1

Field Calculate LandEco = 1

SELECT * FROM WU_LandEco WHERE: ("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 4

Field Calculate LandEco = 2

SELECT * FROM WU_LandEco WHERE: ("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 7

Field Calculate LandEco = 3

Clear all selections

5.6.47 LandHydro: Landscape-level Hydrologic Connectivity

Version date: 3 October 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 10/3/2017 EAB; results verified 10/3/2017 EAB
Python coding:
Final review by EAB:

Purpose:

Input to Habitat & Ecological Integrity / Landscape Opportunity
Max 3 points
Rationale: Landscape-level hydrologic conductivity is a key component of ecological integrity. See WshdPos and AquaAbund metrics for additional information on rationale.
Strategy: Sum the points for WshdPos and AquaAbund.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_WshdPos
    - Field: WshdPos
  - Feature Class: WU_AquaAbound
    - Field: AquaAbound

Method:

Note that the Spatial Joins could be replaced by Joins on WUKey if that is easier.

## Spatial join to merge WshdPos and AquaAbund into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_WshdPos
Join Feature: WU_AquaAbund
Output Feature Class: WU_LandHydro
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
WshdPos
AquaAbund
Match option: CONTAINS

## Add LandHydro field to Wetland Units and set initial point value to zero.

Open attribute table of WU_LandHydro
Add field "LandHydro" (short integer)

R-click LandHydro and Field Calculate LandHydro = 0

## Sum the points for WshdPos and AquaAbund

R-click LandHydro and Field Calculate “LandHydro” = [WshdPos] + [AquaAbund]

5.6.48 LandInteg: Landscape Integrity Index

Version date: 3 December 2018

Strategy: 3/16/2017 EAB
GIS method: 9/19/2017 EAB; revised 9/22/2017 EAB; revised to include IEI (UMass) data 11/27/2018 EAB
Python code: 9/25/2017 YH
Final review by EAB: 9/26/2017

Purpose:

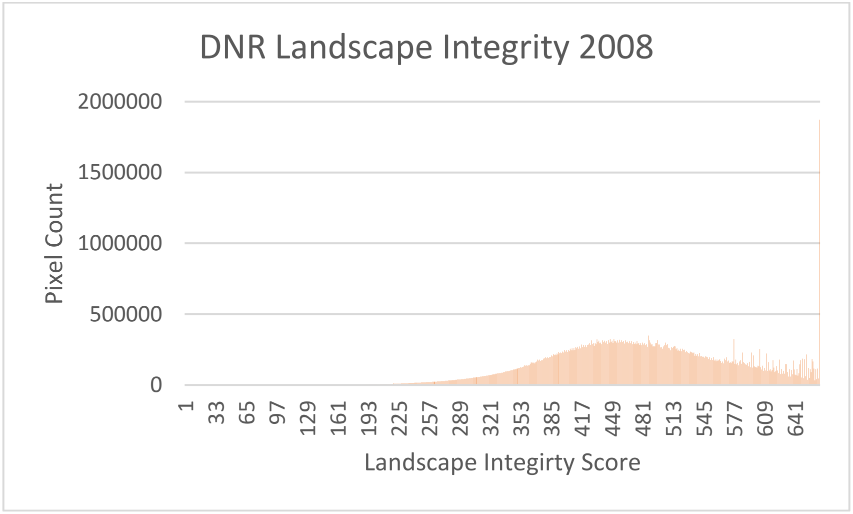
Input to Habitat / Potential (VegFQ, HydIntact) and Habitat/Opportunity (Buffer and Landscape Integrity)

Description:

Maximum 3 points
Rationale: Landscape Integrity is a good measure of overall habitat value, including a positive correlation with Floristic Quality.
Strategy:
Sum the following four weighted measures of landscape integrity. Then calculate a weighted average for a maximum of 3 points.

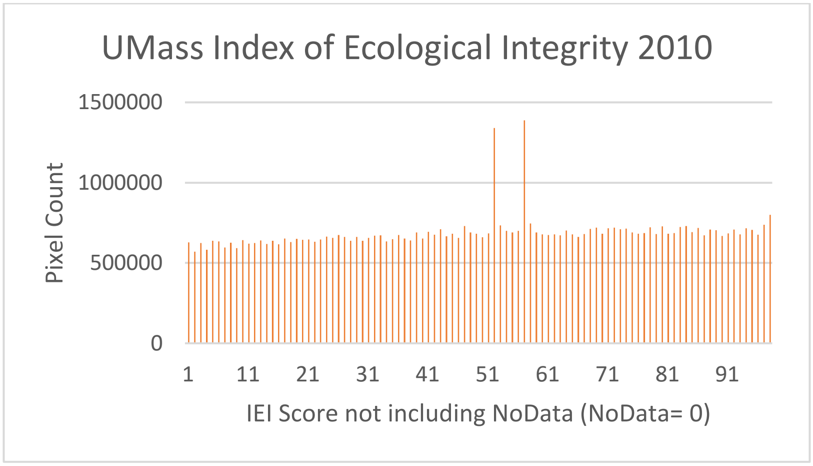
LandIntegDNR: DNR 2008 Landscape Integrity model: mean value for wetland. LandIntegDNR is calculated as the mean value of pixels within a Wetland Unit. For Wetland Units that are smaller or narrower than a single pixel (30m x 30m), LandInteg is calculated from the (contained) centroid of the polygon. Point thresholds were set by E. Byers after comparing a large number of known landscapes to point distributions.

3 points: >= 800
2 points: 700-800
1 point: 600-700
0 points: < 600



IEI: Index of Ecological Integrity from the Conservation and Prioritization System (UMass 2010 with some 2018 updates). This layer is very similar to the LandIntegDNR layer. It has finer resolution and does a better job of capturing roads and other detailed features. However, its treatment of pipelines and powerlines as higher-integrity areas than surrounding wetlands is problematic.
IEI is calculated as the mean value of pixels within a Wetland Unit. For Wetland Units that are smaller or narrower than a single pixel (30m x 30m), IEI is calculated from the (contained) centroid of the polygon. Pixels with "NoData" are all in highly developed areas and we therefore set these pixels equal to 0. Point thresholds were set to approximately mimic the proportions of land in each category of the LandIntegDNR layer, with slightly more land in the lower categories since this layer has finer resolution and picks up fragmentation better (except for the pipelines).

3 points: 70
2 points: 45 – 70
1 point: 15 – 45
0 points: < 15



LandResil: TNC 2016 Resilient and Connected Landscapes model: most common value for wetland. Note that this layer has a small percentage of null pixel values.
LandResil is calculated as the most common value of pixels within a Wetland Unit. For Wetland Units that are smaller or narrower than a single pixel (30m x 30m), LandResil is calculated from the (contained) centroid of the polygon.

3 points: wetland intersects resilient land with confirmed diversity: Value IN (11,12,112)
2 points: wetland intersects resilient land with connectivity: Value IN (2,4,11,12,13,14,33,112)
1 point: wetland intersects resilient land: Value IN (2,3,4,11,12,13,14,33,112)
0 points: wetland does not meet above criteria

Definition of Value codes:
0 Vulnerable
2 Climate Corridor (resilient)
3 Resilient only (unsecured)
4 Climate Corridor (vulnerable)

11 Climate Corridor with confirmed diversity
12 Resilient Area with confirmed diversity
13 Climate Corridor
14 Climate Flow Zone
33 Resilient only (secured)
112 Climate Flow Zone with confirmed diversity

ForestPatch: Forest tract proximity and extent based on TNC 2014 forest patches mapping. Note that this layer has a few small gaps, e.g. no forest patches mapped north of Cranberry Glades.

3 points: wetland intersects forest patch ≥ 1000 ha (2470 acres) in size
2 points: wetland intersects forest patch ≥ 100 ha (247 acres) in size (i.e., is contiguous with a forest patch ≥ 100 ha in size OR wetland itself contains ≥ 100 ha of forest)
1 point: wetland intersects a forest patch ≥ 20 ha (50 acres) in size OR wetland is within 30 m of a forest patch ≥ 100 ha in size
0 points: wetland does not meet above criteria

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
    - Feature Class: WU_20150514
    - Feature Class: WUpoint
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\landscapeIntegrityIndex_WVDNR_2008_utm83_img\
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\ContributedRawData\Resilient_and_Connected_Landscapes\Resilient_and_Connected_Data.gdb
    - Raster: Resilient_and_Connected
        - Field: Value
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\ContributedRawData\forest_patches_over50acres_WVplus10mi.shp
    - Field: Acreage
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb
    - Raster: IEIUMa2010v32

Method:

## STEP 1
## Calculate the intermediate metric LandIntegDNR
## Summarize the DNR Landscape Integrity raster values for each Wetland Unit

Spatial Analyst Tools/Zonal/Zonal Statistics as Table
Input feature zone data: WU_20150514
Zone field: WUKey
Input Value raster: landscapeIntegrityIndex_WVDNR_2008_utm83.img
Output table: LandInteg_zonal
Check Ignore No Data

Statistics Type: MEAN

## Note the null returns are for polygons that are too small to contain an entire pixel. For these polygons, we need to calculate the centroid of the (very small) wetland, and use the centroid to obtain the Landscape Integrity Index value.

## Convert Wetland Unit polygons to points
## Note that this step should be moved to "Creating Wetland Units from NWI Polygons"

Data Management Tools / Features / Feature to Point
Input Features: WU_20150514
Output Feature Class:
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\WUpoint
Check "Inside"

## Extract the DNR Landscape Integrity raster values for each Wetland Unit centroid

Spatial Analyst Tools / Extraction / Extract Values to Points
Input point features: WUpoint
Input raster: landscapeIntegrityIndex_WVDNR_2008_utm83.img
Output point features: WUpoint_LandInteg

## Delete the "Count" field from the LandInteg_zonal table since "Count" is a restricted word
## and may interfere with the join in the next step

R-click table "LandInteg_zonal" to open the table.
R-click the "Count" field and delete field.

## Join Wetland Units to the LandInteg_zonal table

Right-click WU_20151514 in TOC
Click Joins and Related
Click "Join…"

This will display the "Join Data pop-up window"
Select "Join attributes from a table"
"Choose the field…" = WUKey
"Choose the table to join…" = LandInteg_zonal
"Choose the field in the table…" = WUKey
Check "Keep all records"

## Export the joined data to a Feature Class

Conversion Tools/To Geodatabase/Feature Class to FeatureClass
Input Features: WU_20150514

Output Feature Class: WU_LandInteg1

## Remove Join from Wetland Units

R-click WU_20150514 / Joins and Relates / Remove Joins / Remove all joins

## Join WU_LandInteg1 to WUpoint_LandInteg

Right-click WU_LandInteg1 in TOC
Click Joins and Related
Click "Join…"

This will display the "Join Data pop-up window"
Select "Join attributes from a table"
"Choose the field…" = WUKey (choose the first listing of WUKey)
"Choose the table to join…" = WUpoint_LandInteg
"Choose the field in the table…" = WUKey
Check "Keep all records"

## Export joined data to feature class

Conversion Tools/To Geodatabase/Feature Class to FeatureClass
Input Features: WU_LandInteg1
Output Feature Class: WU_LandIntegDNR

## Replace NULL values of MEAN with centroid values of Landscape Integrity

Open attribute table of WU_LandIntegDNR
Select by attributes
SELECT * FROM WU_LandIntegDNR WHERE: "MEAN" IS NULL
R-click MEAN and Field Calculate MEAN = [RASTERVALU]
Clear Selection

## Add field LandIntegDNR, set initial value to zero

Open the attribute table to WU_LandInteg
Add field "LandIntegDNR" (short integer) to attribute table
Field Calculate "LandIntegDNR" = 0

## Assign points

SELECT * FROM WU_LandInteg WHERE: MEAN > 600
Field Calculate LandIntegDNR = 1

SELECT * FROM WU_LandInteg WHERE: MEAN > 700
Field Calculate LandIntegDNR = 2

SELECT * FROM WU_LandInteg WHERE: MEAN > 800
Field Calculate LandIntegDNR = 3

## STEP 2
## Note that method in nearly identical to STEP 1 except for field names and final points
## Changes from STEP 1 are noted in green highlight.
## Calculate the intermediate metric IEI
## Summarize the IEI raster values for each Wetland Unit

Spatial Analyst Tools/Zonal/Zonal Statistics as Table
Input feature zone data: WU_20150514
Zone field: WUKey
Input Value raster: IEIUMa2010v32
Output table: IEI_zonal
Check Ignore No Data
Statistics Type: MEAN

## The null returns are for polygons that are too small to contain an entire pixel. For these polygons, we need to calculate the centroid of the (very small) wetland, and use the centroid to obtain the IEI value.

## Extract the IEI raster values for each Wetland Unit centroid

Spatial Analyst Tools / Extraction / Extract Values to Points
Input point features: WUpoint
Input raster: IEIUMa2010v32
Output point features: WUpoint_IEI

## Delete the "Count" field from the IEI_zonal table since "Count" is a restricted word
## and may interfere with the join in the next step

R-click table "IEI_zonal" to open the table.
R-click the "Count" field and delete field.

## Join Wetland Units to the IEI_zonal table

Right-click WU_20151514 in TOC
Click Joins and Related
Click "Join…"

This will display the "Join Data pop-up window"
Select "Join attributes from a table"
"Choose the field…" = WUKey
"Choose the table to join…" = IEI_zonal
"Choose the field in the table…" = WUKey

Check “Keep all records”

## Export the joined data to a Feature Class

Conversion Tools/To Geodatabase/Feature Class to Feature Class
Input Features: WU_20150514
Output Feature Class: WU_IEI1

## Remove Join from Wetland Units

R-click WU_20150514 / Joins and Relates / Remove Joins / Remove all joins

## Join WU_IEI1 to WUpoint_IEI

Right-click WU_IEI1 in TOC
Click Joins and Related
Click “Join…”

This will display the “Join Data pop-up window”
Select “Join attributes from a table”
“Choose the field…” = WUKey (choose the first listing of WUKey)
“Choose the table to join…” = WUpoint_IEI
“Choose the field in the table…” = WUKey
Check “Keep all records”

## Export joined data to feature class

Conversion Tools/To Geodatabase/Feature Class to FeatureClass
Input Features: WU_IEI1
Output Feature Class: WU_IEI

## Replace NULL values of MEAN with centroid values of IEI

Open attribute table of WU_IEI
Select by attributes
SELECT * FROM WU_IEI WHERE: "MEAN" IS NULL
R-click MEAN and Field Calculate MEAN = [RASTERVALU]
Clear Selection

## Add field IEI, set initial value to zero

Open the attribute table to WU_IEI
Add field “IEI” (short integer) to attribute table
Field Calculate “IEI” = 0

## Assign points

SELECT * FROM WU_IEI WHERE: MEAN > 15
Field Calculate IEI = 1

SELECT * FROM WU_IEI WHERE: MEAN > 45
Field Calculate IEI = 2

SELECT * FROM WU_IEI WHERE: MEAN > 70
Field Calculate IEI = 3

## STEP 3
## Note that method in nearly identical to STEP 1 except for field names, MAJORITY, and final points
## Calculate the intermediate metric LandResil
## Summarize the Resilient_and_Connected raster values for each Wetland Unit

Spatial Analyst Tools/Zonal/Zonal Statistics as Table
    Input feature zone data: WU_20150514
    Zone field: WUKey
    Input Value raster: Resilient_and_Connected
    Output table: LandResil_zonal
    Check Ignore No Data
    Statistics Type: MAJORITY

## Extract the Resilient_and_Connected raster values for each Wetland Unit centroid

Spatial Analyst Tools / Extraction / Extract Values to Points
Input point features: WUpoint
Input raster: Resilient_and_Connected
Output point features: WUpoint_LandResil

## Note the null returns are for polygons that are too small to contain an entire pixel. For these polygons, we need to calculate the centroid of the (very small) wetland, and use the centroid to obtain the Resilient_and_Connected value.

## Delete the "Count" field from the LandResil_zonal table since "Count" is a restricted word
## and may interfere with the join in the next step

R-click table "LandResil_zonal" to open the table.
R-click the "Count" field and delete field.

## Join Wetland Units to the LandResil_zonal table

Right-click WU_20151514 in TOC
Click Joins and Related
Click "Join…"

This will display the "Join Data" pop-up window
Select "Join attributes from a table"
"Choose the field…" = WUKey
"Choose the table to join…" = LandResil_zonal
"Choose the field in the table…" = WUKey
Check "Keep all records"

## Export the joined data to a Feature Class

Conversion Tools/To Geodatabase/Feature Class to FeatureClass
Input Features: WU_20150514
Output Feature Class: WU_LandResil1

## Remove Join from Wetland Units

R-click WU_20150514 / Joins and Relates / Remove Joins / Remove all joins

## Join WU_LandResil1 to WUpoint_LandResil

Right-click WU_LandResil1 in TOC
Click Joins and Related
Click "Join…"

This will display the "Join Data" pop-up window
Select "Join attributes from a table"
"Choose the field…" = WUKey (choose the first listing of WUKey)
"Choose the table to join…" = WUpoint_LandResil
"Choose the field in the table…" = WUKey
Check "Keep all records"

## Export joined data to feature class

Conversion Tools/To Geodatabase/Feature Class to FeatureClass
Input Features: WU_LandResil1
Output Feature Class: WU_LandResil

## Replace NULL values of MAJORITY with centroid values of Resilient_and_Connected

Open attribute table of WU_LandResil
Select by attributes
SELECT * FROM WU_LandResil WHERE: "MAJORITY" IS NULL
R-click MAJORITY and Field Calculate MAJORITY = [RASTERVALU]

Clear Selection

## Add field LandResil, set initial value to zero

Open the attribute table to WU_LandResil
Add field “LandResil” (short integer) to attribute table
Field Calculate “LandResil” = 0

## Assign points

SELECT * FROM WU_LandResil WHERE: MAJORITY IN (2,3,4,11,12,13,14,33,112)
Field Calculate LandResil = 1

SELECT * FROM WU_LandResil WHERE: MAJORITY IN (2,4,11,12,13,14,33,112)
Field Calculate LandResil = 2

SELECT * FROM WU_LandResil WHERE: MAJORITY IN (11,12,112)
Field Calculate LandResil = 3

## STEP 4
## Calculate the intermediate metric ForestPatch

## Spatial Join Wetland Units and Forest Patches, selecting for the largest forest patch
## that is intersected by the Wetland Unit

ArcToolbox/ Spatial Join
Target Features: WU_LandResil
Join Features: forest_patches_over50acres_WVplus10mi.shp
Output Feature Class: WU_ForestPatch
Join Operation: JOIN_ONE_TO_ONE
Check “Keep all target features”
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
LandResil
Acreage (R-click, Merge Rule = Maximum)
Match Option: Intersect

## Add field ForestPatch, set initial value to zero

Open the attribute table to WU_ForestPatch
Add field “ForestPatch” (short integer) to attribute table
Field Calculate “ForestPatch” = 0

## Assign points

SELECT * FROM WU_ForestPatch WHERE: "ACREAGE" >= 50
Field Calculate ForestPatch = 1

SELECT * FROM WU_ForestPatch WHERE: "ACREAGE" >= 247
Field Calculate ForestPatch = 2

SELECT * FROM WU_ForestPatch WHERE: "ACREAGE" >= 2470
Field Calculate ForestPatch = 3

## STEP 5
## Spatial Join LandIntegDNR metric to the other three metrics
## Note that this can also be done with a "Join" on WUKey.

ArcToolbox/ Spatial Join
Target Features: Output Feature Class: WU_LandInteg1
Join Operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
LandResil
ForestPatch
LandIntegDNR
Match Option: Contains

ArcToolbox/ Spatial Join
Target Features: WU_LandInteg1
Join Features: WU_IEI
Output Feature Class: WU_LandInteg
Join Operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
LandResil
ForestPatch
LandIntegDNR
IEI
Match Option: Contains

## Add field LandInteg, set initial value to zero

Open the attribute table to WU_LandInteg

Add field “LandInteg” (short integer) to attribute table
Field Calculate “LandInteg” = 0

## Assign points

SELECT * FROM WU_LandInteg WHERE: ("LandIntegDNR" + "IEI" + "LandResil" + "ForestPatch") / 4 >= 0.5
Field Calculate LandInteg = 1

SELECT * FROM WU_LandInteg WHERE: ("LandIntegDNR" + "IEI" + "LandResil" + "ForestPatch") / 4 >= 1.5
Field Calculate LandInteg = 2

SELECT * FROM WU_LandInteg WHERE: ("LandIntegDNR" + "IEI" + "LandResil" + "ForestPatch") / 4 >= 2.5
Field Calculate LandInteg = 3

5.6.49 LandPos: Landscape Position

Version date: 11 October 2016

Strategy: completed 2/17/2016 EAB
GIS method: drafted 4/12/2016 EAB; verified 4/18/2016 EAB
Python code: Started 6/14/2016 MCA & Completed 6/15/2016 MCA w/Problems

> EAB 10/3/2016: need to re-run this after WFlowPath is updated. Right now the Python code generates too few headwater stream (7667 instead of 9336 LSh) and not enough Terrene (zero instead of 315 TE) wetlands.
> EAB 10/11/2016: see bottom of file for method to calculate "PropWshd".
> MCA 10/11/2016 this has been fixed and some changes have had to be made as indicated below.

Final review by EAB: 10/25/2016

Purpose:

Basic functional variable, used in several functional equations
Input to Water Quality/Headwater Location

Strategy:

1. Assign Lotic Stream Landscape Position to wetlands in an active floodplain but not within 200m of a wide river. Add headwater modifier to wetlands that intersect first- or second-order streams, are outflow wetlands, or have only intermittent flow. *Note: I experimented with adding wetlands that have a small (<40 acres) contributing watershed (Paybins 2003), or make up a large proportion (>5%) of their contributing watershed, but these were generally identified by the earlier criteria, and additions tended to be along streams with very small contributing watersheds perpendicular to the stream – not headwater wetlands.*
2. Assign Lotic River Landscape Position to wetlands in an active floodplain within 200m of a wide river.
3. Assign Lentic Landscape Position to wetlands within 25 meters of a lake.
4. Assign Terrene Landscape Position to all remaining unassigned wetlands. Add headwater modifier to wetlands that intersect first- or second-order streams, are outflow wetlands, have only intermittent flow, have a small (<40 acres) contributing watershed (Paybins 2003), or make up a large proportion (>5%) of their contributing watershed.

Definitions:

Tiner wetland classification is based on Tiner (2011) which describes and classifies wetlands by landscape position, landform, water flow path, and waterbody type (LLWW).

Tiner's landscape position characterizes wetlands based on their location within or outside the active floodplain of a waterbody (stream, river, lake). The basic landscape position types in West Virginia wetlands are:

Code	**Landscape Position**	**Description**

LR	Lotic River	Wetland Unit is located in a river (including in-river ponds and shallow lakes), within its banks, or on its active floodplain and is periodically flooded by the river. River is defined as a broad channel mapped as a polygon or 2-lined watercourse on a 1:24,000 U.S. Geological Survey topographic map.
LS	Lotic Stream	Wetland Unit is located in a stream (including in-stream ponds and shallow lakes), within its banks, or on its active floodplain and is periodically flooded by the stream. Stream is defined as a linear or single-line watercourse on a 1:24,000 U.S. Geological Survey topographic map.
LSh	Lotic Stream - headwater	Wetland Unit is located in a stream (including in-stream ponds and shallow lakes), within its banks, or on its active floodplain and is periodically flooded by the stream. Stream is defined as a linear or single-line watercourse on a 1:24,000 U.S. Geological Survey topographic map. *Modifier*: Headwater (wetlands along first- and second-order perennial streams in hilly terrain including all intermittent streams above these perennial streams).
LSc	*Lotic Stream - channelized*	*Not yet used in WV, but may be part of future development.* Wetland Unit is located in a stream (including in-stream ponds and shallow lakes), within its banks, or on its active floodplain and is periodically flooded by the stream. Stream is defined as a linear or single-line watercourse on a 1:24,000 U.S. Geological Survey topographic map. *Modifier*: Channelized (excavated stream course).
LE	Lentic	Wetland Unit is located in or along a lake or reservoir (permanent waterbody where standing water is typically much deeper than 6.6 feet at low water but including large shallow lakes >20 acres), including streamside wetlands in a lake basin (the depression containing the lake). Wetlands contiguous to the lake but at higher elevations and not in the lake basin should NOT be classified as lentic; these wetlands should be treated as terrene outflow types in most cases. This is especially common where lakes are artificially created by diking and/or excavation.
TE	Terrene	Wetland Unit is completely surrounded by upland (non-hydric soils or filled lands that are now upland development). Terrene wetlands may occur: (1) on a slope or flat, or in a depression (including ponds)

		lacking a stream but may be contiguous to a river or stream, (2) on a historic (inactive) floodplain, (3) in a landscape position crossed by a stream (e.g., an entrenched stream), but where the stream does not periodically inundate the wetland, (4) in a headwater, outflow only, position as the source of a stream.
TEh	Terrene - headwater	Wetland Unit is completely surrounded by upland (non-hydric soils or filled lands that are now upland development). Terrene wetlands (headwater) may occur in a headwater, outflow only, position as the source of a stream.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
    - Feature Class: DrainageArea27m
- M:\basemap\national_hydrology_dataset\wb-rivers.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: FirstSecondOrderFlowlines

Input Variables:

- Wetland in Floodplain (Floodplain)
- Water Flow Path (WFlowPath)
- Contributing Watershed Area (CntrWshd) in DrainageArea27m

Method:

## Joins to add variables to new Landscape Position feature class.

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_Floodplain
Join Feature: WU_WFlowPath
Output feature class: WU_LandPos1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape_Length
Shape_Area
Floodplain
WFlowPath
Match Option: CONTAINS

ArcToolbox / Data Management Tools / Joins / Join Field
Input table: WU_LandPos1

Input join field: OBJECTID
Join table: DrainageArea27m
Output Join Field: WUKey
Join Fields: WUKey, CntrWshd

## Export join to feature class

R-click WU_LandPos1 and select Data / Export Data
Output feature class: WU_LandPos

## Add text field to the Wetland Units feature class.

Add text field (length=5) to WU_LandPos attribute table: LandPos.

## Assign Lotic Stream Landscape Position.

Clear all selections.
Select by attributes from WU_LandPos
SELECT * FROM WU_LandPos WHERE "Floodplain" = 'Y'

Field Calculate selected records LandPos = “LS”
Clear all selections

## Assign Lotic River Landscape Position.

Add temporary text field to WU_LandPos attribute table: River (text, 2 characters)

Select by location
Selection method: select features from
Target layer: WU_LandPos
Source layer: wb-rivers
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 200 meters

Field Calculate selected records River = "Y"

Select by attributes
SELECT * FROM WU_LandPos WHERE: "LandPos" = 'LS' AND "River" = 'Y'

Field Calculate selected records LandPos = “LR”.
Clear all selections

## Assign headwater modifier to Lotic Stream Landscape Position for wetlands intersecting first and second order streams, outflow wetlands, and wetlands with intermittent flow. Include isolated wetlands for now since almost all of these are actually

outflow wetlands, but the streams flowing from them are too small to show up on the NHD.

Add temporary text field to WU_LandPos attribute table: FSOStream (text, 2 characters)

Select by location
Selection method: select features from
Target layer: WU_LandPos
Source layer: FirstSecondOrderFlowlines
Spatial selection method: intersect the source layer feature

Field Calculate selected records FSOStream = "Y"

Select by attributes
SELECT * FROM WU_LandPos WHERE: "LandPos" = 'LS' AND ("FSOStream" = 'Y' OR "WFlowPath" LIKE '%O%' OR "WFlowPath" LIKE '%I%')
The above select only works when using select by attribute from the Selection menu in ArcMap. Using the above select in Python and with the toolbox Select Attribute by yields a different selection. To fix this, the select below must be used.

*SELECT * FROM WU_LandPos WHERE: LandPos = 'LS' AND ((POSITION('O' IN WFlowPath) > 0) OR (POSITION('I' IN WFlowPath) > 0) OR (FSOStream = 'Y'))*

Field Calculate selected records LandPos = "LSh"

## Assign Lentic Landscape Position.

Clear all selections

Select by attributes in EnhWVWetland
Method: Create a new selection
SELECT * FROM EnhWVWetland WHERE: "WETLAND_TYPE" = 'Lake'

Select by location
Selection method: select features from
Target layer: WU_LandPos
Source layer is EnhWVWetland
Check box "use selected features"
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 25 meters

Field Calculate selected records: LANDPOS = "LE"

## Assign Terrene Landscape Position

Clear selections

SELECT * FROM WU_LandPos WHERE: "LandPos" IS NULL

Field Calculate selected records LandPos = “TE”

## Assign headwater modifier to Terrene Landscape Position for wetlands that intersect first and second order streams, are outflow wetlands, or have intermittent flow. Include isolated wetlands for now since almost all of these are actually headwater wetlands – the streams are just too small to show up on the National Hydrography Dataset.

Select by attributes
SELECT * FROM WU_LandPos WHERE: "LandPos" = 'TE' AND ("FSOStream" = 'Y' OR "WFlowPath" LIKE '%O%' OR "WFlowPath" LIKE '%I%')
Use the following select for the same reason as indicated above:

SELECT * FROM WU_LandPos WHERE: LandPos = 'TE' AND ((POSITION('O' IN WFlowPath) > 0) OR (POSITION('I' IN WFlowPath) > 0) OR (FSOStream = 'Y'))

Field Calculate selected records LandPos = "TEh"

## Assign headwater modifier to Terrene Landscape Position for wetlands with small contributing watershed.

Select by attributes.
SELECT * FROM WU_LandPos WHERE: "LandPos" = 'TE' AND "CntrWshd" < 161874

Field Calculate selected records LandPos = "TEh"

## Assign headwater modifier to Terrene Landscape Position for wetlands that occupy a large percentage of their contributing watershed

Add field to WU_LandPos: “PropWshd” (double)
Field calculate PropWshd = ShapeArea / CntrWshd

Select by attributes.
SELECT * FROM WU_LandPos WHERE: "LandPos" = 'TE' AND "PropWshd" > 0.05

Field Calculate selected records LandPos = "TEh"

5.6.50 LowSlope: Low Slope

Version date: 16 Novmeber 2016

Strategy: 2/27/2016 EAB
GIS method: completed 2/27/2016 EAB;
Python code: started & finished 3/3/2016 MCA;
Final review by EAB: 3/3/2016;

Purpose:

Water Quality Function / Potential / Surface Depressions Factor (Max 2 points)
Flood Attenuation Function / Potential

Description:

Rationale: Flat-lying wetlands are more effective at storing and slowing the velocity of flood waters and trapping sediments than sloping wetlands.
Strategy for LowSlope: For Wetland Units in Floodplains, LowSlope (maximum 2 points) is calculated as the median percent slope (SLOPE)

- Slope < 2% (2 points)
- Slope = 2-5% (1 point)
- Slope > 5% (0 points)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

Median percent slope (SLOPE)

Method:

## Create feature class to store LowSlope

R-click WU_SLOPE and select Data/Export Data
Output feature class: WU_LowSlope

## Add LowSlope fields to store points and set initial point value to zero.

Add field “LowSlope” (short integer) to attribute table of WU_LowSlope.
Field Calculate “LowSlope” = 0

## Select Wetland Units and assign points to LowSlope.

SELECT * FROM WU_LowSlope WHERE: "SLOPE" < 2
R-click “LowSlope” field, and Field Calculate “LowSlope” = 2

SELECT * FROM WU_LowSlope WHERE: "SLOPE" > 1 AND "SLOPE" < 6
R-click “LowSlope” field, and Field Calculate “LowSlope” = 1

5.6.51 MarlPEM: Emergent Wetland on Marl Deposits

Version date: 18 Septmeber 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/15/2017 EAB; results verified 9/18/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Potential / Vegetation / Floristic Quality
Max 3 points

Description:

Rationale: Emergent wetlands on marl substrates provide habitat for a large number of rare species and comprise a globally rare and imperilled habitat.

Summary of strategy: Overlay emergent wetland (PEM) on marl soils and calculate area and ratio of area to total wetland area.

3 points: PEM on marl > 1 ha in extent
2 points: PEM on marl comprises > 50% of wetland
1 point: PEM on marl > 200 m^2 in extent
0 points: none of the above criteria are met

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb
  - Feature Class: MarlSoils
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
  - Feature Dataset: CONUS_WVWetlandsProj
  - Feature Class: EnhWVWetland
    - Field: Attribute

Method:

## Select emergent wetlands and export feature class

SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'PEM%'
R-click EnhWVWetland and export selected records as “VegPEM” feature class.

## Intersect MarlSoils and emergent wetlands (PEM)

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: MarlSoils
VegPEM

Output feature class: WU_MarlPEM1
Join attributes: ONLY_FID
Output type: INPUT

## Add field to store emergent wetland/marl area.

Open attribute table of WU_MarlPEM1
Add field "MarlPEMAre" (float)
R-click "MarlPEMAre" and Field Calculate: MarlPEMAre = [Shape_Area]

## Spatial Join MarlPEM to Wetland Units and sum MarlPEM area.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: WU_MarlPEM1
Output Feature Class: WU_MarlPEM
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape_Length
Shape_Area
WUKey
    MarlPEMAre (R-click and select "Merge Rule", "Sum")
Match Option: CONTAINS

Note: if the Spatial Join fails because of difficulty opening the "WU_MarlPEM1" feature class, then export the feature class to a shapefile and re-run the Spatial Join.

## Add fields to store MarlPEMRat and MarlPEM, and set initial value of MarlPEM = 0.

Open attribute table of WU_MarlPEM
Add field MarlPEMRat (float) to WU_MarlPEM attribute table.
Add field MarlPEM (short integer) to WU_MarlPEM attribute table.
Field calculate MarlPEM = 0

## Calculate the ratio of MarlPEM area to total Wetland Unit area.

Field calculate MarlPEMRat = [MarlPEMAre] / [Shape_Area]

## Assign points

SELECT * FROM WU_MarlPEM WHERE: MarlPEMAre > 200
Field Calculate MarlPEM = 1

SELECT * FROM WU_MarlPEM WHERE: MarlPEMRat > 0.5
Field Calculate MarlPEM = 2

```
SELECT * FROM WU_MarlPEM WHERE: MarlPEMAre > 10000
Field Calculate MarlPEM = 3
```

5.6.52 Microtopo: Microtopographic Complexity

Version date: 7 March 2016

Strategy: Completed 3/1/2016 EAB
GIS method: Completed 3/2/2016 EAB & JCC, Verified 3/7/2016 EAB
Python code: Started 3/3/2016 MCA
Final review by EAB:

Purpose:

Input to Water Quality/Surface Depressions
Max 2 points. Floodplain wetlands only.

Description:

Strategy: Perimeter of summed NWI palustrine polygon perimeters divided by the square root of the Wetland Unit area.
Note that GIS determination of surface depressions and actual microtopography is not currently within our analysis reach. We are instead using the proxy of interspersion of Cowardin types (along with low slope and irregularity of the upland edge, which are calculated elsewhere) to estimate these values. The values will be much more accurately measured during rapid field assessment.

Definitions:

The National Wetlands Inventory is comprised of polygons attributed with the wetland classification system developed by Cowardin et al (1979) and updated by the National Wetlands Inventory (2016). The source data "EnhWVWetland" is a slightly enhanced version of the National Wetlands Inventory with updates for a small percentage of the polygons in West Virginia.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Input Variables:

- Shape_Length (EnhWVWetland), Shape_Area (Wetland Units): these should already be in the attribute tables

Method:

## Create a new layer from the palustrine polygons in the National Wetlands Inventory.

SELECT * FROM EnhWVWetland WHERE: "WETLAND_TYPE" IN ('PEM', 'PFOPSS', 'Pond') (57945 out of 59959 selected)
R-click EnhWVWetland / Data / Export Data

Export Selected features
Output feature class: NWIpalustrine

## Sum the perimeters of the palustrine polygons that make up each Wetland Unit.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: NWIpalustrine
Output Feature Class: WU_Microtopo
Join Operation: JOIN_ONE_TO_ONE
Check box: “Keep all target features”
Field Map of Join Features:
R-click SHAPE_Length_1 and select “Merge Rule”, “Sum”
Retain the following features (the rest can be deleted):

- Shape_Area
- Shape_Length
- SHAPE_Area_1
    - SHAPE_Length_1

Match Option: INTERSECT

## Add fields to WU_Microtopo.

Add numeric fields to WU_Microtopo attribute table: MicroRatio (float), Microtopo (short integer)

# Divide the summed perimeters of palustrine polygons by the square root of the Wetland Unit area.

Field Calculate MicroRatio = [SHAPE_Length_1] / ([Shape_Area] ^ 0.5)

# Assign points to Wetland Units.

Field Calculate Microtopo = 0

SELECT * FROM WU_Microtopo WHERE: "MicroRatio" > 8
Field Calculate Microtopo = 1

SELECT * FROM WU_Microtopo WHERE: "MicroRatio" > 15
Field Calculate Microtopo = 2

5.6.53 Organic: Organic Matter near Surface

Version date: 4 October 2016

Strategy: completed 2/29/2016 EAB
GIS method: completed & verified 3/3/2016 EAB
Python code: 4/5/2016 MCA
Final review by EAB: 4/5/2016; 10/4/16 EAB revised SsurgoOrganic layer to depth < 8 cm instead of < 6 cm (closer reading of Hruby 2012) – this does not affect the Python coding, nor does it change the number of Wetland Units with Organic = "Y".

Purpose:

Input to Water Quality / Clay and Organic Soils Factor

Description:

Organic Soil near surface (Y/N).
Summary of strategy:
Select Wetland Units that intersect with any of the following: Peatlands, NWI polygons with organic modifier, Palustrine Plots with muck or peat soils, or SSURGO chorizon with organic soils in the upper 5 cm.
Note that SSURGO mapping is very uneven, with some counties heavily mapped with organic soils and others with little or no organic soils mapped.

Definitions:

SSURGO soils data from NRCS has multiple non-spatial tables, which have one-to-many relationships with the ssurgo_wv table. We will access the component horizon table (chorizon_all) to extract the organic content, horizon, and top depth of the horizon.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
  - Feature Class: EnhWVWetland
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
  - Feature Class: Peatlands_20160228 (update if more recent file is available)
  - Feature Class: PalustrinePlotsMarch2015
- M:\basemap\ssurgo\ssurgo.gdb
  - Feature Class: ssurgo_wv
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SsurgoExports.gdb
  - Feature Class: SsurgoOrganic

NOTE: ArcGIS related tables cannot be programmed in Python, so before this procedure is run, the input data layer must be created in ArcGIS, as shown below. Note that this layer should be re-exported as SSURGO is updated by NRCS.

## Open the related one-to-many SSURGO chorizon_all table.

Open attribute table for ssurgo_wv
Click “Related Tables” (second icon from left).
Click “component to surgo: component_all” to open the component_all table. Note that a tab will appear at the bottom of the attribute table showing the tables that are open.

Click “Related Tables” (second icon from left).
Click “component to chorizon: chorizon_all” to open the chorizon_all table.

## Select soils with organic matter in the upper 8 centimeters of the soil profile with a
## thickness of at least 2 cm.

With the tab at the bottom of the attribute labelled “chorizon_all” highlighted:
SELECT * FROM chorizon_all WHERE: ("hzname" LIKE 'O%' OR "om_r" > 30) AND "hzdept_r" < 8 AND "hzthk_r" > 1
(1488 out of 28520 selected)

## Relate this selection to the spatial data in the ssurgo_wv tab.

Click “Related Tables” again to get back to component_all and then back to ssurgo_wv (146911 out of 413438 selected). Note that the last step takes some time while all of the related tables open up.

## Export data to new feature class

R-click ssurgo_wv / Data / Export Data
Export Selected Features
Output feature class: SSurgoExports.gdb\SsurgoOrganic

Input Variables:

None

Method:

## Add field Organic to Wetland Units attribute table and set initial value to “no organic”.

Add field Organic (Text, Length 2) to Wetland Units attribute table.
R-click Organic and Field Calculate “Organic” = ‘N’

## PART 1: PEATLANDS
## Select Wetland Units that are peatlands.
Select by Location
Selection method: select features from
Target layer: WU_20150514

Source layer: Peatlands_20160228 (or most recent version of Peatlands)
Spatial selection method: intersect the source layer feature

## Update value for "Organic" based on peatlands.

Open Wetland Units attribute table (277 out of 43124 selected).
R-click "Organic" and Field Calculate "Organic" = "Y"
Clear all selections.

## PART 2: NWI ORGANIC MODIFIER
## Select polygons that have an organic modifier in the National Wetland Inventory.
SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE '%g' (143 out of 59959 selected)

## Select Wetland Units that intersect organic NWI polygons.

Select by Location
Selection method: select features from
Target layer: WU_20150514
Source layer: EnhWVWetland
Check "Use selected features" (143 features selected)
Spatial selection method: intersect the source layer feature

## Update value for "Organic" based on NWI.

Open Wetland Units attribute table (68 out of 43124 selected).
R-click "Organic" and Field Calculate "Organic" = "Y"
Clear all selections.

## PART 3: PALUSTRINE PLOTS
## Select Palustrine plots that have peat or muck soils.

SELECT * FROM PalustrinePlotsMarch2015 WHERE: "Soil_Textu" LIKE '%peat%' OR "Soil_Textu" LIKE '%muck%' OR "Profile__1" LIKE '%peat%' OR "Profile__1" LIKE '%muck%' OR "Depth_of_o" NOT IN (' ', '0', '1')

## Select Wetland Units that intersect palustrine plots with organic soils.

Select by Location
Selection method: select features from
Target layer: WU_20150514
Source layer: PalustrinePlotsMarch2015
Check "Use selected features"
Spatial selection method: intersect the source layer feature

## Update value for "Organic" based on palustrine plots.

Open Wetland Units attribute table.
R-click “Organic” and Field Calculate “Organic” = “Y”
Clear all selections.

## PART 4: SSURGO SELECTION
## Select Wetland Units that intersect with the SSURGO selection.

Select by Location
Selection method: select features from
Target layer(s): WU_21050514
Source layer: SsurgoOrganic
Spatial selection method: intersect the source layer feature

## Update value for “Organic” based on ssurgo.

Open Wetland Units attribute table.
R-click “Organic” and Field Calculate “Organic” = “Y”

5.6.54 OwnerAccess: Land Ownership and Accessibility

Version date: 16 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/16/2017 EAB; Results verified 10/16/2017 EAB
Python code: 10/18/2017 YH
Final review by EAB: 10/19/2017

Purpose:

Input to Habitat / Value to Society / HUse

Description:

Maximum 2 points

Rationale: Accessible wetlands and wetlands on public land are more likely to be used and/or appreciated by the public.

GIS Method: public lands layers, plus a few additions from known private preserves with public assess.

2 points: public land (except for U.S. Navy and Air National Guard), or private land with permanent unrestricted public access to the edge of the wetland (e.g., WV Botanical Garden, TNC Cranesville Swamp Preserve, Brush Creek, Brooklyn Heights, Eidolon, Greenland Gap, Hungry Beech, Murphy Preserve, Pike Knob, Slaty Mountain, Yankauer Preserve, TNC Mt. Porte Crayon, Stauffer's Marsh, Williamstown, Camp Dawson Wetland Boardwalk, Core Arboretum)
1 point: private land with seasonal, partial, or case-by-case public access (e.g., Harewood Marsh, Ice Mountain, Upper Shavers, Tygart Valley Mitigation Bank, Wetlands of Winfield, New River Birding & Nature Center, Ward Hollow, John Gottshcal Boardwalk in Boy Scout Camp, Page Jackson Elementary School Wetland)
0 points: private land without public access

Source Data:

Note on Source Data: new public land boundaries are available for 2017 (except for Department of Defense), but have not yet been aggregated into a public lands layers. Once they are aggregated, they can replace the separate files below.

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandUnits.gdb
  - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived
  - \Boundaries2017\countyCityParkBoundaries_20107731_utm83.gdb
    - Feature Class: countyCityParkBoundaries_20170731_utm83
  - \Boundaries2017\nationalForestOwnership_USFWS_20170803_utm83.gdb
    - Feature Class: nationalForestOwnership_USFWS_20170803_utm83
      - Field: Forest Service

    - \Boundaries2017\nationalParkBoundaries_nationalParkService_20170802.gdb
        - Feature Class: nationalParkBoundaries_nationalParkService_20170802
    - \Boundaries2017\nationalWildifeRefuge_USFWS_20170803_utm83.gdb
        - \Boundaries2017nationalWildifeRefuge_USFWS_20170803
    - \Boundaries2017\wvdnrManagedLands_wvdnr_20170731_utm83.gdb
        - Feature Class: wvdnrManagedLands_wvdnr_20170731_utm83
    - \Boundaries2017\wvStateForestBoundaries_wvdof_20171003_utm83.gdb
        - Feature Class: wvStateForestBoundaries_wvdof_20171003_utm83
    - \Boundaries2017\ stateParkBoundaries_WVDNR_20170927_utm83
        - Feature Class: stateParkBoundaries_WVDNR_20170927_utm83
    - \WV_Protected_Lands_2015_PUBLIC\WV_Protected_Lands_2015_PUBLIC.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: RIBITS_banks_ILF_20171007
    - Feature Class: InfrastructureWetlands

Method:

## STEP 1: Create feature class and add field to store results; set initial value to zero

R-click WU_20150514 and select Export Data / All features
Output feature class: WetlandFunction.gdb / WU_OwnerAccess

Open attribute table of WU_OwnerAccess
Add field "OwnerAccess" (short integer)
Field calculate OwnerAccess = 0

## STEP 2: Private lands with seasonal, partial, or case-by-case public access
## Select wetlands that intersect partial-access Protected Lands

Open attribute table of WV_Protected_Lands_2015_PUBLIC.shp
SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: "P_Des_Nm" IN ('Harewood (Washington)', 'Ice Mountain (Riverbirch Inc.)', 'Upper Shavers Fork')

Select by Location
Select features from: WU_OwnerAccess
Source layer: WV_Protected_Lands_2015_PUBLIC
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add wetlands that intersect InfrastructureWetlands, all of which have at least partial access

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: InfrastructureWetlands
Spatial selection method for target layer feature(s): intersect the source feature

## Add wetlands that intersect partial-access RIBITS and ILF sites

Open attribute table of RIBITS_banks_ILF_20171007
SELECT * FROM RIBITS_banks_ILF_20171007 WHERE: "Name" = 'Tygart Valley'

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: RIBITS_banks_ILF_20171007
Use Selected Features
Spatial selection method for target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 100 m

## Assign point and clear selections

Open attribute table of WU_OwnerAccess
R-click OwnerAccess and Field Calculate OwnerAccess = 1
Clear all selections

## STEP 3: Public Ownership
## Select wetlands that intersect state or local public lands
## This step is very similar to that used in the metric HInvest

Select by Location
Select features from: WU_OwnerAccess
Source layer: stateParkBoundaries_WVDNR_20170927_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: wvdnrManagedLands_wvdnr_20170731_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: wvStateForestBoundaries_wvdof_20171003_utm83
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: countyCityParkBoundaries_20170731_utm83
Spatial selection method for target layer feature(s): intersect the source feature

## Add National Parks and Wildlife Refuges to selection

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: nationalParkBoundaries_nationalParkService_20170802
Spatial selection method for target layer feature(s): intersect the source feature

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: nationalWildifeRefuge_USFWS_20170803
Spatial selection method for target layer feature(s): intersect the source feature

## Add U.S. Army Corps of Engineers lands to selection

Open attribute table of WV_Protected_Lands_2015_PUBLIC
SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: "Mang_Name" = 'US Army Corps of Engineers'

Select by Location
Select features from: WU_OwnerAccess
Source layer: WV_Protected_Lands_2015_PUBLIC
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add National Forests to selection

Open attribute table of nationalForestOwnership_USFWS_20170803_utm83
SELECT * FROM nationalForestOwnership_USFWS_20170803_utm83 WHERE: "Ownership" = 'Forest Service'

Select by Location
Select features from: WU_OwnerAccess
Source layer: nationalForestOwnership_USFWS_20170803_utm83
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Assign points and clear selections

Open attribute table of WU_OwnerAccess
R-click OwnerAccess and Field Calculate OwnerAccess = 2
Clear all selections

## STEP 4: Select private lands with permanent public access
## Note that TNC Charlotte Ryde Preserve is contained within Cheat Canyon WMA
## Note that Mt Porte Crayon Preserve is called "Canaan/Dolly Sods (Moshein)
## Note that WV Botanical Garden is included in the county park layer

Open attribute table of WV_Protected_Lands_2015_PUBLIC
SELECT * FROM WV_Protected_Lands_2015_PUBLIC WHERE: "P_Des_Nm" IN ('Brush Creek (McPherson/Robertson)', 'Bear Rocks Lake Wildlife Management Area', 'Brooklyn Heights (Hills)', 'Cranesville Swamp Preserve', 'Eidolon Nature Preserve', 'Greenland Gap (Amendment)(Greenland Lodge Inc)', 'Hungry Beech', 'Murphy Preserve', 'Pike Knob', 'Pike Knob (Smith)', 'Slaty Mountain (Westvaco)', 'Yankauer Nature Preserve', 'Canaan Valley/Dolly Sods (Moshein)', 'Core Arboretum') OR "Comments" = 'Stauffer''s Marsh (PVAS)'

## Select wetlands that intersect selected areas

Select by Location
Select features from: WU_OwnerAccess
Source layer: WV_Protected_Lands_2015_PUBLIC
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Select open-access wetlands from the InfrastructureWetlands feature class

Open attribute table of InfrasturctureWetlands
SELECT * FROM InfrasturctureWetlands WHERE: "Access" = 'public'

## Select wetlands that intersect selected areas

Select by Location
Add to the currently selected features in: WU_OwnerAccess
Source layer: InfrasturctureWetlands
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Assign points and clear selections

Open attribute table of WU_OwnerAccess
R-click OwnerAccess and Field Calculate OwnerAccess = 2
Clear all selections

5.6.55 PublicUse: Public Use and Research

Version date: 19 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/16/2017 EAB; Results verified 10/19/2017 EAB
Python code: 10/18/2017 YH
Final review by EAB: 10/19/2017

Purpose:

Input to Habitat / Value to Society / HUse

Description:

Maximum 2 points

Rationale: Wetlands that are used and/or appreciated by the public, or are of importance to long-term scientific research, have a high value to society.

Strategy
This metric is adjusted based on the field assessment. Assign points as follows:
2 points: high public use, or built infrastructure offers potential for high public use; assign points if any of the following criteria are met. Areas included are National Wildlife Refuges and selected other public and private lands. Even though parts of NWRs are closed to the public, they still offer outstanding opportunities to experience wetlands. Individual wetlands with infrastructure, public use, or sustained scientific use are also included.

- Infrastructure
  - Maintained parking area: paved and/or big enough for a schoolbus
  - Boardwalk
  - Informational kiosk (e.g., Williamstown wetland, WV Botanical Garden)
  - Maintained road within 30 m of wetland with views of wetland (field only)
  - Maintained trail within 10 m of wetland
  - Boat access to wetland
  - Known wetlands with infrastructure, organized by source data include:
    - Wildlife Refuges: Canaan Valley & Ohio River Islands
    - State Parks: Canaan Valley, Blackwater Falls
    - WMAs: Greenbottom, Little Canaan, McClintic, Short Mountain, Valley Bend Wetlands
    - County Parks: Meadowood, WV Botanical Garden, McDonough Wildlife Refuge, Johnson T. Janes Nature Preserve and Conservation Park
    - Exemplary wetlands: Alder Run Bog, Cranberry Glades, Cranesville Swamp, Harewood Marsh, Winfield
    - Infrastructure: New River Birding and Nature Center, Wiiliamstown, Tea Creek Interpretive Trail, Stauffer's Marsh
    - Unknown data source: John Gottschal Boardwalk and Causeway at the Summit Bechtel Reserve, Camp Dawson wetland boardwalk, Page Jackson Trail, Gardens, and Wetlands
- Sustained scientific use

        - Long-term research sites: known sites include CVNWR Research Natural Area and Monongahela National Forest special botanical areas
        - Plants, animals, or water in the wetland have been monitored for >2 years, unrelated to any regulatory requirements, and data are available to the public.
    - Birding Hotspot (Brooks Bird Club, WVDNR, Audubon, and citizen birding organization hotspot lists for WV). Initial list combines eBird download of birding hotspots and main wetland sites from Eddy 2009.
        - Download ebird hotspots from: https://confluence.cornell.edu/display/CLOISAPI/eBird-1.1-HotSpotsByRegion and select wetlands within 100 meters of these hotspots
        - Select wetlands that intersect main birding wetlands areas from Eddy 2009:
            - National Wildlife Refuges: Canaan Valley, Ohio River Islands
            - Wildlife Management Areas: Fairfax Pond / Rehe, Meadow River, Pleasant Creek
            - State Parks: Canaan Valley, Blackwater Falls, Cathedral
            - Exemplary Wetlands: Altona, Cranberry Glades, Dolly Sods: Alder Run, Bear Rocks, Spruce Knob Lake, Winfield, McClintic, Greenbottom, Cranesville
            - Within 10m of trail: Stauffer's Marsh

Combined list of high public use wetlands by source data:

- Any Wetland Unit that is within 10 m of a mapped trail or a public fishing access point
- National Wildlife Refuges
    - Canaan Valley
    - Ohio River Islands
- National Forest
    - Monongahela NF botanical areas
- State Parks
    - Canaan Valley
    - Blackwater Falls
    - Cathedral
- County Parks
    - Meadowood
    - WV Botanical Garden
    - McDonough Wildlife Refuge
    - Johnson T. Janes Nature Preserve and Conservation Park
- Wildlife Management Areas
    - Fairfax Pond / Rehe
    - Meadow River
    - Pleasant Creek
    - Greenbottom
    - Little Canaan
    - McClintic
    - Short Mountain
    - Valley Bend Wetlands
- Exemplary Wetlands (no need to include wetlands already selected in the categories above)
    - Altona

    - Cranesville Swamp
    - Dolly Sods: Alder Run, Bear Rocks
    - Harewood Marsh
    - Spruce Knob Lake
    - Winfield
- Infrastructure wetlands
    - New River Birding and Nature Center
    - Wiiliamstown
    - Tea Creek Interpretive Trail
    - Ward Hollow
    - Camp Dawson wetland boardwalk
    - John Gottschal Boardwalk and Causeway at the Summit Bechtel Reserve
    - Page Jackson Elementary School Trail, Gardens, and Wetlands
- Any Wetland Unit that is within 100 m of an eBird birding hotspot

1 point: Assign point if any of the following criteria are met.
Hunting or trapping area identified by WVDNR as having species that occur in wetlands. This includes WMAs and State Forests with populations of waterfowl, grouse, woodcock, beaver, mink, muskrat, deer, or bear. Also include wetland edge species (Keith Krantz, WVDNR, pers comm 20171010): rabbit, bobcat, coyote, red fox, raccoon, opossum. Keith Krantz writes: "Depending on your location, marsh/swamp/cottontail rabbits are routinely found along the edges of the drier PEM wetlands, anywhere standing cattails and frozen/dry ground can be found. My trapping friends routinely catch bobcats/coyotes/red fox in the winter in these same wetland settings, likely hunting the aforementioned rabbits and rodents. Raccoons and opossums are typically found around water as well foraging for whatever they can find."

0 points: None of the above criteria are met.

Source Data:

Note on Source Data: new public land boundaries are available for 2017 (except for Department of Defense), but have not yet been aggregated into a public lands layers. Once they are aggregated, they can replace the separate files below. The Birding Hotspots can be expanded to include birding destinations identified in "Birding Guide to West Virginia", complied by Greg Eddy 2009, Brooks Bird Club.

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived
    - \Boundaries2017\countyCityParkBoundaries_20107731_utm83.gdb
        - Feature Class: countyCityParkBoundaries_20170731_utm83
    - \Boundaries2017\nationalWildifeRefuge_USFWS_20170803_utm83.gdb
        - Feature Class: nationalWildifeRefuge_USFWS_20170803
    - \Boundaries2017\wvdnrManagedLands_wvdnr_20170731_utm83.gdb
        - Feature Class: wvdnrManagedLands_wvdnr_20170731_utm83
    - \Boundaries2017\stateParkBoundaries_WVDNR_20170927_utm83
        - Feature Class: stateParkBoundaries_WVDNR_20170927_utm83

    - \DNR_Fishing\PublicFishingAccessSites_2017_10.shp
    - \USFS\botanical_areas_MNF.shp *(do not share this layer – sensitive data)*
    - \201710_WVDNR_property_boundary.gdb
        - Feature Class: PropertyBoundaries_WVDNR_20171011
    - trails_Sep_27_2017_webmercator.shp
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb
    - Feature Class: eBirdHotspots_20171011
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
    - Feature Class: InfrastructureWetlands
    - Feature Class: ExemplaryOrBrankedWetlands31Mar2015

Method:

## STEP 1: Create feature class and add field to store results; set initial value to zero

R-click WU_20150514 and select Export Data / All features
Output feature class: WetlandFunction.gdb / WU_PublicUse

Open attribute table of WU_PublicUse
Add field "PublicUse" (short integer)
Field calculate PublicUse = 0

## STEP 2: Moderate public use
## Select WMAs and State Forests identified as hunting/trapping areas for wetland species

Open attribute table of PropertyBoundaries_WVDNR_20171011
SELECT * FROM PropertyBoundaries_WVDNR_20171011 WHERE: "hWaterfowl" = 1 OR "hGrouse" = 1 OR "hWoodcock" = 1 OR "tBeaver" = 1 OR "tMink" = 1 OR "tMuskrat" = 1 OR "hDeer" = 1 OR "hBear" = 1 OR "hRabbit" = 1 OR "tBobcat" = 1 OR "tCoyote" = 1 OR "tRedFox" = 1 OR "tRaccoon" = 1 OR "tOpossum" = 1

Select by Location
Select features from: WU_PublicUse
Source layer: PropertyBoundaries_WVDNR_20171011
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Assign point and clear selections

Open attribute table of WU_PublicUse
R-click PublicUse and Field Calculate PublicUse = 1
Clear all selections

## STEP 3: High public use

## Select wetlands within 10 m of a mapped trail or a public fishing access point

Select by Location
Select features from: WU_PublicUse
Source layer: trails_Sep_27_2017_webmercator
Spatial selection method for target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 10 meters

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: PublicFishingAccessSites_2017_10
Spatial selection method for target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 10 meters

## Add to Selection wetlands in National Wildlife Refuges

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: nationalWildlifeRefuge_USFWS_20170803_utm83
Spatial selection method for target layer feature(s): intersect the source feature
Do not apply a search distance

## Add to Selection wetlands in special botanical areas supporting long-term research

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: botanical_areas_MNF.shp
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands in certain state parks

Open attribute table of stateParkBoundaries_WVDNR_20170927_utm83
SELECT * FROM stateParkBoundaries_WVDNR_20170927_utm83 WHERE: "Unit_Nm" IN ('Blackwater Falls State Park', 'Canaan Valley Resort State Park', 'Cathedral State Park')

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: stateParkBoundaries_WVDNR_20170927_utm83
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands in certain local parks

Open attribute table of countyCityParkBoundaries_20170731_utm83

SELECT * FROM countyCityParkBoundaries_20170731_utm83 WHERE: "Unit_Nm" IN ('WV Botanic Garden', 'Meadowood Park', 'McDonough Wildlife Refuge ', 'Johnson T. Janes Nature Preserve and Conservation Park')

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: countyCityParkBoundaries_20170731_utm83
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands in certain WMAs

Open attribute table of wvdnrManagedLands_wvdnr_20170731_utm83
SELECT * FROM wvdnrManagedLands_wvdnr_20170731_utm83 WHERE: "Unit_Nm" IN ('Fairfax Pond / Rehe Wildlife Management Area', 'Green Bottom Wildlife Management Area', 'Little Canaan Wildlife Management Area', 'McClintic Wildlife Management Area', 'Meadow River Wildlife Management Area', 'Pleasant Creek Wildlife Management Area', 'Short Mountain Wildlife Management Area', 'Valley Bend Wetlands Wildlife Management Area')

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: wvdnrManagedLands_wvdnr_20170731_utm83
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection certain Exemplary Wetlands not already selected

Open attribute table of ExemplaryOrBrankedWetlands31Mar2015
SELECT * FROM ExemplaryOrBrankedWetlands31Mar2015 WHERE: "Name" IN ('Alder Run Bog', 'Altona-Piedmont Marsh', 'Bear Rocks Bog', 'Cranesville Swamp', 'Harewood Marsh', 'Spruce Knob Lake inlet', 'Spruce Knob Lake outlet', 'Winfield Swamp')

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: ExemplaryOrBrankedWetlands31Mar2015
Use Selected Features
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands from the InfrastructureWetlands feature class

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: InfrastructureWetlands
Spatial selection method for target layer feature(s): intersect the source feature

## Add to Selection wetlands within 100 meters of an eBird birding hotspot

Select by Location
Add to the currently selected features in: WU_PublicUse
Source layer: eBirdHotspots_20171011
Spatial selection method for target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 100 meters

## Assign points and clear selections

Open attribute table of WU_PublicUse
R-click PublicUse and Field Calculate PublicUse = 2
Clear all selections

5.6.56 RestoredWetlands: Restored, Enhanced, or Created Wetlands, including In-Lieu Fee and Mitigation Banks

Version date: 12 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/12/2017 EAB
Python code: not needed
Final review by EAB: 10/12/2017

Purpose:

Input to Habitat / Value to Society / HInvest

Update schedule:

This layer should be updated at least every 2 years.

Description:

Rationale: Wetlands that have been restored, enhanced or created represent investments of time and money, and are of high value to society.

Source Data:

- Project documents and/or air photos

Method:

Create named wetland polygons where there are known investments in restoration, enhancement, or wetland creation. Priority is given to adding wetland polygons that are not already included in the RIBITS database maintained by USACE (ILF & banks) or the WV_ProtectedLands database (Wetland Reserve Program) maintained by the Freshwater Institute.

Most additions will need to be heads-up digitized from project documents or from recent air photos. Many project boundaries will be well-defined through wetland delineation. If exact boundaries are not easily available, then the boundaries can be approximate, e.g. +- 100m, and do not need to be attributed to NWI codes. During wetland functional assessment, this layer will be intersected with Wetland Units, and therefore the boundaries do not need to be exact.

Data Fields:

WetlandName [text 50 characters]
Give the wetland a name based on the name in the project documents.

RestoreType [text 50 characters]

Describe the type of activity: restoration, enhancement, creation

RegType [text 20 characters]
Describe the category of regulatory action, if any: ILF, mitigation bank

RestoreDate [text 20 characters]
The date that restoration, enhancement, or creation was completed.

Organization [text 50 characters]
Name of group or organization doing the restoration, enhancement, or creation work.

MapSource [text 50 characters]
Briefly describe the data source used to create the wetland polygon, e.g., "copied from NWI" or "heads-up digitized".

Comment [text 50 characters]
Add brief comments, such as name of educational organization or date of boardwalk construction

5.6.57 RoadRail: Road and Railroads

Version date: 24 January 2018

Strategy: completed 3/12/2016 EAB
GIS method: completed 3/12/2016; verified 3/12/2016; 1/24/2018 EAB Updated to replace railways layer with better source data.
Python coding: started & completed 3/15/2016 MCA; updated 1/24/2018 YH
Final review by EAB: 3/15/2016

Purpose:

Water Quality Function, Opportunity aspect
Max 2 points

Description:

Rationale: Road and rail crossings can increase sediment and contaminant loads (especially salt and petrochemicals) to a wetland (McElfish et al. 2008).
Summary of strategy: Assign 1 point to Wetland Units within 50 meters of a road or railroad track.
Assign 2 points to Wetland Units within 5 meters of a road or railroad track. Note that the "M:\LayerFiles\arcsde_backup.gdb\basemap_cultural_non_replica\SDE_railway_tiger" layer is more accurately and completely mapped than the "M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\Rail" layer.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\basemap\tiger_2013\WV_Transportation_UTM.gdb
  - Feature Class: All_Roads
- M:\LayerFiles\arcsde_backup.gdb\basemap_cultural_non_replica\SDE_railway_tiger

Input Variables:

None

Method:

## Create feature class to store RoadRail variable

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_RoadRail

## Add RoadRail field to Wetland Units and set initial point value to zero.

Open attribute table of WU_RoadRail
Add field "RoadRail" (short integer)

R-click RoadRail and Field Calculate RoadRail = 0

## Select the Wetland Units within 50 meters of a road or railroad track and assign 1 point.

Select by location
Selection method: select features from
Target Layer: WU_RoadRail
Source layer: All_Roads
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 50 meters

Select by location
Selection method: add to the currently selected features in
Target Layer: WU_RoadRail
Source layer: SDE_railway_tiger
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 50 meters

In WU_RoadRail, R-click RoadRail and Field Calculate RoadRail = 1

## Select the Wetland Units within 5 meters of a road or railroad track and assign 2 points.

Select by location
Selection method: select features from
Target Layer: WU_RoadRail
Source layer: All_Roads
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

Select by location
Selection method: add to the currently selected features in
Target Layer: WU_RoadRail
Source layer: SDE_railway_tiger
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 5 meters

In WU_RoadRail, R-click RoadRail and Field Calculate RoadRail = 2

5.6.58 Runoff: Runoff and Storage Potential

Version date: 9 January 2017

Strategy: Completed 3/24/2016 EAB
GIS method: completed 3/25/2016 EAB; verified 4/15/2016 EAB; updated & re-verified
12/20/2016 EAB; re-verified with new SoilRunoff values (peatlands = 0) EAB 1/9/2017
Python coding: Completed 12/20/2016 MCA; 1/11/2017 Re-ran MCA
Final review by EAB: 1/11/2017

Purpose:

Input to Flood Attenuation / Potential
Max 5 points (floodplain), 4 points (groundwater).
Rationale: Slowing and storing runoff is an essential aspect of flood attenuation by wetlands. Soils with low runoff/high infiltration characteristics, seasonal ponding, complex surface topography, complex upland edge or a close hydrologic connection with a stream contribute to the ability of a wetland to perform this function.
Summary of strategy: Combine points for SoilRunoff, SeasonPond, and Microtopo. If total exceeds 5 points for floodplain wetlands, reduce to 5. If total exceeds 4 points for groundwater wetlands, reduce to 4.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- SoilRunoff (2 points)
- SeasonPond (3 points)
- Microtopo (2 points)
- Floodplain

Method:

## Spatial joins to add input variables to Wetland Units attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_20150514
Join Feature: WU_SoilRunoff
Output Feature Class: WU_Runoff1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
SoilRunoff
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Runoff1
Join Feature: WU_SeasonPond
Output Feature Class: WU_Runoff2
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
SoilRunoff
SeasonPond
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Runoff2
Join Feature: WU_Microtopo
Output Feature Class: WU_Runoff3
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
SoilRunoff
SeasonPond
Microtopo
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Runoff3
Join Feature: WU_Floodplain
Output Feature Class: WU_Runoff
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
SoilRunoff
SeasonPond
Microtopo
Floodplain
Match option: CONTAINS

## Add Runoff field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Runoff
Add field “Runoff” (short integer)

R-click Runoff and Field Calculate Runoff = 0

## Sum the points for SoilRunoff, SeasonPond, Microtopo.

Open attribute table of WU_Runoff
R-click Runoff and Field Calculate “Runoff” = [SoilRunoff] + [SeasonPond] + [Microtopo]

## Reduce the total points to 5 (floodplain) or 4 (groundwater).

Open attribute table of WU_Runoff
SELECT * FROM WU_Runoff WHERE: "Runoff" > 5
Field Calculate (selection only) “Runoff” = 5

Clear Selection
SELECT * FROM WU_Runoff WHERE: "Runoff" > 4 AND "Floodplain" = 'N'
Field Calculate (selection only) “Runoff” = 4

5.6.59 Runoff50m: Lands producing runoff within 50 meters of wetland boundary

Version date: 28 Feb 2017

Strategy: completed 4/21/2016 EAB
GIS method: 4/21/2016 EAB
Python coding: 2/14/2017 MCA *Note that this procedure is based on the Disturb50m variable, and much of the code can be shared.* Revision with RunoffLand Godzilla polygons simplified 2/28/2017.
Final review by EAB: 2/28/2017

Purpose: :

Flood Attenuation Function / Opportunity aspect
Max 2 points

Description:

Rationale: Impervious surfaces, urban areas, agricultural areas, mining, industrial and commercial land uses, and recent timber harvests contribute to increased runoff. Soil types with high runoff/low infiltration characteristics also produce runoff. The land use and soil type immediately adjacent to a wetland have a strong influence on the surface runoff that the wetland receives.
Strategy: Overlay 50m wetland buffer with land uses and soil types that are likely to contribute to increased runoff. Assign points as follows: >33% of area within 50 meters in such land uses = 2 points; > 10% = 1 point; <=10% = 0 points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
  - Feature Class: Buffer50m (this was created for Disturb50m variable)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: RunoffLand

Input Variables:

None

Method:

## Intersect the 50m buffers and the runoff lands.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: Buffer50m
RunoffLand
Output feature class: Buffer50mRun
Join attributes: ALL
Output type: INPUT

## Dissolve runoff lands by wetland buffer

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: Buffer50mRun
Output Feature Class: Buffer50mRun_diss
Dissolve Fields: WUKey
Statistics Fields: BufferArea (Statistic Type = First)
Check box "Create multipart features" (default)

## Add field and calculate ratio of runoff area to total drainage area.

Open attribute table of Buffer50mRun_diss
Add field "Run50mRat" (float)
Field calculate Run50mRat = [Shape_Area] / [FIRST_BufferArea]

## Join ratio of runoff land to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_20150514
Input Join Field: WUKey
Join Table: Buffer50mRun_diss
Output Join Field: WUKey

## Export joined data

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_Runoff50m

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_Runoff50m
Add field "Runoff50m" (short integer)
R-click Runoff50m and Field Calculate Runoff50m = 0

## Assign points.

SELECT * FROM WU_Runoff50m WHERE: "Run50mRat" > 0.1
R-click Runoff50m and Field Calculate Runoff50m = 1

SELECT * FROM WU_Runoff50m WHERE: "Run50mRat" > 0.33
R-click Runoff50m and Field Calculate Runoff50m = 2

5.6.60 RunoffWshd: Runoff within contributing watershed

Version date: 7 March 2017

Strategy: completed 4/21/2016 EAB
GIS method: 4/21/2016 EAB
Python coding: 3/7/2017 MCA
Final review by EAB: 3/7/2017
Note that this procedure is based on the DisturbWshd variable, and much of the code can probably be shared.

Purpose:

Flood Attenuation Function / Opportunity aspect
Max 2 points

Description:

Rationale: Impervious surfaces, urban areas, agricultural areas, mining, industrial and commercial land uses contribute to increased runoff in a catchment (Wisconsin GIS-RAM metric). Recent logging, and high runoff/low infiltration soil types also contribute to runoff. The presence of these areas in the contributing watershed of a wetland is a good indicator that surface runoff may be reaching the wetland, especially during storm events.
Summary of strategy: Calculate the ratio of runoff-producing area to total area within the contributing watershed of each Wetland Unit. Assign points as follows: 25% of contributing watershed area in runoff lands = 2 points; 10-25% = 1 point; <10% = 0 points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
  - Feature Class: DrainageArea27m
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: RunoffLand

Method:

## Intersect the drainage areas and the runoff areas.

ArcToolbox / Analysis Tools / Overlay / Intersect
Input features: DrainageArea27m
RunoffLand
Output feature class: DrainAreaRun
Join attributes: ALL
Output type: INPUT

## Dissolve runoff areas by drainage area

ArcToolbox / Data Management Tools / Generalization / Dissolve
Input feature: DrainAreaRun
Output Feature Class: DrainAreaRun_diss
Dissolve Fields: WUKey
Statistics Fields: CntrWshd (Statistic Type = First)
Check box “Create multipart features” (default)

## Add field to DrainAreaRun_diss and calculate ratio of runoff area to total drainage area.

Open attribute table of DrainAreaRun_diss
Add field “RunWshdRat” (float)
Field calculate RunWshdRat = [SHAPE_Area] / [FIRST_CntrWshd]

## Join ratio of runoff area to Wetland Units

ArcToolbox / Data Management Tools / Joins / Add Join
Input table: WU_20150514
Input Join Field: WUKey
Join Table: DrainAreaRun_diss
Output Join Field: WUKey

## Export joined data

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_RunoffWshd

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_RunoffWshd
Add field “RunoffWshd” (short integer)
R-click RunoffWshd and Field Calculate RunoffWshd = 0

## Assign points.

SELECT * FROM WU_RunoffWshd WHERE: RunWshdRat > 0.1
R-click RunoffWshd and Field Calculate RunoffWshd = 1

SELECT * FROM WU_RunoffWshd WHERE: RunWshdRat > 0.25
R-click RunoffWshd and Field Calculate RunoffWshd = 2

5.6.61 SeasonPond: Seasonal Ponding

Version date: 10 March 2016

Strategy: completed 2/27/2016 EAB
GIS method: completed 3/4/2016 & verified 3/10/2016 EAB
Python code: started and completed on 3/15/2016 MCA
Final review by EAB: 3/15/2016

Purpose:

Input to Water Quality/Potential aspect/ChemTime Factor
Max 3 points. Groundwater wetlands only.

Description:

Rationale: The area of the wetland that is seasonally ponded is an important characteristic in understanding how well it will remove nutrients, specifically nitrogen. The highest levels of nitrogen transformation occur in areas of the wetland that undergo a cyclic change between oxic (oxygen present) and anoxic (oxygen absent) conditions. The oxic regime (oxygen present) is needed so certain types of bacteria will change nitrogen that is in the form of ammonium ion ($NH4+$) to nitrate, and the anoxic regime is needed for denitrification (changing nitrate to nitrogen gas) (Mitsch and Gosselink 1993). The area that is seasonally ponded is used as an indicator of the area in the wetland that undergoes this seasonal cycling. The soils are oxygenated when dry but become anoxic during the time they are flooded.
Summary of strategy: EnhWVWetland. Select wetland polygons from the NWI that are NOT permanently flooded. Calculate the ratio of the non-permanently flooded area to the total area of the Wetland Unit. Assign points as follows:

- SeaPondRatio = 70-100% cover: 3 points
- SeaPondRatio = 40-70% cover: 2 points
- SeaPondRatio = 10-40% cover: 1 point
- SeaPondRatio < 10% cover: 0 point

Definitions:

Cowardin Water Regime modifier H = permanently flooded

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Input Variables:

None

Method:

## Select all the palustrine wetlands that are not permanently flooded

Clear all selections.
Select * FROM EnhWVWetland WHERE: "ATTRIBUTE" NOT LIKE '%H%' AND "ATTRIBUTE" LIKE 'P%'

## Create layer of non-permanently flooded wetlands from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: “SeasonPond”

## Join non-permanently flooded wetlands to Wetland Units and sum the non-permanently flooded area

Analysis Tools / Overlay / Spatial Join
Target features: WU_20150514
Join features: SeasonPond
Output feature class: WU_SeasonPond
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep All Target Featues”
Field Map of Join Features
    R-click Shape_Area_1 and select “Merge Rule” / “Sum”
Match Option: INTERSECT

## Add field to store ratio of non-permanently-flooded area to total area

In WU_SeasonPond, add field: SeaPondRatio (float)

## Calculate ratio of non-permanently flooded area to Wetland Unit area.

In WU_SeasonPond, R-click attribute SeaPondRatio
Field Calculate SeaPondRatio = [SHAPE_Area_1] / [Shape_Area]

## Add field to store points for non-permanently flooded area and set initial value to zero.

In WU_SeasonPond, add field: SeasonPond (short integer)
Field calculate SeasonPond = 0

## Assign points to Wetland Units for seasonal ponding

SELECT * FROM WU_SeasonPond WHERE: "SeaPondRatio" > 0.1
R-click SeasonPond, Field Calculate SeasonPond = 1

SELECT * FROM WU_SeasonPond WHERE: "SeaPondRatio" > 0.4
R-click SeasonPond, Field Calculate SeasonPond = 2

```
SELECT * FROM WU_SeasonPond WHERE: "SeaPondRatio" > 0.7
R-click SeasonPond, Field Calculate SeasonPond = 3
```

5.6.62 Septic: Septic System Failure Risk

Version date: 17 March 2016

Strategy: completed 3/11/2016 EAB
GIS method: completed 3/11/2016 EAB; verified 3/11/2016 EAB
Python coding: started & completed 3/17/2016 MCA
Final review by EAB: 3/17/2016

Purpose:

Water Quality Function, Opportunity aspect, input to Discharges (discharges to the wetland within 100 meter buffer)
This procedure creates a data layer. No points at this level, maximum of 1 point can be assigned in Discharges variable

Description:

Rationale: Septic systems can pollute groundwater because nitrogen is not removed underground. Plumes of nitrogen from septic systems can be traced at least 250ft in the groundwater (Aravena and others 1993). Use an aerial photograph of the unit to determine if there are any residences within 250ft of the unit. Septic systems are still in common use in many areas outside of city boundaries. If you are outside city limits in areas with lots of 1/2 acre or larger you can assume the houses are on septic systems unless another type of water supply system is indicated on the water supply GIS layer.
Summary of strategy: Calculate as known septic systems (NPDES permit) or presence of structures. Exclude sewered areas, urbanized areas, and areas with low or very low risk of septic failure.

Source Data:

- M:\wr\owrnpdes_.shp
- M:\basemap\WVSAMB\structures_SAMB_points_UTM83.shp
  - Geometry Type: Point
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: SeweredAreas
  - Feature Class: SepticFailureRiskStatsgo
- M:\LayerFiles\arcsde_backup.gdb
  - Feature Dataset: tiger2010 Feature Class: urbanized_areas

Method:

## PART 1: Select structures that intersect sewered areas or urbanized areas.

Select by Location
Selection method: select features from
Target layer: structures_SAMB_points_UTM83
Source layer: SeweredAreas
Spatial selection method: intersect the source layer feature

Select by Location
Selection method: add to the currently selected features in
Target layer: structures_SAMB_points_UTM83
Source layer: urbanized_areas
Spatial selection method: intersect the source layer feature

## Reverse selection.

Open attribute table of structures_SAMB_points_UTM83 and click “Switch Selection”

## Select only those structures that intersect moderate and high risk septic failure areas.

Select by attributes
SELECT * FROM SepticFailureRiskStatsgo WHERE: "SepticZone" = 'high' OR "SepticZone" = 'moderate'

Select by Location
Selection method: select from the currently selected features in
Target layer: structures_SAMB_points_UTM83
Source layer: SepticFailureRiskStatsgo
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

## Export points.

R-click structures_SAMB_points_UTM83 and Data/Export Data/Export selected features
Output feature class: Septic

## PART 2: Select septic permits from owrnpdes_ and export to point feature class.

Select by attributes from owrnpdes_
SELECT * FROM owrnpdes_ WHERE "perm_type" = 'Septic Tank'

R-click owrnpdes_ and Data/Export Data/Export selected features
Output feature class: SepticTanks

## Select septic permits that intersect sewered areas or urbanized areas.

Select by Location
Selection method: select features from
Target layer: SepticTanks
Source layer: SeweredAreas
Spatial selection method: intersect the source layer feature

Select by Location

Selection method: add to the currently selected features in
Target layer: SepticTanks
Source layer: urbanized_areas
Spatial selection method: intersect the source layer feature

## Reverse selection.

Open attribute table of SepticTanks and click “Switch Selection”

## Select only those permits that intersect moderate and high risk septic failure areas.

Select by attributes
SELECT * FROM SepticFailureRiskStatsgo WHERE: "SepticZone" = 'high' OR "SepticZone" = 'moderate'

Select by Location
Selection method: select from the currently selected features in
Target layer: SepticTanks
Source layer: SepticFailureRiskStatsgo
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

## Export points.

R-click structures_SAMB_points_UTM83 and Data/Export Data/Export selected features
Output file: SepticTankRisk

## PART 3: Append SepticTankRisk to Septic to include both septic tanks and structures.

In ArcCatalog, R-click Septic and Load/Load Data
Input data: SepticTankRisk
Add
Check radio button “I do not want to load all features into a subtype” (default)
No need to link any target fields with matching source fields
Check radio button “Load all of the source data”

5.6.63 Slope: Median Percent Slope

Version date:

Strategy: completed 2/17/2016 EAB
GIS method: completed 2/26/2016 JCC (reviewed by EAB)
Python code: Started 3/2/2016 MCA, Finished 3/3/2016 but the code takes a long time to run
Python results verified by MCA and EAB: 7/6/2017

Purpose:

Input to multiple variables and functions

Description:

SLOPE is calculated as the median value of percent slope pixels within a Wetland Unit.

Definitions:

None

Source Data:

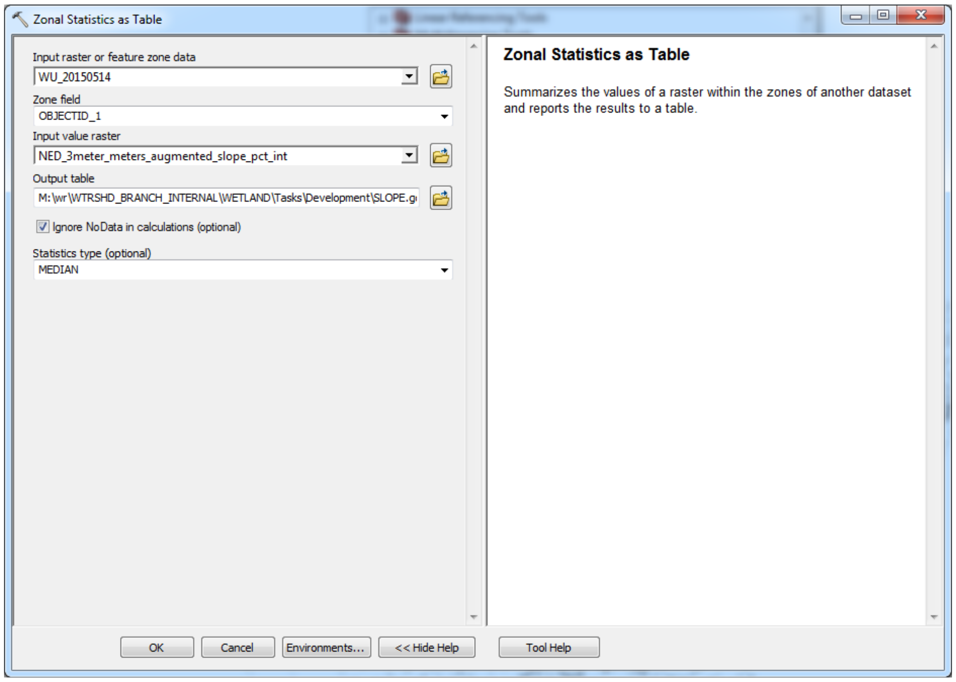
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- Database: M:\dems\ned_slope_aspect.gdb
  Raster: NED_3meter_meters_augmented_slope_pct_int*Note that the zonal statistics tool will run faster if an intermediate raster layer of slope median for each Wetland Unit is calculated first. However, it takes significant time to calculate the intermediate raster layer, so it is preferable to run the zonal statistics directly from the NED raster above. The intermediate raster layer corresponding to WU_20150514 is at: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Slope_raster\ Raster: median_slope Please note that this "median_slope" raster ONLY works with WU_20150514; it must be re-generated for any other set of Wetland Unit polygons.*

Input Variables:

None

Method:

1. Spatial Analyst Tools/Zonal/Zonal Statistics as Table Tool
    a. Input feature zone data = WU20150514
    b. Zone field = ObjectID
    c. Input Value raster = NED_3meter_meters_augmented_slope_pct_int
    d. Output table: <select location>
    e. Check Ignore No Data
    f. Statistics Type: Median


Zoonal Statistics as Table

Input raster or feature zone data: WU_20150514

Zone field: OBJECTID_1

Input value raster: NED_3meter_meters_augmented_slope_pct_int

Output table: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Development\SLOPE.g

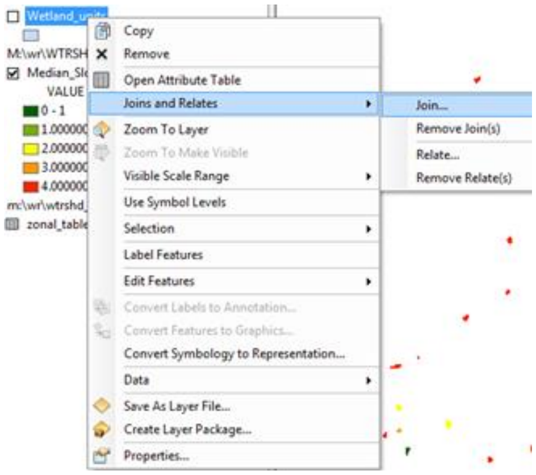
☑ Ignore NoData in calculations (optional)

Statistics type (optional): MEDIAN

Zonal Statistics as Table

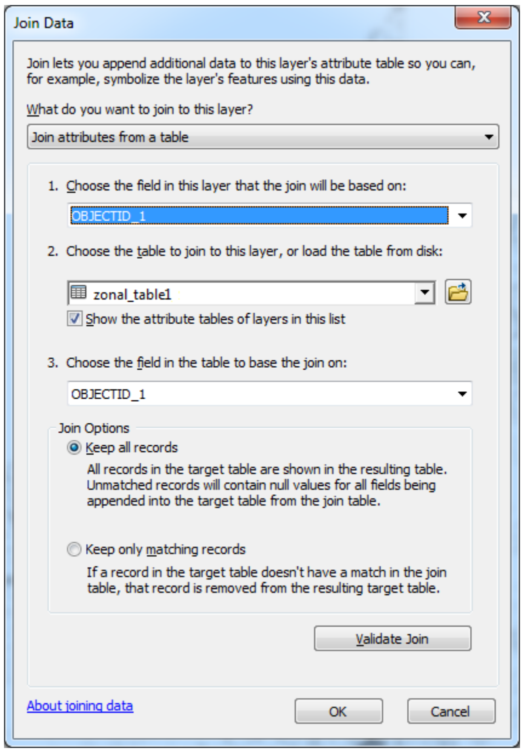
Summarizes the values of a raster within the zones of another dataset and reports the results to a table.

2. Join Wetland Units so the Zonal Statistic Table Output
    a. Right-click WU20151514 Shapefile in TOC
        i.
    b. Click Joins and Related
    c. Click "Join..."



    d.

3. This will display the "Join Data pop-up window"
    a. Select "Join attributes from a table"
    b. "Choose the field..." = OBJECTID_1
    c. "Choose the table to join..." = zonal_table1
    d. "Choose the field in the table..." = OBJECTID_1
    e. Check "Keep all records"

f. 
Join Data

Join lets you append additional data to this layer's attribute table so you can, for example, symbolize the layer's features using this data.

What do you want to join to this layer?
Join attributes from a table

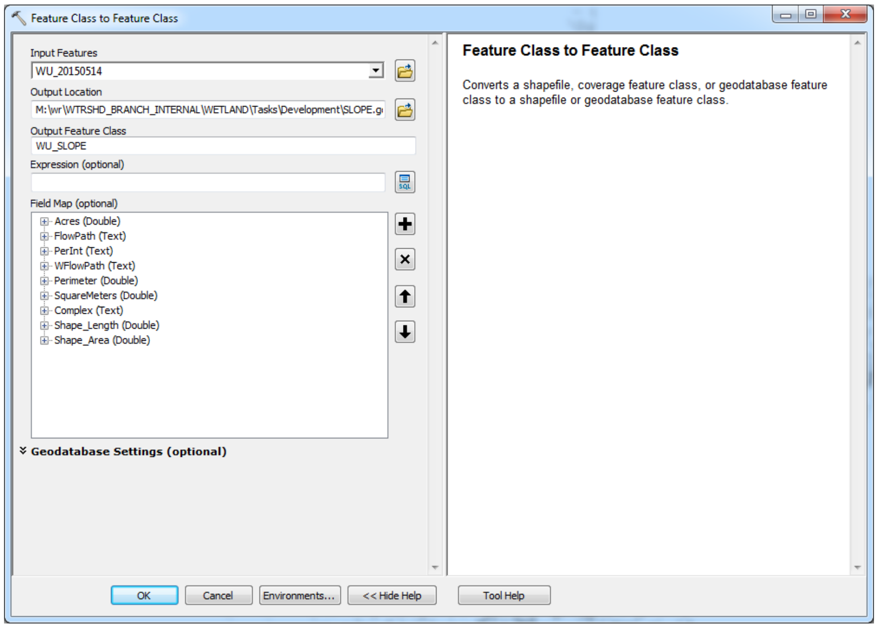
1. Choose the field in this layer that the join will be based on:
OBJECTID_1

2. Choose the table to join to this layer, or load the table from disk:
zonal_table1
[✓] Show the attribute tables of layers in this list

3. Choose the field in the table to base the join on:
OBJECTID_1

Join Options
● Keep all records
All records in the target table are shown in the resulting table. Unmatched records will contain null values for all fields being appended into the target table from the join table.

○ Keep only matching records
If a record in the target table doesn't have a match in the join table, that record is removed from the resulting target table.

4. Export the joined data to a Feature Class (Conversion Tools/To Geodatabase/Feature Class to FeatureClass)
   a. Go to the “Feature Class to FeatureClass” Tool
   b. Input Features: WU_20150514
   c. Output Location: Desired Geodatabase
   d. Output Feature Class: WU_SLOPE
   e. Click “OK”
   f. 

Feature Class to Feature Class

Input Features
WU_20150514
Output Location
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Development\SLOPE.g
Output Feature Class
WU_SLOPE
Expression (optional)

Field Map (optional)
- Acres (Double)
- FlowPath (Text)
- PerInt (Text)
- WFlowPath (Text)
- Perimeter (Double)
- SquareMeters (Double)
- Complex (Text)
- Shape_Length (Double)
- Shape_Area (Double)

Geodatabase Settings (optional)

Feature Class to Feature Class

Converts a shapefile, coverage feature class, or geodatabase feature class to a shapefile or geodatabase feature class.

5. Rename MEDIAN field to SLOPE
    a. Open the attribute table to WU_SLOPE
    b. Add field “SLOPE” (short integer) to attribute table
    c. Field Calculate “SLOPE” = MEDIAN
    d. Delete the MEDIAN field

5.6.64 SlopeWshd: Mean Percent Slope of the Contributing Watershed

Version date: 22 December 2016

Strategy: completed 4/21/2016 EAB
GIS method: completed 4/21/2016 EAB & JCC; 10/6/16 EAB I think we need to do step 1 in Python rather than ArcGIS because of need for iteration
Python code: Completed 2/8/2017 MCA
Final review by EAB: 2/8/2017
Note that this procedure is based on the SLOPE variable, and some of the code can probably be shared.

Purpose:

Input to Flood Attenuation Function / Opportunity aspect

Description:

Rationale: Steep slopes contribute to rapid runoff and increases in flood flows during storm events. Wetlands below these slopes will have more opportunity to intercept and slow flood flows (Wisconsin GIS-RAM metric).
Strategy: Calculate mean percent slope of contributing watershed. Slopes > 15% = 2 points; slopes 5-15% = 1 point; slopes < 5% = 0 points.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
  - Feature Class: DrainageArea27m
- M:\dems\ned_slope_aspect.gdb
  Raster: NED_3meter_meters_augmented_slope_pct

Method:

6. Spatial Analyst Tools/Zonal/Zonal Statistics as Table Tool
   a. Input feature zone data = DrainageArea27m
   b. Zone field = WUKey
   c. Input Value raster = NED_3meter_meters_augmented_slope_pct
   d. Output table: slope_wshd
   e. Check box "Ignore NoData in calculations"
   f. Statistics Type: Mean

   Step 1 issue: DrainageArea27m has overlapping polygons, and the resulting output "slope_wshd" table only has values for one-third of the total drainage areas. Can we re-design this step to iterate the zonal analysis for each polygon, in order to get around the problem of overlapping polygons? Here is the text from ArcGIS help:

   > If the zone feature input has overlapping polygons, the zonal analysis will not be performed for each individual polygon. Since the feature input is converted to a raster, each location can only have one value.

An alternative method is to process the zonal statistics iteratively for each of the polygon zones and collate the results.

Resolution: done, but it is time-consuming to run the entire state. It will not be a problem for smaller datasets.

7. Join DrainageArea27m to the Zonal Statistic Table Output
    a. Right-click DrainageArea27m
    b. Click Joins and Relates
    c. Click "Join…"

8. This will display the "Join Data" pop-up window
    a. Select "Join attributes from a table"
    b. "Choose the field…" = WUKey
    c. "Choose the table to join…" = slope_wshd
    d. "Choose the field in the table…" = WUKey
    e. Check "Keep all records"

9. Export the joined data to a Feature Class (Conversion Tools/To Geodatabase/Feature Class to FeatureClass)
    a. Go to the "Feature Class to FeatureClass" Tool
    b. Input Features: DrainageArea27m
    c. Output Location: Desired Geodatabase
    d. Output Feature Class: SlopeWshd1
    e. Click "OK"

10. Rename MEAN field
    a. Open the attribute table to SlopeWshd1
    b. Add field "MnSlopeWshd" (short integer) to attribute table
    c. Field Calculate "MnSlopeWshd" = MEAN

## Join the slope values to Wetland Units.

ArcToolbox / Data Management Tools / Joins / Join Field
Input table: WU_20150514
Input Join Field: OBJECTID_1
Join Table: SlopeWshd1
Output Join Field: WUKey
Join Fields: WUKey, MnSlopeWshd

## Export joined data

R-click WU_20150514 and select Data / Export Data
Output feature class: WU_SlopeWshd

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_SlopeWshd

Add field “SlopeWshd” (short integer)
R-click SlopeWshd and Field Calculate SlopeWshd = 0

## Assign points.

SELECT * FROM WU_SlopeWshd WHERE: MnSlopeWshd > 5
R-click SlopeWshd and Field Calculate SlopeWshd = 1

SELECT * FROM WU_SlopeWshd WHERE: MnSlopeWshd > 15
R-click SlopeWshd and Field Calculate SlopeWshd = 2

5.6.65 SoilH: Hydrologic regime for soil

Version date: 27 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/27/2017 EAB; results verified 9/27/2017 EAB
Python coding: 9/28/2017 YH
Final review by EAB: 10/2/2017

Description:

Input to Habitat & Ecological Integrity / Potential
Max 6 points
Rationale: Undisturbed soils, organic or calcareous soils, and structural patches all contribute important physical habitat characteristics to wetlands.

Strategy: Sum the points for the metrics SoilIntact, SoilOrgCalc, and StrucPatch.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_Disturb50m
    - Field: SoilIntact
  - Feature Class: WU_SoilOrgCalc
    - Field: SoilOrgCalc
  - Feature Class: WU_StrucPatch
    - Field: StrucPatch

Method:

## Note that the step below could be done with a Join on WUKey instead of a Spatial Join if that is easier to code.

## Spatial Joins to merge fields into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Features: WU_Disturb50m
Join Features: WU_SoilOrgCalc
Output Feature Class: WetlandFunction.gdb\WU_SoilH1
Join Operation: JOIN_ONE_TO_ONE
Check "Keep All Target Features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
SoilIntact
SoilOrgCalc
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Features: WU_SoilH1
Join Features: WU_StrucPatch
Output Feature Class: WetlandFunction.gdb\WU_SoilH
Join Operation: JOIN_ONE_TO_ONE
Check "Keep All Target Features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
SoilIntact
SoilOrgCalc
StrucPatch
Match Option: CONTAINS

## Add field to store SoilH and set initial value to zero

Open attribute table of WU_SoilH
Add field "SoilH" (short integer)
Field Calculate SoilH = 0

## Assign points to SoilH

Open attribute table of WU_SoilH
Field Calculate SoilH = [SoilIntact] + [SoilOrgCalc] + [StrucPatch]

5.6.66 SoilIntact: Lack of Soil Disturbance or Compaction

Version date: 27 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/27/2017 EAB; results verified 9/27/2017 EAB
Python coding: 9/28/2017 YH
Final review by EAB: 10/2/2017

Description:

Input to Habitat & Ecological Integrity / Potential / SoilH
Max 2 points
Rationale: Soil disturbance or compaction reduces the habitat value and ecological integrity of a wetland. This metric is best observed in the field.

Strategy: Estimate using the proxy of land use disturbance in 50-meter buffer (NLCD developed areas, urbanized areas, recent timber harvest, and grazed pastures). Assign points as follows:

- 2 points: No disturbed land uses within 50m buffer
- 1 point: Trace to 50% of 50m buffer is covered by disturbed land uses
- 0 points: > 50% of 50m buffer is covered by disturbed land uses

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_Disturb50m
    - Field: Dist50mRat

Method:

## Add SoilIntact field and set initial point value to zero.

Open attribute table of WU_Disturb50m
Add field "SoilIntact" (short integer)
R-click SoilIntact and Field Calculate SoilIntact = 0

## Assign points to SoilIntact

Open attribute table of WU_Disturb50m
SELECT * FROM WU_Disturb50m WHERE: Dist50mRat IS NULL
Field Calculate SoilIntact = 2

SELECT * FROM WU_Disturb50m WHERE: Dist50mRat > 0
Field Calculate SoilIntact = 1

SELECT * FROM WU_Disturb50m WHERE: Dist50mRat > 0.5
Field Calculate SoilIntact = 0

5.6.67 SoilOrgCalc: Special soil types, i.e., organic or calcareous soil

Version date: 24 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/27/2017 EAB; results verified 9/27/2017 EAB
Python coding: 9/28/2017 YH
Final review by EAB: 10/2/2017 EAB

Purpose:

Input to Habitat & Ecological Integrity / Potential / SoilH
Max 1 point

Description:

Rationale: Soils developed on limestone, dolomite, or marl deposits contain elevated levels of calcium or magnesium. A rich and uniquely adapted flora and fauna are characteristic of calcareous wetlands. Peatlands, characterized by deep organic soils, also provide habitat for uniquely adapted flora and fauna.

Strategy: Assign one point if the criteria are met for either organic soil or for calcareous soil

- Histosol variable (SSURGO organic content or known peatland) OR
- Karst metric: SSURGO calcareous soil, SSURGO karst or limestone/dolomite bedrock geology

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_Histosol
    - Field: Histosol
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_Karst
    - Field: Karst

Method:

## Note that the Spatial Join could be replaced by a Join on WUKey if that is easier.

## Spatial join to merge Histosol and Karst into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Histosol
Join Feature: WU_Karst
Output Feature Class: WetlandFunctionResults\WetlandFunction.gdb\WU_SoilOrgCalc
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”

Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
Histosol
Karst
Match option: CONTAINS

## Add SoilOrgCalc field and set initial point value to zero.

Open attribute table of WU_SoilOrgCalc
Add field “SoilOrgCalc” (short integer)
R-click SoilOrgCalc and Field Calculate SoilOrgCalc = 0

## Assign points to SoilOrgCalc

Open attribute table of WU_SoilOrgCalc
SELECT * FROM WU_SoilOrgCalc WHERE: ("Histosol" + "Karst") > 0
Field Calculate SoilOrgCalc = 1

5.6.68 SoilRunoff: Soil Runoff and Infiltration Potential

Version date: 7 January 2017

Strategy: Completed 3/24/2016 EAB
GIS method: Completed 3/24/2016 EAB; revised 1/7/2017 EAB (peatlands changed from 2 to 0 points)
Python code: 5/24/2016 Nate Gunn; Revision 1/10/2017 MCA
Final review by EAB: 1/11/2017

Purpose:

Input to Flood Attenuation / Potential
Max 2 points.

Description:

Rationale: Wetlands with soil that have characteristics of low runoff and high infiltration are better able to slow and absorb flood flows. Soils are characterized by NRCS according to hydrologic group based on runoff and infiltration characteristics. The high water table in peatlands places them in the high runoff group.
Strategy: Assign 0 points to Wetland Units with soils that have high runoff characteristics. Assign 1 point to Wetland Units with soils that have moderate runoff/infiltration. Assign 2 points to Wetland Units with soils that have low runoff/high infiltration characteristics.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\basemap\ssurgo\ssurgo.gdb
  - Feature Class: ssurgo_wv
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb
  - Feature Class: Peatlands_20160228

Method:

## Create feature class to store SoilRunoff

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_SoilRunoff

## Add SoilRunoff field to Wetland Units and set initial point value to zero.

Open attribute table of WU_SoilRunoff
Add field "SoilRunoff" (short integer)
R-click SoilRunoff and Field Calculate SoilRunoff = 0

## Select SSURGO soils that have moderate runoff/infiltration

Select by Attributes

Method: Create a new selection
SELECT * FROM ssurgo_wv WHERE: "hydgrpdcd" IN ('C', 'C/D')

## Select Wetland Units that contain soils with moderate runoff/infiltration

Select by Location
Selection method: select features from
Target layer: WU_SoilRunoff
Source layer: ssurgo_wv
Check box "Use selected features"
Spatial selection method: intersect the source layer feature

## Assign points to SoilRunoff.

In WU_SoilRunoff, Field Calculate (selection only) SoilRunoff = 1

## Select SSURGO soils that have low runoff/high infiltration

Select by Attributes
Method: Create a new selection
SELECT * FROM ssurgo_wv WHERE: "hydgrpdcd" IN ('A', 'A/D', 'B', 'B/D')

## Select Wetland Units that contain soils with low runoff/high infiltration

Select by Location
Selection method: select features from
Target layer: WU_SoilRunoff
Source layer: ssurgo_wv
Check box "Use selected features"
Spatial selection method: intersect the source layer feature

## Update points in SoilRunoff.

In WU_SoilRunoff, Field Calculate (selection only) SoilRunoff = 2

## Select Wetland Units with peatlands

Select by Location
Selection method: select features from
Target layer: WU_SoilRunoff
Source layer: Peatlands_20160228
Spatial selection method: intersect the source layer feature

## Update points in SoilRunoff.

In WU_SoilRunoff, Field Calculate (selection only) SoilRunoff = 0

5.6.69 StreamEdge: Complexity of wetland/stream interface

Version date: 16 November 2016

Strategy: completed 4/14/2016 EAB
GIS method: drafted 4/14/2016 EAB, verified 4/15/2016 EAB; re-verified 11/16/16 EAB;
    replace EnhWVWetland with NWIExports.gdb\Rivers 2/15/2018 EAB
Python coding: Started 8/30/2016 MCA, Completed 12/16/2016 MCA
Final review by EAB: 12/19/2016

Purpose:

Used in Flood Attenuation Function / Potential aspect / Runoff
Max 2 points

Description:

Rationale: Wetlands that are strongly connected to streams have a high capacity to receive overbank flow and intercept floodwaters (Wisconsin GIS-RAM metric). *Note that we cannot determine whether a stream is disconnected/entrenched from GIS. This will be measured during rapid field assessment. The GIS metric is limited to the length/complexity of shared stream/wetland boundaries. Ditches and drains should NOT be included in this metric.*
Strategy: Sum of shared Wetland Unit/river (polygonal stream) boundary lengths and length of NHD stream segments within wetland, divided by the square root of the Wetland Unit area. Assign 2 points to Wetland Units with a ratio > 3.4, 1 point if the ratio is 1-3.4, and zero points if the ratio < 1. *Note that these thresholds were set by examining the histogram of values, with the highest class (2 points) representing the upper tail of the distribution, i.e., more than one standard deviation above the median. The middle class (1 point) represents the middle of the peak to where the upper tail begins, and also has the real-world significance of being more sinuous than a straight line through a theoretical square wetland.*

Source data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  Feature Class: **WU_20150514**
- M:\basemap\NHDH_WV.gdb, Feature Dataset: Hydrography, Feature Class: **NHDFlowline**
  Note that NHD data is incorrectly attributed (no ephemeral or intermittent streams) in Preston County and parts of Monongalia, Barbour, Morgan, Kanawha, Putnam, Wirt, Marshall, Tucker, and small parts of about 10 additional counties. Not much we can do about this for now.
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIExports.gdb
  - Feature Class: **Rivers**

Method:

## PART 1: RIVER LENGTH

## Select Wetland Units that share a boundary with a river and export them.

Clear all selections
Select by Location
Target: WU_20150514
Source: NWIExports.gdb\Rivers
Spatial selection: intersect the source layer feature.

R-click WU_20150514 and select Data / Export Data
Export: Selected features
Output feature class: WUbyRiver

## Convert Wetland polygons to lines

Data Management Tools / Features / Polygon to Line
Input Features: WUbyRiver
Output Feature Class: WURiverLines
Do not check box "Identify and store polygon…"

##Retain only the wet perimeter lines

Analysis Tools / Overlay / Intersect
Input features: WURiverLines
NWIExports.gdb\Rivers
Output feature class: RiverEdges
Join Attributes: ONLY_FID
Output Type: Input

## Add field and calculate wet perimeter in RiverEdges

In RiverEdges, add field "RiverPerim" (float)
Field Calculate "RiverPerim" = [Shape_Length]

## Spatial Join Wetland Units to RiverEdges

Clear all selections.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: RiverEdges
Output Feature Class: WU_StreamEdge1
Join operation: Join_one_to_one
Check box "Keep all target features"
Field map of join features: (retain the following)
Shape_Length
Shape_Area
RiverPerim (R-click, Merge Rule = SUM)

Match Option: Intersect

## PART 2: STREAM LENGTH
## Intersect stream lengths with Wetland Units

Analysis Tools / Overlay / Intersect
Input features: WU_StreamEdge1
NHDFlowline
Output feature class: WUStream
Join Attributes: ONLY_FID
Output Type: Input

## Add field to WUStream to store stream length

Add field StreamL (float) to WUStream attribute table
Calculate Geometry / Property: Length; Units: meters

## Sum the stream lengths within each WU.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_StreamEdge1
Join Features: WUStream
Output Feature Class: WU_StreamEdge
Join operation: Join_one_to_one
Check box "Keep all target features"
Field map of join features: (retain the following)
Shape_Length
Shape_Area
RiverPerim
StreamL (R-click, Merge Rule = SUM)
Match Option: Intersect

## PART 3: Calculate ratio and assign points.

Open attribute table of WU_StreamEdge

Select by Attributes
SELECT * FROM WU_StreamEdge WHERE: "RiverPerim" IS NULL
Field calculate RiverPerim = 0

SELECT * FROM WU_StreamEdge WHERE: "StreamL" IS NULL
Field calculate StreamL = 0
Clear selections

Add fields to attribute table: StreamEdge (short integer), StreamRatio (float)
Field calculate StreamEdge = 0, StreamRatio = 0

Field calculate StreamRatio = ([RiverPerim] + [StreamL]) / ([Shape_Area] ^ 0.5)

SELECT * FROM WU_StreamEdge WHERE: "StreamRatio" > 1
Field calculate StreamEdge = 1

SELECT * FROM WU_StreamEdge WHERE: "StreamRatio" > 3.4
Field calculate StreamEdge = 2

5.6.70 StrucPatch: Structural Patch Richness

Version date: 27 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/27/2017 EAB; results verified 9/27/2017 EAB
Python coding: 9/28/2017 YH (YH: results are off by 3 out of 43124 Wetland Units – this was traced back to StreamEdge3. EAB: this should resolve itself when the earlier metrics are re-run)
Final review by EAB: 9/28/2017 EAB

Purpose:

Input to Habitat & Ecological Integrity / Potential / SoilH
Max 3 points

Description:

Rationale: Structural patches offer key habitat elements to species, increasing biodiversity, complexity, and ecological integrity of a wetland. This metric is best assessed in the field.

Strategy: In GIS, this metric is estimated using the proxies of interspersion of NWI polygons, complexity of the upland-wetland interface, stream channel complexity within the wetland, and the amount of woody vegetation in the wetland. Sum VegHorInt (interspersion of NWI codes, 0-3), VegVerStr (0-3), and StreamRatio (0-3 points in quantiles 0, 1.4, 2.4, >2.4). Sum all points and divide by the number of factors.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_Microtopo
        - Field: VegHorInt
    - Feature Class: WU_VegVerStr
        - Field: VegVerStr
    - Feature Class: WU_StreamEdge
        - Field: StreamEdge

Method:

## Note that the Spatial Join could be replaced by a Join on WUKey if that is easier.

## Spatial joins to merge input metrics (VegHorInt, VegVerStr, StreamEdge) into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_Microtopo
Join Feature: WU_VerVerStr
Output Feature Class: WetlandFunctionResults\WetlandFunction.gdb\WU_StrucPatch1
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”

Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
VegHorInt
VegVerStr
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_StrucPatch1
Join Feature: WU_StreamEdge
Output Feature Class: WetlandFunctionResults\WetlandFunction.gdb\WU_StrucPatch
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
VegHorInt
VegVerStr
StreamEdge
Match option: CONTAINS

## Add StreamEdge3 field and set initial point value to zero.

Open attribute table of WU_StrucPatch
Add field "StreamEdge3" (short integer)
R-click StreamEdge3 and Field Calculate StreamEdge3 = 0

## Assign points to StreamEdge3

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: "StreamRatio" > 0
Field Calculate StreamEdge3 = 1

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: "StreamRatio" > 1.4
Field Calculate StreamEdge3 = 2

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: "StreamRatio" > 2.4
Field Calculate StreamEdge3 = 3

## Add StrucPatch field and set initial point value to zero.

Open attribute table of WU_StrucPatch

Add field “StrucPatch” (short integer)
R-click StrucPatch and Field Calculate StrucPatch = 0

## Assign points to StrucPatch

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: ("VegHorInt" + "VegVerStr" + "StreamEdge3") > 1
Field Calculate StrucPatch = 1

## Assign points to StrucPatch

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: ("VegHorInt" + "VegVerStr" + "StreamEdge3") > 3
Field Calculate StrucPatch = 2

## Assign points to StrucPatch

Open attribute table of WU_StrucPatch
SELECT * FROM WU_StrucPatch WHERE: ("VegHorInt" + "VegVerStr" + "StreamEdge3") > 5
Field Calculate StrucPatch = 3

5.6.71 SWoutflow, SWOutflow2: Surface Water Outflows

Version date: 25 October 2016

Strategy: Completed 2/17/2016 EAB
GIS method: Completed 2/17/2016 EAB, Revised & Verified 3/24/2016 EAB
Python code: Started & completed 6/14/2016 MCA * see notes to talk to EAB
    EAB 10/3/2016: need to re-run after WFlowPath is updated. Revised 10/27/16 MCA; completed 12/19/2016.
Final review by EAB: 12/19/2016

Purpose:

SWOutflow: Input to Water Quality.
    Max 4 points. Groundwater wetlands only.
SWOutflow2: Input to Flood Attenuation.
    Max 2 points. Groundwater wetlands only.

Description:

Rationale for SWOutflow (water quality): Pollutants that are in the form of particulates (e.g., sediment, or phosphorus that is bound to sediment) will be retained in a wetland with no outlet. Wetlands with no outlet are scored the highest for this indicator. An outlet that flows only seasonally is usually better at trapping particulates than one that is flowing all the time.
Rationale for SWOutflow2 (flood attenuation): Surface water is retained in a wetland with no outlet. Wetlands with an outlet that is highly constricted, or flows only seasonally, are more likely to retain water than those with permanently flowing outlets.
Strategy for SWOutflow (water quality): No surface water outlet = 4 points; Intermittent or highly constricted permanent outlet = 3 points; permanently flowing surface outlet = 1 point.
Strategy for SWOutflow2 (flood attenuation): Assign points based on Tiner Water Flow Path (WFlowPath) as follows: No surface water outlet = 2 points; Intermittent or highly constricted permanent outlet = 1 point; permanently flowing surface outlet = 0 point.
Note 1: Highly constricted outlets can be determined during rapid field assessment, but we are not able to identify them remotely at this point.
Note 2: SWoutflow2 differs from SWoutflow in that the total points are 2 instead of 4.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Water Flow Path (WFlowPath)
- Floodplain

Method:

## Spatial join to add input variables WFlowPath and Floodplain to attribute table

## and create feature class to store Surface Water Outflow factor

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Features: WU_Floodplain
Join Features: WU_WFlowPath
Output Feature Class: WU_SWOutflow
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep All Target Features”
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
WFlowPath
Floodplain
Match Option: CONTAINS

## Add SWOutflow fields to Wetland Units and set initial point value to zero.

Open attribute table of WU_SWOutflow
Add fields “SWOutflow” (short integer) and “SWOutflow2” (short integer)
R-click SWOutflow and Field Calculate SWOutflow = 0
R-click SWOutflow2 and Field Calculate SWOutflow2 = 0

## Assign points for to SWoutflow.

Clear all selections.
SELECT * FROM WU_SWOutflow WHERE: "Floodplain" = 'N'
Field Calculate SWoutflow = 1

Clear all selections.
SELECT * FROM WU_SWOutflow WHERE: "Floodplain" = 'N' AND "WFlowPath" IN ('OI', 'TI', 'BI', 'IB')
Field Calculate (selection only) SWoutflow = 3
Field Calculate (selection only) SWoutflow2 = 1

Clear all selections.
SELECT * FROM WU_SWOutflow WHERE: "Floodplain" = 'N' AND "WFlowPath" IN ('IS')
Field Calculate (selection only) SWoutflow = 4
Field Calculate (selection only) SWoutflow2 = 2

5.6.72 TMDL: Wetland is in a watershed with a TMDL plan

Version date: 16 March 2016

Strategy: 3/16/2016
GIS method: drafted 3/16/2016
Python code: not needed – this is the procedure to create a spatial dataset and only needs to be updated every year or two.
Final review by EAB: 3/16/2016

Purpose:

Water Quality Function, Value to Society aspect
Maximum 2 points (assigned in WQPlan factor)
This procedure creates a layer showing the watersheds with a TMDL plan.

Description:

TMDL (Total Maximum Daily Load). A TMDL exists for the drainage in which the wetland is found (2 points). A Total Maximum Daily Load (TMDL) plan is a plan of action used to clean up streams that are not meeting water quality standards. The TMDL program is part of the Watershed Branch of the WVDEP. TMDLs have been completed for 32 watersheds in West Virginia, listed at: http://www.dep.wv.gov/wwe/watershed/tmdl/Pages/default.aspx

Source Data:

- M:\wr\WTRSHD_BRANCH\TMDL FINAL GIS DATA\TMDL Subsheds (SWS) alphabetical order.lyr

Method:

Create layer showing watersheds with a TMDL plan

Open TMDL Subsheds layers. Export the first watershed to a new feature class "TMDL" and load data from all subsequent watersheds into this polygon feature class.

Final data layer is stored in:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: TMDL

Annual updates

Updates should be done annually, comparing "TMDL" to the newest delineations and adding any additional watersheds to the compiled feature class.

5.6.73 TotalLossRP100: Estimated Total Loss from 100-yr Flood

Version date: 9 November 2016

Strategy: 4/21/2016 EAB
GIS method: 4/21/2016 EAB
Python coding: not needed (unless it makes it easier to compile the layer)
 Complete 11/9/2016 MCA (I used python to make this process quicker and easier)
Final verification by EAB: 11/9/2016

Purpose:

Flood Attenuation / Value to Society/EconRisk

Description:

Rationale: Wetlands upstream of economically valuable flood-prone infrastructure (structures, roads, developed lands, cropland) can reduce the costs and negative impacts of flood damages on society.
Summary of strategy: Merge the county "Hazus" shapefiles into a statewide layer.

Source Data:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\wvHazusOutputs_wvdhsem_012011_gcs83_gdb\wvHazusOutputs_wvdhsem_012011_gcs83.gdb

Feature Dataset:	Analysis_Data
Feature Class:	(all counties)

Output Layer:

M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\FloodplainData.gdb
 Feature Class: TotalLossRP100

Method:

## Create new feature class to store statewide total loss data.

R-click Analysis Data / Barbour and select Data / Export Data
Output feature class:
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Floodplain\FloodplainData.gdb
 Feature Class: TotalLossRP100

## Delete unnecessary fields to keep file size manageable

Delete all fields EXCEPT
ObjectID, Shape, TotalLossRP100, Shape_Length, Shape_Area

## Load counties into statewide feature class

In ArcCatalog, R-click TotalLossRP100 and select Load / Load Data
This brings up the Simple Data Loader. Select Next.
Input data: Browse to Berkeley in Hazus / Analysis_Data. Open.
List of source data to load (click "Add" button below and Berkeley will appear in the white box.
Next.

Choose an existing geodatabase (automatically populated with FloodplainData.gdb)
Select the target feature class (automatically populated with TotalLossRP100)
Check radio button "I do not want to load all features into a subtype" (default)
Next.

For each target field, select the source field that should be loaded into it (TotalLossRP100 should be pre-selected).
Next.

Check radio button "Load all of the source data" (default).
Next.
Finish.

Repeat "load counties" step for the rest of the counties in the state.

5.6.74 VegAll: All Vegetation Types

Version date: 24 March 2016

Strategy: 3/24/2016 EAB
GIS method: completed & verified 3/24/2016 EAB
Python code: started 5/12/2016 MCA, completed 5/13/2016
Final review by EAB: 10/4/2016

Purpose:

Input to Flood Attenuation / Potential / Vegetation Factor
Max 1 point.

Description:

Rationale: All vegetation, even grazed pastures and aquatic bed vegetation, plays at least a minor role in slowing flood flows.
Summary of strategy: Assign 1 point to Wetland Units with at least 50% areal cover by vegetation of any type.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Method:

## Select the forest, shrubland, emergent, moss, and aquatic bed vegetation.
## (Note that the VegAll layer was created in VegByLP and could be accessed from there also, thus skipping the next two steps of selecting and creating the layer)

Clear all selections
SELECT * FROM EnhWVWetland WHERE:
"ATTRIBUTE" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PML%'

## Create layer of all vegetation from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: “VegAll”

## Add field to store vegetation area

Open attribute table of VegAll
Add field VegArea (float)
R-click VegArea and Field Calculator: VegArea = [SHAPE_Area]

## Join vegetation to wetland units and sum the vegetation area

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: VegAll
Output Feature Class: WU_VegAll
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain Shape_Length, Shape_Area, VegArea
R-click VegArea and select “Merge Rule”, “Sum”
Match Option: INTERSECT

## Add field to store ratio of vegetated area to total area

In WU_VegAll, add field: VegRatio (float)

## Calculate ratio of vegetation to Wetland Unit area.

In WU_VegAll, R-click attribute VegRatio
Field Calculate VegRatio = [VegArea] / [Shape_Area]

## Add new attribute field to store points for VegAll and set initial value to zero.

In WU_VegAll, add field: VegAll (short integer)
Field calculate VegAll = 0

## Assign points to Wetland Units for woody vegetation

SELECT * FROM WU_VegAll WHERE: "VegRatio" > 0.5
R-click VegAll, Field Calculate VegAll = 1

5.6.75 VegByLP: Vegetation fringing open water

Version date: 14 March 2018

Strategy: completed EAB 18 Feb 2016
GIS method: completed 3/2/2016 EAB & JCC; verified 3/2/2016 EAB; revised to (a) replace EnhWVWetland with NWIExports/RiverLakes, (b) replace procedure to select Lakes with NWI export of Lakes, and (c) replace selection criteria for VegAll to include vegetated lacustrine and riverine wetlands 2/15/2018 EAB. Revised to replace Lakes with RiversLakes and to add perennial streams 3/6/2018 EAB.
Python coding: Started 3/22/2016 MCA & completed 3/23/2016 MCA
Final review by EAB: 3/24/2016

Purpose:

Used in Water Quality Function / WQPotential / VegWQ
Max 1 point

Description:

Rationale: Vegetation fringing the banks of open water, including lakes, reservoirs, ponds or streams, provides vertical structure to filter out pollutants or absorb them, enhancing sediment retention and stabilization, phosphorus retention, and nitrate removal (Adamus et al. 2010, Hruby 2012). Wetlands in which the average width of shoreline vegetation is large are more likely to retain sediment and toxic compounds than where shoreline vegetation is narrow (Adamus et al. 1991). Aquatic bed species that die back every year play a role in improving water quality. These plants take up nutrients in the spring and summer that would otherwise be available to stimulate algal blooms in the water body (Reynolds and Davies 2001). In addition, aquatic bed species change the chemistry of the lake/pond bottom to facilitate the binding of phosphorus (Moore et al. 1994). Vegetated shorelines provide physical protection from erosion, including shoreline anchoring and the dissipation of erosive forces (Adamus and others. 1991). Fringing wetlands that have extensive, persistent (especially woody) plants provide protection from overland flows or waves associated with large storms (Adamus and others 1991). GIS estimation of this metric is replaced during the field assessment.

Summary of strategy: Assign 1 point to vegetated Wetland Units that

- intersect a river, lake, or reservoir
- contain pond(s)
- contain a through-flowing perennial stream

Source data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: **WU_20150514**
- [NewInputPolygons]
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIExports.gdb
  - Feature Class: RiversLakes
- WFlowPath

Method:

## Add fields to Wetland Units feature class.

Add fields: VegByLP (short integer), PondRatio (float),

## Set initial value of VegByLP to zero.

Field calculate VegByLP = 0

## PART 1: CREATE VEGETATION LAYER.
## Select the forest, shrubland, emergent, moss, and aquatic bed vegetation.

Clear all selections
SELECT * FROM [NewInputPolygons] WHERE:
"ATTRIBUTE" LIKE '%EM%' OR "ATTRIBUTE" LIKE '%AB%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%' OR "ATTRIBUTE" LIKE 'PML%'
Create layer from selection.
Export data to feature class: VegAll

## PART 2: CREATE RIVER AND LAKE SELECTIONS.

## Select Wetland Units that intersect rivers or lakes.

Clear all selections
Select by Location
Target: Wetland Units
Source: NWIExports.gdb\RiversLakes
Spatial selection: intersect the source layer feature.
Create layer from selection.
Export data to feature class: WUbyLake

## Select WUbyLake that are vegetated

Select by Location
Target: WUbyLake
Source: VegAll
Spatial selection: contain the source layer feature
Create layer from selection.
Export data to feature class: WUbyLakeVegAll

## Assign 1 point to VegByLP for the records in WUbyLakeVegAll.
Field calculate VegByLP = 1

## PART 3: CREATE POND SELECTIONS.
## Select ponds.

Clear all selections
SELECT * FROM [NewInputPolygons] WHERE:
"ATTRIBUTE" LIKE 'PA%' OR "ATTRIBUTE" LIKE 'PU%'
Create layer from selection.
Export data to feature class: Ponds

## Select Wetland Units that contain ponds

Clear all selections
Select by Location
Target: Wetland Units
Source: Ponds
Spatial selection: contain the source layer feature
Create layer from selection.
Export data to feature class: WUwithPond

## Select WUwithPond that are vegetated

Select by Location
Target: WUwithPond
Source: VegAll
Spatial selection: contain the source layer feature
Create layer from selection.
Export data to feature class: WUwithPondVegAll

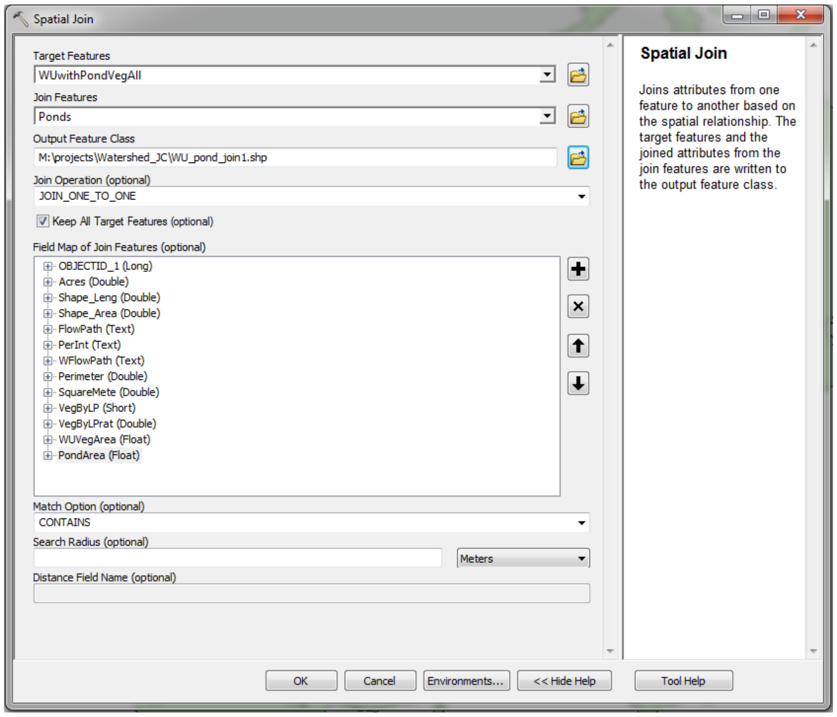
## Add Area fields and calculate geometry in WUwithPondVegAll

In Ponds, add field: PondArea (Float)
R-click PondArea, Field Calculator, PondArea = [Shape_Area]
In WUwithPondVegAll = WUarea (Float)
R-click WUArea, Field Calculator, WUArea = [Shape_Area]

## Spatial join vegetated WU with ponds

Analysis Tools / Overlay / Spatial Join Tool

a. Target: WUwithPondVegAll
b. Join Features: Ponds
c. Output: WUwithPondVegAll_join
d. Join: One-to-one
e. Field Map: retain fields:
   i. Shape_Length
   ii. Shape_Area
   iii. VegByLP (short integer)
   iv. WUarea (float)
   v. PondArea (float)
   vi. PondRatio (float)
f. Match Option: Contains


Spatial Join

Target Features: WUwithPondVegAll
Join Features: Ponds
Output Feature Class: M:\projects\Watershed_JC\WU_pond_join1.shp
Join Operation (optional): JOIN_ONE_TO_ONE
☑ Keep All Target Features (optional)
Field Map of Join Features (optional):
- OBJECTID_1 (Long)
- Acres (Double)
- Shape_Leng (Double)
- Shape_Area (Double)
- FlowPath (Text)
- PerInt (Text)
- WFlowPath (Text)
- Perimeter (Double)
- SquareMete (Double)
- VegByLP (Short)
- VegByLPrat (Double)
- WUVegArea (Float)
- PondArea (Float)

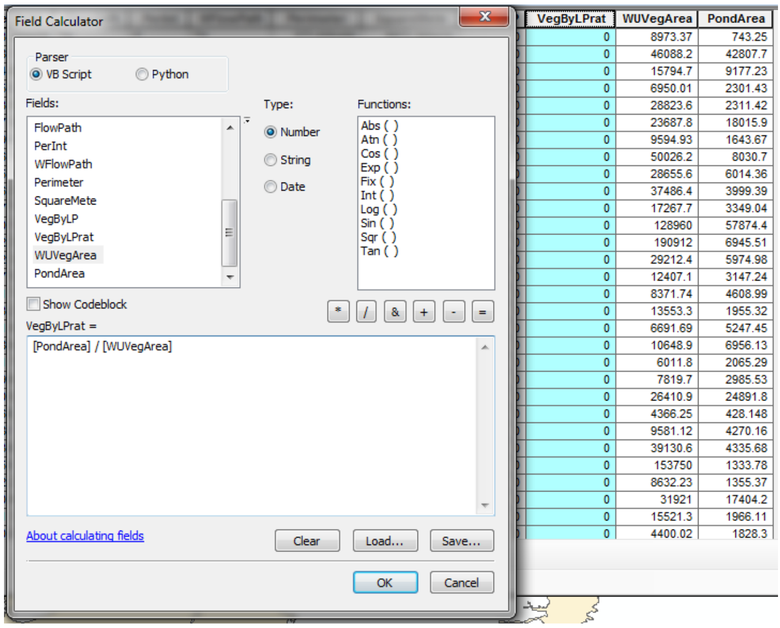
Match Option (optional): CONTAINS
Search Radius (optional): Meters
Distance Field Name (optional)

Spatial Join

Joins attributes from one feature to another based on the spatial relationship. The target features and the joined attributes from the join features are written to the output feature class.

## Calculate ratio of pond area (Ponds) to vegetated Wetland Unit (WUwithPondVegAll_join)

Right click PondRatio in attribute field of WUwithPondVegAll_join
Field Calculate PondRatio = [PondArea] / [WUarea]


Field Calculator

Parser: (●) VB Script ( ) Python

Fields: FlowPath, PerInt, WFlowPath, Perimeter, SquareMete, VegByLP, VegByLPrat, WUVegArea, PondArea

Type: (●) Number ( ) String ( ) Date

Functions: Abs ( ), Atn ( ), Cos ( ), Exp ( ), Fix ( ), Int ( ), Log ( ), Sin ( ), Sqr ( ), Tan ( )

☐ Show Codeblock

VegByLPrat =
```
[PondArea] / [WUVegArea]
```

VegByLPrat	WUVegArea	PondArea
0	8973.37	743.25
0	46088.2	42807.7
0	15794.7	9177.23
0	6950.01	2301.43
0	28823.6	2311.42
0	23687.8	18015.9
0	9594.93	1643.67
0	50026.2	8030.7
0	28655.6	6014.36
0	37486.4	3999.39
0	17267.7	3349.04
0	128960	57874.4
0	190912	6945.51
0	29212.4	5974.98
0	12407.1	3147.24
0	8371.74	4608.99
0	13553.3	1955.32
0	6691.69	5247.45
0	10648.9	6956.13
0	6011.8	2065.29
0	7819.7	2985.53
0	26410.9	24891.8
0	4366.25	428.148
0	9581.12	4270.16
0	39130.6	4335.68
0	153750	1333.78
0	8632.23	1355.37
0	31921	17404.2
0	15521.3	1966.11
0	4400.02	1828.3

## Assign 1 point to VegByLP if pond area is <100% of Wetland Unit area.
Clear all selections
SELECT * FROM WUwithPondVegAll_join WHERE: "PondRatio" < 1
Field calculate VegByLP = 1

## PART 4: COMBINE RESULTS
## Join lake results to Wetland Units

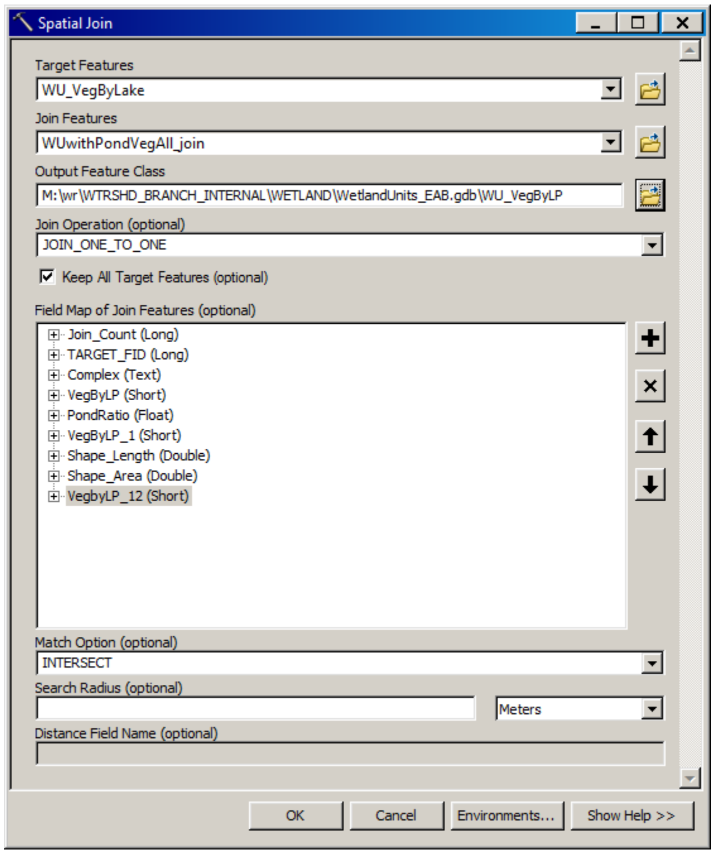
Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: WUbyLakeVegAll
Output Feature Class: WU_VegByLake
Join operation: JOIN_ONE_TO_ONE
Checkbox "Keep all target features"
Match Option: INTERSECT

## Select lake results and write to Wetland Units

SELECT * FROM WU_VegByLake WHERE: "VegByLP_1" = 1 (93 out of 43214 selected)
R-click VegByLP and Field Calculate VegByLP = 1
Clear all selections.

## Join pond results to Wetland Units

Analysis Tools / Overlay / Spatial Join
Target Features: WU_VegByLake
Join Features: WUwithPondVegAll_join
Output Feature Class: WU_VegByLP1
Join operation: JOIN_ONE_TO_ONE
Checkbox "Keep all target features"
Match Option: INTERSECT


Spatial Join

Target Features
WU_VegByLake

Join Features
WUwithPondVegAll_join

Output Feature Class
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits_EAB.gdb\WU_VegByLP

Join Operation (optional)
JOIN_ONE_TO_ONE

☑ Keep All Target Features (optional)

Field Map of Join Features (optional)
- Join_Count (Long)
- TARGET_FID (Long)
- Complex (Text)
- VegByLP (Short)
- PondRatio (Float)
- VegByLP_1 (Short)
- Shape_Length (Double)
- Shape_Area (Double)
- VegbyLP_12 (Short)

Match Option (optional)
INTERSECT

Search Radius (optional)
Meters

Distance Field Name (optional)

## Select pond results and write to Wetland Units

SELECT * FROM WU_VegByLP1 WHERE:
"VegByLP_12" = 1
R-click VegByLP1 and Field Calculate VegByLP = 1
Clear all selections.

## Delete unnecessary fields from WU_VegByLP1

Delete VegByLP_1 and VegByLP_12 from the WU_VegByLP1 feature class. This data has been consolidated into the VegByLP field and is no longer needed.

## PART 5: Add vegetated wetlands that contain a through-flowing perennial stream.
## Spatial Join WU_VegByLP1 to WU_WFlowPath

ArcToolBox / Analysis Tools / Spatial Join
Target features: WU_VegByLP1
Join features: WU_WFlowPath
Output feature class: WU_VegByLP
Join operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field map of join features:
VegByLP
PondRatio
Shape_Length
Shape_Area
WFlowPath

WUKey
Match Option: CONTAINS

## Select vegetated wetlands

Select by Location
Selection method: select features from
Target layer: WU_VegByLP
Source layer: VegAll
Spatial selection method: intersect the source layer feature

## Selected vegetated wetlands the contain a through-flowing perennial stream

Open attribute table of WU_VegByLP
Select by Attributes
Method: Select from current selection
SELECT * FROM WU_VegByLP WHERE: WFlowPath = 'TP'

## Assign points to VegByLP

Open attribute table of WU_VegByLP
R-click VegByLP and Field Calculate VegByLP = 1

5.6.76 VegFA: Vegetation

Version date: 16 November 2016

Strategy: Completed 3/24/2016 EAB
GIS method: Completed & verified 3/25/2016 EAB
Python coding: started 6/10/2016; completed 12/16/2016
Final review by EAB: 12/19/2016

Purpose:

Input to Flood Attenuation / Potential
Max 9 points for floodplain Wetland Units; 5 points for groundwater Wetland Units.

Description:

Rationale: Plants enhance flood attenuation by physically impeding flows, creating microtopographic depressions to store water, and by actively taking up water through their root systems.
Plants that persist throughout the year and provide a complex vertical structure to slow overland flows (live or dead trees, shrubs, and persistent herbs) enhance flood attenuation. However, their effectiveness is reduced if the plants are grazed or mowed to less than 6 inches in height, since their low stature offers little resistance to flood flows. Aquatic bed plants play a smaller role in slowing floods. Forest vegetation provides high interception and evapotranspiration during rainfall events. In floodplains, forest vegetation is particularly effective at slowing flows and providing temporary storage due to the structural complexity of tree trunks and branches, coarse woody debris and microtopographically complex root structures.
Summary of strategy: Sum the points for VegAll (1 point), VegPerUng (4 points max), VegWoody (4 points max) for each Wetland Unit. If the total for groundwater wetlands is greater than the maximum allowed points (5), reduce to the allowed amount. There is no point reduction for floodplain wetlands.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- VegAll (All Vegetation Types)
- VegPerUng4 (Persistent Ungrazed Vegetation)
- VegWoody4 (Woody Vegetation)
- Floodplain

Method:

## Spatial joins to add input variables to Wetland Units attribute table
ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_20150514
Join Feature: WU_Floodplain

Output Feature Class: WU_VegFA1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
Floodplain
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegFA1
Join Feature: WU_VegAll
Output Feature Class: WU_VegFA2
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
Floodplain
VegAll
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegFA2
Join Feature: WU_VegPerUng
Output Feature Class: WU_VegFA3
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
Floodplain
VegAll
VerPerUng4
Match option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegFA3
Join Feature: WU_VegWoody
Output Feature Class: WU_VegFA
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
Floodplain

VegAll
VerPerUng4
VegWoody4
Match option: CONTAINS

## Add VegFA field to Wetland Units and set initial point value to zero.

Open attribute table of WU_VegFA
Add field “VegFA” (short integer)
R-click VegFA and Field Calculate VegFA = 0

## Sum the points for VegAll, VegPerUng4, VegWoody4.

R-click VegFA and Field Calculate “VegFA” = [VegAll] + [VegPerUng4] + [VegWoody4]

## Reduce any excess point scores for groundwater wetlands to the maximum allowed.

Clear all selections.
SELECT * FROM WU_VegFA WHERE: "VegFA" > 5 AND “Floodplain” = ‘N’
Field Calculate (selection only) “VegFA” = 5

5.6.77 VegFQ: Vegetation Floristic Quality

Version date: 26 September 2017

Strategy: completed 3/16/2017 EAB
GIS method: completed 9/20/2017 EAB; verified 9/24/2017 EAB
Python code: 9/25/2017 YH
Final review by EAB: 9/26/2017

Purpose:

Input to Habitat and Ecological Integrity / Intrinsic Potential
Max 9 points.

Description:

Rationale**:** Floristic quality assessment (FQA) evaluates the ecological condition and integrity of natural habitats based on the plant species that grow in them. Each species is characterized by a Coefficient of Conservatism (CoC) based on its tolerance of disturbance and its fidelity to intact natural habitats (Swink and Wilhelm 1994, Wilhelm and Masters 1999). CoC values have been assigned to all species in the West Virginia flora (Rentch and Anderson 2006, WVDNR 2015). The assemblages of plant species present in a wetland reflects the potential number of niches available for invertebrates, birds, and mammals (Bourdaghs 2014, Hruby et al. 2000, Knops et al. 1999). Plant biodiversity affects fundamental ecosystem processes such as nutrient dynamics, autotrophic production, susceptibility to invasive species and fungal disease, richness and structure of insect communities, and the overall integrity and functioning of ecosystems (Knops et al. 1999, Fennessy et al. 1998). Excessive nutrients, particularly total P and NO_3-NO_2-N, have been significantly correlated with lower floristic quality (Fennessy et al. 1998).

Strategy:
Proxies must be used for GIS estimation of floristic quality; this attribute is overwritten by field assessment data. Assign points as follows for each criterion below; sum is capped at 9 points:

- Persistent ungrazed vegetation (VerPerUng1, >50% veg, 9695 wetlands) = 1 point
- Forested wetlands (VegWoodyFor in VegWoody; 3 points (1487 wetlands); 2 points (960 wetlands); 1 point (498 wetlands); 0 points (40179 wetlands)
- PEM on marl. 3 points: PEM on marl > 1 ha in extent (30 wetlands); 2 points: PEM on marl comprises > 50% of wetland (20 wetlands); 1 point: PEM on marl > 200 m2 in extent (19 wetlands)
- Deep organic soils (Histosol) present in wetland. Histosol = 3 points (291 wetlands), histic epipedon = 2 points (310 wetlands)
- Karst area (limestone/dolomite bedrock geology or SSURGO karst) > 0.67 of total wetland area = 3 points; karst 0.33-0.67 = 2 points; > 0.1 = 1 point
- Buffer disturbance is known to be correlated with floristic quality (Fennessy et al 1998). Disturb50m normally would go in "landscape opportunity", but since this is a way to discern floristic quality through GIS, it is also used here. *Evaluate weighting by comparing final composite scores for Floristic Quality. Tentatively, no buffer disturbance = 3 points, Dist50mRat < 10% buffer disturbance = 2 points, < 25% buffer disturbance = 1 point.*

- Landscape integrity index. This belongs in the "landscape opportunity" aspect, but it is also a way to discern floristic quality through GIS, and is used here. *Evaluate weighting by comparing final composite scores for Floristic Quality. LandInteg > 800 = 3 points, 700-800 = 2 points, 600-700 = 1 point*

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegPerUng.gdb
  - Feature Class: WU_VegPerUng
    - Field: VegPerUng1
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegWoody.gdb
  - Feature Class: WU_VegWoody
    - Field: VegWoodyFor
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\MarlPEM.gdb
  - Feature Class: WU_MarlPEM
    - Field: MarlPEM
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\Histosol.gdb
  - Feature Class: WU_Histosol
    - Field: Histosol
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\Karst.gdb
  - Feature Class: WU_Karst
    - Field: Karst
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\Disturb50m.gdb
  - Feature Class: WU_Disturb50m
    - Field: Dist50mRat
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\LandInteg.gdb
  - Feature Class: WU_LandInteg
    - Field: LandInteg
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegAll.gdb
  - Feature Class: WU_VegAll
    - Field: VegArea

Method:

Note: the Spatial Joins below could also be done as Joins on the WUKey field if that is easier.

## Spatial join to merge VegPerUng1 and VegWoodyFor metrics

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegPerUng
Join Feature: WU_VegWoody
Output feature class: WU_VegFQ1
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length

Shape_Area
VegPerUng1
VegWoodyFor
Match Option: CONTAINS

## Spatial join to merge MarlPEM metric

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ1
Join Feature: WU_MarlPEM
Output feature class: WU_VegFQ2
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Match Option: CONTAINS

## Spatial join to merge Histosol metric

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ2
Join Feature: WU_Histosol
Output feature class: WU_VegFQ3
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Histosol
Match Option: CONTAINS

## Spatial join to merge Karst metric

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ3
Join Feature: WU_Karst
Output feature class: WU_VegFQ4
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following

WUKey
    Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Histosol
Karst
Match Option: CONTAINS

## Spatial join to merge Dist50mRat

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ4
Join Feature: WU_Disturb50m
Output feature class: WU_VegFQ5
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Histosol
Karst
Dist50mRat
Match Option: CONTAINS

## Spatial join to merge LandInteg metric

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ5
Join Feature: WU_LandInteg
Output feature class: WU_VegFQ6
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
    Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Histosol
Karst
Dist50mRat

LandInteg
Match Option: CONTAINS

## Spatial join to merge VegArea

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegFQ6
Join Feature: WU_VegAll
Output feature class: WU_VegFQ
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
VegPerUng1
VegWoodyFor
MarlPEM
Histosol
Karst
Dist50mRat
LandInteg
VegArea
Match Option: CONTAINS

## Add fields to store Dist50mFQ and VegFQ and set initial values to zero

Open attribute table of WU_VegFQ
Add fields Dist50mFQ (short integer) and VegFQ (short integer)
Field Calculate Dist50mFQ = 0
Field Calculate VegFQ = 0

## Assign points to Dist50mFQ

SELECT * FROM WU_VegFQ WHERE Dist50mRat < 0.25
Field Calculate Dist50mFQ = 1

SELECT * FROM WU_VegFQ WHERE Dist50mRat < 0.1
(11343 out of 43124 selected)
Field Calculate Dist50mFQ = 2

SELECT * FROM WU_VegFQ WHERE Dist50mRat = 0
Field Calculate Dist50mFQ = 3

## Sum all points for VegFQ

Open attribute table of WU_VegFQ

Field Calculate VegFQ = [VegPerUng1] + [VegWoodyFor] + [MarlPEM] + [Histosol] + [Karst] + [Dist50mFQ] + [LandInteg]

## Reduce VegFQ values to cap of 9 points total

Open attribute table of WU_VegFQ
SELECT * FROM WU_VegFQ WHERE VegFQ > 9
Field Calculate VegFQ = 9

## Reduce VegFQ values to zero for unvegetated wetlands

Open attribute table of WU_VegFQ
SELECT * FROM WU_VegFQ WHERE VegArea IS NULL
Field Calculate VegFQ = 0

5.6.78 VegH: Vegetation Structure and Quality

Version date: 24 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: 9/24/2017 EAB completed and results verified
Python coding: 9/25/2017 YH
Final review by EAB: 9/26/2017

Purpose:

Input to Habitat & Ecological Integrity / Potential
Max 15 points
Rationale: Vegetation is an outstanding proxy for overall biodiversity and habitat quality.
Strategy: Sum the points for all vegetation metrics.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegVerStr.gdb
  - Feature Class: WU_VegVerStr
    - Field: VegVerStr
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\Microtopo.gdb
  - Feature Class: WU_Microtopo
    - Field: VegHorInt
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegFQ.gdb
  - Feature Class: WU_VegFQ
    - Field: VegFQ

Method:

Note that the Spatial Joins could be replaced by Joins on WUKey if that is easier.

## Spatial join to merge VegVerStr and VegHorInt into one attribute table

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegVerStr
Join Feature: WU_Microtopo
Output Feature Class: WU_VegH1
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
Shape_Length
Shape_Area
VegVerStr
VegHorInt
Match option: CONTAINS

## Spatial join to merge VegFQ

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegH1
Join Feature: WU_VegFQ
Output Feature Class: WU_VegH
Join operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following:
WUKey
    Shape_Length
Shape_Area
VegVerStr
VegHorInt
VegFQ
Match option: CONTAINS

## Add VegH field to Wetland Units and set initial point value to zero.

Open attribute table of WU_VegH
Add field “VegH” (short integer)
R-click VegH and Field Calculate VegH = 0

## Sum the points for VegVerStr, VegHorInt, VegFQ

R-click VegH and Field Calculate “VegH” = [VegVerStr] + [VegHorInt] + [VegFQ]

5.6.79 VegHorInt: Horizontal Interspersion

Version date: 15 September 2017

Strategy: Completed 3/16/2017 EAB
GIS method: Completed 9/15/2017 EAB ; Verified 9/15/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Potential / Vegetation
Max 3 points.

Description:

Rationale: Interspersion of different habitat types provides multiple niches for species and is important in supporting biodiversity. Interspersion tends to be greater in larger wetlands, and serves as a partial proxy for wetland size (Adamus et al. 2010a, CWMW 2013, Hruby 2012, Mack 2001).
Strategy: Divide the perimeter of summed NWI (all) polygon perimeters by the square root of the Wetland Unit area. Then, compare this ratio to the number of NWI polygons in the wetland (part of method is the same as Flood Attenuation/Microtopo but with different point spread and addition of count of NWI polygons).

- 3 points: ratio > 10 AND at least 5 NWI polygons present
- 2 points: ratio > 6 AND at least 3 NWI polygons present
- 1 point: ratio > 4 AND at least 2 NWI polygons present
- 0 points: ratio <= 4 OR only 1 NWI polygon present

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\Microtopo.gdb
    - Feature Class: WU_Microtopo

Method:

## Add new attribute field to store points for VegHorInt and set initial value to zero.

In WU_Microtopo, add field: VegHorInt (short integer)
Field calculate VegHorInt = 0

## Assign points to Wetland Units for VegHorInt

SELECT * FROM WU_Microtopo WHERE: "MicroRatio" > 4 AND "Join_Count" > 1
R-click VegHorInt, Field Calculate VegHorInt = 1

SELECT * FROM WU_Microtopo WHERE: "MicroRatio" > 6 AND "Join_Count" > 2
R-click VegHorInt, Field Calculate VegHorInt = 2

SELECT * FROM WU_Microtopo WHERE: "MicroRatio" > 10 AND "Join_Count" > 4
R-click VegHorInt, Field Calculate VegHorInt = 3

5.6.80 VegPerUng, VegPerUng4, VegPerUng1: Persistent ungrazed vegetation

Version date: 14 February 2017

Strategy: completed EAB 14 Feb 2016
GIS method: completed EAB/CMA 2/27/2016, revised to include VegPerUng4 and VerPerUng1 on 4/13/2016 EAB; verified 4/14/2016 EAB
Python coding: VegPerUng Python coding completed 3/1/2016 MCA; VegPerUng4 Python coding completed 5/24/2016 MCA; VegPerUng1 Python coding completed 5/24/2016 MCA
QAPP updated: 2/27/2016 (PasturesNotHayfields source metadata added)
Final review by EAB: 3/2/2016, revision approved 10/4/2016

Purpose:

VegPerUng: Used in Water Quality Function / Potential / Vegetation. Max 5 points.
VegPerUng4: Used in Flood Attenuation Function / Potential / Vegetation. Max 4 points.
VegPerUng1: Used in Habitat and Ecological Integrity Function / Potential / Vegetation. Max 1 point.

Description:

VegPerUng rationale (water quality): Persistent, ungrazed vegetation improves water quality by acting as a filter to trap sediment, nutrients, and pollutants.
VegPerUng strategy (water quality): Sum the persistent ungrazed vegetation based on the Cowardin attribute in the Enhanced National Wetlands Inventory. Erase the mapped grazed pastures from this layer. Calculate the percentage of persistent ungrazed vegetation for each Wetland Unit, and assign points as follows:

- 5 points: VegPerUng is >2/3 of Wetland Unit area
- 3 points: VerPerUng is 1/3 to 2/3 of Wetland Unit area
- 1 point: VegPerUng is 1/10 to 1/3 of Wetland Unit area
- 0 points: VegPerUng is <1/10 of Wetland Unit area

Note that this variable could be improved by a better estimation of grazed areas. These are currently available for only a few parts of the state. James Summers has provided a few mapped watersheds, used in this variable. He has also provided estimates of the percentage of NLCD grassland that is probably pasture statewide. This latter data has not yet been incorporated into the VegPerUng variable, but could be used in the future.

VegPerUng4 rationale (flood attenuation): Persistent, ungrazed vegetation enhances flood attenuation by physically impeding flows, creating microtopographic depressions to store water, and by actively taking up water through root systems.
VerPerUng4 strategy (flood attenuation): Sum the persistent ungrazed vegetation based on the Cowardin attribute in the Enhanced National Wetlands Inventory. Erase the mapped grazed pastures from this layer. Calculate the percentage of persistent ungrazed vegetation for each Wetland Unit, and assign points as follows:

- 4 points: VegPerUng is >2/3 of Wetland Unit area
- 3 points: VerPerUng is 1/2 to 2/3 of Wetland Unit area
- 2 points: VegPerUng is 1/3 to 1/2 of Wetland Unit area

- 1 point: VegPerUng is 1/10 to 1/3 of Wetland Unit area
- 0 points: VegPerUng is <1/10 of Wetland Unit area

VegPerUng1 rationale (habitat and ecological integrity): Persistent, ungrazed vegetation uses carbon from the atmosphere and stores carbon in above- and below-ground biomass. This biomass provides important habitat and energy to organisms.
VegPerUng1 strategy (habitat and ecological integrity): Sum the persistent ungrazed vegetation based on the Cowardin attribute in the Enhanced National Wetlands Inventory. Erase the mapped grazed pastures from this layer. Calculate the percentage of persistent ungrazed vegetation for each Wetland Unit, and assign points as follows:

- 1 point: VegPerUng is 1/2 or more of Wetland Unit area
- 0 points: VegPerUng is <1/2 of Wetland Unit area

Source data:

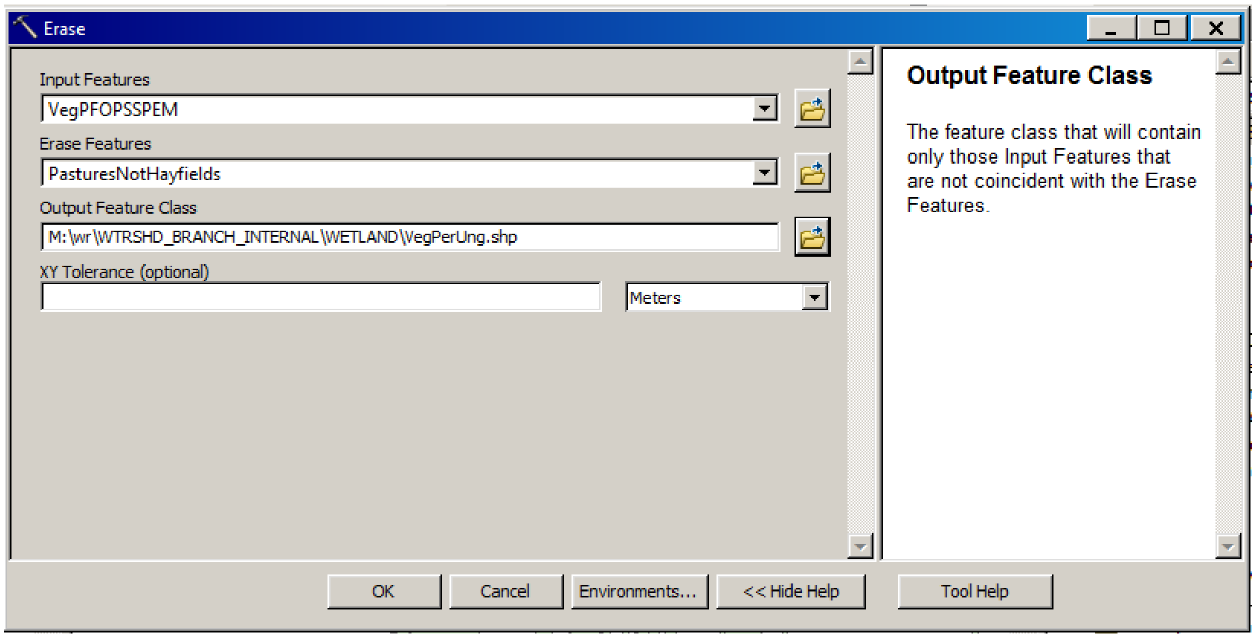
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb
  Feature Dataset: CONUS_WVWetlandsProj
  Feature Class: **EnhWVWetland**
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  Feature Class: **PasturesNotHayfields**
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  Feature Class: **WU_20150514**

Method:

Select the forests, shrublands, and persistent emergent vegetation.
SELECT * FROM EnhWVWetland WHERE:
"ATTRIBUTE" NOT LIKE 'PEM2%' AND ("ATTRIBUTE" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%')

Create layer from selection. Export data to geodatabase.
This intermediate data can be called VegPFOPSSPEM

Erase the known grazed pastures from the intermediate vegetation layer.
ArcToolBox / Analysis / Overlay / Erase
Input: VegPFOPSSPEM
Erase: PasturesNotHayfields
Output: VegPerUng


Input Features
VegPFOPSSPEM
Erase Features
PasturesNotHayfields
Output Feature Class
M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\VegPerUng.shp
XY Tolerance (optional)
Meters

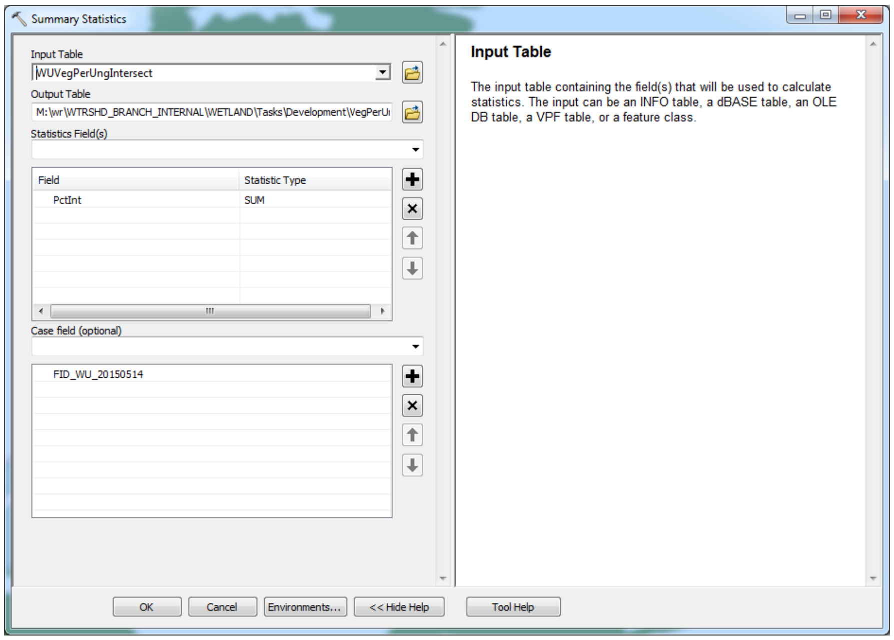
Output Feature Class

The feature class that will contain only those Input Features that are not coincident with the Erase Features.

Calculate the percentage of each Wetland Unit that is persistent ungrazed vegetation (VegPerUng)

Attribute each Wetland Unit with the percentage of VerPerUng.

- Add new field (OrigArea) to the Wetland Unit attribute table.
    - This will retain the original area to be used to calculate the percent after the intersection has occurred.
- Right click on the (OrigArea) field and FieldCalculate where OrigArea = [Shape_Area]
- Intersect the Wetland Units data with VegPerUng
    - The output feature class is "WUVegPerUngIntersect"
- Add new field (PctIntersect) to the WUVegPerUngIntersect attribute table.
- Right click on the (PctIntersect) field and FieldCalculate where PctIntersect = !SHAPE_Area!/!OrigArea!*100
    - Please note, the code above is python so please check the python radio button before executing the FieldCalculate function.
- Execute the Summary Statistics function



Summary Statistics

Input Table: WUVegPerUngIntersect

Output Table: M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Development\VegPerU

Statistics Field(s)

Field	Statistic Type
PctInt	SUM

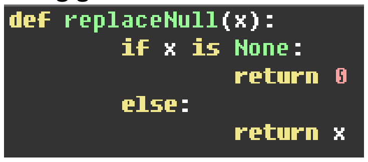
Case field (optional)

FID_WU_20150514

Input Table

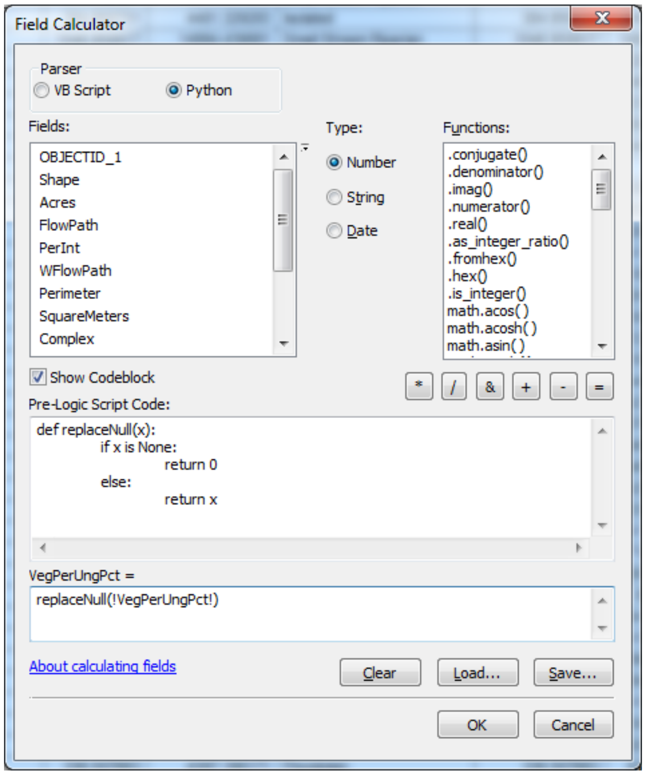
The input table containing the field(s) that will be used to calculate statistics. The input can be an INFO table, a dBASE table, an OLE DB table, a VPF table, or a feature class.

- This function creates a table (WUVegPerUngIntersect_SUM_STAT) that contains the sums of the percentages for each Wetland Unit making it easier to join back to the Wetland Units data, copy the data over, and then assign points based on the percentages.

- Add new field (VegPerUngPct) to attribute table to store the percentage of persistent ungrazed vegetation for each Wetland Unit.
- Set the Null values to zero in VegPerUngPct.
- Add new field (VegPerUng) to attribute table to store the score in relation to the percentage of persistent ungrazed vegetation for each Wetland Unit.
- Set the Null values to zero in VegPerUng.
- Join Wetland Units to the WUVegPerUngIntersect_SUM_STAT table.
- Right click on the (VegPerUngPct) field and FieldCalculate where VegPerUngPct = !WUVegPerUngIntersect_SUM_STAT.SUM_PctIntersect!
  - Please note, the code above is python so please check the python radio button before executing the FieldCalculate function.
- Remove the join to WUVegPerUngIntersect_SUM_STAT on the Wetland Units data.
- As a result of copying the percentages to the Wetland Units data, null values might be carried over. To fix this follow these steps:
  - Right click on the (VegPerUngPct) field and select Field Calculator
  - Make sure the python radio button is checked.
  - Check “Show Codeblock”
  - The following goes into the Codeblock textarea
    - 

```
def replaceNull(x):
        if x is None:
                return 0
        else:
                return x
```

  - VegPerUngPct = replaceNull(!VegPerUngPct!).
  - Now click ok



Parser: Python

Show Codeblock [✓]

Pre-Logic Script Code:

```
def replaceNull(x):
        if x is None:
                return 0
        else:
                return x
```

VegPerUngPct =

```
replaceNull(!VegPerUngPct!)
```

○

Assign points to Wetland Units for VegPerUng.

SELECT * FROM WU_20150514 WHERE: "VegPerUngPct" > 66.7
(R-click "VegPerUng" and Field Calculate "VegPerUng" = 5
Clear selections.

SELECT * FROM WU_20150514 WHERE: "VegPerUngPct" < 66.701 AND "VegPerUngPct" > 33.3
R-click "VegPerUng" and Field Calculate "VegPerUng" = 3
Clear selections.

SELECT * FROM WU_20150514 WHERE: "VegPerUngPct" < 33.301 AND "VegPerUngPct" > 10
R-click "VegPerUng" and Field Calculate "VegPerUng" = 1
Clear selections.

SELECT * FROM WU_20150514 WHERE: "VegPerUngPct" < 10.001
R-click "VegPerUng" and Field Calculate "VegPerUng" = 0
Clear selections.

## Export Wetland Units to WU_VegPerUng if this has not already been done.

## Add VegPerUng4 and VerPerUng1 to attribute table and set initial points to zero.

Open attribute table of WU_VegPerUng and add field VegPerUng4 and VerPerUng1 (short integers)
R-click VegPerUng4 and Field Calculate VegPerUng4 = 0
R-click VegPerUng1 and Field Calculate VegPerUng1 = 0

## Assign points to Wetland Units for VegPerUng4 and VegPerUng1

SELECT * FROM WU_VegPerUng WHERE: "VegPerUngPct" > 10
R-click “VegPerUng4” and Field Calculate “VegPerUng4” = 1

SELECT * FROM WU_VegPerUng WHERE: "VegPerUngPct" > 33.
R-click “VegPerUng4” and Field Calculate “VegPerUng4” = 2

SELECT * FROM WU_VegPerUng WHERE: "VegPerUngPct" > 50
R-click “VegPerUng4” and Field Calculate “VegPerUng4” = 3
R-click “VegPerUng1” and Field Calculate “VegPerUng1” = 1

SELECT * FROM WU_VegPerUng WHERE: "VegPerUngPct" > 66.7
R-click “VegPerUng4” and Field Calculate “VegPerUng4” = 4

5.6.81 VegVerStr: Vegetation Vertical Structure

Version date: 18 September 2017

Strategy: completed 3/16/2017 EAB
GIS method: completed 9/14/2017 EAB; verified 9/14/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Potential / Vegetation Factor. Max 3 points.

Description:

Rationale for VegVerStr: Plant communities with complex vertical structure offer enhanced habitat niches for a variety of plants and animals.
Strategy for VegVerStr: Assign points based on the ratio of forest and vegetated classes to total area; minimum polygon size 0.1 acre (only count those codes whose aggregated area >= 0.05 ha).

- 3 points: PFO > 50% of wetland AND PFO >=0.05 ha
- 2 points: (PFO >5% of wetland AND PFO >=0.05 ha) AND vegetated classes (PFO+PSS+PEM+PAB+PML) > 50% of wetland
- 1 point: vegetated classes (PFO+PSS+PEM+PAB+PML) > 5% of wetland AND vegetated classes >=0.05 ha
- 0 point: none of the above criteria are met

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegWoody.gdb
  - Feature Class: WU_VegWoody
    - Fields: PFOarea, PFOratio
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\Tasks\Complete\VegAll.gdb
  - Feature Class: WU_VegAll
    - Fields: VegArea, VegRatio

Method:

## Retrieve the fields PFOarea (forest area in m^2) and PFOratio (ratio of forest area to total wetland
## area) from the WU_VegWoody feature class.
## Retrieve the fields VegArea (vegetated area in m^2) and VegRatio (ratio of vegetated area to total
## wetland area) from the WU_VegAll feature class.
## Store the fields in a new feature class: WU_VegVerStr.

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_VegWoody
Join Feature: WU_VegAll

Output feature class: WU_VegVerStr
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
WUKey
Shape-Length
Shape_Area
PFOarea
PFOratio
VegArea
VegRatio
Match Option: CONTAINS

## Add new attribute field to store points for VegVerStr and set initial value to zero.

In WU_VegVerStr, add field: VegVerStr (short integer)
Field calculate VegVerStr = 0

## Assign points to Wetland Units for VegVerStr

SELECT * FROM WU_VegVerStr WHERE: "VegRatio" > 0.05 AND "VegArea" > 500
R-click VegVerStr, Field Calculate VegVerStr = 1

SELECT * FROM WU_VegVerStr WHERE: "PFOratio" > 0.05 AND "PFOarea" > 500 AND "VegRatio" > 0.5
R-click VegVerStr, Field Calculate VegVerStr = 2

SELECT * FROM WU_VegVerStr WHERE: "PFOratio" > 0.5 AND "PFOarea" > 500
R-click VegVerStr, Field Calculate VegVerStr = 3

Background notes: (not for coding)

Steps skipped by accessing already-computed values:

## Select the forest vegetation
## (Note that the PFOratio and PFOarea were created in WU_VegWoody and could be accessed from there, thus skipping multiple steps)

Clear Selection
SELECT * FROM ENHWVWetland WHERE: "ATTRIBUTE" LIKE 'PFO%'

## Create layer of forest vegetation from selection.

R-click EnhWVWetland / Data / Export Data
Export: Selected features

Output feature class: “VegPFO”
Clear Selection

## Add field to store forest area

Open attribute table of VegPFO
Add field PFOarea (float)
R-click PFOarea and Field Calculator: PFOarea = [SHAPE_Area]

## Select the forest, shrubland, emergent, moss, and aquatic bed vegetation.
## (Note that VegArea and VegRatio were created in WU_VegAll and could be accessed from there, thus skipping multiple steps)

Clear all selections
SELECT * FROM EnhWVWetland WHERE:
"ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%' OR "ATTRIBUTE" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PML%'

## Create layer of all vegetation from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: “VegAll”

## Add field to store vegetation area

Open attribute table of VegAll
Add field VegArea (float)
R-click VegArea and Field Calculator: VegArea = [SHAPE_Area]

## Join forests to wetland units and sum the forest area

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: VegPFO
Output Feature Class: WUPFOjoin
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features
R-click PFOarea and select “Merge Rule”, “Sum”
Match Option: INTERSECT

## Join vegetated classes and sum area

Analysis Tools / Overlay / Spatial Join

Target Features: WUPFOjoin
Join Features: VegAll
Output Feature Class: WU_VegVerStr
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features
R-click VegArea and select “Merge Rule”, “Sum”
Match Option: INTERSECT

## Add fields to store ratios of forest or vegetated area to total area

In WU_VegVerStr, add two fields: PFOratio (float) and VegRatio (float)

## Calculate ratio of forest vegetation to Wetland Unit area.

In WU_VegVerStr, R-click attribute PFOratio
Field Calculate PFOratio = [PFOarea] / [Shape_Area]

## Calculate ratio of vegetated classes to Wetland Unit area.

In WU_VegVerStr, R-click attribute VegRatio
Field Calculate VegRatio = [VegArea] / [Shape_Area]

5.6.82 VegWoody: Woody Vegetation

Version date: 14 February 2017

Strategy: completed 2/27/2016 EAB
GIS method: completed 3/3/2016 EAB & JCC; verified 3/5/2016 EAB; revision to include VegWoody4 and VegWoody2 on 4/13/2016; new point roll-up at end; verified 4/14/2016 EAB; revision to include VegWoodyFor on 2/14/2017
Python code: started & completed 3/15/2016 MCA; VegWoodyFor revision started & completed 2/15/2017 MCA
Final review by EAB: 3/15/2016; results of revision approved 10/4/2016; VegWoodyFor revision approved 2/15/2017

Purpose:

VegWoody: Input to Water Quality / Potential / Vegetation Factor. Max 5 points. Floodplain wetlands only.
VegWoody4: Input to Flood Attenuation / Potential / Vegetation Factor. Max 4 points. All wetland types.
VegWoody2: Input to Habitat and Ecological Integrity / Potential / Structural Patches. Max 2 points. All wetland types.
VegWoodyFor: Input to Habitat and Ecological Integrity / Potential / Vegetation Factor. Max 3 points. All wetland types.

Description:

<u>Rationale for VegWoody (water quality)</u>: Plants in a riverine wetland will improve water quality by acting as a filter to trap sediments and associated pollutants. The plants also slow the velocity of water which results in the deposition of sediments. Persistent, multi-stemmed plants enhance sedimentation by offering frictional resistance to water flow (review in Adamus et al. 1991). Shrubs and trees are considered to be better at resisting water velocities than emergent plants during flooding and are scored higher. Aquatic bed species or grazed, herbaceous (non-woody) plants are not judged to provide much resistance to water flows.
<u>Strategy for VegWoody</u>: Assign points based on the ratio of forest and shrub cover to total area.

- 5 points: forested wetlands cover more than 2/3 of the Wetland Unit
- 4 points: forest > 1/3 AND shrub/forest > 90%
- 3 points: shrub/forest > 2/3
- 2 points: shrub/forest > 1/3
- 1 point: shrub/forest > 1/10
- 0 point: shrub/forest < 1/10

<u>Rationale for VegWoody4 (flood attenuation)</u>: Woody vegetation provides high interception and evapotranspiration during rainfall events. Woody vegetation in a floodplain slows the velocity of water and offers frictional resistance to water flow (Adamus et al. 1991). Shrubs and trees are considered to be better at resisting water velocities than emergent plants during flooding.

Strategy for VegWoody4: Assign points based on the ratio of forest and shrub cover to total area.

- 4 points: forested wetlands cover more than 2/3 of the Wetland Unit
- 3 points: forest > 1/3 AND shrub/forest > 90%
- 2 points: shrub/forest > 1/2
- 1 point: shrub/forest > 1/10
- 0 point: shrub/forest < 1/10

Rationale for VegWoody2 (habitat and ecological integrity): Woody vegetation provides multiple layers for habitat niches to develop. It is one indicator of structural patch richness.
Strategy for VegWoody2: Assign points based on the ratio of forest and woody shrub cover to total Wetland Unit area.

- 2 points: forested wetlands cover more than 2/3 of the Wetland Unit
- 1 point: forested or shrub wetlands cover at least 10% of the Wetland Unit
- 0 point: shrub/forest < 1/10

Rationale for VegWoodyFor (habitat and ecological integrity): Forested wetlands include many of the highest-quality wetlands in the state, such as conifer peatlands, pin oak swamps, and blackgum swamps. Forested wetlands have many layers of habitat to support a large diversity of organisms. Forested wetlands take decades or centuries to develop and are not easily replaced. Woody vegetation in forested wetlands provides long-term carbon storage in roots, branches, and trunks. Large intact patches of forested wetland have greater habitat value than smaller or fragmented patches.

Strategy for VegWoodyFor: Assign points based on the ratio of forest cover to total wetland area, and size of forest patch.

- 3 points: forested wetlands cover more than 2/3 of the Wetland Unit AND forested wetlands total > 0.5 ha; OR forested wetlands comprise > 5 ha within the Wetland Unit
- 2 points: forested wetlands cover 1/3 to 2/3 of the Wetland Unit AND forested wetlands total > 0.2 ha; OR forested wetlands comprise 2-5 ha
- 1 point: forested wetlands cover 1/10 to 1/3 of the Wetland Unit OR forested wetlands comprise > 1 ha

Definitions:

Cowardin Classification:
PFO: Palustrine Forested
PSS: Palustrine Shrubland

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)

Method:

## Select all the woody vegetation, both forest and shrubland

Clear all selections.
SELECT * FROM ENHWVWetland WHERE: "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%'

## Create layer of woody vegetation from selection

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: "VegPFOPSS"

## Add field to store woody area

Open attribute table of VegPFOPSS
Add field PFOPSSarea (float)
R-click PFOPSSarea and Field Calculator: PFOPSSarea = [SHAPE_Area]

## Select just the forest vegetation, not including the shrubs

Clear Selection
SELECT * FROM ENHWVWetland WHERE: "ATTRIBUTE" LIKE 'PFO%'

## Create layer of forest vegetation from selection.

R-click EnhWVWetland / Data / Export Data
Export: Selected features
Output feature class: "VegPFO"
Clear Selection

## Add field to store forest area

Open attribute table of VegPFO
Add field PFOarea (float)
R-click PFOarea and Field Calculator: PFOarea = [SHAPE_Area]

## Join forests to wetland units and sum the forest area

Analysis Tools / Overlay / Spatial Join
Target Features: WU_20150514
Join Features: VegPFO
Output Feature Class: WUPFOjoin
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features
R-click PFOarea and select "Merge Rule", "Sum"
Match Option: INTERSECT

## Join woody vegetation and sum area

Analysis Tools / Overlay / Spatial Join
Target Features: WUPFOjoin
Join Features: VegPFOPSS
Output Feature Class: WU_VegWoody
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features
R-click PFOPSSarea and select "Merge Rule", "Sum"
Match Option: INTERSECT

## Add fields to store ratios of forest or woody area to total area

In WU_VegWoody, add two fields: PFOratio (float) and PFOPSSratio (float)

## Calculate ratio of woody vegetation to Wetland Unit area.

In WU_VegWoody, R-click attribute PFOPSSratio
Field Calculate PFOPSSratio = [PFOPSSarea] / [Shape_Area]

## Calculate ratio of forest vegetation to Wetland Unit area.

In WU_VegWoody, R-click attribute PFOratio
Field Calculate PFOratio = [PFOarea] / [Shape_Area]

## Add new attribute field to store points for VegWoody and set initial value to zero.

In WU_VegWoody, add field: VegWoody (short integer)
Field calculate VegWoody = 0

## Assign points to Wetland Units for VegWoody

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.1
R-click VegWoody, Field Calculate VegWoody = 1

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.333
R-click VegWoody, Field Calculate VegWoody = 2

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.667
R-click VegWoody, Field Calculate VegWoody = 3

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.667 AND "PFOratio" > 0.333
R-click VegWoody, Field Calculate VegWoody = 4

SELECT * FROM WU_VegWoody WHERE: "PFOratio" > 0.667
R-click VegWoody, Field Calculate VegWoody = 5

## Add new attribute fields to store points for VegWoody4, VegWoody2, and VegWoodyFor and set initial values to zero.

In WU_VegWoody, add field: VegWoody4, VegWoody2, and VegWoodyFor (short integer)
Field calculate VegWoody4 = 0
Field calculate VegWoody2 = 0
Field calculate VegWoodyFor = 0

## Assign points to Wetland Units for VegWoody4, VegWoody2, and VegWoodyFor

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.1
R-click VegWoody4, Field Calculate VegWoody4 = 1
R-click VegWoody2, Field Calculate VegWoody2 = 1

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.5
R-click VegWoody4, Field Calculate VegWoody4 = 2

SELECT * FROM WU_VegWoody WHERE: "PFOPSSratio" > 0.667 AND "PFOratio" > 0.333
R-click VegWoody4, Field Calculate VegWoody4 = 3

SELECT * FROM WU_VegWoody WHERE: "PFOratio" > 0.667
R-click VegWoody4, Field Calculate VegWoody4 = 4
R-click VegWoody2, Field Calculate VegWoody2 = 2

SELECT * FROM WU_VegWoody WHERE: "PFOratio" > 0.1 OR "PFOarea" > 10000

R-click VegWoodyFor, Field Calculate VegWoodyFor = 1

SELECT * FROM WU_VegWoody WHERE: ("PFOratio" > 0.33 AND "PFOarea" > 2000) OR "PFOarea" > 20000

R-click VegWoodyFor, Field Calculate VegWoodyFor = 2

SELECT * FROM WU_VegWoody WHERE: ("PFOratio" > 0.667 AND "PFOarea" > 5000) OR "PFOarea" > 50000

R-click VegWoodyFor, Field Calculate VegWoodyFor = 3

5.6.83 VegWQ: Vegetation

Version date: 14 March 2016

Strategy: Completed 2/27/2016 EAB
GIS method: completed 2/27/2016 EAB, verified 3/14/2016
Python coding: Started 3/23/2016 MCA & completed 3/24/2016 MCA
Final review by EAB:

Purpose:

Input to Water Quality
Max 10 points for floodplain Wetland Units; 5 points for groundwater Wetland Units.
Rationale: Plants enhance sedimentation by acting as a filter, and cause sediment particles to drop to the wetland surface (review in Adamus and others 1991). Plants in wetlands can take on different forms and structures. The intent of this question is to characterize how much of the wetland is covered with plants that persist throughout the year and provide a vertical structure to trap or filter out pollutants (live or dead trees, shrubs, and persistent herbs). It is assumed, however, that the effectiveness at trapping sediments and pollutants is severely reduced if the plants are grazed. Aquatic bed plants are not considered important in sequestering toxic compounds because the toxics will be released in the fall when the plants decompose. NOTE: this question applies only to persistent plants that are not grazed or mowed (or if grazed or mowed, the plants are taller than 6 inches). NOTE for Level 1: To meet the "class" requirement for Cowardin, a polygon of plants within the wetland unit needs at least 30% cover of the specified plants type (forest, shrub, etc.). However, to count the Cowardin polygon as a "plants structure" in the rating system the Cowardin polygon itself has to represent at least 10% of the wetland unit in units that are smaller than 2.5 acres, or at least 1/4 acre in units that are larger.
Summary of strategy: Sum the points for VegPerUng, VegWoody, VegByLP for each Wetland Unit. If the total is greater than the maximum allowed points, reduce to the allowed amount.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- VegPerUng (Persistent Ungrazed Vegetation)
- VegWoody (Woody Vegetation)
- VegByLP (Vegetation fronting Lakes and Ponds)
- Floodplain

Method:

## Create feature class to store VegWQ factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_VegWQ

## Add VegWQ field to Wetland Units and set initial point value to zero.

Open attribute table of WU_VegWQ
Add field “VegWQ” (short integer)
R-click VegWQ and Field Calculate VegWQ = 0

## Spatial join to add input variables to attribute table

Spatial join (contains) to add the following to the WU_VegWQ: VegPerUng, VegWoody, VegByLP, Floodplain

## Sum the points for VegPerUng, VegWoody, and VegByLP.

R-click VegWQ and Field Calculate “VegWQ” = VegPerUng + VegWoody + VegByLP

## Reduce any excess point scores to the maximum allowed.

Clear all selections.
SELECT * FROM WU_VegWQ WHERE: "VegWQ" > 10
Field Calculate (selection only) “VegWQ” = 10

Clear all selections.
SELECT * FROM WU_VegWQ WHERE: "VegWQ" > 5 AND “Floodplain” = ‘N’
Field Calculate (selection only) “VegWQ” = 5

5.6.84 WaterSupply: Wetland discharges to water supply intake area

Version date: 3 October 2016

Strategy: 2/14/2016 EAB
GIS method: completed and verified 3/17/2016 EAB, MCA, and Nate Gunn
Python coding: Nate Gunn 5/24/2016
Final review by EAB: 10/3/2016

Purpose:

Water Quality Function, WaterUse factor
Max 2 points

Description:

Wetland Unit is in the contributing watershed of known water supply, including either a surface water intake or groundwater intake under direct influence of surface water.
Summary of strategy:

- Assign 1 point if the Wetland Unit intersects a polygon in a Zone of Peripheral Concern or a Secondary Protection Area with surface water connections OR if the Wetland Unit makes up 0.1-1% of a Surface Intake Drainage Area.
- Assign 2 points if the Wetland Unit intersects a polygon in a Zone of Critical Concern, Protection Area, or Wellhead Protection Areas where the source is surface water or ground water under the influence of surface water OR if the Wetland Unit makes up more than 1% of a Surface Intake Drainage Area.

Definitions:

FAC_SRC: Type of water source for facility.
SW surface water
GW ground water
GU ground water under the influence of surface water

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\environmental\CONFIDENTIAL-public_surface_water_intakes\CONFIDENTIAL-source_water_assessment_and_protection.gdb
    - Feature Class: ZPC_statewide_5hrabove (ZPC_5_hr_travel)
    - Feature Class: ZCC_statewide (Zone of Critical Concern)
    - Feature Class: Source_Water_Protection_Areas (Conjunctive Delineation, Wellhead Protection Areas)
- M:\environmental\CONFIDENTIAL-public_surface_water_intakes\pswi_distance_analysis_9m.gdb
    - Feature Class: pswi_watersheds_with_out_of_state_drainage (Surface Intake Drainage Area)
- *Note that data is also served publicly at http://tagis.dep.wv.gov/WVWaterPlan/*

Input Variables:

None

Method:

(Python method below documented by Nathan Gunn)
## Create Original Area Field to the Surface Intake Drainage Area
Add Field
Input Table: Surface Intake Drainage Area
Field Name: "OrigArea"
Type: "DOUBLE"

## Store the Shape Area to the Original Area Field
Input Table: Surface Intake Drainage Area
FieldName: "OrigArea"
Expression: "!SHAPE_Area!"
Expression Type: "PYTHON_9.3"

## Intersect the Supplied Wetland Layer and the Surface Intake Drainage Area

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WaterSupply1

## Remove the OrigArea Field
Input Table: Surface Intake Drainage Area
Delete Field: “OrigArea”

## Calculate Intersection Ratios and Scores for Wetlands and Surface Intake Drainage Areas
Pseudo Code:

Let W be all wetlands in the supplied wetland feature class
let A_w be wetland’s area
let A_{wp} be an specific intersection between A_w and a particular public intake drainage area p
let I be the set of intersections of wetlands and public intake drainage areas. For a wetland w, $w \in I$ if w intersects with a drainage area.

```
wetland list = { wetland $\in$ W}
for each wetland in W:
    if wetland $\in$ I:
    wetland.ratio = maximum( {$A_{wp}$ / $A_w$ })
else:
    wetland.ratio = 0
```

$$wetland.score(wetland.ratio) = \begin{cases} 0, & wetland.ratio \leq 0.001 \\ 1, & 0.001 < wetland.ratio \leq 0.01 \\ 2, & wetland.ratio > 0.01 \end{cases}$$

<u>Selection Subroutine: “score_two_by_selection</u>“

(method continues with GIS instructions from Elizabeth Byers)

## Select Secondary Protection Areas with surface water connections.

Select by Attributes
Layer: Source_Water_Protection_Areas (=Conjunctive Delineation)
Method: Create a new selection
SELECT * FROM Source_Water_Protection_Areas WHERE: "P_TYPE" IN (NULL, 'Secondary Protection Area') AND "FAC_SRC" IN ('GU', 'SW')

## Select Wetland Units within a Secondary Protection Area.

Select by location
Selection method: select features from
Target layer: WU_WaterSupply1
Source layer: Source_Water_Protection_Areas (=Conjunctive Delineation)
Check box “Use selected features”
Spatial selection method: intersect the source layer feature

## Select Wetland Units within the 5-hour travel distance in a Zone of Peripheral Concern.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_WaterSupply1
Source layer: <u>ZPC_statewide_5hrabove</u>
Spatial selection method: intersect the source layer feature

(method continues with Python documentation by Nate Gunn)

## Assign 1 point to Wetland Units.
This loop is done in memory to save time and steps.
Pseudo Code:

```
For each wetland_unit in  selection:
wetland_list[ wetland_unit ].score = max(wetland_unit.score, 1)
```

This assigns 1 to each selected item, but does not overwrite scores of 2, if encountered in the previous step.

<u>Selection Subroutine: "score one by selection"</u>

(method continues with GIS documentation by Elizabeth Byers)

## Select Protection Areas and Wellhead Critical Areas with surface water connections

Select by Attributes
Layer: Source_Water_Protection_Areas (=Conjunctive Delineation)
Method: Create a new selection
SELECT * FROM Source_Water_Protection_Areas WHERE: "P_TYPE" IN ('Protection Area', 'rotection Area', 'Wellhead Critical Area') AND "FAC_SRC" IN ('GU', 'SW')

## Select Wetland Units within Protection Areas or Wellhead Critical Areas with surface connections.

Select by location
Selection method: select features from
Target layer: WU_WaterSupply
Source layer: Source_Water_Protection_Areas (=Conjunctive Delineation)
Check box "Use selected features" (38 features selected)
Spatial selection method: intersect the source layer feature

## Select Wetland Units within Zone of Critical Concern.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_WaterSupply
Source layer: ZCC_statewide
Spatial selection method: intersect the source layer feature

(method continues with Python method by Nate Gunn)

## Assign 2 points to Wetland Units within 1 km of special WaterSupply.
This loop is done in memory to save time and steps.
Pseudo Code:

```
For each wetland_unit in  selection:
wetland_list[ wetland_unit .score = max(wetland_unit.score, 2)
```

<u>Output Subroutine:</u>

##Create WU_WATERSUPPLY Feature Class
Create Feature Class
Feature Class Name: WU_WATERSUPPLY
Template Feature Class: pwsi_scoring_output_template

Type: Polygon
SpatialReference: NAD_1983_UTM_Zone_17N (26917)

##Write Output to Feature Class
Pseudocode:

```
For each supplied_wetland in supplied_wetland_feature_class:
   geometry = supplied_wetland.geometry
   id = supplied_wetland.object_id

   new_record = [id, geometry, wetland_units[ id ][ratio], wetland_units[id][score] ]
   WU_WATERSUPPLY.insert(new_record)
```

5.6.85 Wetland Units: Assigning Site Codes to updated statewide Wetland Units

Version date: 8 March 2019

Purpose:

Note that there is no national system for coding wetlands yet, probably because wetland mapping still in a state of flux nationwide. This is a WVDEP protocol based on the nearest stream code.

SiteCode =

1. Nearest stream FINALCODE (to assist with WAB visualization of location) +
2. "_W" (for wetland) +
3. Sequential number based on size of wetland for initial statewide batch, then subsequently based on assessment order. Check existing Wetland Codes and codes assigned in the WVWRAM MS-Access database to ensure that the first available (unused) sequential number is assigned.
4. For replicates, add "_R1" or "_R2" (sequential number as needed)

Here are a few examples:

- PL-63_W14 (unnamed wetland)
- BS-16-A_W1 (unnamed wetland)
- MC-123-DI_W1_R1 (Whitmeadow Run Peatland, first replicate assessment)
- KG-103-BN_W1 (Cranberry Glades on the South Fork Cranberry River)
- KG-103-BN_W4 (unnamed wetland)

Procedure for updating Site Codes for statewide Wetland Units

Create Wetland Units from the updated NWI_WV polygons (see instructions to Create Wetland Units).

Spatial join with existing Wetland Unit Site Codes from M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\WU_WetlCode

Update Site Codes from wetlands in the WVWRAM MS-Access database

For the remaining Wetland Units that do not have a Site Code, follow (or modify as necessary) the Feb 2018 instructions beginning at "Iterate".

Procedure for statewide dataset as done in Feb 2018:
Find nearest stream
ArcToolbox / Proximity / Near
Input: WU_20150514

Proximity to:
M:\wr\WTRSHD_BRANCH\NHD_24K_STREAM_LINES\NHD_StreamMerge_20180202_inWV.shp

Note that this takes a long time to run on the statewide dataset – 4.5 hours

Export Near to WUstream

Add field WetlandCode (text)

Link WUstream to stream FINALCODE and FINALNAME2 fields (NearFID linked to FID)

Iterate:

In WUstream, summarize FINALCODE by Shape_Area (Maximum)

Join Sum_Output table back to WUstream via FINALCODE

Select by Attributes where Shape_Area = Max_Shape_Area

Field Calculate WetlandCode = [FINALCODE]&"_W1"

Select by Attributes where [WetlandCode] IS NULL

Remove all joins

Return to "Iterate" until all wetlands are coded.

Additional notes:

From Chris Daugherty: the most recent updated, as correct as possible for now stream code/name file as of 2/7/2018.

- Most known issues are in Greenbrier and some Upper New areas where there is KARST. If you are working in these areas, we may need to take a look at what is going on there. Steve Stutler knows a little more about the issues there.
- M:\wr\WTRSHD_BRANCH\NHD_24K_STREAM_LINES\NHD_StreamMerge_20180202_inWV.shp

- FINALCODE field for the stream code, to use in building the Wetland Code, and
- include as additional descriptive fields in the attribute table:
    - FINALNAME2 field, and
    - WetlandName assigned by the field surveyor.

5.6.86 WetldBird: Wetland Breeding Bird Areas

Version date: 11 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/4/2017 EAB; results verified 10/4/2017 EAB; changed path to source data 10/11/2017
Python code: 10/9/2017 YH
Final review by EAB: 10/9/2017

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)

Description:

Maximum 3 points
Rationale: Breeding Bird Atlas blocks with high occupancy by wetland-dependent birds provide a strong indicator of extant biodiversity and the presence of high quality wetland habitat. Breeding Bird Atlas blocks comprise approximately 10 square miles, or one-sixth of a USGS topographic quadrangle.

GIS Method (no field method):
This metric is based on occupancy maps for wetland breeding birds from the WV Breeding Bird Atlas project 2017 (Rich Bailey, WVDNR Coordinator).
3 points: wetland intersects atlas block with upper 10% of values (WetBird > 0.493)
2 points: wetland intersects in atlas block with upper 10-50% of values (WetBird > 0.408)
1 point: wetland intersects atlas block in the upper 50-75% of values (WetBird > 0.354)
(Values of "WetBird" from table: upper 5% > 0.525, 20% > 0.466, 50% > 0.408)
Note that Breeding Bird Atlas blocks do not cover the entire state; slivers of the state are missing along the Maryland border.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
    - Feature Class: WU_20150514
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\HabitatData.gdb
    - Feature Class: WetlandBirds_WetBirdColumnOnly

Method:

## Create feature class to store results for WetldBird and set initial value to zero

R-click WU_20150514 and select Data / Export Data / All Features
Output feature class: WetlandFunction.gdb / WU_WetlandBird1

Open attribute table of WU_WetlandBird1
Add field "WetldBird" (short integer)

Field calculate WetldBird = 0

## Select atlas blocks in the upper 75% of wetland bird occupancy values

SELECT * FROM WetlandBirds_WetBirdColumnOnly WHERE: "WetBird" > 0.354

## Select Wetland Units that intersect the selected atlas blocks

Select by Location
Selection method: select features from
Target layer(s): WU_WetlandBird1
Source layer: WetlandBirds_WetBirdColumnOnly
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 1 point

Open attribute table of WU_WetlandBird1
Field Calculate WetldBird = 1

## Select atlas blocks in the upper 50% of wetland bird occupancy values

SELECT * FROM WetlandBirds_WetBirdColumnOnly WHERE: "WetBird" > 0.408

## Select Wetland Units that intersect the selected atlas blocks

Select by Location
Selection method: select features from
Target layer(s): WU_WetlandBird1
Source layer: WetlandBirds_WetBirdColumnOnly
Check “Use selected features”
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 2 points

Open attribute table of WU_WetlandBird1
Field Calculate WetldBird = 2

## Select atlas blocks in the upper 10% of wetland bird occupancy values

SELECT * FROM WetlandBirds_WetBirdColumnOnly WHERE: "WetBird" > 0.493

## Select Wetland Units that intersect the selected atlas blocks

Select by Location
Selection method: select features from

Target layer(s): WU_WetlandBird1
Source layer: WetlandBirds_WetBirdColumnOnly
Check "Use selected features"
Spatial selection method for target layer feature(s): intersect the source layer feature

## Assign 3 points and clear selections

Open attribute table of WU_WetlandBird1
Field Calculate WetldBird = 3

Clear all selections

## Select wetlands that fall outside the atlas block coverage

Select by Location
Selection method: select features from
Target layer(s): WU_WetlandBird1
Source layer: WetlandBirds_WetBirdColumnOnly (no features selected)
Spatial selection method for target layer feature(s): intersect the source layer feature

Switch selection

## Set temporary value of WetldBird for wetlands outside the atlas block coverage

Open attribute table of WU_WetlandBird1
Field Calculate WetldBird = 99

## Spatial Join to find the closest atlas block to each wetland

Spatial Join
Target Features: WU_WetlandBird1
Join Features: WetlandBirds_WetBirdColumnOnly
Output Feature Class: WU_WetlandBird
Join Operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field map of join features (retain the fields below):
WUKey
Shape_Length
Shape_Area
WetldBird
WetBird
Match Option: CLOSEST

## Assign points based on WetBird value to wetlands outside the atlas blocks

Open attribute table of WU_WetlandBird

SELECT * FROM WU_WetlandBird WHERE: "WetlBird" = 99 AND "WetBird" <= 0.354
Field Calculate WetldBird = 0

SELECT * FROM WU_WetlandBird WHERE: "WetlBird" = 99 AND "WetBird" > 0.354
AND "WetBird" <= 0.408
Field Calculate WetldBird = 1

SELECT * FROM WU_WetlandBird WHERE: "WetldBird" = 99 AND "WetBird" > 0.408
AND "WetBird" <= 0.493
Field Calculate WetldBird = 2

SELECT * FROM WU_WetlandBird WHERE: "WetldBird" = 99 AND "WetBird" > 0.493
Field Calculate WetldBird = 3

5.6.87 WFlowPath: Water Flow Path

Version date: 20 April 2016

Strategy: completed 15 Feb 2016 EAB
GIS detailed method: completed, 3/10/2016 EAB & JCC; Verified 3/10/2016; Revised 4/20/2016 EAB
Python coding: completed for first iteration (MCA); unfortunately this must be re-coded for this version. There was almost no overlap in the procedure - EAB

April 11: Apologies, there has been yet another update, highlighted in yellow below, to this procedure. I also changed the interim file naming slightly to accommodate this change (files generated include interim files whose names are not important: WU_WFlowPath1, WU_WFlowPath2; and the final product whose name I did not change: WU_WFlowPath). – EAB

This was completed on 6/14/2016 MCA. Part 1 takes around 2 hours to run and might need to be revisited.

EAB 10/3/16: The steps in part 2 below don't appear to have been added to the code yet:
## Join to add area of contributing watershed (CntrWshd)
## Export join to feature class
## Update isolated wetlands to outflow intermittent if contributing wshed > 40 acres

I missed adding this by using an older version of this document. The changes have been added and I was able to fix some other problems that were occurring. The total runtime with the Wetland Units and the input is now 22 minutes and 27 seconds. With 25 wetland units, the execute time is 2 minutes and 14 seconds.

Final review by EAB: 10/7/2016

Purpose:

Water Quality Function

Background:

Water Flow Path is part of the Tiner wetland functional classification is based on Tiner (2011) which describes and classifies wetlands by landscape position, landform, water flow path, and waterbody type (LLWW).
NHD Fcodes listed at:
http://nhd.usgs.gov/userGuide/Robohelpfiles/NHD_User_Guide/Feature_Catalog/Hydrography_Dataset/Complete_FCode_List.htm

<u>Note on stream intersections:</u>
In the method below, if a WU boundary intersects more than 1 perennial stream, then it is called throughflow perennial. Presumably one of the intersections will be an outflow, and the other(s) will be inflow. This assumption does not work for a small number of headwater wetlands on drainage divides with no inflows but two outflows. Future work might include adding the direction of flow to identify these wetlands.

<u>Note on flow accumulation:</u>

Paybins (2003) in southern WV noted that median watershed size to initiate intermittent flow is 14.5 +-3.4 acres, and for perennial flow is 40.8 +-18.0 acres. In the method below, wetlands that are tagged as "isolated" but have maximum flow accumulation values > 2000, which corresponds to a drainage area > 0.0625 mi or 40 acres), are updated from "isolated" to "outflow". This threshold corresponds roughly to known outflow wetlands according to Elizabeth's field knowledge. Note that the flow accumulation values are not very accurate. If Mike Shank is able to better calculate the contributing watershed, we should replace the flow accumulation below with the contributing watershed value.

Note for future work (karst):
In the future we may be able to identify Inflow Wetland Units (NHD "dangles"). There will be very few, mostly in karst areas.

Definitions:

Options for water flow path are: Paludified, Isolated, Throughflow, Inflow, Outflow, Bi-directional non-tidal. Modifiers are perennial, intermittent, and artificial. Not all of these can be accurately assigned via GIS, but the main codes (IS, OU, OP, OI, TH, TI, and BI) can be approximately assigned. Here is the full list of Tiner (2011) Water Flow Path codes potentially found in WV:

Water Flow Path

PA Paludified
IS Isolated
IT Isolated-throughflow (connected to other wetlands in an isolated complex)
IO Isolated-outflow (connected to other wetlands in an isolated complex)
II Isolated-inflow (connected to other wetlands in an isolated complex)
ITA Isolated-artificial throughflow (connected by ditches to other artificially isolated wetlands)
IOA Isolated-artificial outflow (connected by ditches to other artificially isolated wetlands)
IIA Isolated-artificial inflow (connected by ditches to other artificially isolated wetlands)
IN Inflow
OU Outflow
OA Outflow-artificial (wetland connected to stream by ditches)
OP Outflow-perennial
OI Outflow-intermittent
TH Throughflow (=Throughflow-perennial)
TA Throughflow-artificial (wetland connected to stream by ditches)
TN Throughflow-entrenched
TI Throughflow-intermittent
BI Bidirectional-nontidal
BIA Bidirectional-nontidal Artificial (e.g., diked wetland)
BO Bidirectional-nontidal/outflow (lake)
TB Bidirectional-nontidal/throughflow (lake)
IB Bidirectional-nontidal/isolated (lake)
NB Bidirectional-nontidal/inflow (lake)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb
  - Feature Class: WU_20150514
  - Feature Class: DrainageArea27m
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb (EnhWVWetland)
- M:\basemap\NHDH_WV.gdb, Feature Dataset: Hydrography, Feature Class: NHDFlowline
  - Note that NHD data is incorrectly attributed (no ephemeral or intermittent streams) in Preston County and parts of Monongalia, Barbour, Morgan, Kanawha, Putnam, Wirt, Marshall, Tucker, and small parts of about 10 additional counties. Not much we can do about this for now.
- M:\LayerFiles\arcsde_backup.gdb, Feature Dataset: basemap_physical_non_replica, Feature Class: SDE_NHD_waterbodies_24k_rivers

Method:

# Create feature class to store Water Flow Path

R-click WU_20150514 and select Data / Export Data.
Output feature class: WU_WflowPath1

# Add text fields to the Wetland Units feature class, to allow computation of Water Flow Path (WFlowPath).

Add 3 text fields (4 characters wide) to WU_WFlowPath1 attribute table: FlowPath, PerInt, WFlowPath.

## PART 1: Intersect NHDflowlines and wetlands.

ArcToolbox/Analysis/Overlay/Intersect
Input features: NHDflowline, WU_WflowPath1
Output Feature Class: NHDflowline_Intersect
Join Attributes: ONLY_FID
Output Type: POINT

## Create multiple points for single line segments

ArcToolbox/Data Management Tools/Features/Multipart to Singlepart
Input Features: NHDflowline_Intersect
Output Features: NHDFlowline_IntMult

## Remove doubles

ArcToolbox/Data Management Tools/Generalization/Dissolve
Input Features: NHDFlowline_IntMult
Output Feature Class: NHDFlowline_IntMultDiss

Uncheck Create Multipart Features box

## Select the intermittent or ephemeral streams
## FCode definitions: 46003 Stream/river/intermittent, 46007 Stream/river/ephemeral,
##   33600 Canal/ditch/unattributed, 33601 Canal/ditch/aqueduct (the latter two are the C&O canal
##   along the Potomac: artificial intermittent)
## Select by Attribute, within attribute table of NHDFlowline

Clear all selections.
SELECT * FROM NHDFlowline WHERE: "FCODE" IN (46003, 46007, 33600, 33601)

## Select the Wetland Units that intersect intermittent or ephemeral stream(s)

Select by Location
Selection method: select features from
Target layer: WU_WFlowPath1
Source layer: NHDFlowline
Check “Use selected features”
Spatial selection: intersect the source layer feature

## Attribute the Wetland Units that intersect intermittent or ephemeral stream(s)

Open Attribute table of WU_WFlowPath1
R-click “PerInt” and Field Calculator PerInt = "I"

## Select the perennial streams
## FCode definitions: 33400 Connector, 46000 Stream/river/unattributed,
##   46006 Stream/river/perennial , 55800 Artificial path (most of these are incorrectly
##   attributed perennial streams)
## Select by Attribute, within attribute table of NHDFlowline

Clear all selections.
SELECT * FROM NHDFlowline WHERE: "FCODE" IN (33400, 46000, 46006, 55800)

## Select the Wetland Units that intersect perennial stream(s)

Select by Location
Selection method: select features from
Target layer: WU_WFlowPath1
Source layer: NHDFlowline
Check “Use selected features”
Spatial selection: intersect the source layer feature

## Attribute the Wetland Units that intersect perennial stream(s)

Open Attribute table of WU_WFlowPath1
R-click “PerInt” and Field Calculate PerInt = "P"
Clear all selections.

## Join Wetland Units to stream intersection points.

Analysis Tools / Overlay / Spatial Join
Target Features: WU_WFlowPath1
Join Features: NHDFlowline_IntMultDiss
Output Feature Class: WU_WFlowPath2
Join Operation: JOIN_ONE_TO_ONE
Check “Keep all target features”
Match Option: INTERSECT

## Attribute FlowPath (Throughflow, Outflow, Isolated)

Select by Attributes
Select * FROM WU_WFlowPath2 WHERE: "Join_Count" > 1
R-click “FlowPath”, Field Calculator = "TH"
Select * FROM WU_WFlowPath2 WHERE: "Join_Count" = 1
R-click “FlowPath”, Field Calculator = "OU"
Select * FROM WU_WFlowPath2 WHERE: "Join_Count" = 0
R-click “FlowPath”, Field Calculator = "IS"
Clear Selections

## PART 2: Update FlowPath based on adjacent streams, rivers, and impoundments
## Update FlowPath from isolated to outflow for wetlands within 30 m of a mapped stream

Select by Location
Selection method: select features from
Target layers: WU_WFlowPath2
Source layer: NHDFlowline
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 30 meters

Select by attributes
Method: select from current selection
SELECT * FROM WU_WFlowPath2 WHERE: “FlowPath” = ‘IS’

R-click “FlowPath”, Field Calculator = “OU”
R-click “PerInt”, Field Calculator = “I”

## Update FlowPath from isolated to outflow for wetlands that contain an impoundment (NWI).

Select by Attributes
Method: create a new selection
SELECT * FROM EnhWVWetland WHERE: "ATTRIBUTE" LIKE 'P%h%'

Select by Location
Selection method: select features from
Target layers: WU_WFlowPath2
Source layer: EnhWVWetland
Check box “Use selected features”
Spatial selection method: contain the source layer feature

Select by attributes
Method: select from current selection
SELECT * FROM WU_WFlowPath2 WHERE: “FlowPath” = ‘IS’

R-click “FlowPath”, Field Calculator = “OU”
R-click “PerInt”, Field Calculator = “I”

## Join to add area of contributing watershed (CntrWshd)

ArcToolbox / Data Management Tools / Joins / Join Field
Input table: WU_WFlowPath2
Input join field: OBJECTID
Join table: DrainageArea27m
Output Join Field: WUKey
Join Fields: WUKey, CntrWshd

## Export join to feature class

R-click WU_WFlowPath2 and select Data / Export Data
Output feature class: WU_WFlowPath

## Update isolated wetlands to outflow intermittent if contributing watershed > 40 acres

Select by Attributes from WU_WFlowPath
SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'IS' AND "CntrWshd" > 161874

R-click “FlowPath” field and select “Field Calculator”. Set FlowPath = "OU"
R-click “PerInt” field and select “Field Calculator”. Set PerInt = "I"

## Set Flow Path to throughflow perennial for wetlands adjacent to NWI rivers
## Select the NWI rivers.

Clear selections.
Select by attributes.
SELECT * FROM EnhWVWetland WHERE: "WETLAND_TYPE" = 'Riverine'

Select by location.
Select method: select features from
Target layer: WU_WFlowPath
Source layer: EnhWVWetland
Check box “Use selected features (1790 features selected)
Spatial selection method: share a line segment with the source layer feature

## Attribute Water Flow Path for Wetland Units adjacent to NWI rivers

Open attribute table of WU_WFlowPath. View selected records
R-click “FlowPath” field and select “Field Calculator”. Set FlowPath = “TH”
R-click “PerInt” field and select “Field Calculator”. Set PerInt = “P”
Clear selections.

## Select the NHD 24K Rivers and set intersecting wetlands to throughflow perennial.

Select by location.
Selection method: select features from
Target layer: WU_WFlowPath
Source layer: SDE_NHD_waterbodies_24k_rivers
Spatial selection method: intersect the source layer feature

## Attribute Flow Path for Wetland Units intersecting NHD 24K rivers

Open attribute table of WU_WFlowPath. View selected records.
R-click “FlowPath” field and select “Field Calculator”. Set FlowPath = “TH”
R-click “PerInt” field and select “Field Calculator”. Set PerInt = “P”

# PART 3: Attribute Water Flow Path for Outflow, Throughflow, and Isolated Wetland Units

Clear selections. Select by Attributes in WU_WFlowPath.

SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'TH' AND "PerInt" = 'I'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = "TI"

SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'TH' AND "PerInt" = 'P'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = "TP"

SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'OU' AND "PerInt" = 'I'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = "OI"

SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'OU' AND "PerInt" = 'P'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = "OP"

SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'IS'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = "IS"

## PART 4: Update Water Flow Path based on adjacent lakes.
## Select the lakes.

Clear selections.
SELECT * FROM EnhWVWetland WHERE: "WETLAND_TYPE" = 'Lake'

## Select wetlands in lake basins with bi-directional flow. This will over-write FlowPath. Note
## that there are some errors generated by over-writing: some wetlands that border a lake but are
## primarily stream wetlands are selected. However, the number of errors is smaller than if over-
## writing is not done. Future refinements could include measuring the perimeter bordering the lake,
## and if it is a small fraction of the total perimeter, then FlowPath “OU” or “TH” is not over-written.

Select by location.
Selection method: select features from
Target layer: WU_WFlowPath
Source layer: EnhWVWetland
Check box “Use selected features” (203 features selected)
Spatial selection method: intersect the source layer feature

## Attribute Flow Path for Wetland Units adjacent to lakes.

Open attribute table of WU_WFlowPath. View selected records.
R-click “FlowPath” field and select “Field Calculator”. Set FlowPath = “BI”

## Attribute Water Flow Path for Wetland Units adjacent to lakes.

Select by Attributes in WU_WFlowPath.
SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'BI' AND "WFlowPath" IN ('OP', 'OI')
R-click “WFlowPath”, select Field calculator, and set “WFlowPath” = ‘BO’.

Select by Attributes in WU_WFlowPath.
SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'BI' AND "WFlowPath" IN ('TP', 'TI')
R-click “WFlowPath”, select Field calculator, and set WFlowPath = “TB”.

Select by Attributes in WU_WFlowPath.
SELECT * FROM WU_WFlowPath WHERE: "FlowPath" = 'BI' AND "WFlowPath" = 'IS'
R-click “WFlowPath”, select Field calculator, and set WFlowPath = “IB”.

5.6.88 WQFunction: Water Quality

Version date: 26 October 2016

Strategy: completed 3/12/2016 EAB
GIS method: completed 3/12/2016 EAB; needs verification once factors are available
Python code:
Final verification by EAB:

Purpose:

Water Quality Function
Maximum 25 points (floodplain wetlands); 24 points (groundwater wetlands)

Description:

Summary of strategy: For each Wetland Unit, sum the points for the three aspects (wetland potential to provide function, landscape offers opportunity to carry out function, and value to society)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- WQPotential
- WQOpportun
- WQSociety

Method:

## Create feature class to store WQFunction

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WQFunction

## Spatial join to bring in aspect values

Spatial join (contains) to bring WQPotential, WQOpportun, and WQSociety into the same table

## Add WQFunction field to Wetland Units and set initial point value to zero.

Open attribute table of WU_WQFunction
Add field "WQFunction" (short integer)
R-click WQFunction and Field Calculate WQFunction = 0

## Sum the points for each aspect of Water Quality Function for Wetland Units

R-click WQFunction and Field Calculate WQFunction = [WQPotential] + [WQOpportun] + [WQSociety]

5.6.89 WQOpportun: Water Quality Opportunity

Version date: 26 October 2016

Strategy: completed 3/12/2016 EAB
GIS method: completed 3/12/2016 EAB; verified 10/26/2016 EAB
Python code:
Final review by EAB:

Purpose:

Water Quality Function
Maximum 5 points (floodplain wetlands); 4 points (groundwater wetlands)

Description:

<u>Summary of strategy</u>: For each Wetland Unit, sum the points for all factors within the "Opportunity" aspect. Note that different point values for floodplain vs. groundwater wetlands are assigned at the factor level. After points have been summed, reduce any point values that exceed the maximum allowable points for this aspect of water quality function.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- Discharges
- ImpairedIn
- RoadRail
- Disturb50m
- DisturbWshd
- Floodplain

Method:

## Create feature class to store WQOpportun

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WQOpportun

## Spatial join to bring in factor values

Spatial join (contains) to add the following to the WU_WQOpportun table: Discharges, ImpairedIn, RoadRail, Disturb50m, DisturbWshd, Floodplain

## Add WQOpportun field and set initial point value to zero.

Open attribute table of WU_WQOpportun

Add field “WQOpportun” (short integer)
R-click WQOpportun and Field Calculate WQOpportun = 0

## Sum the factor points

R-click WQOpportun and Field Calculate WQOpportun = [Discharges] + [ImpairedIn] + [RoadRail] + [Disturb50m] + [DisturbWshd]

## Reduce values that exceed the maximum allowable points

Select by attributes
Layer: WU_Opportun
Method: Create a new selection
SELECT * FROM WU_Opportun WHERE: "Floodplain" = 'Y' AND "WQOpportun" > 5
R-click WQOpportun and Field Calculate WQOpportun = 5

Select by attributes
Layer: WU_Opportun
Method: Create a new selection
SELECT * FROM WU_Opportun WHERE: "Floodplain" = 'N' AND "WQOpportun" > 4
R-click WQOpportun and Field Calculate WQOpportun = 4

5.6.90 WQPlan : Watershed or Water Quality Plan Exists

Version date: 16 March 2016

Strategy: completed 3/12/2016 EAB
GIS method: completed and verified 3/16/2016 EAB
Python coding: started & complete 3/18/2016 MCA
Final review by EAB: 3/18/2016

Purpose:

Water Quality Function
Max 2 point

Description:

Rationale: Inclusion in a watershed plan, water quality plan, or having legal protected status are all indicators that society values the water quality function of a wetland highly.
Sum the points below, for a maximum of 2 points.

- Watershed Plan. Wetland has been identified in a watershed or local plan as important for maintaining water quality (2 points). Not all pollution and water quality problems are identified by state water quality monitoring program. Local and watershed planning efforts sometimes identify wetlands that are important in maintaining existing water quality. These wetlands provide a value to society at the local level that needs to be replaced if they are impacted.
- TMDL. A TMDL exists for the drainage in which the wetland is found (2 points). A Total Maximum Daily Load (TMDL) plan is a plan of action used to clean up streams that are not meeting water quality standards. The TMDL program is part of the Watershed Branch of the WVDEP.
- NSPA. Wetland is in the contributing watershed of a stream reach protected by the Natural Streams Preservation Act (2 points). These include (a) Greenbrier River from its confluence with Knapps Creek to its confluence with the New River, (b) Anthony Creek from its headwaters to its confluence with the Greenbrier River, (c) Cranberry River from its headwaters to its confluence with the Gauley River, (d) Birch River from the Cora Brown bridge in Nicholas county to the confluence of the river with the Elk River, and (e) New River from its confluence with the Gauley River to its confluence with the Greenbrier River.
- Most federally-owned lands have watershed plans to protect water quality. This includes National Forests, National Wildlife Refuges, and National Parks.
- Conservation easements related to water quality (none known at this time)

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
  - Feature Class: WatershedPlan
  - Feature Class: TMDL
  - Feature Class: NatStrProAct_HUC10
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\DNR_PublicLands_20Aug2015\PublicLandsWVDNR20150820.shp

Input Variables:

None (but note that WshdPlan is already coded and can be added to this)

Method:

## Create feature class to store WQPlan factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WQPlan

## Add field to Wetland Units and set initial point value to zero.

Open attribute table of WU_WQPlan
Add field “WQPlan” (short integer)
R-click WQPlan and Field Calculate WQPlan = 0

## Select Wetland Units with Watershed Plan.

Select by location
Selection method: select features from
Target layer: WU_WQPlan
Source layer: WatershedPlan
Spatial selection method: Intersect the source layer feature

## Select Wetland Units with TMDL Plan.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_WQPlan
Source layer: TMDL
Spatial selection method: Intersect the source layer feature

## Select Wetland Units in watersheds drained by National Streams Preservation Act streams.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_WQPlan
Source layer: NatStrPreAct_HUC10
Spatial selection method: Intersect the source layer feature

## Select National Parks, Forests, and Wildlife Refuges

Select by Attributes
Layer: PublicLandsWVDNR20150820.shp
Method: Create a new selection

SELECT * FROM PublicLandsWVDNR20150820.shp WHERE: "Ownership_" IN ('U.S Park Service', 'U.S. Fish and Wildlife Service', 'U.S. Forest Service')

## Select Wetland Units in National Parks, Forests, and Wildlife Refuges.

Select by location
Selection method: add to the currently selected features in
Target layer: WU_WQPlan
Source layer: PublicLandsWVDNR20150820.shp
Check box "Use selected features" (137 features selected)
Spatial selection method: Intersect the source layer feature

## Assign 2 points to Wetland Units that are included in a watershed plan of some kind.

R-click "WQPlan" in WU_WQPlan and Field Calculate WQPlan = 2

5.6.91 WQPotential: Water Quality Potential

Version date: 28 October 2016

Strategy: completed 3/12/2016 EAB
GIS method: completed 3/12/2016 EAB; verified 4/18/2016 EAB
Python code: Started & Complete 6/15/2016 MCA – Problem with point counts
EAB 10/25/16: I revised the point counts with the new values for Headwater and SWOutflow. Can you please re-run and see if they match?
Completed 10/28/2016 MCA
Final review by EAB: 10/28/2016

Purpose:

Water Quality Function
Maximum 16 points (all wetlands)

Description:

Rationale**:** Wetlands have an intrinsic potential to improve water quality, through filtering of contaminants, capture of sediment, absorption of nutrients, and chemical reactions that convert noxious compounds to benign ones (e.g., nitrates to nitrogen gas). This intrinsic capability is related to landscape position, vegetation, microtopography, drainage patterns, soils, wetland shape, and slope of the wetland.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the "Potential" aspect. Note that different point values for floodplain vs. groundwater wetlands are assigned at the factor level.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- ChemTime (3 points GW)
- ClayOrganic (3 points GW)
- Depressions (5 points FL)
- Headwater (1 point – all types)
- SWOutflow (4 points GW)
- VegWQ (10 points FL, 5 points GW)

Method:

## Spatial joins to bring together factor values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_ChemTime
Join Feature: WU_ClayOrganic
Output feature class: WU_WQPotential1

Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
ChemTime
ClayOrganic
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQPotential1
Join Feature: WU_Depressions
Output feature class: WU_WQPotential2
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
ChemTime
ClayOrganic
Depressions
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQPotential2
Join Feature: WU_Headwater
Output feature class: WU_WQPotential3
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
ChemTime
ClayOrganic
Depressions
Headwater
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQPotential3
Join Feature: WU_SWOutflow
Output feature class: WU_WQPotential4
Join Operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following
Shape-Length

Shape_Area
ChemTime
ClayOrganic
Depressions
Headwater
SWOutflow
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQPotential4
Join Feature: WU_VegWQ
Output feature class: WU_WQPotential
Join Operation: JOIN_ONE_TO_ONE
Check box “Keep all target features”
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
ChemTime
ClayOrganic
Depressions
Headwater
SWOutflow
VegWQ
Match Option: CONTAINS

## Add WQPotential field to Wetland Units and set initial point value to zero.

Open attribute table of WU_WQPotential
Add field “WQPotential” (short integer)
R-click WQPotential and Field Calculate WQPotential = 0

## Sum the factor points

R-click WQPotential and Field Calculate WQPotential = [ChemTime]+ [ClayOrganic]+ [Depressions]+ [Headwater]+ [SWoutflow]+ [VegWQ]

5.6.92 WQSociety: Water Quality Value to Society

Vesrion date: 18 March 2016

Strategy: completed 3/12/2016 EAB
GIS method: completed and verified 3/18/2016 EAB
Python code: started & complete 5/26/2016 MCA
Final check by EAB: 10/3/2016

Purpose:

Water Quality Function
Maximum 4 points (all wetlands)

Description:

Rationale: Wetland discharges to (<1 km above) a stream, river, or lake that is on the 303d list, or a water body that is impacted by chronic algal blooms or power boat use. The term, "303(d) list," is short for the list of impaired waters (stream segments, lakes) that the Clean Water Act requires all states to submit to the Environmental Protection Agency (EPA) every two years. Wetlands that discharge directly to these polluted waters are judged to be more valuable than those that discharge to unpolluted bodies of water because their role at cleaning up the pollution is critical for reducing further degradation of water quality.
Summary of strategy: For each Wetland Unit, sum the points for all factors within the “Society” aspect. Reduce values that exceed the maximum allowable points for this aspect of water quality function. Note that floodplain and groundwater wetlands are treated the same for this aspect.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)

Input Variables:

- HUC12WQ (1 point)
- ImpairedOut (1 point)
- WQPlan (up to 2 points)
- WQUse (up to 2 points)

Method:

## Create feature class to store WQSociety

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WQSociety1

## Add WQSociety field to Wetland Units and set initial point value to zero.

Open attribute table of WU_WQSociety1

Add field “WQSociety” (short integer)
R-click WQSociety and Field Calculate WQSociety = 0

## Spatial joins to bring in factor values

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQSociety1
Join Feature: HUC12WQ
Output feature class: WU_WQSociety2
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
WQSociety
HUC12WQ
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQSociety2
Join Feature: ImpairedOut
Output feature class: WU_WQSociety3
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
WQSociety
HUC12WQ
ImpairedOut
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQSociety3
Join Feature: WQPlan
Output feature class: WU_WQSociety4
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
WQSociety
HUC12WQ
ImpairedOut
WQPlan
Match Option: CONTAINS

ArcToolbox / Analysis Tools / Overlay / Spatial join
Target Feature: WU_WQSociety4

Join Feature: WQUse
Output feature class: WU_WQSociety
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following
Shape-Length
Shape_Area
WQSociety
HUC12WQ
ImpairedOut
WQPlan
WQUse
Match Option: CONTAINS

## Sum the factor points

R-click WQSociety and Field Calculate WQSociety = [HUC12WQ] + [ImpairedOut] + [WQPlan] + [WQUse]

## Reduce values that exceed the maximum allowable points

Select by attributes
Layer: WU_Society
Method: Create a new selection
SELECT * FROM WU_WQSociety WHERE: "WQSociety" > 4

R-click WQSociety and Field Calculate WQSociety = 4

5.6.93 WQUse: Water quality used by public

Version Date: 27 March 2016

Strategy: completed 3/12/2016 EAB
GIS method: 3/17/2016 completed and verified EAB
Python coding: started 3/17/2016 & completed 5/26/2016 MCA
Final review by EAB: 10/3/2016

Purpose:

Water Quality Function, Value to Society
Max 2 points

Description:

Rationale: Water quality is particularly important in areas where public use is high. Water supply intakes, swimming areas, and economically important fisheries are some of the uses that require good water quality. Wetlands can contribute to improved water quality if they are in the contributing watershed for these uses.
Summary of strategy: Sum the points for Water Supply and Fisheries. Add an additional point if the Wetland Unit is within 1 km of a swimming area, and 2 points if the Wetland Unit is within 50 meters of a swimming area. If the total points for a Wetland Unit exceeds the maximum allowable points for this factor, reduce the total points back to 2.

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb (WU_20150514)
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WaterQualityDatasets.gdb
    - Feature Class: SwimmingAreas2016

Input Variables:

- WaterSupply
- Fisheries

Method:

## Create feature class to store WQUse factor

R-click WU_20150514 and select Data/Export Data
Output feature class: WU_WQUse1

## Add fields to Wetland Units and set initial values to zero.

Open attribute table of WU_WQUse1
Add field "WQUse" (short integer)
R-click WQUse and Field Calculate WQUse = 0

Add field “Swim” (short integer)
R-click Swim and Field Calculate Swim = 0

## Select Wetland Units within 1 km of Swimming Area.

Select by location
Selection method: select features from
Target layer: WU_WQUse1
Source layer: SwimmingAreas2016
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 1000 meters

## Assign 1 point to selected Wetland Units.

R-click “Swim” in WU_WQUse1 and Field Calculate Swim = 1

## Select Wetland Units within 50 m of Swimming Area.

Select by location
Selection method: select features from
Target layer: WU_WQUse1
Source layer: SwimmingAreas2016
Spatial selection method: are within a distance of the source layer feature
Apply a search distance: 50 meters

## Assign 2 point to selected Wetland Units.

R-click “Swim” in WU_WQUse1 and Field Calculate Swim = 2

## Spatial Join to add variables WaterSupply and Fisheries

ArcToolbox / Analysis / Overlay / Spatial Join
Target Feature: WU_WQUse1
Join Feature: WU_WaterSupply
Output Feature Class: WU_WQUse2
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
WQUse
Swim
WaterSupply

ArcToolbox / Analysis / Overlay / Spatial Join
Target Feature: WU_WQUse2
Join Feature: WU_Fisheries

Output Feature Class: WU_WQUse
Join Operation: JOIN_ONE_TO_ONE
Field Map of Join Features: retain the following:
Shape_Length
Shape_Area
WQUse
Swim
WaterSupply
Fisheries

## Sum the points for water quality use

In WU_WQUse, R-click WQUse and Field Calculate WQUse = [Swim] + [WaterSupply] + [Fisheries]

## Reduce points to maximum allowed for WQUse

Select by Attributes
Layer: WU_WQUse
Method: Create a new selection
SELECT * FROM WU_WQUse WHERE: "WQUse" > 2

In WU_WQUse, R-click WQUse and Field Calculate WQUse = 2

5.6.94 WshdPos: Watershed Position, headwaters, major river floodplains, and karst

Version date: 2 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/2/2017 EAB; results verified 10/2/2017 EAB
Python code:
Final review by EAB:

Purpose:

Input to Habitat / Landscape Opportunity (LandHydro: Landscape Hydrologic Connectivity)

Description:

Maximum 1 point
Rationale: Headwater wetlands are upstream of all aquatic habitats and provide important protection to these ecosystems. Major river floodplains are an important and highly threatened habitat for toads, frogs, wetland birds, and dragonflies. Karst areas have a uniquely sensitive underground ecology and provide calcium-rich water to above-ground ecosystems.

GIS Method (no field method):
Assign 1 point if the wetland is a

- headwater wetland OR
- Amphibian habitat: wetland is in the floodplain of a major river, defined as having a drainage area > 5000 square miles, i.e., the Ohio, Kanawha, and lower Potomac (below Little Conococheague Creek, 1 mile downstream of Dam No 5, 7 miles upstream of Rt. 81 bridge). *Note: Don't include the Monongahela, New, Big Sandy, Greenbrier, Gauley, Elk, Guyandotte, Little Kanawha, N & S Branch Potomac? Check with Tom Pauley re: amphibian habitat* OR
- Odonates: *Ohio, *Kanawha, Meadow, Potomac, Cacapon, Tygart (higher elevation, but slower and sinuous), and lower portions of the North and South Branch. The Mon and Bluestone really don't have much in the way of wetlands because of the dams (except a few localized places Sue Olcott can think of on the Mon)(Note that amphibian habitat is included in Odonata habitat; since there is only one point available for this metric, these two categories can be combined) OR
- if the wetland occurs on karst (limestone/dolomite bedrock geology or SSURGO karst).

Source Data:

- M:\LayerFiles\arcsde_backup.gdb
  - Feature Dataset: basemap_physical_non_replica
    - Feature Class: SDE_NHD_reach_24k_gt_50_mi_drainage
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
  - Feature Class: WU_FloodArea
    - Field: Floodplain
  - Feature Class: WU_Headwater
    - Field: Headwater
  - Feature Class: WU_Karst

- Field: Karst

Method:

## STEP 1: Wetland Units in the Floodplain of a Major River
## Create feature class to store WshdPos

R-click WU_FloodArea and select Data / Export Data / All features
Output feature class: WetlandFunction.gdb / WU_WshdPos1

## Add field to store major river floodplain (MajorRiverFP) and set initial value to zero

Open attribute table of WU_WshdPos1
Add field MajorRiverFP (short integer)
Field calculate MajorRiverFP = 0

## Select major rivers with floodplains

SELECT * FROM SDE_NHD_reach_24k_gt_50_mi_drainage WHERE: "DA_sq_mi" > 1000 OR "GNIS_Name" IN ('Meadow River', 'Cacapon River', 'Tygart Valley River') OR ("GNIS_Name" IN ('South Branch Potomac River', 'North Branch Potomac River') AND "DA_sq_mi" > 1000)

## Select wetlands that are within 500 meters of selected rivers

Select by Location
Selection method: select features from
Target layer(s): WU_WshdPos1
Source layer: SDE_NHD_reach_24k_gt_50_mi_drainage
Use selected features (3107 features selected)
Spatial selection method for the target layer feature(s): are within a distance of the source layer feature
Apply a search distance: 500 meters

## Remove from selection any wetlands not in the floodplain

Select by Attributes
Layer: WU_WshdPos1
Method: Remove from current selection
SELECT * FROM WU_WshdPos1 WHERE: "Floodplain" = 'N'

## Assign values to MajorRiverFP

Open attribute table of WU_WshdPos1
Field calculate MajorRiverFP = 1

## STEP 2: Join metrics and assign points

## Spatial Join MajorRiversFP to Headwater and Karst
## Note that this can also be done with a "Join" on WUKey.

ArcToolbox/ Spatial Join
Target Features: WU_WshdPos1
Join Features: WU_Headwater
Output Feature Class: WU_WshdPos2
Join Operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
MajorRiverFP
Headwater
Match Option: Contains

ArcToolbox/ Spatial Join
Target Features: WU_WshdPos2
Join Features: WU_Karst
Output Feature Class: WU_WshdPos
Join Operation: JOIN_ONE_TO_ONE
Check "Keep all target features"
Field Map of Join Features:
WUKey
Shape_Length
Shape_Area
MajorRiverFP
Headwater
Karst
Match Option: Contains

## Add field WshdPos, set initial value to zero

Open the attribute table to WU_WshdPos
Add field "WshdPos" (short integer) to attribute table
Field Calculate "WshdPos" = 0

## Assign points

SELECT * FROM WU_WshdPos WHERE: "MajorRiverFP" > 0 OR "Headwater" > 0 OR "Karst" > 0
Field Calculate WshdPos = 1

5.6.95 WshdUniq: Watershed Wetland Size and Uniqueness

Version date: 9 October 2017

Strategy: 3/16/2017 EAB
GIS method: 10/6/2017 EAB; results verified 10/6/2017 EAB
Python code: 10/10/20117 YH
Final review by EAB: 10/10/2017

Purpose:

Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)

Description:

Maximum 2 points

Rationale: Wetlands embedded in a dense or diverse network of nearby wetlands provide greater opportunities to species to thrive and disperse.

GIS Method (no field method):
Make heat map of 12-digit HUC watershed layer with the following fields, highlighting HUCs in the top 10% of the state. If wetland is within a "hot" HUC, award the points. Note that thresholds are based on the values for 772 HUCs in the layer "HUCWetlandSizeUniq".

Type diversity: number of unique NWI codes in the watershed, not including spoil wetlands.

Density: number of vegetated NWI polygons; many of these polygons may be contiguous with each other, forming a single wetland.

Proportional Area: proportion of the watershed's total area occupied by vegetated wetlands as mapped by NWI.

Also, award points if the Wetland Unit is the largest vegetated wetland in its 12-digit HUC watershed (compare area of Wetland Unit with area of largest vegetated Wetland Unit (MaxVegArea) in the HUCWetlandSizeUniq attribute table created for this metric.

2 points: largest vegetated wetland in HUC, or HUC is in top 5% for type diversity, density, or proportional area

1 point: HUC is in top 10% for type diversity, density, or proportional area

0 points: none of the above criteria are met

Threshold value table

	Top 5%	Top 10%
DiverseNWI	28	22
DensVegNWI	70	45
RatioVeg	0.009	0.005

Source Data:

- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Watershed.gdb
    - Feature Class: HUCWetlandSizeUniq
- M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\WetlandFunction.gdb
    - Feature Class: WU_VegAll

Method:

## Spatial Join of Wetland Units (including VegArea) and HUC characteristics

ArcToolbox / Analysis Tools / Overlay / Spatial Join
Target Feature: WU_VegAll
Join Feature: HUCWetlandSizeUniq
Output Feature Class: WU_WshdUniq
Join operation: JOIN_ONE_TO_ONE
Check box "Keep all target features"
Field Map of Join Features: retain the following:
WUKey
VegArea
Shape_Length
Shape_Area
HUC_12
HU_12_NAME
MaxVegArea
DiverseNWI
DensVegNWI
RatioVeg
Match option: HAVE_THEIR_CENTER_IN

## Create feature class to store results for WshdUniq and set initial value to zero

Open attribute table of WU_WshdUniq
Add field "WshdUniq" (short integer)
Field calculate WshdUniq = 0

## Select wetlands in top 10% of HUC watersheds based on
## the number of unique NWI codes, the number of vegetated NWI polygons, or
## the proportional area of vegetated wetlands

SELECT * FROM WU_WshdUniq WHERE: "DiverseNWI" > 22 OR "DensVegNWI" > 45
OR "RatioVeg" > 0.005

## Assign 1 point

Open attribute table of WU_WshdUniq
Field Calculate WshdUniq = 1

## Select largest vegetated wetland in each HUC AND wetlands in top 5% of HUC watersheds based on
## type diversity, density, or proportional area of vegetated wetlands

SELECT * FROM WU_WshdUniq WHERE: "VegArea" >= "MaxVegArea" OR "DiverseNWI" > 28 OR "DensVegNWI" > 70 OR "RatioVeg" > 0.009

## Assign 2 points

Open attribute table of WU_WshdUniq
Field Calculate WshdUniq = 2

Clear all selections

5.7 Python 2.7 Code

5.7.1 Flood Attenuation Function

```
##########################################################################
#
# File Name: FAFunction.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/13/2017 (modified 10/32/2017)
# Purpose:
#    Flood Attenuation Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../..")

import arcpy
from utilities import actions
import logging

def DetermineFAFunction(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAFunction")

    # Clean up if needed
    if arcpy.Exists("WU_FAFunction"):
        arcpy.Delete_management("WU_FAFunction")
    if arcpy.Exists("WU_FAFunction1"):
        arcpy.Delete_management("WU_FAFunction1")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FAPotential","fcFAPotential")
    arcpy.MakeFeatureLayer_management(r"WU_FAOpportun","fcFAOpportun")
    arcpy.MakeFeatureLayer_management(r"WU_FASociety","fcFASociety")
    logger.info("feature layers ready")


##############################################################################
#################
    # SJ: FAPotential & FAOpportun

##############################################################################
#################
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcFAPotential")
    fmSJ.addTable("fcFAOpportun")
```

```
   keepers = []
   keepers = ["WUKey","Shape_Length","Shape_Area","FAPotential","FAOpportun"]

   for field in fmSJ.fields:
      if field.name not in keepers:
         fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFAPotential","fcFAOpportun","WU_FAFunction1","JOIN_ONE
_TO_ONE","KEEP_ALL",fmSJ,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_FAFunction1","fcWUFAFunction1")
   logger.info("spatial join FAPotential and FAOpportun completed")


#################################################################################
#################
   # SJ: FASociety

#################################################################################
#################
   fmSJ = arcpy.FieldMappings()
   fmSJ.addTable("fcWUFAFunction1")
   fmSJ.addTable("fcFASociety")

   keepers = []
   keepers =
["WUKey","Shape_Length","Shape_Area","FAPotential","FAOpportun","FASociety"]

   for field in fmSJ.fields:
      if field.name not in keepers:
         fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUFAFunction1","fcFASociety","WU_FAFunction","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJ,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_FAFunction","fcWUFAFunction")
   logger.info("spatial join FASociety completed")

   ## Add FAFuntion field to Wetland Units and set initial point value to zero
   actions.DeleteField("fcWUFAFunction","FAFunction")
   arcpy.AddField_management("fcWUFAFunction", "FAFunction", "SHORT")
   arcpy.CalculateField_management("fcWUFAFunction","FAFunction","0","VB","#")
   logger.info("FAFuntion field added to Wetland Units and initial point value set to zero")

   ## Sum the points for each aspect of Water Quality Function for Wetland Units
```

```
    arcpy.CalculateField_management("fcWUFAFunction","FAFunction","[FAPotential] +
[FAOpportun] + [FASociety]","VB","#")
    logger.info("field FAFunction calculated")

    ## Clean up
    if arcpy.Exists("WU_FAFunction1"):
        arcpy.Delete_management("WU_FAFunction1")
```

5.7.2 Flood Attenuation Opportunity

```
################################################################################
#
# File Name: FAOpportun.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Flood Attenuation Opportunity metrics.
#
################################################################################
import datetime
import logging
import traceback
import arcpy

from Variable import Runoff50m, RunoffWshd, SlopeWshd, StreamEdge, FloodArea
from Factor import FloodIn, ConnectFL
from Aspects import FAOpportun

def procFAOpportun(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun")

    ##############################################################
    ## 1. Run Variables
    ##############################################################

    Runoff50m.DetermineRunoff50m(WetlandPoly)

    RunoffWshd.DetermineRunoffWshd(WetlandPoly)

    SlopeWshd.DetermineSlopeWshd(WetlandPoly)

    StreamEdge.DetermineStreamEdge(WetlandPoly)

    #FloodArea.DetermineFloodArea(WetlandPoly) #this should run first during initRequest

    ##############################################################
    ## 2. Run Factors
    ##############################################################

    FloodIn.DetermineFloodIn(WetlandPoly)

    ConnectFL.DetermineConnectFL()

    ##############################################################
```

```
## 3. Run Aspect
############################################################

FAOpportun.DetermineFAOpportun(WetlandPoly)
```

5.7.3 Flood Attenuation Opportunity Aspects

```
############################################################################
#
# File Name: FAOpportun.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/7/2017 (modified 10/31/2017)
# Purpose:
#    Flood Attenuation Function
#
############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineFAOpportun(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.FAOpportun")

    # Clean up if needed
    if arcpy.Exists("WU_FAOpportun1"):
        arcpy.Delete_management("WU_FAOpportun1")
    if arcpy.Exists("WU_FAOpportun"):
        arcpy.Delete_management("WU_FAOpportun")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FloodIn","fcFloodIn")
    arcpy.MakeFeatureLayer_management(r"WU_Connect","fcConnectFL")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    logger.info("feature layers ready")

    ## Spatial joins to add input variables to Wetland Units attribute table

################################################################################
#################
    # SJ: FloodIn  & ConnectFL

################################################################################
#################
    fmSJFICN = arcpy.FieldMappings()
    fmSJFICN.addTable("fcFloodIn")
    fmSJFICN.addTable("fcConnectFL")

    keepers = []
```

```
   keepers = ["WUKey","Shape_Length","Shape_Area","FloodIn","ConnectFL"]

   for field in fmSJFICN.fields:
      if field.name not in keepers:
         fmSJFICN.removeFieldMap(fmSJFICN.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFloodIn","fcConnectFL","WU_FAOpportun1","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJFICN,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_FAOpportun1","fcWUFAOpportun1")
   logger.info("spatial join FloodIn and ConnectFL completed")


##############################################################################
#################
   # SJ: Floodplain

##############################################################################
#################
   fmSJFL = arcpy.FieldMappings()
   fmSJFL.addTable("fcWUFAOpportun1")
   fmSJFL.addTable("fcFloodplain")

   keepers = []
   keepers = ["WUKey","Shape_Length","Shape_Area","FloodIn","ConnectFL","Floodplain"]

   for field in fmSJFL.fields:
      if field.name not in keepers:
         fmSJFL.removeFieldMap(fmSJFL.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUFAOpportun1","fcFloodplain","WU_FAOpportun","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJFL,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_FAOpportun","fcWUFAOpportun")
   logger.info("spatial join Floodplain completed")

   ## Add FAOpportun field and set initial value to 0
   actions.DeleteField("fcWUFAOpportun","FAOpportun")
   arcpy.AddField_management("fcWUFAOpportun", "FAOpportun", "SHORT")
   arcpy.CalculateField_management("fcWUFAOpportun","FAOpportun","0","VB","#")
   logger.info("FAOpportun field added to Wetland Units and initial point value set to zero")

   ## Sum the factor points

arcpy.CalculateField_management("fcWUFAOpportun","FAOpportun","[FloodIn]+[ConnectF
L]","VB","#")
```

```
    logger.info("field FAOpportun calculated")

    ## Reduce points for groudwater wetlands to a maximum of 2
    strWHERE = """"FAOpportun" > 2 AND "Floodplain" = 'N'"""

arcpy.SelectLayerByAttribute_management("fcWUFAOpportun","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUFAOpportun","FAOpportun","2","VB","#")
    logger.info("points for groudwater wetlands reduced to a maximum of 2")
    arcpy.SelectLayerByAttribute_management("fcWUFAOpportun","CLEAR_SELECTION")

    ## Clean Up
    if arcpy.Exists("WU_FAOpportun1"):
        arcpy.Delete_management("WU_FAOpportun1")
```

5.7.4 ConnectFL: Flood Attenuation Opportunity

```
###########################################################################
#
# File Name: ConnectFL.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 12/20/2016 (modified 10/31/2017)
# Purpose:
#    Input to Flood Attenuation / Opportunity
#
###########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineConnectFL():
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.ConnectFL")

    # Clean up if needed
    if arcpy.Exists("WU_Connect"):
        arcpy.Delete_management("WU_Connect")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodArea")
    arcpy.MakeFeatureLayer_management(r"WU_StreamEdge","fcStreamEdge")
    logger.info("feature layers ready")

    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcFloodArea")
    fmSJ.addTable("fcStreamEdge")
    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","FloodArea","StreamEdge"]
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFloodArea","fcStreamEdge","WU_Connect","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJ,"CONTAINS")
    fcConnect = arcpy.mapping.Layer(r"WU_Connect")
    logger.info("spatial join of StreamEdge to wetland units completed")
    ## Add ConnectFL field and set initial point value to 0
    actions.DeleteField(fcConnect,"ConnectFL")
    arcpy.AddField_management(fcConnect,"ConnectFL","SHORT")
    arcpy.CalculateField_management(fcConnect,"ConnectFL","0","VB","#")
    logger.info("field ConnectFL added and initial point value set to 0")

    ## Sum the points for FloodArea and StreamEdge.
    arcpy.CalculateField_management(fcConnect,"ConnectFL","[FloodArea]+
[StreamEdge]","VB","#")
    logger.info("point values calculated for field ConnectFL")

    ## Reduce the total points to a maximum of 2
    strWHERE = """"ConnectFL" > 2"""
    arcpy.SelectLayerByAttribute_management(fcConnect,"NEW_SELECTION",strWHERE)
    arcpy.CalculateField_management(fcConnect,"ConnectFL","2","VB","#")
    logger.info("total points reduced to a maximum of 2")

    arcpy.SelectLayerByAttribute_management(fcConnect, "CLEAR_SELECTION")

    ## Clean Up
```

5.7.5 FloodIn: Flood Attenuation Opportunity

```
##############################################################################
#
# File Name: FloodIn.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/7/2017 (modified 12/05/2017)
# Purpose:
#    Input to Flood Attenuation / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineFloodIn(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.FloodIn")

    # Clean up if needed
    if arcpy.Exists("WU_FloodIn0"):
        arcpy.Delete_management("WU_FloodIn0")
    if arcpy.Exists("WU_FloodIn1"):
        arcpy.Delete_management("WU_FloodIn1")
    if arcpy.Exists("WU_FloodIn2"):
        arcpy.Delete_management("WU_FloodIn2")
    if arcpy.Exists("WU_FloodIn"):
        arcpy.Delete_management("WU_FloodIn")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"WU_FloodIn0")
    arcpy.MakeFeatureLayer_management(r"WU_SlopeWshd","fcSlopeWshd")
    arcpy.MakeFeatureLayer_management(r"WU_Runoff50m","fcRunoff50m")
    arcpy.MakeFeatureLayer_management(r"WU_RunoffWshd","fcRunoffWshd")
    logger.info("feature layers ready")

    ## Spatial joins to add input variables to Wetland Units attribute table

##############################################################################
#################
    # SJ: SlopeWshd
```

```
###############################################################################
#################
    fmSJSW = arcpy.FieldMappings()
    fmSJSW.addTable("WU_FloodIn0")
    fmSJSW.addTable("fcSlopeWshd")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","SlopeWshd"]

    for field in fmSJSW.fields:
        if field.name not in keepers:
            fmSJSW.removeFieldMap(fmSJSW.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WU_FloodIn0","fcSlopeWshd","WU_FloodIn1","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJSW,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FloodIn1","fcWUFloodIn1")
    logger.info("spatial join to add SlopeWshd completed")


###############################################################################
#################
    # SJ: Runoff50m

###############################################################################
#################
    fmSJR50M = arcpy.FieldMappings()
    fmSJR50M.addTable("fcWUFloodIn1")
    fmSJR50M.addTable("fcRunoff50m")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","SlopeWshd","Runoff50m"]

    for field in fmSJR50M.fields:
        if field.name not in keepers:
            fmSJR50M.removeFieldMap(fmSJR50M.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUFloodIn1","fcRunoff50m","WU_FloodIn2","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJR50M,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FloodIn2","fcWUFloodIn2")
    logger.info("spatial join to add Runoff50m completed")
```

```
################################################################################
#################
   # SJ: RunoffWshd

################################################################################
#################
   fmSJRWSHD = arcpy.FieldMappings()
   fmSJRWSHD.addTable("fcWUFloodIn2")
   fmSJRWSHD.addTable("fcRunoffWshd")

   keepers = []
   keepers =
["WUKey","Shape_Length","Shape_Area","SlopeWshd","Runoff50m","RunoffWshd"]

   for field in fmSJRWSHD.fields:
      if field.name not in keepers:
         fmSJRWSHD.removeFieldMap(fmSJRWSHD.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUFloodIn2","fcRunoffWshd","WU_FloodIn","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJRWSHD,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_FloodIn","fcWUFloodIn")
   logger.info("spatial join to add RunoffWshd completed")

   ## Add FloodIn field and set initial value to 0
   actions.DeleteField("fcWUFloodIn","FloodIn")
   arcpy.AddField_management("fcWUFloodIn", "FloodIn", "SHORT")
   arcpy.CalculateField_management("fcWUFloodIn","FloodIn","0","VB","#")
   logger.info("field FloodIn added and initial value set to 0")

   ## Calculate sum of SlopeWshd, Runoff50m, and RunoffWshd and assign points to FloodIn

arcpy.SelectLayerByAttribute_management("fcWUFloodIn","NEW_SELECTION","""("Slope
Wshd"+ "RunoffWshd"+ "Runoff50m") > 2""")
   arcpy.CalculateField_management("fcWUFloodIn","FloodIn","1","VB","#")


arcpy.SelectLayerByAttribute_management("fcWUFloodIn","NEW_SELECTION","""("Slope
Wshd"+ "RunoffWshd"+ "Runoff50m") > 4""")
   arcpy.CalculateField_management("fcWUFloodIn","FloodIn","2","VB","#")
   logger.info("points assigned to certain Wetland Units")

   ## Clean up
   if arcpy.Exists("WU_FloodIn0"):
      arcpy.Delete_management("WU_FloodIn0")
```

```
if arcpy.Exists("WU_FloodIn1"):
    arcpy.Delete_management("WU_FloodIn1")
if arcpy.Exists("WU_FloodIn2"):
    arcpy.Delete_management("WU_FloodIn2")
```

5.7.6 FloodArea: Flood Attenuation Opportunity

```
################################################################################
#
# File Name: FloodArea.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/8/2017 (modified 11/10/2017)
# Purpose:
#    Floodplain (Y/N): Input to numerous Water Quality and Flood Attenuation metrics
#    FloodArea: Input to Flood Attenuation / Opportunity; Max 2 points (all wetlands, but only
Floodplain wetlands will score high enough to get points)
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineFloodArea(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.FloodArea")

    ## Clean up if needed
    if arcpy.Exists("WU_FloodArea1"):
        arcpy.Delete_management("WU_FloodArea1")
    if arcpy.Exists("WU_FloodArea"):
        arcpy.Delete_management("WU_FloodArea")

    ## Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcFPARAFEMA,"fcFPARAFEMA")
    arcpy.MakeFeatureLayer_management(globalvars.srcPeatlands,"fcPeatlands")
    logger.info("feature layers ready")

    ## Intersect floodplain and Wetland Units
    arInputs = ["fcFPARAFEMA","fcWU"]
    arcpy.Intersect_analysis(arInputs,"WU_FloodArea1","ONLY_FID","#","INPUT")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea1","fcWUFloodArea1")
    logger.info("floodplain intersected with wetland units")

    ## Add field to store floodplain area
    actions.DeleteField("fcWUFloodArea1","FloodAreaAF")
    arcpy.AddField_management("fcWUFloodArea1", "FloodAreaAF", "FLOAT")
arcpy.CalculateField_management("fcWUFloodArea1","FloodAreaAF","[Shape_Area]","VB",
"#")
   logger.info("field FloodAreaAF added to store floodplain area")


################################################################################
################
   ## Spatial join floodplain selection to Wetland Units and sum floodplain area.

################################################################################
################
   fieldmappings = arcpy.FieldMappings()
   fieldmappings.addTable("fcWU")
   fieldmappings.addTable("fcWUFloodArea1")

   fldKeyIndex = fieldmappings.findFieldMapIndex("FloodAreaAF")
   fieldmap = fieldmappings.getFieldMap(fldKeyIndex)
   fieldmap.mergeRule = "sum"
   fieldmappings.replaceFieldMap(fldKeyIndex, fieldmap)
   keepers = ["Shape_Length","Shape_Area","WUKey","FloodAreaAF"]

   for field in fieldmappings.fields:
      if field.name not in keepers:
         fieldmappings.removeFieldMap(fieldmappings.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWU","fcWUFloodArea1","WU_FloodArea","JOIN_ONE_TO_O
NE","KEEP_ALL",fieldmappings)
   arcpy.MakeFeatureLayer_management("WU_FloodArea","fcWUFloodArea")
   logger.info("spatuak join completed to sum floodplain area")


################################################################################
################
   ## Add fields to store Flood Area Ratio and Flood Area points
   actions.DeleteField("fcWUFloodArea","FloodRatio")
   arcpy.AddField_management("fcWUFloodArea", "FloodRatio", "FLOAT")
   arcpy.CalculateField_management("fcWUFloodArea","FloodRatio","0","VB","#")
   logger.info("field FloodRatio added and initial point value set to 0")

   actions.DeleteField("fcWUFloodArea","FloodArea")
   arcpy.AddField_management("fcWUFloodArea", "FloodArea", "SHORT")
   arcpy.CalculateField_management("fcWUFloodArea","FloodArea","0","VB","#")
   logger.info("field FloodArea added and initial point value set to 0")
   ## Calculate the ratio of floodplain are to total Wetland  Units Area

arcpy.CalculateField_management("fcWUFloodArea","FloodRatio","[FloodAreaAF]/[Shape_A
rea]","VB","#")
   logger.info("field FloodRation calculated")

   ## Assign Points

arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""Floo
dRatio" > 0.1""")
   arcpy.CalculateField_management("fcWUFloodArea","FloodArea","1","VB","#")


arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""Floo
dRatio" > 0.5""")
   arcpy.CalculateField_management("fcWUFloodArea","FloodArea","2","VB","#")
   logger.info("points assigned to field FloodArea")
   arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

   ## Select Wetland Units that contain peat deposits

arcpy.SelectLayerByLocation_management("fcWUFloodArea","INTERSECT","fcPeatlands","
#","NEW_SELECTION")
   arcpy.CalculateField_management("fcWUFloodArea","FloodArea","0","VB","#")
   logger.info("points assigned to wetland units that contain peat deposits")
   arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

   ## Add field Floodplain to Wetland Units attribute table and set initial value to "N"
   actions.DeleteField("fcWUFloodArea","Floodplain")
   arcpy.AddField_management("fcWUFloodArea", "Floodplain", "TEXT",2)
   arcpy.CalculateField_management("fcWUFloodArea","Floodplain","'N'","PYTHON","#")
   logger.info("field Floodplain added and initial value set to 'N'")

   ## Select Wetland Units that have at least 10% of their area in a FEMA floodplain or Active
River Area

arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""Floo
dArea" > 0""")
   arcpy.CalculateField_management("fcWUFloodArea","Floodplain","'Y'","PYTHON","#")
   logger.info("field Floodplain identified in selected wetlanf units")
   arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

   ## Clean up
   if arcpy.Exists("WU_FloodArea1"):
      arcpy.Delete_management("WU_FloodArea1")
```

5.7.7 Runoff50: Flood Attenuation Opportunity

```
##########################################################################
#
# File Name: Runoff50m.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/16/2016 (modified 11/16/2017)
# Purpose:
#    Flood Attenuation Function / Opportunity aspect
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineRunoff50m(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.Runoff50m")

    # Clean up if needed
    if arcpy.Exists("WU_Runoff50m1"):
        arcpy.Delete_management("WU_Runoff50m1")
    if arcpy.Exists("Buffer50mRun"):
        arcpy.Delete_management("Buffer50mRun")
    if arcpy.Exists("Buffer50mRun_diss"):
        rcpy.Delete_management("Buffer50mRun_diss")
    if arcpy.Exists("WU_Runoff50m"):
        arcpy.Delete_management("WU_Runoff50m")

    # Setting python variables
    fcRunoffLand = arcpy.mapping.Layer(globalvars.srcRunoffLand)
    fcBuffer50m = arcpy.mapping.Layer(globalvars.srcBuffer50m)
    logger.info("feature layers ready")

    ## Create feature class to store intermediate results for Runoff50m
    arcpy.CopyFeatures_management(WetlandPoly,"WU_Runoff50m1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_Runoff50m1", "fcWURunoff50m1")
    logger.info("feature class WU_Runoff50m1 created")

    # Intersect the 50m buffers and the runoff land uses
    arInputData = [fcBuffer50m,fcRunoffLand]
    arcpy.Intersect_analysis(arInputData,"Buffer50mRun","ALL",output_type="INPUT")
    logger.info("intersected runoff land uses with 50m buffers")
    fcBuffer50mRun = arcpy.mapping.Layer(r"Buffer50mRun")

    # Dissolve runoff lands by wetland buffer
    arcpy.Dissolve_management(fcBuffer50mRun,"Buffer50mRun_diss","WUKey","BufferArea
FIRST","MULTI_PART","DISSOLVE_LINES")
    fcBuffer50mRun_diss = arcpy.mapping.Layer(r"Buffer50mRun_diss")
    logger.info("dissolved runoff lands by wetland buffer")

    # Add field and calculate ration of runoff are to total drainage
    actions.DeleteField(fcBuffer50mRun_diss,"Run50mRat")
    arcpy.AddField_management(fcBuffer50mRun_diss, "Run50mRat", "FLOAT")

arcpy.CalculateField_management(fcBuffer50mRun_diss,"Run50mRat","[Shape_Area]/[FIRST
_BufferArea]","VB","#")
    logger.info("field Run50mRat added and calculated")

    # Join ratio of runoff land to Wetland Units

arcpy.AddJoin_management("fcWURunoff50m1","WUKey","Buffer50mRun_diss","WUKey",
"KEEP_ALL")
    logger.info("ratio of runoff land joined to Wetland Units")

    # Export joined data
    #arcpy.CopyFeatures_management("fcWURunoff50m1","WU_Runoff50m","#","0","0","0")

arcpy.FeatureClassToFeatureClass_conversion("fcWURunoff50m1",arcpy.env.workspace,"WU
_Runoff50m")
    fcWURunoff50m = arcpy.mapping.Layer(r"WU_Runoff50m")
    logger.info("joined data exported")

    ## Remove Join
    arcpy.RemoveJoin_management("fcWURunoff50m1")
    logger.info("joined removed")

    # Update Null records to 0 in the Run50mRat field

arcpy.SelectLayerByAttribute_management(fcWURunoff50m,"NEW_SELECTION","""Buffe
r50mRun_diss_Run50mRat" IS NULL""")

arcpy.CalculateField_management(fcWURunoff50m,"Buffer50mRun_diss_Run50mRat","0","
VB","#")
    logger.info("Null records updated to 0 in the Run50mRat field")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(fcWURunoff50m,"Runoff50m")
    arcpy.AddField_management(fcWURunoff50m, "Runoff50m", "SHORT")
    arcpy.CalculateField_management(fcWURunoff50m,"Runoff50m","0","VB","#")
    logger.info("field Runoff50m added to Wetland Units and initial point value set to zero")

    # Assign Points

arcpy.SelectLayerByAttribute_management(fcWURunoff50m,"NEW_SELECTION","""""Buffe
r50mRun_diss_Run50mRat" > 0.1""")
    arcpy.CalculateField_management(fcWURunoff50m,"Runoff50m","1","VB","#")


arcpy.SelectLayerByAttribute_management(fcWURunoff50m,"NEW_SELECTION","""""Buffe
r50mRun_diss_Run50mRat" > 0.33""")
    arcpy.CalculateField_management(fcWURunoff50m,"Runoff50m","2","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWURunoff50m, "CLEAR_SELECTION")
    logger.info("points assigned to wetland units")

    # Update NULL values with 0

arcpy.SelectLayerByAttribute_management(fcWURunoff50m,"NEW_SELECTION","""""Runo
ff50m" IS NULL""")
    arcpy.CalculateField_management(fcWURunoff50m,"Runoff50m","0","VB","#")
    logger.info("Null records updated to 0 in the Runoff50m field")
    arcpy.SelectLayerByAttribute_management(fcWURunoff50m, "CLEAR_SELECTION")

    if arcpy.Exists("WU_Runoff50m1"):
        arcpy.Delete_management("WU_Runoff50m1")
    if arcpy.Exists("Buffer50mRun"):
        arcpy.Delete_management("Buffer50mRun")
    if arcpy.Exists("Buffer50mRun_diss"):
        arcpy.Delete_management("Buffer50mRun_diss")
```

5.7.8 RunoffWshd: Flood Attenuation Opportunity

```
###########################################################################
#
# File Name: RunoffWshd.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 1/31/2017 (modified 11/10/2017)
# Purpose:
#    Flood Attenuation Function / Opportunity aspect
#
###########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineRunoffWshd(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.RunoffWshd")

    # Clean up if needed
    if arcpy.Exists("DrainAreaRun"):
        arcpy.Delete_management("DrainAreaRun")
    if arcpy.Exists("DrainAreaRun_diss"):
        arcpy.Delete_management("DrainAreaRun_diss")
    if arcpy.Exists("WU_RunoffWshd"):
        arcpy.Delete_management("WU_RunoffWshd")
    if arcpy.Exists("WU_RunoffWshd1"):
        arcpy.Delete_management("WU_RunoffWshd1")

    # Setting python variables
    fcDA27m = arcpy.mapping.Layer(globalvars.srcDrainageArea)
    fcRL = arcpy.mapping.Layer(globalvars.srcRunoffLand)
    logger.info("feature layers ready")

    ## Create feature class to store intermediate results for Runoff50m
    arcpy.CopyFeatures_management(WetlandPoly,"WU_RunoffWshd1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_RunoffWshd1", "fcWURunoffWshd1")
    logger.info("feature class WU_RunoffWshd1 created")

    # Intersect the drainage areas and the runoff land uses (Takes 32 minutes 23 seconds to run)
    arInputData = [fcDA27m,fcRL]
    arcpy.Intersect_analysis(arInputData,"DrainAreaRun","ALL","#","INPUT")
    fcDrainAreaRun = arcpy.mapping.Layer(r"DrainAreaRun")
    logger.info("drainage areas and the disturbed land uses intersected")

    # Dissolve runoff lands by drainage area (Takes around 23 hours)
    arcpy.Dissolve_management(fcDrainAreaRun,"DrainAreaRun_diss","WUKey","CntrWshd
FIRST","MULTI_PART","DISSOLVE_LINES")
    fcDrainAreaRunDiss = arcpy.mapping.Layer(r"DrainAreaRun_diss")
    logger.info("runoff lands dissolved by drainage area")

    # Add field to DrainAreaRun_diss and calculate ratio of runoff area to total drainange
    actions.DeleteField(fcDrainAreaRunDiss,"RunWshdRat")
    arcpy.AddField_management(fcDrainAreaRunDiss, "RunWshdRat", "FLOAT")
    arcpy.CalculateField_management(fcDrainAreaRunDiss, "RunWshdRat",
"[SHAPE_Area]/[FIRST_CntrWshd]", "VB")
    logger.info("field RunWshdRat added to DrainAreaRun_diss and calculated")

    # Join ratio of runoff land to Wetland Units

arcpy.AddJoin_management("fcWURunoffWshd1","WUKey",fcDrainAreaRunDiss,"WUKey",
"KEEP_ALL")
    logger.info("ratio of runoff land joined to wetland units")

    # Export joined data
    #arcpy.CopyFeatures_management("fcWURunoffWshd1", "WU_RunoffWshd")

arcpy.FeatureClassToFeatureClass_conversion("fcWURunoffWshd1",arcpy.env.workspace,"W
U_RunoffWshd")
    fcRunoffWshd = arcpy.mapping.Layer(r"WU_RunoffWshd")
    logger.info("joined data exported")
    arcpy.RemoveJoin_management("fcWURunoffWshd1")

    # Set NULL values in the DistWshdRat to 0

arcpy.SelectLayerByAttribute_management(fcRunoffWshd,"NEW_SELECTION","""DrainAr
eaRun_diss_RunWshdRat" IS NULL""")

arcpy.CalculateField_management(fcRunoffWshd,"DrainAreaRun_diss_RunWshdRat","0","V
B","#")
    logger.info("null values in DistWshdRat set to 0")
    arcpy.SelectLayerByAttribute_management(fcRunoffWshd, "CLEAR_SELECTION")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(fcRunoffWshd,"RunoffWshd")
    arcpy.AddField_management(fcRunoffWshd, "RunoffWshd", "SHORT")
    arcpy.CalculateField_management(fcRunoffWshd, "RunoffWshd", "0", "VB")
    logger.info("field RunoffWshd added to Wetland Units and initial point value set to zero")
    # Assign points

arcpy.SelectLayerByAttribute_management(fcRunoffWshd,"NEW_SELECTION",""""DrainAr
eaRun_diss_RunWshdRat" > 0.1""")
    arcpy.CalculateField_management(fcRunoffWshd, "RunoffWshd", "1", "VB")


arcpy.SelectLayerByAttribute_management(fcRunoffWshd,"NEW_SELECTION",""""DrainAr
eaRun_diss_RunWshdRat" > 0.25""")
    arcpy.CalculateField_management(fcRunoffWshd, "RunoffWshd", "2", "VB")
    logger.info("points assigned to field RunoffWshd")
    arcpy.SelectLayerByAttribute_management(fcRunoffWshd, "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("DrainAreaRun"):
        arcpy.Delete_management("DrainAreaRun")
    if arcpy.Exists("DrainAreaRun_diss"):
        arcpy.Delete_management("DrainAreaRun_diss")
    if arcpy.Exists("WU_RunoffWshd1"):
        arcpy.Delete_management("WU_RunoffWshd1")
```

5.7.9 SlopeWshd: Flood Attenuation Opportunity

```
##############################################################################
#
# File Name: SlopeWshd.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/12/2016 (modified 12/07/2017)
# Purpose:
#    Input to Flood Attenuation Function / Opportunity aspect
#
##############################################################################
#!/usr/bin/python
import sys, os, gc
sys.path.append("../../..")
gc.enable()

import logging
import arcpy
from arcpy.sa import *
from globalvars import globalvars
from utilities import actions

def Get_V(aKey, smallDict):
    try:
        return smallDict[aKey]
    except:
        return (-1)

def DetermineSlopeWshd(WetlandPoly):
    arcpy.CheckOutExtension("Spatial")
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.SlopeWshd")

    # Clean up if needed
    if arcpy.Exists(r"WU_SlopeWshd"):
        arcpy.Delete_management(r"WU_SlopeWshd")
    if arcpy.Exists(r"WU_SlopeWshd1"):
        arcpy.Delete_management(r"WU_SlopeWshd1")

    # Setting python variables
    parID = "WUKey"
    parID2 = "WUKey_1"
    dbf = "slope_wshd"
    temp_dir =
"M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WorkingFiles\\slope_wshd_temp"
    if not os.path.exists(temp_dir):
        os.mkdir(temp_dir)
    if arcpy.Exists(temp_dir + "\\tempGDB.gdb"):
        arcpy.Delete_management(temp_dir + "\\tempGDB.gdb")
    arcpy.CreateFileGDB_management(temp_dir, "tempGDB")

    #arcpy.CopyFeatures_management(globalvars.srcDrainageArea, "DrainageArea2")
    fcDA = arcpy.mapping.Layer(globalvars.srcDrainageArea)

#arcpy.MakeRasterLayer_management(r"M:\dems\ned_slope_aspect.gdb\NED_3meter_meters
_augmented_slope_pct", "fcNED3mAugSlpPct")
    logger.info("feature layers ready")

    # Add field MEAN to store zonal statistics outputs
    actions.DeleteField(fcDA,"MEAN")
    arcpy.AddField_management(fcDA, "MEAN", "SHORT")
```

```
logger.info("field MEAN added to store zonal statistics outputs")
orig_dir = arcpy.env.workspace
arcpy.env.workspace = temp_dir + "\\tempGDB.gdb"
arcpy.env.overwriteOutput = True

# Zonal Statistics on DrainageArea, which contains multiple overlapping polygons
## Defining neighbors using spatial join
arcpy.SpatialJoin_analysis(fcDA, fcDA, "SD" ,"JOIN_ONE_TO_MANY")
#joinLR = arcpy.mapping.Layer(r"SD.shp")
logger.info("Spatial join completed to define neighbors")

## Creating empty dictionary and load data
dictFeatures = {}
with arcpy.da.SearchCursor(fcDA, parID) as cursor:
    for row in cursor:
        dictFeatures[row[0]]=()
    del row, cursor
logger.info("Empty dictionary created and data loaded")

## Assigning neighbors
with arcpy.da.SearchCursor(arcpy.env.workspace + "\\SD", (parID,parID2)) as cursor:
    for row in cursor:
        aKey=row[0]
        aList=dictFeatures[aKey]
        aList+=(row[1],)
        dictFeatures[aKey]=aList
    del row, cursor
logger.info("Neighbors assigned")
```

```
## Defining non-overlapping subsets and running zonal statistics on each
runNo=0
while (True):
    toShow,toHide=(),()
    nF=len(dictFeatures)
    for item in dictFeatures:
        if item not in toShow and item not in toHide:
            toShow+=(item,)
            toHide+=(dictFeatures[item])
    m=len(toShow)
    quer='"WUKey" IN '+str(toShow)
    if m==1:
        quer='"WUKey" = '+str(toShow[0])
    fcDA.definitionQuery=quer
    runNo+=1
    print("Run %i, %i polygon(s) found" % (runNo,m))
    print("Running Statistics...")
    arcpy.env.extent = fcDA.getExtent()
    zonalTable = ZonalStatisticsAsTable(fcDA, parID, globalvars.srcSlopePCT, dbf, "DATA",
"MEAN")
    print("Data transfer...")
    smallDict={}
    with arcpy.da.SearchCursor(dbf, ("WUKEY","MEAN")) as cursor:
        for row in cursor:
            smallDict[row[0]]=row[1]
        del row, cursor
    with arcpy.da.UpdateCursor(fcDA, (parID,"MEAN")) as cursor:
        for row in cursor:
            aKey=row[0]
            row[1]=Get_V(aKey, smallDict)
            cursor.updateRow(row)
         del row, cursor
      for item in toShow:
         del dictFeatures[item]
      m=len(dictFeatures)
      if m==0:
         break
      gc.collect()
   fcDA.definitionQuery=""
   arcpy.env.extent = fcDA.getExtent()
   logger.info("zonal statistics completed and appended to DrainageArea")

   arcpy.env.workspace = orig_dir

   ## Export the joined data to a Feature Class
   arcpy.CopyFeatures_management(fcDA, "WU_SlopeWshd1")
   fcSlopeWshd1 = arcpy.mapping.Layer(r"WU_SlopeWshd1")
   logger.info("joined data exported as WU_SlopeWshd1")

   ## Rename MEAN field
   actions.DeleteField(fcSlopeWshd1,"MnSlopeWshd")
   arcpy.AddField_management(fcSlopeWshd1, "MnSlopeWshd", "SHORT")

#arcpy.CalculateField_management(fcSlopeWshd1,"MnSlopeWshd","[slope_wshd_MEAN]","
VB","#")
   arcpy.CalculateField_management(fcSlopeWshd1,"MnSlopeWshd","[MEAN]","VB","#")
   logger.info("MEAN field renamed as MnSlopeWshd")

   ## Join the slope values to Wetland Units
```

```
arcpy.AddJoin_management(WetlandPoly,"WUKey",fcSlopeWshd1,"WUKey","KEEP_ALL")

    ## Export joined data

arcpy.FeatureClassToFeatureClass_conversion(WetlandPoly,arcpy.env.workspace,"WU_Slope
Wshd")
    fcWUSlopeWshd = arcpy.mapping.Layer(r"WU_SlopeWshd")
    logger.info("joined data exported")

    arcpy.RemoveJoin_management(WetlandPoly)

    ## Add field to Wetland Unites and set initial point value to zero
    actions.DeleteField(fcWUSlopeWshd,"SlopeWshd")
    arcpy.AddField_management(fcWUSlopeWshd, "SlopeWshd", "SHORT")
    arcpy.CalculateField_management(fcWUSlopeWshd,"SlopeWshd","0","VB","#")
    logger.info("field SlopeWshd added to Wetland Unites and initial point set value to zero")

    ## Assign points

#arcpy.SelectLayerByAttribute_management(fcWUSlopeWshd,"NEW_SELECTION","""Slop
eWshd1_MedSlopeWshd" > 5""")

arcpy.SelectLayerByAttribute_management(fcWUSlopeWshd,"NEW_SELECTION","""WU_
SlopeWshd1_MnSlopeWshd" > 5""")
    arcpy.CalculateField_management(fcWUSlopeWshd,"SlopeWshd","1","VB","#")


#arcpy.SelectLayerByAttribute_management(fcWUSlopeWshd,"NEW_SELECTION","""Slop
eWshd1_MedSlopeWshd" > 15""")

arcpy.SelectLayerByAttribute_management(fcWUSlopeWshd,"NEW_SELECTION","""WU_
SlopeWshd1_MnSlopeWshd" > 15""")
```

```
arcpy.CalculateField_management(fcWUSlopeWshd,"SlopeWshd","2","VB","#")
logger.info("points assigned to field SlopeWshd")

arcpy.SelectLayerByAttribute_management(fcWUSlopeWshd, "CLEAR_SELECTION")

if arcpy.Exists(r"WU_SlopeWshd1"):
    arcpy.Delete_management(r"WU_SlopeWshd1")
```

5.7.10 StreamEdge: Flood Attenuation Opportunity

```
##############################################################################
#
# File Name: StreamEdge.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 8/30/2016 (modified 10/31/2017)
# Purpose:
#    Used in Flood Attenuation Function / Potential aspect / Runoff
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineStreamEdge(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAOpportun.StreamEdge")

    # Clean up if needed
    if arcpy.Exists("Rivers"):
        arcpy.Delete_management("Rivers")
    if arcpy.Exists("WUbyRivers"):
        arcpy.Delete_management("WUbyRivers")
    if arcpy.Exists("WURiverLines"):
        arcpy.Delete_management("WURiverLines")
    if arcpy.Exists("RiverEdges"):
        arcpy.Delete_management("RiverEdges")
    if arcpy.Exists("WUStream"):
        arcpy.Delete_management("WUStream")
    if arcpy.Exists("WU_StreamEdge1"):
        arcpy.Delete_management("WU_StreamEdge1")
    if arcpy.Exists("WU_StreamEdge"):
        arcpy.Delete_management("WU_StreamEdge")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    #arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcENWI")
    arcpy.MakeFeatureLayer_management(globalvars.srcNHDFlowline, "fcNHDFlowline")
    arcpy.MakeFeatureLayer_management(globalvars.srcRivers,"fcRivers")
    logger.info("feature layers ready")
```

```
################################################################################
################
   # PART 1: RIVER LENGTH

################################################################################
################
   # Select NWI Rivers
   '''strWHERE = """"ATTRIBUTE" LIKE 'R%'"""
   arcpy.SelectLayerByAttribute_management("fcENWI","NEW_SELECTION",strWHERE)

   arcpy.CopyFeatures_management("fcENWI", "Rivers")
   arcpy.MakeFeatureLayer_management(r"Rivers","fcRivers")
   logger.info("NWI Rivers selected and exported")'''

   # Select Wetland Units that share a boundary with a river and export them

arcpy.SelectLayerByLocation_management("fcWU","INTERSECT","fcRivers","#","NEW_SE
LECTION")
   arcpy.CopyFeatures_management("fcWU", "WUbyRivers")
   arcpy.MakeFeatureLayer_management(r"WUbyRivers","fcWUbyRivers")
   logger.info("Wetland Units that share a boundary with a river selected and exported")

   # Convert Wetland polygons to lines

arcpy.PolygonToLine_management("fcWUbyRivers","WURiverLines","IGNORE_NEIGHBO
RS")
   arcpy.MakeFeatureLayer_management(r"WURiverLines","fcWURiverLines")
   logger.info("Wetland polygons converted to lines")

   # Retain only the wet perimeter lines
   arLayers = ["fcWURiverLines", "fcRivers"]
   arcpy.Intersect_analysis(arLayers,"RiverEdges","ONLY_FID","#","INPUT")
   arcpy.MakeFeatureLayer_management("RiverEdges","fcRiverEdges")
   logger.info("only the wet perimeter lines retained")

   # Add field and calculate wet perimeter in RiverEdges
   actions.DeleteField("fcRiverEdges","RiverPerim")
   arcpy.AddField_management("fcRiverEdges", "RiverPerim", "FLOAT")

arcpy.CalculateField_management("fcRiverEdges","RiverPerim","[Shape_Length]","VB","#")
   logger.info("field RiverPerim added and wet perimeter in RiverEdges calculated")

   # Spatial Join Wetland Units to RiverEdges
   arcpy.SelectLayerByAttribute_management("fcWURiverLines", "CLEAR_SELECTION")
   arcpy.SelectLayerByAttribute_management("fcRivers", "CLEAR_SELECTION")
```

```
    arcpy.SelectLayerByAttribute_management("fcWU", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcRiverEdges", "CLEAR_SELECTION")

    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWU")
    fmSJ.addTable("fcRiverEdges")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","RiverPerim"]
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJ.findFieldMapIndex("RiverPerim")
    fieldmap = fmSJ.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJ.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcRiverEdges","WU_StreamEdge1","JOIN_ONE_TO_ON
E","KEEP_ALL",fmSJ,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WU_StreamEdge1","fcWUStreamEdge1")
    logger.info("Spatial Join Wetland Units to RiverEdges completed")


################################################################################
################
    # PART 2: STREAM LENGTH

################################################################################
################
    # Intersect stream lengths with Wetland Units
    arLayers = ["fcWUStreamEdge1", "fcNHDFlowline"]
    arcpy.Intersect_analysis(arLayers,"WUStream","ONLY_FID","#","INPUT")
    arcpy.MakeFeatureLayer_management("WUStream","fcWUStream")
    logger.info("stream lengths intersected with Wetland Units")

    # Add field to WUStream to store stream length
    actions.DeleteField("fcWUStream","StreamL")
    arcpy.AddField_management("fcWUStream", "StreamL", "FLOAT")
    arcpy.CalculateField_management("fcWUStream","StreamL","[Shape_Length]","VB","#")
    logger.info("field StreamL created and calculated to store stream length")

    ## Sum the stream lengths within each WU
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWUStreamEdge1")
    fmSJ.addTable("fcWUStream")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","RiverPerim","StreamL"]

    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJ.findFieldMapIndex("StreamL")
    fieldmap = fmSJ.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJ.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWUStreamEdge1","fcWUStream","WU_StreamEdge","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJ,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WU_StreamEdge","fcWUStreamEdge")
    logger.info("spatial join completed to sum the stream lengths within each WU")


################################################################################
################
    # PART 3: Calculate ratio and assign points

################################################################################
################
    ## Convert Null values to 0
    strWHERE = """"RiverPerim" IS NULL"""

arcpy.SelectLayerByAttribute_management("fcWUStreamEdge","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUStreamEdge","RiverPerim","0","VB","#")

    strWHERE = """"StreamL" IS NULL"""

arcpy.SelectLayerByAttribute_management("fcWUStreamEdge","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUStreamEdge","StreamL","0","VB","#")
    logger.info("Null values converted to 0")
    arcpy.SelectLayerByAttribute_management("fcWUStreamEdge", "CLEAR_SELECTION")

    ## Create field StreamRatio and set initial values to 0
    actions.DeleteField("fcWUStreamEdge","StreamRatio")
    arcpy.AddField_management("fcWUStreamEdge", "StreamRatio", "FLOAT")
    arcpy.CalculateField_management("fcWUStreamEdge","StreamRatio","0","VB","#")
    ## Calculate field StreamRatio values

arcpy.CalculateField_management("fcWUStreamEdge","StreamRatio","([RiverPerim]+[Stream
L])/([Shape_Area]^0.5)","VB","#")
    logger.info("field StreamRatio created and calculated for wetland units")

    ## Create field StreamEdge and set initial values to 0
    actions.DeleteField("fcWUStreamEdge","StreamEdge")
    arcpy.AddField_management("fcWUStreamEdge", "StreamEdge", "SHORT")
    arcpy.CalculateField_management("fcWUStreamEdge","StreamEdge","0","VB","#")

    ## Assign StreamEdge point values
    strWHERE = """"StreamRatio" > 1"""

arcpy.SelectLayerByAttribute_management("fcWUStreamEdge","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUStreamEdge","StreamEdge","1","VB","#")

    strWHERE = """"StreamRatio" > 3.4"""

arcpy.SelectLayerByAttribute_management("fcWUStreamEdge","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUStreamEdge","StreamEdge","2","VB","#")
    logger.info("field StreamEdge created and calculated for wetland units")
    arcpy.SelectLayerByAttribute_management("fcWUStreamEdge", "CLEAR_SELECTION")

    # Clean Up
    if arcpy.Exists("Rivers"):
        arcpy.Delete_management("Rivers")
    if arcpy.Exists("WUbyRivers"):
        arcpy.Delete_management("WUbyRivers")
    if arcpy.Exists("WURiverLines"):
        arcpy.Delete_management("WURiverLines")
    if arcpy.Exists("RiverEdges"):
        arcpy.Delete_management("RiverEdges")
    if arcpy.Exists("WUStream"):
        arcpy.Delete_management("WUStream")
    if arcpy.Exists("WU_StreamEdge1"):
        arcpy.Delete_management("WU_StreamEdge1")
```

5.7.11 Flood Attenuation Potential

```
##############################################################################
#
# File Name: FAPotential.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Flood Attenuation Potential metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import VegAll, VegPerUng, VegWoody
from Factors import VegFA, Runoff
from Aspects import FAPotential

def procFAPotential(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential")

    ############################################################
    ## 1. Run Variables
    ############################################################

    VegAll.DetermineVegAll(WetlandPoly)

    VegWoody.CalcVegWoody(WetlandPoly)

    #SeasonPond, Microtopo, SLOPE, WFlowPath, VegPerUng were executed with WQuality

    ############################################################
    ## 2. Run Factors
    ############################################################

    #Headwater: executed with WQuality

    #LowSlope: executed with WQuality

    VegFA.DetermineVegFA(WetlandPoly)

    Runoff.DetermineRunoff(WetlandPoly)

    #SWoutflow: executed with WQuality
```

```
############################################################
## 3. Run Aspect
############################################################

FAPotential.DetermineFAPotential(WetlandPoly)
```

5.7.12 Flood Attenuation Potential Aspects

```
##########################################################################
#
# File Name: FAPotential.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 12/21/2016 (modified 10/31/2017)
# Purpose:
#    Flood Attenuation Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineFAPotential(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.FAPotential")

    # Clean up if needed
    if arcpy.Exists("WU_FAPotential"):
        arcpy.Delete_management("WU_FAPotential")
    if arcpy.Exists("WU_FAPotential1"):
        arcpy.Delete_management("WU_FAPotential1")
    if arcpy.Exists("WU_FAPotential2"):
        arcpy.Delete_management("WU_FAPotential2")
    if arcpy.Exists("WU_FAPotential3"):
        arcpy.Delete_management("WU_FAPotential3")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Headwater","fcHeadwater")
    arcpy.MakeFeatureLayer_management(r"WU_LowSlope","fcLowSlope")
    arcpy.MakeFeatureLayer_management(r"WU_VegFA","fcVegFA")
    arcpy.MakeFeatureLayer_management(r"WU_Runoff","fcRunoff")
    arcpy.MakeFeatureLayer_management(r"WU_SWOutflow","fcSWOutflow")
    logger.info("feature layers ready")


###############################################################################
#################
    # SJ: Headwater & LowSlope

###############################################################################
#################
```

```
    fmSJHWLS = arcpy.FieldMappings()
    fmSJHWLS.addTable("fcHeadwater")
    fmSJHWLS.addTable("fcLowSlope")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Headwater","LowSlope"]

    for field in fmSJHWLS.fields:
        if field.name not in keepers:
            fmSJHWLS.removeFieldMap(fmSJHWLS.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcHeadwater","fcLowSlope","WU_FAPotential1","JOIN_ONE_T
O_ONE","KEEP_ALL",fmSJHWLS,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FAPotential1","fcFAPotential1")
    logger.info("spatial join Headwater and LowSlope completed")


###############################################################################
#################
    # SJ: VegFA

###############################################################################
#################
    fmSJVFA = arcpy.FieldMappings()
    fmSJVFA.addTable("fcFAPotential1")
    fmSJVFA.addTable("fcVegFA")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Headwater","LowSlope","VegFA"]

    for field in fmSJVFA.fields:
        if field.name not in keepers:
            fmSJVFA.removeFieldMap(fmSJVFA.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFAPotential1","fcVegFA","WU_FAPotential2","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJVFA,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FAPotential2","fcFAPotential2")
    logger.info("spatial join VegFA completed")


###############################################################################
#################
    # SJ: Runoff
```

```
################################################################################
#################
    fmSJRO = arcpy.FieldMappings()
    fmSJRO.addTable("fcFAPotential2")
    fmSJRO.addTable("fcRunoff")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Headwater","LowSlope","VegFA","Runoff"]

    for field in fmSJRO.fields:
        if field.name not in keepers:
            fmSJRO.removeFieldMap(fmSJRO.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFAPotential2","fcRunoff","WU_FAPotential3","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJRO,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FAPotential3","fcFAPotential3")
    logger.info("spatial join Runoff completed")


################################################################################
#################
    # SJ: SWOutflow2

################################################################################
#################
    fmSJSWOF2 = arcpy.FieldMappings()
    fmSJSWOF2.addTable("fcFAPotential3")
    fmSJSWOF2.addTable("fcSWOutflow")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Headwater","LowSlope","VegFA","Runoff","SW
Outflow2"]

    for field in fmSJSWOF2.fields:
        if field.name not in keepers:
            fmSJSWOF2.removeFieldMap(fmSJSWOF2.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFAPotential3","fcSWOutflow","WU_FAPotential","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJSWOF2,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FAPotential","fcWUFAPotential")
    logger.info("spatial join SWOutflow2 completed")
    ## Add FAPotential field to Wetland Units and set initial point value to 0
    actions.DeleteField("fcWUFAPotential","FAPotential")
    arcpy.AddField_management("fcWUFAPotential", "FAPotential", "SHORT")
    arcpy.CalculateField_management("fcWUFAPotential","FAPotential","0","VB","#")
    logger.info("FAPotential field added to Wetland Units and initial point value set to zero")

    ## Sum the factor points

arcpy.CalculateField_management("fcWUFAPotential","FAPotential","[Headwater]+[LowSlop
e]+[VegFA]+[Runoff]+[SWOutflow2]","VB","#")
    logger.info("field FAPotential calculated")

    ## Clean Up
    if arcpy.Exists("WU_FAPotential1"):
        arcpy.Delete_management("WU_FAPotential1")
    if arcpy.Exists("WU_FAPotential2"):
        arcpy.Delete_management("WU_FAPotential2")
    if arcpy.Exists("WU_FAPotential3"):
        arcpy.Delete_management("WU_FAPotential3")
```

5.7.13 Runoff: Flood Attenuation Potential

```
##############################################################################
#
# File Name: Runoff.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 12/19/2016 (modified 12/01/2017)
# Purpose:
#    Input to Flood Attenuation / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineRunoff(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.Runoff")

    # Clean up if needed
    if arcpy.Exists("WU_Runoff1"):
        arcpy.Delete_management("WU_Runoff1")
    if arcpy.Exists("WU_Runoff2"):
        arcpy.Delete_management("WU_Runoff2")
    if arcpy.Exists("WU_Runoff3"):
        arcpy.Delete_management("WU_Runoff3")
    if arcpy.Exists("WU_Runoff"):
        arcpy.Delete_management("WU_Runoff")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(r"WU_StreamEdge","fcStreamEdge")
    arcpy.MakeFeatureLayer_management(r"WU_SeasonPond","fcSeasonPond")
    arcpy.MakeFeatureLayer_management(r"WU_Microtopo","fcMicrotopo")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    logger.info("feature layers ready")

    ## Spatial Joins to add input variables to Wetland Units attribute table

###############################################################################
#################
    # SJ: StreamEdge
```

```
###############################################################################
#################
   fmSJSR = arcpy.FieldMappings()
   fmSJSR.addTable("fcWU")
   fmSJSR.addTable("fcStreamEdge")

   keepers = []
   keepers = ["WUKey","Shape_Length","Shape_Area","StreamEdge"]

   for field in fmSJSR.fields:
      if field.name not in keepers:
         fmSJSR.removeFieldMap(fmSJSR.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWU","fcStreamEdge","WU_Runoff1","JOIN_ONE_TO_ONE","
KEEP_ALL",fmSJSR,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_Runoff1","fcRunoff1")
   logger.info("spatial join to add StreamEdge completed")


###############################################################################
#################
   # SJ: SeasonPond

###############################################################################
#################
   fmSJSP = arcpy.FieldMappings()
   fmSJSP.addTable("fcRunoff1")
   fmSJSP.addTable("fcSeasonPond")

   keepers = []
   keepers = ["WUKey","Shape_Length","Shape_Area","StreamEdge","SeasonPond"]

   for field in fmSJSP.fields:
      if field.name not in keepers:
         fmSJSP.removeFieldMap(fmSJSP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcRunoff1","fcSeasonPond","WU_Runoff2","JOIN_ONE_TO_ON
E","KEEP_ALL",fmSJSP,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_Runoff2","fcRunoff2")
   logger.info("spatial join to add SeasonPond completed")
```

```
##################################################################################
#################
   # SJ: Microtopo

##################################################################################
#################
   fmSJMicro = arcpy.FieldMappings()
   fmSJMicro.addTable("fcRunoff2")
   fmSJMicro.addTable("fcMicrotopo")

   keepers = []
   keepers =
["WUKey","Shape_Length","Shape_Area","StreamEdge","SeasonPond","Microtopo"]

   for field in fmSJMicro.fields:
      if field.name not in keepers:
         fmSJMicro.removeFieldMap(fmSJMicro.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcRunoff2","fcMicrotopo","WU_Runoff3","JOIN_ONE_TO_ONE
","KEEP_ALL",fmSJMicro,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_Runoff3","fcRunoff3")
   logger.info("spatial join to add Microtopo completed")


##################################################################################
#################
   # SJ: Floodplain

##################################################################################
#################
   fmSJFP = arcpy.FieldMappings()
   fmSJFP.addTable("fcRunoff3")
   fmSJFP.addTable("fcFloodplain")

   keepers = []
   keepers =
["WUKey","Shape_Length","Shape_Area","StreamEdge","SeasonPond","Microtopo","Floodpl
ain"]

   for field in fmSJFP.fields:
      if field.name not in keepers:
         fmSJFP.removeFieldMap(fmSJFP.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcRunoff3","fcFloodplain","WU_Runoff","JOIN_ONE_TO_ONE",
"KEEP_ALL",fmSJFP,"CONTAINS")
  arcpy.MakeFeatureLayer_management("WU_Runoff","fcRunoff")
  logger.info("spatial join to add Floodplain completed")

  # Add Runoff field to Wetland Units ans set initial point value to zero
  actions.DeleteField("fcRunoff","Runoff")
  arcpy.AddField_management("fcRunoff", "Runoff", "SHORT")
  arcpy.CalculateField_management("fcRunoff","Runoff","0","VB","#")
  logger.info("field Runoff added and initial value set to 0")

  # Sum the points of StreamEdge, SeasonPond, Microtopo

arcpy.CalculateField_management("fcRunoff","Runoff","[StreamEdge]+[SeasonPond]+[Microt
opo]","VB","#")
  logger.info("points of StreamEdge, SeasonPond, Microtopo summed")

  # Reduce the total points to 5 (floodplain) or 4 (groundwater).
  arcpy.SelectLayerByAttribute_management("fcRunoff","NEW_SELECTION","""""Runoff" >
5""")
  arcpy.CalculateField_management("fcRunoff","Runoff","5","VB","#")
  arcpy.SelectLayerByAttribute_management("fcRunoff","NEW_SELECTION","""""Runoff" >
4 AND "Floodplain" = 'N'""")
  arcpy.CalculateField_management("fcRunoff","Runoff","4","VB","#")
  logger.info("total maximum points reduced to 5 for floodplain and 4 for groundwater")

  arcpy.SelectLayerByAttribute_management("fcRunoff", "CLEAR_SELECTION")

  # Clean Up
  if arcpy.Exists("WU_Runoff1"):
    arcpy.Delete_management("WU_Runoff1")
  if arcpy.Exists("WU_Runoff2"):
    arcpy.Delete_management("WU_Runoff2")
  if arcpy.Exists("WU_Runoff3"):
    arcpy.Delete_management("WU_Runoff3")
```

5.7.14 VegFA: Flood Attenuation Potential

```
##############################################################################
#
# File Name: VegFA.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/10/2016 (modified 11/03/2017)
# Purpose:
#    Input to Flood Attenuation / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineVegFA(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.VegFA")

    # Clean up if needed
    if arcpy.Exists("WU_VegFA1"):
        arcpy.Delete_management("WU_VegFA1")
    if arcpy.Exists("WU_VegFA2"):
        arcpy.Delete_management("WU_VegFA2")
    if arcpy.Exists("WU_VegFA3"):
        arcpy.Delete_management("WU_VegFA3")
    if arcpy.Exists("WU_VegFA"):
        arcpy.Delete_management("WU_VegFA")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(r"WU_VegAll","fcVegAll")
    arcpy.MakeFeatureLayer_management(r"WU_VegPerUng","fcVegPerUng")
    arcpy.MakeFeatureLayer_management(r"WU_VegWoody","fcVegWoody")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    logger.info("feature layers ready")

    # Spatial joins to add input variables to Wetland Units attribute table
################################################################################
################
    # SJ: Floodplain
################################################################################
################
    fmSJ1 = arcpy.FieldMappings()
    fmSJ1.addTable("fcWU")
    fmSJ1.addTable("fcFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain"]

    for field in fmSJ1.fields:
        if field.name not in keepers:
            fmSJ1.removeFieldMap(fmSJ1.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcWU","fcFloodplain","WU_VegFA1","JOIN_ONE_TO_ONE","
KEEP_ALL",fmSJ1,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_VegFA1","fcWUVegFA1")
    logger.info("spatial join of Floodplain completed")


################################################################################
################
    # SJ: VegAll

################################################################################
################
    fmSJ2 = arcpy.FieldMappings()
    fmSJ2.addTable("fcWUVegFA1")
    fmSJ2.addTable("fcVegAll")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","VegAll"]

    for field in fmSJ2.fields:
        if field.name not in keepers:
            fmSJ2.removeFieldMap(fmSJ2.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WU_VegFA1","fcVegAll","WU_VegFA2","JOIN_ONE_TO_ON
E","KEEP_ALL",fmSJ2,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_VegFA2","fcWUVegFA2")
    logger.info("spatial join of VegAll completed")
```

```
################################################################################
################
    # SJ: VegPerUng

################################################################################
################
    fmSJ3 = arcpy.FieldMappings()
    fmSJ3.addTable("fcWUVegFA2")
    fmSJ3.addTable("fcVegPerUng")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","VegAll","VegPerUng4"]

    for field in fmSJ3.fields:
        if field.name not in keepers:
            fmSJ3.removeFieldMap(fmSJ3.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUVegFA2","fcVegPerUng","WU_VegFA3","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJ3,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_VegFA3","fcWUVegFA3")
    logger.info("spatial join of VegPerUng completed")


################################################################################
################
    # SJ: VegWoody

################################################################################
################
    fmSJ4 = arcpy.FieldMappings()
    fmSJ4.addTable("fcWUVegFA3")
    fmSJ4.addTable("fcVegWoody")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Floodplain","VegAll","VegPerUng4","VegWoody
4"]

    for field in fmSJ4.fields:
        if field.name not in keepers:
            fmSJ4.removeFieldMap(fmSJ4.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUVegFA3","fcVegWoody","WU_VegFA","JOIN_ONE_TO_
ONE","KEEP_ALL",fmSJ4,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_VegFA","fcWUVegFA")
    logger.info("spatial join of VegWoody completed")

    # Add VegFA field to Wetland Units ans set initial point value to zero
    actions.DeleteField("fcWUVegFA","VegFA")
    arcpy.AddField_management("fcWUVegFA", "VegFA", "SHORT")
    arcpy.CalculateField_management("fcWUVegFA","VegFA","0","VB","#")
    logger.info("field VegWoodyFor added to store points for VegFA and initial value set to
zero")

    # Sum the points for VegALL, VegPerUng4, VegWoody4
    arcpy.CalculateField_management("fcWUVegFA","VegFA","[VegAll] + [VegPerUng4] +
[VegWoody4]","VB","#")
    logger.info("the points for VegALL, VegPerUng4, VegWoody4 summed")

    # Reduce any excess point scores to the maximum allowed
    arcpy.SelectLayerByAttribute_management("fcWUVegFA", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("fcWUVegFA","NEW_SELECTION","""""VegFA
" > 5 AND "Floodplain" = 'N'""")
    arcpy.CalculateField_management("fcWUVegFA","VegFA","5","VB","#")
    logger.info("excess point scores reduced to the maximum allowed")


    # Clean up
    if arcpy.Exists("WU_VegFA1"):
        arcpy.Delete_management("WU_VegFA1")
    if arcpy.Exists("WU_VegFA2"):
        arcpy.Delete_management("WU_VegFA2")
    if arcpy.Exists("WU_VegFA3"):
        arcpy.Delete_management("WU_VegFA3")
```

5.7.15 VegAll: Flood Attenuation Potential

```
##############################################################################
#
# File Name: VegAll.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/12/2016 (modified 11/02/2017)
# Purpose:
#    Input to Flood Attenuation / Potential / Vegetation Factor
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineVegAll(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.VegAll")

    # Clean up if needed
    if arcpy.Exists("WU_VegAll"):
        arcpy.Delete_management("WU_VegAll")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    #arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcENWI")
    arcpy.MakeFeatureLayer_management(r"VegAll","fcVegAll")
    logger.info("feature layers ready")

    # Select the forest, shrubland, emergent, moss, and aquaitc bed vegetation
    '''strWHERE = """"ATTRIBUTE" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR
"ATTRIBUTE" LIKE 'PSS%' OR "ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE
'PML%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    logger.info("the forest, shrubland, emergent, moss, and aquaitc bed vegetation selected")

    # Create layer of all vegetation area
    arcpy.CopyFeatures_management("fcENWI", "WU_VegAll1")
    arcpy.MakeFeatureLayer_management(r"WU_VegAll1","fcWUVegAll1")
    logger.info("layer of all vegetation area created")'''

    # Add field to store vegetation area
    actions.DeleteField("fcVegAll","VegArea")
    arcpy.AddField_management("fcVegAll", "VegArea", "FLOAT")
    arcpy.CalculateField_management("fcVegAll","VegArea","[Shape_Area]","VB","#")
    logger.info("field VegArea added to store vegetation area")

    # Join vegetation to wetland units and sume the vegetation area

##############################################################################
#################
    # SJ: FloodIn

##############################################################################
#################
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcWU")
    fmSJFLIN.addTable("fcVegAll")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","VegArea"]

    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJFLIN.findFieldMapIndex("VegArea")
    fieldmap = fmSJFLIN.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJFLIN.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcVegAll","WU_VegAll","JOIN_ONE_TO_ONE","KEEP
_ALL",fmSJFLIN,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WU_VegAll","fcWUVegAll")
    logger.info("vegetation joined to wetland units and the vegetation area summed up")


    # Add field to store ratio of vegetated area to total area
    actions.DeleteField("fcWUVegAll","VegRatio")
    arcpy.AddField_management("fcWUVegAll", "VegRatio", "FLOAT")

    # Calculate ratio of vegetation to Wetland Units area

arcpy.CalculateField_management("fcWUVegAll","VegRatio","[VegArea]/[Shape_Area]","VB
","#")
    logger.info("field VegRatio added and calculated")

    # Add new attribute field to store points for VegAll and set initial value to zero
    actions.DeleteField("fcWUVegAll","VegAll")
    arcpy.AddField_management("fcWUVegAll", "VegAll", "SHORT")
    arcpy.CalculateField_management("fcWUVegAll","VegAll","0","VB","#")
    logger.info("field VegAll added and initial value set to 0")

    # Assign points to Wetland Units for woody vegetation
    strWHERE = """"VegRatio" > 0.5"""
    arcpy.SelectLayerByAttribute_management("fcWUVegAll", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegAll","VegAll","1","VB","#")
    logger.info("points assigned to Wetland Units for woody vegetation")
    arcpy.SelectLayerByAttribute_management("fcWUVegAll", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("VegAll"):
        arcpy.Delete_management("VegAll")
```

5.7.16 VegPerUng: Flood Attenuation Potential

```
##############################################################################
#
# File Name: VegPerUng.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 7/17/2015 (modified 11/08/2017)
# Purpose:
#     VegPerUng: Used in Water Quality Function / Potential / Vegetation. Max 5 points.
#     VegPerUng4: Used in Flood Attenuation Function / Potential / Vegetation. Max 4 points.
#     VegPerUng1: Used in Habitat and Ecological Integrity Function / Potential / Vegetation.
Max 1 point.
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegPerUng(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.VegPerUng")

    # Clean up if needed
    if arcpy.Exists("VegPFOPSSPEM"):
        arcpy.Delete_management("VegPFOPSSPEM")
    if arcpy.Exists("VegPerUng"):
        arcpy.Delete_management("VegPerUng")
    if arcpy.Exists("WUVegPerUngIntersect"):
        arcpy.Delete_management("WUVegPerUngIntersect")
    if arcpy.Exists("WUVegPerUngIntersect_SUM_STAT"):
        arcpy.Delete_management("WUVegPerUngIntersect_SUM_STAT")
    if arcpy.Exists("WU_VegPerUng"):
        arcpy.Delete_management("WU_VegPerUng")
    if arcpy.Exists("WU_VegPerUng1"):
        arcpy.Delete_management("WU_VegPerUng1")

    # setting the variables
    arcpy.MakeFeatureLayer_management(globalvars.srcPasturesNotHayfields,
"fcPasturesNotHayfields")
    arcpy.MakeFeatureLayer_management(globalvars.srcEnhWetland, "fcEnhWVWetland")
    logger.info("feature layers ready")

    # Create feature class to store VegPerUng variables
    arcpy.CopyFeatures_management(WetlandPoly, "WU_VegPerUng1")
    arcpy.MakeFeatureLayer_management("WU_VegPerUng1", "fcWUVegPerUng1")
    logger.info("feature layer WU_VegPerUng1 created")

    # selecting and creating a feature class of forests, shrublands, and persistent emergent
vegetation sites within EnhWVWetland
    strWHERE = """"ATTRIBUTE" NOT LIKE  'PEM2%' AND ("ATTRIBUTE" LIKE
'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%')"""
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "NEW_SELECTION",
strWHERE)
    arcpy.CopyFeatures_management("fcEnhWVWetland", "VegPFOPSSPEM")
    arcpy.MakeFeatureLayer_management("VegPFOPSSPEM", "fcVegPFOPSSPEM")
    logger.info("feature class of forests, shrublands, and persistent emergent vegetation sites
within EnhWVWetland created")

    # Erase the known grazed pastures from the intermediate vegetation layer
    arcpy.Erase_analysis("fcVegPFOPSSPEM","fcPasturesNotHayfields","VegPerUng","#")
    arcpy.MakeFeatureLayer_management("VegPerUng", "fcVegPerUng")
    logger.info("known grazed pastures from the intermediate vegetation layer erased")

    # Calculate the percentage of each Wetland Unit that is persistent ungrazed vegetation
(VegPerUng)
    # Attribute each Wetland Unit with the percentage of VerPerUng
    actions.DeleteField("fcWUVegPerUng1","OrigArea")
    arcpy.AddField_management("fcWUVegPerUng1", "OrigArea", "DOUBLE")
    arcpy.CalculateField_management("fcWUVegPerUng1", "OrigArea", "!SHAPE_Area!",
"PYTHON")
    logger.info("field OrigArea created and calculated")

    # intersect "fcWUVegPerUng1" with VegPerUng
    arInputData = ["fcWUVegPerUng1","fcVegPerUng"]
    arcpy.Intersect_analysis(arInputData, "WUVegPerUngIntersect","ALL","#","INPUT")
    arcpy.MakeFeatureLayer_management("WUVegPerUngIntersect",
"fcWUVegPerUngIntersect")
    logger.info("Wetland units intersected with VegPerUng")

    # add and calculate PctIntersect field
    actions.DeleteField("fcWUVegPerUngIntersect","PctIntersect")
    arcpy.AddField_management("fcWUVegPerUngIntersect", "PctIntersect", "FLOAT")
    arcpy.CalculateField_management("fcWUVegPerUngIntersect", "PctIntersect",
"!SHAPE_Area!/!OrigArea!*100", "PYTHON")
    logger.info("field PctIntersect added and calculated")

    # Execute the Summary Statistics function
    arcpy.Statistics_analysis("fcWUVegPerUngIntersect",
"WUVegPerUngIntersect_SUM_STAT", [["PctIntersect", "SUM"]],
["FID_WU_VegPerUng1"])
    arcpy.MakeTableView_management(r"WUVegPerUngIntersect_SUM_STAT",
"tvWUVegPerUngIntersectSumm")
    logger.info("summary table WUVegPerUngIntersect_SUM_STAT created")

    # Add new field (VegPerUngPct) to attribute table to store the percentage of persistent
ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng1","VegPerUngPct")
    arcpy.AddField_management("fcWUVegPerUng1", "VegPerUngPct", "FLOAT")
    arcpy.CalculateField_management("fcWUVegPerUng1", "VegPerUngPct", "0", "VB")
    logger.info("field VegPerUngPct added and initial value set to 0")

    # Add new field (VegPerUng) to attribute table to store the score in relation to the percentage
of persistent ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng1","VegPerUng")
    arcpy.AddField_management("fcWUVegPerUng1", "VegPerUng", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng1", "VegPerUng", "0", "VB")
    logger.info("field VegPerUng added and initial value set to 0")

    # Join Wetland Units to the WUVegPerUngIntersect_SUM_STAT table
    arcpy.AddJoin_management( "fcWUVegPerUng1", "WUKey",
"tvWUVegPerUngIntersectSumm", "FID_WU_VegPerUng1")
    logger.info("joined added to the WUVegPerUngIntersect_SUM_STAT table")

arcpy.FeatureClassToFeatureClass_conversion("fcWUVegPerUng1",arcpy.env.workspace,"W
U_VegPerUng")
    arcpy.MakeFeatureLayer_management(r"WU_VegPerUng", "fcWUVegPerUng")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUngPct",
"!WUVegPerUngIntersect_SUM_STAT_SUM_PctIntersect!", "PYTHON")
    logger.info("field VegPerUngPct calculated")

    # Remove Join
    arcpy.RemoveJoin_management("fcWUVegPerUng1")
    logger.info("join removed")

    # Replace null values with 0
    strWHERE = """"VegPerUngPct" IS NULL"""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUngPct", "0", "VB")

    # Assign points to Wetland Units for VegPerUng
    strWHERE = """"VegPerUngPct" > 66.7"""
```

```
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "5", "VB")

    strWHERE = """"VegPerUngPct" < 66.701 AND "VegPerUngPct" > 33.3""""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "3", "VB")

    strWHERE = """"VegPerUngPct" < 33.301 AND "VegPerUngPct" > 10""""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "1", "VB")

    strWHERE = """"VegPerUngPct" < 10.001""""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "0", "VB")
    logger.info("points assigned to field VegPerUng")
    arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","CLEAR_SELECTION")

    # add VegPerUng4 field
    actions.DeleteField("fcWUVegPerUng","VegPerUng4")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUng4", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "0", "VB")
    logger.info("field VegPerUng4 added and initial value set to 0")

    # add VegPerUng1 field
    actions.DeleteField("fcWUVegPerUng","VegPerUng1")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUng1", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng1", "0", "VB")
    logger.info("field VegPerUng1 added and initial value set to 0")

    # Assign points to Wetland Units for VegPerUng4 and VegPerUng1
    strWHERE = """"VegPerUngPct" > 10""""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "1", "VB")

    strWHERE = """"VegPerUngPct" > 33.3""""
```

```
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "2", "VB")

    strWHERE = """"VegPerUngPct" > 50"""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "3", "VB")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng1", "1", "VB")

    strWHERE = """"VegPerUngPct" > 66.7"""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "4", "VB")
    logger.info("points assigned to fields VegPerUng1 and VegPerUng4")
    arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("VegPFOPSSPEM"):
        arcpy.Delete_management("VegPFOPSSPEM")
    if arcpy.Exists("VegPerUng"):
        arcpy.Delete_management("VegPerUng")
    if arcpy.Exists("WUVegPerUngIntersect"):
        arcpy.Delete_management("WUVegPerUngIntersect")
    if arcpy.Exists("WUVegPerUngIntersect_SUM_STAT"):
        arcpy.Delete_management("WUVegPerUngIntersect_SUM_STAT")
    if arcpy.Exists("WU_VegPerUng1"):
        arcpy.Delete_management("WU_VegPerUng1")
```

5.7.17 VegWoody: Flood Attenuation Potential

```
###########################################################################
#
# File Name: VegWoody.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/15/2016 (modified 11/03/2017)
# Purpose:
#    Input to Water Quality / Potential / Vegetation Factor
#
###########################################################################
#!/usr/: Input to Water Quality / Potential / Vegetation Factor.bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegWoody(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FAPotential.VegWoody")

    # Clean up if needed
    if arcpy.Exists("VegPFOPSS"):
        arcpy.Delete_management("VegPFOPSS")
    if arcpy.Exists("VegPFO"):
        arcpy.Delete_management("VegPFO")
    if arcpy.Exists("WUPFOJoin"):
        arcpy.Delete_management("WUPFOJoin")
    if arcpy.Exists("WU_VegWoody"):
        arcpy.Delete_management("WU_VegWoody")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcENWI")
    logger.info("feature layers ready")

    # Select all woody vegetation, both forest and shrubland
    strWHERE = """"ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    logger.info("all woody vegetation, both forest and shrubland selected")

    # Create layer of woody vegetation from selection
    arcpy.CopyFeatures_management("fcENWI", "VegPFOPSS")
    arcpy.MakeFeatureLayer_management(r"VegPFOPSS", "fcVegPFOPSS")
    logger.info("layer VegPFOPSS created of woody vegetation from selection")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    # Add field to store woody area
    actions.DeleteField("fcVegPFOPSS","PFOPSSarea")
    arcpy.AddField_management("fcVegPFOPSS", "PFOPSSarea", "FLOAT")

arcpy.CalculateField_management("fcVegPFOPSS","PFOPSSarea","[SHAPE_Area]","VB","#"
)
    logger.info("field PFOPSSarea to store woody area")

    # Select just the forest vegetation, not including the shrubs
    strWHERE = """"ATTRIBUTE" LIKE 'PFO%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    logger.info("the forest vegetation, not including the shrubs selected")

    # Create layer of forest vegetation from selection
    arcpy.CopyFeatures_management("fcENWI", "VegPFO")
    arcpy.MakeFeatureLayer_management(r"VegPFO", "fcVegPFO")
    logger.info("layer VegPFO created of forest vegetation from selection")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    # Add field to store forest area
    actions.DeleteField("fcVegPFO","PFOarea")
    arcpy.AddField_management("fcVegPFO", "PFOarea", "FLOAT")
    arcpy.CalculateField_management("fcVegPFO","PFOarea","[SHAPE_Area]","VB","#")
    logger.info("field PFOarea added to store forest area")

    # Join forests to Wetland Units and sum the forest area

################################################################################
#################
    # SJ: VegPFO

################################################################################
#################
    fmSJPFO = arcpy.FieldMappings()
    fmSJPFO.addTable("fcWU")
    fmSJPFO.addTable("fcVegPFO")

    fldKeyIndex1 = fmSJPFO.findFieldMapIndex("PFOarea")
    fieldmap1 = fmSJPFO.getFieldMap(fldKeyIndex1)
    fieldmap1.mergeRule = "Sum"
    fmSJPFO.replaceFieldMap(fldKeyIndex1, fieldmap1)
```

```
arcpy.SpatialJoin_analysis("fcWU","fcVegPFO","WUPFOjoin","JOIN_ONE_TO_ONE","KEE
P_ALL",fmSJPFO,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WUPFOjoin","fcWUPFOjoin")
    logger.info("forests joined to wetland units and the forest area summed up")


##################################################################################
#################
    # SJ: VegPFOPSS

##################################################################################
#################
    fmSJPSS = arcpy.FieldMappings()
    fmSJPSS.addTable("fcWUPFOjoin")
    fmSJPSS.addTable("fcVegPFOPSS")

    fldKeyIndex2 = fmSJPSS.findFieldMapIndex("PFOPSSarea")
    fieldmap2 = fmSJPSS.getFieldMap(fldKeyIndex2)
    fieldmap2.mergeRule = "Sum"
    fmSJPSS.replaceFieldMap(fldKeyIndex2, fieldmap2)


arcpy.SpatialJoin_analysis("fcWUPFOjoin","fcVegPFOPSS","WU_VegWoody","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJPSS,"INTERSECT")
    fcWUVegWoody = arcpy.mapping.Layer(r"WU_VegWoody")
    logger.info("woody vegetation joined to wetland units and the woody area summed up")

    # Add fields to store rations of forest and woody area to total area
    actions.DeleteField(fcWUVegWoody,"PFOratio")
    arcpy.AddField_management(fcWUVegWoody, "PFOratio", "FLOAT")
    actions.DeleteField(fcWUVegWoody,"PFOPSSratio")
    arcpy.AddField_management(fcWUVegWoody, "PFOPSSratio", "FLOAT")
    logger.info("fields PFOratio and PFOPSSratio added to store rations of forest and woody area
to total area")

    # Calculate ratio of woody vegetation to Wetland Unit area

arcpy.CalculateField_management(fcWUVegWoody,"PFOPSSratio","[PFOPSSarea]/[Shape_A
rea]","VB","#")
    logger.info("ratio of woody vegetation calculated to Wetland Unit area")

    # Calculate ratio of forest vegetation to Wetland Unit area

arcpy.CalculateField_management(fcWUVegWoody,"PFOratio","[PFOarea]/[Shape_Area]","V
B","#")
```

```
   logger.info("ratio of forest vegetation calculated to Wetland Unit area")


################################################################################
########################
   # Add new attribute field to store points for VegWoody and set initial value to zero

################################################################################
########################
   actions.DeleteField(fcWUVegWoody,"VegWoody")
   arcpy.AddField_management(fcWUVegWoody, "VegWoody", "SHORT")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","0","VB","#")
   logger.info("field VegWoody added to store points for VegWoody and initial value set to
zero")

   actions.DeleteField(fcWUVegWoody,"VegWoody4")
   arcpy.AddField_management(fcWUVegWoody, "VegWoody4", "SHORT")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","0","VB","#")
   logger.info("field VegWoody4 added to store points for VegWoody and initial value set to
zero")

   actions.DeleteField(fcWUVegWoody,"VegWoody2")
   arcpy.AddField_management(fcWUVegWoody, "VegWoody2", "SHORT")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","0","VB","#")
   logger.info("field VegWoody2 added to store points for VegWoody and initial value set to
zero")

   actions.DeleteField(fcWUVegWoody,"VegWoodyFor")
   arcpy.AddField_management(fcWUVegWoody, "VegWoodyFor", "SHORT")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","0","VB","#")
   logger.info("field VegWoodyFor added to store points for VegWoody and initial value set to
zero")

   # Assign points to Wetland Units for woody vegetation
   strWHERE = """"PFOPSSratio" > 0.1"""
   arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","1","VB","#")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","1","VB","#")
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","1","VB","#")

   strWHERE = """"PFOPSSratio" > 0.5"""
   arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
   arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","2","VB","#")
```

```
    strWHERE = """"PFOPSSratio" > 0.333"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","2","VB","#")

    strWHERE = """"PFOPSSratio" > 0.667"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","3","VB","#")

    strWHERE = """"PFOPSSratio" > 0.667 AND "PFOratio" > 0.333"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","4","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","3","VB","#")

    strWHERE = """"PFOratio" > 0.667"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","5","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","4","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","2","VB","#")

    strWHERE = """"PFOratio" > 0.1 OR "PFOarea" > 10000"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","1","VB","#")

    strWHERE = """("PFOratio" > 0.33 AND "PFOarea" > 2000) OR "PFOarea" > 20000"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","2","VB","#")

    strWHERE = """("PFOratio" > 0.667 AND "PFOarea" > 5000) OR "PFOarea" > 50000"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","3","VB","#")
    logger.info("points assigned to Wetland Units for woody vegetation")
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("VegPFOPSS"):
        arcpy.Delete_management("VegPFOPSS")
    if arcpy.Exists("VegPFO"):
        arcpy.Delete_management("VegPFO")
    if arcpy.Exists("WUPFOJoin"):
```

```
arcpy.Delete_management("WUPFOJoin")
```

5.7.18 Flood Attenuation Society

```
################################################################################
#
# File Name: FASociety.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Flood Attenuation Value to Society metrics.
#
################################################################################
import datetime
import logging
import traceback
import arcpy

from Factors import Floodway, EconRisk
from Aspects import FASociety

def procFASociety(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FASociety")

    ################################################################
    ## 1. Run Variables
    ################################################################

    # None

    ################################################################
    ## 2. Run Factors
    ################################################################

    Floodway.DetermineFloodway(WetlandPoly)

    EconRisk.DetermineEconRisk(WetlandPoly)

    ################################################################
    ## 3. Run Aspect
    ################################################################

    FASociety.DetermineFASociety(WetlandPoly)
```

5.7.19 Flood Attenuation Society Aspects

```
##########################################################################
#
# File Name: FASociety.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 11/10/2016 (modified 10/31/2017)
# Purpose:
#    Flood Attenuation Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineFASociety(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FASociety.FASociety")

    # Clean up if needed
    if arcpy.Exists("WU_FASociety"):
        arcpy.Delete_management("WU_FASociety")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Floodway","fcFloodway")
    arcpy.MakeFeatureLayer_management(r"WU_EconRisk","fcEconRisk")
    logger.info("feature layers ready")

    # Spatial join Floodway & EconRisk to bring in factor values
    fmSJE = arcpy.FieldMappings()
    fmSJE.addTable("fcFloodway")
    fmSJE.addTable("fcEconRisk")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodway","EconRisk"]

    for field in fmSJE.fields:
        if field.name not in keepers:
            fmSJE.removeFieldMap(fmSJE.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcFloodway","fcEconRisk","WU_FASociety","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJE,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_FASociety","fcWUFASociety")
    logger.info("spatial join Floodway and EconRisk completed")

    # Add FASociety field to Wetland Units and set initial point value to zero
    actions.DeleteField("fcWUFASociety","FASociety")
    arcpy.AddField_management("fcWUFASociety", "FASociety", "SHORT")
    arcpy.CalculateField_management("fcWUFASociety","FASociety","0","VB","#")
    logger.info("FASociety field added to Wetland Units and initial point value set to zero")

    # Sum the factor points

arcpy.CalculateField_management("fcWUFASociety","FASociety","[Floodway]+[EconRisk]",
"VB","#")
    logger.info("field FASociety calculated")

    # Reduce the values that exceed the maximum allowable points
    strWHERE = """"FASociety" > 4"""

arcpy.SelectLayerByAttribute_management("fcWUFASociety","NEW_SELECTION",strWHE
RE)
    arcpy.CalculateField_management("fcWUFASociety","FASociety","4","VB","#")
    logger.info("points for wetlands reduced to a maximum of 4")
    arcpy.SelectLayerByAttribute_management("fcWUFASociety","CLEAR_SELECTION")
```

5.7.20 EconRisk: Flood Attenuation Society

```
##############################################################################
#
# File Name: EconRisk.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/12/2016 (modified 12/04/2017)
# Purpose:
#    Flood Attenuation / Value to Society
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineEconRisk(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FASociety.EconRisk")

    # Clean up if needed
    if arcpy.Exists("WU_EconRisk"):
        arcpy.Delete_management("WU_EconRisk")

    # Setting python variables
    fcTotalLossRP100 = arcpy.mapping.Layer(globalvars.srcTotalLossRP100)
    fcWS12digit = arcpy.mapping.Layer(globalvars.srcWS12digit)

    # Create new feature class to store EconRisk data
    arcpy.CopyFeatures_management(WetlandPoly, "WU_EconRisk")
    fcWUEconRisk = arcpy.mapping.Layer(r"WU_EconRisk")
    logger.info("feature layers ready")

    # Add field to store EconRisk and set initial value to zero
    actions.DeleteField(fcWUEconRisk,"EconRisk")
    arcpy.AddField_management(fcWUEconRisk, "EconRisk", "SHORT")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","0","VB","#")
    logger.info("field EconRisk added and intital value set to 0")

    # Select Census blocks and assign point to Wetland Units
    # Assign 1 point to wetland in socond lowest quintile or within HUC12 with loss areas

arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""""Tota
lLossRP100" > 0""")
```

```
arcpy.SelectLayerByLocation_management(fcWS12digit,"INTERSECT",fcTotalLossRP100,"#
","NEW_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcWS12digit,"#","
NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","1","VB","#")
    logger.info("1 point assigned to wetland in socond lowest quintile or within HUC12 with loss
areas")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWS12digit, "CLEAR_SELECTION")

    # Assign 2 points to wetlands in middle quintile or within 1km of second highest quintile

arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""Tota
lLossRP100" > 42""")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcTotalLossRP100
,"#","NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","2","VB","#")
    logger.info("2 points assigned to wetland in middle quintile")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")


arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""Tota
lLossRP100" > 200""")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcTotalLossRP100
,"1000 Meters","NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","2","VB","#")
    logger.info("2 points assigned to wetland within 1km of second highest quintile")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWS12digit, "CLEAR_SELECTION")

    # Assign 3 points to wetlands in second highest quintile or within 2km of highest quintile

arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""Tota
lLossRP100" > 200""")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcTotalLossRP100
,"#","NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","3","VB","#")
    logger.info("3 points assigned to wetland in second highest quintile")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")


arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""""Tota
lLossRP100" > 1204""")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcTotalLossRP100
,"1000 Meters","NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","3","VB","#")
    logger.info("3 points assigned to wetland within 2km of highest quintile")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWS12digit, "CLEAR_SELECTION")

    # Asign 4 points to wetlands in highest quintile

arcpy.SelectLayerByAttribute_management(fcTotalLossRP100,"NEW_SELECTION","""""Tota
lLossRP100" > 1204""")

arcpy.SelectLayerByLocation_management(fcWUEconRisk,"INTERSECT",fcTotalLossRP100
,"#","NEW_SELECTION")
    arcpy.CalculateField_management(fcWUEconRisk,"EconRisk","4","VB","#")
    logger.info("4 points assigned to wetland in highest quintile")
    arcpy.SelectLayerByAttribute_management(fcWUEconRisk, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcTotalLossRP100, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWS12digit, "CLEAR_SELECTION")
```

5.7.21 Floodway: Flood Attenuation Society

```
##############################################################################
#
# File Name: Floodway.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/27/2016 (modified 12/04/2017)
# Purpose:
#    Flood Attenuation Function / Value to Society
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineFloodway(WetlandPoly):
    logger = logging.getLogger("WFA.FloodAttn.FASociety.Floodway")

    # Clean up if needed
    if arcpy.Exists("WU_Floodway"):
        arcpy.Delete_management("WU_Floodway")

    # Setting python variables
    fcFHFloodway = arcpy.mapping.Layer(globalvars.srcFHFloodway)

    # Create feature class to store Floodway points
    arcpy.CopyFeatures_management(WetlandPoly, "WU_Floodway")
    fcWUFloodway = arcpy.mapping.Layer(r"WU_Floodway")
    logger.info("feature layers ready")

    # Create field to store Floodway points and set initial value to zero
    actions.DeleteField(fcWUFloodway,"Floodway")
    arcpy.AddField_management(fcWUFloodway, "Floodway", "SHORT")
    arcpy.CalculateField_management(fcWUFloodway, "Floodway", "0", "PYTHON_9.3")
    logger.info("field Floodway created and initial value set to zero")

    # Select Wetland Units that intersect a Floodway

arcpy.SelectLayerByLocation_management(fcWUFloodway,"INTERSECT",fcFHFloodway,"#
","NEW_SELECTION")
    logger.info("Wetland Units that intersect a Floodway selected")
```

```
# Assign Points
arcpy.CalculateField_management(fcWUFloodway, "Floodway", "4", "VB")
logger.info("4 points assigned to Wetland Units that intersect a Floodway")
arcpy.SelectLayerByAttribute_management(fcWUFloodway, "CLEAR_SELECTION")
```

5.7.22 Globalvars

```
##############################################################################
#
# File Name: globalvar.py
# Developer: Yibing Han @ West Virginia GIS Tech Center
# Date: 02/2018
# Purpose:
#    Stores source data paths that are used by all the metrics involved in the Wetland Functional Assessment
#
##############################################################################
#!/usr/bin/python

# Input wetlands polygon
srcInput = ""

# Log file directory
srcLogFolder = "Q:\\WATER RESOURCES\\WAB\\WETLANDS\\Functional Assessment\\3_Code\\FunctionalAssessmentFramework\\logs"

# Results geodatabase directory
srcGDBFolder = "M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WetlandFunctionResults\\test\\"

# Working file directory
srcTempFolder = "M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WorkingFiles"

# Final Database location
srcFinalDatabase = "M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WetlandFunctionResults\\test\\WVWRAMGISresults.mdb"
```

```
# AquaAbund, Histosol, HydSW, IrrEdge, LandPos, MarlPEM, Microtopo, Organic,
SeasonPond, StreamEdge, VegAll, VegByLP, VegPerUng, VegWoody, WFlowPath

#srcEnhWetland =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\EnhancedNWI_20150511.gdb\CON
US_WVWetlandsProj\EnhWVWetland"

srcEnhWetland =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\NWI_WV_20190304.gdb\NWI_WV
_20190304"


# Basic Geometries

srcWUPoint = r"WUPoint"

srcBuffer10m = r"Buffer10m"

srcBuffer50m = r"Buffer50m"

srcBuffer300m = r"Buffer300m"

srcBuffer1km = r"Buffer1km"

srcDrainageArea = r"DrainageArea"


# AquaAbund

#srcBuffer1km =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\Buffer1km"

srcStreams = r"M:\LayerFiles\arcsde_backup.gdb\wr\NHD_with_stream_codes"


# BRank

srcBRankInput =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Habitat
Data.gdb\BRankInput"


# BRankHUC

srcBRHUC =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\watershedBiodive
rsityRankHUC_WVDNR_12Nov2014_utm83.shp"
```

```
# BufferContig, BufferPerim, Disturb50m, DisturbWshd
srcDisturbedLand =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\DisturbedLand"

# BufferContig
#srcBuffer300m =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\Buffer300m"

# BufferPerim
#srcBuffer10m =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\Buffer10m"
srcInterstates = r"M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\Interstates"
srcPrimaryRoads = r"M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\Primary_Roads"
srcLocalRoads = r"M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\Local_Roads"
srcOtherRoads =
r"M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\Other_Roads_And_Trails"
srcRailway =
r"M:\LayerFiles\arcsde_backup.gdb\basemap_cultural_non_replica\SDE_railway_tiger" # Also:
RoadRail

# Clay
srcPalustringPlots =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\PalustrineP
lotsMarch2015"
srcSSURGO =
"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\SsurgoE
xports.gdb\SsurgoClay"

# ConsFocus
```

```
srcCFArea =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Habitat
Data.gdb\ConsFocusArea"


# Discharges

srcSeptic =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\Septic"

srcOWRNPDES = r"M:\wr\owrnpdes_.shp" # Also: Septic

srcOWRNPDESOutlets = r"M:\wr\owrnpdes_outlets.shp"

srcHPU = r"M:\mr\hpu.shp"

srcAMLAMD =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\AMLAMDFeb2016"

srcWellPads =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\WellPads_20160325"

srcNPLPoint =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\NPL_point_20160406"

srcNPLBndry =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\NPL_Bndry_20160406"


# DisturbWshd, LandPos, RunoffWshd, SlopeWshd, WFlowPath, ImpairedIn

#srcDrainageArea =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\DrainageArea27m
"


# DrainageArea

srcFlowDir =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Waters
hed.gdb\hydrogrid_16U_flowdir_27m"
```

```
# EconRisk
srcTotalLossRP100 = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\FloodplainData.gdb\TotalLossRP100"
srcWS12digit = r"M:\basemap\watersheds_12digit.shp"


# Fisheries
srcTrout = r"M:\wr\WTRSHD_BRANCH\TROUT\Trout_Streams.shp"
srcHQSFisheries = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\DNR_Fishing\HighQualityStreamFisheriesWVDNR20150820.shp"
srcTroutStreams = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\DNR_Fishing\TrStStreams.shp"


# FloodArea
srcFPARAFEMA = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\FloodplainData.gdb\FloodplainARAFEMA"
srcPeatlands = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\Peatlands_20160228" # Also: Histosol, Organic, SoilRunoff, WFlowPath


# Floodway
srcFHFloodway = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Floodplain\wvFlodHazardFeatures_WVGISTC_20130410\wvFloodHazardFeatures20130207.gdb\WV_Floodway_20130205_wgs84wmA"


# HInvest, OwnerAccess, PublicUse
srcLocalPark = r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\countyCityParkBoundaries_20107731_utm83.gdb\countyCityParkBoundaries_20170731_utm83"
```

```
srcNF =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\
nationalForestOwnership_USFWS_20170803_utm83.gdb\nationalForestOwnership_USFWS_2
0170803_utm83" # Also: WQPlan

srcNP =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\
nationalParkBoundaries_nationalParkService_20170802.gdb\nationalParkBoundaries_nationalP
arkService_20170802" # Also: WQPlan

srcNWR =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\
nationalWildlifeRefuge_USFWS_20170803_utm83.gdb\nationalWildlifeRefuge_USFWS_201
70803_utm83" # Also: WQPlan

srcWMA =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\
wvdnrManagedLands_wvdnr_20170731_utm83.gdb\wvdnrManagedLands_wvdnr_20170731_
utm83"

srcSP =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\s
tateParkBoundaries_WVDNR_20170927_utm83.gdb\stateParkBoundaries_WVDNR_2017092
7_utm83"

srcSF =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Boundaries2017\
wvStateForestBoundaries_wvdof_20171003_utm83.gdb\wvStateForestBoundaries_wvdof_201
71003_utm83"

srcBotanicalAreas =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\USFS\botanical_a
reas_MNF.shp"

srcProtectedLands =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\WV_Protected_L
ands_2015_PUBLIC\WV_Protected_Lands_2015_PUBLIC.shp"

srcNatStrPreAct =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Waters
hed.gdb\NatStrPreAct_HUC10" # Also: WQPlan

srcILF =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\ILF_banks
"
```

```
srcRestoredWetlands =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\RestoredW
etlands"

srcInfrastructure =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\Infrastruct
ureWetlands"

srcFishAccess =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\DNR_Fishing\Pu
blicFishingAccessSites_2017_10.shp"

srcPropertyBoundary =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\201710_WVDNR
_property_boundary.gdb\PropertyBoundaries_WVDNR_20171011"

srcEBird =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Habitat
Data.gdb\eBirdHotspots_20171011"

srcTrails =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\trails_Sep_27_20
17_webmercator.shp" #also: BufferPerim

srcExempBranked =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\Exemplary
OrBrankedWetlands31Mar2015"



# Histosol, Organic

srcPalustrineplots =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandsGeodatasets.gdb\PalustrineP
lotsMarch2015"

srcHisticEpipedon =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Ssurgo
Exports.gdb\HisticEpipedon"

srcHistosol =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Ssurgo
Exports.gdb\Histosol"

srcSSURGOWV = r"M:\basemap\ssurgo\SSURGO.gdb\ssurgo_wv"
```

```
srcSsurgoOrganic =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Ssurgo
Exports.gdb\SsurgoOrganic"


# HUC12WQ, ImpairedIn, ImpairedOut

srcHUC12s = r"M:\basemap\watersheds_12digit.shp"

srcPublicFishingLakes =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\DNR_Fishing\Pu
blicFishingLakesWVDNR20150820.shp"

srcAlgalStreams =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\AlgalStreams"

srcAlgalLakes =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\AlgalLakes"

srcImpairedStreams =
r"M:\wr\WTRSHD_BRANCH\303D_TMDL_IMPAIRED\WV2016_ImpairedStreams_24KNH
D.shp"

srcLimeStone = r"M:\basemap\geology_shapefiles\type\geology-TYPE-limestone.shp"

srcDolostone = r"M:\basemap\geology_shapefiles\type\geology-TYPE-dolostone.shp"

#srcEPAOverlist =
r"M:\wr\WTRSHD_BRANCH\303D_TMDL_IMPAIRED\Historic_GIS_ImpairedStreams\WV
2014_EPAOverlist_24kNHD_20170215.shp"


# HydSW

srcNWIOpenWater =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIEx
ports.gdb\NWIOpenWater"


# IrrEdge

srcRiversLakes =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIEx
ports.gdb\RiversLakes"
```

```
# Karst

srcKarstComp =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\KarstComposite"


# LandInteg

srcForestPatches =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\forest_patches_ov
er50acres_WVplus10mi.shp"

#srcResilientConnected =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\Resilient_and_Co
nnected_Landscapes\Resilient_and_Connected_Data.gdb\Resilient_and_Connected"

srcResilientConnected =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Res_C
onnected"

srcLIIndex =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceAsReceived\landscapeIntegrit
yIndex_WVDNR_2008_utm83_img\landscapeIntegrityIndex_WVDNR_2008_utm83.img"

#srcWUPoint =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\WUpoint"

srcIEI =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Habitat
Data.gdb\IEIUMa2010v32"


# LandPos

srcWBRivers = r"M:\basemap\national_hydrology_dataset\wb-rivers.shp"

srcFSOFlow =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\FirstSecondOrderFlowlines"


# MarlPEM

srcMarlSoils =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Ssurgo
Exports.gdb\MarlSoils"
```

```
# RoadRail

srcTransUTMAllRoads = r"M:\basemap\tiger_2013\WV_Transportation_UTM.gdb\All_Roads"


# Runoff50m, RunoffWshd

srcRunoffLand =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\Runoffland"

#srcBuffer50m =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandUnits.gdb\Buffer50m"


# Septic

srcSAMBPoints = r"M:\basemap\WVSAMB\structures_SAMB_points_UTM83.shp"

srcSeweredAreas =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\SeweredAreas"

srcSepticFailed =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\SepticFailureRiskStatsgo"

srcUrbanized = r"M:\LayerFiles\arcsde_backup.gdb\tiger2010\urbanized_areas"


# Slope, SlopeWshd

#srcSlopePCT =
r"M:\dems\ned_slope_aspect.gdb\NED_3meter_meters_augmented_slope_pct_int"

srcSlopePCT =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\slope_p
ct_int"


# StreamEdge

srcRivers =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIEx
ports.gdb\Rivers"
```

```
# VegByLP
srcLakes =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\NWIEx
ports.gdb\Lakes"


# VegPerUng
srcPasturesNotHayfields =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\PasturesNotHayfields"


# WatershedPlan, WQPlan
srcWatershedPlan =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\WatershedPlan"


# WaterSupply
PSWI_WATERSHEDS = r"M:\environmental\CONFIDENTIAL-
public_surface_water_intakes\pswi_distance_analysis_9m.gdb\pswi_watersheds_with_out_of_s
tate_drainage"
ZPC_5HR = r"M:\environmental\CONFIDENTIAL-
public_surface_water_intakes\CONFIDENTIAL-
source_water_assessment_and_protection.gdb\ZPC_statewide_5hrabove"
ZCC_WV = r"M:\environmental\CONFIDENTIAL-
public_surface_water_intakes\CONFIDENTIAL-
source_water_assessment_and_protection.gdb\ZCC_statewide"
PROTECTION_AREAS = r"M:\environmental\CONFIDENTIAL-
public_surface_water_intakes\CONFIDENTIAL-
source_water_assessment_and_protection.gdb\Source_Water_Protection_Areas"
PSWI_TEMPLATE =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\templat
es.gdb\pwsi_scoring_output_template"


# WetlandBird
```

```
srcWetBird =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Habitat
Data.gdb\WetlandBirds_WetBirdColumnOnly"


# WFlowPath

srcNHDWB24kRivers =
r"M:\LayerFiles\arcsde_backup.gdb\basemap_physical_non_replica\SDE_NHD_waterbodies_2
4k_rivers"

srcNHDFlowline = r"M:\basemap\NHD_H_West_Virginia.gdb\Hydrography\NHDFlowline" #
Also: StreamEdge


# WshdPos

srcDrainage =
r"M:\LayerFiles\arcsde_backup.gdb\basemap_physical_non_replica\SDE_NHD_reach_24k_gt_
50_mi_drainage"


# WshdUniq

srcHUCWetSizeUniq =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Waters
hed.gdb\HUCWetlandSizeUniq"


# WQUse

srcSwimmingAreas =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\SwimmingAreas2016"


# WQPlan

srcTMDL =
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\SourceFunctionalAssessment\Water
QualityDatasets.gdb\TMDL"
```

5.7.23 Habitat and Ecological Integrity

```
##############################################################################
#
# File Name: HabEco.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all metrics within the Habitat & Ecological Integrity
module.
#
##############################################################################
#!/usr/bin/python
import sys
import arcpy
import datetime
import logging
import traceback

from HPotential import HPotential
from HOpportun import HOpportun
from HSociety import HSociety
from HFunction import HFuncNoBR, BRank, HFunction

logger = logging.getLogger("WFA.HabEco")

def RunHabEco(WetlandPoly):

    ## 1. Run Habitat and Ecological Integrity Opportunity Variables/Aspects/Factors
    logger.info("Running Habitat and Ecological Integrity Opportunity
Variables/Aspects/Factors...")
```

```
    try:
        HOpportun.procHOpportun(WetlandPoly)
        logger.info("Habitat and Ecological Integrity Opportunity Variables/Aspects/Factors
completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 2. Run Habitat and Ecological Integrity Potential Variables/Aspects/Factors
    logger.info("Running Habitat and Ecological Integrity Potential
Variables/Aspects/Factors...")
    try:
        HPotential.procHPotential(WetlandPoly)
        logger.info("Habitat and Ecological Integrity Potential Variables/Aspects/Factors
completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 3. Run Habitat and Ecological Integrity Society Variables/Aspects/Factors
    logger.info("Running Habitat and Ecological Integrity Society Variables/Aspects/Factors...")
    try:
        HSociety.procHSociety(WetlandPoly)
        logger.info("Habitat and Ecological Integrity Society Variables/Aspects/Factors
completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 4. Run Habitat and Ecological Integrity Function to roll up Variables/Aspects/Factors
excluding BRank
    logger.info("Running Habitat and Ecological Integrity Function to roll up
Variables/Aspects/Factors excluding BRank...")
    try:
        HFuncNoBR.CalcHFuncNoBR()
        logger.info("Habitat and Ecological Integrity Function to roll up Variables/Aspects/Factors
excluding BRank completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 5. Run BRank Function
    logger.info("Running BRank Function...")
    try:
        BRank.CalcBRank(WetlandPoly)
        logger.info("BRank Function completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 6. Run Habitat and Ecological Integrity Function to roll up Variables/Aspects/Factors
    logger.info("Running Habitat and Ecological Integrity Function to roll up
Variables/Aspects/Factors...")
    try:
        HFunction.CalcHFunction()
        logger.info("Habitat and Ecological Integrity Function to roll up Variables/Aspects/Factors
completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)
```

5.7.24 BRank: Habitat and Ecological Integrity Function

```
############################################################################
#
# File Name: BRank.py
# Developer: Yibing Han
# Date: 10/17/2017 (revised 11/6/2017)
# Purpose:
#    Input to Habitat / Value to Society / HUse
#
############################################################################
#!/usr/bin/python
import sys
sys.path.append("../..")

import arcpy
from utilities import actions
from globalvars import globalvars
import logging

def CalcBRank(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.BRank")

    # Clean up if needed
    if arcpy.Exists("WU_BRank"):
        arcpy.Delete_management("WU_BRank")
    if arcpy.Exists("WU_BRank_join"):
        arcpy.Delete_management("WU_BRank_join")
    if arcpy.Exists("WU_BRank_summ"):
        arcpy.Delete_management("WU_BRank_summ")
    if arcpy.Exists("WU_BRank_summB5"):
        arcpy.Delete_management("WU_BRank_summB5")
    if arcpy.Exists("WU_BRank_summB4"):
        arcpy.Delete_management("WU_BRank_summB4")
    if arcpy.Exists("WU_BRank_summB3"):
        arcpy.Delete_management("WU_BRank_summB3")
    if arcpy.Exists("WU_BRank_summB2"):
        arcpy.Delete_management("WU_BRank_summB2")
    if arcpy.Exists("WU_BRank_summB1"):
        arcpy.Delete_management("WU_BRank_summB1")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcBRankInput, 'fcBRankInput')
    logger.info("feature layers ready")
```

```
   ## STEP 1: Create feature class and add fields to store BRank values
   arcpy.CopyFeatures_management("fcWU","WU_BRank","#","0","0","0")
   arcpy.MakeFeatureLayer_management(r"WU_BRank", "fcBRank")
   logger.info("feature class WU_BRank created")

   ## Add fields to store partial B-Rank based on single elements, concentrations, and final
BRank
   actions.DeleteField("fcBRank", "BSing")
   actions.DeleteField("fcBRank", "BConc")
   actions.DeleteField("fcBRank", "BRank")
   arcpy.AddField_management("fcBRank", "BSing", "TEXT", field_length = 10)
   arcpy.AddField_management("fcBRank", "BConc", "TEXT", field_length = 10)
   arcpy.AddField_management("fcBRank", "BRank", "TEXT", field_length = 10)
   logger.info("fields added to store partial B-Rank based on single elements, concentrations,
and final BRank")


   ## STEP 2: Assign Site Biodiversity Ranks based on single element occurrences
   ## Rank B6 selection
   strWHERE = """"Srank" = 'S3' AND "OQrank" IN ( 'B', 'C')"""
   arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
   arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
   arcpy.CalculateField_management("fcBRank","BSing","'B6'","PYTHON")
   logger.info("B6 selection ranked")

   ## Rank B5 selection
   strWHERE = """("Grank" = 'G3' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" =
'BestState')) OR
               ("Srank" = 'S1' AND "OQrank" = 'C') OR ("Srank" = 'S2' AND "OQrank" IN
('B','C')) OR
               ("Srank" = 'S3' AND "OQrank" = 'A') OR ("SpecComm" = 'C' AND "Grank" IN
('G4', 'G5') AND "OQrank" = 'C')"""
   arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
   arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
   arcpy.CalculateField_management("fcBRank","BSing","'B5'","PYTHON")
   logger.info("B5 selection ranked")

   ## Rank B4 selection
   strWHERE = """("Grank" = 'G2' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" =
'BestState') ) OR
               ("Grank" = 'G3' AND "OQrank" = 'C') OR ("Srank" = 'S1' AND "OQrank" IN ('A',
'B')) OR
               ("Srank" = 'S1' AND "OQrank" = 'C' AND ("EO_Count" = 1 OR "Flag" =
'BestState') ) OR
```

```
                ("Srank" = 'S2' AND "OQrank" = 'A') OR ("Grank" IN ('G4', 'G5') AND "OQrank"
IN ('A', 'B', 'C')
                AND "Flag" = 'Disjunct') OR ("SpecComm" = 'C' AND "Grank" IN ('G4', 'G5')
AND "OQrank" IN ('A', 'B'))"""
    arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
    arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
    arcpy.CalculateField_management("fcBRank","BSing","'B4'","PYTHON")
    logger.info("B4 selection ranked")

    ## Rank B3 selection
    strWHERE = """("Grank" = 'G1' AND "OQrank" = 'D' AND ("EO_Count" = 1 OR "Flag" =
'BestState') ) OR
                ("Grank" = 'G2' AND "OQrank" = 'C') OR ("Grank" = 'G3' AND "OQrank" IN ('A',
'B')) OR
                ("Srank" = 'S1' AND "OQrank" IN ('A', 'B') AND "Flag" = 'Disjunct') OR
                (("SpecComm" = 'C' AND "Grank" IN ('G4', 'G5')) AND "OQrank" IN ('A', 'B')
AND "Flag" = 'Best5Ecoregion')"""
    arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
    arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
    arcpy.CalculateField_management("fcBRank","BSing","'B3'","PYTHON")
    logger.info("B3 selection ranked")

    ## Rank B2 selection
    strWHERE = """("Grank" = 'G1' AND "OQrank" IN ('B', 'C')) OR ("Grank" = 'G2' AND
"OQrank" IN ('A','B')) OR
                ("Grank" = 'G3' AND "OQrank" IN ('A', 'B') AND "Flag" = 'Best5Range')"""
    arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
    arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
    arcpy.CalculateField_management("fcBRank","BSing","'B2'","PYTHON")
    logger.info("B2 selection ranked")

    ## Rank B1 selection
    strWHERE = """("Flag" = 'OnlyRange') OR ("Grank" = 'G1' AND "OQrank" = 'A') OR
                ("Grank" = 'G1' AND "OQrank" IN ('B', 'C') AND ("EO_Count" = 1 OR "Flag" =
'BestState'))"""
    arcpy.SelectLayerByAttribute_management("fcBRankInput", "NEW_SELECTION",
strWHERE)
    arcpy.SelectLayerByLocation_management("fcBRank", "INTERSECT", "fcBRankInput")
    arcpy.CalculateField_management("fcBRank","BSing","'B1'","PYTHON")
    logger.info("B1 selection ranked")
    arcpy.SelectLayerByAttribute_management("fcBRankInput", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcBRank", "CLEAR_SELECTION")
    ## STEP 3: Join Wetland Units with BRankInput
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable('fcWU')
    fmSJ.addTable('fcBRankInput')
    #removeFields = []
    #removeFields = ['Shape_Length1','Shape_Area1']
    #for field in fmSJ.fields:
        #if field.name in removeFields:
            #fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcWU', 'fcBRankInput', 'WU_BRank_join',
'JOIN_ONE_TO_MANY', 'KEEP_ALL', fmSJ, 'INTERSECT')
    arcpy.MakeFeatureLayer_management('WU_BRank_join', 'fcBRankJoin')
    logger.info("Wetland Units joined with BRankInput")


    ## STEP 4: Find the highest occurrence quality rank for each element in each Wetland Unit
    ## Set null values of RandID to zero to allow summarize function to work
    strWHERE = """"RandID" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcBRankJoin","RandID","0","VB","#")
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "CLEAR_SELECTION")
    logger.info("null values of RandID set to zero to allow summarize function to work")

    ## Add field to hold integer transformation of OQRank
    actions.DeleteField("fcBRankJoin", "OQrankInt")
    arcpy.AddField_management("fcBRankJoin", "OQrankInt", "SHORT")
    strWHERE = """"OQrank" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcBRankJoin","OQrankInt","0","VB","#")

    strWHERE = """"OQrank" = 'D'"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcBRankJoin","OQrankInt","1","VB","#")

    strWHERE = """"OQrank" = 'C'"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcBRankJoin","OQrankInt","2","VB","#")

    strWHERE = """"OQrank" = 'B'"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
```

```
    arcpy.CalculateField_management("fcBRankJoin","OQrankInt","3","VB","#")

    strWHERE = """"OQrank" = 'A'"""
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcBRankJoin", "OQrankInt", "4", "VB", "#")
    logger.info("field added to hold integer transformation of OQRank")
    arcpy.SelectLayerByAttribute_management("fcBRankJoin", "CLEAR_SELECTION")


    ## Find the highest OQrank for each WUKey |RandID pair
    actions.DeleteField("fcBRankJoin", "Concat")
    arcpy.AddField_management("fcBRankJoin", "Concat", "TEXT", field_length = 20)
    arcpy.CalculateField_management("fcBRankJoin","Concat","""[WUKey]&" | "&
[RandID]""","VB","#")
    arcpy.Statistics_analysis("fcBRankJoin", "WU_BRank_summ", [["WUKey", "MIN"],
["RandID", "MIN"], ["Srank", "FIRST"], ["Grank", "FIRST"], ["SpecComm", "FIRST"],
["OQrankInt", "MAX"]], ["Concat"])
    arcpy.MakeTableView_management(r"WU_BRank_summ", "tvBRankSumm")
    logger.info("the highest OQrank for each WUKey |RandID pair found")


    ## STEP 5: Assign Site Biodiversity Rank based on concentrations of elements
    ## Rank B5 selection (Rank B6 does not have a criterion for concentrations)
    ## Select B- or C-ranked occurrences of S3 elements
    strWHERE = """"First_Srank" = 'S3' AND "Max_OQrankInt" IN (2,3)"""
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "NEW_SELECTION",
strWHERE)
    arcpy.Statistics_analysis("tvBRankSumm", "WU_BRank_summB5", [["MIN_WUKey",
"COUNT"]], ["MIN_WUKey"])
    arcpy.MakeTableView_management(r"WU_BRank_summB5", "tvBRankSummB5")
    logger.info("all unique elements within a wetland unit counted")

    ## Join table back to WU_BRank
    arcpy.AddJoin_management( "fcBRank", "WUKey", "tvBRankSummB5", "MIN_WUKey")
    logger.info("table WU_BRank_summB5 joined back to WU_BRank")

    ## Select records with 4 or more elements and assign value to BConc
    strWHERE = """WU_BRank_summB5.COUNT_MIN_WUKey > 3"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BConc", "'B5'", "PYTHON")
    logger.info("records with 4 or more elements selected and value assigned to BConc")

    ## Remove Join
    arcpy.RemoveJoin_management("fcBRank")
    logger.info("join removed")
    ## Rank B4 selection
    ## Select C-ranked S1, B- or C-ranked S2, and A-ranked S3 element occurrences and C-
ranked G4 or G5 communities
    strWHERE = """( "First_Srank" = 'S1' AND "Max_OQrankInt" = 2) OR ("First_Srank" = 'S2'
AND "Max_OQrankInt" IN (2,3)) OR
                ("First_Srank" = 'S3' AND "Max_OQrankInt" = 4) OR ("First_SpecComm" = 'C'
AND
                "First_Grank" IN ('G4', 'G5') AND "Max_OQrankInt" = 2)"""
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "NEW_SELECTION",
strWHERE)
    arcpy.Statistics_analysis("tvBRankSumm", "WU_BRank_summB4", [["MIN_WUKey",
"COUNT"]], ["MIN_WUKey"])
    arcpy.MakeTableView_management(r"WU_BRank_summB4", "tvBRankSummB4")
    logger.info("C-ranked S1, B- or C-ranked S2, and A-ranked S3 element occurrences and C-
ranked G4 or G5 communities selected")

    ## Join table back to WU_BRank
    arcpy.AddJoin_management( "fcBRank", "WUKey", "tvBRankSummB4", "MIN_WUKey")
    logger.info("table WU_BRank_summB4 joined back to WU_BRank")

    ## Select records with 4 or more elements and assign value to BConc
    strWHERE = """WU_BRank_summB4.COUNT_MIN_WUKey > 3"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BConc", "'B4'", "PYTHON")
    logger.info("records with 4 or more elements selected and value assigned to BConc")

    ## Remove Join
    arcpy.RemoveJoin_management("fcBRank")
    logger.info("join removed")

    ## Rank B3 selection
    ## Select C-ranked occurrences of G3 elements and A- or B-ranked occurrences of S1
elements
    strWHERE = """("First_Grank" = 'G3' AND "Max_OQrankInt" = 2) OR ("First_Srank" =
'S1' AND "Max_OQrankInt" IN (3,4))"""
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "NEW_SELECTION",
strWHERE)
    arcpy.Statistics_analysis("tvBRankSumm", "WU_BRank_summB3", [["MIN_WUKey",
"COUNT"]], ["MIN_WUKey"])
    arcpy.MakeTableView_management(r"WU_BRank_summB3", "tvBRankSummB3")
    logger.info("C-ranked occurrences of G3 elements and A- or B-ranked occurrences of S1
elements selected")

    ## Join table back to WU_BRank
    arcpy.AddJoin_management( "fcBRank", "WUKey", "tvBRankSummB3", "MIN_WUKey")
    logger.info("table WU_BRank_summB3 joined back to WU_BRank")

    ## Select records with 4 or more elements and assign value to BConc
    strWHERE = """WU_BRank_summB3.COUNT_MIN_WUKey > 3"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BConc", "'B3'", "PYTHON")
    logger.info("records with 4 or more elements selected and value assigned to BConc")

    ## Remove Join
    arcpy.RemoveJoin_management("fcBRank")
    logger.info("join removed")

    ## Rank B2 selection
    ## Select C-ranked occurrences of G2 elements and A- or B-ranked occurrences of G3
elements
    strWHERE = """("First_Grank" = 'G2' AND "Max_OQrankInt" = 2) OR ("First_Grank" =
'G3' AND "Max_OQrankInt" IN (3,4))"""
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "NEW_SELECTION",
strWHERE)
    arcpy.Statistics_analysis("tvBRankSumm", "WU_BRank_summB2", [["MIN_WUKey",
"COUNT"]], ["MIN_WUKey"])
    arcpy.MakeTableView_management(r"WU_BRank_summB2", "tvBRankSummB2")
    logger.info("C-ranked occurrences of G2 elements and A- or B-ranked occurrences of G3
elements selected")

    ## Join table back to WU_BRank
    arcpy.AddJoin_management( "fcBRank", "WUKey", "tvBRankSummB2", "MIN_WUKey")
    logger.info("table WU_BRank_summB2 joined back to WU_BRank")

    ## Select records with 4 or more elements and assign value to BConc
    strWHERE = """WU_BRank_summB2.COUNT_MIN_WUKey > 3"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BConc", "'B2'", "PYTHON")
    logger.info("records with 4 or more elements selected and value assigned to BConc")

    ## Remove Join
    arcpy.RemoveJoin_management("fcBRank")
    logger.info("join removed")

    ## Rank B1 selection
    ## Select B-ranked occurrences of G1 elements and A- or B-ranked occurrences of G2
elements
    strWHERE = """("First_Grank" = 'G1' AND "Max_OQrankInt" = 3) OR ("First_Grank" =
'G2' AND "Max_OQrankInt" IN (3,4))"""
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "NEW_SELECTION",
strWHERE)
```

```
    arcpy.Statistics_analysis("tvBRankSumm", "WU_BRank_summB1", [["MIN_WUKey",
"COUNT"]], ["MIN_WUKey"])
    arcpy.MakeTableView_management(r"WU_BRank_summB1", "tvBRankSummB1")
    logger.info("B-ranked occurrences of G1 elements and A- or B-ranked occurrences of G2
elements selected")

    ## Join table back to WU_BRank
    arcpy.AddJoin_management( "fcBRank", "WUKey", "tvBRankSummB1", "MIN_WUKey")
    logger.info("table WU_BRank_summB1 joined back to WU_BRank")

    ## Select records with 4 or more elements and assign value to BConc
    strWHERE = """WU_BRank_summB1.COUNT_MIN_WUKey > 3"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BConc", "'B1'", "PYTHON")
    logger.info("records with 4 or more elements selected and value assigned to BConc")

    ## Remove Join
    arcpy.RemoveJoin_management("fcBRank")
    logger.info("join removed")
    arcpy.SelectLayerByAttribute_management("tvBRankSumm", "CLEAR_SELECTION")

    ## STEP 6: Calculate final Site Biodiversity Rank
    strWHERE = """"BSing" = 'B6'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B6'", "PYTHON")

    strWHERE = """"BSing" = 'B5' OR "BConc" = 'B5'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B5'", "PYTHON")

    strWHERE = """"BSing" = 'B4' OR "BConc" = 'B4'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B4'", "PYTHON")

    strWHERE = """"BSing" = 'B3' OR "BConc" = 'B3'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B3'", "PYTHON")

    strWHERE = """"BSing" = 'B2' OR "BConc" = 'B2'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B2'", "PYTHON")

    strWHERE = """"BSing" = 'B1' OR "BConc" = 'B1'""""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'B1'", "PYTHON")
    logger.info("final Site Biodiversity Ranks calculated")
    ## Populate the "Null" BRank records with "none"
    strWHERE = """"BRank" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "'none'", "PYTHON")
    logger.info("the 'Null' BRank records populated with 'none'")

    arcpy.SelectLayerByAttribute_management("fcBRank", "CLEAR_SELECTION")

    ## Note that it is preferable to have WVNHP staff check B1 sites manually to ensure they are
correct
    ## Several Wetland Units should have the BRank changed to "NULL" because the qualifying
element
    ## occurrences are not wetland species nor wetland communities.

    ## Add fields to store expert review input from the WV Natural Heritage Program
    arcpy.AddField_management("fcBRank", "WVNHP_BRank", "TEXT", field_length = 10)
    arcpy.AddField_management("fcBRank", "WVNHP_note", "TEXT", field_length = 254)
    logger.info("fields added to store expert review input from the WV Natural Heritage
Program")

    '''
    ## Solicit review from WVNHP experts or their designees regarding B1 and B2 sites (and
any other sites of interest).

    ## Check the "WVNHP_BRank" field and if it is populated, change the rank to the
WVNHP_BRank.
    strWHERE = """"WVNHP_BRank" <>''"""
    arcpy.SelectLayerByAttribute_management("fcBRank", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcBRank", "BRank", "[WVNHP_BRank]", "VB")
    logger.info("the 'Null' BRank records populated with 'none'")
    '''
    # Clean up
    if arcpy.Exists("WU_BRank_join"):
        arcpy.Delete_management("WU_BRank_join")
    if arcpy.Exists("WU_BRank_summ"):
        arcpy.Delete_management("WU_BRank_summ")
    if arcpy.Exists(r"WU_BRank_summB5"):
        arcpy.Delete_management(r"WU_BRank_summB5")
    if arcpy.Exists(r"WU_BRank_summB4"):
        arcpy.Delete_management(r"WU_BRank_summB4")
    if arcpy.Exists(r"WU_BRank_summB3"):
        arcpy.Delete_management(r"WU_BRank_summB3")
    if arcpy.Exists(r"WU_BRank_summB2"):
        arcpy.Delete_management(r"WU_BRank_summB2")
    if arcpy.Exists(r"WU_BRank_summB1"):
```

```
arcpy.Delete_management(r"WU_BRank_summB1")
```

5.7.25 HFuncNoBR: Habitat and Ecological Integrity Function

```
###############################################################################
#
# File Name: HFuncNoBR.py
# Developer: Yibing Han
# Date: 10/19/2017
# Purpose:
#    Input to Habitat and Ecological Integrity Function
#
###############################################################################
#!/usr/bin/python
import sys
sys.path.append("../..")

import arcpy
from utilities import actions
import logging

def CalcHFuncNoBR():
    logger = logging.getLogger("WFA.HabEco.HFuncNoBR")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_HPotential","fcHPotential")
    arcpy.MakeFeatureLayer_management(r"WU_HSociety","fcHSociety")
    arcpy.MakeFeatureLayer_management(r"WU_HOpportun","fcHOpportun")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_HFuncNoBR1'):
        arcpy.Delete_management('WU_HFuncNoBR1')
    if arcpy.Exists('WU_HFuncNoBR'):
        arcpy.Delete_management('WU_HFuncNoBR')

    ## Spatial Join to merge metrics and create feature class to store HFuncNoBR
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcHPotential")
    fmSJFLIN.addTable("fcHOpportun")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','HPotential','HOpportun']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis("fcHPotential", "fcHOpportun", 'WU_HFuncNoBR1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_HFuncNoBR1", "fcHFuncNoBR1")
    logger.info("spatial join HPotential and HSociety completed")

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcHFuncNoBR1")
    fmSJFLIN.addTable("fcHSociety")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','HPotential','HOpportun',"HSociety"]
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis("fcHFuncNoBR1", "fcHSociety", 'WU_HFuncNoBR',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcHFuncNoBR = arcpy.mapping.Layer(r"WU_HFuncNoBR")
    logger.info("spatial join HFuncNoBR1 and HSociety completed")

    ## Add HFuncNoBR field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcHFuncNoBR, 'HFuncNoBR')
    arcpy.AddField_management(fcHFuncNoBR, 'HFuncNoBR', 'SHORT')
    arcpy.CalculateField_management(fcHFuncNoBR, 'HFuncNoBR', '0', 'VB', '#')
    logger.info("field HFuncNoBR added and initial value set to 0")

    ## Sum the points for HInvest and HUse
    arcpy.CalculateField_management(fcHFuncNoBR, 'HFuncNoBR', '[HPotential] +
[HOpportun] + [HSociety]', 'VB', '#')
    logger.info("field HFuncNoBR calculated")

    # Clean up
    if arcpy.Exists('WU_HFuncNoBR1'):
        arcpy.Delete_management('WU_HFuncNoBR1')
```

5.7.26 HFunction: Habitat and Ecological Integrity Function

```
################################################################################
#
# File Name: HFunction.py
# Developer: Yibing Han
# Date: 10/19/2017
# Purpose:
#    Habitat & Ecological Integrity function
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../..")

import arcpy
from utilities import actions
import logging

def CalcHFunction():
    logger = logging.getLogger("WFA.HabEco.HFunction")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_BRank","fcBRank")
    arcpy.MakeFeatureLayer_management(r"WU_HFuncNoBR","fcHFuncNoBR")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_HFunction'):
        arcpy.Delete_management('WU_HFunction')
    ## Spatial Join to merge metrics and create feature class to store Function
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcBRank")
    fmSJ.addTable("fcHFuncNoBR")
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','HPotential','HOpportun','HSociety','HFuncNoBR','BRa
nk']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))


    arcpy.SpatialJoin_analysis("fcBRank", "fcHFuncNoBR", 'WU_HFunction',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    fcHFunction = arcpy.mapping.Layer(r"WU_HFunction")
    logger.info("spatial join BRank and HFuncNoBR completed")


    ## Add Function field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcHFunction, 'HPotB6')
    arcpy.AddField_management(fcHFunction, 'HPotB6', 'SHORT')
    actions.DeleteField(fcHFunction, 'HFunction')
    arcpy.AddField_management(fcHFunction, 'HFunction', 'SHORT')
    arcpy.CalculateField_management(fcHFunction, 'HFunction', '[HFuncNoBR]', 'VB', '#')
    logger.info("fields added to store intermediate value for B6 sites and results and initial values
set")


    ## Select B6 wetlands
    strWHERE = """"BRank" = 'B6'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION",
strWHERE)
```

```
    logger.info("B6 wetlands selected")

    ## Add 5 points to HPotential for B6 wetlands, up to a maximum of 30 points
    arcpy.CalculateField_management(fcHFunction, "HPotB6", "[HPotential] + 5", "VB", "#")
    strWHERE = """"HPotB6" > 30"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HPotB6", "30", "VB", "#")
    logger.info("5 points added to HPotential for B6 wetlands, up to a maximum of 30 points")

    ## Select B6 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B6'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "[HPotB6] + [HOpportun]+ 
[HSociety]", "VB", "#")

    ## Select B5 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B5'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "[HOpportun]+ [HSociety] + 
30", "VB", "#")

    ## Select B4 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B4'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "50", "VB", "#")

    ## Select B3 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B3'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "75", "VB", "#")


    ## Select B2 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B2'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "100", "VB", "#")


    ## Select B1 wetlands and calculate HFunction
    strWHERE = """"BRank" = 'B1'"""
    arcpy.SelectLayerByAttribute_management(fcHFunction, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcHFunction, "HFunction", "150", "VB", "#")


    # Clear all selections
    arcpy.SelectLayerByAttribute_management(fcHFunction, "CLEAR_SELECTION")


    # Clean up
```

5.7.27 HOpportun: Habitat and Ecological Integrity

```
##############################################################################
#
# File Name: HOpportun.py
# Developer: Yibing Han
# Date: 12/12/2017
# Purpose:
#    This script handles the execution of all the Habitat and Ecological Integrity Value to Society
metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import AquaAbund, BRankHUC, BufferContig, BufferPerim, ConsFocus,
LandInteg, WetldBird, Karst, WshdPos, WshdUniq
from Factors import BufferLand, LandHydro, LandEco
from Aspects import HOpportun

def procHOpportun(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun")

    ##############################################################
    ## 1. Run Variables
    ##############################################################
    AquaAbund.CalcAquaAbund(WetlandPoly)
    BRankHUC.CalcBRankHUC(WetlandPoly)
    BufferContig.CalcBufferContig(WetlandPoly)
    BufferPerim.CalcBufferPerim(WetlandPoly)
    ConsFocus.CalcConsFocus(WetlandPoly)
    LandInteg.CalcLandInteg(WetlandPoly)
    WetldBird.CalcWetldBird(WetlandPoly)
    Karst.CalcKarst(WetlandPoly)
    WshdPos.CalcWshdPos()
    WshdUniq.CalcWshdUniq()

    ##############################################################
    ## 2. Run Factors
    ##############################################################
    BufferLand.CalcBufferLand()
    LandHydro.CalcLandHydro()
    LandEco.CalcLandEco()

    ##############################################################
```

```
## 3. Run Aspect
#############################################################
HOpportun.CalcHOpportun()
```

5.7.28 HOpportun: Habitat and Ecological Integrity Aspects

```
##########################################################################
#
# File Name: HOpportun.py
# Developer: Yibing Han
# Date: 10/10/2017
# Purpose:
#    Habitat Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def CalcHOpportun():
    logger = logging.getLogger("WFA.HabEco.HOpportun.HOpportun")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_BufferLand","fcBufferLand")
    arcpy.MakeFeatureLayer_management(r"WU_LandHydro","fcLandHydro")
    arcpy.MakeFeatureLayer_management(r"WU_LandEco","fcLandEco")

    # Clean up if needed
    if arcpy.Exists("WU_HOpportun"):
        arcpy.Delete_management("WU_HOpportun")
    if arcpy.Exists("WU_HOpportun1"):
        arcpy.Delete_management("WU_HOpportun1")

    ## Spatial joins to bring together factor values
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcBufferLand")
    fmSJFLIN.addTable("fcLandHydro")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BufferLand','LandHydro']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcBufferLand', 'fcLandHydro', 'WU_HOpportun1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_HOpportun1", "fcHOpportun1")
    logger.info("Spatial Join BufferLand and LandHydro completed")
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcHOpportun1")
    fmSJFLIN.addTable("fcLandEco")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BufferLand','LandHydro','LandEco']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcHOpportun1', 'fcLandEco', 'WU_HOpportun',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcHOpportun = arcpy.mapping.Layer(r"WU_HOpportun")
    logger.info("Spatial Join HOpportun1 and LandEco completed")

    ## Add HOpportun field and set initial point value to zero
    actions.DeleteField(fcHOpportun, "HOpportun")
    arcpy.AddField_management(fcHOpportun, "HOpportun", "SHORT")
    arcpy.CalculateField_management(fcHOpportun,"HOpportun","0","VB","#")
    logger.info("field HOpportun added and initial value set to 0")

    ## Sum the factor points
    arcpy.CalculateField_management(fcHOpportun,"HOpportun","[BufferLand]+
[LandHydro]+ [LandEco]","VB","#")
    logger.info("Points calculated to wetland units")

    # Clean up
    if arcpy.Exists("WU_HOpportun1"):
        arcpy.Delete_management("WU_HOpportun1")
```

5.7.29 BufferLand: Habitat and Ecological Integrity Opportunity

```
##########################################################################
#
# File Name: BufferLand.py
# Developer: Yibing Han
# Date: 10/03/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Landscape Opportunity
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcBufferLand():
    logger = logging.getLogger("WFA.HabEco.HOpportun.BufferLand")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_BufferPerim", 'fcBufferPerim')
    arcpy.MakeFeatureLayer_management(r"WU_BufferContig", 'fcBufferContig')
    arcpy.MakeFeatureLayer_management(r"WU_LandInteg", 'fcLandInteg')

# Clean up if needed
    if arcpy.Exists('WU_BufferLand1'):
        arcpy.Delete_management('WU_BufferLand1')
    if arcpy.Exists('WU_BufferLand'):
        arcpy.Delete_management('WU_BufferLand')
    logger.info("feature layers ready")

    ## Spatial join to merge BufferPerim and BufferContig into one attribute table
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable('fcBufferPerim')
    fmSJ.addTable('fcBufferContig')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BufferPerim','BufferContig']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcBufferPerim', 'fcBufferContig', 'WU_BufferLand1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_BufferLand1', 'fcBufferLand1')
    logger.info("spatial join BufferPerim and BufferContig completed")

    ## Spatial join to merge LandInteg
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable('fcBufferLand1')
    fmSJ.addTable('fcLandInteg')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BufferPerim','BufferContig','LandInteg']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcBufferLand1', 'fcLandInteg', 'WU_BufferLand',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    fcBufferLand = arcpy.mapping.Layer(r"WU_BufferLand")
    logger.info("spatial join BufferLand1 and LandInteg completed")

    ## Add BufferLand field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcBufferLand, 'BufferLand')
    arcpy.AddField_management(fcBufferLand, 'BufferLand', 'SHORT')
    arcpy.CalculateField_management(fcBufferLand, 'BufferLand', '0', 'VB', '#')
    logger.info("field BufferLand added")

    ## Sum the points for VegVerStr, VegHorInt, VegFQ
    arcpy.CalculateField_management(fcBufferLand, 'BufferLand', '[BufferPerim] +
[BufferContig] + [LandInteg]', 'VB', '#')
    logger.info("field BufferLand calculated")

    # Clean up
    if arcpy.Exists('WU_BufferLand1'):
        arcpy.Delete_management('WU_BufferLand1')
```

5.7.30 LandEco: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: LandEco.py
# Developer: Yibing Han
# Date: 10/10/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcLandEco():
    logger = logging.getLogger("WFA.HabEco.HOpportun.HOpportun")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_BRankHUC","fcBRankHUC")
    arcpy.MakeFeatureLayer_management(r"WU_WshdUniq","fcWshdUniq")
    arcpy.MakeFeatureLayer_management(r"WU_ConsFocus","fcConsFocus")
    arcpy.MakeFeatureLayer_management(r"WU_WetlandBird","fcWetlandBird")

    # Clean up if needed
    if arcpy.Exists("WU_LandEco"):
        arcpy.Delete_management("WU_LandEco")
    if arcpy.Exists("WU_LandEco1"):
        arcpy.Delete_management("WU_LandEco1")
    if arcpy.Exists("WU_LandEco2"):
        arcpy.Delete_management("WU_LandEco2")

    ## Spatial Join to merge attributes into one table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcBRankHUC")
    fmSJFLIN.addTable("fcWshdUniq")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BRankHUC','WshdUniq']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcBRankHUC', 'fcWshdUniq', 'WU_LandEco1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_LandEco1", "fcLandEco1")
    logger.info("Spatial Join BRankHUC and WshdUniq completed")

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcLandEco1")
    fmSJFLIN.addTable("fcConsFocus")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','BRankHUC','WshdUniq','ConsFocus']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcLandEco1', 'fcConsFocus', 'WU_LandEco2', 
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_LandEco2", "fcLandEco2")
    logger.info("Spatial Join LandEco1 and ConsFocus completed")

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcLandEco2")
    fmSJFLIN.addTable("fcWetlandBird")
    keepers = []
    keepers = 
['WUKey','Shape_Length','Shape_Area','BRankHUC','WshdUniq','ConsFocus','WetldBird']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcLandEco2', 'fcWetlandBird', 'WU_LandEco', 
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcLandEco = arcpy.mapping.Layer(r"WU_LandEco")
    logger.info("Spatial Join LandEco1 and ConsFocus completed")

    ## Create feature class to store results for LandEco and set initial value to zero
    actions.DeleteField(fcLandEco, "LandEco")
    arcpy.AddField_management(fcLandEco, "LandEco", "SHORT")
    arcpy.CalculateField_management(fcLandEco,"LandEco","0","VB","#")
    logger.info("field LandEco added and initial value set to 0")

    ## Sum the metrics and assign points:
    strWHERE = """("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 1"""
    arcpy.SelectLayerByAttribute_management(fcLandEco, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcLandEco,"LandEco","1","VB","#")
    logger.info("1 point assigned to qualifying wetland units")

    strWHERE = """("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 4"""
    arcpy.SelectLayerByAttribute_management(fcLandEco, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcLandEco,"LandEco","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")

    strWHERE = """("ConsFocus" + "WetldBird" + "BRankHUC" + "WshdUniq") > 7"""
    arcpy.SelectLayerByAttribute_management(fcLandEco, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcLandEco,"LandEco","3","VB","#")
    logger.info("2 points assigned to qualifying wetland units")

    arcpy.SelectLayerByAttribute_management(fcLandEco, "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("WU_LandEco1"):
        arcpy.Delete_management("WU_LandEco1")
    if arcpy.Exists("WU_LandEco2"):
        arcpy.Delete_management("WU_LandEco2")
```

5.7.31 LandHydro: Habitat and Ecological Integrity Opportunity

```
##########################################################################
#
# File Name: LandHydro.py
# Developer: Yibing Han
# Date: 10/03/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Landscape Opportunity
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcLandHydro():
    logger = logging.getLogger("WFA.HabEco.HOpportun.LandHydro")

    # Setting python variables
    fcWshdPos = arcpy.mapping.Layer(r"WU_WshdPos")
    fcAquaAbund = arcpy.mapping.Layer(r"WU_AquaAbund")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_LandHydro'):
        arcpy.Delete_management('WU_LandHydro')

    ## Spatial join to merge WshdPos and AquaAbund into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable(fcWshdPos)
    fmSJFLIN.addTable(fcAquaAbund)
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','WshdPos','AquaAbund']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis(fcWshdPos, fcAquaAbund, 'WU_LandHydro',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcLandHydro = arcpy.mapping.Layer(r"WU_LandHydro")
    logger.info("spatial join WshdPos and AquaAbund completed")

    ## Add LandHydro field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcLandHydro, 'LandHydro')
    arcpy.AddField_management(fcLandHydro, 'LandHydro', 'SHORT')
    arcpy.CalculateField_management(fcLandHydro, 'LandHydro', '0', 'VB', '#')
    logger.info("field LandHydro added and initial value set to 0")

    ## Sum the points for WshdPos and AquaAbund
    arcpy.CalculateField_management(fcLandHydro, 'LandHydro', '[WshdPos] + [AquaAbund]',
'VB', '#')
    logger.info("field LandHydro calculated")

    # Clean up
```

5.7.32 AquaAbund: Habitat and Ecological Integrity Opportunity

```
################################################################################
#
# File Name: AquaAbund.py
# Developer: Yibing Han
# Date: 10/03/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandHydro: Landscape Hydrologic Connectivity)
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcAquaAbund(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.AquaAbund")

    # Setting python variables
    fcBuffer1km = arcpy.mapping.Layer(globalvars.srcBuffer1km)
    fcWVWetland = arcpy.mapping.Layer(globalvars.srcEnhWetland)
    fcStreams = arcpy.mapping.Layer(globalvars.srcStreams)
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('Buffer1kAqua'):
        arcpy.Delete_management('Buffer1kAqua')
    if arcpy.Exists('Buffer1kAqua_diss'):
        arcpy.Delete_management('Buffer1kAqua_diss')
    if arcpy.Exists('Buffer1kStrm'):
        arcpy.Delete_management('Buffer1kStrm')
    if arcpy.Exists('Buffer1kStrm_diss'):
        arcpy.Delete_management('Buffer1kStrm_diss')
    if arcpy.Exists('WU_AquaAbund1'):
        arcpy.Delete_management('WU_AquaAbund1')
    if arcpy.Exists('WU_AquaAbund2'):
        arcpy.Delete_management('WU_AquaAbund2')
    if arcpy.Exists('WU_AquaAbund'):
        arcpy.Delete_management('WU_AquaAbund')

    ## STEP 1 Calculate percent of 1 km buffer that contains aquatic features from NWI
    ## Create feature class to store intermediate results for AquaAbund
    arcpy.CopyFeatures_management(WetlandPoly,"WU_AquaAbund1","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WU_AquaAbund1", "fcWUAquaAbund1")
    logger.info("feature class WU_AquaAbund1 created")

    ## Intersect the 1 km buffers and the Enhanced National Wetlands Inventory.
    arInputData = [fcBuffer1km,fcWVWetland]
    arcpy.Intersect_analysis(arInputData,"Buffer1kAqua","ALL",output_type="INPUT")
    fcBuffer1kAqua = arcpy.mapping.Layer(r"Buffer1kAqua")
    logger.info("intersect completed")

    ## Dissolve aquatic portion of wetland buffer by WUKey
    arcpy.Dissolve_management(fcBuffer1kAqua,"Buffer1kAqua_diss","WUKey","Buf1kArea
FIRST")
```

```
    arcpy.MakeFeatureLayer_management("Buffer1kAqua_diss", "fcBuffer1kAqua_diss")
    logger.info("aquatic portion of wetland buffer dissolved")

    ## Add field and calculate ratio of aquatic area to total buffer area.
    actions.DeleteField("fcBuffer1kAqua_diss","Aqua1kRat")
    arcpy.AddField_management("fcBuffer1kAqua_diss", "Aqua1kRat", "FLOAT")

arcpy.CalculateField_management("fcBuffer1kAqua_diss","Aqua1kRat","[Shape_Area]/[FIRS
T_Buf1kArea]","VB","#")
    logger.info("added and calculated field of Aqua1kRat")

    ## Join ratio of distrubed land to Wetland Units

arcpy.AddJoin_management("fcWUAquaAbund1","WUKey","fcBuffer1kAqua_diss","WUKey
","KEEP_ALL")
    logger.info("added join to WU_AquaAbund1")

    ## Export joined data

#arcpy.CopyFeatures_management("fcWUAquaAbund1","WU_AquaAbund2","#","0","0","0")
    arcpy.FeatureClassToFeatureClass_conversion("fcWUAquaAbund1", arcpy.env.workspace,
"WU_AquaAbund2")
    arcpy.MakeFeatureLayer_management("WU_AquaAbund2", "fcWUAquaAbund2")
    logger.info("Joined data exported as WU_AquaAbund2")

    ## Remove Join
    arcpy.RemoveJoin_management("fcWUAquaAbund1")
    logger.info("Join removed")

    # Set value of Aqua1kRat to zero for null intersections
    strWHERE = """"Buffer1kAqua_diss_Aqua1kRat" IS NULL"""
```

```
arcpy.SelectLayerByAttribute_management("fcWUAquaAbund2","NEW_SELECTION",strW
HERE)

arcpy.CalculateField_management("fcWUAquaAbund2","Buffer1kAqua_diss_Aqua1kRat","0"
,"VB","#")
    logger.info("field Aqua1kRat value added for null intersections")
    arcpy.SelectLayerByAttribute_management("fcWUAquaAbund2","CLEAR_SELECTION")

    ## STEP 2 Calculate total length of NHD streams in 1 km buffer
    ## Intersect the 1 km buffers and the NHD stream reaches
    arInputData = [fcBuffer1km,fcStreams]
    arcpy.Intersect_analysis(arInputData,"Buffer1kStrm","ALL",output_type="INPUT")
    fcBuffer1kStrm = arcpy.mapping.Layer(r"Buffer1kStrm")
    logger.info("intersect completed")

    ## Add field to store Stream Reach Length
    actions.DeleteField(fcBuffer1kStrm,"StrmLength")
    arcpy.AddField_management(fcBuffer1kStrm, "StrmLength", "FLOAT")


arcpy.CalculateField_management(fcBuffer1kStrm,"StrmLength","[Shape_Length]","VB","#")
    logger.info("added and calculated field of StrmLength")

    ## Delete unnecessary fields in this large table to reduce processing time in the Dissolve
below
    fields = arcpy.ListFields(fcBuffer1kStrm)
    keepList = ['WUKey','StrmLength']
    fieldsDel = [f.name for f in arcpy.ListFields(fcBuffer1kStrm) if not(f.type in
["OID","Geometry"] or f.name in ["Shape_Length","Shape_Area"] or f.name in keepList)]
    arcpy.DeleteField_management(fcBuffer1kStrm,fieldsDel)
    logger.info("unnecessary fields in table deleted to reduce processing time below")

    ## Dissolve stream reach lengths in the wetland buffer by WUKey
    arcpy.Dissolve_management(fcBuffer1kStrm,"Buffer1kStrm_diss","WUKey","StrmLength
SUM")
    arcpy.MakeFeatureLayer_management("Buffer1kStrm_diss", "flBuffer1kStrm_diss")
    logger.info("stream reach lengths in the wetland buffer dissolved")

    ## Join sum of stream lengths to Wetland Units

arcpy.AddJoin_management("fcWUAquaAbund2","WU_AquaAbund1_WUKey","flBuffer1kS
trm_diss","WUKey","KEEP_ALL")
    logger.info("added join to WU_AquaAbund2")

    ## Export joined data

#arcpy.CopyFeatures_management("fcWUAquaAbund2","WU_AquaAbund","#","0","0","0")
    arcpy.FeatureClassToFeatureClass_conversion("fcWUAquaAbund2", arcpy.env.workspace,
"WU_AquaAbund")
    fcWUAquaAbund = arcpy.mapping.Layer(r"WU_AquaAbund")
    logger.info("Joined data exported as WU_AquaAbund")

    ## Remove Join
    arcpy.RemoveJoin_management("fcWUAquaAbund2")
    logger.info("Join removed")

    ## Set value of FINALCODE to zero for null intersections
    strWHERE = """"Buffer1kStrm_diss_SUM_StrmLength" IS NULL"""

arcpy.SelectLayerByAttribute_management(fcWUAquaAbund,"NEW_SELECTION",strWHE
RE)
```

```
arcpy.CalculateField_management(fcWUAquaAbund,"Buffer1kStrm_diss_SUM_StrmLength",
"0","VB","#")
    logger.info("field SUM_StrmLength value added for null intersections")

    ## STEP 3: Assign points
    ## Add field AquaAbund, set initial value to zero
    arcpy.SelectLayerByAttribute_management(fcWUAquaAbund, "CLEAR_SELECTION")
    actions.DeleteField(fcWUAquaAbund, "AquaAbund")
    arcpy.AddField_management(fcWUAquaAbund, "AquaAbund", "SHORT")
    arcpy.CalculateField_management(fcWUAquaAbund,"AquaAbund","0","VB","#")
    logger.info("field AquaAbund added")

    ## Assign points
    ## StrmLength one SD below mean = 0, within 1 SD of mean = 1, >1 SD above mean = 2
points
    strWHERE = """"WU_AquaAbund2_Buffer1kAqua_diss_Aqua1kRat" > 0.01 OR
"Buffer1kStrm_diss_SUM_StrmLength" > 6000"""

arcpy.SelectLayerByAttribute_management(fcWUAquaAbund,"NEW_SELECTION",strWHE
RE)
    arcpy.CalculateField_management(fcWUAquaAbund,"AquaAbund","1","VB","#")

    strWHERE = """"WU_AquaAbund2_Buffer1kAqua_diss_Aqua1kRat" > 0.05 OR
"Buffer1kStrm_diss_SUM_StrmLength" > 8000"""

arcpy.SelectLayerByAttribute_management(fcWUAquaAbund,"NEW_SELECTION",strWHE
RE)
    arcpy.CalculateField_management(fcWUAquaAbund,"AquaAbund","2","VB","#")
    logger.info("AquaAbund points assigned")
    arcpy.SelectLayerByAttribute_management(fcWUAquaAbund,"CLEAR_SELECTION")
```

```
# Clean up
if arcpy.Exists('Buffer1kAqua'):
    arcpy.Delete_management('Buffer1kAqua')
if arcpy.Exists('Buffer1kAqua_diss'):
    arcpy.Delete_management('Buffer1kAqua_diss')
if arcpy.Exists('Buffer1kStrm'):
    arcpy.Delete_management('Buffer1kStrm')
if arcpy.Exists('Buffer1kStrm_diss'):
    arcpy.Delete_management('Buffer1kStrm_diss')
if arcpy.Exists('WU_AquaAbund1'):
    arcpy.Delete_management('WU_AquaAbund1')
if arcpy.Exists('WU_AquaAbund2'):
    arcpy.Delete_management('WU_AquaAbund2')
```

5.7.33 BRankHUC: Habitat and Ecological Integrity Opportunity

```
##########################################################################
#
# File Name: BRankHUC.py
# Developer: Yibing Han
# Date: 10/09/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcBRankHUC(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.BRankHUC")

    # Setting python variables
    fcBRHUC = arcpy.mapping.Layer(globalvars.srcBRHUC)

    # Clean up if needed
    if arcpy.Exists("WU_BRankHUC"):
        arcpy.Delete_management("WU_BRankHUC")
```

```
## Create feature class to store results for BRankHUC and set initial value to zero
arcpy.CopyFeatures_management(WetlandPoly,"WU_BRankHUC","#","0","0","0")
fcBRankHUC = arcpy.mapping.Layer(r"WU_BRankHUC")
logger.info("feature class WU_BRankHUC created")

actions.DeleteField(fcBRankHUC, "BRankHUC")
arcpy.AddField_management(fcBRankHUC, "BRankHUC", "SHORT")
arcpy.CalculateField_management(fcBRankHUC,"BRankHUC","0","VB","#")
logger.info("field BRankHUC added and initial value set to 0")

## Select B4- or B5 ranked watersheds
strWHERE = """"Brank" = 'B4' OR "Brank" = 'B5'"""
arcpy.SelectLayerByAttribute_management(fcBRHUC, "NEW_SELECTION", strWHERE)
logger.info("B4- or B5 ranked watersheds selected")

## Select Wetland Units that intersect B4- or B5-ranked watershed
arcpy.SelectLayerByLocation_management(fcBRankHUC, "INTERSECT", fcBRHUC)
logger.info("Wetland Units that intersect B4- or B5-ranked watershed selected")

## Assign 1 point
arcpy.CalculateField_management(fcBRankHUC,"BRankHUC","1","VB","#")
logger.info("1 point assigned to qualifying wetland units")

## Select B3-ranked watersheds
strWHERE = """"Brank" = 'B3'"""
arcpy.SelectLayerByAttribute_management(fcBRHUC, "NEW_SELECTION", strWHERE)
logger.info("B3-ranked watersheds selected")

## Select Wetland Units that intersect B3-ranked watershed
```

```
arcpy.SelectLayerByLocation_management(fcBRankHUC, "INTERSECT", fcBRHUC)
logger.info("Wetland Units that intersect B2-ranked watershed selected")

## Assign 2 points
arcpy.CalculateField_management(fcBRankHUC,"BRankHUC","2","VB","#")
logger.info("2 points assigned to qualifying wetland units")

## Select B2-ranked watersheds
strWHERE = """"Brank" = 'B2'"""
arcpy.SelectLayerByAttribute_management(fcBRHUC, "NEW_SELECTION", strWHERE)
logger.info("B2-ranked watersheds selected")

## Select Wetland Units that intersect B2-ranked watershed
arcpy.SelectLayerByLocation_management(fcBRankHUC, "INTERSECT", fcBRHUC)
logger.info("Wetland Units that intersect B2-ranked watershed selected")

## Assign 3 points and clear selections
arcpy.CalculateField_management(fcBRankHUC,"BRankHUC","3","VB","#")
logger.info("3 points assigned to qualifying wetland units")

 ## Select B2-ranked watersheds
strWHERE = """"Brank" = 'B1'"""
arcpy.SelectLayerByAttribute_management(fcBRHUC, "NEW_SELECTION", strWHERE)
logger.info("B2-ranked watersheds selected")

## Select Wetland Units that intersect B1-ranked watershed
arcpy.SelectLayerByLocation_management(fcBRankHUC, "INTERSECT", fcBRHUC)
logger.info("Wetland Units that intersect B1-ranked watershed selected")
```

```
## Assign 4 points and clear selections
arcpy.CalculateField_management(fcBRankHUC,"BRankHUC","4","VB","#")
logger.info("4 points assigned to qualifying wetland units")

arcpy.SelectLayerByAttribute_management(fcBRHUC, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcBRankHUC, "CLEAR_SELECTION")

# Clean up
```

5.7.34 BufferContig: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: BufferContig.py
# Developer: Yibing Han
# Date: 10/02/2017
# Purpose:
#    Input to Habitat / Opportunity / BufferLand (Buffer and Landscape Integrity)
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcBufferContig(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.BufferContig")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    fcBuffer300m = arcpy.mapping.Layer(globalvars.srcBuffer300m)
    fcDisturbedLand = arcpy.arcpy.mapping.Layer(globalvars.srcDisturbedLand)

    # Clean up if needed
    if arcpy.Exists("Buffer300mUndist"):
        arcpy.Delete_management("Buffer300mUndist")
    if arcpy.Exists("Buffer300mUndist_sing"):
        arcpy.Delete_management("Buffer300mUndist_sing")
    if arcpy.Exists("Buffer300mUndist_diss"):
        arcpy.Delete_management("Buffer300mUndist_diss")
    if arcpy.Exists("WU_BufferContig1"):
        arcpy.Delete_management("WU_BufferContig1")
    if arcpy.Exists("WU_BufferContig"):
        arcpy.Delete_management("WU_BufferContig")

    ## Erase the portions of the 300m buffer that overlap DisturbedLand
    arcpy.Erase_analysis(fcBuffer300m, fcDisturbedLand, "Buffer300mUndist")
    fcBuffer300mUndist = arcpy.mapping.Layer(r"Buffer300mUndist")
    logger.info("Feature Layers ready")

    ## The Erase tool produces multipart polygons.  Change these to singlepart polygons.
    arcpy.MultipartToSinglepart_management(fcBuffer300mUndist, "Buffer300mUndist_sing")
    fcBuffer300mUndist_sing = arcpy.mapping.Layer("Buffer300mUndist_sing")
    actions.DeleteField(fcBuffer300mUndist_sing,"ContigSingArea")
    arcpy.AddField_management(fcBuffer300mUndist_sing, "ContigSingArea", "FLOAT")
    logger.info("Field ContigSingArea added")

    ## Select undisturbed buffer polygons that share a line segment with Wetland Units.
    arcpy.SelectLayerByLocation_management(fcBuffer300mUndist_sing,
"SHARE_A_LINE_SEGMENT_WITH", "fcWU")
    logger.info("undisturbed buffer polygons selected by location")

    ## Calculate area of contiguous singlepart polygons
```

```
arcpy.CalculateField_management(fcBuffer300mUndist_sing,"ContigSingArea","[Shape_Area]
","VB","#")
    arcpy.SelectLayerByAttribute_management(fcBuffer300mUndist_sing,
"CLEAR_SELECTION")
    strWHERE = """"ContigSingArea" IS NULL"""
    arcpy.SelectLayerByAttribute_management(fcBuffer300mUndist_sing,
"NEW_SELECTION", strWHERE)

arcpy.CalculateField_management(fcBuffer300mUndist_sing,"ContigSingArea","0","VB","#")
    arcpy.SelectLayerByAttribute_management(fcBuffer300mUndist_sing,
"CLEAR_SELECTION")
    logger.info("area of contiguous singlepart polygons calculated")


    ## Dissolve undisturbed portion of wetland buffer by WUKey
    arcpy.Dissolve_management(fcBuffer300mUndist_sing, "Buffer300mUndist_diss",
"WUKey", "Buf300Area FIRST;ContigSingArea SUM")
    arcpy.MakeFeatureLayer_management("Buffer300mUndist_diss",
"fcBuffer300mUndist_diss")
    logger.info("undisturbed portion of wetland buffer dissolved")


    ## Add field and calculate ratio of contiguous undisturbed area to total buffer area.
    actions.DeleteField("fcBuffer300mUndist_diss","ContigUndRat")
    arcpy.AddField_management("fcBuffer300mUndist_diss", "ContigUndRat", "FLOAT")

arcpy.CalculateField_management("fcBuffer300mUndist_diss","ContigUndRat","[SUM_Conti
gSingArea]/[FIRST_Buf300Area]","VB","#")
    logger.info("field ContigUndRat added and calculated")


    ## Create feature class to store intermediate results for BufferContig
    arcpy.CopyFeatures_management("fcWU","WU_BufferContig1","#","0","0","0")
    logger.info("feature class WU_BufferContig1 created")
    ## Join ratio of contiguous undisturbed buffer to Wetland Units
    arcpy.MakeFeatureLayer_management("WU_BufferContig1", "fcWUBufferContig1")

arcpy.AddJoin_management("fcWUBufferContig1","WUKey","fcBuffer300mUndist_diss","W
UKey","KEEP_ALL")
    logger.info("join added to Wetland Units")

    ## Export joined data

#arcpy.CopyFeatures_management("fcWUBufferContig1","WU_BufferContig","#","0","0","0"
)
    arcpy.FeatureClassToFeatureClass_conversion("fcWUBufferContig1", arcpy.env.workspace,
"WU_BufferContig")
    fcBufferContig = arcpy.mapping.Layer("WU_BufferContig")
    logger.info("joined data exported")

    ## Remove Join
    arcpy.RemoveJoin_management("fcWUBufferContig1")
    logger.info("joined removed")

    ## Set value of ContigUndRat to zero for null intersections
    strWHERE = """"Buffer300mUndist_diss_ContigUndRat" IS NULL"""
    arcpy.SelectLayerByAttribute_management(fcBufferContig, "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management(fcBufferContig,"Buffer300mUndist_diss_ContigUndRat","0
","VB","#")
    arcpy.SelectLayerByAttribute_management(fcBufferContig, "CLEAR_SELECTION")
    logger.info("field ContigUndRat value set to 0 for null intersections")

    ## Add field BufferContig, set initial value to zero
    actions.DeleteField(fcBufferContig,"BufferContig")
    arcpy.AddField_management(fcBufferContig, "BufferContig", "SHORT")
    arcpy.CalculateField_management(fcBufferContig,"BufferContig","0","VB","#")
    logger.info("field BufferContig added")

    ## Assign points to Wetland Units
    strWHERE = """"Buffer300mUndist_diss_ContigUndRat" > 0.6"""
    arcpy.SelectLayerByAttribute_management(fcBufferContig, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcBufferContig,"BufferContig","1","VB","#")

    strWHERE = """"Buffer300mUndist_diss_ContigUndRat" > 0.9"""
    arcpy.SelectLayerByAttribute_management(fcBufferContig, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcBufferContig,"BufferContig","2","VB","#")
    logger.info("field BufferContig values calculated")
    arcpy.SelectLayerByAttribute_management(fcBufferContig, "CLEAR_SELECTION")

    arcpy.AddField_management(fcBufferContig, "WUKey", "LONG")

arcpy.CalculateField_management(fcBufferContig,"WUKey","[WU_BufferContig1_WUKey]",
"VB","#")
    arcpy.DeleteField_management(fcBufferContig,"WU_BufferContig1_WUKey")

    # Clean up
    if arcpy.Exists("Buffer300mUndist"):
        arcpy.Delete_management("Buffer300mUndist")
    if arcpy.Exists("Buffer300mUndist_sing"):
        arcpy.Delete_management("Buffer300mUndist_sing")
    if arcpy.Exists("Buffer300mUndist_diss"):
        arcpy.Delete_management("Buffer300mUndist_diss")
```

```
if arcpy.Exists("WU_BufferContig1"):
    arcpy.Delete_management("WU_BufferContig1")
```

5.7.35 BufferPerim: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: BufferPerim.py
# Developer: Yibing Han
# Date: 10/03/2017
# Purpose:
#    Input to Habitat / Opportunity / BufferLand (Buffer and Landscape Integrity)
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcBufferPerim(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.BufferPerim")

    # Setting python variables
    fcBuffer10m = arcpy.mapping.Layer(globalvars.srcBuffer10m)
    fcDisturbedLand = arcpy.mapping.Layer(globalvars.srcDisturbedLand)
    fcInterstates = arcpy.mapping.Layer(globalvars.srcInterstates)
    fcPrimaryRoads = arcpy.mapping.Layer(globalvars.srcPrimaryRoads)
    fcLocalRoads = arcpy.mapping.Layer(globalvars.srcLocalRoads)
    fcOthers = arcpy.mapping.Layer(globalvars.srcOtherRoads)
    fcRailway = arcpy.mapping.Layer(globalvars.srcRailway)
    fcTrails = arcpy.mapping.Layer(globalvars.srcTrails)
    logger.info("Feature Layers ready")

    # Clean up if needed
    if arcpy.Exists("WU_BufferPerim1"):
        arcpy.Delete_management("WU_BufferPerim1")
    if arcpy.Exists("WU_BufferPerim"):
        arcpy.Delete_management("WU_BufferPerim")
    if arcpy.Exists("Buffer10mDist"):
        arcpy.Delete_management("Buffer10mDist")

    ## STEP 1 Roads and Railways
    ## Create feature class to store intermediate results for BufferPerim
    arcpy.CopyFeatures_management(WetlandPoly,"WU_BufferPerim1","#","0","0","0")
    fcBufferPerim1 = arcpy.mapping.Layer(r"WU_BufferPerim1")

    ## Add field to store road and rail type
    actions.DeleteField(fcBufferPerim1,"RoadRailType")
    arcpy.AddField_management(fcBufferPerim1, "RoadRailType", "TEXT","","","10")
    logger.info("field RoadRailType added to WU_BufferPerim1")

    ## Select Wetland Units that intersect or are within 10 meters of mapped trails
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT", fcTrails, "10
Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Trail'","VB","#")
    logger.info("field RoadRailType calculated with type 'Trail'")

    ## Select Wetland Units that intersect or are within 10 meters of other roads & trails
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT", fcOthers, "10
Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Other'","VB","#")
    logger.info("field RoadRailType calculated with type 'Other'")

    ## Select Wetland Units that intersect or are within 10 meters of local roads
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT", fcLocalRoads,
"10 Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Local'","VB","#")
    logger.info("field RoadRailType calculated with type 'Local'")

    ## Select Wetland Units that intersect or are within 10 meters of railways
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT", fcRailway, "10
Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Rail'","VB","#")
    logger.info("field RoadRailType calculated with type 'Rail'")

    ## Select Wetland Units that intersect or are within 10 meters of primary roads
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT",
fcPrimaryRoads, "10 Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Primary'","VB","#")
    logger.info("field RoadRailType calculated with type 'Primary'")

    ## Select Wetland Units that intersect or are within 10 meters of interstate highways
    arcpy.SelectLayerByLocation_management(fcBufferPerim1, "INTERSECT", fcInterstates,
"10 Meters")
    arcpy.CalculateField_management(fcBufferPerim1,"RoadRailType","'Interstate'","VB","#")
    logger.info("field RoadRailType calculated with type 'Interstate'")
    arcpy.SelectLayerByAttribute_management(fcBufferPerim1, "CLEAR_SELECTION")

    ## STEP 2 DisturbedLand
    ## Note that much of this step is the same as the method for Disturb50m

    # Intersect the 10m buffers and the disturbed land uses
    arInputData = [fcBuffer10m,fcDisturbedLand]
    arcpy.Intersect_analysis(arInputData,"Buffer10mDist","ALL",output_type="INPUT")
    fcBuffer10mDist = arcpy.mapping.Layer(r"Buffer10mDist")
    logger.info("intersect completed")

    # Dissolve disturbed portion of wetland buffer by WUKey
    arcpy.Dissolve_management(fcBuffer10mDist,"Buffer10mDist_diss","WUKey","Buf10Area
FIRST")
    fcBuffer10mDist_diss = arcpy.mapping.Layer(r"Buffer10mDist_diss")
    logger.info("disturbed portion of wetland buffer dissolved")

    ## Add field and calculate ratio of disturbed area to total buffer area.
    actions.DeleteField(fcBuffer10mDist_diss,"Dist10mRat")
    arcpy.AddField_management(fcBuffer10mDist_diss, "Dist10mRat", "FLOAT")

arcpy.CalculateField_management(fcBuffer10mDist_diss,"Dist10mRat","[Shape_Area]/[FIRST
_Buf10Area]","VB","#")
    logger.info("added and calculated field of Dist10mRat")

    # Join ratio of distrubed land to Wetland Units
    arcpy.MakeFeatureLayer_management("WU_BufferPerim1", "flWUBufferPerim1")
    arcpy.MakeFeatureLayer_management("Buffer10mDist_diss", "flBuffer10mDist_diss")

arcpy.AddJoin_management("flWUBufferPerim1","WUKey","flBuffer10mDist_diss","WUKey
","KEEP_ALL")
    logger.info("added join to WU_BufferPerim1")

    # Export joined data
```

```
#arcpy.CopyFeatures_management("flWUBufferPerim1","WU_BufferPerim","#","0","0","0")
    arcpy.FeatureClassToFeatureClass_conversion("flWUBufferPerim1", arcpy.env.workspace,
"WU_BufferPerim")
    fcBufferPerim = arcpy.mapping.Layer(r"WU_BufferPerim")
    logger.info("Joined data exported as WU_BufferPerim")

    # Remove Join
    arcpy.RemoveJoin_management("flWUBufferPerim1")
    logger.info("Join removed")

    # Set value of Dist10mRat to zero for null intersections
    strWHERE = """"Buffer10mDist_diss_Dist10mRat" IS NULL"""

arcpy.SelectLayerByAttribute_management(fcBufferPerim,"NEW_SELECTION",strWHERE)

arcpy.CalculateField_management(fcBufferPerim,"Buffer10mDist_diss_Dist10mRat","0","VB"
,"#")
    logger.info("field Dist10mRat value added for null intersections")

    ## STEP 3
    ## Assign points

    # Add field BufferPerim, set initial value to zero
    arcpy.SelectLayerByAttribute_management(fcBufferPerim, "CLEAR_SELECTION")
    actions.DeleteField(fcBufferPerim, "BufferPerim")
    arcpy.AddField_management(fcBufferPerim, "BufferPerim", "FLOAT")
    arcpy.CalculateField_management(fcBufferPerim,"BufferPerim","0","VB","#")
    logger.info("field BufferPerim added")

    # Assign points
    strWHERE = """"Buffer10mDist_diss_Dist10mRat" = 0 AND
"WU_BufferPerim1_RoadRailType" IS NULL"""
    arcpy.SelectLayerByAttribute_management(fcBufferPerim, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcBufferPerim,"BufferPerim","2","VB","#")

    strWHERE = """"Buffer10mDist_diss_Dist10mRat" > 0 OR
"WU_BufferPerim1_RoadRailType" IN ('Trail', 'Local', 'Other')"""
    arcpy.SelectLayerByAttribute_management(fcBufferPerim, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcBufferPerim,"BufferPerim","1","VB","#")

    strWHERE = """"Buffer10mDist_diss_Dist10mRat" > 0.25 OR
"WU_BufferPerim1_RoadRailType" IN ('Rail', 'Primary', 'Interstate')"""
    arcpy.SelectLayerByAttribute_management(fcBufferPerim, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcBufferPerim,"BufferPerim","0","VB","#")
    logger.info("calculation completed for field BufferPerim")
    arcpy.SelectLayerByAttribute_management(fcBufferPerim, "CLEAR_SELECTION")

    arcpy.AddField_management(fcBufferPerim, "WUKey", "LONG")

arcpy.CalculateField_management(fcBufferPerim,"WUKey","[WU_BufferPerim1_WUKey]","
VB","#")
    arcpy.DeleteField_management(fcBufferPerim,"WU_BufferPerim1_WUKey")

    # Clean up
    if arcpy.Exists("WU_BufferPerim1"):
        arcpy.Delete_management("WU_BufferPerim1")
    if arcpy.Exists("Buffer10mDist"):
        arcpy.Delete_management("Buffer10mDist")
```

5.7.36 ConsFocus: Habitat and Ecological Integrity Opportunity

```
##########################################################################
#
# File Name: ConsFocus.py
# Developer: Yibing Han
# Date: 10/09/2017 (updated 12/27/2018)
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcConsFocus(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.ConsFocus")

    # Setting python variables
    fcCFA = arcpy.mapping.Layer(globalvars.srcCFArea)

    # Clean up if needed
    if arcpy.Exists("WU_ConsFocus"):
        arcpy.Delete_management("WU_ConsFocus")
```

```
## Create feature class to store results for ConsFocus and set initial value to zero
arcpy.CopyFeatures_management(WetlandPoly,"WU_ConsFocus","#","0","0","0")
fcConstFocus = arcpy.mapping.Layer(r"WU_ConsFocus")
logger.info("feature class WU_ConsFocus created")

actions.DeleteField(fcConstFocus, "ConsFocus")
arcpy.AddField_management(fcConstFocus, "ConsFocus", "SHORT")
arcpy.CalculateField_management(fcConstFocus,"ConsFocus","0","PYTHON","#")
logger.info("field ConsFocus added and initial value set to 0")

'''## Select all CFAs except the general CFA
strWHERE = """"CFA_Name" <> 'General Conservation Area'"""
arcpy.SelectLayerByAttribute_management(fcCFA, "NEW_SELECTION", strWHERE)
logger.info("all CFAs except the general CFA selected")'''

## Select Wetland Units that intersect the selected CFAs
arcpy.SelectLayerByLocation_management(fcConstFocus, "INTERSECT", fcCFA)
logger.info("Wetland Units that intersect the selected CFAs selected")

## Assign 1 point
arcpy.CalculateField_management(fcConstFocus,"ConsFocus","1","PYTHON","#")
logger.info("1 point assigned to qualifying wetland units")

## Select CFAs with wetland focus
strWHERE = """"WetlFocus" = 'yes'"""
arcpy.SelectLayerByAttribute_management(fcCFA, "NEW_SELECTION", strWHERE)
logger.info("CFAs with wetland focus selected")

## Select Wetland Units that intersect CFAs with wetland focus
```

```
arcpy.SelectLayerByLocation_management(fcConstFocus, "INTERSECT", fcCFA)
logger.info("Wetland Units that intersect CFAs with wetland focus selected")

## Assign 2 points
arcpy.CalculateField_management(fcConstFocus,"ConsFocus","2","PYTHON","#")
logger.info("2 points assigned to qualifying wetland units")

arcpy.SelectLayerByAttribute_management(fcCFA, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcConstFocus, "CLEAR_SELECTION")

# Clean up
```

5.7.37 Karst: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: Karst.py
# Developer: Yibing Han
# Date 9/20/2017
# Purpose:
#    Input to Habitat / Potential / Vegetation / Floristic Quality
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcKarst(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HPotential.Karst")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcKarstComp,"fcKarstComposite")

    # Clean up if needed
    if arcpy.Exists("WU_Karst"):
        arcpy.Delete_management("WU_Karst")
    if arcpy.Exists("WU_Karst1"):
        arcpy.Delete_management("WU_Karst1")
    logger.info("feature layers ready")

    ## Intersect karst and Wetland Units
    arcpy.Intersect_analysis(["fcKarstComposite", "fcWU"], "WU_Karst1", "ONLY_FID")
    logger.info("Intersect of karst and Wetland Units completed")

    ## Add field to store karst area.
    actions.DeleteField("WU_Karst1","KarstArea")
    arcpy.AddField_management("WU_Karst1", "KarstArea", "FLOAT")
    arcpy.CalculateField_management("WU_Karst1","KarstArea","[Shape_Area]","VB","#")
    arcpy.MakeFeatureLayer_management(r"WU_Karst1","fcWUKarst1")
    logger.info("field KarstArea added to store karst area")

    ## Spatial Join karst selection to Wetland Units and sum karst area.
##################################################################################
#################
    # SJ: FloodIn
##################################################################################
#################
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcWU")
    fmSJFLIN.addTable("fcWUKarst1")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","KarstArea"]

    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJFLIN.findFieldMapIndex("KarstArea")
    fieldmap = fmSJFLIN.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJFLIN.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcWUKarst1","WU_Karst","JOIN_ONE_TO_ONE","KEE
P_ALL",fmSJFLIN,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_Karst","fcWUKarst")
    logger.info("Spatial Join karst selection to Wetland Units completed")

    ## Add fields to store KarstRatio and Karst.
    actions.DeleteField("fcWUKarst","KarstRatio")
    arcpy.AddField_management("fcWUKarst", "KarstRatio", "FLOAT")
    arcpy.AddField_management("fcWUKarst", "Karst", "SHORT")
    arcpy.CalculateField_management("fcWUKarst","Karst","0","VB","#")

arcpy.CalculateField_management("fcWUKarst","KarstRatio","[KarstArea]/[Shape_Area]","V
B","#")
    logger.info("field KarstRatio added to store KarstRatio and Karst")

    ## Assign points to Wetland Units for VegHorInt
    strWHERE = """"KarstRatio" > 0.1"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","1","VB","#")

    strWHERE = """"KarstRatio" > 0.33"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","2","VB","#")


    strWHERE = """"KarstRatio" > 0.67"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","3","VB","#")
    logger.info("points assigned to Wetland Units for field KarstRatio")


    arcpy.SelectLayerByAttribute_management("fcWUKarst", "CLEAR_SELECTION")


    # Clean up
    if arcpy.Exists("WU_Karst1"):
        arcpy.Delete_management("WU_Karst1")
```

5.7.38 LandInteg: Habitat and Ecological Integrity Opportunity

```
############################################################################
#
# File Name: LandInteg.py
# Developer: Yibing Han
# Date 9/21/2017 (updated 12/27/2018)
# Purpose:
#    Input to Habitat / Potential (VegFQ, HydIntact) and Habitat/Opportunity (Buffer and
Landscape Integrity)
#
############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions
from arcpy.sa import *

def CalcLandInteg(WetlandPoly):
    arcpy.CheckOutExtension("Spatial")
    logger = logging.getLogger("WFA.HabEco.HOpportun.LandInteg")

    # Clean up if needed
    if arcpy.Exists("LandInteg_zonal"):
        arcpy.Delete_management("LandInteg_zonal")
    if arcpy.Exists("LandResil_zonal"):
        arcpy.Delete_management("LandResil_zonal")
```

```
if arcpy.Exists("WU_LandIntegDNR"):
    arcpy.Delete_management("WU_LandIntegDNR")
if arcpy.Exists("WU_IEI1"):
    arcpy.Delete_management("WU_IEI1")
if arcpy.Exists("WU_IEI"):
    arcpy.Delete_management("WU_IEI")
if arcpy.Exists("WUpoint_IEI"):
    arcpy.Delete_management("WUpoint_IEI")
if arcpy.Exists("WU_LandResil1"):
    arcpy.Delete_management("WU_LandResil1")
if arcpy.Exists("WU_LandResil"):
    arcpy.Delete_management("WU_LandResil")
if arcpy.Exists("WUpoint_LandResil"):
    arcpy.Delete_management("WUpoint_LandResil")
if arcpy.Exists("WU_ForestPatch"):
    arcpy.Delete_management("WU_ForestPatch")
if arcpy.Exists("WU_LandInteg"):
    arcpy.Delete_management("WU_LandInteg")
if arcpy.Exists("WU_LandInteg0"):
    arcpy.Delete_management("WU_LandInteg0")
if arcpy.Exists("WU_LandInteg1"):
    arcpy.Delete_management("WU_LandInteg1")
if arcpy.Exists("WUpoint_LandInteg"):
    arcpy.Delete_management("WUpoint_LandInteg")

# Setting python variables
arcpy.CopyFeatures_management(WetlandPoly,"WU_LandInteg0","#","0","0","0")
arcpy.MakeFeatureLayer_management("WU_LandInteg0", "fcWULandInteg0")
arcpy.MakeFeatureLayer_management(globalvars.srcForestPatches, "fcForestPatches")
```

```
    arcpy.MakeRasterLayer_management(globalvars.srcIEI, "IEI")
    arcpy.MakeRasterLayer_management(globalvars.srcLIIndex, "LIIndex")
    arcpy.MakeRasterLayer_management(globalvars.srcResilientConnected, "ResilConnected")
    logger.info("feature & raster layers ready")

    ## Calculate the intermediate metric LandIntegDNR
    ## Summarize the DNR Landscape Integrity raster values for each Wetland Unit
    arcpy.gp.ZonalStatisticsAsTable("fcWULandInteg0", "WUKey", "LIIndex",
"LandInteg_zonal", "DATA", "MEAN")
    logger.info("the DNR Landscape Integrity raster values for each Wetland Unit summarized")

    ## Convert Wetland Unit polygons to points
    arcpy.MakeFeatureLayer_management(globalvars.srcWUPoint, "fcWUpoint")
    logger.info("Wetland Unit polygons converted to points")

    ## Extract the DNR Landscape Integrity raster values for each Wetland Unit centroid
    ExtractValuesToPoints("fcWUpoint", "LIIndex", "WUpoint_LandInteg")
    logger.info("DNR Landscape Integrity raster values for each Wetland Unit centroid
extracted")

    ## Delete the "Count" field from the LandInteg_zonal table since "Count" is a restricted word
    ## and may interfere with the join in the next step
    arcpy.DeleteField_management ("LandInteg_zonal","Count")
    logger.info("field Count deleted")

    ## Join Wetland Units to the LandInteg_zonal table
    arcpy.AddJoin_management("fcWULandInteg0","WUKey","LandInteg_zonal","WUKey")
    logger.info("Join Wetland Units to the LandInteg_zonal table completed")

    ## Export the joined data to a Feature Class
    arcpy.FeatureClassToFeatureClass_conversion("fcWULandInteg0", arcpy.env.workspace,
"WU_LandInteg1")
    logger.info("joined data exported as WU_LandInteg1")

    ## Remove Join from Wetland Units
    arcpy.RemoveJoin_management("fcWULandInteg0")
    logger.info("join removed from Wetland Units")

    ## Join WU_LandInteg1 to WUpoint_LandInteg
    arcpy.MakeFeatureLayer_management("WU_LandInteg1", "fcWULandInteg1")
    arcpy.MakeFeatureLayer_management("WUpoint_LandInteg", "fcWUPoint_LandInteg")

arcpy.AddJoin_management("fcWULandInteg1","WU_LandInteg0_WUKey","fcWUPoint_La
ndInteg","WUKey")
    logger.info("joined WU_LandInteg1 to WUpoint_LandInteg")

    ## Export joined data to feature class
    arcpy.FeatureClassToFeatureClass_conversion("fcWULandInteg1", arcpy.env.workspace,
"WU_LandIntegDNR")
    arcpy.MakeFeatureLayer_management("WU_LandIntegDNR", "fcWULandIntegDNR")
    logger.info("joined data exported as WU_LandIntegDNR")

    ## Replace NULL values of MEAN with centroid values of Landscape Integrity
    strWHERE = """"WU_LandInteg1_LandInteg_zonal_MEAN" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWULandIntegDNR","WU_LandInteg1_LandInteg_zonal
_MEAN","!WUpoint_LandInteg_RASTERVALU!","PYTHON","#")
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR",
"CLEAR_SELECTION")
    logger.info("NULL values of MEAN with centroid values of Landscape Integrity replaced")
    ## Add field LandIntegDNR, set initial value to zero
    actions.DeleteField("fcWULandIntegDNR","LandIntegDNR")
    arcpy.AddField_management("fcWULandIntegDNR", "LandIntegDNR", "SHORT")

arcpy.CalculateField_management("fcWULandIntegDNR","LandIntegDNR","0","PYTHON","
#")
    logger.info("field LandIntegDNR added and initial value set to zero")


    ## Assign points
    strWHERE = """"WU_LandInteg1_LandInteg_zonal_MEAN" > 600"""
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWULandIntegDNR","LandIntegDNR","1","PYTHON","
#")


    strWHERE = """"WU_LandInteg1_LandInteg_zonal_MEAN" > 700"""
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWULandIntegDNR","LandIntegDNR","2","PYTHON","
#")


    strWHERE = """"WU_LandInteg1_LandInteg_zonal_MEAN" > 800"""
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWULandIntegDNR","LandIntegDNR","3","PYTHON","
#")
    logger.info("points assigned in field LandIntegDNR")
    logger.info("Step 1 completed")
    arcpy.SelectLayerByAttribute_management("fcWULandIntegDNR",
"CLEAR_SELECTION")


    ## STEP 2
    ## Note that method in nearly identical to STEP 1 except for field names and final points
    ## Calculate the intermediate metric IEI
    ## Summarize the IEI raster values for each Wetland Unit
    arcpy.gp.ZonalStatisticsAsTable("fcWULandInteg0", "WUKey", "IEI", "IEI_zonal",
"DATA", "MEAN")
    logger.info("the IEI raster values for each Wetland Unit summarized")


    ## Extract the IEI raster values for each Wetland Unit centroid
    ExtractValuesToPoints("fcWUpoint", "IEI", "WUpoint_IEI")
    logger.info("IEI raster values for each Wetland Unit centroid extracted")


    ## Delete the "Count" field from the IEI_zonal table since "Count" is a restricted word
    ## and may interfere with the join in the next step
    arcpy.DeleteField_management ("IEI_zonal","Count")
    logger.info("field Count deleted")


    ## Join Wetland Units to the IEI_zonal table
    arcpy.AddJoin_management("fcWULandInteg0","WUKey","IEI_zonal","WUKey")
    logger.info("Join Wetland Units to the IEI_zonal table completed")


    ## Export the joined data to a Feature Class
    arcpy.FeatureClassToFeatureClass_conversion("fcWULandInteg0", arcpy.env.workspace,
"WU_IEI1")
    logger.info("joined data exported as WU_IEI1")


    ## Remove Join from Wetland Units
    arcpy.RemoveJoin_management("fcWULandInteg0")
    logger.info("join removed from Wetland Units")

    ## Join WU_IEI1 to WUpoint_IEI
    arcpy.MakeFeatureLayer_management("WU_IEI1", "fcWUIEI1")
    arcpy.MakeFeatureLayer_management("WUpoint_IEI", "fcWUpoint_IEI")

arcpy.AddJoin_management("fcWUIEI1","WU_LandInteg0_WUKey","fcWUpoint_IEI","WU
Key")
    logger.info("joined WU_IEI1 to WUpoint_IEI")

    ## Export joined data to feature class
    arcpy.FeatureClassToFeatureClass_conversion("fcWUIEI1", arcpy.env.workspace,
"WU_IEI")
    arcpy.MakeFeatureLayer_management("WU_IEI", "fcWUIEI")
    logger.info("joined data exported as WU_IEI")

    ## Replace NULL values of MEAN with centroid values of IEI
    strWHERE = """"WU_IEI1_IEI_zonal_MEAN" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcWUIEI", "NEW_SELECTION", strWHERE)

arcpy.CalculateField_management("fcWUIEI","WU_IEI1_IEI_zonal_MEAN","!WUpoint_IEI
_RASTERVALU!","PYTHON","#")
    arcpy.SelectLayerByAttribute_management("fcWUIEI", "CLEAR_SELECTION")
    logger.info("NULL values of MEAN with centroid values of IEI replaced")

    ## Add field IEI, set initial value to zero
    actions.DeleteField("fcWUIEI","IEI")
    arcpy.AddField_management("fcWUIEI", "IEI", "SHORT")
    arcpy.CalculateField_management("fcWUIEI","IEI","0","PYTHON","#")
    logger.info("field IEI added and initial value set to zero")
```

```
## Assign points
strWHERE = """"WU_IEI1_IEI_zonal_MEAN" > 15"""
arcpy.SelectLayerByAttribute_management("fcWUIEI", "NEW_SELECTION", strWHERE)
arcpy.CalculateField_management("fcWUIEI","IEI","1","PYTHON","#")

strWHERE = """"WU_IEI1_IEI_zonal_MEAN" > 45"""
arcpy.SelectLayerByAttribute_management("fcWUIEI", "NEW_SELECTION", strWHERE)
arcpy.CalculateField_management("fcWUIEI","IEI","2","PYTHON","#")

strWHERE = """"WU_IEI1_IEI_zonal_MEAN" > 70"""
arcpy.SelectLayerByAttribute_management("fcWUIEI", "NEW_SELECTION", strWHERE)
arcpy.CalculateField_management("fcWUIEI","IEI","3","PYTHON","#")
logger.info("points assigned in field IEI")
logger.info("Step 2 completed")
arcpy.SelectLayerByAttribute_management("fcWUIEI", "CLEAR_SELECTION")

## STEP 3
## Note that method in nearly identical to STEP 1 except for field names, MAJORITY, and final points
## Calculate the intermediate metric LandResil
## Summarize the Resilient_and_Connected raster values for each Wetland Unit
arcpy.gp.ZonalStatisticsAsTable("fcWULandInteg0", "WUKey", "ResilConnected", "LandResil_zonal", "DATA", "MAJORITY")
logger.info("the Resilient_and_Connected raster values for each Wetland Unit summarized")

## Extract the Resilient_and_Connected raster values for each Wetland Unit centroid
ExtractValuesToPoints("fcWUpoint", "ResilConnected", "WUpoint_LandResil")
logger.info("Resilient_and_Connected raster values for each Wetland Unit centroid extracted")
```

```
    ## Delete the "Count" field from the LandResil_zonal table since "Count" is a restricted word
    ## and may interfere with the join in the next step
    arcpy.DeleteField_management ("LandResil_zonal","Count")
    logger.info("field Count deleted")

    ## Join Wetland Units to the LandResil_zonal table
    arcpy.AddJoin_management("fcWULandInteg0","WUKey","LandResil_zonal","WUKey")
    logger.info("Join Wetland Units to the LandResil_zonal table completed")

    ## Export the joined data to a Feature Class
    arcpy.FeatureClassToFeatureClass_conversion("fcWULandInteg0", arcpy.env.workspace,
"WU_LandResil1")
    logger.info("joined data exported as WU_LandResil1")

    ## Remove Join from Wetland Units
    arcpy.RemoveJoin_management("fcWULandInteg0")
    logger.info("join removed from Wetland Units")

    ## Join WU_LandResil1 to WUpoint_LandResil
    arcpy.MakeFeatureLayer_management("WU_LandResil1", "fcWULandResil1")
    arcpy.MakeFeatureLayer_management("WUpoint_LandResil", "fcWUpoint_LandResil")

arcpy.AddJoin_management("fcWULandResil1","WU_LandInteg0_WUKey","fcWUpoint_LandResil","WUKey")
    logger.info("joined WU_LandResil1 to WUpoint_LandResil")

    ## Export joined data to feature class
    arcpy.FeatureClassToFeatureClass_conversion("fcWULandResil1", arcpy.env.workspace,
"WU_LandResil")
    arcpy.MakeFeatureLayer_management("WU_LandResil", "fcWULandResil")
    logger.info("joined data exported as WU_LandResil")


    ## Replace NULL values of MEAN with centroid values of Landscape Integrity
    strWHERE = """"WU_LandResil1_LandResil_zonal_MAJORITY" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWULandResil","WU_LandResil1_LandResil_zonal_MA
JORITY","!WUpoint_LandResil_RASTERVALU!","PYTHON","#")
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "CLEAR_SELECTION")
    logger.info("NULL values of MEAN with centroid values of Landscape Integrity replaced")


    ## Add field LandResil, set initial value to zero
    actions.DeleteField("fcWULandResil","LandResil")
    arcpy.AddField_management("fcWULandResil", "LandResil", "SHORT")
    arcpy.CalculateField_management("fcWULandResil","LandResil","0","PYTHON","#")
    logger.info("field LandResil added and initial value set to zero")


    ## Assign points
    strWHERE = """"WU_LandResil1_LandResil_zonal_MAJORITY" IN
(2,3,4,11,12,13,14,33,112)"""
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWULandResil","LandResil","1","PYTHON","#")


    strWHERE = """"WU_LandResil1_LandResil_zonal_MAJORITY" IN
(2,4,11,12,13,14,33,112)"""
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWULandResil","LandResil","2","PYTHON","#")
    strWHERE = """"WU_LandResil1_LandResil_zonal_MAJORITY" IN (11,12,112)"""
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWULandResil","LandResil","3","PYTHON","#")
    logger.info("points assigned in field LandResil")
    logger.info("Step 2 completed")
    arcpy.SelectLayerByAttribute_management("fcWULandResil", "CLEAR_SELECTION")

    ## STEP 4
    ## Calculate the intermediate metric ForestPatch

    ## Spatial Join Wetland Units and Forest Patches, selecting for the largest forest patch
    ## that is intersected by the Wetland Unit
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcWULandResil')
    fmSJFLIN.addTable('fcForestPatches')
    keepers = []
    keepers =
['WU_LandResil1_WU_LandInteg0_WUKey','Shape_Length','Shape_Area','LandResil','ACRE
AGE']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJFLIN.findFieldMapIndex("ACREAGE")
    fieldmap = fmSJFLIN.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Maximum"
    fmSJFLIN.replaceFieldMap(fldKeyIndex, fieldmap)

    arcpy.SpatialJoin_analysis('WU_LandResil', 'fcForestPatches', 'WU_ForestPatch',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'INTERSECT')
```

```
    arcpy.MakeFeatureLayer_management('WU_ForestPatch', 'fcForestPatch')
    logger.info("Spatial Join Wetland Units and Forest Patches completed")

    ## Add field ForestPatch, set initial value to zero
    actions.DeleteField("fcForestPatch","ForestPatch")
    arcpy.AddField_management("fcForestPatch", "ForestPatch", "SHORT")
    arcpy.CalculateField_management("fcForestPatch","ForestPatch","0","PYTHON","#")
    logger.info("field ForestPatch added and initial value set to zero")

    ## Assign points
    strWHERE = """"ACREAGE" >= 50"""
    arcpy.SelectLayerByAttribute_management("fcForestPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcForestPatch","ForestPatch","1","PYTHON","#")

    strWHERE = """"ACREAGE" >= 247"""
    arcpy.SelectLayerByAttribute_management("fcForestPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcForestPatch","ForestPatch","2","PYTHON","#")

    strWHERE = """"ACREAGE" >= 2470"""
    arcpy.SelectLayerByAttribute_management("fcForestPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcForestPatch","ForestPatch","3","PYTHON","#")
    logger.info("points assigned in field ForestPatch")
    arcpy.SelectLayerByAttribute_management("fcForestPatch", "CLEAR_SELECTION")

    ## Spatial Join LandIntegDNR metric to the other two metrics
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcForestPatch')
    fmSJFLIN.addTable('fcWULandIntegDNR')
    keepers = []
    keepers =
['WU_LandResil1_WU_LandInteg0_WUKey','Shape_Length','Shape_Area','LandResil','Forest
Patch','LandIntegDNR']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcForestPatch', 'fcWULandIntegDNR', 'WU_LandInteg',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_LandInteg', 'fcLandInteg')
    logger.info("Spatial Join LandIntegDNR metric to the other two metrics completed")

    ## Add field LandInteg, set initial value to zero
    actions.DeleteField("fcLandInteg","LandInteg")
    arcpy.AddField_management("fcLandInteg", "LandInteg", "SHORT")
    arcpy.CalculateField_management("fcLandInteg","LandInteg","0","PYTHON","#")
    logger.info("field LandInteg added and initial value set to zero")

    ## Assign points
    strWHERE = """(( "LandIntegDNR" * 2) + "LandResil" + "ForestPatch") / 4 >= 0.5"""
    arcpy.SelectLayerByAttribute_management("fcLandInteg", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcLandInteg","LandInteg","1","PYTHON","#")

    strWHERE = """(( "LandIntegDNR" * 2) + "LandResil" + "ForestPatch") / 4 >= 1.5"""
    arcpy.SelectLayerByAttribute_management("fcLandInteg", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcLandInteg","LandInteg","2","PYTHON","#")
    strWHERE = """(( "LandIntegDNR" * 2) + "LandResil" + "ForestPatch") / 4 >= 2.5"""
    arcpy.SelectLayerByAttribute_management("fcLandInteg", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcLandInteg","LandInteg","3","PYTHON","#")
    logger.info("points assigned in field LandInteg")
    logger.info("Step 3 completed")
    arcpy.SelectLayerByAttribute_management("fcLandInteg", "CLEAR_SELECTION")
    arcpy.CheckInExtension("spatial")

    # Clean up
    if arcpy.Exists("LandInteg_zonal"):
        arcpy.Delete_management("LandInteg_zonal")
    if arcpy.Exists("LandResil_zonal"):
        arcpy.Delete_management("LandResil_zonal")
    if arcpy.Exists("WU_LandInteg0"):
        arcpy.Delete_management("WU_LandInteg0")
    if arcpy.Exists("WU_LandInteg1"):
        arcpy.Delete_management("WU_LandInteg1")
    if arcpy.Exists("WU_IEI1"):
        arcpy.Delete_management("WU_IEI1")
    if arcpy.Exists("WU_IEI"):
        arcpy.Delete_management("WU_IEI")
    if arcpy.Exists("WUpoint_IEI"):
        arcpy.Delete_management("WUpoint_IEI")
    if arcpy.Exists("WU_LandResil1"):
        arcpy.Delete_management("WU_LandResil1")
    if arcpy.Exists("WU_LandResil"):
        arcpy.Delete_management("WU_LandResil")
    if arcpy.Exists("WU_LandIntegDNR"):
        arcpy.Delete_management("WU_LandIntegDNR")
```

```
if arcpy.Exists("WU_ForestPatch"):
    arcpy.Delete_management("WU_ForestPatch")
if arcpy.Exists("WUpoint_LandInteg"):
    arcpy.Delete_management("WUpoint_LandInteg")
if arcpy.Exists("WUpoint_LandResil"):
    arcpy.Delete_management("WUpoint_LandResil")
```

5.7.39 WetldBird: Habitat and Ecological Integrity Opportunity

```
#############################################################################
#
# File Name: WetldBird.py
# Developer: Yibing Han
# Date: 10/09/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)
#
#############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcWetldBird(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HOpportun.WetldBird")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcWetBird, "fcWetBird")

    # Clean up if needed
    if arcpy.Exists("WU_WetlandBird1"):
        arcpy.Delete_management("WU_WetlandBird1")
    if arcpy.Exists("WU_WetlandBird"):
        arcpy.Delete_management("WU_WetlandBird")
    logger.info("feature layers ready")

    ## Create feature class to store results for WetldBird and set initial value to zero
    arcpy.CopyFeatures_management(WetlandPoly,"WU_WetlandBird1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_WetlandBird1", "fcWetlandBird1")
    logger.info("feature class WU_WetlandBird1 created")

    actions.DeleteField("fcWetlandBird1", "WetldBird")
    arcpy.AddField_management("fcWetlandBird1", "WetldBird", "SHORT")
    arcpy.CalculateField_management("fcWetlandBird1","WetldBird","0","VB","#")
    logger.info("field WetldBird added and initial value set to 0")

    ## Select atlas blocks in the upper 75% of wetland bird occupancy values
    strWHERE = """"WetBird" > 0.354"""
    arcpy.SelectLayerByAttribute_management("fcWetBird", "NEW_SELECTION",
strWHERE)
    logger.info("atlas blocks in the upper 75% of wetland bird occupancy values selected")

    ## Select Wetland Units that intersect the selected atlas blocks
    arcpy.SelectLayerByLocation_management("fcWetlandBird1", "INTERSECT",
"fcWetBird")
    logger.info("Wetland Units that intersect the selected atlas blocks selected")

    ## Assign 1 points
    arcpy.CalculateField_management("fcWetlandBird1","WetldBird","1","VB","#")
    logger.info("1 point assigned to qualifying wetland units")

    ## Select atlas blocks in the upper 50% of wetland bird occupancy values
    strWHERE = """"WetBird" > 0.408"""
    arcpy.SelectLayerByAttribute_management("fcWetBird", "NEW_SELECTION", 
strWHERE)
    logger.info("atlas blocks in the upper 50% of wetland bird occupancy values selected")


    ## Select Wetland Units that intersect the selected atlas blocks
    arcpy.SelectLayerByLocation_management("fcWetlandBird1", "INTERSECT", 
"fcWetBird")
    logger.info("Wetland Units that intersect the selected atlas blocks selected")


    ## Assign 2 points
    arcpy.CalculateField_management("fcWetlandBird1","WetldBird","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")


    ## Select atlas blocks in the upper 10% of wetland bird occupancy values
    strWHERE = """"WetBird" > 0.493"""
    arcpy.SelectLayerByAttribute_management("fcWetBird", "NEW_SELECTION", 
strWHERE)
    logger.info("atlas blocks in the upper 10% of wetland bird occupancy values selected")


    ## Select Wetland Units that intersect the selected atlas blocks
    arcpy.SelectLayerByLocation_management("fcWetlandBird1", "INTERSECT", 
"fcWetBird")
    logger.info("Wetland Units that intersect the selected atlas blocks selected")


    ## Assign 3 points and clear selections
    arcpy.CalculateField_management("fcWetlandBird1","WetldBird","3","VB","#")
    logger.info("3 points assigned to qualifying wetland units")


    arcpy.SelectLayerByAttribute_management("fcWetBird", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcWetlandBird1", "CLEAR_SELECTION")
    ## Select wetlands that fall outside the atlas block coverage
    arcpy.SelectLayerByLocation_management("fcWetlandBird1", "INTERSECT",
"fcWetBird")
    arcpy.SelectLayerByAttribute_management("fcWetlandBird1", "SWITCH_SELECTION")
    logger.info("wetlands that fall outside the atlas block coverage selected")

    ## Set temporary value of WetldBird for wetlands outside the atlas block coverage
    arcpy.CalculateField_management("fcWetlandBird1","WetldBird","99","VB","#")
    logger.info("temporary value of WetldBird for wetlands outside the atlas block coverage set")
    arcpy.SelectLayerByAttribute_management("fcWetlandBird1", "CLEAR_SELECTION")

    ## Spatial Join to find the closest atlas block to each wetland
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWetlandBird1")
    fmSJ.addTable("fcWetBird")
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','WetldBird','WetBird']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis("fcWetlandBird1", "fcWetBird", 'WU_WetlandBird',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CLOSEST')
    fcWetlandBird = arcpy.mapping.Layer(r"WU_WetlandBird")
    logger.info("Spatial Join to find the closest atlas block to each wetland completed")

    ## Assign points based on WetBird value to wetlands outside the atlas blocks
    strWHERE = """"WetldBird" = 99 AND "WetBird" <= 0.354"""
    arcpy.SelectLayerByAttribute_management(fcWetlandBird, "NEW_SELECTION",
strWHERE)
```

```
    arcpy.CalculateField_management(fcWetlandBird,"WetldBird","0","VB","#")


    strWHERE = """"WetldBird" = 99 AND "WetBird" > 0.354 AND "WetBird" <= 0.408""""
    arcpy.SelectLayerByAttribute_management(fcWetlandBird, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcWetlandBird,"WetldBird","1","VB","#")


    strWHERE = """"WetldBird" = 99 AND "WetBird" > 0.408 AND "WetBird" <= 0.493""""
    arcpy.SelectLayerByAttribute_management(fcWetlandBird, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcWetlandBird,"WetldBird","2","VB","#")


    strWHERE = """"WetldBird" = 99 AND "WetBird" > 0.493""""
    arcpy.SelectLayerByAttribute_management(fcWetlandBird, "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management(fcWetlandBird,"WetldBird","3","VB","#")
    logger.info("Points assigned to WetldBird")
    arcpy.SelectLayerByAttribute_management(fcWetlandBird, "CLEAR_SELECTION")


    # Clean up
    if arcpy.Exists("WU_WetlandBird1"):
        arcpy.Delete_management("WU_WetlandBird1")
```

5.7.40 WshdPos: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: WshdPos.py
# Developer: Yibing Han
# Date: 10/03/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandHydro: Landscape Hydrologic Connectivity)
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcWshdPos():
    logger = logging.getLogger("WFA.HabEco.HOpportun.WshdPos")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea", 'fcFloodArea')
    arcpy.MakeFeatureLayer_management(r"WU_Headwater", 'fcHeadwater')
    arcpy.MakeFeatureLayer_management(r"WU_Karst", 'fcKarst')
    arcpy.MakeFeatureLayer_management(globalvars.srcDrainage, 'fcDrainage')
    logger.info("feature layers ready")
    # Clean up if needed
    if arcpy.Exists('WU_WshdPos1'):
        arcpy.Delete_management('WU_WshdPos1')
    if arcpy.Exists('WU_WshdPos2'):
        arcpy.Delete_management('WU_WshdPos2')
    if arcpy.Exists('WU_WshdPos'):
        arcpy.Delete_management('WU_WshdPos')

    ## STEP 1: Wetland Units in the Floodplain of a Major River
    ## Create feature class to store WshdPos
    arcpy.CopyFeatures_management('fcFloodArea',"WU_WshdPos1","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WU_WshdPos1", 'fcWshdPos1')
    logger.info("feature class WU_WshdPos1 created")

    ## Add field to store major river floodplain (MajorRiverFP) and set initial value to zero
    actions.DeleteField('fcWshdPos1', 'MajorRiverFP')
    arcpy.AddField_management('fcWshdPos1', 'MajorRiverFP', 'SHORT')
    arcpy.CalculateField_management('fcWshdPos1', 'MajorRiverFP', '0', 'VB', '#')
    logger.info("field MajorRiverFP added and set to 0")

    ## Select major rivers with floodplains
    strWHERE = """"DA_sq_mi" > 1000 OR "GNIS_Name" IN ('Meadow River', 'Cacapon
River', 'Tygart Valley River') OR ("GNIS_Name" IN ('South Branch Potomac River', 'North
Branch Potomac River') AND "DA_sq_mi" > 1000)"""
    arcpy.SelectLayerByAttribute_management('fcDrainage', "NEW_SELECTION",
strWHERE)
    logger.info("Major rivers with floodplains selected")

    ## Select wetlands that are within 500 meters of selected rivers
    arcpy.SelectLayerByLocation_management('fcWshdPos1', "WITHIN_A_DISTANCE",
'fcDrainage', "500 Meters")
```

```
    logger.info("wetlands that are within 500 meters of selected rivers selected")

    ## Remove from selection any wetlands not in the floodplain
    strWHERE = """"Floodplain" = 'N'"""
    arcpy.SelectLayerByAttribute_management('fcWshdPos1',
"REMOVE_FROM_SELECTION", strWHERE)
    logger.info("wetlands not in the floodplain removed from selection")

    ## Assign values to MajorRiverFP
    arcpy.CalculateField_management('fcWshdPos1', 'MajorRiverFP', '1', 'VB', '#')
    arcpy.SelectLayerByAttribute_management('fcWshdPos1', "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management('fcDrainage', "CLEAR_SELECTION")
    logger.info("values Assigned for MajorRiverFP")

    ## STEP 2: Join metrics and assign points
    ## Spatial Join MajorRiversFP to Headwater and Karst

    ## Spatial join to merge BufferPerim and BufferContig into one attribute table
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable('fcWshdPos1')
    fmSJ.addTable('fcHeadwater')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','MajorRiverFP','Headwater']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcWshdPos1', 'fcHeadwater', 'WU_WshdPos2',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_WshdPos2", 'fcWshdPos2')
    logger.info("spatial join WshdPos1 and Headwater completed")

    ## Spatial join to merge LandInteg
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable('fcWshdPos2')
    fmSJ.addTable('fcKarst')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','MajorRiverFP','Headwater','Karst']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcWshdPos2', 'fcKarst', 'WU_WshdPos', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJ, 'CONTAINS')
    fcWshdPos = arcpy.mapping.Layer(r"WU_WshdPos")
    logger.info("spatial join WshdPos2 and Karst completed")

    ## Add WshdPos field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcWshdPos, 'WshdPos')
    arcpy.AddField_management(fcWshdPos, 'WshdPos', 'SHORT')
    arcpy.CalculateField_management(fcWshdPos, 'WshdPos', '0', 'VB', '#')
    logger.info("field WshdPos added and value set to 0")

    ## Assign points
    strWHERE = """"MajorRiverFP" > 0 OR "Headwater" > 0 OR "Karst" > 0"""
    arcpy.SelectLayerByAttribute_management(fcWshdPos, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWshdPos, 'WshdPos', '1', 'VB', '#')
    logger.info("field WshdPos calculated")
    arcpy.SelectLayerByAttribute_management(fcWshdPos, "CLEAR_SELECTION")
```

```
# Clean up
if arcpy.Exists('WU_WshdPos1'):
    arcpy.Delete_management('WU_WshdPos1')
if arcpy.Exists('WU_WshdPos2'):
    arcpy.Delete_management('WU_WshdPos2')
```

5.7.41 WshdUniq: Habitat and Ecological Integrity Opportunity

```
##############################################################################
#
# File Name: ConsFocus.py
# Developer: Yibing Han
# Date: 10/10/2017
# Purpose:
#    Input to Habitat / Landscape Opportunity (LandEco: Landscape Ecological Connectivity)
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcWshdUniq():
    logger = logging.getLogger("WFA.HabEco.HOpportun.WshdUniq")

    # Setting python variables

arcpy.MakeFeatureLayer_management(globalvars.srcHUCWetSizeUniq,"fcHUCWetSizeUniq"
)
    arcpy.MakeFeatureLayer_management(r"WU_VegAll","fcVegAll")

    # Clean up if needed
    if arcpy.Exists("WU_WshdUniq"):
        arcpy.Delete_management("WU_WshdUniq")

    ## Spatial Join of Wetland Units (including VegArea) and HUC characteristics
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcVegAll")
    fmSJFLIN.addTable("fcHUCWetSizeUniq")
    keepers = []
    keepers =
['WUKey','VegArea','Shape_Length','Shape_Area','HUC_12','HU_12_NAME','MaxVegArea','D
iverseNWI','DensVegNWI','RatioVeg']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegAll', 'fcHUCWetSizeUniq', 'WU_WshdUniq',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'HAVE_THEIR_CENTER_IN')
    fcWshdUniq = arcpy.mapping.Layer(r"WU_WshdUniq")
    logger.info("Spatial Join of Wetland Units (including VegArea) and HUC characteristics
completed")

    ## Create feature class to store results for WshdUniq and set initial value to zero
    actions.DeleteField(fcWshdUniq, "WshdUniq")
    arcpy.AddField_management(fcWshdUniq, "WshdUniq", "SHORT")
    arcpy.CalculateField_management(fcWshdUniq,"WshdUniq","0","VB","#")
    logger.info("field WshdUniq added and initial value set to 0")

    ## Select wetlands in top 10% of HUC watersheds based on
    ## the number of unique NWI codes, the number of vegetated NWI polygons, or the
proportional area of vegetated wetlands
    strWHERE = """"DiverseNWI" > 22 OR "DensVegNWI" > 45 OR "RatioVeg" > 0.005"""
    arcpy.SelectLayerByAttribute_management(fcWshdUniq, "NEW_SELECTION",
strWHERE)
    logger.info("all CFAs except the general CFA selected")


    ## Assign 1 point
    arcpy.CalculateField_management(fcWshdUniq,"WshdUniq","1","VB","#")
    logger.info("1 point assigned to qualifying wetland units")


    ## Select largest vegetated wetland in each HUC AND wetlands in top 5% of HUC
watersheds based on
    ## type diversity, density, or proportional area of vegetated wetlands
    strWHERE = """"VegArea" >= "MaxVegArea" OR "DiverseNWI" > 28 OR "DensVegNWI"
> 70 OR "RatioVeg" > 0.009"""
    arcpy.SelectLayerByAttribute_management(fcWshdUniq, "NEW_SELECTION",
strWHERE)
    logger.info("CFAs with wetland focus selected")


    ## Assign 2 points
    arcpy.CalculateField_management(fcWshdUniq,"WshdUniq","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")


    arcpy.SelectLayerByAttribute_management(fcWshdUniq, "CLEAR_SELECTION")


    # Clean up
```

5.7.42 Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: HPotential.py
# Developer: Yibing Han
# Date: 12/12/2017
# Purpose:
#    This script handles the execution of all the Habitat and Ecological Integrity Potential
metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import Histosol, HydIntact, HydSW, Karst, MarlPEM, SoilIntact, SoilOrgCalc,
StrucPatch, VegFQ, VegHorInt, VegVerStr
from Factors import VegH, HydroH, SoilH
from Aspects import HPotential

def procHPotential(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HPotential")

    ############################################################
    ## 1. Run Variables
    ############################################################

    Histosol.CalcHistosol(WetlandPoly)
    #Karst.CalcKarst(WetlandPoly)
    MarlPEM.CalcMarlPEM(WetlandPoly)
    HydIntact.CalcHydIntact()
    HydSW.CalcHydSW()
    SoilIntact.CalcSoilIntact()
    SoilOrgCalc.CalcSoilOrgCalc()
    VegFQ.CalcVegFQ()
    VegHorInt.CalcVegHorInt()
    VegVerStr.CalcVegVerStr()
    StrucPatch.CalcStrucPatch()

    ## Executed with FloodAttenuation:
    # ConnectFL
    # Runoff50m
    # RunoffWshd
    # StreamEdge
```

```
## Executed with WaterQuality:
# Disturb50m
# VegPerUng1
# VegWoody2
# VegWoodyFor
# WQOpportun

##########################################################
## 2. Run Factors
##########################################################
VegH.CalcVegH()
HydroH.CalcHydroH()
SoilH.CalcSoilH()

##########################################################
## 3. Run Aspect
##########################################################
HPotential.CalcHPotential()
```

5.7.43 HPotential: Habitat and Ecological Integrity Potential Aspects

```
###############################################################################
#
# File Name: HPotential.py
# Developer: Yibing Han
# Date 9/28/2017
# Purpose:
#    Habitat Function
#
###############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def CalcHPotential():
    logger = logging.getLogger("WFA.HabEco.HPotential.HPotential")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_VegH", 'fcVegH')
    arcpy.MakeFeatureLayer_management(r"WU_HydroH", 'fcHydroH')
    arcpy.MakeFeatureLayer_management(r"WU_SoilH", 'fcSoilH')

    # Clean up if needed
    if arcpy.Exists('WU_HPotential1'):
        arcpy.Delete_management('WU_HPotential1')
    if arcpy.Exists('WU_HPotential'):
        arcpy.Delete_management('WU_HPotential')
    logger.info("feature layers ready")

    ## Spatial joins to bring together factor values
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegH')
    fmSJFLIN.addTable('fcHydroH')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegH','HydroH']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegH', 'fcHydroH', 'WU_HPotential1', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_HPotential1', 'fcHPotential1')
    logger.info("spatial join of VegH and HydroH completed")

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcHPotential1')
    fmSJFLIN.addTable('fcSoilH')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegH','HydroH','SoilH']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcHPotential1', 'fcSoilH', 'WU_HPotential',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
```

```
    arcpy.MakeFeatureLayer_management('WU_HPotential', 'fcHPotential')
    logger.info("spatial join of HPotential1 and SoilH completed")

    ## Add HPotential field and set initial point value to zero.
    actions.DeleteField('fcHPotential', 'HPotential')
    arcpy.AddField_management('fcHPotential', 'HPotential', 'SHORT')
    arcpy.CalculateField_management('fcHPotential', 'HPotential', '0', 'VB', '#')
    logger.info("HPotential field added and initial point value set to zero")

    ## Sum the factor points
    arcpy.CalculateField_management("fcHPotential","HPotential","[VegH]+ [HydroH]+
[SoilH]","VB","#")
    logger.info("HPotential field caculated")

    # Clean up
    if arcpy.Exists('WU_HPotential1'):
        arcpy.Delete_management('WU_HPotential1')
```

5.7.44 HydroH: Habitat and Ecological Integrity Potential

```
##########################################################################
#
# File Name: HydroH.py
# Developer: Yibing Han
# Date 9/27/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHydroH():
    logger = logging.getLogger("WFA.HabEco.HPotential.HydroH")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Connect", 'fcConnectFL')
    arcpy.MakeFeatureLayer_management(r"WU_HydroH", 'fcHydroH')

    # Clean up if needed
    logger.info("feature layers ready")
    ## Add field to store ConnectFL
    actions.DeleteField('fcHydroH', 'ConnectFL1')
    arcpy.AddField_management('fcHydroH', 'ConnectFL1', 'SHORT')
    logger.info("field ConnectFL1 added to store ConnectFL")

    ## Add Join to merge fields into one attribute table
    arcpy.AddJoin_management("fcHydroH","WUKey","fcConnectFL","WUKey")
    logger.info("Add Join to merge fields into one attribute table completed")

    ## Calculate field ConnectFL1
    arcpy.CalculateField_management('fcHydroH', 'WU_HydroH.ConnectFL1',
'[WU_Connect.ConnectFL]', 'VB', '#')
    logger.info("field ConnectFL1 calculated")

    ## Remove Join
    arcpy.RemoveJoin_management("fcHydroH")
    logger.info("join removed")

    ## Add field to store HydroH and set initial value to zero
    actions.DeleteField('fcHydroH', 'HydroH')
    arcpy.AddField_management('fcHydroH', 'HydroH', 'SHORT')
    arcpy.CalculateField_management('fcHydroH', 'HydroH', '0', 'VB', '#')
    logger.info("field added to store HydroH and initial value set to zero")

    ## Assign points to HydroH
    arcpy.CalculateField_management('fcHydroH', 'HydroH', '[HydIntact] + [HydSW] +
[ConnectFL1]', 'VB', '#')
    logger.info("points assigned to HydroH")

    # Clean up if needed
```

5.7.45 SoilH: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: SoilH.py
# Developer: Yibing Han
# Date 9/28/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcSoilH():
    logger = logging.getLogger("WFA.HabEco.HPotential.SoilH")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Disturb50m", 'fcDisturb50m')
    arcpy.MakeFeatureLayer_management(r"WU_SoilOrgCalc", 'fcSoilOrgCalc')
    arcpy.MakeFeatureLayer_management(r"WU_StrucPatch", 'fcStrucPatch')

    # Clean up if needed
    if arcpy.Exists('WU_SoilH1'):
        arcpy.Delete_management('WU_SoilH1')
    if arcpy.Exists('WU_SoilH'):
        arcpy.Delete_management('WU_SoilH')
    logger.info("feature layers ready")

    ## Spatial Joins to merge fields into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcDisturb50m')
    fmSJFLIN.addTable('fcSoilOrgCalc')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','SoilIntact','SoilOrgCalc']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcDisturb50m', 'fcSoilOrgCalc', 'WU_SoilH1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_SoilH1', 'fcSoilH1')

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcSoilH1')
    fmSJFLIN.addTable('fcStrucPatch')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','SoilIntact','SoilOrgCalc','StrucPatch']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcSoilH1', 'fcStrucPatch', 'WU_SoilH', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
```

```
    arcpy.MakeFeatureLayer_management('WU_SoilH', 'fcSoilH')
    logger.info("Spatial Joins to merge fields into one attribute table completed")

    ## Add SoilH field and set initial point value to zero.
    actions.DeleteField('fcSoilH', 'SoilH')
    arcpy.AddField_management('fcSoilH', 'SoilH', 'SHORT')
    arcpy.CalculateField_management('fcSoilH', 'SoilH', '0', 'VB', '#')
    logger.info("field SoilH added and initial value set to zero")

    ## Assign points to fcSoilH
    arcpy.CalculateField_management("fcSoilH","SoilH","[SoilIntact] + [SoilOrgCalc] +
[StrucPatch]","VB","#")
    logger.info("points assigned to SoilH")

    # Clean up
    if arcpy.Exists('WU_SoilH1'):
        arcpy.Delete_management('WU_SoilH1')
```

5.7.46 VegH: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: VegH.py
# Developer: Yibing Han
# Date 9/25/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegH():
    logger = logging.getLogger("WFA.HabEco.HPotential.VegH")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_VegVerStr", 'fcVegVerStr')
    arcpy.MakeFeatureLayer_management(r"WU_Microtopo", 'fcMicrotopo')
    arcpy.MakeFeatureLayer_management(r"WU_VegFQ", 'fcVegFQ')

    # Clean up if needed
    if arcpy.Exists('WU_VegH'):
        arcpy.Delete_management('WU_VegH')
    if arcpy.Exists('WU_VegH1'):
        arcpy.Delete_management('WU_VegH1')
    logger.info("feature layers ready")

    ## Spatial join to merge VegVerStr and VegHorInt into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegVerStr')
    fmSJFLIN.addTable('fcMicrotopo')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegVerStr','VegHorInt']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegVerStr', 'fcMicrotopo', 'WU_VegH1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegH1', 'fcVegH1')
    logger.info("Spatial join to merge VegVerStr and VegHorInt completed")

    ## Spatial join to merge VegFQ
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegH1')
    fmSJFLIN.addTable('fcVegFQ')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegVerStr','VegHorInt','VegFQ']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcVegH1', 'fcVegFQ', 'WU_VegH', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegH', 'fcVegH')
    logger.info("Spatial join to merge VegFQ completed")

    ## Add VegH field to Wetland Units and set initial point value to zero.
    actions.DeleteField('fcVegH', 'VegH')
    arcpy.AddField_management('fcVegH', 'VegH', 'SHORT')
    arcpy.CalculateField_management('fcVegH', 'VegH', '0', 'VB', '#')
    logger.info("field VegH added and initial value set to 0")

    ## Sum the points for VegVerStr, VegHorInt, VegFQ
    arcpy.CalculateField_management('fcVegH', 'VegH', '[VegVerStr] + [VegHorInt] +
[VegFQ]', 'VB', '#')
    logger.info("Points calculated for field VegH")

    # Clean up
    if arcpy.Exists('WU_VegH1'):
        arcpy.Delete_management('WU_VegH1')
```

5.7.47 Histosol: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: Histosol.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 2/7/2017 (modified 11/10/2017)
# Purpose:
#    Input to Habitat and Ecological Integrity / Intrinsic Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHistosol(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HPotential.Histosol")

    # Clean up if needed
    if arcpy.Exists("WU_Histosol"):
        arcpy.Delete_management("WU_Histosol")

    # Setting python variables
    arcpy.CopyFeatures_management(WetlandPoly, "WU_Histosol")
    fcWUHistosol = arcpy.mapping.Layer(r"WU_Histosol")
    fcPeatlands = arcpy.mapping.Layer(globalvars.srcPeatlands)
    fcPalustrineplots = arcpy.mapping.Layer(globalvars.srcPalustrineplots)
    fcHisticEpipedon = arcpy.mapping.Layer(globalvars.srcHisticEpipedon)
    fcHistosol = arcpy.mapping.Layer(globalvars.srcHistosol)
    # fcSSURGOWV = arcpy.mapping.Layer(globalvars.srcSSURGOWV)
    fcNWI = arcpy.mapping.Layer(globalvars.srcInput)
    logger.info("feature layers ready")

    # Add Histosol field to Wetland Units and set initial point value to zero
    actions.DeleteField(fcWUHistosol,"Histosol")
    arcpy.AddField_management(fcWUHistosol, "Histosol", "SHORT")
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","0","VB","#")
    logger.info("field Histosol added and initial point value set to 0")


##################################################################################
##################################
    ## Part 1: Histic Epipedon

##################################################################################
##################################
    ## Select Palustrine plots that have peat or muck soils (conservative assumption that these are
histic epipedons) or organic oils 20-39 cm thick
    strSelectAttr = """"Soil_Textu" LIKE '%peat%' OR "Soil_Textu" LIKE '%muck%' OR
"Profile__1" LIKE '%peat%' OR "Profile__1" LIKE '%muck%' or( "DepOrgSoil" > 19 AND
"DepOrgSoil" < 40)"""

arcpy.SelectLayerByAttribute_management(fcPalustrineplots,"NEW_SELECTION",strSelectA
ttr)

    ## Select Wetland Units that intersect palustrine plots with histic epipedons.
```

```
arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcPalustrineplots,"#
","NEW_SELECTION")

    ## Update value for "Histosol" based on palustrine plots.
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","2","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcPalustrineplots, "CLEAR_SELECTION")

    ## Select Wetland Units that intersect with the SSURGO histic epipedon selection.

arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcHisticEpipedon,"#
","NEW_SELECTION")

    ## Update calue for "Histosol" based on ssurgo data
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","2","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcHisticEpipedon, "CLEAR_SELECTION")


################################################################################
##################################
    ## Part 2: Histosol

################################################################################
##################################
    ## Select Palustrine plots that have organic soils at least 40cm thick
    strSelectAttr = """"DepOrgSoil" > 39"""

arcpy.SelectLayerByAttribute_management(fcPalustrineplots,"NEW_SELECTION",strSelectA
ttr)
```

```
    ## Select Wetland Units that intersect palustrine plots with histic histosols.

arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcPalustrineplots,"#
","NEW_SELECTION")

    ## Update value for "Histosol" based on palustrine plots.
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","3","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcPalustrineplots, "CLEAR_SELECTION")

    ## Select Wetland Units that intersect with the SSURGO histosol selection.

arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcHistosol,"#","NE
W_SELECTION")

    ## Update calue for "Histosol" based on ssurgo data
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","3","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcHistosol, "CLEAR_SELECTION")


################################################################################
##################################
    ## Part 3: Peatlands

################################################################################
##################################
    ## Select Wetland Units that are peatlands.

arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcPeatlands,"#","NE
W_SELECTION")
```

```
    ## Update calue for "Histosol" based on peatlands
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","3","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcPeatlands, "CLEAR_SELECTION")


################################################################################
##################################
    ## Part 4: NWI Organic Modifier

################################################################################
##################################
    ## Select polygons that have an organic modifier in the Nation Wetland Inventory

arcpy.SelectLayerByAttribute_management(fcNWI,"NEW_SELECTION","""ATTRIBUTE"
LIKE '%g'""")

    ## Select Wetland Unit that intersect organic NWI polygons.

arcpy.SelectLayerByLocation_management(fcWUHistosol,"INTERSECT",fcNWI,"#","NEW_
SELECTION")

    ## Update calue for "Histosol" based on NWI
    arcpy.CalculateField_management(fcWUHistosol,"Histosol","3","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUHistosol, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcNWI, "CLEAR_SELECTION")

    # Clean up
```

5.7.48 HydIntact: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: HydIntact.py
# Developer: Yibing Han
# Date 9/25/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHydIntact():
    logger = logging.getLogger("WFA.HabEco.HPotential.HydIntact")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_WQOpportun", 'fcWQOpportun')
    arcpy.MakeFeatureLayer_management(r"WU_LandInteg", 'fcLandInteg')

    # Clean up if needed
    if arcpy.Exists('WU_HydroH'):
        arcpy.Delete_management('WU_HydroH')
    logger.info("feature layers ready")


    ## Spatial join to merge WQOpportun and LandInteg into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcWQOpportun')
    fmSJFLIN.addTable('fcLandInteg')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','WQOpportun','LandInteg']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))


    arcpy.SpatialJoin_analysis('fcWQOpportun', 'fcLandInteg', 'WU_HydroH',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_HydroH', 'fcHydroH')
    logger.info("Spatial join to merge WQOpportun and LandInteg completed")


    ## Add HydIntact field to Wetland Units and set initial point value to zero.
    actions.DeleteField('fcHydroH', 'HydIntact')
    arcpy.AddField_management('fcHydroH', 'HydIntact', 'SHORT')
    arcpy.CalculateField_management('fcHydroH', 'HydIntact', '0', 'VB', '#')
    logger.info("field HydIntact added and initial value set to 0")


    ## Assign points to HydIntact
    strWHERE = """("WQOpportun" = 5 AND "LandInteg" IN (2,3)) OR ("WQOpportun" = 4
AND "LandInteg" IN (0,1))"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","1","VB","#")
    strWHERE = """("WQOpportun" = 4 AND "LandInteg" IN (2,3)) OR ("WQOpportun" = 3 AND "LandInteg" IN (0,1))"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","2","VB","#")

    strWHERE = """("WQOpportun" = 3 AND "LandInteg" IN (2,3)) OR ("WQOpportun" = 2 AND "LandInteg" IN (0,1))"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","3","VB","#")

    strWHERE = """("WQOpportun" = 2 AND "LandInteg" IN (2,3)) OR ("WQOpportun" = 1 AND "LandInteg" IN (0,1))"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","4","VB","#")

    strWHERE = """("WQOpportun" = 1 AND "LandInteg" IN (2,3)) OR ("WQOpportun" = 0 AND "LandInteg" IN (0,1))"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","5","VB","#")

    strWHERE = """"WQOpportun" = 0 AND "LandInteg" IN (2,3)"""
    arcpy.SelectLayerByAttribute_management("fcHydroH", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcHydroH","HydIntact","6","VB","#")
    logger.info("Points assigned for wetland units to HydIntact")
    arcpy.SelectLayerByAttribute_management("fcHydroH", "CLEAR_SELECTION")

    # Clean up
```

5.7.49 HydSW: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: HydSW.py
# Developer: Yibing Han
# Date 9/25/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Intrinsic Potential / Hydrology
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHydSW():
    logger = logging.getLogger("WFA.HabEco.HPotential.HydSW")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_HydroH", 'fcHydroH')
    arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcInput")
    arcpy.MakeFeatureLayer_management(globalvars.srcNWIOpenWater, 'fcNWIOpenWater')
    logger.info("feature layers ready")

    ## Select the wetland polygons that are attributed as open water, including lakes, rivers, and
open
```

```
    ## water palustrine (aquatic bed, unconsolidated bottom, unconsolidated shore) AND have a
    ## hydrologic regime that is (permanently flooded, semipermanently flooded, intermittently
exposed)
    ## AND are not spoil.
    '''strWHERE = """("ATTRIBUTE" LIKE 'L%' OR"ATTRIBUTE" LIKE 'R%' OR
"ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PUB%' OR "ATTRIBUTE" LIKE
'PUS%') AND ("ATTRIBUTE" LIKE '%H%' OR "ATTRIBUTE" LIKE '%G%' OR
"ATTRIBUTE" LIKE '%F%') AND "ATTRIBUTE" NOT LIKE '%s%'"""
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "NEW_SELECTION",
strWHERE)
    logger.info("wetland polygons that are attributed as open water selected")

    ## Create open water layer from selection

arcpy.CreateFileGDB_management(r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND",
"NWIExports.gdb")
    arcpy.CopyFeatures_management("fcEnhWVWetland", globalvars.srcNWIOpenWater)
    arcpy.MakeFeatureLayer_management(globalvars.srcNWIOpenWater, 'fcNWIOpenWater')
    logger.info("open water layer from selection created")'''

    ## Add field to store HydSW and set initial value to zero
    actions.DeleteField("fcHydroH","HydSW")
    arcpy.AddField_management("fcHydroH", "HydSW", "SHORT")
    arcpy.CalculateField_management("fcHydroH","HydSW","0","PYTHON","#")
    logger.info("field HydSW created and initial value set to zero")

    ## Select the Wetland Units that intersect or touch NWIOpenWater
    arcpy.SelectLayerByLocation_management("fcHydroH", "INTERSECT",
"fcNWIOpenWater")
    logger.info("Wetland Units that intersect or touch NWIOpenWater selected")

    ## Select the Wetland Units that contain open water within their boundaries
    strWHERE = """("ATTRIBUTE" LIKE 'L%' OR"ATTRIBUTE" LIKE 'R%' OR
"ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PUB%' OR "ATTRIBUTE" LIKE
'PUS%') AND ("ATTRIBUTE" LIKE '%H%' OR "ATTRIBUTE" LIKE '%G%' OR
"ATTRIBUTE" LIKE '%F%') AND "ATTRIBUTE" NOT LIKE '%s%'"""
    arcpy.SelectLayerByAttribute_management("fcInput", "NEW_SELECTION", strWHERE)
    arcpy.SelectLayerByLocation_management("fcHydroH","INTERSECT","fcInput")

    ## Assign points to HydSW
    arcpy.CalculateField_management("fcHydroH","HydSW","1","PYTHON","#")
    arcpy.SelectLayerByAttribute_management("fcHydroH", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcInput", "CLEAR_SELECTION")
    logger.info("points assigned to HydSW")

    # Clean up
```

5.7.50 Karst: Habitat and Ecological Integrity Potential

```
##########################################################################
#
# File Name: Karst.py
# Developer: Yibing Han
# Date 9/20/2017
# Purpose:
#    Input to Habitat / Potential / Vegetation / Floristic Quality
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcKarst(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HPotential.Karst")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcKarstComp,"fcKarstComposite")

    # Clean up if needed
    if arcpy.Exists("WU_Karst"):
        arcpy.Delete_management("WU_Karst")
    if arcpy.Exists("WU_Karst1"):
        arcpy.Delete_management("WU_Karst1")
    logger.info("feature layers ready")

    ## Intersect karst and Wetland Units
    arcpy.Intersect_analysis(["fcKarstComposite", "fcWU"], "WU_Karst1", "ONLY_FID")
    logger.info("Intersect of karst and Wetland Units completed")

    ## Add field to store karst area.
    actions.DeleteField("WU_Karst1","KarstArea")
    arcpy.AddField_management("WU_Karst1", "KarstArea", "FLOAT")
    arcpy.CalculateField_management("WU_Karst1","KarstArea","[Shape_Area]","VB","#")
    arcpy.MakeFeatureLayer_management(r"WU_Karst1","fcWUKarst1")
    logger.info("field KarstArea added to store karst area")

    ## Spatial Join karst selection to Wetland Units and sum karst area.
##################################################################################
#################
    # SJ: FloodIn
##################################################################################
#################
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcWU")
    fmSJFLIN.addTable("fcWUKarst1")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","KarstArea"]

    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJFLIN.findFieldMapIndex("KarstArea")
    fieldmap = fmSJFLIN.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJFLIN.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcWUKarst1","WU_Karst","JOIN_ONE_TO_ONE","KEE
P_ALL",fmSJFLIN,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_Karst","fcWUKarst")
    logger.info("Spatial Join karst selection to Wetland Units completed")

    ## Add fields to store KarstRatio and Karst.
    actions.DeleteField("fcWUKarst","KarstRatio")
    arcpy.AddField_management("fcWUKarst", "KarstRatio", "FLOAT")
    arcpy.AddField_management("fcWUKarst", "Karst", "SHORT")
    arcpy.CalculateField_management("fcWUKarst","Karst","0","VB","#")

arcpy.CalculateField_management("fcWUKarst","KarstRatio","[KarstArea]/[Shape_Area]","V
B","#")
    logger.info("field KarstRatio added to store KarstRatio and Karst")

    ## Assign points to Wetland Units for VegHorInt
    strWHERE = """"KarstRatio" > 0.1"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","1","VB","#")

    strWHERE = """"KarstRatio" > 0.33"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","2","VB","#")


    strWHERE = """"KarstRatio" > 0.67"""
    arcpy.SelectLayerByAttribute_management("fcWUKarst", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUKarst","Karst","3","VB","#")
    logger.info("points assigned to Wetland Units for field KarstRatio")


    arcpy.SelectLayerByAttribute_management("fcWUKarst", "CLEAR_SELECTION")


    # Clean up
    if arcpy.Exists("WU_Karst1"):
        arcpy.Delete_management("WU_Karst1")
```

5.7.51 MarlPEM: Habitat and Ecological Integrity Potential

```
###############################################################################
#
# File Name: MarlPEM.py
# Developer: Yibing Han
# Date 9/20/2017
# Purpose:
#    Input to Habitat / Potential / Vegetation / Floristic Quality
#
###############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcMarlPEM(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HPotential.MarlPEM")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcMarlSoils,"fcMarlSoils")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcENWI")

    # Clean up if needed
    if arcpy.Exists("WU_MarlPEM"):
        arcpy.Delete_management("WU_MarlPEM")
    if arcpy.Exists("WU_MarlPEM1"):
        arcpy.Delete_management("WU_MarlPEM1")
    if arcpy.Exists("VegPEM"):
        arcpy.Delete_management("VegPEM")
    logger.info("feature layers ready")

    ## Select emergent wetlands and export feature class
    strWHERE = """"ATTRIBUTE" LIKE 'PEM%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    arcpy.CopyFeatures_management("fcENWI", "VegPEM")
    arcpy.MakeFeatureLayer_management(r"VegPEM","fcVegPEM")
    logger.info("emergent wetlands selected and exported as VegPEM")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    ## Intersect MarlSoils and emergent wetlands (PEM)
    arcpy.Intersect_analysis(["fcMarlSoils", "fcVegPEM"], "WU_MarlPEM1", "ONLY_FID")
    logger.info("Intersect of MarlSoils and emergent wetland completed")

    ## Add field to store emergent wetland/marl area.
    actions.DeleteField("WU_MarlPEM1","MarlPEMAre")
    arcpy.AddField_management("WU_MarlPEM1", "MarlPEMAre", "FLOAT")

arcpy.CalculateField_management("WU_MarlPEM1","MarlPEMAre","[Shape_Area]","VB","#
")
    arcpy.MakeFeatureLayer_management(r"WU_MarlPEM1","fcWUMarlPEM1")
    logger.info("field MarlPEMAre added to store emergent wetland/marl area")

    ## Spatial Join MarlPEM to Wetland Units and sum MarlPEM area.
```

```
##############################################################################
#################
    # SJ: FloodIn
##############################################################################
#################
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcWU")
    fmSJFLIN.addTable("fcWUMarlPEM1")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","MarlPEMAre"]

    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJFLIN.findFieldMapIndex("MarlPEMAre")
    fieldmap = fmSJFLIN.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJFLIN.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcWUMarlPEM1","WU_MarlPEM","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJFLIN,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_MarlPEM","fcWUMarlPEM")
    logger.info("Spatial Join MarlPEM to Wetland Units completed")

    ## Add fields to store MarlPEMRat and MarlPEM, and set initial value of MarlPEM = 0.
    actions.DeleteField("fcWUMarlPEM","KarstRatio")
    arcpy.AddField_management("fcWUMarlPEM", "MarlPEMRat", "FLOAT")
    arcpy.AddField_management("fcWUMarlPEM", "MarlPEM", "SHORT")
    arcpy.CalculateField_management("fcWUMarlPEM","MarlPEM","0","VB","#")
    logger.info("field MarlPEM added and initial value set to 0")

    ## Calculate the ratio of MarlPEM area to total Wetland Unit area.

arcpy.CalculateField_management("fcWUMarlPEM","MarlPEMRat","[MarlPEMAre]/[Shape_
Area]","VB","#")
    logger.info("field MarlPEMRat calculated")

    ## Assign points to Wetland Units
    strWHERE = """"MarlPEMAre" > 200"""
    arcpy.SelectLayerByAttribute_management("fcWUMarlPEM", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMarlPEM","MarlPEM","1","VB","#")

    strWHERE = """"MarlPEMRat" > 0.5"""
    arcpy.SelectLayerByAttribute_management("fcWUMarlPEM", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMarlPEM","MarlPEM","2","VB","#")

    strWHERE = """"MarlPEMAre" > 10000"""
    arcpy.SelectLayerByAttribute_management("fcWUMarlPEM", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMarlPEM","MarlPEM","3","VB","#")
    logger.info("points assigned to Wetland Units for field MarlPEM")

    arcpy.SelectLayerByAttribute_management("fcWUMarlPEM", "CLEAR_SELECTION")

    # Clean up
```

```
if arcpy.Exists("WU_MarlPEM1"):
    arcpy.Delete_management("WU_MarlPEM1")
if arcpy.Exists("VegPEM"):
    arcpy.Delete_management("VegPEM")
```

5.7.52 SoilIntact: Habitat and Ecological Integrity Potential

```
################################################################################
#
# File Name: SoilIntact.py
# Developer: Yibing Han
# Date 9/28/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential / SoilH
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcSoilIntact():
    logger = logging.getLogger("WFA.HabEco.HPotential.SoilIntact")

    # Setting python variables
    fcDisturb50m = arcpy.mapping.Layer(r"WU_Disturb50m")
    logger.info("feature layers ready")

    # Clean up if needed

    ## Add SoilIntact field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcDisturb50m, 'SoilIntact')
    arcpy.AddField_management(fcDisturb50m, 'SoilIntact', 'SHORT')
    arcpy.CalculateField_management(fcDisturb50m, 'SoilIntact', '0', 'VB', '#')
    logger.info("field SoilIntact added and initial value set to 0")


    ## Assign points to SoilIntact
    strWHERE = """"Buffer50mDist_diss_Dist50mRat" = 0"""
    arcpy.SelectLayerByAttribute_management(fcDisturb50m, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcDisturb50m,"SoilIntact","2","VB","#")


    strWHERE = """"Buffer50mDist_diss_Dist50mRat" > 0"""
    arcpy.SelectLayerByAttribute_management(fcDisturb50m, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcDisturb50m,"SoilIntact","1","VB","#")


    strWHERE = """"Buffer50mDist_diss_Dist50mRat" > 0.5"""
    arcpy.SelectLayerByAttribute_management(fcDisturb50m, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcDisturb50m,"SoilIntact","0","VB","#")
    logger.info("points assigned to SoilIntact")
    arcpy.SelectLayerByAttribute_management(fcDisturb50m, "CLEAR_SELECTION")


    #clean up
```

5.7.53 SoilOrgCalc: Habitat and Ecological Integrity Potential

```
##########################################################################
#
# File Name: SoilOrgCalc.py
# Developer: Yibing Han
# Date 9/28/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential / SoilH
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcSoilOrgCalc():
    logger = logging.getLogger("WFA.HabEco.HPotential.SoilOrgCalc")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Histosol", 'fcHistosol')
    arcpy.MakeFeatureLayer_management(r"WU_Karst", 'fcKarst')

    # Clean up if needed
    if arcpy.Exists('WU_SoilOrgCalc'):
        arcpy.Delete_management('WU_SoilOrgCalc')
    logger.info("feature layers ready")

    ## Spatial Joins to merge fields into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcHistosol')
    fmSJFLIN.addTable('fcKarst')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','Histosol','Karst']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcHistosol', 'fcKarst', 'WU_SoilOrgCalc', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_SoilOrgCalc', 'fcSoilOrgCalc')
    logger.info("Spatial Joins to merge fields into one attribute table completed")

    ## Add SoilOrgCalc field and set initial point value to zero.
    actions.DeleteField('fcSoilOrgCalc', 'SoilOrgCalc')
    arcpy.AddField_management('fcSoilOrgCalc', 'SoilOrgCalc', 'SHORT')
    arcpy.CalculateField_management('fcSoilOrgCalc', 'SoilOrgCalc', '0', 'VB', '#')
    logger.info("SoilOrgCalc field added and initial point value set to zero")

    ## Assign points to fcSoilH
    strWHERE = """("Histosol" + "Karst") > 0"""
    arcpy.SelectLayerByAttribute_management("fcSoilOrgCalc", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcSoilOrgCalc","SoilOrgCalc","1","VB","#")
    logger.info("points assigned to fcSoilH")
```

```
arcpy.SelectLayerByAttribute_management("fcSoilOrgCalc", "CLEAR_SELECTION")


# Clean up
```

5.7.54 StrucPatch: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: StrucPatch.py
# Developer: Yibing Han
# Date 9/29/2017
# Purpose:
#    Input to Habitat & Ecological Integrity / Potential / SoilH
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcStrucPatch():
    logger = logging.getLogger("WFA.HabEco.HPotential.StrucPatch")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Microtopo", 'fcMicrotopo')
    arcpy.MakeFeatureLayer_management(r"WU_StreamEdge", 'fcStreamEdge')
    arcpy.MakeFeatureLayer_management(r"WU_VegVerStr", 'fcVegVerStr')

    # Clean up if needed
    if arcpy.Exists('WU_StrucPatch1'):
        arcpy.Delete_management('WU_StrucPatch1')
    if arcpy.Exists('WU_StrucPatch'):
        arcpy.Delete_management('WU_StrucPatch')
    logger.info("feature layers ready")

    ## Spatial joins to merge input metrics (VegHorInt, VegVerStr, StreamEdge) into one
attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcMicrotopo')
    fmSJFLIN.addTable('fcVegVerStr')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegHorInt','VegVerStr']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcMicrotopo', 'fcVegVerStr', 'WU_StrucPatch1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_StrucPatch1', 'fcStrucPatch1')

    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcStrucPatch1')
    fmSJFLIN.addTable('fcStreamEdge')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegHorInt','VegVerStr','StreamRatio']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcStrucPatch1', 'fcStreamEdge', 'WU_StrucPatch',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_StrucPatch', 'fcStrucPatch')
    logger.info("Spatial Joins to to merge input metrics (VegHorInt, VegVerStr, StreamEdge)
into one attribute table completed")

    ## Add StreamEdge3 field and set initial point value to zero.
    actions.DeleteField('fcStrucPatch', 'StreamEdge3')
    arcpy.AddField_management('fcStrucPatch', 'StreamEdge3', 'SHORT')
    arcpy.CalculateField_management('fcStrucPatch', 'StreamEdge3', '0', 'VB', '#')
    logger.info("StreamEdge3 field added and initial point value set to zero")

    ## Assign points to StreamEdge3
    strWHERE = """"StreamRatio" > 0"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StreamEdge3","1","VB","#")

    strWHERE = """"StreamRatio" > 1.4"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StreamEdge3","2","VB","#")

    strWHERE = """"StreamRatio" > 2.4"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StreamEdge3","3","VB","#")
    logger.info("points assigned to StreamEdge3")

    ## Clear previous selection
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "CLEAR_SELECTION")
    ## Add StrucPatch field and set initial point value to zero.
    actions.DeleteField('fcStrucPatch', 'StrucPatch')
    arcpy.AddField_management('fcStrucPatch', 'StrucPatch', 'SHORT')
    arcpy.CalculateField_management('fcStrucPatch', 'StrucPatch', '0', 'VB', '#')
    logger.info("StrucPatch field added and initial point value set to zero")

    ## Assign points to StrucPatch
    strWHERE = """("VegHorInt" + "VegVerStr" + "StreamEdge3") > 1"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StrucPatch","1","VB","#")

    strWHERE = """("VegHorInt" + "VegVerStr" + "StreamEdge3") > 3"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StrucPatch","2","VB","#")

    strWHERE = """("VegHorInt" + "VegVerStr" + "StreamEdge3") > 5"""
    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcStrucPatch","StrucPatch","3","VB","#")
    logger.info("points assigned to StrucPatch")

    arcpy.SelectLayerByAttribute_management("fcStrucPatch", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists('WU_StrucPatch1'):
        arcpy.Delete_management('WU_StrucPatch1')
```

5.7.55 VegFQ: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: VegFQ.py
# Developer: Yibing Han
# Date 9/25/2017
# Purpose:
#    Input to Habitat and Ecological Integrity / Intrinsic Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegFQ():
    logger = logging.getLogger("WFA.HabEco.HPotential.VegFQ")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r'WU_VegPerUng', 'fcVegPerUng')
    arcpy.MakeFeatureLayer_management(r'WU_VegWoody', 'fcVegWoody')
    arcpy.MakeFeatureLayer_management(r'WU_MarlPEM', 'fcMarlPEM')
    arcpy.MakeFeatureLayer_management(r'WU_Histosol', 'fcHistosol')
    arcpy.MakeFeatureLayer_management(r'WU_Karst', 'fcKarst')
    arcpy.MakeFeatureLayer_management(r'WU_Disturb50m', 'fcDisturb50m')
    arcpy.MakeFeatureLayer_management(r'WU_LandInteg', 'fcLandInteg')
    arcpy.MakeFeatureLayer_management(r'WU_VegAll', 'fcVegAll')

    # Clean up if needed
    if arcpy.Exists('WU_VegFQ'):
        arcpy.Delete_management('WU_VegFQ')
    if arcpy.Exists('WU_VegFQ1'):
        arcpy.Delete_management('WU_VegFQ1')
    if arcpy.Exists('WU_VegFQ2'):
        arcpy.Delete_management('WU_VegFQ2')
    if arcpy.Exists('WU_VegFQ3'):
        arcpy.Delete_management('WU_VegFQ3')
    if arcpy.Exists('WU_VegFQ4'):
        arcpy.Delete_management('WU_VegFQ4')
    if arcpy.Exists('WU_VegFQ5'):
        arcpy.Delete_management('WU_VegFQ5')
    if arcpy.Exists('WU_VegFQ6'):
        arcpy.Delete_management('WU_VegFQ6')
    logger.info("feature layers ready")

    ## Spatial join to merge VegPerUng1 and VegWoodyFor metrics
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegPerUng')
    fmSJFLIN.addTable('fcVegWoody')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcVegPerUng', 'fcVegWoody', 'WU_VegFQ1',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ1', 'fcVegFQ1')

    ## Spatial join to merge MarlPEM metric
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ1')
    fmSJFLIN.addTable('fcMarlPEM')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegFQ1', 'fcMarlPEM', 'WU_VegFQ2',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ2', 'fcVegFQ2')
    logger.info("Spatial join to merge MarlPEM metric completed")

    ## Spatial join to merge Histosol metric
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ2')
    fmSJFLIN.addTable('fcHistosol')
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM',"Histosol"]
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))
    arcpy.SpatialJoin_analysis('fcVegFQ2', 'fcHistosol', 'WU_VegFQ3', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ3', 'fcVegFQ3')
    logger.info("Spatial join to merge Histosol metric completed")

    ## Spatial join to merge Karst metric
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ3')
    fmSJFLIN.addTable('fcKarst')
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM',"Histosol","
Karst"]
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegFQ3', 'fcKarst', 'WU_VegFQ4', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ4', 'fcVegFQ4')
    logger.info("Spatial join to merge Karst metric completed")

    ## Spatial join to merge Dist50mRat
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ4')
    fmSJFLIN.addTable('fcDisturb50m')
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM',"Histosol","
Karst","Buffer50mDist_diss_Dist50mRat"]
```

```
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegFQ4', 'fcDisturb50m', 'WU_VegFQ5',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ5', 'fcVegFQ5')
    logger.info("Spatial join to merge Dist50mRat completed")

    ## Spatial join to merge LandInteg metric
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ5')
    fmSJFLIN.addTable('fcLandInteg')
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM',"Histosol","
Karst","Buffer50mDist_diss_Dist50mRat","LandInteg"]
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegFQ5', 'fcLandInteg', 'WU_VegFQ6',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ6', 'fcVegFQ6')
    logger.info("Spatial join to merge LandInteg metric completed")

    ## Spatial join to merge VegArea
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable('fcVegFQ6')
    fmSJFLIN.addTable('fcVegAll')
    keepers = []
    keepers =
['WUKey','Shape_Length','Shape_Area','VegPerUng1','VegWoodyFor','MarlPEM',"Histosol","
Karst","Buffer50mDist_diss_Dist50mRat","LandInteg","VegArea"]
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcVegFQ6', 'fcVegAll', 'WU_VegFQ', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    arcpy.MakeFeatureLayer_management('WU_VegFQ', 'fcVegFQ')
    logger.info("Spatial join to merge VegArea completed")

    ## Add fields to store Dist50mFQ and VegFQand set initial values to zero
    actions.DeleteField('fcVegFQ', 'VegFQ')
    arcpy.AddField_management('fcVegFQ', 'Dist50mFQ', 'SHORT')
    arcpy.CalculateField_management('fcVegFQ', 'Dist50mFQ', '0', 'VB', '#')
    arcpy.AddField_management('fcVegFQ', 'VegFQ', 'SHORT')
    arcpy.CalculateField_management('fcVegFQ', 'VegFQ', '0', 'VB', '#')
    logger.info("fields added to store Dist50mFQ and VegFQand set initial values to zero")

    ## Assign points to Dist50mFQ
    strWHERE = """"Buffer50mDist_diss_Dist50mRat" < 0.25"""
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcVegFQ","Dist50mFQ","1","VB","#")

    strWHERE = """"Buffer50mDist_diss_Dist50mRat" < 0.1"""
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcVegFQ","Dist50mFQ","2","VB","#")

    strWHERE = """"Buffer50mDist_diss_Dist50mRat" = 0"""
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcVegFQ","Dist50mFQ","3","VB","#")
    logger.info("points assigned to Dist50mFQ")
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "CLEAR_SELECTION")

    ## Sum all points for VegFQ
    arcpy.CalculateField_management('fcVegFQ', 'VegFQ', '[VegPerUng1] + [VegWoodyFor] +
[MarlPEM] + [Histosol] + [Karst] + [Dist50mFQ] + [LandInteg]', 'VB', '#')
    logger.info("points summed for VegFQ")

    ## Reduce VegFQ values to cap of 9 points total
    strWHERE = """"VegFQ" > 9"""
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcVegFQ","VegFQ","9","VB","#")
    logger.info("VegFQ values reduced to cap of 9 points total")

    ## Reduce VegFQ values to zero for unvegetated wetlands
    strWHERE = """"VegArea" IS NULL"""
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management("fcVegFQ","VegFQ","0","VB","#")
    logger.info("VegFQ values reduced to zero for unvegetated wetlands")
    arcpy.SelectLayerByAttribute_management("fcVegFQ", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists('WU_VegFQ1'):
        arcpy.Delete_management('WU_VegFQ1')
    if arcpy.Exists('WU_VegFQ2'):
        arcpy.Delete_management('WU_VegFQ2')
    if arcpy.Exists('WU_VegFQ3'):
        arcpy.Delete_management('WU_VegFQ3')
```

```
if arcpy.Exists('WU_VegFQ4'):
    arcpy.Delete_management('WU_VegFQ4')
if arcpy.Exists('WU_VegFQ5'):
    arcpy.Delete_management('WU_VegFQ5')
if arcpy.Exists('WU_VegFQ6'):
    arcpy.Delete_management('WU_VegFQ6')
```

5.7.56 VegHorInt: Habitat and Ecological Integrity Potential

```
##########################################################################
#
# File Name: VegHorInt.py
# Developer: Yibing Han
# Date 9/20/2017
# Purpose:
#    Input to Habitat / Potential / Vegetation
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegHorInt():
    logger = logging.getLogger("WFA.HabEco.HPotential.VegHorInt")

    # Clean up if needed

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_Microtopo","fcWUMicrotopo")
    logger.info("feature layers ready")

    ## Add new attribute field to store points for VegHorInt and set initial value to zero.
    actions.DeleteField("fcWUMicrotopo","VegHorInt")
    arcpy.AddField_management("fcWUMicrotopo", "VegHorInt", "SHORT")
    arcpy.CalculateField_management("fcWUMicrotopo","VegHorInt","0","VB","#")
    logger.info("field VegHorInt added and initial value set to 0")

    ## Assign points to Wetland Units for VegHorInt
    strWHERE = """"MicroRatio" > 4 AND "Join_Count" > 1"""
    arcpy.SelectLayerByAttribute_management("fcWUMicrotopo", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMicrotopo","VegHorInt","1","VB","#")

    strWHERE = """"MicroRatio" > 6 AND "Join_Count" > 2"""
    arcpy.SelectLayerByAttribute_management("fcWUMicrotopo", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMicrotopo","VegHorInt","2","VB","#")

    strWHERE = """"MicroRatio" > 10 AND "Join_Count" > 4"""
    arcpy.SelectLayerByAttribute_management("fcWUMicrotopo", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUMicrotopo","VegHorInt","3","VB","#")
    logger.info("Points assigned to field VegHorInt")

    arcpy.SelectLayerByAttribute_management("fcWUMicrotopo", "CLEAR_SELECTION")

    # Clean up
```

5.7.57 VegVerStr: Habitat and Ecological Integrity Potential

```
##############################################################################
#
# File Name: VegVerStr.py
# Developer: Yibing Han
# Date 9/18/2017
# Purpose:
#    Input to Habitat / Potential / Vegetation Factor.
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegVerStr():
    logger = logging.getLogger("WFA.HabEco.HPotential.VegVerStr")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_VegWoody","fcVegWoody")
    arcpy.MakeFeatureLayer_management(r"WU_VegAll","fcVegAll")

    # Clean up if needed
    if arcpy.Exists("WU_VegVerStr"):
        arcpy.Delete_management("WU_VegVerStr")
    logger.info("feature layers ready")


    ## Retrieve the fields PFOarea (forest area in m2) and PFOratio (ratio of forest area to total
wetland
    ## area) from the WU_VegWoody feature class.
    ## Retrieve the fields VegArea (vegetated area in m2) and VegRatio (ratio of vegetated area
to total
    ## wetland area) from the WU_VegAll feature class.
    ## Store the fields in a new feature class: WU_VegVerStr.
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable("fcVegAll")
    fmSJFLIN.addTable("fcVegWoody")


    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","PFOarea","PFOratio","VegArea","VegRatio"]


    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))



arcpy.SpatialJoin_analysis("fcVegAll","fcVegWoody","WU_VegVerStr","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJFLIN,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_VegVerStr","fcWUVegVerStr")
    logger.info("spatial join of VegArea and VegAll completed")


    ## Add new attribute field to store points for VegVerStr and set initial value to zero.
    actions.DeleteField("fcWUVegVerStr","VegVerStr")
    arcpy.AddField_management("fcWUVegVerStr", "VegVerStr", "SHORT")
    arcpy.CalculateField_management("fcWUVegVerStr","VegVerStr","0","VB","#")
    logger.info("field VegVerStr added and initial value set to 0")


    ## Assign points to Wetland Units for VegVerStr
    strWHERE = """"VegRatio" > 0.05 AND "VegArea" > 500"""
    arcpy.SelectLayerByAttribute_management("fcWUVegVerStr", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegVerStr","VegVerStr","1","VB","#")


    strWHERE = """"PFOratio" > 0.05 AND "PFOarea" > 500 AND "VegRatio" > 0.5"""
    arcpy.SelectLayerByAttribute_management("fcWUVegVerStr", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegVerStr","VegVerStr","2","VB","#")


    strWHERE = """"PFOratio" > 0.5 AND "PFOarea" > 500"""
    arcpy.SelectLayerByAttribute_management("fcWUVegVerStr", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegVerStr","VegVerStr","3","VB","#")
    logger.info("Points assigned to field VegVerStr")


    arcpy.SelectLayerByAttribute_management("fcWUVegVerStr", "CLEAR_SELECTION")


    # Clean up
```

5.7.58 Habitat and Ecological Integrity Society

```
################################################################################
#
# File Name: HSociety.py
# Developer: Yibing Han
# Date: 12/12/2017
# Purpose:
#        This script handles the execution of all the Habitat and Ecological Integrity Value to
Society metrics.
#
################################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import OwnerAccess, PublicUse
from Factors import HInvest, HUse
from Aspects import HSociety

def procHSociety(WetlandPoly):
logger = logging.getLogger("WFA.HabEco.HSociety")

##############################################################
## 1. Run Variables
##############################################################
OwnerAccess.CalcOwnerAccess(WetlandPoly)
PublicUse.CalcPublicUse(WetlandPoly)

##############################################################
## 2. Run Factors
##############################################################
HInvest.CalcHInvest(WetlandPoly)
HUse.CalcHUse()

##############################################################
## 3. Run Aspect
##############################################################
HSociety.CalcHSociety()
```

5.7.59 Habitat and Ecological Integrity Society Aspects

```
##########################################################################
#
# File Name: HSociety.py
# Developer: Yibing Han
# Date: 10/19/2017
# Purpose:
#    Habitat Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def CalcHSociety():
    logger = logging.getLogger("WFA.HabEco.HSociety.HSociety")

    # Setting python variables
    fcHInvest = arcpy.mapping.Layer(r"WU_HInvest")
    fcHUse = arcpy.mapping.Layer(r"WU_HUse")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_HSociety'):
        arcpy.Delete_management('WU_HSociety')
    ## Spatial join to merge WshdPos and AquaAbund into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable(fcHInvest)
    fmSJFLIN.addTable(fcHUse)
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','HInvest','HUse']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis(fcHInvest, fcHUse, 'WU_HSociety', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcHSociety = arcpy.mapping.Layer(r"WU_HSociety")
    logger.info("spatial join HInvest and HUse completed")

    ## Add HSociety field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcHSociety, 'HSociety')
    arcpy.AddField_management(fcHSociety, 'HSociety', 'SHORT')
    arcpy.CalculateField_management(fcHSociety, 'HSociety', '0', 'VB', '#')
    logger.info("field HSociety added and initial value set to 0")

    ## Sum the points for HInvest and HUse
    arcpy.CalculateField_management(fcHSociety, 'HSociety', '[HInvest] + [HUse]', 'VB', '#')
    logger.info("field HSociety calculated")

    # Clean up
```

5.7.60 Hinvest: Habitat and Ecological Integrity Society

```
##########################################################################
#
# File Name: HInvest.py
# Developer: Yibing Han
# Date: 10/19/2017
# Purpose:
#    Input to Habitat / Value to Society
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHInvest(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HSociety.HInvest")

    # Clean up if needed
    if arcpy.Exists("WU_HInvest"):
        arcpy.Delete_management("WU_HInvest")

    # Setting python variables
    fcLocalPark = arcpy.mapping.Layer(globalvars.srcLocalPark)
    fcNF = arcpy.mapping.Layer(globalvars.srcNF)
    fcNP = arcpy.mapping.Layer(globalvars.srcNP)
    fcNWR = arcpy.mapping.Layer(globalvars.srcNWR)
    fcWMA = arcpy.mapping.Layer(globalvars.srcWMA)
    fcSP = arcpy.mapping.Layer(globalvars.srcSP)
    fcSF = arcpy.mapping.Layer(globalvars.srcSF)
    fcBotanicalAreas = arcpy.mapping.Layer(globalvars.srcBotanicalAreas)
    fcProtectedLands = arcpy.mapping.Layer(globalvars.srcProtectedLands)
    fcNatStrPreAct = arcpy.mapping.Layer(globalvars.srcNatStrPreAct)
    fcILF = arcpy.mapping.Layer(globalvars.srcILF)
    fcRestoredWetlands = arcpy.mapping.Layer(globalvars.srcRestoredWetlands)
    logger.info("feature layers ready")

    ## STEP 1: Create feature class and add field to store results; set initial value to zero
    arcpy.CopyFeatures_management(WetlandPoly,"WU_HInvest","#","0","0","0")
    fcHInvest = arcpy.mapping.Layer(r"WU_HInvest")
    logger.info("feature class WU_HInvest created")

    actions.DeleteField(fcHInvest, "HInvest")
    arcpy.AddField_management(fcHInvest, "HInvest", "SHORT")
    arcpy.CalculateField_management(fcHInvest,"HInvest","0","VB","#")
    logger.info("field HInvest added and initial value set to 0")

    ## Low investment
    ## Select wetlands that intersect state or local public lands
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcSP)
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcWMA, "",
"ADD_TO_SELECTION")
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcSF, "",
"ADD_TO_SELECTION")
```

```
  arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcLocalPark, "",
"ADD_TO_SELECTION")
  logger.info("wetlands that intersect state or local public lands selected")

  ## Add wetlands that intersect Natural Streams Preservation Act watersheds to selection
  arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcNatStrPreAct, "",
"ADD_TO_SELECTION")
  logger.info("wetlands that intersect Natural Streams Preservation Act watersheds added to
selection")

  ## Add wetlands that intersect Department of Defense Lands to selection
  strWHERE = """"OwnName" = 'Department of Defense (DOD)'""""
  arcpy.SelectLayerByAttribute_management(fcProtectedLands, "NEW_SELECTION",
strWHERE)
  arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcProtectedLands, "",
"ADD_TO_SELECTION")
  logger.info("wetlands that intersect Department of Defense Lands added to selection")

  ## Assign point and clear selections
  arcpy.CalculateField_management(fcHInvest,"HInvest","1","VB","#")
  logger.info("1 point assigned to qualifying wetland units")
  arcpy.SelectLayerByAttribute_management(fcHInvest, "CLEAR_SELECTION")

  ## Moderate investment
  ## Select lands owned by USFS
  strWHERE = """"Ownership" = 'Forest Service'""""
  arcpy.SelectLayerByAttribute_management(fcNF, "NEW_SELECTION", strWHERE)
  ## Select wetlands that intersect selected areas
  arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcNF)
  logger.info("wetlands that intersect selected National Forests added to selection")
```

```
    ## Assign points and clear selections
    arcpy.CalculateField_management(fcHInvest,"HInvest","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcHInvest, "CLEAR_SELECTION")

    ## High investment
    ## Select Private Conservation Lands with Gap Status = 1 and USFS Wilderness Areas
    strWHERE = """("Mang_Name" IN ('EWPP-FPE', 'WRP', 'CLRLT', 'Potomac Conservancy',
'The Nature Conservancy (TNC)', 'WVLT', 'Forest Legacy') OR "GAP_Sts" = '1' OR "PdesTp"
= 'Wilderness Area') AND ("PdesTp" <> 'Wild and Scenic River' AND "PdesTp" <> 'National
Wildlife Refuge')"""
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "NEW_SELECTION",
strWHERE)
    ## Select wetlands that intersect selected areas
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcProtectedLands)
    logger.info("wetlands that intersect selected Private Conservation Lands added to selection")

    ## Add to Selection wetlands that intersect National Wildlife Refuges or National Parks
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcNWR, "",
"ADD_TO_SELECTION")
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcNP, "",
"ADD_TO_SELECTION")
    logger.info("wetlands that intersect National Parks and Wildlife Refuges added to selection")

    ## Add to Selection wetlands that intersect special botanical areas
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcBotanicalAreas, "",
"ADD_TO_SELECTION")
    logger.info("wetlands that intersect special botanical areas added to selection")

    ## Add to Selection wetlands that intersect Mitigation Banks and In-Lieu Fee sites
    arcpy.SelectLayerByLocation_management(fcHInvest, "WITHIN_A_DISTANCE", fcILF,
"100 Meters", "ADD_TO_SELECTION")
```

```
    logger.info("wetlands that intersect Mitigation Banks and In-Lieu Fee sites added to
selection")


    ## Add to Selection wetlands that intersect other restored, enhanced or created wetland sites
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcRestoredWetlands,
"", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect other restored, enhanced or created wetland sites added to
selection")


    ## Add to Selection wetlands in WVDNR State Natural Areas
    strWHERE = """"Unit_Nm" IN ('Canaan Valley Resort State Park', 'Cathedral State Park',
'Beartown State Park')"""
    arcpy.SelectLayerByAttribute_management(fcSP, "NEW_SELECTION", strWHERE)
    ## Select wetlands that intersect selected areas
    arcpy.SelectLayerByLocation_management(fcHInvest, "INTERSECT", fcSP, "",
"ADD_TO_SELECTION")
    logger.info("wetlands in WVDNR State Natural Areas added to selection")


    ## Assign points and clear selections
    arcpy.CalculateField_management(fcHInvest,"HInvest","3","VB","#")
    logger.info("3 points assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcHInvest, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcNF, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "CLEAR_SELECTION")

    # Clean up
```

5.7.61 HUse: Habitat and Ecological Integrity Society

```
##############################################################################
#
# File Name: HUse.py
# Developer: Yibing Han
# Date: 10/18/2017
# Purpose:
#    Habitat Function / Value to Society
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcHUse():
    logger = logging.getLogger("WFA.HabEco.HSociety.HUse")

    # Setting python variables
    fcOwnerAccess = arcpy.mapping.Layer(r"WU_OwnerAccess")
    fcPublicUse = arcpy.mapping.Layer(r"WU_PublicUse")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_HUse'):
        arcpy.Delete_management('WU_HUse')

    ## Spatial join to merge WshdPos and AquaAbund into one attribute table
    fmSJFLIN = arcpy.FieldMappings()
    fmSJFLIN.addTable(fcOwnerAccess)
    fmSJFLIN.addTable(fcPublicUse)
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','OwnerAccess','PublicUse']
    for field in fmSJFLIN.fields:
        if field.name not in keepers:
            fmSJFLIN.removeFieldMap(fmSJFLIN.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis(fcOwnerAccess, fcPublicUse, 'WU_HUse',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJFLIN, 'CONTAINS')
    fcHUse = arcpy.mapping.Layer(r"WU_HUse")
    logger.info("spatial join PublicUse and OwnerAccess completed")

    ## Add HUse field to Wetland Units and set initial point value to zero.
    actions.DeleteField(fcHUse, 'HUse')
    arcpy.AddField_management(fcHUse, 'HUse', 'SHORT')
    arcpy.CalculateField_management(fcHUse, 'HUse', '0', 'VB', '#')
    logger.info("field HUse added and initial value set to 0")

    ## Sum the points for OwnerAccess and PublicUse
    arcpy.CalculateField_management(fcHUse, 'HUse', '[OwnerAccess] + [PublicUse]', 'VB', '#')
    logger.info("field HUse calculated")
```

```
# Clean up
```

5.7.62 OwnerAccess: Habitat and Ecological Integrity Society

```
##############################################################################
#
# File Name: OwnerAccess.py
# Developer: Yibing Han
# Date: 10/17/2017
# Purpose:
#    Input to Habitat / Value to Society / HUse
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcOwnerAccess(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HSociety.OwnerAccess")

    # Clean up if needed
    if arcpy.Exists("WU_OwnerAccess"):
        arcpy.Delete_management("WU_OwnerAccess")

    # Setting python variables
    fcLocalPark = arcpy.mapping.Layer(globalvars.srcLocalPark)
    fcNF = arcpy.mapping.Layer(globalvars.srcNF)
    fcNP = arcpy.mapping.Layer(globalvars.srcNP)
    fcNWR = arcpy.mapping.Layer(globalvars.srcNWR)
    fcDNRLands = arcpy.mapping.Layer(globalvars.srcWMA)
    fcSP = arcpy.mapping.Layer(globalvars.srcSP)
    fcSF = arcpy.mapping.Layer(globalvars.srcSF)
    fcProtectedLands = arcpy.mapping.Layer(globalvars.srcProtectedLands)
    fcILF = arcpy.mapping.Layer(globalvars.srcILF)
    fcInfrastructure = arcpy.mapping.Layer(globalvars.srcInfrastructure)
    logger.info("feature layers ready")

    ## STEP 1: Create feature class and add field to store results; set initial value to zero
    arcpy.CopyFeatures_management(WetlandPoly,"WU_OwnerAccess","#","0","0","0")
    fcOwnerAccess = arcpy.mapping.Layer(r"WU_OwnerAccess")
    logger.info("feature class WU_OwnerAccess created")

    actions.DeleteField(fcOwnerAccess, "OwnerAccess")
    arcpy.AddField_management(fcOwnerAccess, "OwnerAccess", "SHORT")
    arcpy.CalculateField_management(fcOwnerAccess,"OwnerAccess","0","VB","#")
    logger.info("field OwnerAccess added and initial value set to 0")

    ## STEP 2: Private lands with seasonal, partial, or case-by-case public access
    ## Select wetlands that intersect partial-access Protected Lands
    strWHERE = """"P_Des_Nm" IN ('Harewood (Washington)', 'Ice Mountain (Riverbirch
Inc.)', 'Upper Shavers Fork')"""
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "NEW_SELECTION",
strWHERE)

    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT",
fcProtectedLands)
    logger.info("wetlands that intersect partial-access Protected Lands selected")
    ## Add wetlands that intersect InfrastructureWetlands, all of which have at least partial
access
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT",
fcInfrastructure, "", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect InfrastructureWetlands added")

    ## Add wetlands that intersect partial-access RIBITS and ILF sites
    strWHERE = """"Project_Name" = 'Tygart Valley'"""
    arcpy.SelectLayerByAttribute_management(fcILF, "NEW_SELECTION", strWHERE)

    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "WITHIN_A_DISTANCE",
fcILF, "100 Meters", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect partial-access RIBITS and ILF sites added")

    ## Assign point and clear selections
    arcpy.CalculateField_management(fcOwnerAccess,"OwnerAccess","1","VB","#")
    logger.info("1 point assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcOwnerAccess, "CLEAR_SELECTION")

    ## STEP 3: Public Ownership
    ## Select wetlands that intersect state or local public lands
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcSP)
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcDNRLands,
"", "ADD_TO_SELECTION")
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcSF, "",
"ADD_TO_SELECTION")
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcLocalPark,
"", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect state or local public lands selected")
    ## Add National Parks and Wildlife Refuges to selection
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcNP, "", "ADD_TO_SELECTION")
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcNWR, "", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect National Parks and Wildlife Refuges added to selection")

    ## Add U.S. Army Corps of Engineers lands to selection
    strWHERE = """"Mang_Name" = 'US Army Corps of Engineers'"""
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "NEW_SELECTION", strWHERE)
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcProtectedLands, "", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect U.S. Army Corps of Engineers lands added to selection")

    ## Add National Forests to selection
    strWHERE = """"Ownership" = 'Forest Service'"""
    arcpy.SelectLayerByAttribute_management(fcNF, "NEW_SELECTION", strWHERE)
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT", fcNF, "", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect National Forests added to selection")

    ## Assign points and clear selections
    arcpy.CalculateField_management(fcOwnerAccess,"OwnerAccess","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcOwnerAccess, "CLEAR_SELECTION")

    ## STEP 4: Select private lands with permanent public access
    strWHERE = """"P_Des_Nm" IN ('Brush Creek (McPherson/Robertson)', 'Bear Rocks Lake Wildlife Management Area', 'Brooklyn Heights (Hills)', 'Cranesville Swamp Preserve', 'Eidolon Nature Preserve', 'Greenland Gap (Amendment)(Greenland Lodge Inc)', 'Hungry Beech', 'Murphy Preserve', 'Pike Knob', 'Pike Knob (Smith)', 'Slaty Mountain (Westvaco)', 'Yankauer
Nature Preserve', 'Canaan Valley/Dolly Sods (Moshein)', 'Core Arboretum' ) OR "Comments" =
'Stauffer''s Marsh (PVAS)'"""
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "NEW_SELECTION",
strWHERE)
    logger.info("private lands with permanent public access selected")

    ## Select wetlands that intersect selected areas
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT",
fcProtectedLands)

    ## Select open-access wetlands from the InfrastructureWetlands feature class
    strWHERE = """"Access" = 'public'"""
    arcpy.SelectLayerByAttribute_management(fcInfrastructure, "NEW_SELECTION",
strWHERE)
    logger.info("open-access wetlands selected")

    ## Select wetlands that intersect selected areas
    arcpy.SelectLayerByLocation_management(fcOwnerAccess, "INTERSECT",
fcInfrastructure, "", "ADD_TO_SELECTION")
    logger.info("wetlands that intersect selected areas added to selection")

    ## Assign point and clear selections
    arcpy.CalculateField_management(fcOwnerAccess,"OwnerAccess","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcOwnerAccess, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcProtectedLands, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcILF, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcNF, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcInfrastructure, "CLEAR_SELECTION")

    # Clean up
```

5.7.63 PublicUse: Habitat and Ecological Integrity Society

```
##############################################################################
#
# File Name: PublicUse.py
# Developer: Yibing Han
# Date: 10/17/2017
# Purpose:
#    Input to Habitat / Value to Society / HUse
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcPublicUse(WetlandPoly):
    logger = logging.getLogger("WFA.HabEco.HSociety.PublicUse")

    # Clean up if needed
    if arcpy.Exists("WU_PublicUse"):
        arcpy.Delete_management("WU_PublicUse")

    # Setting python variables
    fcLocalPark = arcpy.mapping.Layer(globalvars.srcLocalPark)
    fcNWR = arcpy.mapping.Layer(globalvars.srcNWR)
    fcWMA = arcpy.mapping.Layer(globalvars.srcWMA)
    fcSP = arcpy.mapping.Layer(globalvars.srcSP)
    fcFishAccess = arcpy.mapping.Layer(globalvars.srcFishAccess)
    fcBotanicalAreas = arcpy.mapping.Layer(globalvars.srcBotanicalAreas)
    fcPropertyBoundary = arcpy.mapping.Layer(globalvars.srcPropertyBoundary)
    fcEBird = arcpy.mapping.Layer(globalvars.srcEBird)
    fcTrails = arcpy.mapping.Layer(globalvars.srcTrails)
    fcExemp_Branked = arcpy.mapping.Layer(globalvars.srcExempBranked)
    fcInfrastructure = arcpy.mapping.Layer(globalvars.srcInfrastructure)
    logger.info("feature layers ready")

    ## STEP 1: Create feature class and add field to store results; set initial value to zero
    arcpy.CopyFeatures_management(WetlandPoly,"WU_PublicUse","#","0","0","0")
    fcPublicUse = arcpy.mapping.Layer(r"WU_PublicUse")
    logger.info("feature class WU_PublicUse created")

    actions.DeleteField(fcPublicUse, "PublicUse")
    arcpy.AddField_management(fcPublicUse, "PublicUse", "SHORT")
    arcpy.CalculateField_management(fcPublicUse,"PublicUse","0","VB","#")
    logger.info("field PublicUse added and initial value set to 0")

    ## STEP 2: Moderate public use
    ## Select WMAs and State Forests identified as hunting/trapping areas for wetland species
    strWHERE = """"hWaterfowl" =  1 OR "hGrouse" =  1 OR "hWoodcock" =  1 OR "tBeaver"
=  1 OR "tMink" =  1 OR "tMuskrat" =  1 OR "hDeer" =  1 OR "hBear" =  1 OR "hRabbit" =  1
OR "tBobcat" =  1 OR "tCoyote" =  1 OR "tRedFox" =  1 OR "tRaccoon" =  1 OR "tOpossum"
=  1"""
    arcpy.SelectLayerByAttribute_management(fcPropertyBoundary, "NEW_SELECTION",
strWHERE)
```

```
    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT",
fcPropertyBoundary)
    logger.info("WMAs and State Forests identified as hunting/trapping areas for wetland species
selected")

    ## Assign point and clear selections
    arcpy.CalculateField_management(fcPublicUse,"PublicUse","1","VB","#")
    logger.info("1 point assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcPublicUse, "CLEAR_SELECTION")

    ## STEP 3: High public use
    ## Select wetlands within 10 m of a mapped trail or a public fishing access point
    arcpy.SelectLayerByLocation_management(fcPublicUse, "WITHIN_A_DISTANCE",
fcTrails, "10 Meters", "NEW_SELECTION")
    arcpy.SelectLayerByLocation_management(fcPublicUse, "WITHIN_A_DISTANCE",
fcFishAccess, "10 Meters", "ADD_TO_SELECTION")
    logger.info("wetlands within 10 m of a mapped trail or a public fishing access point
selected")

    ## Add to Selection wetlands in National Wildlife Refuges
    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcNWR, "",
"ADD_TO_SELECTION")
    logger.info("wetlands that intersect National Parks and Wildlife Refuges added to selection")

    ## Add to Selection wetlands in special botanical areas supporting long-term research
    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcBotanicalAreas,
"", "ADD_TO_SELECTION")
    logger.info("wetlands in special botanical areas supporting long-term research added to
selection")

    ## Add to Selection wetlands in certain state parks
    strWHERE = """"Unit_Nm" IN ('Blackwater Falls State Park', 'Canaan Valley Resort State
Park', 'Cathedral State Park')"""
    arcpy.SelectLayerByAttribute_management(fcSP, "NEW_SELECTION", strWHERE)

    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcSP, "",
"ADD_TO_SELECTION")
    logger.info("wetlands in certain state parks added to selection")

    ## Add to Selection wetlands in certain local parks
    strWHERE = """"Unit_Nm" IN ('WV Botanic Garden', 'Meadowood Park', 'McDonough
Wildlife Refuge ', 'Johnson T. Janes Nature Preserve and Conservation Park')"""
    arcpy.SelectLayerByAttribute_management(fcLocalPark, "NEW_SELECTION",
strWHERE)

    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcLocalPark, "",
"ADD_TO_SELECTION")
    logger.info("wetlands in certain local parks added to selection")

    ## Add to Selection wetlands in certain WMAs
    strWHERE = """"Unit_Nm" IN ('Fairfax Pond / Rehe Wildlife Management Area', 'Green
Bottom Wildlife Management Area', 'Little Canaan Wildlife Management Area', 'McClintic
Wildlife Management Area', 'Meadow River Wildlife Management Area', 'Pleasant Creek
Wildlife Management Area', 'Short Mountain Wildlife Management Area', 'Valley Bend
Wetlands Wildlife Management Area')"""
    arcpy.SelectLayerByAttribute_management(fcWMA, "NEW_SELECTION", strWHERE)

    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcWMA, "",
"ADD_TO_SELECTION")
    logger.info("wetlands in certain WMAs added to selection")

    ## Add to Selection certain Exemplary Wetlands not already selected
    strWHERE = """"Name" IN ('Alder Run Bog', 'Altona-Piedmont Marsh', 'Bear Rocks Bog', 
'Cranesville Swamp', 'Harewood Marsh', 'Spruce Knob Lake inlet', 'Spruce Knob Lake outlet', 
'Winfield Swamp')"""
    arcpy.SelectLayerByAttribute_management(fcExemp_Branked, "NEW_SELECTION", 
strWHERE)


    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", 
fcExemp_Branked, "", "ADD_TO_SELECTION")
    logger.info("wetlands in certain WMAs added to selection")


    ## Add to Selection wetlands from the InfrastructureWetlands feature class
    arcpy.SelectLayerByLocation_management(fcPublicUse, "INTERSECT", fcInfrastructure, 
"", "ADD_TO_SELECTION")
    logger.info("wetlands in certain Infrastructure Wetlands added to selection")


    ## Add to Selection wetlands within 100 meters of an eBird birding hotspot
    arcpy.SelectLayerByLocation_management(fcPublicUse, "WITHIN_A_DISTANCE", 
fcEBird, "100 Meters", "ADD_TO_SELECTION")
    logger.info("wetlands within 100 meters of an eBird birding hotspot added to selection")


    ## Assign points and clear selections
    arcpy.CalculateField_management(fcPublicUse,"PublicUse","2","VB","#")
    logger.info("2 points assigned to qualifying wetland units")
    arcpy.SelectLayerByAttribute_management(fcPublicUse, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcPropertyBoundary, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcSP, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcLocalPark, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWMA, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcExemp_Branked, "CLEAR_SELECTION")


    # Clean up
```

5.7.64 Actions: Utilities

```
#############################################################################
#
# File Name: actions.py
# Developer: Chad Ashworth
# Date 7/8/2015
# Purpose:
#	This script is a module that contains functions that perform
#	actions against python object (delete/update/etc)
#
#############################################################################
#!/usr/bin/python
import arcpy, sys
import sqlite3 as lite

def CreateGeoDB(gdbName):
strFullPath = ""
#strWorkspace = GetGISData("Workspace")
strWorkspace =
"M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WetlandFunctionResults\\test"
arcpy.CreateFileGDB_management(strWorkspace, gdbName)

strFullPath = strWorkspace + "\\" + gdbName
return strFullPath
'''
def GetGISData(lyrName):
con = None
data = ""
strLayerLoc = ""
```

```
try:
            con = lite.connect(r"db\wetlandFA.db")
            cur = con.cursor()
            strSelect = "select d_location from dataSource where d_name = '" + lyrName + "'"
            cur.execute(strSelect)
            data = cur.fetchone()
            strLayerLoc = data[0]

except lite.Error, e:


    print "Error %s:" % e.args[0]
    sys.exit(1)

finally:


    if con:
            con.close()
return str(strLayerLoc)'''

def CheckFieldExists(table,field):
lstFields = arcpy.ListFields(table,field)
fieldCount = len(lstFields)
if fieldCount == 1:
            retVal = True
elif fieldCount == 0:
            retVal = False

return retVal
```

```
def DeleteField(table,field):
retVal = ""
blnFieldFind = CheckFieldExists(table,field)

#commented out screen prints so code will work with ArcMap Toolbox code
#if blnFieldFind == True:
#           retVal = raw_input("\nThe " + str(field) + " field in the " + str(table) + " table already
exists.\nDo you wish to delete the field(Y/N): ")

if retVal.upper() == "Y":
            blnFieldFind = False
elif retVal.upper() == "N":
            #print "field " + str(field) + " did not exist"
            exit()

if blnFieldFind == True:
            arcpy.DeleteField_management(table, field)


def UpdateTable(table,fields,ufield,value,where):
with arcpy.da.UpdateCursor(table,fields,where) as cUT:
            for rUT in cUT:
                        rUT[ufield] = value
                        cUT.updateRow(rUT)
```

5.7.65 ActiveRiverArea: Utilities

```
##############################################################################
#
# File Name: ActiveRiverArea.py
# Developer: Chad Ashworth
# Date 5/1/2015
# Purpose:
#	This script determines if a wetland polygon is in an active river area.
#
##############################################################################
#!/usr/bin/python
import arcpy
from utilities import actions


def WetlandInARA(WetlandPoly):
# setting the work environment
arcpy.CheckOutExtension("Spatial")
lyrARA =
arcpy.mapping.Layer("M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\ActiveRiver
Area_gdb\\ActiveRiverArea.gdb\\ara_wv_514")
strARA_WV_514_NMC_POLY_SHP =
"M:\\wr\WTRSHD_BRANCH_INTERNAL\\WETLAND\\" + str(WetlandPoly) +
"_ARA_TEMP.shp"


#clean up if needed
strRefDelLyr = str(WetlandPoly) + "_DevLyr"
if arcpy.Exists(strRefDelLyr):
	arcpy.Delete_management(strRefDelLyr)
strRefARA = str(WetlandPoly) + "_ARA"
if arcpy.Exists(strRefARA):
	arcpy.Delete_management(strRefARA)
```

```
strRefWetIsecARA = str(WetlandPoly) + "_WetIsecARA"
if arcpy.Exists(strRefWetIsecARA):
        arcpy.Delete_management(strRefWetIsecARA)
if arcpy.Exists(strARA_WV_514_NMC_POLY_SHP):
        arcpy.Delete_management(strARA_WV_514_NMC_POLY_SHP)

# adding/deleting the Floodplain_FEMA
actions.DeleteField(WetlandPoly,"Floodplain_ARA")
arcpy.AddField_management(WetlandPoly, "Floodplain_ARA", "TEXT",1)

################################################################################
###############

# 1.
sql = "DESC_SHORT NOT LIKE 'Material Contribution Zone%'"
outExtractByAtrb = arcpy.sa.ExtractByAttributes(lyrARA,sql)
outExtractByAtrb.save(strRefARA)

# 2.
arcpy.RasterToPolygon_conversion(strRefARA, strARA_WV_514_NMC_POLY_SHP,
"NO_SIMPLIFY")

# 3.
arcpy.Intersect_analysis([[WetlandPoly], strARA_WV_514_NMC_POLY_SHP],
strRefWetIsecARA,"ALL")

arcpy.MakeFeatureLayer_management(strRefWetIsecARA, strRefDelLyr)
strWetlandPolyPK = "FID_" + str(WetlandPoly) # dynamically createing the foreign key field
to the WetlandPoly
arcpy.AddJoin_management(strRefDelLyr, strWetlandPolyPK, str(WetlandPoly),
"OBJECTID") # join to WetlandPoly
```

```
#arcpy.CopyFeatures_management(strRefDelLyr, strRefDelLyr)



# declaring what fields i need to calculate the percent intersect
## EDIT TO DYNAMICALLY CREATE FIELD NAMES
strWetlandId = str(WetlandPoly) + ".OBJECTID"
strWetlandShapeArea = str(WetlandPoly) + ".Shape_Area"
strWetlandFlARA = str(WetlandPoly) + ".Floodplain_ARA"
fldsDL = [strWetlandId,strRefWetIsecARA +
".Shape_Area",strWetlandShapeArea,strWetlandFlARA]

##############################################################################
#######################################################
#arcpy.CopyFeatures_management("development_layer", "development_layer")

edit = arcpy.da.Editor(arcpy.env.workspace)
edit.startEditing(False, True)
edit.startOperation()

# updating the UA field with a Y if 10% or more are in an urban area
with arcpy.da.SearchCursor(strRefDelLyr, fldsDL) as curWetlandAll:
	for rWetlandAll in curWetlandAll:
		intPercentIntersect = int(rWetlandAll[1]/rWetlandAll[2]*100)
		if intPercentIntersect >= 10:
			strWhere = "OBJECTID = " + str(rWetlandAll[0])
			actions.UpdateTable(WetlandPoly,["Floodplain_ARA"],0,"Y",strWhere)

# updating the rest of the fields to N
strWhere = "Floodplain_ARA is null"
actions.UpdateTable(WetlandPoly,["Floodplain_ARA"],0,"N",strWhere)
```

```
####################################################
# ADD CODE TO SELECT NOTHING
####################################################
edit.stopEditing(True)

arcpy.Delete_management(strRefWetIsecARA)
arcpy.Delete_management(strRefDelLyr)
arcpy.Delete_management(strRefARA)
```

5.7.66 AllResults: Utilities

```
##############################################################################
#
# File Name: AllResults.py
# Developer: Yibing Han
# Date: 02/25/2018
# Purpose:
#    This script calculates the total assessment scores of wetland units
#
#
###############################################################################
#!/usr/bin/python
import logging
import datetime
import arcpy
import actions
from globalvars import globalvars

def CalcAllResults(WetlandPoly):
    logger = logging.getLogger("WFA.AllResults")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(r"WU_Function", "fcWUFunction")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcENWI")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_AllResults1'):
        arcpy.Delete_management('WU_AllResults1')
    if arcpy.Exists('WU_AllResults'):
        arcpy.Delete_management('WU_AllResults')
    if arcpy.Exists('AllResultsTable'):
        arcpy.Delete_management('AllResultsTable')

    ## STEP 1: Add the identifier fields (link to original input polygons) to Wetland units scoring
fields
    fmSJAll0 = arcpy.FieldMappings()
    fmSJAll0.addTable("fcENWI")
    fmSJAll0.addTable("fcWU")
    keepers = []
    keepers = ['SiteCode','WetlandName','SurveyDate',"WUKey"]
    for field in fmSJAll0.fields:
        if field.name not in keepers:
            fmSJAll0.removeFieldMap(fmSJAll0.findFieldMapIndex(field.name))

arcpy.SpatialJoin_analysis("fcWU","fcENWI","WU_AllResults1","JOIN_ONE_TO_ONE","K
EEP_ALL",fmSJAll0,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WU_AllResults1","fcWUAllResults1")
    logger.info("identifier fields joined to WU_Function")

    ## STEP 2: Add the scoring fields
    # Make Query Table to join all scoring fields
    #fc_not_needed = [WetlandPoly,
"Buffer10m","Buffer1km","Buffer300m","Buffer50m","DrainageArea", "WU_BRank",
"WU_FAFunction", "WU_FAOpportun", "WU_FAPotential", "WU_FASociety",
"WU_Function", "WU_HFunction", "WU_HFunctionNoBR", "WU_HOpportun",
"WU_HPotential", "WU_HSociety", "WU_WQFunction", "WU_WQOpportun",
"WU_WQPotential", "WU_WQSociety", "WUPoint"];

#arcpy.MakeQueryTable_management(in_table="WU_AllResults1;WU_AquaAbund;WU_Con
nect;WU_ClayOrganic;WU_Clay;WU_ChemTime;WU_BufferPerim;WU_BufferLand;WU_B
RankHUC;WU_BufferContig;WU_ConsFocus;WU_Depressions;WU_Discharges;WU_Disturb
50m;WU_DisturbWshd;WU_EconRisk;WU_Fisheries;WU_FloodArea;WU_FloodIn;WU_Floo
dway;WU_Headwater;WU_HInvest;WU_Histosol;WU_HUC12WQ;WU_HUse;WU_HydroH;
WU_ImpairedIn;WU_ImpairedOut;WU_IrrEdge;WU_Karst;WU_LandEco;WU_LandHydro;W
U_LandInteg;WU_LandPos;WU_LowSlope;WU_MarlPEM;WU_Microtopo;WU_Organic;WU
_OwnerAccess;WU_PublicUse;WU_RoadRail;WU_Runoff;WU_Runoff50m;WU_RunoffWsh
d;WU_SeasonPond;WU_SLOPE;WU_SlopeWshd;WU_SoilH;WU_SoilOrgCalc;WU_StreamE
dge;WU_StrucPatch;WU_SWOutflow;WU_VegAll;WU_VegByLP;WU_VegFA;WU_VegFQ;
WU_VegH;WU_VegPerUng;WU_VegVerStr;WU_VegWoody;WU_VegWQ;WU_WaterSuppl
y;WU_WetlandBird;WU_WflowPath;WU_WQPlan;WU_WQUse;WU_WshdPos;WU_WshdU
niq", out_table="AllResultsTable", in_key_field_option="NO_KEY_FIELD", in_key_field="",
in_field="WU_AllResults1.WetlandName #;WU_AllResults1.SurveyDate
#;WU_AllResults1.WUKey #;WU_AquaAbund.AquaAbund #;WU_Connect.ConnectFL
#;WU_ClayOrganic.ClayOrganic #;WU_Clay.Clay #;WU_ChemTime.ChemTime
#;WU_BufferPerim.BufferPerim #;WU_BufferLand.BufferLand
#;WU_BRankHUC.BRankHUC #;WU_BufferContig.BufferContig
#;WU_ConsFocus.ConsFocus #;WU_Depressions.Depressions #;WU_Discharges.Discharges
#;WU_Disturb50m.Disturb50m #;WU_Disturb50m.SoilIntact
#;WU_DisturbWshd.DisturbWshd #;WU_EconRisk.EconRisk #;WU_Fisheries.Fisheries
#;WU_FloodArea.FloodArea #;WU_FloodArea.Floodplain #;WU_FloodIn.FloodIn
#;WU_Floodway.Floodway #;WU_Headwater.Headwater #;WU_HInvest.HInvest
#;WU_Histosol.Histosol #;WU_HUC12WQ.HUC12WQ #;WU_HUse.HUse
#;WU_HydroH.HydIntact #;WU_HydroH.HydSW #;WU_HydroH.HydroH
#;WU_ImpairedIn.ImpairedIn #;WU_ImpairedOut.ImpairedOut #;WU_IrrEdge.IrrEdge
#;WU_Karst.Karst #;WU_LandEco.LandEco #;WU_LandHydro.LandHydro
#;WU_LandInteg.LandResil #;WU_LandInteg.LandInteg #;WU_LandPos.LandPos
#;WU_LowSlope.LowSlope #;WU_MarlPEM.MarlPEM #;WU_Microtopo.Microtopo
#;WU_Microtopo.VegHorInt #;WU_Organic.Organic #;WU_OwnerAccess.OwnerAccess
#;WU_PublicUse.PublicUse #;WU_RoadRail.RoadRail #;WU_Runoff.Runoff
#;WU_Runoff50m.Runoff50m #;WU_RunoffWshd.RunoffWshd
#;WU_SeasonPond.SeaPondRatio #;WU_SeasonPond.SeasonPond #;WU_SLOPE.SLOPE
#;WU_SlopeWshd.SlopeWshd #;WU_SoilH.SoilH #;WU_SoilOrgCalc.SoilOrgCalc
#;WU_StreamEdge.StreamEdge #;WU_StrucPatch.StrucPatch #;WU_SWOutflow.SWOutflow
#;WU_SWOutflow.SWOutflow2 #;WU_VegAll.VegAll #;WU_VegByLP.VegByLP
#;WU_VegFA.VegFA #;WU_VegFQ.Dist50mFQ #;WU_VegFQ.VegFQ #;WU_VegH.VegH
#;WU_VegPerUng.VegPerUng #;WU_VegPerUng.VegPerUng4
#;WU_VegPerUng.VegPerUng1 #;WU_VegVerStr.VegVerStr #;WU_VegWoody.VegWoody
#;WU_VegWoody.VegWoody4 #;WU_VegWoody.VegWoodyFor #;WU_VegWQ.VegWQ
#;WU_WaterSupply.WaterSupply #;WU_WetlandBird.WetldBird
#;WU_WflowPath.WFlowPath #;WU_WQPlan.WQPlan #;WU_WQUse.WQUse
```

```
#;WU_WshdPos.WshdPos #;WU_WshdUniq.WshdUniq #",
where_clause="WU_AllResults1.WUKey =
WU_AquaAbund.WU_AquaAbund2_WU_AquaAbund1_WUKey AND
WU_AllResults1.WUKey = WU_BRankHUC.WUKey AND WU_AllResults1.WUKey =
WU_BufferContig.WUKey AND WU_AllResults1.WUKey = WU_BufferLand.WUKey AND
WU_AllResults1.WUKey = WU_BufferPerim.WUKey AND WU_AllResults1.WUKey =
WU_ChemTime.WUKey AND WU_AllResults1.WUKey = WU_Clay.WUKey AND
WU_AllResults1.WUKey = WU_ClayOrganic.WUKey AND WU_AllResults1.WUKey =
WU_Connect.WUKey AND WU_AllResults1.WUKey = WU_ConsFocus.WUKey AND
WU_AllResults1.WUKey = WU_EconRisk.WUKey AND WU_AllResults1.WUKey =
WU_DisturbWshd.WU_DisturbWshd1_WUKey AND WU_AllResults1.WUKey =
WU_Discharges.WUKey AND WU_AllResults1.WUKey =
WU_Disturb50m.WU_Disturb50m1_WUKey AND WU_AllResults1.WUKey =
WU_Depressions.WUKey AND WU_AllResults1.WUKey = WU_Fisheries.WUKey AND
WU_AllResults1.WUKey = WU_FloodArea.WUKey AND WU_AllResults1.WUKey =
WU_FloodIn.WUKey AND WU_AllResults1.WUKey = WU_Floodway.WUKey AND
WU_AllResults1.WUKey = WU_Headwater.WUKey AND WU_AllResults1.WUKey =
WU_HInvest.WUKey AND WU_AllResults1.WUKey = WU_Histosol.WUKey AND
WU_AllResults1.WUKey = WU_HUC12WQ.WUKey AND WU_AllResults1.WUKey =
WU_HUse.WUKey AND WU_AllResults1.WUKey = WU_HydroH.WUKey AND
WU_AllResults1.WUKey = WU_ImpairedIn.WUKey AND WU_AllResults1.WUKey =
WU_ImpairedOut.WUKey AND WU_AllResults1.WUKey = WU_IrrEdge.WUKey AND
WU_AllResults1.WUKey = WU_Karst.WUKey AND WU_AllResults1.WUKey =
WU_LandEco.WUKey AND WU_AllResults1.WUKey = WU_LandHydro.WUKey AND
WU_AllResults1.WUKey = WU_LandInteg.WU_LandResil1_WU_LandInteg0_WUKey AND
WU_AllResults1.WUKey = WU_LandPos.WUKey AND WU_AllResults1.WUKey =
WU_LowSlope.WUKey AND WU_AllResults1.WUKey = WU_MarlPEM.WUKey AND
WU_AllResults1.WUKey = WU_Microtopo.WUKey AND WU_AllResults1.WUKey =
WU_Organic.WUKey AND WU_AllResults1.WUKey = WU_OwnerAccess.WUKey AND
WU_AllResults1.WUKey = WU_PublicUse.WUKey AND WU_AllResults1.WUKey =
WU_RoadRail.WUKey AND WU_AllResults1.WUKey = WU_Runoff.WUKey AND
WU_AllResults1.WUKey = WU_Runoff50m.WU_Runoff50m1_WUKey AND
WU_AllResults1.WUKey = WU_RunoffWshd.WU_RunoffWshd1_WUKey AND
WU_AllResults1.WUKey = WU_VegAll.WUKey AND WU_AllResults1.WUKey =
WU_SWOutflow.WUKey AND WU_AllResults1.WUKey = WU_StrucPatch.WUKey AND
WU_AllResults1.WUKey = WU_StreamEdge.WUKey AND WU_AllResults1.WUKey =
WU_SoilOrgCalc.WUKey AND WU_AllResults1.WUKey = WU_SoilH.WUKey AND
WU_AllResults1.WUKey = WU_SlopeWshd.WU_SlopeWshd1_WUKey AND
WU_AllResults1.WUKey = WU_SLOPE.WUKey AND WU_AllResults1.WUKey =
WU_SeasonPond.WUKey AND WU_AllResults1.WUKey = WU_VegByLP.WUKey AND
WU_AllResults1.WUKey = WU_VegFA.WUKey AND WU_AllResults1.WUKey =
WU_VegFQ.WUKey AND WU_AllResults1.WUKey = WU_VegH.WUKey AND
WU_AllResults1.WUKey = WU_VegPerUng.WU_VegPerUng1_WUKey AND
WU_AllResults1.WUKey = WU_VegVerStr.WUKey AND WU_AllResults1.WUKey =
WU_VegWoody.WUKey AND WU_AllResults1.WUKey = WU_VegWQ.WUKey AND
WU_AllResults1.WUKey = WU_WaterSupply.WUKey AND WU_AllResults1.WUKey =
WU_WetlandBird.WUKey AND WU_AllResults1.WUKey = WU_WflowPath.WUKey AND
WU_AllResults1.WUKey = WU_WQPlan.WUKey AND WU_AllResults1.WUKey =
WU_WQUse.WUKey AND WU_AllResults1.WUKey = WU_WshdPos.WUKey AND
WU_AllResults1.WUKey = WU_WshdUniq.WUKey")

    in_tables =
"WU_AllResults1;WU_AquaAbund;WU_Connect;WU_ClayOrganic;WU_Clay;WU_ChemTi
me;WU_BufferPerim;WU_BufferLand;WU_BRankHUC;WU_BufferContig;WU_ConsFocus;
WU_Depressions;WU_Discharges;WU_Disturb50m;WU_DisturbWshd;WU_EconRisk;WU_Fi
sheries;WU_FloodArea;WU_FloodIn;WU_Floodway;WU_Headwater;WU_HInvest;WU_Histo
sol;WU_HUC12WQ;WU_HUse;WU_HydroH;WU_ImpairedIn;WU_ImpairedOut;WU_IrrEdg
e;WU_Karst;WU_LandEco;WU_LandHydro;WU_LandInteg;WU_LandPos;WU_LowSlope;W
U_MarlPEM;WU_Microtopo;WU_Organic;WU_OwnerAccess;WU_PublicUse;WU_RoadRail;
WU_Runoff;WU_Runoff50m;WU_RunoffWshd;WU_SeasonPond;WU_SLOPE;WU_SlopeWs
hd;WU_SoilH;WU_SoilOrgCalc;WU_StreamEdge;WU_StrucPatch;WU_SWOutflow;WU_Ve
gAll;WU_VegByLP;WU_VegFA;WU_VegFQ;WU_VegH;WU_VegPerUng;WU_VegVerStr;
WU_VegWoody;WU_VegWQ;WU_WaterSupply;WU_WetlandBird;WU_WflowPath;WU_W
QPlan;WU_WQUse;WU_WshdPos;WU_WshdUniq"

    in_fields = "WU_AllResults1.WetlandName #;WU_AllResults1.SurveyDate #;"

    if len(arcpy.ListFields("fcWUAllResults1", "SiteCode")) > 0:

        in_fields += "WU_AllResults1.SiteCode #;"

    in_fields += "WU_AllResults1.WUKey #;WU_AquaAbund.AquaAbund
#;WU_Connect.ConnectFL #;WU_ClayOrganic.ClayOrganic #;WU_Clay.Clay
#;WU_ChemTime.ChemTime #;WU_BufferPerim.BufferPerim #;WU_BufferLand.BufferLand
#;WU_BRankHUC.BRankHUC #;WU_BufferContig.BufferContig
#;WU_ConsFocus.ConsFocus #;WU_Depressions.Depressions #;WU_Discharges.Discharges
#;WU_Disturb50m.Disturb50m #;WU_Disturb50m.SoilIntact
#;WU_DisturbWshd.DisturbWshd #;WU_EconRisk.EconRisk #;WU_Fisheries.Fisheries
#;WU_FloodArea.FloodArea #;WU_FloodArea.Floodplain #;WU_FloodIn.FloodIn
#;WU_Floodway.Floodway #;WU_Headwater.Headwater #;WU_HInvest.HInvest
#;WU_Histosol.Histosol #;WU_HUC12WQ.HUC12WQ #;WU_HUse.HUse
#;WU_HydroH.HydIntact #;WU_HydroH.HydSW #;WU_HydroH.HydroH
#;WU_ImpairedIn.ImpairedIn #;WU_ImpairedOut.ImpairedOut #;WU_IrrEdge.IrrEdge
#;WU_Karst.Karst #;WU_LandEco.LandEco #;WU_LandHydro.LandHydro
#;WU_LandInteg.LandResil #;WU_LandInteg.LandInteg #;WU_LandPos.LandPos
```

```
#;WU_LowSlope.LowSlope #;WU_MarlPEM.MarlPEM #;WU_Microtopo.Microtopo
#;WU_Microtopo.VegHorInt #;WU_Organic.Organic #;WU_OwnerAccess.OwnerAccess
#;WU_PublicUse.PublicUse #;WU_RoadRail.RoadRail #;WU_Runoff.Runoff
#;WU_Runoff50m.Runoff50m #;WU_RunoffWshd.RunoffWshd
#;WU_SeasonPond.SeaPondRatio #;WU_SeasonPond.SeasonPond #;WU_SLOPE.SLOPE
#;WU_SlopeWshd.SlopeWshd #;WU_SoilH.SoilH #;WU_SoilOrgCalc.SoilOrgCalc
#;WU_StreamEdge.StreamEdge #;WU_StrucPatch.StrucPatch #;WU_SWOutflow.SWOutflow
#;WU_SWOutflow.SWOutflow2 #;WU_VegAll.VegAll #;WU_VegByLP.VegByLP
#;WU_VegFA.VegFA #;WU_VegFQ.Dist50mFQ #;WU_VegFQ.VegFQ #;WU_VegH.VegH
#;WU_VegPerUng.VegPerUng #;WU_VegPerUng.VegPerUng4
#;WU_VegPerUng.VegPerUng1 #;WU_VegVerStr.VegVerStr #;WU_VegWoody.VegWoody
#;WU_VegWoody.VegWoody4 #;WU_VegWoody.VegWoodyFor #;WU_VegWQ.VegWQ
#;WU_WaterSupply.WaterSupply #;WU_WetlandBird.WetldBird
#;WU_WflowPath.WFlowPath #;WU_WQPlan.WQPlan #;WU_WQUse.WQUse
#;WU_WshdPos.WshdPos #;WU_WshdUniq.WshdUniq #"

   where_clause = "WU_AllResults1.WUKey =
WU_AquaAbund.WU_AquaAbund2_WU_AquaAbund1_WUKey AND
WU_AllResults1.WUKey = WU_BRankHUC.WUKey AND WU_AllResults1.WUKey =
WU_BufferContig.WUKey AND WU_AllResults1.WUKey = WU_BufferLand.WUKey AND
WU_AllResults1.WUKey = WU_BufferPerim.WUKey AND WU_AllResults1.WUKey =
WU_ChemTime.WUKey AND WU_AllResults1.WUKey = WU_Clay.WUKey AND
WU_AllResults1.WUKey = WU_ClayOrganic.WUKey AND WU_AllResults1.WUKey =
WU_Connect.WUKey AND WU_AllResults1.WUKey = WU_ConsFocus.WUKey AND
WU_AllResults1.WUKey = WU_EconRisk.WUKey AND WU_AllResults1.WUKey =
WU_DisturbWshd.WU_DisturbWshd1_WUKey AND WU_AllResults1.WUKey =
WU_Discharges.WUKey AND WU_AllResults1.WUKey =
WU_Disturb50m.WU_Disturb50m1_WUKey AND WU_AllResults1.WUKey =
WU_Depressions.WUKey AND WU_AllResults1.WUKey = WU_Fisheries.WUKey AND
WU_AllResults1.WUKey = WU_FloodArea.WUKey AND WU_AllResults1.WUKey =
WU_FloodIn.WUKey AND WU_AllResults1.WUKey = WU_Floodway.WUKey AND
WU_AllResults1.WUKey = WU_Headwater.WUKey AND WU_AllResults1.WUKey =
WU_HInvest.WUKey AND WU_AllResults1.WUKey = WU_Histosol.WUKey AND
WU_AllResults1.WUKey = WU_HUC12WQ.WUKey AND WU_AllResults1.WUKey =
WU_HUse.WUKey AND WU_AllResults1.WUKey = WU_HydroH.WUKey AND
WU_AllResults1.WUKey = WU_ImpairedIn.WUKey AND WU_AllResults1.WUKey =
WU_ImpairedOut.WUKey AND WU_AllResults1.WUKey = WU_IrrEdge.WUKey AND
WU_AllResults1.WUKey = WU_Karst.WUKey AND WU_AllResults1.WUKey =
WU_LandEco.WUKey AND WU_AllResults1.WUKey = WU_LandHydro.WUKey AND
WU_AllResults1.WUKey = WU_LandInteg.WU_LandResil1_WU_LandInteg0_WUKey AND
WU_AllResults1.WUKey = WU_LandPos.WUKey AND WU_AllResults1.WUKey =
WU_LowSlope.WUKey AND WU_AllResults1.WUKey = WU_MarlPEM.WUKey AND
```

```
WU_AllResults1.WUKey = WU_Microtopo.WUKey AND WU_AllResults1.WUKey =
WU_Organic.WUKey AND WU_AllResults1.WUKey = WU_OwnerAccess.WUKey AND
WU_AllResults1.WUKey = WU_PublicUse.WUKey AND WU_AllResults1.WUKey =
WU_RoadRail.WUKey AND WU_AllResults1.WUKey = WU_Runoff.WUKey AND
WU_AllResults1.WUKey = WU_Runoff50m.WU_Runoff50m1_WUKey AND
WU_AllResults1.WUKey = WU_RunoffWshd.WU_RunoffWshd1_WUKey AND
WU_AllResults1.WUKey = WU_VegAll.WUKey AND WU_AllResults1.WUKey =
WU_SWOutflow.WUKey AND WU_AllResults1.WUKey = WU_StrucPatch.WUKey AND
WU_AllResults1.WUKey = WU_StreamEdge.WUKey AND WU_AllResults1.WUKey =
WU_SoilOrgCalc.WUKey AND WU_AllResults1.WUKey = WU_SoilH.WUKey AND
WU_AllResults1.WUKey = WU_SlopeWshd.WU_SlopeWshd1_WUKey AND
WU_AllResults1.WUKey = WU_SLOPE.WU_SLOPE1_WUKey AND
WU_AllResults1.WUKey = WU_SeasonPond.WUKey AND WU_AllResults1.WUKey =
WU_VegByLP.WUKey AND WU_AllResults1.WUKey = WU_VegFA.WUKey AND
WU_AllResults1.WUKey = WU_VegFQ.WUKey AND WU_AllResults1.WUKey =
WU_VegH.WUKey AND WU_AllResults1.WUKey =
WU_VegPerUng.WU_VegPerUng1_WUKey AND WU_AllResults1.WUKey =
WU_VegVerStr.WUKey AND WU_AllResults1.WUKey = WU_VegWoody.WUKey AND
WU_AllResults1.WUKey = WU_VegWQ.WUKey AND WU_AllResults1.WUKey =
WU_WaterSupply.WUKey AND WU_AllResults1.WUKey = WU_WetlandBird.WUKey AND
WU_AllResults1.WUKey = WU_WflowPath.WUKey AND WU_AllResults1.WUKey =
WU_WQPlan.WUKey AND WU_AllResults1.WUKey = WU_WQUse.WUKey AND
WU_AllResults1.WUKey = WU_WshdPos.WUKey AND WU_AllResults1.WUKey =
WU_WshdUniq.WUKey"

    arcpy.MakeQueryTable_management(in_tables, "AllResultsTable", "NO_KEY_FIELD", "",
in_fields, where_clause)

    arcpy.TableToGeodatabase_conversion("AllResultsTable",arcpy.env.workspace)

    arcpy.MakeTableView_management(arcpy.env.workspace +
"\\AllResultsTable","tvAllResults")


    ## STEP 3: Join output table to WU_Function


arcpy.AddJoin_management("fcWUFunction","WUKey","tvAllResults","WU_AllResults1_W
UKey")

    arcpy.FeatureClassToFeatureClass_conversion("fcWUFunction", arcpy.env.workspace,
"WU_AllResults")

    arcpy.RemoveJoin_management("fcWUFunction")
    arcpy.MakeFeatureLayer_management("WU_AllResults", "fcWUAllResults")
    logger.info("output table joined to WU_Function")


    # Clean and simplify field names
    arcpy.DeleteField_management("fcWUAllResults",["AllResultsTable_OBJECTID",
"AllResultsTable_WU_AllResults1_WUKey"])
    fields = arcpy.ListFields("fcWUAllResults")
    for field in fields:
        if len(field.name.split("_")) == 3:
            newFieldName = field.name.split("_")[2]
            arcpy.AddField_management("fcWUAllResults", newFieldName, field.type)
            arcpy.CalculateField_management("fcWUAllResults", newFieldName, "!" + field.name
+ "!", "PYTHON")
            arcpy.DeleteField_management("fcWUAllResults", field.name)
        else if len(field.name.split("_")) == 4:
            newFieldName = field.name.split("_")[3]
            arcpy.AddField_management("fcWUAllResults", newFieldName, field.type)
            arcpy.CalculateField_management("fcWUAllResults", newFieldName, "!" + field.name
+ "!", "PYTHON")
            arcpy.DeleteField_management("fcWUAllResults", field.name)
    logger.info("field names cleaned and simplified")


    ## STEP 4: Export to personal geodatabase in preperation for Access Database merge
    strOutputName = "AllResults_" + str(datetime.date.today().year) +
'{:02}'.format(datetime.date.today().month) + '{:02}'.format(datetime.date.today().day)
    arcpy.FeatureClassToFeatureClass_conversion("fcWUAllResults",
r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WetlandFunctionResults\test\WVWR
AMGISresults.mdb", strOutputName)


    # Clean up
    if arcpy.Exists('WU_AllResults1'):
```

```
    arcpy.Delete_management('WU_AllResults1')
if arcpy.Exists('AllResultsTable'):
    arcpy.Delete_management('AllResultsTable')
```

5.7.67 CalcFunction: Utilities

```
##############################################################################
#
# File Name: CalcFunction.py
# Developer: Yibing Han
# Date: 02/25/2018
# Purpose:
#    This script calculates the total assessment scores of wetland units
#
#
###############################################################################
#!/usr/bin/python
import logging
import arcpy
import actions

def CalcFunction():
    logger = logging.getLogger("WFA.FinalRun")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_WQFunction","fcWQFunction")
    arcpy.MakeFeatureLayer_management(r"WU_FAFunction","fcFAFunction")
    arcpy.MakeFeatureLayer_management(r"WU_HFunction","fcHFunction")
    logger.info("feature layers ready")

    # Clean up if needed
    if arcpy.Exists('WU_Function1'):
        arcpy.Delete_management('WU_Function1')
    if arcpy.Exists('WU_Function2'):
        arcpy.Delete_management('WU_Function2')
    if arcpy.Exists('WU_Function'):
        arcpy.Delete_management('WU_Function')

    ## STEP 1: Calculate the value of HCondition
    ## Add field HCondition to feature class WU_HFunction and set initial value
    actions.DeleteField("fcHFunction", 'HCondition')
    arcpy.AddField_management("fcHFunction", 'HCondition', 'SHORT')
    arcpy.CalculateField_management("fcHFunction", 'HCondition', '!HPotential! +
!HOpportun!', 'PYTHON', '#')
    logger.info("field HCondition added and initial value calculated")

    ## Select B6 wetlands and re-calculate HCondition
    strWHERE = """"BRank" = 'B6'"""
    arcpy.SelectLayerByAttribute_management("fcHFunction", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcHFunction", 'HCondition', '!HPotB6! + !HOpportun!',
'PYTHON', '#')
    logger.info("B6 wetlands selected and HCondition re-calculated")

    ## Select B5 wetlands and re-calculate HCondition
    strWHERE = """"BRank" = 'B5'"""
    arcpy.SelectLayerByAttribute_management("fcHFunction", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcHFunction", 'HCondition', '!HOpportun! + 30',
'PYTHON', '#')
    logger.info("B5 wetlands selected and HCondition re-calculated")

    ## Select B1-B4 wetlands and re-calculate HCondition
    strWHERE = """"BRank" IN ('B1', 'B2', 'B3', 'B4')"""
    arcpy.SelectLayerByAttribute_management("fcHFunction", "NEW_SELECTION", 
strWHERE)
    arcpy.CalculateField_management("fcHFunction", 'HCondition', '!HFunction!', 'PYTHON', 
'#')
    logger.info("B1-B4 wetlands selected and HCondition re-calculated")
    arcpy.SelectLayerByAttribute_management("fcHFunction", "CLEAR_SELECTION")



    ## STEP 2: Spatial Join to merge metrics and create feature class to store Function
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWQFunction")
    fmSJ.addTable("fcFAFunction")
    keepers = []
    keepers = 
['WUKey','Shape_Length','Shape_Area','WQFunction','WQOpportun','WQPotential','WQSociet
y','FAFunction','FAOpportun','FAPotential','FASociety']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis("fcWQFunction", "fcFAFunction", 'WU_Function1', 
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_Function1", "fcFunction1")
    logger.info("spatial join WQFunction and FAFunction completed")

    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcFunction1")
    fmSJ.addTable("fcHFunction")
    keepers = []
    keepers = 
['WUKey','Shape_Length','Shape_Area','WQFunction','WQOpportun','WQPotential','WQSociet
y','FAFunction','FAOpportun','FAPotential','FASociety','HFunction','HOpportun','HPotential','H
Society','HCondition','BRank','HFuncNoBR']
    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis("fcFunction1", "fcHFunction", 'WU_Function2',
'JOIN_ONE_TO_ONE', 'KEEP_ALL', fmSJ, 'CONTAINS')
    arcpy.MakeFeatureLayer_management(r"WU_Function2","fcFunction2")
    logger.info("spatial join Function1 and HFunction completed")

    ## STEP 3: Add Function field to Wetland Units and set initial point value to zero.
    actions.DeleteField("fcFunction2", 'Function')
    arcpy.AddField_management("fcFunction2", 'Function', 'SHORT')
    arcpy.CalculateField_management("fcFunction2", 'Function', '!WQFunction! + !FAFunction!
+ !HFunction!', 'PYTHON', '#')
    logger.info("field Function added and calculated")

    ## STEP 4: Add and calculate fields to store roll-up scores Condition, RegFunction, and
DNRLandAcq
    actions.DeleteField("fcFunction2","Condition")
    arcpy.AddField_management("fcFunction2", "Condition", "SHORT")
    arcpy.CalculateField_management("fcFunction2","Condition","!WQPotential! +
!FAPotential! + !HCondition!","PYTHON","#")

    actions.DeleteField("fcFunction2","RegFunction")
    arcpy.AddField_management("fcFunction2", "RegFunction", "SHORT")
    arcpy.CalculateField_management("fcFunction2","RegFunction","!Condition! +
!WQOpportun! + !FAOpportun!","PYTHON","#")

    actions.DeleteField("fcFunction2","DNRLandAcq")
    arcpy.AddField_management("fcFunction2", "DNRLandAcq", "SHORT")
    arcpy.CalculateField_management("fcFunction2","DNRLandAcq","!Condition! +
!WQSociety! + !FASociety! + !HSociety!","PYTHON","#")
    logger.info("fields added and calculated to store roll-up scores Condition, RegFunction, and
DNRLandAcq")


    ## Select B1-B4 wetlands and re-calculate DNRLandAcq
    strWHERE = """"BRank" IN ('B1', 'B2', 'B3', 'B4')"""
    arcpy.SelectLayerByAttribute_management("fcFunction2", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcFunction2", 'DNRLandAcq', '!Condition! +
!WQSociety! + !FASociety!', 'PYTHON', '#')
    logger.info("B1-B4 wetlands selected and DNRLandAcq re-calculated")
    arcpy.SelectLayerByAttribute_management("fcFunction2", "CLEAR_SELECTION")


    ## Reorder fields
    new_field_order = ["WUKey", "Function", "RegFunction", "Condition", "DNRLandAcq",
"BRank", "FAFunction", "FAOpportun", "FAPotential", "FASociety", "HCondition",
"HFuncNoBR", "HFunction", "HOpportun", "HPotential", "HSociety", "WQFunction",
"WQOpportun", "WQPotential", "WQSociety"]
    existing_fields = arcpy.ListFields("fcFunction2")
    existing_mapping = arcpy.FieldMappings()
    existing_mapping.addTable("fcFunction2")
    new_mapping = arcpy.FieldMappings()


    for field_name in new_field_order:
        mapping_index = existing_mapping.findFieldMapIndex(field_name)
        if mapping_index != -1:
            field_map = existing_mapping.fieldMappings[mapping_index]
            new_mapping.addFieldMap(field_map)
    arcpy.Merge_management("fcFunction2", "WU_Function", new_mapping)
```

```
logger.info("fields reordered")


# Clean up
if arcpy.Exists('WU_Function1'):
    arcpy.Delete_management('WU_Function1')
if arcpy.Exists('WU_Function2'):
    arcpy.Delete_management('WU_Function2')
```

5.7.68 CreateBasicGeom: Utilities

```
##############################################################################
#
# File Name: CreateBasicGeom.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 1/27/2017 (modified 11/1/2017)
# Purpose:
#    Create 50m buffers for wetland units
#
##############################################################################
#!/usr/bin/python
import datetime
import time
import logging
import arcpy
import actions

def CreateBasicGeom(WetlandPoly):
    logger = logging.getLogger("WFA.initRequest.CreateBasicGeom")

    # Clean up if needed
    if arcpy.Exists("WUpoint"):
        arcpy.Delete_management("WUpoint")
    if arcpy.Exists("Buffer10m"):
        arcpy.Delete_management("Buffer10m")
    if arcpy.Exists("Buffer50m"):
        arcpy.Delete_management("Buffer50m")
    if arcpy.Exists("Buffer300m"):
        arcpy.Delete_management("Buffer300m")
    if arcpy.Exists("WU_Buffer1km"):
        arcpy.Delete_management("WU_Buffer1km")


    # Setting python variables
    # fcDrainageAreas =
arcpy.mappng.Layer(r"M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\WorkingFiles\T
AGIS\wetlands_27m.gdb\drainageareas")
    # logger.info("feature layers ready")


    # Create Wetland Units Points
    arcpy.FeatureToPoint_management(WetlandPoly, "WUPoint", "INSIDE")
    logger.info("Wetland Units Points layer created")


    # Create Buffer Wetland Units by 10/50/300/1000 meters
    arcpy.Buffer_analysis(WetlandPoly,"Buffer10m","10
Meters","OUTSIDE_ONLY","ROUND","NONE","#")
    arcpy.Buffer_analysis(WetlandPoly,"Buffer50m","50
Meters","OUTSIDE_ONLY","ROUND","NONE","#")
    arcpy.Buffer_analysis(WetlandPoly,"Buffer300m","300
Meters","OUTSIDE_ONLY","ROUND","NONE","#")
    arcpy.Buffer_analysis(WetlandPoly,"Buffer1km","1000
Meters","OUTSIDE_ONLY","ROUND","NONE","#")
    fcBuffer10m = arcpy.mapping.Layer(r"Buffer10m")
    fcBuffer50m = arcpy.mapping.Layer(r"Buffer50m")
    fcBuffer300m = arcpy.mapping.Layer(r"Buffer300m")
    fcBuffer1km = arcpy.mapping.Layer(r"Buffer1km")
    logger.info("buffers layers of Wetland Units created")


    # Add field to store buffer area
    actions.DeleteField(fcBuffer10m, "BufferArea")
    arcpy.AddField_management(fcBuffer10m, "Buf10Area", "FLOAT")
    arcpy.CalculateField_management(fcBuffer10m, "Buf10Area", "!Shape_Area!",
"PYTHON", "#")


    actions.DeleteField(fcBuffer50m, "BufferArea")
    arcpy.AddField_management(fcBuffer50m, "BufferArea", "FLOAT")
    arcpy.CalculateField_management(fcBuffer50m, "BufferArea", "!Shape_Area!",
"PYTHON", "#")


    actions.DeleteField(fcBuffer300m, "Buf300Area")
    arcpy.AddField_management(fcBuffer300m, "Buf300Area", "FLOAT")
    arcpy.CalculateField_management(fcBuffer300m, "Buf300Area", "!Shape_Area!",
"PYTHON", "#")


    actions.DeleteField(fcBuffer1km,"Buf1kArea")
    arcpy.AddField_management(fcBuffer1km, "Buf1kArea", "FLOAT")
    arcpy.CalculateField_management(fcBuffer1km, "Buf1kArea", "!Shape_Area!",
"PYTHON", "#")
    logger.info("field BufferArea created and calculated for each buffer layer")


    # Create DrainageArea27m
    '''
    actions.DeleteField(fcDrainageAreas,"WUKey")
    arcpy.AddField_management(fcDrainageAreas, "WUKey", "LONG")
    arcpy.CalculateField_management(fcDrainageAreas, "WUKey", "!grid_code! + 1",
"PYTHON", "#")

arcpy.Dissolve_management(fcDrainageAreas,"DrainageArea27m","WUKey","","MULTI_PA
RT","DISSOLVE_LINES")
    logger.info("layer DraomageArea27m created")
    '''
```

5.7.69 CreateWetlandUnits: Utilities

```
##############################################################################
#
# File Name: CreateWetlandUnits.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 7/8/2015 (modified 12/14/2017)
# Purpose:
#    This script intersects is to create wetland units from
#    NWI polygons.
#
#    Note:  The numbers in the commments reflect the number in
#           "2b Creating Wetland Units from NWI polygons.docx"
#
###############################################################################
#!/usr/bin/python
import datetime
import time
import logging
import arcpy
import actions

def CreateWetlandUnits(WetlandPoly):
    logger = logging.getLogger("WFA.initRequest.CreateWetlandUnits")

    #fcWetlandUnits = uid + "_" + str(time.strftime("%Y%m%d_%H%M%S"))
    now = datetime.datetime.now()
    intCurMonth = datetime.date.today().month
    intCurDay = datetime.date.today().day
    fcWetlandUnits = "WU_" + str(datetime.date.today().year) + '{:02}'.format(intCurMonth) + 
'{:02}'.format(intCurDay)

    # Clean up if needed
    if arcpy.Exists(fcWetlandUnits):
        arcpy.Delete_management(fcWetlandUnits)
    if arcpy.Exists("WUWorkingDS"):
        arcpy.Delete_management("WUWorkingDS")
    if arcpy.Exists("WUdissolve"):
        arcpy.Delete_management("WUdissolve")

    ## Create new feature class that doesn't contain  permanently flooded and unvegetated rivers 
and lakes, and industrial and waste disposal ponds
    strWHERE = """"ATTRIBUTE" NOT LIKE 'R%BH' AND "ATTRIBUTE" NOT LIKE 
'L%Hh' AND "ATTRIBUTE" NOT LIKE 'L%Hx' AND "ATTRIBUTE" NOT LIKE 'L%s%' 
AND "ATTRIBUTE" NOT LIKE 'P%r%' AND "ATTRIBUTE" NOT LIKE 'P%K%' AND 
"ATTRIBUTE" NOT LIKE 'PU%s%' AND"ATTRIBUTE" NOT LIKE 'PR%s%'"""
    arcpy.SelectLayerByAttribute_management(WetlandPoly, "NEW_SELECTION", 
strWHERE)
    arcpy.CopyFeatures_management(WetlandPoly,"WUWorkingDS")
    fcWorkingDS = arcpy.mapping.Layer("WUWorkingDS")
    logger.info("WUWorkingDS created to store wetland units that doesn't contain permanently 
flooded and unvegetated rivers and lakes, and industrial and waste disposal ponds")

    ## Dissolve NWI polygons

arcpy.Dissolve_management(fcWorkingDS,"WUdissolve","","","SINGLE_PART","DISSOLV
E_LINES")
    arcpy.MakeFeatureLayer_management("WUdissolve", "fcWUDissolve")
    logger.info("NWI polygons dissolved")

    # Create output feature class
    lyrWUDissolve = arcpy.mapping.Layer("WUdissolve")
    arcpy.FeatureClassToFeatureClass_conversion(lyrWUDissolve, arcpy.env.workspace,
fcWetlandUnits)
    arcpy.RepairGeometry_management(fcWetlandUnits)
    arcpy.AddSpatialIndex_management(fcWetlandUnits)
    logger.info("output feature class created for Wetland Units")

    # Add unique ID field for Wetland Units
    actions.DeleteField(fcWetlandUnits,"WUKey")
    arcpy.AddField_management(fcWetlandUnits, "WUKey", "LONG")
    arcpy.CalculateField_management(fcWetlandUnits, "WUKey", "!OBJECTID!", "PYTHON")
    logger.info("field WUKey added to Wetland Units")

    # Clean up
    if arcpy.Exists("WUWorkingDS"):
        arcpy.Delete_management("WUWorkingDS")
    if arcpy.Exists("WUdissolve"):
        arcpy.Delete_management("WUdissolve")
    return fcWetlandUnits
```

5.7.70 DrainageArea: Utilities

```
##############################################################################
#
# File Name: DrainageArea.py
# Developer: Yibing Han
# Date: 01/26/2018
# Purpose:
#    Input to Flood Attenuation Function / Opportunity aspect
#
##############################################################################
#!/usr/bin/python
import sys, os, gc
sys.path.append("../..")
gc.enable()

import logging
import arcpy
from arcpy.sa import *
from globalvars import globalvars
import actions

def DetermineDrainageArea(WetlandPoly):
    arcpy.CheckOutExtension("Spatial")
    logger = logging.getLogger("WFA.DrainageArea")

    # Clean up if needed
    if arcpy.Exists(r"DrainageArea0"):
        arcpy.Delete_management(r"DrainageArea0")
    if arcpy.Exists(r"DrainageArea"):
        arcpy.Delete_management(r"DrainageArea")


    # Setting python variables
    orig_dir = arcpy.env.workspace
    temp_dir =
"M:\\wr\\WTRSHD_BRANCH_INTERNAL\\WETLAND\\WorkingFiles\\drainage_area_temp
"
    if not os.path.exists(temp_dir):
        os.mkdir(temp_dir)
    if arcpy.Exists(temp_dir + "\\tempGDB.gdb"):
        arcpy.Delete_management(temp_dir + "\\tempGDB.gdb")
    arcpy.CreateFileGDB_management(temp_dir, "tempGDB")
    ## Create feature class to hold watershed polygon output
    sr = arcpy.SpatialReference(26917)
    arcpy.CreateFeatureclass_management(orig_dir, "DrainageArea0", "POLYGON","","","",sr)
    actions.DeleteField("DrainageArea0", "Gridcode")
    arcpy.AddField_management("DrainageArea0", "Gridcode", "LONG")
    logger.info("feature class DrainageArea0 created to hold watershed polygon")


    ## BEGIN LOOP; start with WUKey = 1 and repeat until all Wetland Units are processed
    arcpy.FeatureClassToFeatureClass_conversion(WetlandPoly, temp_dir + "\\tempGDB.gdb",
"WU_temp")
    arcpy.env.workspace = temp_dir + "\\tempGDB.gdb"
    arcpy.env.overwriteOutput = True
    arcpy.env.extent = globalvars.srcFlowDir
    arcpy.env.snapRaster = globalvars.srcFlowDir
    with arcpy.da.SearchCursor("WU_temp", "WUKey") as cursor:
        for row in cursor:
            # Copy Wetland Unit polygon to a temporary feature class
            strWhere = "WUKey = " + str(row[0])
            #tmpName = "temp_{0}".format(row[0])
            arcpy.FeatureClassToFeatureClass_conversion("WU_temp", arcpy.env.workspace,
"temppoly", strWhere)
            # Convert temporary feature class to a temporary grid
            arcpy.PolygonToRaster_conversion(arcpy.env.workspace + "\\temppoly", "WUKey",
"tempras", "CELL_CENTER", "", 27)
            if arcpy.GetRasterProperties_management(arcpy.env.workspace + "\\tempras",
"ALLNODATA").getOutput(0) == '1':
                arcpy.Buffer_analysis(arcpy.env.workspace + "\\temppoly", "temppolyB", 27)
                arcpy.PolygonToRaster_conversion(arcpy.env.workspace + "\\temppolyB",
"WUKey", "tempras", "CELL_CENTER", "WUKey", 27)
            # Run watershed command using temporary grid as input
            #outWatershed = arcpy.sa.Watershed(globalvars.srcFlowDir, "tempras", "Value")
            #outWatershed.save("temprasout.tif")
            #arcpy.gp.Watershed_sa(globalvars.srcFlowDir,arcpy.env.workspace +
"\\tempras","temprasout","Value")
            outWatershed = Watershed(globalvars.srcFlowDir,arcpy.env.workspace +
"\\tempras","Value")


            # Convert watershed grid to a polygon
            #arcpy.RasterToPolygon_conversion(arcpy.env.workspace + "\\temprasout",
"temppolyout", "NO_SIMPLIFY", "Value")
            arcpy.RasterToPolygon_conversion(outWatershed, "temppolyout", "NO_SIMPLIFY",
"Value")


            # Append polygon to DrainageArea0 feature class
            arcpy.Append_management(arcpy.env.workspace + "\\temppolyout", orig_dir +
"\\DrainageArea0", "NO_TEST")
            logger.info("Row " + str(row[0]) +" appended to feature class")
        del row, cursor
        gc.collect()
    ## END LOOP when all Wetland Units are processed
    arcpy.env.workspace = orig_dir

#arcpy.FeatureClassToFeatureClass_conversion("DrainageArea0",orig_dir,"DrainageArea1",""
""Gridcode" > 0""")
    logger.info("All Wetland Units have now been added to DrainageArea0")


    ## Add field to DrainageArea feature class to store WUKey
    actions.DeleteField("DrainageArea0", "WUKey")
    arcpy.AddField_management("DrainageArea0", "WUKey", "LONG")
    arcpy.CalculateField_management("DrainageArea0", "WUKey", "!Gridcode!", "PYTHON")

arcpy.Dissolve_management("DrainageArea0","DrainageArea","WUKey","","MULTI_PART",
"DISSOLVE_LINES")
    logger.info("field WUKey added to DrainageArea")


    ## Add field to store Contributing Watershed Area
    actions.DeleteField("DrainageArea", "CntrWshd")
    arcpy.AddField_management("DrainageArea", "CntrWshd", "LONG")
    arcpy.CalculateField_management("DrainageArea", "CntrWshd", "int(!Shape_Area!)",
"PYTHON")


    # Clean Up
    if arcpy.Exists(r"DrainageArea0"):
        arcpy.Delete_management(r"DrainageArea0")
```

5.7.71 FloodArea: Utilities

```
##############################################################################
#
# File Name: FloodArea.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/8/2017 (modified 11/10/2017)
# Purpose:
#    Floodplain (Y/N): Input to numerous Water Quality and Flood Attenuation metrics
#    FloodArea: Input to Flood Attenuation / Opportunity; Max 2 points (all wetlands, but only
Floodplain wetlands will score high enough to get points)
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../..")

import logging
import arcpy
from globalvars import globalvars
import actions

def DetermineFloodArea(WetlandPoly):
    logger = logging.getLogger("WFA.FloodArea")

    ## Clean up if needed
    if arcpy.Exists("WU_FloodArea1"):
        arcpy.Delete_management("WU_FloodArea1")
    if arcpy.Exists("WU_FloodArea"):
        arcpy.Delete_management("WU_FloodArea")
    ## Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcFPARAFEMA,"fcFPARAFEMA")
    arcpy.MakeFeatureLayer_management(globalvars.srcPeatlands,"fcPeatlands")
    logger.info("feature layers ready")

    ## Intersect floodplain and Wetland Units
    arInputs = ["fcFPARAFEMA","fcWU"]
    arcpy.Intersect_analysis(arInputs,"WU_FloodArea1","ONLY_FID","#","INPUT")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea1","fcWUFloodArea1")
    logger.info("floodplain intersected with wetland units")

    ## Add field to store floodplain area
    actions.DeleteField("fcWUFloodArea1","FloodAreaAF")
    arcpy.AddField_management("fcWUFloodArea1", "FloodAreaAF", "FLOAT")

arcpy.CalculateField_management("fcWUFloodArea1","FloodAreaAF","!Shape_Area!","PYT
HON","#")
    logger.info("field FloodAreaAF added to store floodplain area")


################################################################################
################
    ## Spatial join floodplain selection to Wetland Units and sum floodplain area.

################################################################################
################
    fieldmappings = arcpy.FieldMappings()
    fieldmappings.addTable("fcWU")
    fieldmappings.addTable("fcWUFloodArea1")
    fldKeyIndex = fieldmappings.findFieldMapIndex("FloodAreaAF")
    fieldmap = fieldmappings.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "sum"
    fieldmappings.replaceFieldMap(fldKeyIndex, fieldmap)
    keepers = ["Shape_Length","Shape_Area","WUKey","FloodAreaAF"]

    for field in fieldmappings.fields:
        if field.name not in keepers:
            fieldmappings.removeFieldMap(fieldmappings.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWU","fcWUFloodArea1","WU_FloodArea","JOIN_ONE_TO_O
NE","KEEP_ALL",fieldmappings)
    arcpy.MakeFeatureLayer_management("WU_FloodArea","fcWUFloodArea")
    logger.info("spatuak join completed to sum floodplain area")


################################################################################
################
    ## Add fields to store Flood Area Ratio and Flood Area points
    actions.DeleteField("fcWUFloodArea","FloodRatio")
    arcpy.AddField_management("fcWUFloodArea", "FloodRatio", "FLOAT")
    arcpy.CalculateField_management("fcWUFloodArea","FloodRatio","0","PYTHON","#")
    logger.info("field FloodRatio added and initial point value set to 0")

    actions.DeleteField("fcWUFloodArea","FloodArea")
    arcpy.AddField_management("fcWUFloodArea", "FloodArea", "SHORT")
    arcpy.CalculateField_management("fcWUFloodArea","FloodArea","0","PYTHON","#")
    logger.info("field FloodArea added and initial point value set to 0")

    ## Calculate the ratio of floodplain are to total Wetland  Units Area
```

```
arcpy.CalculateField_management("fcWUFloodArea","FloodRatio","!FloodAreaAF!/!Shape_A
rea!","PYTHON","#")
    logger.info("field FloodRation calculated")

    ## Assign Points

arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""Floo
dRatio" > 0.1""")
    arcpy.CalculateField_management("fcWUFloodArea","FloodArea","1","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""Floo
dRatio" > 0.5""")
    arcpy.CalculateField_management("fcWUFloodArea","FloodArea","2","PYTHON","#")
    logger.info("points assigned to field FloodArea")
    arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

    ## Select Wetland Units that contain peat deposits

arcpy.SelectLayerByLocation_management("fcWUFloodArea","INTERSECT","fcPeatlands","
#","NEW_SELECTION")
    arcpy.CalculateField_management("fcWUFloodArea","FloodArea","0","PYTHON","#")
    logger.info("points assigned to wetland units that contain peat deposits")
    arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

    ## Add field Floodplain to Wetland Units attribute table and set initial value to "N"
    actions.DeleteField("fcWUFloodArea","Floodplain")
    arcpy.AddField_management("fcWUFloodArea", "Floodplain", "TEXT",2)
    arcpy.CalculateField_management("fcWUFloodArea","Floodplain","'N'","PYTHON","#")
    logger.info("field Floodplain added and initial value set to 'N'")
    ## Select Wetland Units that have at least 10% of their area in a FEMA floodplain or Active
River Area

arcpy.SelectLayerByAttribute_management("fcWUFloodArea","NEW_SELECTION","""""Floo
dArea" > 0""")
    arcpy.CalculateField_management("fcWUFloodArea","Floodplain","'Y'","PYTHON","#")
    logger.info("field Floodplain identified in selected wetland units")
    arcpy.SelectLayerByAttribute_management("fcWUFloodArea", "CLEAR_SELECTION")

    ## Clean up
    if arcpy.Exists("WU_FloodArea1"):
        arcpy.Delete_management("WU_FloodArea1")
```

5.7.72 Floodplain: Utilities

```
#############################################################################
#
# File Name: floodplain.py
# Developer: Chad Ashworth
# Date 4/24/2015
# Purpose:
#	This script intersects the WVWRAP wetland polygons and the FEMA
#	floodplain polygons and determines if the WVWRAP wetland is
#	10% or more in the FEMA floodplain.
#
##############################################################################
#!/usr/bin/python
import arcpy
from utilities import actions
arcpy.env.overwriteOutput = True

def WetlandInFloodplain(WetlandPoly):
## cleaning up if needed
if arcpy.Exists("WETLAND_FP_INTERSECT"):
	arcpy.Delete_management("WETLAND_FP_INTERSECT")
if arcpy.Exists("lyrWetlandAllData"):
	arcpy.Delete_management("lyrWetlandAllData")

## reference to the floodzone layer
fcFP = actions.GetGISData("FloodZone")

# adding/deleting the Floodplain_FEMA
actions.DeleteField(WetlandPoly,"Floodplain_FEMA")
```

```
arcpy.AddField_management(WetlandPoly, "Floodplain_FEMA", "TEXT",1)

## Perform an intersect between WVWRAP sites & FloodPlain feature classes
arcpy.Intersect_analysis([[WetlandPoly], [fcFP]], "WETLAND_FP_INTERSECT","ALL")

## Join intersecting feature class with lyrWetlandAllData
arcpy.MakeFeatureLayer_management("WETLAND_FP_INTERSECT", "lyrWetlandAllData")
arcpy.AddJoin_management("lyrWetlandAllData", "FID_wvFldZone_20130410_wgs84wmA",
fcFP, "OBJECTID") # join to Flood Zone Data
strWetlandPolyPK = "FID_" + str(WetlandPoly) # dynamically createing the foreign key field
to the WetlandPoly
arcpy.AddJoin_management("lyrwetlandalldata", strWetlandPolyPK, str(WetlandPoly),
"OBJECTID") # join to WetlandPoly

arcpy.CopyFeatures_management("lyrWetlandAllData", "lyrWetlandAllData")

##########################################################################
edit = arcpy.da.Editor(arcpy.env.workspace)
edit.startEditing(False, True)
edit.startOperation()

# declaring what fields i need to calculate the percent intersect
strWetlandId = str(WetlandPoly) + ".OBJECTID"
strWetlandShapeArea = str(WetlandPoly) + ".Shape_Area"
strWetlandFlFEMA = str(WetlandPoly) + ".Floodplain_FEMA"
fldsDL =
[strWetlandId,"WETLAND_FP_INTERSECT.Shape_Area",strWetlandShapeArea,strWetlandF
lFEMA]

# updating the UA field with a Y if 10% or more are in an urban area
with arcpy.da.SearchCursor("lyrWetlandAllData", fldsDL) as curWetlandAll:
        for rWetlandAll in curWetlandAll:
                intPercentIntersect = int(rWetlandAll[1]/rWetlandAll[2]*100)
                if intPercentIntersect >= 10:
                        strWhere = "OBJECTID = " + str(rWetlandAll[0])

        actions.UpdateTable(WetlandPoly,["Floodplain_FEMA"],0,"Y",strWhere)

# updating the rest of the fields to N
strWhere = "Floodplain_FEMA is null"
actions.UpdateTable(WetlandPoly,["Floodplain_FEMA"],0,"N",strWhere)

edit.stopEditing(True)

arcpy.Delete_management("lyrWetlandAllData")
arcpy.Delete_management("WETLAND_FP_INTERSECT")
arcpy.Delete_management("lyrWetlandAllData")
```

5.7.73 InitRequest: Utilities

```
##############################################################################
#
# File Name: initRequest.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 10/24/2015 (modified 02/2018)
# Purpose:
#    This script starts the Wetland Functional Assessment process by creating wetland unit
polygons, basic geometries and flood-related layers.
#
##############################################################################
#!/usr/bin/python
import sys, os
import arcpy
import logging
import traceback
import actions
from CreateWetlandUnits import *
from CreateBasicGeom import *
from FloodArea import *
from DrainageArea import *
#from Floodplain import *
#from ActiveRiverArea import *

def MainUtil(WetlandPoly):
    logger = logging.getLogger("WFA.initRequest")
    ## 1. Check Geometry
    '''sr = arcpy.SpatialReference(26917)
    arcpy.CreateFeatureDataset_management(arcpy.env.workspace,"CONUS_wetlands", sr)
    arcpy.CreateFeatureDataset_management(arcpy.env.workspace,"CONUS_projects", sr)
    arcpy.FeatureClassToFeatureClass_conversion(WetlandPoly, arcpy.env.workspace + 
"\\CONUS_wetlands","CONUS_wet_poly")
    arcpy.FeatureClassToFeatureClass_conversion("WUboundary", arcpy.env.workspace + 
"\\CONUS_projects","CONUS_wet_projects")
    arcpy.AddField_management(r"CONUS_wetlands\CONUS_wet_poly", "QAQC_CODE", 
"TEXT")
    arcpy.AddField_management(r"CONUS_wetlands\CONUS_wet_poly", 
"WETLAND_TYPE", "TEXT")
    arcpy.AddField_management(r"CONUS_wetlands\CONUS_wet_poly", "ACRES", 
"FLOAT")
    arcpy.AddField_management(r"CONUS_projects\CONUS_wet_projects", "ACRES", 
"FLOAT")
    arcpy.CreateTopology_management(r"CONUS_wetlands","CONUS_wet_poly_Topology")
    logger.info("Wetland database schema for QAQC set up")

arcpy.ImportToolbox('M:\wr\WTRSHD_BRANCH_INTERNAL\WETLAND\NWI_QAQC_T
ool\NWI_QAQC_Tool.tbx')
    arcpy.Model_QAQC(arcpy.env.workspace,"Name","CONUS","true")
    logger.info("QAQC completed for input wetlands")
    arcpy.CopyFeatures_management(r"CONUS_wetlands\CONUS_wet_poly", 
"WetlandInput_QAQC")
    WetlandPoly = arcpy.mapping.Layer("WetlandInput_QAQC")
    arcpy.Delete_management(r"CONUS_projects")
    arcpy.Delete_management(r"CONUS_wetlands")'''


    ## 2. Create Wetland Units
    logger.info("Creating wetland Units")
    try:
        strWU = CreateWetlandUnits(WetlandPoly)
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        logger.error("Failed to create Wetland Units")
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        logger.error("Failed to create Wetland Units")
        sys.exit(1)

    ## 3. Create basic geometries used for assessment of Wetland Units
    logger.info("Creating basic geometries used for assessment of Wetland Units")
    try:
        CreateBasicGeom(strWU)
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        logger.error("Failed to create basic geometries used for assessment of Wetland Units")
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        logger.error("Failed to create basic geometries used for assessment of Wetland Units")
        sys.exit(1)

    ## 4. Floodplain
    try:
        DetermineFloodArea(strWU)
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        logger.error("Failed to calculate flood area")
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        logger.error("Failed to calculate flood area")
        sys.exit(1)

    ## 5. DrainageArea
    try:
        DetermineDrainageArea(strWU)
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        logger.error("Failed to calculate flood area")
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        logger.error("Failed to calculate flood area")
        sys.exit(1)

    return strWU
```

5.7.74 Water Quality

```
##############################################################################
#
# File Name: WQuality.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all metrics within the Water Quality module.
#
##############################################################################
#!/usr/bin/python
import sys
import arcpy
import datetime
import logging
import traceback

from WQPotential import WQPotential
from WQOpportun import WQOpportun
from WQSociety import WQSociety
from WQFunction import WQFunction

logger = logging.getLogger("WFA.WQuality")

def RunWaterQuality(WetlandPoly):

    ## 1. Run Water Quality Potential Variables/Aspects/Factors
    logger.info("Running Water Quality Potential Variables/Aspects/Factors...")
    try:
        WQPotential.procWQPotential(WetlandPoly)
        logger.info("Water Quality Potential Variables/Aspects/Factors completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 2. Run Water Quality Opportunity Variables/Aspects/Factors
    logger.info("Running Water Quality Opportunity Variables/Aspects/Factors...")
    try:
        WQOpportun.procWQOpportun(WetlandPoly)
        logger.info("Water Quality Opportunity Variables/Aspects/Factors completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 3. Run Water Quality Society Variables/Aspects/Factors
    logger.info("Running Water Quality Society Variables/Aspects/Factors...")
    try:
        WQSociety.procWQSociety(WetlandPoly)
        logger.info("Water Quality Society Variables/Aspects/Factors completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(1) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)

    ## 4. Run Water Quality Function to roll up Variables/Aspects/Factors
    logger.info("Running Water Quality Function to roll up Variables/Aspects/Factors...")
    try:
        WQFunction.DetermineWQFunction(WetlandPoly)
        logger.info("Water Quality Function to roll up Variables/Aspects/Factors completed")
    except arcpy.ExecuteError:
        msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
        arcpy.AddError(msgs)
        logger.error(msgs)
        sys.exit(1)
    except:
        tb = sys.exc_info()[2]
        tbinfo = traceback.format_tb(tb)[0]
        pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
        msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

        arcpy.AddError(pymsg)
        arcpy.AddError(msgs)
        logger.error(pymsg)
        logger.error(msgs)
        sys.exit(1)
```

5.7.75 Water Quality Function

```
##############################################################################
#
# File Name: WQFunction.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/13/2017 (modified 11/02/2017)
# Purpose:
#    Water Quality Function
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineWQFunction(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQFunction")

    # Clean up if needed
    if arcpy.Exists("WU_WQFunction0"):
        arcpy.Delete_management("WU_WQFunction0")
    if arcpy.Exists("WU_WQFunction1"):
        arcpy.Delete_management("WU_WQFunction1")
    if arcpy.Exists("WU_WQFunction2"):
        arcpy.Delete_management("WU_WQFunction2")
    if arcpy.Exists("WU_WQFunction"):
        arcpy.Delete_management("WU_WQFunction")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_WQPotential","fcWQPotential")
    arcpy.MakeFeatureLayer_management(r"WU_WQOpportun","fcWQOpportun")
    arcpy.MakeFeatureLayer_management(r"WU_WQSociety","fcWQSociety")
    logger.info("feature layers ready")

    ## Create feature class to store WQFunction
    arcpy.CopyFeatures_management(WetlandPoly, "WU_WQFunction0")
    arcpy.MakeFeatureLayer_management("WU_WQFunction0","fcWUWQFunction0")
    logger.info("feature class WU_WQFunction0 created")

    ## Spatial join to bring in aspect values
################################################################################
#################
    # SJ: WQPotential

################################################################################
#################
    fmSJWQPO = arcpy.FieldMappings()
    fmSJWQPO.addTable("fcWUWQFunction0")
    fmSJWQPO.addTable("fcWQPotential")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WQPotential"]

    for field in fmSJWQPO.fields:
        if field.name not in keepers:
            fmSJWQPO.removeFieldMap(fmSJWQPO.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcWUWQFunction0","fcWQPotential","WU_WQFunction1","JOI
N_ONE_TO_ONE","KEEP_ALL",fmSJWQPO,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQFunction1","fcWQFunction1")
    logger.info("Spatial Join completed to add variable WQPotential")


################################################################################
#################
    # SJ: WQOpportun

################################################################################
#################
    fmSJWQOP = arcpy.FieldMappings()
    fmSJWQOP.addTable("fcWQFunction1")
    fmSJWQOP.addTable("fcWQOpportun")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WQPotential","WQOpportun"]

    for field in fmSJWQOP.fields:
        if field.name not in keepers:
            fmSJWQOP.removeFieldMap(fmSJWQOP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQFunction1","fcWQOpportun","WU_WQFunction2","JOIN_
ONE_TO_ONE","KEEP_ALL",fmSJWQOP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQFunction2","fcWQFunction2")
    logger.info("Spatial Join completed to add variable WQOpportun")
```

```
################################################################################
#################
    # SJ: WQSociety

################################################################################
#################
    fmSJWQS = arcpy.FieldMappings()
    fmSJWQS.addTable("fcWQFunction2")
    fmSJWQS.addTable("fcWQSociety")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","WQPotential","WQOpportun","WQSociety"]

    for field in fmSJWQS.fields:
        if field.name not in keepers:
            fmSJWQS.removeFieldMap(fmSJWQS.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQFunction2","fcWQSociety","WU_WQFunction","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJWQS,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQFunction","fcWQFunction")
    logger.info("Spatial Join completed to add variable WQSociety")

    ## Add WQFunction field to Wetland Units and set initial point value to 0
    actions.DeleteField("fcWQFunction","WQFunction")
    arcpy.AddField_management("fcWQFunction", "WQFunction", "SHORT")
    arcpy.CalculateField_management("fcWQFunction","WQFunction","0","VB","#")
    logger.info("WQFunction field added to Wetland Units and initial point values set to zero")
    ## Sum the points for each aspect of Water Quality Function for Wetland Units

arcpy.CalculateField_management("fcWQFunction","WQFunction","[WQPotential]+[WQOpp
ortun]+[WQSociety]","VB","#")
    logger.info("points summed for WQFunction")


    ## Clean up
    if arcpy.Exists("WU_WQFunction0"):
        arcpy.Delete_management("WU_WQFunction0")
    if arcpy.Exists("WU_WQFunction1"):
        arcpy.Delete_management("WU_WQFunction1")
    if arcpy.Exists("WU_WQFunction2"):
        arcpy.Delete_management("WU_WQFunction2")
```

5.7.76 Water Quality Opportunity

```
##############################################################################
#
# File Name: WQOpportun.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Water Quality Opportunity metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Factors import Discharges, ImpairedIn, RoadRail, Disturb50m, DisturbWshd
from Aspects import WQOpportun

def procWQOpportun(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQOpportun")

    ############################################################
    ## 1. Run Variables
    ############################################################

    # None

    ############################################################
    ## 2. Run Factors
```

```
############################################################

Discharges.DetermineDischarges(WetlandPoly)

ImpairedIn.DetermineImpairedIn()

RoadRail.CalcRoadRail(WetlandPoly)

Disturb50m.DetermineDisturb50m(WetlandPoly)

DisturbWshd.DetermineDisturbWshd(WetlandPoly)

############################################################
## 3. Run Aspect
############################################################

WQOpportun.DetermineWQOpportun(WetlandPoly)
```

5.7.77 Water Quality Opportunity Aspects

```
##############################################################################
#
# File Name: WQOpportun.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/15/2016 (modified 11/02/2017)
# Purpose:
#    Water Quality Function
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineWQOpportun(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQOpportun")

    # Clean up if needed
    if arcpy.Exists("WU_WQOpportun0"):
        arcpy.Delete_management("WU_WQOpportun0")
    if arcpy.Exists("WU_WQOpportun1"):
        arcpy.Delete_management("WU_WQOpportun1")
    if arcpy.Exists("WU_WQOpportun2"):
        arcpy.Delete_management("WU_WQOpportun2")
    if arcpy.Exists("WU_WQOpportun3"):
        arcpy.Delete_management("WU_WQOpportun3")
    if arcpy.Exists("WU_WQOpportun4"):
        arcpy.Delete_management("WU_WQOpportun4")
    if arcpy.Exists("WU_WQOpportun5"):
        arcpy.Delete_management("WU_WQOpportun5")
    if arcpy.Exists("WU_WQOpportun"):
        arcpy.Delete_management("WU_WQOpportun")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"WU_WQOpportun0")
    arcpy.MakeFeatureLayer_management(r"WU_Discharges","fcDischarges")
    arcpy.MakeFeatureLayer_management(r"WU_ImpairedIn","fcImpairedIn")
    arcpy.MakeFeatureLayer_management(r"WU_RoadRail","fcRoadRail")
    arcpy.MakeFeatureLayer_management(r"WU_Disturb50m","fcDisturb50m")
    arcpy.MakeFeatureLayer_management(r"WU_DisturbWshd","fcDisturbWshd")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    logger.info("feature layers ready")

    # Spatial join to bring in factor values

################################################################################
#################
    # SJ: Discharges

################################################################################
#################
    fmSJDis = arcpy.FieldMappings()
    fmSJDis.addTable("WU_WQOpportun0")
    fmSJDis.addTable("fcDischarges")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Discharges"]

    for field in fmSJDis.fields:
        if field.name not in keepers:
            fmSJDis.removeFieldMap(fmSJDis.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WU_WQOpportun0","fcDischarges","WU_WQOpportun1","JOIN
_ONE_TO_ONE","KEEP_ALL",fmSJDis,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun1","fcWQOpportun1")
    logger.info("Spatial Join completed to add variable Discharges")


###############################################################################
#################
    # SJ: ImpairedIn

###############################################################################
#################
    fmSJII = arcpy.FieldMappings()
    fmSJII.addTable("fcWQOpportun1")
    fmSJII.addTable("fcImpairedIn")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Discharges","ImpairedIn"]

    for field in fmSJII.fields:
        if field.name not in keepers:
            fmSJII.removeFieldMap(fmSJII.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcWQOpportun1","fcImpairedIn","WU_WQOpportun2","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJII,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun2","fcWQOpportun2")
    logger.info("Spatial Join completed to add variable ImpairedIn")


################################################################################
#################
    # SJ: RoadRail

################################################################################
#################
    fmSJRR = arcpy.FieldMappings()
    fmSJRR.addTable("WU_WQOpportun2")
    fmSJRR.addTable("fcRoadRail")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Discharges","ImpairedIn","RoadRail"]

    for field in fmSJRR.fields:
        if field.name not in keepers:
            fmSJRR.removeFieldMap(fmSJRR.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQOpportun2","fcRoadRail","WU_WQOpportun3","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJRR,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun3","fcWQOpportun3")
    logger.info("Spatial Join completed to add variable RoadRail")
```

```
##############################################################################
#################
    # SJ: Distrub50m

##############################################################################
#################
    fmSJD50 = arcpy.FieldMappings()
    fmSJD50.addTable("fcWQOpportun3")
    fmSJD50.addTable("fcDisturb50m")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Discharges","ImpairedIn","RoadRail","Disturb50m
"]

    for field in fmSJD50.fields:
        if field.name not in keepers:
            fmSJD50.removeFieldMap(fmSJD50.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQOpportun3","fcDisturb50m","WU_WQOpportun4","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJD50,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun4","fcWQOpportun4")
    logger.info("Spatial Join completed to add variable Distrub50m")


##############################################################################
#################
    # SJ: DistrubWshd

##############################################################################
#################
```

```
    fmSJDW = arcpy.FieldMappings()
    fmSJDW.addTable("fcWQOpportun4")
    fmSJDW.addTable("fcDisturbWshd")

    keepers = []
    keepers = 
["WUKey","Shape_Length","Shape_Area","Discharges","ImpairedIn","RoadRail","Disturb50m
","DisturbWshd"]

    for field in fmSJDW.fields:
        if field.name not in keepers:
            fmSJDW.removeFieldMap(fmSJDW.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQOpportun4","fcDisturbWshd","WU_WQOpportun5","JOIN_
ONE_TO_ONE","KEEP_ALL",fmSJDW,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun5","fcWQOpportun5")
    logger.info("Spatial Join completed to add variable DistrubWshd")


################################################################################
#################
    # SJ: Floodplain

################################################################################
#################
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("fcWQOpportun5")
    fmSJFP.addTable("fcFloodplain")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Discharges","ImpairedIn","RoadRail","Disturb50m
","DisturbWshd","Floodplain"]

    for field in fmSJFP.fields:
        if field.name not in keepers:
            fmSJFP.removeFieldMap(fmSJFP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQOpportun5","fcFloodplain","WU_WQOpportun","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJFP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQOpportun","fcWQOpportun")
    logger.info("Spatial Join completed to add variable Floodplain")

    # Add WQOpportun field and set initial value to 0
    actions.DeleteField("fcWQOpportun","WQOpportun")
    arcpy.AddField_management("fcWQOpportun", "WQOpportun", "SHORT")
    arcpy.CalculateField_management("fcWQOpportun","WQOpportun","0","VB","#")
    logger.info("WQOpportun field added to Wetland Units and initial point values set to zero")

    # Sum the factor points
    arcpy.CalculateField_management("fcWQOpportun","WQOpportun","[Discharges] +
[ImpairedIn] + [RoadRail] + [Disturb50m] + [DisturbWshd]","VB","#")
    logger.info("points summed for WQOpportun")

    # Reduce values that exceed the maximum allowable points

arcpy.SelectLayerByAttribute_management("fcWQOpportun","NEW_SELECTION","""""Flood
plain" = 'Y' AND "WQOpportun" > 5""")
    arcpy.CalculateField_management("fcWQOpportun","WQOpportun","5","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWQOpportun", "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management("fcWQOpportun","NEW_SELECTION","""""Flood
plain" = 'N' AND "WQOpportun" > 4""")
    arcpy.CalculateField_management("fcWQOpportun","WQOpportun","4","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWQOpportun", "CLEAR_SELECTION")
    logger.info("points reduced to maximum allowed for WQOpportun")

    # Clean up
    if arcpy.Exists("WU_WQOpportun0"):
        arcpy.Delete_management("WU_WQOpportun0")
    if arcpy.Exists("WU_WQOpportun1"):
        arcpy.Delete_management("WU_WQOpportun1")
    if arcpy.Exists("WU_WQOpportun2"):
        arcpy.Delete_management("WU_WQOpportun2")
    if arcpy.Exists("WU_WQOpportun3"):
        arcpy.Delete_management("WU_WQOpportun3")
    if arcpy.Exists("WU_WQOpportun4"):
        arcpy.Delete_management("WU_WQOpportun4")
    if arcpy.Exists("WU_WQOpportun5"):
        arcpy.Delete_management("WU_WQOpportun5")
```

5.7.78 Discharges: Water Quality Opportunity

```
##########################################################################
#
# File Name: Discharges.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/8/2016 (modified 12/01/2017)
# Purpose:
#    Water Quality Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineDischarges(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQOpportun.Discharges")

    # Clean up if needed
    if arcpy.Exists("WU_Discharges"):
        arcpy.Delete_management("WU_Discharges")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcSeptic, "fcSeptic")
    arcpy.MakeFeatureLayer_management(globalvars.srcOWRNPDES, "fcOWRNPDES")
    arcpy.MakeFeatureLayer_management(globalvars.srcOWRNPDESOutlets,
"fcOWRNPDESOutlets")
    arcpy.MakeFeatureLayer_management(globalvars.srcHPU, "fcHPU")
    arcpy.MakeFeatureLayer_management(globalvars.srcAMLAMD, "fcAMLAMD")
    arcpy.MakeFeatureLayer_management(globalvars.srcWellPads, "fcWellPads")
    arcpy.MakeFeatureLayer_management(globalvars.srcNPLPoint, "fcNPLPoint")
    arcpy.MakeFeatureLayer_management(globalvars.srcNPLBndry, "fcNPLBndry")

    # Create feature class to store Discharges variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_Discharges")
    fcWUDischarges = arcpy.mapping.Layer(r"WU_Discharges")
    logger.info("feature layers ready")

    # Add Discharges field to Wetland Units and set the initial value to zero
    actions.DeleteField(fcWUDischarges,"Discharges")
    arcpy.AddField_management(fcWUDischarges, "Discharges", "SHORT")
    arcpy.CalculateField_management(fcWUDischarges, "Discharges", "0", "VB")
    logger.info("field Discharges added and initial value set to 0")


##############################################################################
#
    ## PART 1: Select the owrnpdes_ records that are not deep injection points
    ##       and not spectic tanks

##############################################################################
#
    strWHERE = """"perm_type" = '401 Certification' OR "perm_type" = 'Industrial' OR
"perm_type" = 'Sewage' OR "perm_type" = 'UIC Sewage' OR "perm_type" = 'UIC Stormwater
Industrial'"""
    arcpy.SelectLayerByAttribute_management("fcOWRNPDES", "NEW_SELECTION",
strWHERE)
```

```
    # Select Wetland Units within 100m of Septic or selected NPDES records and assign 1 point

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcSeptic","100
Meters","NEW_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcOWRNPDES"
,"100 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units within 100m of Septic or selected NPDES records selected")


    arcpy.CalculateField_management(fcWUDischarges,"Discharges","1","VB","#")
    logger.info("1 point assigned to selected wetland units")


    arcpy.SelectLayerByAttribute_management(fcWUDischarges, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcSeptic", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcOWRNPDES", "CLEAR_SELECTION")



################################################################################
#########
    ## PART 2: Select relevant records from owrnpdes_outlets that are not deep injection

################################################################################
#########
    strWHERE = """"perm_type" = 'Industrial' OR "perm_type" = 'Sewage' OR "perm_type" =
'UIC Sewage' OR "perm_type" = 'UIC Stormwater Industrial""""
    arcpy.SelectLayerByAttribute_management("fcOWRNPDESOutlets",
"NEW_SELECTION", strWHERE)


    # Select relevant records (outlets with open status) from Hydrologic Protection Units
    #strWHERE = """"stat_flag" = 'O' AND "insp_type" = 'OUTLT'""" #Field names change
3/15/2017 MCA
    strWHERE = """"STATUS_FLA" = 'O' AND "INSPECTA_1" = 'OUTLT'""""
    arcpy.SelectLayerByAttribute_management("fcHPU", "NEW_SELECTION", strWHERE)
    logger.info("relevant records (outlets with open status) from Hydrologic Protection Units
selected")

    # Select Wetland Units within 100m of potential discharges and assign 2 points

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcOWRNPDES
Outlets","100 Meters","NEW_SELECTION")
    arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcHPU","100
Meters","ADD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcAMLAMD","
100 Meters","ADD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcWellPads","10
0 Meters","ADD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcNPLPoint","1
00 Meters","ADD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(fcWUDischarges,"INTERSECT","fcNPLBndry","
100 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units within 100m of potential discharges selected")

    arcpy.CalculateField_management(fcWUDischarges,"Discharges","2","VB","#")
    logger.info("2 points assigned to selected wetland units")

    arcpy.SelectLayerByAttribute_management(fcWUDischarges, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcOWRNPDESOutlets",
"CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcHPU", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcAMLAMD", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcWellPads", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNPLPoint", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNPLBndry", "CLEAR_SELECTION")
```

5.7.79 Distrub50: Water Quality Opportunity

```
##############################################################################
#
# File Name: Disturb50m.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/16/2016 (modified 10/31/2017)
# Purpose:
#    Water Quality Function / Opportunity aspect
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineDisturb50m(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQOpportun.Disturb50m")

    # Clean up if needed
    if arcpy.Exists("Buffer50mDist"):
        arcpy.Delete_management("Buffer50mDist")
    if arcpy.Exists("Buffer50mDist_diss"):
        arcpy.Delete_management("Buffer50mDist_diss")
    if arcpy.Exists("WU_Disturb50m"):
        arcpy.Delete_management("WU_Disturb50m")
    if arcpy.Exists("WU_Disturb50m1"):
        arcpy.Delete_management("WU_Disturb50m1")

    # Setting python variables
    fcDisturbedLand = arcpy.mapping.Layer(globalvars.srcDisturbedLand)
    fcBuffer50m = arcpy.mapping.Layer(globalvars.srcBuffer50m)
    logger.info("feature layers ready")

    ## Create feature class to store intermediate results for Disturb50m
    arcpy.CopyFeatures_management(WetlandPoly,"WU_Disturb50m1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_Disturb50m1", "fcWUDisturb50m1")
    logger.info("feature class WU_Disturb50m1 created")

    # Intersect the 50m buffers and the disturbed land uses
    arInputData = [fcBuffer50m,fcDisturbedLand]
    arcpy.Intersect_analysis(arInputData,"Buffer50mDist","ALL",output_type="INPUT")
    fcBuffer50mDist = arcpy.mapping.Layer(r"Buffer50mDist")
    logger.info("intersected disturbed land uses with 50m buffers")

    # Dissolve disturbed lands by wetland buffer
    arcpy.Dissolve_management(fcBuffer50mDist,"Buffer50mDist_diss","WUKey","BufferArea
FIRST","MULTI_PART","DISSOLVE_LINES")
    fcBuffer50mDist_diss = arcpy.mapping.Layer(r"Buffer50mDist_diss")
    logger.info("dissolved disturbed lands by wetland buffer")

    # Add field and calculate ration of disturbed are to total drainage
    actions.DeleteField(fcBuffer50mDist_diss,"Dist50mRat")
    arcpy.AddField_management(fcBuffer50mDist_diss, "Dist50mRat", "FLOAT")
arcpy.CalculateField_management(fcBuffer50mDist_diss,"Dist50mRat","[Shape_Area]/[FIRST
_BufferArea]","VB","#")
    logger.info("field Dist50mRat added and calculated")

    # Join ratio of Disturbed land to Wetland Units

arcpy.AddJoin_management("fcWUDisturb50m1","WUKey","Buffer50mDist_diss","WUKey",
"KEEP_ALL")
    logger.info("ratio of Disturbed land joined to Wetland Units")

    # Export joined data
    #arcpy.CopyFeatures_management("fcWUDisturb50m1","WU_Disturb50m","#","0","0","0")

arcpy.FeatureClassToFeatureClass_conversion("fcWUDisturb50m1",arcpy.env.workspace,"W
U_Disturb50m")
    fcWUDisturb50m = arcpy.mapping.Layer(r"WU_Disturb50m")
    logger.info("joined data exported")

    ## Remove Join
    arcpy.RemoveJoin_management("fcWUDisturb50m1")
    logger.info("joined removed")

    # Update Null records to 0 in the Dist50mRat field

arcpy.SelectLayerByAttribute_management(fcWUDisturb50m,"NEW_SELECTION","""Buff
er50mDist_diss_Dist50mRat" IS NULL""")

arcpy.CalculateField_management(fcWUDisturb50m,"Buffer50mDist_diss_Dist50mRat","0","
VB","#")
    logger.info("Null records updated to 0 in the Dist50mRat field")
    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(fcWUDisturb50m,"Disturb50m")
    arcpy.AddField_management(fcWUDisturb50m, "Disturb50m", "SHORT")
    arcpy.CalculateField_management(fcWUDisturb50m,"Disturb50m","0","VB","#")
    logger.info("field Disturb50m added to Wetland Units and initial point value set to zero")

    # Assign Points

arcpy.SelectLayerByAttribute_management(fcWUDisturb50m,"NEW_SELECTION","""""Buff
er50mDist_diss_Dist50mRat" > 0.1""")
    arcpy.CalculateField_management(fcWUDisturb50m,"Disturb50m","1","VB","#")



arcpy.SelectLayerByAttribute_management(fcWUDisturb50m,"NEW_SELECTION","""""Buff
er50mDist_diss_Dist50mRat" > 0.25""")
    arcpy.CalculateField_management(fcWUDisturb50m,"Disturb50m","2","VB","#")



arcpy.SelectLayerByAttribute_management(fcWUDisturb50m,"NEW_SELECTION","""""Buff
er50mDist_diss_Dist50mRat" > 0.5""")
    arcpy.CalculateField_management(fcWUDisturb50m,"Disturb50m","3","VB","#")
    logger.info("points assigned to wetland units")
    arcpy.SelectLayerByAttribute_management(fcWUDisturb50m, "CLEAR_SELECTION")

    # Update NULL values with 0

arcpy.SelectLayerByAttribute_management(fcWUDisturb50m,"NEW_SELECTION","""""Distu
rb50m" IS NULL""")
    arcpy.CalculateField_management(fcWUDisturb50m,"Disturb50m","0","VB","#")
    arcpy.SelectLayerByAttribute_management(fcWUDisturb50m, "CLEAR_SELECTION")
    logger.info("Null records updated to 0 in the Disturb50m field")
```

```
# Clean up
if arcpy.Exists("Buffer50mDist"):
    arcpy.Delete_management("Buffer50mDist")
if arcpy.Exists("Buffer50mDist_diss"):
    arcpy.Delete_management("Buffer50mDist_diss")
if arcpy.Exists("WU_Disturb50m1"):
    arcpy.Delete_management("WU_Disturb50m1")
```

5.7.80 DistubWshd: Water Quality Opportunity

```
##############################################################################
#
# File Name: DisturbWshd.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/27/2016 (modified 11/10/2017)
# Purpose:
#    Water Quality Function / Opportunity aspect
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineDisturbWshd(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQOpportun.DisturbWshd")

    # Clean up if needed
    #if arcpy.Exists("DrainAreaDist"):
        #arcpy.Delete_management("DrainAreaDist")
    if arcpy.Exists("DrainAreaDist_diss"):
        arcpy.Delete_management("DrainAreaDist_diss")
    if arcpy.Exists("WU_DisturbWshd"):
        arcpy.Delete_management("WU_DisturbWshd")
    if arcpy.Exists("WU_DisturbWshd1"):
        arcpy.Delete_management("WU_DisturbWshd1")

    # Setting python variables
    fcDA27m = arcpy.mapping.Layer(globalvars.srcDrainageArea)
    fcDL = arcpy.mapping.Layer(globalvars.srcDisturbedLand)
    logger.info("feature layers ready")

    ## Create feature class to store intermediate results for Disturb50m
    arcpy.CopyFeatures_management(WetlandPoly,"WU_DisturbWshd1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_DisturbWshd1", "fcWUDisturbWshd1")
    logger.info("feature class WU_DisturbWshd1 created")

    # Intersect the drainage areas and the disturbed land uses (Takes 32 minutes 23 seconds to
run)
    #arInputData = [fcDA27m,fcDL]
    #arcpy.Intersect_analysis(arInputData,"DrainAreaDist","ALL","#","INPUT")
    fcDrainAreaDist = arcpy.mapping.Layer(r"DrainAreaDist")
    logger.info("drainage areas and the disturbed land uses intersected")

    # Dissolve disturbed lands by drainage area (Takes around 23 hours)
    arcpy.Dissolve_management(fcDrainAreaDist,"DrainAreaDist_diss","WUKey","CntrWshd
FIRST","MULTI_PART","DISSOLVE_LINES")
    fcDrainAreaDistDiss = arcpy.mapping.Layer(r"DrainAreaDist_diss")
    logger.info("disturbed lands dissolved by drainage area")

    # Add field to DrainAreaDist_diss and calculate ratio of disturbed area to total drainange
    actions.DeleteField(fcDrainAreaDistDiss,"DistWshdRat")
    arcpy.AddField_management(fcDrainAreaDistDiss, "DistWshdRat", "FLOAT")
    arcpy.CalculateField_management(fcDrainAreaDistDiss, "DistWshdRat",
"[SHAPE_Area]/[FIRST_CntrWshd]", "VB")
    logger.info("field DistWshdRat added to DrainAreaDist_diss and calculated")

    # Join ratio of disturbed land to Wetland Units

arcpy.AddJoin_management("fcWUDisturbWshd1","WUKey",fcDrainAreaDistDiss,"WUKey"
,"KEEP_ALL")
    logger.info("ratio of disturbed land joined to wetland units")

    # Export joined data
    #arcpy.CopyFeatures_management("fcWUDisturbWshd1", "WU_DisturbWshd")

arcpy.FeatureClassToFeatureClass_conversion("fcWUDisturbWshd1",arcpy.env.workspace,"W
U_DisturbWshd")
    fcDisturbWshd = arcpy.mapping.Layer(r"WU_DisturbWshd")
    logger.info("joined data exported")
    arcpy.RemoveJoin_management("fcWUDisturbWshd1")

    # Set NULL values in the DistWshdRat to 0

arcpy.SelectLayerByAttribute_management(fcDisturbWshd,"NEW_SELECTION",""""DrainAr
eaDist_diss_DistWshdRat" IS NULL""")

arcpy.CalculateField_management(fcDisturbWshd,"DrainAreaDist_diss_DistWshdRat","0","V
B","#")
    logger.info("null values in DistWshdRat set to 0")
    arcpy.SelectLayerByAttribute_management(fcDisturbWshd, "CLEAR_SELECTION")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(fcDisturbWshd,"DisturbWshd")
    arcpy.AddField_management(fcDisturbWshd, "DisturbWshd", "SHORT")
    arcpy.CalculateField_management(fcDisturbWshd, "DisturbWshd", "0", "VB")
    logger.info("field DisturbWshd added to Wetland Units and initial point value set to zero")

    # Assign points

arcpy.SelectLayerByAttribute_management(fcDisturbWshd,"NEW_SELECTION","""""DrainAr
eaDist_diss_DistWshdRat" > 0.1""")
    arcpy.CalculateField_management(fcDisturbWshd, "DisturbWshd", "1", "VB")
    logger.info("points assigned to field DisturbWshd")
    arcpy.SelectLayerByAttribute_management(fcDisturbWshd, "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("DrainAreaDist"):
        arcpy.Delete_management("DrainAreaDist")
    if arcpy.Exists("DrainAreaDist_diss"):
        arcpy.Delete_management("DrainAreaDist_diss")
    if arcpy.Exists("WU_DisturbWshd1"):
        arcpy.Delete_management("WU_DisturbWshd1")
```

5.7.81 ImpairedIn: Water Quality Opportunity

```
##############################################################################
#
# File Name: ImpairedIn.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/10/2016 (modified 12/06/2017)
# Purpose:
#    Water Quality Function, Opportunity Aspect
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineImpairedIn():
    logger = logging.getLogger("WFA.WQuality.WQOpportun.ImpairedIn")

    # Clean up if needed
    if arcpy.Exists("WU_ImpairedIn"):
        arcpy.Delete_management("WU_ImpairedIn")

    # Setting python variables
    fcPublicFishing = arcpy.mapping.Layer(globalvars.srcPublicFishingLakes)
    fcAlgalStreams = arcpy.mapping.Layer(globalvars.srcAlgalStreams)
    fcAlgalLakes = arcpy.mapping.Layer(globalvars.srcAlgalLakes)
    fcDA27m = arcpy.mapping.Layer(globalvars.srcDrainageArea)
    fcWV202IS24kNHD = arcpy.mapping.Layer(globalvars.srcImpairedStreams)
    #fcEPAOverlist24KNHD = arcpy.mapping.Layer(globalvars.srcEPAOverlist)

    # Create feature class to store ImpairedIn variable
    arcpy.CopyFeatures_management(r"WU_FloodArea", "WU_ImpairedIn")
    fcWUImpairedIn = arcpy.mapping.Layer(r"WU_ImpairedIn")
    logger.info("feature layers ready")

    # Add ImpairedIn field to Wetland Units and set the initial value to zero
    actions.DeleteField(fcWUImpairedIn,"ImpairedIn")
    arcpy.AddField_management(fcWUImpairedIn, "ImpairedIn", "SHORT")
    arcpy.CalculateField_management(fcWUImpairedIn, "ImpairedIn", "0", "VB")
    logger.info("field ImpairedIn added and initial value set to 0")

    # Add field to DrainageArea27m and set initial value to "N"
    actions.DeleteField(fcDA27m,"ImpairSrc")
    arcpy.DeleteField_management(fcDA27m,"ImpairSrc")
    arcpy.AddField_management(fcDA27m, "ImpairSrc", "TEXT", 2)
    arcpy.CalculateField_management(fcDA27m,"ImpairSrc","'N'","PYTHON_9.3")
    logger.info("field ImpairSrc added and initial value set to 'N'")

    # Select lakes with power boat use
    strWHERE = """"BoatType" NOT LIKE 'No%'"""
    arcpy.SelectLayerByAttribute_management(fcPublicFishing, "NEW_SELECTION",
strWHERE)
    logger.info("lakes with power boat use selected")

    # Select Drainage Areas with power boat use
```

```
arcpy.SelectLayerByLocation_management(fcDA27m,"INTERSECT",fcPublicFishing,"#","NE
W_SELECTION")
    logger.info("Drainage Areas with power boat use selected")

    # Select Drainage Areas with algal lakes

arcpy.SelectLayerByLocation_management(fcDA27m,"INTERSECT",fcAlgalLakes,"#","ADD
_TO_SELECTION")
    logger.info("Drainage Areas with algal lakes selected")

    # Select Drainage Areas with algal streams

arcpy.SelectLayerByLocation_management(fcDA27m,"INTERSECT",fcAlgalStreams,"#","AD
D_TO_SELECTION")
    logger.info("Drainage Areas with algal streams selected")

    # Select Drainage Area with impaired streams

arcpy.SelectLayerByLocation_management(fcDA27m,"INTERSECT",fcWV202IS24kNHD,"#
","ADD_TO_SELECTION")

#arcpy.SelectLayerByLocation_management(fcDA27m,"INTERSECT",fcEPAOverlist24KNH
D,"#","ADD_TO_SELECTION")
    logger.info("Drainage Area with impaired streams selected")

    # Set ImpairSrc to yes for Drainage Areas with an impaired water source
    arcpy.CalculateField_management(fcDA27m,"ImpairSrc","'Y'","PYTHON_9.3")
    logger.info("ImpairSrc set to yes for Drainage Areas with an impaired water source")

    arcpy.SelectLayerByAttribute_management(fcWUImpairedIn, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcPublicFishing, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcAlgalLakes, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcAlgalStreams, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcWV202IS24kNHD, "CLEAR_SELECTION")

    # Join Drainage Areas to Wetland Units

arcpy.JoinField_management(fcWUImpairedIn,"WUKey",fcDA27m,"WUKey","WUKey;Impa
irSrc")
    logger.info("Drainage Areas joined to Wetland Units")

    # Assign 1 point to floodplain Wetland Units with impaired waters in their Drainage Area
    strWHERE = """"Floodplain" = 'Y' AND "ImpairSrc" = 'Y'"""
    arcpy.SelectLayerByAttribute_management(fcWUImpairedIn, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUImpairedIn, "ImpairedIn", "1", "VB")
    logger.info("1 point assigned to floodplain Wetland Units with impaired waters in their
Drainage Area")

    # Clear selections
    arcpy.SelectLayerByAttribute_management(fcWUImpairedIn, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcDA27m, "CLEAR_SELECTION")

    # Select lakes with power boat use
    strWHERE = """"BoatType" NOT LIKE 'No%'"""
    arcpy.SelectLayerByAttribute_management(fcPublicFishing, "NEW_SELECTION",
strWHERE)
    logger.info("lakes with power boat use selected")

    # Select Wetland Units adjacent to lakes with power boat use
```

```
arcpy.SelectLayerByLocation_management(fcWUImpairedIn,"INTERSECT",fcPublicFishing,"
5 Meters","NEW_SELECTION")
    logger.info("Wetland Units adjacent to lakes with power boat use selected")

    # Select Wetland Units adjacent to algal lakes

arcpy.SelectLayerByLocation_management(fcWUImpairedIn,"INTERSECT",fcAlgalLakes,"5
Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units adjacent to algal lakes selected")

    # Select Wetland Units adjacent to algal streams

arcpy.SelectLayerByLocation_management(fcWUImpairedIn,"INTERSECT",fcAlgalStreams,"
5 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units adjacent to algal streams selected")

    # Select Wetland Units adjacent to impaired streams

arcpy.SelectLayerByLocation_management(fcWUImpairedIn,"INTERSECT",fcWV202IS24kN
HD,"5 Meters","ADD_TO_SELECTION")

#arcpy.SelectLayerByLocation_management(fcWUImpairedIn,"INTERSECT",fcEPAOverlist2
4KNHD,"#","ADD_TO_SELECTION")
    logger.info("Wetland Units adjacent to impaired streams selected")

    # Assign 2 points to floodplain Wetland Units adjacent to impaired waters in their Drainage
Area
    arcpy.CalculateField_management(fcWUImpairedIn,"ImpairedIn","2","PYTHON_9.3")
    logger.info("1 point assigned to floodplain Wetland Units adjacent to impaired waters in their
Drainage Area")

    # Clear selections
```

```
arcpy.SelectLayerByAttribute_management(fcWUImpairedIn, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcPublicFishing, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcAlgalLakes, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcAlgalStreams, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(fcWV202IS24kNHD, "CLEAR_SELECTION")
```

5.7.82 RoadRail: Water Quality Opportunity

```
##############################################################################
#
# File Name: RoadRail.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/15/2016 (modified 11/28/2017)
# Purpose:
#         Water Quality Function, Opportunity aspect
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcRoadRail(WetlandPoly):
logger = logging.getLogger("WFA.WQuality.WQOpportun.RoadRail")

# Clean up if needed
if arcpy.Exists("WU_RoadRail"):
        arcpy.Delete_management("WU_RoadRail")

WVRailway = arcpy.mapping.Layer(globalvars.srcRailway)
WVTransUTMAllRoads = arcpy.mapping.Layer(globalvars.srcTransUTMAllRoads)
```

```
# Create feature class to store RoadRail variable
arcpy.CopyFeatures_management(WetlandPoly, "WU_RoadRail")
WURoadRail = arcpy.mapping.Layer(r"WU_RoadRail")
logger.info("feature layers ready")

# Add RoadRail field to Wetland Units and set initial point value to zero
actions.DeleteField(WURoadRail,"RoadRail")
arcpy.AddField_management(WURoadRail, "RoadRail", "SHORT")
arcpy.CalculateField_management(WURoadRail,"RoadRail","0","VB","#")
logger.info("RoadRail field added and initial value set to 0")

# Select the Wetland Units withing 50 meters of a road of railroad track and assign 1 points
arcpy.SelectLayerByLocation_management(WURoadRail,"WITHIN_A_DISTANCE",WVTransUTMAllRoads,"50 Meters","NEW_SELECTION")
arcpy.SelectLayerByLocation_management(WURoadRail,"WITHIN_A_DISTANCE",WVRailway,"50 Meters","ADD_TO_SELECTION")
arcpy.CalculateField_management(WURoadRail,"RoadRail","1","VB","#")
logger.info("1 point assigned to certain wetland units")
arcpy.SelectLayerByAttribute_management(WURoadRail, "CLEAR_SELECTION")

# Select the Wetland Units withing 5 meters of a road of railroad track and assign 2 points
arcpy.SelectLayerByLocation_management(WURoadRail,"WITHIN_A_DISTANCE",WVTransUTMAllRoads,"5 Meters","NEW_SELECTION")
arcpy.SelectLayerByLocation_management(WURoadRail,"WITHIN_A_DISTANCE",WVRailway,"5 Meters","ADD_TO_SELECTION")
arcpy.CalculateField_management(WURoadRail,"RoadRail","2","VB","#")
logger.info("2 points assigned to certain wetland units")
arcpy.SelectLayerByAttribute_management(WURoadRail, "CLEAR_SELECTION")
```

5.7.83 Water Quality Potential

```
##############################################################################
#
# File Name: WQPotential.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Water Quality Potential metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import Clay, IrrEdge, LandPos, LowSlope, Microtopo, Organic, SeasonPond,
SLOPE, VegByLP, VegPerUng, VegWoody, WFlowPath
from Factors import ChemTime, ClayOrganic, Depressions, Headwater, SWoutflow, VegWQ
from Aspects import WQPotential

def procWQPotential(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential")

    ##############################################################
    ## 1. Run Variables
    ##############################################################

    Clay.Clay(WetlandPoly)

    IrrEdge.CalcIrrEdge(WetlandPoly)
```

```
WFlowPath.DetermineWFlowPath(WetlandPoly)

LandPos.DetermineLandPos(WetlandPoly)

SLOPE.CalcSLOPE(WetlandPoly)

LowSlope.CalcLowSlope(WetlandPoly)

Microtopo.MicroTopo(WetlandPoly)

Organic.OrganicFactor(WetlandPoly)

SeasonPond.CalcSeasonPond(WetlandPoly)

VegByLP.CalcVegByLP(WetlandPoly)

VegPerUng.CalcVegPerUng(WetlandPoly)

VegWoody.CalcVegWoody(WetlandPoly)

##########################################################
## 2. Run Factors
##########################################################

ChemTime.CalcChemTime(WetlandPoly)

ClayOrganic.DetermineClayOrganic(WetlandPoly)
```

```
Depressions.CalcDepressions(WetlandPoly)

Headwater.DetermineHW(WetlandPoly)

SWoutflow.DetermineSWoutflow(WetlandPoly)

VegWQ.CalcVegWQ(WetlandPoly)

############################################################
## 3. Run Aspect
############################################################

WQPotential.DetermineWQPotential(WetlandPoly)
```

5.7.84 Water Quality Potential Aspects

```
##########################################################################
#
# File Name: WQPotential.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/16/2016 (modified 11/02/2017)
# Purpose:
#    Water Quality Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineWQPotential(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.WQPotential")

    # Clean up if needed
    if arcpy.Exists("WU_WQPotential1"):
        arcpy.Delete_management("WU_WQPotential1")
    if arcpy.Exists("WU_WQPotential2"):
        arcpy.Delete_management("WU_WQPotential2")
    if arcpy.Exists("WU_WQPotential3"):
        arcpy.Delete_management("WU_WQPotential3")
    if arcpy.Exists("WU_WQPotential4"):
        arcpy.Delete_management("WU_WQPotential4")
    if arcpy.Exists("WU_WQPotential"):
        arcpy.Delete_management("WU_WQPotential")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_ChemTime","fcChemTime")
    arcpy.MakeFeatureLayer_management(r"WU_ClayOrganic","fcClayOrganic")
    arcpy.MakeFeatureLayer_management(r"WU_Depressions","fcDepressions")
    arcpy.MakeFeatureLayer_management(r"WU_Headwater","fcHeadwater")
    arcpy.MakeFeatureLayer_management(r"WU_SWOutflow","fcSWOutflow")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ","fcVegWQ")
    logger.info("feature layers ready")

    # Spatial join to bring together the factor values
##############################################################################
#################
    # SJ: ChemTime & ClayOrganic
##############################################################################
#################
    fmSJCC = arcpy.FieldMappings()
    fmSJCC.addTable("fcChemTime")
    fmSJCC.addTable("fcClayOrganic")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","ChemTime","ClayOrganic"]

    for field in fmSJCC.fields:
        if field.name not in keepers:
            fmSJCC.removeFieldMap(fmSJCC.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcChemTime","fcClayOrganic","WU_WQPotential1","JOIN_ONE
_TO_ONE","KEEP_ALL",fmSJCC,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQPotential1","fcWQPotential1")
    logger.info("Spatial Join completed to add variables ChemTime and ClayOrganic")


################################################################################
#################
    # SJ: Depression

################################################################################
#################
    fmSJD = arcpy.FieldMappings()
    fmSJD.addTable("fcWQPotential1")
    fmSJD.addTable("fcDepressions")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","ChemTime","ClayOrganic","Depressions"]

    for field in fmSJD.fields:
        if field.name not in keepers:
            fmSJD.removeFieldMap(fmSJD.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQPotential1","fcDepressions","WU_WQPotential2","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJD,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQPotential2","fcWQPotential2")
    logger.info("Spatial Join completed to add variable Depression")
```

```
##################################################################################
#################
    # SJ: Headwater

##################################################################################
#################
    fmSJH = arcpy.FieldMappings()
    fmSJH.addTable("fcWQPotential2")
    fmSJH.addTable("fcHeadwater")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","ChemTime","ClayOrganic","Depressions","Headw
ater"]

    for field in fmSJH.fields:
        if field.name not in keepers:
            fmSJH.removeFieldMap(fmSJH.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQPotential2","fcHeadwater","WU_WQPotential3","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJH,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQPotential3","fcWQPotential3")
    logger.info("Spatial Join completed to add variable Depression")


##################################################################################
#################
    # SJ: SWOutFlow

##################################################################################
#################
```

```
    fmSJSW = arcpy.FieldMappings()
    fmSJSW.addTable("fcWQPotential3")
    fmSJSW.addTable("fcSWOutflow")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","ChemTime","ClayOrganic","Depressions","Headw
ater","SWOutflow"]

    for field in fmSJSW.fields:
        if field.name not in keepers:
            fmSJSW.removeFieldMap(fmSJSW.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQPotential3","fcSWOutflow","WU_WQPotential4","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJSW,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQPotential4","fcWQPotential4")
    logger.info("Spatial Join completed to add variable SWOutflow")


##############################################################################
#################
    # SJ: VegWQ

##############################################################################
#################
    fmSJV = arcpy.FieldMappings()
    fmSJV.addTable("fcWQPotential4")
    fmSJV.addTable("fcVegWQ")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","ChemTime","ClayOrganic","Depressions","Headw
ater","SWOutflow","VegWQ"]

    for field in fmSJV.fields:
        if field.name not in keepers:
            fmSJV.removeFieldMap(fmSJV.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWQPotential4","fcVegWQ","WU_WQPotential","JOIN_ONE_T
O_ONE","KEEP_ALL",fmSJV,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQPotential","fcWQPotential")
    logger.info("Spatial Join completed to add variable VegWQ")

    # Add WQPotential field to Wetland Units and set the initial point value to zero
    actions.DeleteField("fcWQPotential","WQPotential")
    arcpy.AddField_management("fcWQPotential", "WQPotential", "SHORT")
    arcpy.CalculateField_management("fcWQPotential","WQPotential","0","VB")
    logger.info("WQPotential field added to Wetland Units and initial point values set to zero")

    # Sum the factor points
    arcpy.CalculateField_management("fcWQPotential","WQPotential","[ChemTime] +
[ClayOrganic] + [Depressions] + [Headwater] + [SWOutflow] + [VegWQ]","VB","#")
    logger.info("points summed for WQPotential")

    # Clean up
    if arcpy.Exists("WU_WQPotential1"):
        arcpy.Delete_management("WU_WQPotential1")
    if arcpy.Exists("WU_WQPotential2"):
        arcpy.Delete_management("WU_WQPotential2")
    if arcpy.Exists("WU_WQPotential3"):
```

```
    arcpy.Delete_management("WU_WQPotential3")
if arcpy.Exists("WU_WQPotential4"):
    arcpy.Delete_management("WU_WQPotential4")
```

5.7.85 ChemTime: Water Quality Potential

```
################################################################################
#
# File Name: ChemTime.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/21/2016 (modified 12/05/2017)
# Purpose:
#    Input to Water Quality
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcChemTime(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.ChemTime")

    # Clean up if needed
    if arcpy.Exists("WU_ChemTime"):
        arcpy.Delete_management("WU_ChemTime")
    if arcpy.Exists("WU_ChemTime0"):
        arcpy.Delete_management("WU_ChemTime0")
    if arcpy.Exists("WU_ChemTime1"):
        arcpy.Delete_management("WU_ChemTime1")
    if arcpy.Exists("WU_ChemTime2"):
        arcpy.Delete_management("WU_ChemTime2")
    if arcpy.Exists("WU_ChemTime3"):
        arcpy.Delete_management("WU_ChemTime3")

    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    arcpy.MakeFeatureLayer_management(r"WU_SeasonPond","fcSeasonPond")
    arcpy.MakeFeatureLayer_management(r"WU_SLOPE","fcSLOPE")
    arcpy.MakeFeatureLayer_management(r"WU_IrrEdge","fcIrrEdge")

    # Create feature class to store ChemTime variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_ChemTime0")
    arcpy.MakeFeatureLayer_management(r"WU_ChemTime0","fcWUChemTime0")
    logger.info("feature layers ready")


################################################################################
#################
    # SJ: Floodplain

################################################################################
#################
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("fcWUChemTime0")
    fmSJFP.addTable("fcFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain"]

    for field in fmSJFP.fields:
        if field.name not in keepers:
            fmSJFP.removeFieldMap(fmSJFP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUChemTime0","fcFloodplain","WU_ChemTime1","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJFP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ChemTime1","fcWUChemTime1")
    logger.info("spatial join completed to add field Floodplain")


##############################################################################
#################
    # SJ: SeasonPond

##############################################################################
#################
    fmSJSP = arcpy.FieldMappings()
    fmSJSP.addTable("fcWUChemTime1")
    fmSJSP.addTable("fcSeasonPond")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","SeasonPond"]

    for field in fmSJSP.fields:
        if field.name not in keepers:
            fmSJSP.removeFieldMap(fmSJSP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUChemTime1","fcSeasonPond","WU_ChemTime2","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJSP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ChemTime2","fcWUChemTime2")
    logger.info("spatial join completed to add field SeasonPond")
```

```
################################################################################
#################
   # SJ: SLOPE

################################################################################
#################
   fmSJSL = arcpy.FieldMappings()
   fmSJSL.addTable("fcWUChemTime2")
   fmSJSL.addTable("fcSLOPE")

   keepers = []
   keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","SeasonPond","SLOPE"]

   for field in fmSJSL.fields:
      if field.name not in keepers:
         fmSJSL.removeFieldMap(fmSJSL.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUChemTime2","fcSLOPE","WU_ChemTime3","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJSL,"CONTAINS")
   arcpy.MakeFeatureLayer_management("WU_ChemTime3","fcWUChemTime3")
   logger.info("spatial join completed to add field SLOPE")


################################################################################
#################
   # SJ: IrrEdge

################################################################################
#################
   fmSJIR = arcpy.FieldMappings()
```

```
    fmSJIR.addTable("fcWUChemTime3")
    fmSJIR.addTable("fcIrrEdge")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Floodplain","SeasonPond","SLOPE","IrrEdge"]

    for field in fmSJIR.fields:
        if field.name not in keepers:
            fmSJIR.removeFieldMap(fmSJIR.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUChemTime3","fcIrrEdge","WU_ChemTime","JOIN_ONE_T
O_ONE","KEEP_ALL",fmSJIR,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ChemTime","fcWUChemTime")
    logger.info("spatial join completed to add field IrrEdge")

    # Add field to Wetland Units and set initial point value to SeasonPond
    actions.DeleteField("fcWUChemTime","ChemTime")
    arcpy.AddField_management("fcWUChemTime", "ChemTime", "SHORT")

arcpy.CalculateField_management("fcWUChemTime","ChemTime","[SeasonPond]","VB","#")
    logger.info("field ChemTime added and initial value set to SeasonPond")

    # Filter Seasonal Ponding (SeasonPond) points based on Slope and Floodplain
    strWHERE = """"SLOPE" > 5 OR "Floodplain" = 'Y'"""
    arcpy.SelectLayerByAttribute_management("fcWUChemTime", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUChemTime","ChemTime","0","VB","#")

    strWHERE = """"SLOPE" > 2 AND"SLOPE" < 6 AND "ChemTime" > 2"""
    arcpy.SelectLayerByAttribute_management("fcWUChemTime", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUChemTime","ChemTime","2","VB","#")
    logger.info("points assigned to ChemTime")

    # Add point for irregular edge (IrrEdge)
    strWHERE = """"IrrEdge" = 1 AND "Floodplain" = 'N' AND "ChemTime" < 3"""
    arcpy.SelectLayerByAttribute_management("fcWUChemTime", "NEW_SELECTION",
strWHERE)

arcpy.CalculateField_management("fcWUChemTime","ChemTime","[ChemTime]+1","VB","#
")
    logger.info("point added for irregular edge")

    arcpy.SelectLayerByAttribute_management("fcWUChemTime", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("WU_ChemTime0"):
        arcpy.Delete_management("WU_ChemTime0")
    if arcpy.Exists("WU_ChemTime1"):
        arcpy.Delete_management("WU_ChemTime1")
    if arcpy.Exists("WU_ChemTime2"):
        arcpy.Delete_management("WU_ChemTime2")
    if arcpy.Exists("WU_ChemTime3"):
        arcpy.Delete_management("WU_ChemTime3")
```

5.7.86 ClayOrganic: Water Quality Potential

```
##############################################################################
#
# File Name: ClayOrganic.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date 4/5/2016 (modified 12/08/2017)
# Purpose:
#    Input to Water Quality
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineClayOrganic(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.ClayOrganics")

    # Clean up if needed
    if arcpy.Exists("WU_ClayOrganic"):
        arcpy.Delete_management("WU_ClayOrganic")
    if arcpy.Exists("WU_ClayOrganic0"):
        arcpy.Delete_management("WU_ClayOrganic0")
    if arcpy.Exists("WU_ClayOrganic1"):
        arcpy.Delete_management("WU_ClayOrganic1")
    if arcpy.Exists("WU_ClayOrganic2"):
        arcpy.Delete_management("WU_ClayOrganic2")
    if arcpy.Exists("WU_ClayOrganic3"):
        arcpy.Delete_management("WU_ClayOrganic3")

    # Setting environment variables
    arcpy.MakeFeatureLayer_management(r"WU_Organic","fcOrganic")
    arcpy.MakeFeatureLayer_management(r"WU_Clay","fcClay")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcWUFloodplain")
    arcpy.MakeFeatureLayer_management(r"WU_SeasonPond","fcWUSeasonPond")

    arcpy.CopyFeatures_management(WetlandPoly, "WU_ClayOrganic0")
    arcpy.MakeFeatureLayer_management(r"WU_ClayOrganic0","fcWUClayOrganic0")
    logger.info("feature layers ready")

    # Join Clay and Organic to combine the two fields into one feature class called
WU_ClayOrganic

################################################################################
#################
    # SJ: Organic

################################################################################
#################
    fmSJORG = arcpy.FieldMappings()
    fmSJORG.addTable("fcWUClayOrganic0")
    fmSJORG.addTable("fcOrganic")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Organic"]
    for field in fmSJORG.fields:
        if field.name not in keepers:
            fmSJORG.removeFieldMap(fmSJORG.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUClayOrganic0","fcOrganic","WU_ClayOrganic1","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJORG,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ClayOrganic1","fcWUClayOrganic1")
    logger.info("spatial join completed to add field Organic")


################################################################################
#################
    # SJ: Clay

################################################################################
#################
    fmSJClay = arcpy.FieldMappings()
    fmSJClay.addTable("fcWUClayOrganic1")
    fmSJClay.addTable("fcClay")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Organic","Clay"]

    for field in fmSJClay.fields:
        if field.name not in keepers:
            fmSJClay.removeFieldMap(fmSJClay.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUClayOrganic1","fcClay","WU_ClayOrganic2","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJClay,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ClayOrganic2","fcWUClayOrganic2")
    logger.info("spatial join completed to add field Clay")


#################################################################################
#################
    # SJ: Floodplain

#################################################################################
#################
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("fcWUClayOrganic2")
    fmSJFP.addTable("fcWUFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Organic","Clay","Floodplain"]

    for field in fmSJClay.fields:
        if field.name not in keepers:
            fmSJClay.removeFieldMap(fmSJClay.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUClayOrganic2","fcWUFloodplain","WU_ClayOrganic3","JO
IN_ONE_TO_ONE","KEEP_ALL",fmSJFP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ClayOrganic3","fcWUClayOrganic3")
    logger.info("spatial join completed to add field Floodplain")


#################################################################################
#################
    # SJ: SeasonPond
```

```
##############################################################################
#################
    fmSJSP = arcpy.FieldMappings()
    fmSJSP.addTable("fcWUClayOrganic3")
    fmSJSP.addTable("fcWUSeasonPond")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Organic","Clay","Floodplain","SeaPondRatio","Se
asonPond"]

    for field in fmSJSP.fields:
        if field.name not in keepers:
            fmSJSP.removeFieldMap(fmSJSP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUClayOrganic3","fcWUSeasonPond","WU_ClayOrganic","JO
IN_ONE_TO_ONE","KEEP_ALL",fmSJSP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_ClayOrganic","fcWUClayOrganic")
    logger.info("spatial join completed to add field SeasonPond")

    # Add ClayOrganic field to Wetland Units and set initial point value to zero
    actions.DeleteField("fcWUClayOrganic","ClayOrganic")
    arcpy.AddField_management("fcWUClayOrganic", "ClayOrganic", "SHORT")
    arcpy.CalculateField_management("fcWUClayOrganic","ClayOrganic","0","VB","#")
    logger.info("field ClayOrganic added to Wetland Units and initial point value set to zero")

    # Assign points to wetland units not in floodplain with clay or organic soils and seasonal
ponding
    strwhere = """"Floodplain" = 'N' and ( "Clay" = 'Y' or "Organic" = 'Y') and "SeaPondRatio" >
0.1"""
```

```
arcpy.SelectLayerByAttribute_management("fcWUClayOrganic","NEW_SELECTION",strwhe
re)
    arcpy.CalculateField_management("fcWUClayOrganic","ClayOrganic","1","python","#")

    strwhere = """"Floodplain" = 'N' and ( "Clay" = 'Y' or "Organic" = 'Y') and "SeaPondRatio" >
0.5"""

arcpy.SelectLayerByAttribute_management("fcWUClayOrganic","NEW_SELECTION",strwhe
re)
    arcpy.CalculateField_management("fcWUClayOrganic","ClayOrganic","2","python","#")

    strwhere = """"Floodplain" = 'N' and ( "Clay" = 'Y' or "Organic" = 'Y') and "SeaPondRatio" >
0.9"""

arcpy.SelectLayerByAttribute_management("fcWUClayOrganic","NEW_SELECTION",strwhe
re)
    arcpy.CalculateField_management("fcWUClayOrganic","ClayOrganic","3","python","#")
    logger.info("points 1, 2, or 3 assigned to wetland units not in floodplain with clay or organic
soils and seasonal ponding")

    arcpy.SelectLayerByAttribute_management("fcWUClayOrganic", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("WU_ClayOrganic0"):
        arcpy.Delete_management("WU_ClayOrganic0")
    if arcpy.Exists("WU_ClayOrganic1"):
        arcpy.Delete_management("WU_ClayOrganic1")
    if arcpy.Exists("WU_ClayOrganic2"):
        arcpy.Delete_management("WU_ClayOrganic2")
    if arcpy.Exists("WU_ClayOrganic3"):
        arcpy.Delete_management("WU_ClayOrganic3")
```

5.7.87 Depressions: Water Quality Potential

```
##############################################################################
#
# File Name: Depressions.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/21/2016 (modified 11/29/2017)
# Purpose:
#    Input to Water Quality; Max 5 points, floodplain wetlands only.
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcDepressions(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.Depressions")

    # Clean up if needed
    if arcpy.Exists("WU_Depressions"):
        arcpy.Delete_management("WU_Depressions")
    if arcpy.Exists("WU_Depressions0"):
        arcpy.Delete_management("WU_Depressions0")
    if arcpy.Exists("WU_Depressions1"):
        arcpy.Delete_management("WU_Depressions1")
    if arcpy.Exists("WU_Depressions2"):
        arcpy.Delete_management("WU_Depressions2")
    if arcpy.Exists("WU_Depressions3"):
        arcpy.Delete_management("WU_Depressions3")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    arcpy.MakeFeatureLayer_management(r"WU_Microtopo","fcMicrotopo")
    arcpy.MakeFeatureLayer_management(r"WU_LowSlope","fcLowSlope")
    arcpy.MakeFeatureLayer_management(r"WU_IrrEdge","fcIrrEdge")
    logger.info("feature layers ready")

    # Create feature class to store Depressions variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_Depressions0")
    arcpy.MakeFeatureLayer_management(r"WU_Depressions0","fcWUDepressions0")
    logger.info("feature copied to store Depressions variable")

    # Spatial join to add input variables to attribute table
###############################################################################
#################
    # SJ: Floodplain
###############################################################################
#################
    fmSJWPO0 = arcpy.FieldMappings()
    fmSJWPO0.addTable("fcWUDepressions0")
    fmSJWPO0.addTable("fcFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain"]
    for field in fmSJWPO0.fields:
        if field.name not in keepers:
            fmSJWPO0.removeFieldMap(fmSJWPO0.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUDepressions0","fcFloodplain","WU_Depressions1","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJWPO0,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_Depressions1","fcWUDepressions1")
    logger.info("Spatial join to add input Floodplain to attribute table completed")


################################################################################
#################
    # SJ: Microtopo

################################################################################
#################
    fmSJWPO1 = arcpy.FieldMappings()
    fmSJWPO1.addTable("fcWUDepressions1")
    fmSJWPO1.addTable("fcMicrotopo")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","Microtopo"]

    for field in fmSJWPO1.fields:
        if field.name not in keepers:
            fmSJWPO1.removeFieldMap(fmSJWPO1.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUDepressions1","fcMicrotopo","WU_Depressions2","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJWPO1,"CONTAINS")
```

```
    arcpy.MakeFeatureLayer_management("WU_Depressions2","fcWUDepressions2")
    logger.info("Spatial join to add input Microtopo to attribute table completed")


################################################################################
#################
    # SJ: LowSlope
################################################################################
#################
    fmSJWPO2 = arcpy.FieldMappings()
    fmSJWPO2.addTable("fcWUDepressions2")
    fmSJWPO2.addTable("fcLowSlope")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","Floodplain","Microtopo","LowSlope"]

    for field in fmSJWPO2.fields:
        if field.name not in keepers:
            fmSJWPO2.removeFieldMap(fmSJWPO2.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUDepressions2","fcLowSlope","WU_Depressions3","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJWPO2,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_Depressions3","fcWUDepressions3")
    logger.info("Spatial join to add input LowSlope to attribute table completed")


################################################################################
#################
    # SJ: IrrEdge
```

```
################################################################################
#################
    fmSJWPO3 = arcpy.FieldMappings()
    fmSJWPO3.addTable("fcWUDepressions3")
    fmSJWPO3.addTable("fcIrrEdge")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","Floodplain","Microtopo","LowSlope","IrrEdge"]

    for field in fmSJWPO3.fields:
        if field.name not in keepers:
            fmSJWPO3.removeFieldMap(fmSJWPO3.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUDepressions3","fcIrrEdge","WU_Depressions","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJWPO3,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_Depressions","fcWUDepressions")
    logger.info("Spatial join to add input IrrEdge to attribute table completed")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField("fcWUDepressions","Depressions")
    arcpy.AddField_management("fcWUDepressions", "Depressions", "SHORT")
    arcpy.CalculateField_management("fcWUDepressions","Depressions","0","VB","#")
    logger.info("field Depressions added and initial value set to 0")

    # Sum points for surface depression (Depressions) in floodplain Wetland Units

arcpy.SelectLayerByAttribute_management("fcWUDepressions","NEW_SELECTION","""""Fl
oodplain" = 'Y'""")
```

```
arcpy.CalculateField_management("fcWUDepressions","Depressions","[Microtopo]+[LowSlop
e]+[IrrEdge]","VB","#")
    logger.info("field Depressions calculated")
    arcpy.SelectLayerByAttribute_management("fcWUDepressions", "CLEAR_SELECTION")

    ## Clean up
    if arcpy.Exists("WU_Depressions0"):
        arcpy.Delete_management("WU_Depressions0")
    if arcpy.Exists("WU_Depressions1"):
        arcpy.Delete_management("WU_Depressions1")
    if arcpy.Exists("WU_Depressions2"):
        arcpy.Delete_management("WU_Depressions2")
    if arcpy.Exists("WU_Depressions3"):
        arcpy.Delete_management("WU_Depressions3")
```

5.7.89 Headwater: Water Quality Potential

```
##########################################################################
#
# File Name: Headwater.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/15/2016 (modified 11/28/2017)
# Purpose:
#    Input to Water Quality (Potential) and Flood Attenuation (Potential) functions
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineHW(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.Headwater")

    # Clean up if needed
    if arcpy.Exists("WU_Headwater"):
        arcpy.Delete_management("WU_Headwater")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(r"WU_LandPos","fcWULandPos")
    logger.info("feature layers ready")

    # Spatial join to add input variable to attribute table
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWU")
    fmSJ.addTable("fcWULandPos")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","LandPos"]

    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWU","fcWULandPos","WU_Headwater","JOIN_ONE_TO_ONE
","KEEP_ALL",fmSJ,"CONTAINS")
    logger.info("spatial join completed to add input variable to attribute table")
    fcWUHeadwater = arcpy.mapping.Layer(r"WU_Headwater")

    # Add Headwater field to Wetland Units and set the inital point value to zero
    actions.DeleteField(fcWUHeadwater,"Headwater")
    arcpy.AddField_management(fcWUHeadwater, "Headwater", "SHORT")
    arcpy.CalculateField_management(fcWUHeadwater, "Headwater", "0", "PYTHON")
    logger.info("field Headwater added and initial point value set to 0")

    # Assin 1 point to Wetland Units with Landscape Position headwater modifier

arcpy.SelectLayerByAttribute_management(fcWUHeadwater,"NEW_SELECTION","""""LandP
os" LIKE '%h'""")
    arcpy.CalculateField_management(fcWUHeadwater,"Headwater","1","PYTHON","#")
```

```
logger.info("point assigned to Wetland Units with Landscape Position headwater modifier")
arcpy.SelectLayerByAttribute_management(fcWUHeadwater, "CLEAR_SELECTION")
```

5.7.90 SWoutflow: Water Quality Potential

```
##########################################################################
#
# File Name: SWoutflow.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date 6/14/2016 (modified 12/05/2017)
# Purpose:
#    SWOutflow: Input to Water Quality. / SWOutflow2: Input to Flood Attenuation.
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineSWoutflow(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.SWoutflow")

    # Clean up if needed
    if arcpy.Exists("WU_SWOutflow"):
        arcpy.Delete_management("WU_SWOutflow")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_WFlowPath","fcWFlowPath")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcWUFloodplain")
    logger.info("feature layers ready")



##################################################################################
################
    # Spatial join to add input variables to attribute table

##################################################################################
################
    fmSJ2 = arcpy.FieldMappings()
    fmSJ2.addTable("fcWFlowPath")
    fmSJ2.addTable("fcWUFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WFlowPath","Floodplain"]

    for field in fmSJ2.fields:
        if field.name not in keepers:
            fmSJ2.removeFieldMap(fmSJ2.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWFlowPath","fcWUFloodplain","WU_SWOutflow","JOIN_ON
E_TO_ONE","KEEP_ALL",fmSJ2,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_SWOutflow","fcWUSWoutflow")
    logger.info("spatial join to add WFlowPath and Floodplain completed")



##################################################################################
################
    # Add SWOutflow fields to Wetland Units and set inital value to zero
```

```
##############################################################################
################
    actions.DeleteField("fcWUSWoutflow","SWOutflow")
    arcpy.AddField_management("fcWUSWoutflow", "SWOutflow", "SHORT")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow","0","PYTHON_9.3")
    logger.info("field SWOutflow added and initial value set to 0")

    actions.DeleteField("fcWUSWoutflow","SWOutflow2")
    arcpy.AddField_management("fcWUSWoutflow", "SWOutflow2", "SHORT")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow2","0","PYTHON_9.3")
    logger.info("field SWOutflow2 added and initial value set to 0")


##############################################################################
################
    # Assign points for to SWoutflow
##############################################################################
################

arcpy.SelectLayerByAttribute_management("fcWUSWoutflow","NEW_SELECTION","""""Flo
odplain" = 'N'""")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow","1","VB","#")


arcpy.SelectLayerByAttribute_management("fcWUSWoutflow","NEW_SELECTION","""""Flo
odplain" = 'N' AND "WFlowPath" IN ('OI','TI','BI','IB')""")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow","3","VB","#")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow2","1","VB","#")
arcpy.SelectLayerByAttribute_management("fcWUSWoutflow","NEW_SELECTION","""""Flo
odplain" = 'N' AND "WFlowPath" IN ('IS')""")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow","4","VB","#")
    arcpy.CalculateField_management("fcWUSWoutflow","SWOutflow2","2","VB","#")
    logger.info("points assigned to SWOutflow and SWOutflow2")
```

5.7.91 WegWQ: Water Quality Potential

```
##############################################################################
#
# File Name: VegWQ.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/23/2016 (modified 11/02/2017)
# Purpose:
#    Input to Water Quality
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegWQ(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.VegWQ")

    #clean up if needed
    if arcpy.Exists("WU_VegWQ0"):
        arcpy.Delete_management("WU_VegWQ0")
    if arcpy.Exists("WU_VegWQ1"):
        arcpy.Delete_management("WU_VegWQ1")
    if arcpy.Exists("WU_VegWQ2"):
        arcpy.Delete_management("WU_VegWQ2")
    if arcpy.Exists("WU_VegWQ3"):
        arcpy.Delete_management("WU_VegWQ3")
    if arcpy.Exists("WU_VegWQ"):
        arcpy.Delete_management("WU_VegWQ")

    arcpy.MakeFeatureLayer_management(r"WU_VegPerUng","fcVegPerUng")
    arcpy.MakeFeatureLayer_management(r"WU_VegWoody","fcVegWoody")
    arcpy.MakeFeatureLayer_management(r"WU_VegByLP","fcVegByLP")
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcFloodplain")
    logger.info("feature layers ready")

    # Create feature class to store VegWQ variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_VegWQ0")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ0","WUVegWQ0")
    logger.info("feature class WU_VegWQ0 created")

    # Add VegWQ field to Wetland Units and set initial point value to zero
    actions.DeleteField("WUVegWQ0","VegWQ")
    arcpy.AddField_management("WUVegWQ0", "VegWQ", "SHORT")
    arcpy.CalculateField_management("WUVegWQ0","VegWQ","0","VB","#")
    logger.info("field VegWQ added to Wetland Units and initial point value set to zero")


##################################################################################
#################
    # SJ: VegPerUng

##################################################################################
#################
    fmSJVPU = arcpy.FieldMappings()
    fmSJVPU.addTable("WUVegWQ0")
    fmSJVPU.addTable("fcVegPerUng")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","VegWQ","VegPerUng"]

    for field in fmSJVPU.fields:
        if field.name not in keepers:
            fmSJVPU.removeFieldMap(fmSJVPU.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WUVegWQ0","fcVegPerUng","WU_VegWQ1","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJVPU,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ1","WUVegWQ1")
    logger.info("spatial join for VegPerUng completed")

    ## Spatial join to add input variables to attribute table

##############################################################################
#################
    # SJ: VegWoody (3 NULL Records)

##############################################################################
#################
    fmSJVW = arcpy.FieldMappings()
    fmSJVW.addTable("WUVegWQ1")
    fmSJVW.addTable("fcVegWoody")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","VegWQ","VegPerUng","VegWoody"]

    for field in fmSJVW.fields:
        if field.name not in keepers:
            fmSJVW.removeFieldMap(fmSJVW.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WUVegWQ1","fcVegWoody","WU_VegWQ2","JOIN_ONE_TO_
ONE","KEEP_ALL",fmSJVW,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ2","WUVegWQ2")
    logger.info("spatial join for VegWoody completed")


################################################################################
#################
    # SJ: VegByLP (10776 NULL Records)

################################################################################
#################
    fmSJVLP = arcpy.FieldMappings()
    fmSJVLP.addTable("WUVegWQ2")
    fmSJVLP.addTable("fcVegByLP")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","VegWQ","VegPerUng","VegWoody","VegByLP"]

    for field in fmSJVLP.fields:
        if field.name not in keepers:
            fmSJVLP.removeFieldMap(fmSJVLP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WUVegWQ2","fcVegByLP","WU_VegWQ3","JOIN_ONE_TO_
ONE","KEEP_ALL",fmSJVLP,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ3","WUVegWQ3")
    logger.info("spatial join for VegByLP completed")


################################################################################
#################
    # SJ: Floodplain

################################################################################
#################
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("WUVegWQ3")
    fmSJFP.addTable("fcFloodplain")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","VegWQ","VegPerUng","VegWoody","VegByLP",
"Floodplain"]

    for field in fmSJFP.fields:
        if field.name not in keepers:
            fmSJFP.removeFieldMap(fmSJFP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("WUVegWQ3","fcFloodplain","WU_VegWQ","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJFP,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_VegWQ","WUVegWQ")
    logger.info("spatial join for Floodplain completed")

    # Sum the point for VegPerUng, VegWoody, and VegByLP
    arcpy.CalculateField_management("WUVegWQ","VegWQ","[VegPerUng] + [VegWoody]
+ [VegByLP]","VB","#")
    logger.info("the point summed for VegPerUng, VegWoody, and VegByLP")
    # Reduce any excess point scores to the maxiumim allowed
    arcpy.SelectLayerByAttribute_management("WUVegWQ", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("WUVegWQ","NEW_SELECTION","""""VegWQ
" > 10""")
    arcpy.CalculateField_management("WUVegWQ","VegWQ","10","VB","#")


    arcpy.SelectLayerByAttribute_management("WUVegWQ", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("WUVegWQ","NEW_SELECTION","""""VegWQ
" > 5 AND "Floodplain" = 'N'""")
    arcpy.CalculateField_management("WUVegWQ","VegWQ","5","VB","#")
    logger.info("excess point scores reduced to the maxiumim allowed")


    arcpy.SelectLayerByAttribute_management("WUVegWQ", "CLEAR_SELECTION")


    # Clean up
    if arcpy.Exists("WU_VegWQ0"):
        arcpy.Delete_management("WU_VegWQ0")
    if arcpy.Exists("WU_VegWQ1"):
        arcpy.Delete_management("WU_VegWQ1")
    if arcpy.Exists("WU_VegWQ2"):
        arcpy.Delete_management("WU_VegWQ2")
    if arcpy.Exists("WU_VegWQ3"):
        arcpy.Delete_management("WU_VegWQ3")
```

5.7.92 Clay: Water Quality Potential

```
##############################################################################
#
# File Name: Clay.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/2/2016 (modified 11/16/2017)
# Purpose:
#    Input to Water Quality / Clay and Organic Soils Factor
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def Clay(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.Clay")

    # Clean up if needed
    if arcpy.Exists("WU_Clay"):
        arcpy.Delete_management("WU_Clay")

    PalustringPlots = arcpy.mapping.Layer(globalvars.srcPalustringPlots)
    SSURGO = arcpy.mapping.Layer(globalvars.srcSSURGO)
    logger.info("feature layers ready")
```

```
################################################################################
##########
    # Part 1: Palustrine Plots

################################################################################
##########
    #Create feature class to store Clay value
    arcpy.CopyFeatures_management(WetlandPoly,"WU_Clay","#","0","0","0")
    fcWUClay = arcpy.mapping.Layer("WU_Clay")
    logger.info("WU_Clay created to store Clay value")

    # add Clay field to Wetland Poly
    actions.DeleteField(fcWUClay,"Clay")
    arcpy.AddField_management(fcWUClay, "Clay", "TEXT", "", "", "2")
    arcpy.CalculateField_management(fcWUClay, "Clay", "'N'", "PYTHON_9.3")
    logger.info("field Clay added to Wetland units and initial value set to 'N'")

    # Select Wetland Units that intersect palustrine plots selection
    strWHERE = """("Soil_Textu" LIKE '%clay%' OR "Profile__1" LIKE '%clay%') AND
"Depth_of_o" IN ( ' ','0','1','2','3','4')"""
    arcpy.SelectLayerByAttribute_management(PalustringPlots, "NEW_SELECTION",
strWHERE)
    logger.info("Wetland Units that intersect palustrine plots selection selected")

    # Update the value for Clay based on Palustrine plots
    arcpy.SelectLayerByLocation_management(fcWUClay, "INTERSECT", PalustringPlots, "",
"NEW_SELECTION")
    arcpy.CalculateField_management(fcWUClay, "Clay", "'Y'", "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management(fcWUClay, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(PalustringPlots, "CLEAR_SELECTION")
    logger.info("the value for Clay based on Palustrine plots updated")


##############################################################################
##########
    # Part 2: SSURGO

##############################################################################
##########
    # Select Wetland Units that intersect with the SSURGO selection.
    arcpy.SelectLayerByLocation_management(fcWUClay, "INTERSECT", SSURGO, "", 
"NEW_SELECTION")
    logger.info("Wetland Units that intersect with the SSURGO selection selected")

    # Update the value for Clay based on SSURGO
    arcpy.CalculateField_management(fcWUClay, "Clay", "'Y'", "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management(fcWUClay, "CLEAR_SELECTION")
    logger.info("the value for Clay based on SSURGO updated")

    # Clean up
```

5.7.93 IrrEdge: Water Quality Potential

```
##############################################################################
#
# File Name: IrrEdge.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/7/2016 (modified 11/16/2017)
# Purpose:
#    Input to Water Quality/Surface Depressions
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcIrrEdge(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.IrrEdge")

    # Clean up if needed
    if arcpy.Exists("RiversLakes"):
        arcpy.Delete_management("RiversLakes")
    if arcpy.Exists("WU_RiversLakes"):
        arcpy.Delete_management("WU_RiversLakes")
    if arcpy.Exists("Intersecting_lines"):
        arcpy.Delete_management("Intersecting_lines")
    if arcpy.Exists("DryEdges"):
        arcpy.Delete_management("DryEdges")
    if arcpy.Exists("WU_IrrEdge"):
        arcpy.Delete_management("WU_IrrEdge")
    if arcpy.Exists("WU_IrrEdge1"):
        arcpy.Delete_management("WU_IrrEdge1")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcRiversLakes,"fcRiversLakes")
    logger.info("feature layers ready")

    #Create feature class to store intermediate value
    arcpy.CopyFeatures_management(WetlandPoly,"WU_IrrEdge1","#","0","0","0")
    arcpy.MakeFeatureLayer_management("WU_IrrEdge1", "fcWUIrrEdge1")
    logger.info("WU_IrrEdge1 created to store intermediate value")

    # deleting then adding the IrrEdgeRat
    actions.DeleteField("fcWUIrrEdge1","IrrEdgeRat")
    arcpy.AddField_management("fcWUIrrEdge1", "IrrEdgeRat", "FLOAT")

    # Calculate initial value of IrrEdgeRat for Wetland Units
    arcpy.CalculateField_management("fcWUIrrEdge1","IrrEdgeRat","[Shape_Length] /(
[Shape_Area]^0.5)","VB","#")
    logger.info("field IrrEdgeRat added and calculated")

    # Select the Wetland Units that border a river or lake

arcpy.SelectLayerByLocation_management("fcWUIrrEdge1","INTERSECT","fcRiversLakes",
"#","NEW_SELECTION")
    arcpy.CopyFeatures_management("fcWUIrrEdge1", "WU_RiversLakes")
    logger.info("Wetland Units that border a river or lake selected and exported")


    # Convert Wetland polygons to lines

arcpy.PolygonToLine_management("WU_RiversLakes","Intersecting_lines","IGNORE_NEIG
HBORS")
    arcpy.MakeFeatureLayer_management("Intersecting_lines", "fcIntersectingLines")
    logger.info("wetland polygons converted to lines")


    # Erase the wet perimeter lines
    arcpy.Erase_analysis("fcIntersectingLines","fcRiversLakes","DryEdges","#")
    arcpy.MakeFeatureLayer_management(r"DryEdges","fcDryEdges")
    logger.info("wet perimeter lines erased")


    # deleting then adding the DryPerim
    actions.DeleteField("fcDryEdges","DryPerim")
    arcpy.AddField_management("fcDryEdges", "DryPerim", "FLOAT")
    arcpy.CalculateField_management("fcDryEdges","DryPerim","[Shape_Length]","VB","#")
    logger.info("field DryPerim added and value calculated")


    # Reset to NULL records from DryEdges with length < 26.  These are mapping or
computational errors
    strWHERE = """"DryPerim" < 26"""
    arcpy.SelectLayerByAttribute_management("fcDryEdges", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcDryEdges","DryPerim","NULL","VB","#")
    logger.info("NULL records from DryEdges with length < 26 reseted to NULL")


##################################################################################
#################
```

```
    # SJ: Wetland Units & DryEdges

################################################################################
#################
    arcpy.SelectLayerByAttribute_management("fcDryEdges", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcWUIrrEdge1", "CLEAR_SELECTION")

    fmSJWUD = arcpy.FieldMappings()
    fmSJWUD.addTable("fcWUIrrEdge1")
    fmSJWUD.addTable("fcDryEdges")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","IrrEdgeRat","DryPerim"]

    for field in fmSJWUD.fields:
        if field.name not in keepers:
            fmSJWUD.removeFieldMap(fmSJWUD.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUIrrEdge1","fcDryEdges","WU_IrrEdge","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJWUD,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_IrrEdge","fcWUIrrEdge")
    logger.info("spatial join of WU_IrrEdge1 and DryEdges completed")

################################################################################
#################

    strWHERE = """"DryPerim" > 0"""
    arcpy.SelectLayerByAttribute_management("fcWUIrrEdge", "NEW_SELECTION",
strWHERE)
```

```
    # Update the value of IrrEdgeRar for the selected records

arcpy.CalculateField_management("fcWUIrrEdge","IrrEdgeRat","[DryPerim]/([Shape_Area]^0
.5)","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWUIrrEdge", "CLEAR_SELECTION")
    logger.info("the value of IrrEdgeRar updated for the selected records")


    # Add IrrEdge point field and set the initial value to zeo
    actions.DeleteField("fcWUIrrEdge","IrrEdge")
    arcpy.AddField_management("fcWUIrrEdge", "IrrEdge", "SHORT")
    arcpy.CalculateField_management("fcWUIrrEdge","IrrEdge","0","VB","#")
    logger.info("field IrrEdge added and initial value set to 0")


    # Assign 1 point if IrrEdgeRat > 6
    strWHERE = """"IrrEdgeRat" > 6"""
    arcpy.SelectLayerByAttribute_management("fcWUIrrEdge", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUIrrEdge","IrrEdge","1","VB","#")
    logger.info("1 point assigned for IrrEdge if IrrEdgeRat > 6 for wetland units")
    arcpy.SelectLayerByAttribute_management("fcWUIrrEdge", "CLEAR_SELECTION")


    # Clean up
    if arcpy.Exists("RiversLakes"):
        arcpy.Delete_management("RiversLakes")
    if arcpy.Exists("WU_RiversLakes"):
        arcpy.Delete_management("WU_RiversLakes")
    if arcpy.Exists("Intersecting_lines"):
        arcpy.Delete_management("Intersecting_lines")
    if arcpy.Exists("DryEdges"):
        arcpy.Delete_management("DryEdges")
```

```
if arcpy.Exists("WU_IrrEdge1"):
    arcpy.Delete_management("WU_IrrEdge1")
```

5.7.94 LandPos: Water Quality Potential

```
###############################################################################
#
# File Name: LandPos.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 6/14/2016 (modified 11/28/2017)
# Purpose:
#    Basic functional variable, used in several functional equations
#    Input to Water Quality/Headwater Location
#
###############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineLandPos(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.LandPos")

    # Clean up if needed
    if arcpy.Exists("WU_LandPos"):
        arcpy.Delete_management("WU_LandPos")
    if arcpy.Exists("WU_LandPos1"):
        arcpy.Delete_management("WU_LandPos1")
    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_FloodArea","fcWUFloodplain")
    arcpy.MakeFeatureLayer_management(r"WU_WflowPath","fcWFlowPath")
    arcpy.MakeFeatureLayer_management(globalvars.srcDrainageArea,"fcDA27m")
    arcpy.MakeFeatureLayer_management(globalvars.srcWBRivers,"fcWBRivers")
    arcpy.MakeFeatureLayer_management(globalvars.srcFSOFlow,"fcFSOFlow")
    arcpy.MakeFeatureLayer_management(globalvars.srcEnhWetland,"fcENWI")
    logger.info("feature layers ready")


#################################################################################
#################
    # SJ: Joins to add variables to new Landscape Position feature class

#################################################################################
#################
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("fcWFlowPath")
    fmSJFP.addTable("fcWUFloodplain")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WFlowPath","Floodplain"]

    for field in fmSJFP.fields:
        if field.name not in keepers:
            fmSJFP.removeFieldMap(fmSJFP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWFlowPath","fcWUFloodplain","WU_LandPos1","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJFP,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_LandPos1","fcWULandPos1")
    logger.info("spatial join WFlowPath and Floodplain completed")


arcpy.JoinField_management("fcWULandPos1","OBJECTID","fcDA27m","WUKey","WUKe
y;CntrWshd")
    logger.info("join field completed")

    # Export join to feature class
    arcpy.CopyFeatures_management("fcWULandPos1","WU_LandPos","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WU_LandPos","fcWULandPos")
    logger.info("joined data exported")

    # Add text field to the Wetland Units feature class
    actions.DeleteField("fcWULandPos","LandPos")
    arcpy.AddField_management("fcWULandPos", "LandPos", "TEXT", 5)
    logger.info("field LandPos added to wetland units")

    # Assign Lotic Stream Lanscape Position
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","""""Flood
plain" = 'Y'""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'LS'","PYTHON_9.3","#")
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")
    logger.info("Lotic stream landscape position assigned")

    # Assign Lotic River Landscape Position
    actions.DeleteField("fcWULandPos","River")
    arcpy.AddField_management("fcWULandPos", "River", "TEXT", 2)
arcpy.SelectLayerByLocation_management("fcWULandPos","WITHIN_A_DISTANCE","fcWBRivers","200 Meters","NEW_SELECTION")
    arcpy.CalculateField_management("fcWULandPos","River","'Y'","PYTHON_9.3")

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","""""LandPos" = 'LS' AND "River" = 'Y'""""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'LR'","PYTHON_9.3","#")
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")
    logger.info("Lotic river landscape position assigned")


    # Assign headwater modifier to Lotic Landscape Position for wetlands intersecting first and second order streams, outflow wetlands, and wetlands with intermittent flow.
    # Include isolated wetlands for now since almost all of these are actually outlfow wetlands, but the streams flowing from them are too small to show up on the NHD.
    actions.DeleteField("fcWULandPos","FSOStream")
    arcpy.AddField_management("fcWULandPos", "FSOStream", "TEXT", 2)

arcpy.SelectLayerByLocation_management("fcWULandPos","INTERSECT","fcFSOFlow","#","NEW_SELECTION")

arcpy.CalculateField_management("fcWULandPos","FSOStream","'Y'","PYTHON_9.3","#")



#arcpy.SelectLayerByAttribute_management(fcWULandPos,"NEW_SELECTION","""""LandPos" = 'LS' AND ("FSOStream" = 'Y' OR "WFlowPath" LIKE '%O%' OR "WFlowPath" LIKE '%I%')""""")

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","LandPos = 'LS' AND ((POSITION('O' IN WFlowPath) > 0) OR (POSITION('I' IN WFlowPath) > 0) OR (FSOStream = 'Y'))")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'LSh'","PYTHON_9.3","#")
    logger.info("headwater modifier assigned to Lotic Landscape position for certain wetlands")
    # Assign Lentic Landscape Position
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("fcENWI","NEW_SELECTION",""""WETLAND
_TYPE" = 'Lake'""")

arcpy.SelectLayerByLocation_management("fcWULandPos","WITHIN_A_DISTANCE","fcE
NWI","25 Meters","NEW_SELECTION")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'LE'","PYTHON_9.3","#")
    logger.info("Lentic Landscape position assigned")

    # Assign Terrene Landscape Position
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION",""""LandP
os" IS NULL""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'TE'","PYTHON_9.3","#")
    logger.info("Terrene Landscape position assigned")

    # Assign headwater modifier to Terrene Landscape Position for wetlands intersecting first
and second order streams, outflow wetlands, and wetlands with intermittent flow.
    # Include isolated wetlands for now since almost all of these are actually outlfow wetlands,
but the streams flowing from them are too small to show up on the NHD.

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","""LandPo
s = 'TE' AND ((POSITION('O' IN WFlowPath) > 0) OR (POSITION('I' IN WFlowPath) > 0)
OR (FSOStream = 'Y'))""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'TEh'","PYTHON_9.3","#")
    logger.info("headwater modifier assigned to Terrene Landscape position for certain
wetlands")
```

```
    # Assign headwater modifier to Terrene Landscape Position for wetlands with small
contributing watershed

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","""""LandP
os" = 'TE' AND "CntrWshd" < 161874""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'TEh'","PYTHON_9.3","#")
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")
    logger.info("headwater modifier assigned to Terrene Landscape Position for wetlands with
small contributing watershed")

    # Assign headwater modifier to Terrene Landscape Position for wetlands that occupy a large
percentage of their contributing watershed
    actions.DeleteField("fcWULandPos","PropWshd")
    arcpy.AddField_management("fcWULandPos", "PropWshd", "DOUBLE")
    arcpy.CalculateField_management("fcWULandPos","PropWshd","[Shape_Area]/
[CntrWshd]","VB","#")

arcpy.SelectLayerByAttribute_management("fcWULandPos","NEW_SELECTION","""""LandP
os" = 'TE' AND "PropWshd" > 0.05""")
    arcpy.CalculateField_management("fcWULandPos","LandPos","'TEh'","PYTHON_9.3","#")
    arcpy.SelectLayerByAttribute_management("fcWULandPos", "CLEAR_SELECTION")
    logger.info("headwater modifier assigned to Terrene Landscape for wetlands that occupy a
large percentage of their contributing watershed")

    # Clean Up
    if arcpy.Exists("WU_LandPos1"):
        arcpy.Delete_management("WU_LandPos1")
```

5.7.95 Lowslope: Water Quality Potential

```
##############################################################################
#
# File Name: LowSlope.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/3/2016 (modified 11/28/2017)
# Purpose:
#    Water Quality Function / Potential / Surface Depressions Factor (Max 2 points)
#     Flood Attenuation Function / Potential
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcLowSlope(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.LowSlope")

    # Clean up if needed
    if arcpy.Exists("WU_LowSlope"):
        arcpy.Delete_management("WU_LowSlope")

    #Setting Python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(r"WU_SLOPE", "fcWUSLOPE")
    logger.info("feature layers ready")

    ## Create feature class to store LowSlope
    fmSJLS = arcpy.FieldMappings()
    fmSJLS.addTable('fcWU')
    fmSJLS.addTable('fcWUSLOPE')
    keepers = []
    keepers = ['WUKey','Shape_Length','Shape_Area','SLOPE']
    for field in fmSJLS.fields:
        if field.name not in keepers:
            fmSJLS.removeFieldMap(fmSJLS.findFieldMapIndex(field.name))

    arcpy.SpatialJoin_analysis('fcWU', 'fcWUSLOPE', 'WU_LowSlope', 'JOIN_ONE_TO_ONE',
'KEEP_ALL', fmSJLS, 'CONTAINS')
    fcWULowSlope = arcpy.mapping.Layer(r"WU_LowSlope")
    logger.info("feature class WU_LowSlope created to store LowSlope")

    # deleting then add the LowSlope field and setting the value to 0
    actions.DeleteField(fcWULowSlope,"LowSlope")
    arcpy.AddField_management(fcWULowSlope, "LowSlope", "SHORT")
    arcpy.CalculateField_management(fcWULowSlope, "LowSlope", "0", "PYTHON_9.3")
    logger.info("field LowSlope added and initial value set to 0")


################################################################################
    # LowSlope field

################################################################################
    # assigning 2 points to wetlands with a slope < 2
    strWHERE = """"SLOPE" < 2"""
    arcpy.SelectLayerByAttribute_management(fcWULowSlope, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWULowSlope, "LowSlope", "2", "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management(fcWULowSlope, "CLEAR_SELECTION")

    # assigning 1 points to wetlands with a slope > 1 and < 6
    strWHERE = """"SLOPE" > 1 AND "SLOPE" < 6 """
    arcpy.SelectLayerByAttribute_management(fcWULowSlope, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWULowSlope, "LowSlope", "1", "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management(fcWULowSlope, "CLEAR_SELECTION")
    logger.info("points assigned to wetland units")
```

5.7.96 Microtopo: Water Quality Potential

```
##############################################################################
#
# File Name: Microtopo.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/3/2016 (modified 11/09/2017)
# Purpose:
#    Input to Water Quality/Surface Depressions
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def MicroTopo(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.Microtopo")

    # Clean up if needed
    if arcpy.Exists("NWIpalustrine"):
        arcpy.Delete_management("NWIpalustrine")
    if arcpy.Exists("WU_Microtopo"):
        arcpy.Delete_management("WU_Microtopo")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcEnhWVWetland")
    logger.info("feature layers ready")

    # Create a new layer form the palustring polygons in the NWI
    strWHERE = """"ATTRIBUTE" LIKE 'P%'"""
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "NEW_SELECTION",
strWHERE)
    arcpy.CopyFeatures_management("fcEnhWVWetland", "NWIpalustrine")
    arcpy.MakeFeatureLayer_management("NWIpalustrine","fcNWIpalustrine")
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "CLEAR_SELECTION")


################################################################################
#################
    # SJ: NWIpalustrine

################################################################################
#################
    # Sum the permieters of the palustrine polygons that make up each Wetland Unit

arcpy.SpatialJoin_analysis("fcWU","fcNWIpalustrine","WU_Microtopo","JOIN_ONE_TO_O
NE","KEEP_ALL",
        """WUKey "WUKey" true true false 4 Long 0 0 ,First,#,fcWU,WUKey,-1,-1;
           Shape_Length "Shape_Length" false true true 8 Double 0 0
,First,#,fcWU,Shape_Length,-1,-1;
        Shape_Area "Shape_Area" false true true 8 Double 0 0 ,First,#,fcWU,Shape_Area,-1,-1;
        SHAPE_Length_1 "SHAPE_Length" false true true 8 Double 0 0
,Sum,#,fcNWIpalustrine,SHAPE_Length,-1,-1;
        SHAPE_Area_1 "SHAPE_Area" false true true 8 Double 0 0
,First,#,fcNWIpalustrine,SHAPE_Area,-1,-1""",
        "INTERSECT","#","#")
```

```
   fcWUMicrotopo = arcpy.mapping.Layer(r"WU_Microtopo")
   logger.info("Spatial Join completed to sum the permieters of the palustrine polygons that
make up each Wetland Unit")


################################################################################
#################


   # deleting then adding the MicroRatio, and Microtopo fields
   actions.DeleteField(fcWUMicrotopo,"MicroRatio")
   arcpy.AddField_management(fcWUMicrotopo, "MicroRatio", "FLOAT")
   actions.DeleteField(fcWUMicrotopo,"Microtopo")
   arcpy.AddField_management(fcWUMicrotopo, "Microtopo", "SHORT")

   # Divide the perimeter of summed palustrine polygons by the square roof of the Wetland Unit
Area
   arcpy.CalculateField_management(fcWUMicrotopo,"MicroRatio","[SHAPE_Length_1]/(
[Shape_Area]^0.5)","VB","#")

   # Assign points to Wetland Units
   arcpy.CalculateField_management(fcWUMicrotopo, "Microtopo", "0", "PYTHON")

   strWHERE = """"MicroRatio" > 8"""
   arcpy.SelectLayerByAttribute_management(fcWUMicrotopo, "NEW_SELECTION",
strWHERE)
   arcpy.CalculateField_management(fcWUMicrotopo, "Microtopo", "1", "PYTHON")
   arcpy.SelectLayerByAttribute_management(fcWUMicrotopo, "CLEAR_SELECTION")

   strWHERE = """"MicroRatio" > 15"""
   arcpy.SelectLayerByAttribute_management(fcWUMicrotopo, "NEW_SELECTION",
strWHERE)
```

```
arcpy.CalculateField_management(fcWUMicrotopo, "Microtopo", "2", "PYTHON")
arcpy.SelectLayerByAttribute_management(fcWUMicrotopo, "CLEAR_SELECTION")

if arcpy.Exists("NWIpalustrine"):
    arcpy.Delete_management("NWIpalustrine")
```

5.7.97 Organic: Water Quality Potential

```
##############################################################################
#
# File Name: Organic.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/18/2016 (modified 11/29/2017)
# Purpose:
#    Input to Water Quality / Clay and Organic Soils Factor
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def OrganicFactor(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.Organic")

    # Clean up if needed
    if arcpy.Exists("WU_Organic"):
        arcpy.Delete_management("WU_Organic")

    # Setting python variables
    fcSsurgoOrganic = arcpy.mapping.Layer(globalvars.srcSsurgoOrganic)
    fcEnhancedNWI = arcpy.mapping.Layer(globalvars.srcInput)
    fcPeatlands = arcpy.mapping.Layer(globalvars.srcPeatlands)
    fcPalustrinePlotsMarch2015 = arcpy.mapping.Layer(globalvars.srcPalustrineplots)
    fcSSURGO = arcpy.mapping.Layer(globalvars.srcSSURGOWV)

    arcpy.CopyFeatures_management(WetlandPoly, "WU_Organic")
    fcWUOrganic = arcpy.mapping.Layer(r"WU_Organic")
    logger.info("feature layers ready")

    # Add field Organic to Wetland Units and set initial point value to "no organic"
    actions.DeleteField(fcWUOrganic,"Organic")
    arcpy.AddField_management(fcWUOrganic, "Organic", "TEXT", 2)
    arcpy.CalculateField_management(fcWUOrganic,"Organic","'N'","PYTHON","#")
    logger.info("field Organic added and initial value set to N")

    ## PART 1: PEATLANDS
    # Select Wetland Units that are peatlands
arcpy.SelectLayerByLocation_management(fcWUOrganic,"INTERSECT",fcPeatlands,"#","NE
W_SELECTION")
    logger.info("wetland units that are peatlands selected")

    # Update value for "Organic" based on peatlands
    arcpy.CalculateField_management(fcWUOrganic,"Organic","'Y'","PYTHON","#")
    arcpy.SelectLayerByAttribute_management(fcWUOrganic, "CLEAR_SELECTION")
    logger.info("field Organic updated to wetland units that are peatlands")

    ## PART 2: NWI ORGANIC MATTER
    # Select polygons that have an organic modifier in the National Wetland Inventory
    strWHERE = """"ATTRIBUTE" LIKE '%g'"""
    arcpy.SelectLayerByAttribute_management(fcEnhancedNWI, "NEW_SELECTION",
strWHERE)

    # Select Wetland Units that intersect organic NWI polygons

arcpy.SelectLayerByLocation_management(fcWUOrganic,"INTERSECT",fcEnhancedNWI,"#"
,"NEW_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcEnhancedNWI, "CLEAR_SELECTION")
    logger.info("wetland units that intersect organic NWI polygons selected")

    # Update value for "Organic" based on NWI
    arcpy.CalculateField_management(fcWUOrganic,"Organic","'Y'","PYTHON","#")
    arcpy.SelectLayerByAttribute_management(fcWUOrganic, "CLEAR_SELECTION")
    logger.info("field Organic updated to wetland units that intersect organic NWI polygons")

    ## PART 3: PALUSTRINE PLOTS
    # Select Palustrine plots that have peat or much soils
    strWHERE = """"Soil_Textu" LIKE '%peat%' OR "Soil_Textu" LIKE '%muck%' OR
"Profile__1" LIKE '%peat%' OR "Profile__1" LIKE '%muck%' OR "Depth_of_o" NOT IN
('','0','1')"""

arcpy.SelectLayerByAttribute_management(fcPalustrinePlotsMarch2015,"NEW_SELECTION
",strWHERE)

    # Select Wetland Units that intersect palustrine plots with organic soils

arcpy.SelectLayerByLocation_management(fcWUOrganic,"INTERSECT",fcPalustrinePlotsMa
rch2015,"#","NEW_SELECTION")
    logger.info("wetland units that intersect palustrine plots with organic soils selected")

    # Update value for "Organic" based on palustrine plots
    arcpy.CalculateField_management(fcWUOrganic,"Organic","'Y'","PYTHON","#")
    arcpy.SelectLayerByAttribute_management(fcWUOrganic, "CLEAR_SELECTION")
    logger.info("field Organic updated to wetland units that intersect palustrine plots with organic
soils")


    ## PART 4: SSURGO SELECTION
    # Select Wetland Units that intersect with SSURGO selection

arcpy.SelectLayerByLocation_management(fcWUOrganic,"INTERSECT",fcSsurgoOrganic,"#
","NEW_SELECTION")
    logger.info("wetland units that intersect with SSURGO selection selected")


    # Update value for "Organic" based on SSURGO
    arcpy.CalculateField_management(fcWUOrganic,"Organic","'Y'","PYTHON","#")
    arcpy.SelectLayerByAttribute_management(fcWUOrganic, "CLEAR_SELECTION")
    logger.info("field Organic updated to wetland units that intersect with SSURGO selection")
```

5.7.98 SeasonPond: Water Quality Potential

```
##############################################################################
#
# File Name: SeasonPond.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/15/2016 (modified 12/01/2017)
# Purpose:
#    Input to Water Quality/Potential aspect/ChemTime Factor
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcSeasonPond(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.SeasonPond")

    # Clean up if needed
    if arcpy.Exists("WU_SeasonPond1"):
        arcpy.Delete_management("WU_SeasonPond1")
    if arcpy.Exists("WU_SeasonPond"):
        arcpy.Delete_management("WU_SeasonPond")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcEnhWVWetland")
    arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
    logger.info("feature layers ready")

    # Select all the aplustrine wetland that are not permanently flooded
    strWHERE = """"ATTRIBUTE" NOT LIKE '%H%' AND "ATTRIBUTE" LIKE 'P%'"""
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "NEW_SELECTION",
strWHERE)
    logger.info("all the aplustrine wetland that are not permanently flooded selected")

    # Create layer of non-permanently flooded wetlands from selection
    arcpy.CopyFeatures_management("fcEnhWVWetland", "WU_SeasonPond1")
    arcpy.MakeFeatureLayer_management("WU_SeasonPond1","fcSeasonPond1")
    logger.info("layer of non-permanently flooded wetlands from selection created")
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "CLEAR_SELECTION")

    #Add field to store shape area
    actions.DeleteField("fcSeasonPond1","SeaPondRatio")
    arcpy.AddField_management("fcSeasonPond1", "SHAPE_Area_1", "FLOAT")

arcpy.CalculateField_management("fcSeasonPond1","SHAPE_Area_1","[SHAPE_Area]","VB
","#")
    logger.info("field SHAPE_Area_1 added and value calculated")


#################################################################################
#################
    # SJ: SeasonPond

#################################################################################
#################
```

```
    # Join non-permananently flooded wetlands to Wetland Units and sum the non-permanently
flooded area
    fmSJ = arcpy.FieldMappings()
    fmSJ.addTable("fcWU")
    fmSJ.addTable("fcSeasonPond1")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","SHAPE_Area_1"]

    for field in fmSJ.fields:
        if field.name not in keepers:
            fmSJ.removeFieldMap(fmSJ.findFieldMapIndex(field.name))

    fldKeyIndex = fmSJ.findFieldMapIndex("SHAPE_Area_1")
    fieldmap = fmSJ.getFieldMap(fldKeyIndex)
    fieldmap.mergeRule = "Sum"
    fmSJ.replaceFieldMap(fldKeyIndex, fieldmap)


arcpy.SpatialJoin_analysis("fcWU","fcSeasonPond1","WU_SeasonPond","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJ,"INTERSECT")
    fcSeasonPond = arcpy.mapping.Layer(r"WU_SeasonPond")
    logger.info("Spatial join non-permananently flooded wetlands to Wetland Units and sum the
non-permanently flooded area")

    # Add field to store ration of non-permanently-flooded area to total area
    actions.DeleteField(fcSeasonPond,"SeaPondRatio")
    arcpy.AddField_management(fcSeasonPond, "SeaPondRatio", "FLOAT")
    logger.info("field SeaPondRatio added to store ration of non-permanently-flooded area to
total area")
```

```
    # Calculate raio o fnon-permanently flooded to Wetland Unit area

arcpy.CalculateField_management(fcSeasonPond,"SeaPondRatio","[SHAPE_Area_1]/[Shape_
Area]","VB","#")
    logger.info("raio o fnon-permanently flooded calculated to Wetland Unit area")

    # Add field to store points for fnon-permanently flooded area and set initial value to zero
    actions.DeleteField(fcSeasonPond,"SeasonPond")
    arcpy.AddField_management(fcSeasonPond, "SeasonPond", "SHORT")
    arcpy.CalculateField_management(fcSeasonPond,"SeasonPond","0","VB","#")
    logger.info("field SeasonPond added and initial value set to 0")

    # Assign points to Wetland Units for seasonal ponding
    strWHERE = """"SeaPondRatio" > 0.1"""
    arcpy.SelectLayerByAttribute_management(fcSeasonPond, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcSeasonPond,"SeasonPond","1","VB","#")

    strWHERE = """"SeaPondRatio" > 0.4"""
    arcpy.SelectLayerByAttribute_management(fcSeasonPond, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcSeasonPond,"SeasonPond","2","VB","#")

    strWHERE = """"SeaPondRatio" > 0.7"""
    arcpy.SelectLayerByAttribute_management(fcSeasonPond, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcSeasonPond,"SeasonPond","3","VB","#")
    logger.info("points assigned to Wetland Units for SeasonPond")
    arcpy.SelectLayerByAttribute_management(fcSeasonPond, "CLEAR_SELECTION")

    # Clean up
```

```
if arcpy.Exists("WU_SeasonPond1"):
    arcpy.Delete_management("WU_SeasonPond1")
```

5.7.99 Slope: Water Quality Potential

```
##########################################################################
#
# File Name: SLOPE.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/2/2016 (modified 11/28/2017)
# Purpose:
#    Calculate slope as the median value of percent
#    slope pixels within a Wetland Unit
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions
from arcpy.sa import *

def CalcSLOPE(WetlandPoly):
    arcpy.CheckOutExtension("Spatial")
    logger = logging.getLogger("WFA.WQuality.WQPotential.SLOPE")

    # Clean up if needed
    if arcpy.Exists("ZonalSt_WU_SLOPE"):
        arcpy.Delete_management("ZonalSt_WU_SLOPE")
    if arcpy.Exists("WU_SLOPE"):
        arcpy.Delete_management("WU_SLOPE")
    if arcpy.Exists("WU_SLOPE1"):
        arcpy.Delete_management("WU_SLOPE1")

    # Copy Wetland Units feature class
    arcpy.CopyFeatures_management(WetlandPoly,"WU_SLOPE1")
    arcpy.MakeFeatureLayer_management("WU_SLOPE1", "fcWUSLOPE1")
    logger.info("feature layer ready")

    # Zonal statistics
    #arcpy.CalculateStatistics_management(globalvars.srcSlopePCT)
    #ZonalSt_WU_SLOPE = ZonalStatisticsAsTable(WetlandPoly, "WUKey",
globalvars.srcSlopePCT, "ZonalSt_WU_SLOPE", "NODATA", "MEDIAN")
    arcpy.gp.ZonalStatisticsAsTable("fcWUSLOPE1", "WUKey", globalvars.srcSlopePCT,
"ZonalSt_WU_SLOPE", "NODATA", "MEDIAN")
    logger.info("zonal statistics completed")

    # join table
    arcpy.AddJoin_management("fcWUSLOPE1", "WUKey", "ZonalSt_WU_SLOPE",
"WUKey")
    logger.info("zonal statistics table joined to wetland units")

    # create feature class from the Wetland Poly data joined with the Zonal Statistics Output
Table
    arcpy.FeatureClassToFeatureClass_conversion("fcWUSLOPE1", arcpy.env.workspace,
"WU_SLOPE")
    WUSlope = arcpy.mapping.Layer(r"WU_SLOPE")
    logger.info("feature class WU_SLOPE created")

    actions.DeleteField(WUSlope,"SLOPE")
    arcpy.AddField_management(WUSlope, "SLOPE", "SHORT")
arcpy.CalculateField_management(WUSlope,"SLOPE","[ZonalSt_WU_SLOPE_MEDIAN]","
VB","#")
    logger.info("field SLOPE added and calculated")


    # remove join
    arcpy.RemoveJoin_management("fcWUSLOPE1")


    if arcpy.Exists("ZonalSt_WU_SLOPE"):
        arcpy.Delete_management("ZonalSt_WU_SLOPE")
    if arcpy.Exists("WU_SLOPE1"):
        arcpy.Delete_management("WU_SLOPE1")
```

5.7.100 VegByLP: Water Quality Potential

```
############################################################################
#
# File Name: VegByLP.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/22/2016 (modified 12/06/2017)
# Purpose:
#    Used in Water Quality Function / Potential aspect / Vegetation Factor
#
############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegByLP(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.VegByLP")

    # Clean up if needed
    if arcpy.Exists("WU_VegByLP1"):
        arcpy.Delete_management("WU_VegByLP1")
    if arcpy.Exists("WU_VegByLP"):
        arcpy.Delete_management("WU_VegByLP")
    if arcpy.Exists("VegAll"):
        arcpy.Delete_management("VegAll")
```

```
if arcpy.Exists("Lakes"):
    arcpy.Delete_management("Lakes")
if arcpy.Exists("WUbyLake"):
    arcpy.Delete_management("WUbyLake")
if arcpy.Exists("WUbyLakeVegAll"):
    arcpy.Delete_management("WUbyLakeVegAll")
if arcpy.Exists("Ponds"):
    arcpy.Delete_management("Ponds")
if arcpy.Exists("WUwithPond"):
    arcpy.Delete_management("WUwithPond")
if arcpy.Exists("WUwithPondVegAll"):
    arcpy.Delete_management("WUwithPondVegAll")
if arcpy.Exists("WUwithPondVegAll_join"):
    arcpy.Delete_management("WUwithPondVegAll_join")
if arcpy.Exists("WU_VegByLake"):
    arcpy.Delete_management("WU_VegByLake")

# Setting Python variables
arcpy.MakeFeatureLayer_management(WetlandPoly,"fcWU")
arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcInput")
arcpy.MakeFeatureLayer_management(globalvars.srcRiversLakes,"fcRiversLakes")
arcpy.MakeFeatureLayer_management(r"WU_WflowPath","fcWUflowPath")
logger.info("feature layers ready")

# Add fields to Wetland Units and set initial point value to zero
actions.DeleteField("fcWU","VegByLP")
arcpy.AddField_management("fcWU", "VegByLP", "SHORT")
arcpy.CalculateField_management("fcWU","VegByLP","0","VB","#")
logger.info("field VegByLP added and initial value set to 0")
```

```
    actions.DeleteField("fcWU","VegStream")
    arcpy.AddField_management("fcWU", "VegStream", "SHORT")
    logger.info("field VegStream added")

    actions.DeleteField("fcWU","PondRatio")
    arcpy.AddField_management("fcWU", "PondRatio", "FLOAT")
    logger.info("field PondRatio added")


################################################################################
#######
    # PART 1: CREATE VEGETATION LAYER

################################################################################
#######
    # Select the forest, shrubland, ermegent, moss, and aquatic bed vegetation

arcpy.SelectLayerByAttribute_management("fcInput","NEW_SELECTION",""""ATTRIBUTE
" LIKE 'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%' OR
"ATTRIBUTE" LIKE 'PAB%' OR "ATTRIBUTE" LIKE 'PML%'""")
    arcpy.CopyFeatures_management("fcInput", "VegAll")
    arcpy.MakeFeatureLayer_management(r"VegAll","fcVegAll")
    logger.info("forest, shrubland, ermegent, moss, and aquatic bed vegetation selected and
exported")


################################################################################
#######
    # PART 2: CREATE RIVER AND LAKE SELECTIONS

################################################################################
#######
```

```
    # Select Wetland Units that share a boundary with rivers or lakes
    arcpy.SelectLayerByAttribute_management("fcInput", "CLEAR_SELECTION")

arcpy.SelectLayerByLocation_management("fcWU","INTERSECT","fcRiversLakes","#","NE
W_SELECTION")
    arcpy.CopyFeatures_management("fcWU","WUbyLake","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WUbyLake","fcWUbyLake")
    logger.info("Wetland Units that share a boundary with lakes selected and exported")


    # Select WUbyLake that are vegetated

arcpy.SelectLayerByLocation_management("fcWUbyLake","CONTAINS","fcVegAll","#","N
EW_SELECTION")
    arcpy.CopyFeatures_management("fcWUbyLake","WUbyLakeVegAll","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WUbyLakeVegAll","fcWUbyLakeVegAll")
    logger.info("Wetland Units that share a boundary with lakes selected and exported")


    # Assign 1 point to VegByLP for the selected Wetland Units
    arcpy.CalculateField_management("fcWUbyLakeVegAll","VegByLP","1","VB","#")
    logger.info("1 point assignd to VegByLP for the selected Wetland Units")


################################################################################
#######
    # PART 3: CREATE POND SELECTIONS

################################################################################
#######
    # Select ponds.

arcpy.SelectLayerByAttribute_management("fcInput","NEW_SELECTION","""""ATTRIBUTE
" LIKE 'PA%' OR "ATTRIBUTE" LIKE 'PU%'""")
```

```
    arcpy.CopyFeatures_management("fcInput","Ponds","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"Ponds","fcPonds")
    logger.info("ponds selected and exported")

    # Select Wetland Units that contain ponds

arcpy.SelectLayerByLocation_management("fcWU","CONTAINS","fcPonds","#","NEW_SEL
ECTION")
    arcpy.CopyFeatures_management("fcWU","WUwithPond","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WUwithPond","fcWUwithPond")
    logger.info("Wetland Units that contain ponds selected and exported")

    # Select WUwithPond that are vegetated

arcpy.SelectLayerByLocation_management("fcWUwithPond","CONTAINS","fcVegAll","#","
NEW_SELECTION")
    arcpy.CopyFeatures_management("fcWUwithPond","WUwithPondVegAll","#","0","0","0")
    arcpy.MakeFeatureLayer_management(r"WUwithPondVegAll","fcWUwithPondVegAll")
    logger.info("WUwithPond that are vegetated selected and exported")

    # Add Area fields and calculate geometry
    actions.DeleteField("fcPonds","PondArea")
    arcpy.AddField_management("fcPonds", "PondArea", "FLOAT")
    arcpy.CalculateField_management("fcPonds","PondArea","[SHAPE_Area]","VB","#")
    logger.info("field PondArea added and calculated")


    actions.DeleteField("WUwithPondVegAll","WUArea")
    arcpy.AddField_management("WUwithPondVegAll", "WUArea", "FLOAT")

arcpy.CalculateField_management("WUwithPondVegAll","WUArea","[SHAPE_Area]","VB",
"#")
```

```
    logger.info("field WUArea added and calculated")


    # Spatial join vegetated WU with ponds
    fmSJPonds = arcpy.FieldMappings()
    fmSJPonds.addTable("fcWUwithPondVegAll")
    fmSJPonds.addTable("fcPonds")


    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","VegByLP","PondRatio","VegStream","PondArea",
"WUArea"]


    for field in fmSJPonds.fields:
        if field.name not in keepers:
            fmSJPonds.removeFieldMap(fmSJPonds.findFieldMapIndex(field.name))



arcpy.SpatialJoin_analysis("fcWUwithPondVegAll","fcPonds","WUwithPondVegAll_join","J
OIN_ONE_TO_ONE","KEEP_ALL",fmSJPonds,"CONTAINS")

arcpy.MakeFeatureLayer_management(r"WUwithPondVegAll_join","fcWUwithPondVegAllJo
in")
    logger.info("spatial join WUwithPondVegAll and Ponds completed")


    # Calculate ratio of pond area (Ponds) to vegetated Wetland Unit (WUwithPondVegAll_join)

arcpy.CalculateField_management("fcWUwithPondVegAllJoin","PondRatio","[PondArea]/[W
UArea]","VB","#")
    logger.info("field PondRatio calculated")


    # Assign 1 point to VegByLP if pond area is < 100% of Wetland Unit area
```

```
arcpy.SelectLayerByAttribute_management("fcWUwithPondVegAllJoin","NEW_SELECTION
",""""PondRatio" < 1""")
    arcpy.CalculateField_management("fcWUwithPondVegAllJoin","VegByLP","1","VB","#")
    logger.info("1 point assigned to VegByLP if pond area is < 100% of Wetland Unit area")

    # PART 4: COMBINE RESULTS
    # Spatial Join lake results to Wetland Units
    arcpy.SelectLayerByAttribute_management("fcWU", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcWUwithPondVegAllJoin",
"CLEAR_SELECTION")

    actions.DeleteField("fcWUbyLakeVegAll","VegByLP_1")
    arcpy.AddField_management("fcWUbyLakeVegAll", "VegByLP_1", "SHORT")

arcpy.CalculateField_management("fcWUbyLakeVegAll","VegByLP_1","[VegByLP]","VB","
#")

    fmSJLakes = arcpy.FieldMappings()
    fmSJLakes.addTable("fcWU")
    fmSJLakes.addTable("fcWUbyLakeVegAll")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","VegByLP","VegByLP_1","PondRatio"]

    for field in fmSJLakes.fields:
        if field.name not in keepers:
            fmSJLakes.removeFieldMap(fmSJLakes.findFieldMapIndex(field.name))
```

```
arcpy.SpatialJoin_analysis("fcWU","fcWUbyLakeVegAll","WU_VegByLake","JOIN_ONE_T
O_ONE","KEEP_ALL",fmSJLakes,"INTERSECT")
    arcpy.MakeFeatureLayer_management(r"WU_VegByLake","fcWUVegByLake")
    logger.info("spatial join of lake results completed")

    # Select lake results and write to Wetland Units
    strWHERE = """"VegByLP_1" = 1"""
    arcpy.SelectLayerByAttribute_management("fcWUVegByLake", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegByLake","VegByLP","1","VB","#")
    logger.info("lake results selected and written to Wetland Units")

    arcpy.SelectLayerByAttribute_management("fcWUVegByLake", "CLEAR_SELECTION")

    ## Spatial Join pond results to Wetland Units
    arcpy.SelectLayerByAttribute_management("fcWU", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcWUwithPondVegAllJoin",
"CLEAR_SELECTION")

    actions.DeleteField("fcWUwithPondVegAllJoin","VegByLP_12")
    arcpy.AddField_management("fcWUwithPondVegAllJoin", "VegByLP_12", "SHORT")

arcpy.CalculateField_management("fcWUwithPondVegAllJoin","VegByLP_12","[VegByLP]",
"VB","#")

    fmSJWU = arcpy.FieldMappings()
    fmSJWU.addTable("fcWUVegByLake")
    fmSJWU.addTable("fcWUwithPondVegAllJoin")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","VegByLP","VegByLP_1","VegByLP_12","PondR
atio"]

    for field in fmSJWU.fields:
        if field.name not in keepers:
            fmSJWU.removeFieldMap(fmSJWU.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUVegByLake","fcWUwithPondVegAllJoin","WU_VegByLP1
","JOIN_ONE_TO_ONE","KEEP_ALL",fmSJWU,"INTERSECT")
    arcpy.MakeFeatureLayer_management(r"WU_VegByLP1","fcWUVegByLP1")
    logger.info("spatial join of pond results completed")

    # Select pond results and write to Wetland Units
    strWHERE = """"VegByLP_12" = 1"""
    arcpy.SelectLayerByAttribute_management("fcWUVegByLP1", "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegByLP1","VegByLP","1","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWUVegByLP1", "CLEAR_SELECTION")
    logger.info("pond results selected and written to Wetland Units")

    # Delete unnecessary fields from WU_VegByLP1
    arcpy.DeleteField_management("fcWUVegByLP1","VegByLP_1;VegByLP_12")


################################################################################
#######
    # PART 5: Add vegetated wetlands that contain a through-flowing perennial stream.

################################################################################
#######
```

```
    ## Spatial Join WU_VegByLP1 to WU_WFlowPath
    fmSJPerennial = arcpy.FieldMappings()
    fmSJPerennial.addTable("fcWUVegByLP1")
    fmSJPerennial.addTable("fcWUflowPath")

    keepers = []
    keepers = 
["WUKey","Shape_Length","Shape_Area","VegByLP","PondRatio","VegStream","WFlowPat
h"]

    for field in fmSJPerennial.fields:
        if field.name not in keepers:
            fmSJPerennial.removeFieldMap(fmSJPerennial.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUVegByLP1","fcWUflowPath","WU_VegByLP","JOIN_ONE
_TO_ONE","KEEP_ALL",fmSJPerennial,"CONTAINS")
    arcpy.MakeFeatureLayer_management(r"WU_VegByLP","fcWUVegByLP")
    logger.info("spatial join WU_VegByLP1 to WU_WFlowPath completed")

    # Select vegetated wetlands

arcpy.SelectLayerByLocation_management("fcWUVegByLP","INTERSECT","fcVegAll","#",
"NEW_SELECTION")

    # Select vegetated wetlands that contain a through-flowing perennial stream
    strWHERE = """"WFlowPath" = 'TP'"""
    arcpy.SelectLayerByAttribute_management("fcWUVegByLP", "SUBSET_SELECTION",
strWHERE)
    arcpy.CalculateField_management("fcWUVegByLP","VegByLP","1","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWUVegByLP", "CLEAR_SELECTION")
```

```
logger.info("perennial stream results selected and written to Wetland Units")

# Clean up
arcpy.DeleteField_management("fcWU", "VegByLP;PondRatio;VegStream")
if arcpy.Exists("WU_VegByLP1"):
    arcpy.Delete_management("WU_VegByLP1")
if arcpy.Exists("WUbyLake"):
    arcpy.Delete_management("WUbyLake")
if arcpy.Exists("WUbyLakeVegAll"):
    arcpy.Delete_management("WUbyLakeVegAll")
if arcpy.Exists("Ponds"):
    arcpy.Delete_management("Ponds")
if arcpy.Exists("WUwithPond"):
    arcpy.Delete_management("WUwithPond")
if arcpy.Exists("WUwithPondVegAll"):
    arcpy.Delete_management("WUwithPondVegAll")
if arcpy.Exists("WUwithPondVegAll_join"):
    arcpy.Delete_management("WUwithPondVegAll_join")
if arcpy.Exists("WU_VegByLake"):
    arcpy.Delete_management("WU_VegByLake")
```

5.7.101 VegPerUng: Water Quality Potential

```
################################################################################
#
# File Name: VegPerUng.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 7/17/2015 (modified 11/08/2017)
# Purpose:
#     VegPerUng: Used in Water Quality Function / Potential / Vegetation. Max 5 points.
#     VegPerUng4: Used in Flood Attenuation Function / Potential / Vegetation. Max 4 points.
#     VegPerUng1: Used in Habitat and Ecological Integrity Function / Potential / Vegetation.
Max 1 point.
#
################################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegPerUng(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.VegPerUng")

    # Clean up if needed
    if arcpy.Exists("VegPFOPSSPEM"):
        arcpy.Delete_management("VegPFOPSSPEM")
    if arcpy.Exists("VegPerUng"):
        arcpy.Delete_management("VegPerUng")
    if arcpy.Exists("WUVegPerUngIntersect"):
        arcpy.Delete_management("WUVegPerUngIntersect")
    if arcpy.Exists("WUVegPerUngIntersect_SUM_STAT"):
        arcpy.Delete_management("WUVegPerUngIntersect_SUM_STAT")
    if arcpy.Exists("WU_VegPerUng"):
        arcpy.Delete_management("WU_VegPerUng")
    if arcpy.Exists("WU_VegPerUng1"):
        arcpy.Delete_management("WU_VegPerUng1")

    # setting the variables
    arcpy.MakeFeatureLayer_management(globalvars.srcPasturesNotHayfields, 
"fcPasturesNotHayfields")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcEnhWVWetland")
    logger.info("feature layers ready")

    # Create feature class to store VegPerUng variables
    arcpy.CopyFeatures_management(WetlandPoly, "WU_VegPerUng1")
    arcpy.MakeFeatureLayer_management("WU_VegPerUng1", "fcWUVegPerUng1")
    logger.info("feature layer WU_VegPerUng1 created")

    # selecting and creating a feature class of forests, shrublands, and persistent emergent 
vegetation sites within EnhWVWetland
    strWHERE = """"ATTRIBUTE" NOT LIKE  'PEM2%' AND ("ATTRIBUTE" LIKE 
'PEM%' OR "ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%')"""
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "NEW_SELECTION", 
strWHERE)
    arcpy.CopyFeatures_management("fcEnhWVWetland", "VegPFOPSSPEM")
    arcpy.MakeFeatureLayer_management("VegPFOPSSPEM", "fcVegPFOPSSPEM")
    arcpy.SelectLayerByAttribute_management("fcEnhWVWetland", "CLEAR_SELECTION")
    logger.info("feature class of forests, shrublands, and persistent emergent vegetation sites 
within EnhWVWetland created")
```

```
    # Erase the known grazed pastures from the intermediate vegetation layer
    arcpy.Erase_analysis("fcVegPFOPSSPEM","fcPasturesNotHayfields","VegPerUng","#")
    arcpy.MakeFeatureLayer_management("VegPerUng", "fcVegPerUng")
    logger.info("known grazed pastures from the intermediate vegetation layer erased")

    # Calculate the percentage of each Wetland Unit that is persistent ungrazed vegetation
(VegPerUng)
    # Attribute each Wetland Unit with the percentage of VerPerUng
    actions.DeleteField("fcWUVegPerUng1","OrigArea")
    arcpy.AddField_management("fcWUVegPerUng1", "OrigArea", "DOUBLE")
    arcpy.CalculateField_management("fcWUVegPerUng1", "OrigArea", "!SHAPE_Area!",
"PYTHON")
    logger.info("field OrigArea created and calculated")

    # intersect "fcWUVegPerUng1" with VegPerUng
    arInputData = ["fcWUVegPerUng1","fcVegPerUng"]
    arcpy.Intersect_analysis(arInputData, "WUVegPerUngIntersect","ALL","#","INPUT")
    arcpy.MakeFeatureLayer_management("WUVegPerUngIntersect",
"fcWUVegPerUngIntersect")
    logger.info("Wetland units intersected with VegPerUng")

    # add and calculate PctIntersect field
    actions.DeleteField("fcWUVegPerUngIntersect","PctIntersect")
    arcpy.AddField_management("fcWUVegPerUngIntersect", "PctIntersect", "FLOAT")
    arcpy.CalculateField_management("fcWUVegPerUngIntersect", "PctIntersect",
"!SHAPE_Area!/!OrigArea!*100", "PYTHON")
    logger.info("field PctIntersect added and calculated")

    # Execute the Summary Statistics function
    arcpy.Statistics_analysis("fcWUVegPerUngIntersect",
"WUVegPerUngIntersect_SUM_STAT", [["PctIntersect", "SUM"]],
["FID_WU_VegPerUng1"])
    arcpy.MakeTableView_management(r"WUVegPerUngIntersect_SUM_STAT",
"tvWUVegPerUngIntersectSumm")
    logger.info("summary table WUVegPerUngIntersect_SUM_STAT created")


    '''# Add new field (VegPerUngPct) to attribute table to store the percentage of persistent
ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng1","VegPerUngPct")
    arcpy.AddField_management("fcWUVegPerUng1", "VegPerUngPct", "FLOAT")
    arcpy.CalculateField_management("fcWUVegPerUng1", "VegPerUngPct", "0", "VB")
    logger.info("field VegPerUngPct added and initial value set to 0")


    # Add new field (VegPerUng) to attribute table to store the score in relation to the percentage
of persistent ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng1","VegPerUng")
    arcpy.AddField_management("fcWUVegPerUng1", "VegPerUng", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng1", "VegPerUng", "0", "VB")
    logger.info("field VegPerUng added and initial value set to 0")'''


    # Join Wetland Units to the WUVegPerUngIntersect_SUM_STAT table
    arcpy.AddJoin_management( "fcWUVegPerUng1", "WUKey",
"tvWUVegPerUngIntersectSumm", "FID_WU_VegPerUng1")
    logger.info("join added to the WUVegPerUngIntersect_SUM_STAT table")
    arcpy.FeatureClassToFeatureClass_conversion("fcWUVegPerUng1", arcpy.env.workspace,
"WU_VegPerUng")
    arcpy.MakeFeatureLayer_management(r"WU_VegPerUng", "fcWUVegPerUng")


    # Add new field (VegPerUngPct) to attribute table to store the percentage of persistent
ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng","VegPerUngPct")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUngPct", "FLOAT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUngPct", "0", "PYTHON")
    logger.info("field VegPerUngPct added and initial value set to 0")

    # Add new field (VegPerUng) to attribute table to store the score in relation to the percentage
of persistent ungrazed vegetation
    actions.DeleteField("fcWUVegPerUng","VegPerUng")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUng", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "0", "PYTHON")
    logger.info("field VegPerUng added and initial value set to 0")

    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUngPct",
"!WUVegPerUngIntersect_SUM_STAT_SUM_PctIntersect!", "PYTHON")
    logger.info("field VegPerUngPct calculated")

    # Remove Join
    arcpy.RemoveJoin_management("fcWUVegPerUng1")
    logger.info("join removed")

    # Replace null values with 0
    strWHERE = """"VegPerUngPct" IS NULL"""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUngPct", "0", "VB")

    # Assign points to Wetland Units for VegPerUng
    strWHERE = """"VegPerUngPct" > 66.7"""

arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
```

```
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "5", "VB")

    strWHERE = """"VegPerUngPct" < 66.701 AND "VegPerUngPct" > 33.3"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWHERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "3", "VB")

    strWHERE = """"VegPerUngPct" < 33.301 AND "VegPerUngPct" > 10"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWHERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "1", "VB")

    strWHERE = """"VegPerUngPct" < 10.001"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWHERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng", "0", "VB")
    logger.info("points assigned to field VegPerUng")
    arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","CLEAR_SELECTION")

    # add VegPerUng4 field
    actions.DeleteField("fcWUVegPerUng","VegPerUng4")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUng4", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "0", "VB")
    logger.info("field VegPerUng4 added and initial value set to 0")

    # add VegPerUng1 field
    actions.DeleteField("fcWUVegPerUng","VegPerUng1")
    arcpy.AddField_management("fcWUVegPerUng", "VegPerUng1", "SHORT")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng1", "0", "VB")
    logger.info("field VegPerUng1 added and initial value set to 0")

    # Assign points to Wetland Units for VegPerUng4 and VegPerUng1
    strWHERE = """"VegPerUngPct" > 10"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "1", "VB")

    strWHERE = """"VegPerUngPct" > 33.3"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "2", "VB")

    strWHERE = """"VegPerUngPct" > 50"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "3", "VB")
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng1", "1", "VB")

    strWHERE = """"VegPerUngPct" > 66.7"""
arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","NEW_SELECTION",strWH
ERE)
    arcpy.CalculateField_management("fcWUVegPerUng", "VegPerUng4", "4", "VB")
    logger.info("points assigned to fields VegPerUng1 and VegPerUng4")
    arcpy.SelectLayerByAttribute_management("fcWUVegPerUng","CLEAR_SELECTION")

    # Clean up
```

```
if arcpy.Exists("VegPFOPSSPEM"):
    arcpy.Delete_management("VegPFOPSSPEM")
if arcpy.Exists("VegPerUng"):
    arcpy.Delete_management("VegPerUng")
if arcpy.Exists("WUVegPerUngIntersect"):
    arcpy.Delete_management("WUVegPerUngIntersect")
if arcpy.Exists("WUVegPerUngIntersect_SUM_STAT"):
    arcpy.Delete_management("WUVegPerUngIntersect_SUM_STAT")
if arcpy.Exists("WU_VegPerUng1"):
    arcpy.Delete_management("WU_VegPerUng1")
```

5.7.102 VegWoody: Water Quality Potential

```
##########################################################################
#
# File Name: VegWoody.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/15/2016 (modified 11/03/2017)
# Purpose:
#    Input to Water Quality / Potential / Vegetation Factor
#
##########################################################################
#!/usr/: Input to Water Quality / Potential / Vegetation Factor.bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcVegWoody(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.VegWoody")

    # Clean up if needed
    if arcpy.Exists("VegPFOPSS"):
        arcpy.Delete_management("VegPFOPSS")
    if arcpy.Exists("VegPFO"):
        arcpy.Delete_management("VegPFO")
    if arcpy.Exists("WUPFOJoin"):
        arcpy.Delete_management("WUPFOJoin")
    if arcpy.Exists("WU_VegWoody"):
        arcpy.Delete_management("WU_VegWoody")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(WetlandPoly, "fcWU")
    arcpy.MakeFeatureLayer_management(globalvars.srcInput, "fcENWI")
    logger.info("feature layers ready")

    # Select all woody vegetation, both forest and shrubland
    strWHERE = """"ATTRIBUTE" LIKE 'PFO%' OR "ATTRIBUTE" LIKE 'PSS%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    logger.info("all woody vegetation, both forest and shrubland selected")

    # Create layer of woody vegetation from selection
    arcpy.CopyFeatures_management("fcENWI", "VegPFOPSS")
    arcpy.MakeFeatureLayer_management(r"VegPFOPSS", "fcVegPFOPSS")
    logger.info("layer VegPFOPSS created of woody vegetation from selection")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    # Add field to store woody area
    actions.DeleteField("fcVegPFOPSS","PFOPSSarea")
    arcpy.AddField_management("fcVegPFOPSS", "PFOPSSarea", "FLOAT")

arcpy.CalculateField_management("fcVegPFOPSS","PFOPSSarea","[SHAPE_Area]","VB","#"
)
    logger.info("field PFOPSSarea to store woody area")

    # Select just the forest vegetation, not including the shrubs
    strWHERE = """"ATTRIBUTE" LIKE 'PFO%'"""
    arcpy.SelectLayerByAttribute_management("fcENWI", "NEW_SELECTION", strWHERE)
    logger.info("the forest vegetation, not including the shrubs selected")

    # Create layer of forest vegetation from selection
    arcpy.CopyFeatures_management("fcENWI", "VegPFO")
    arcpy.MakeFeatureLayer_management(r"VegPFO", "fcVegPFO")
    logger.info("layer VegPFO created of forest vegetation from selection")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    # Add field to store forest area
    actions.DeleteField("fcVegPFO","PFOarea")
    arcpy.AddField_management("fcVegPFO", "PFOarea", "FLOAT")
    arcpy.CalculateField_management("fcVegPFO","PFOarea","[SHAPE_Area]","VB","#")
    logger.info("field PFOarea added to store forest area")

    # Join forests to Wetland Units and sum the forest area
################################################################################
#################
    # SJ: VegPFO
################################################################################
#################
    fmSJPFO = arcpy.FieldMappings()
    fmSJPFO.addTable("fcWU")
    fmSJPFO.addTable("fcVegPFO")

    fldKeyIndex1 = fmSJPFO.findFieldMapIndex("PFOarea")
    fieldmap1 = fmSJPFO.getFieldMap(fldKeyIndex1)
    fieldmap1.mergeRule = "Sum"
    fmSJPFO.replaceFieldMap(fldKeyIndex1, fieldmap1)
```

```
arcpy.SpatialJoin_analysis("fcWU","fcVegPFO","WUPFOjoin","JOIN_ONE_TO_ONE","KEE
P_ALL",fmSJPFO,"INTERSECT")
    arcpy.MakeFeatureLayer_management("WUPFOjoin","fcWUPFOjoin")
    logger.info("forests joined to wetland units and the forest area summed up")


################################################################################
#################
    # SJ: VegPFOPSS

################################################################################
#################
    fmSJPSS = arcpy.FieldMappings()
    fmSJPSS.addTable("fcWUPFOjoin")
    fmSJPSS.addTable("fcVegPFOPSS")

    fldKeyIndex2 = fmSJPSS.findFieldMapIndex("PFOPSSarea")
    fieldmap2 = fmSJPSS.getFieldMap(fldKeyIndex2)
    fieldmap2.mergeRule = "Sum"
    fmSJPSS.replaceFieldMap(fldKeyIndex2, fieldmap2)


arcpy.SpatialJoin_analysis("fcWUPFOjoin","fcVegPFOPSS","WU_VegWoody","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJPSS,"INTERSECT")
    fcWUVegWoody = arcpy.mapping.Layer(r"WU_VegWoody")
    logger.info("woody vegetation joined to wetland units and the woody area summed up")

    # Add fields to store rations of forest and woody area to total area
    actions.DeleteField(fcWUVegWoody,"PFOratio")
    arcpy.AddField_management(fcWUVegWoody, "PFOratio", "FLOAT")
    actions.DeleteField(fcWUVegWoody,"PFOPSSratio")
    arcpy.AddField_management(fcWUVegWoody, "PFOPSSratio", "FLOAT")
    logger.info("fields PFOratio and PFOPSSratio added to store rations of forest and woody area
to total area")

    # Calculate ratio of woody vegetation to Wetland Unit area

arcpy.CalculateField_management(fcWUVegWoody,"PFOPSSratio","[PFOPSSarea]/[Shape_A
rea]","VB","#")
    logger.info("ratio of woody vegetation calculated to Wetland Unit area")

    # Calculate ratio of forest vegetation to Wetland Unit area

arcpy.CalculateField_management(fcWUVegWoody,"PFOratio","[PFOarea]/[Shape_Area]","V
B","#")
    logger.info("ratio of forest vegetation calculated to Wetland Unit area")


################################################################################
########################
    # Add new attribute field to store points for VegWoody and set initial value to zero

################################################################################
########################
    actions.DeleteField(fcWUVegWoody,"VegWoody")
    arcpy.AddField_management(fcWUVegWoody, "VegWoody", "SHORT")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","0","VB","#")
    logger.info("field VegWoody added to store points for VegWoody and initial value set to
zero")

    actions.DeleteField(fcWUVegWoody,"VegWoody4")
    arcpy.AddField_management(fcWUVegWoody, "VegWoody4", "SHORT")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","0","VB","#")
    logger.info("field VegWoody4 added to store points for VegWoody and initial value set to
zero")

    actions.DeleteField(fcWUVegWoody,"VegWoody2")
    arcpy.AddField_management(fcWUVegWoody, "VegWoody2", "SHORT")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","0","VB","#")
    logger.info("field VegWoody2 added to store points for VegWoody and initial value set to
zero")

    actions.DeleteField(fcWUVegWoody,"VegWoodyFor")
    arcpy.AddField_management(fcWUVegWoody, "VegWoodyFor", "SHORT")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","0","VB","#")
    logger.info("field VegWoodyFor added to store points for VegWoody and initial value set to
zero")

    # Assign points to Wetland Units for woody vegetation
    strWHERE = """"PFOPSSratio" > 0.1"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","1","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","1","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","1","VB","#")

    strWHERE = """"PFOPSSratio" > 0.5"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","2","VB","#")

    strWHERE = """"PFOPSSratio" > 0.333"""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
```

```
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","2","VB","#")

    strWHERE = """"PFOPSSratio" > 0.667""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","3","VB","#")

    strWHERE = """"PFOPSSratio" > 0.667 AND "PFOratio" > 0.333""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","4","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","3","VB","#")

    strWHERE = """"PFOratio" > 0.667""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody","5","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody4","4","VB","#")
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoody2","2","VB","#")

    strWHERE = """"PFOratio" > 0.1 OR "PFOarea" > 10000""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","1","VB","#")

    strWHERE = """("PFOratio" > 0.33 AND "PFOarea" > 2000) OR "PFOarea" > 20000""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION", strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","2","VB","#")

    strWHERE = """("PFOratio" > 0.667 AND "PFOarea" > 5000) OR "PFOarea" > 50000""""
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "NEW_SELECTION",
strWHERE)
    arcpy.CalculateField_management(fcWUVegWoody,"VegWoodyFor","3","VB","#")
    logger.info("points assigned to Wetland Units for woody vegetation")
    arcpy.SelectLayerByAttribute_management(fcWUVegWoody, "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("VegPFOPSS"):
        arcpy.Delete_management("VegPFOPSS")
    if arcpy.Exists("VegPFO"):
        arcpy.Delete_management("VegPFO")
    if arcpy.Exists("WUPFOJoin"):
        arcpy.Delete_management("WUPFOJoin")
```

5.7.103 WFlowPath: Water Quality Potential

```
##########################################################################
#
# File Name: WFlowPath.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 4/5/2016 (modified 11/28/2017)
# Purpose:
#    Water Quality Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineWFlowPath(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQPotential.WFlowPath")

    # Clean up if needed
    if arcpy.Exists("WU_WflowPath1"):
        arcpy.Delete_management("WU_WflowPath1")
    if arcpy.Exists("NHDflowline_Intersect"):
        arcpy.Delete_management("NHDflowline_Intersect")
    if arcpy.Exists("NHDflowline_IntMult"):
        arcpy.Delete_management("NHDflowline_IntMult")
    if arcpy.Exists("NHDflowline_IntMultDiss"):
        arcpy.Delete_management("NHDflowline_IntMultDiss")
    if arcpy.Exists("WU_WflowPath2"):
        arcpy.Delete_management("WU_WflowPath2")
    if arcpy.Exists("WU_WflowPath"):
        arcpy.Delete_management("WU_WflowPath")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcInput,"fcENWI")
    arcpy.MakeFeatureLayer_management(globalvars.srcNHDWB24kRivers,
"fcNHDWB24kRivers")
    arcpy.MakeFeatureLayer_management(globalvars.srcNHDFlowline,"fcNHDFlowline")
    arcpy.MakeFeatureLayer_management(globalvars.srcDrainageArea,"fcDA27m")
    arcpy.MakeFeatureLayer_management(globalvars.srcLakes,"fcLakes")
    arcpy.MakeFeatureLayer_management(globalvars.srcRivers,"fcRivers")

    arcpy.CopyFeatures_management(WetlandPoly, "WU_WflowPath1")
    arcpy.MakeFeatureLayer_management(r"WU_WflowPath1","fcWUflowPath1")
    logger.info("feature layers ready")

    # Add text fields to the Wetland Units feature class, to allow computation of Water Flow Path
(WFlowPath)
    actions.DeleteField("fcWUflowPath1","FlowPath")
    arcpy.AddField_management("fcWUflowPath1", "FlowPath", "TEXT", 4)

    actions.DeleteField("fcWUflowPath1","PerInt")
    arcpy.AddField_management("fcWUflowPath1", "PerInt", "TEXT", 4)

    actions.DeleteField("fcWUflowPath1","WFlowPath")
    arcpy.AddField_management("fcWUflowPath1", "WFlowPath", "TEXT", 4)
    logger.info("text fields added to the Wetland Units")


##################################################################################
####
    # PART 1: Intersect NHDFlowlines and wetlands

##################################################################################
####
    arInputData = ["fcWUflowPath1","fcNHDFlowline"]
    arcpy.Intersect_analysis(arInputData,"NHDflowline_Intersect","ONLY_FID","#","POINT")
    fcNHDflowline_Intersect = arcpy.mapping.Layer(r"NHDflowline_Intersect")
    logger.info("intersect NHDFlowlines and wetlands completed")

    # Create multiple points for single line segments

arcpy.MultipartToSinglepart_management(fcNHDflowline_Intersect,"NHDflowline_IntMult")
    fcNHDflowline_IntMult = arcpy.mapping.Layer(r"NHDflowline_IntMult")
    logger.info("multiple points feature created")

    # Remove doubles

arcpy.Dissolve_management(fcNHDflowline_IntMult,"NHDflowline_IntMultDiss","#","#","SI
NGLE_PART","DISSOLVE_LINES")
    fcNHDflowline_IntMultDiss = arcpy.mapping.Layer(r"NHDflowline_IntMultDiss")
    logger.info("dissolve completed to remove doubles")

    # Clear all selections
    arcpy.SelectLayerByAttribute_management("fcWUflowPath1", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNHDFlowline", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcNHDflowline_IntMult,
"CLEAR_SELECTION")
```

```
    arcpy.SelectLayerByAttribute_management(fcNHDflowline_IntMultDiss,
"CLEAR_SELECTION")

    # Select by attribute, within attribute table of NHDFlowline
    strWhere = """"FCode" IN (46003,46007,33600,33601)""""

arcpy.SelectLayerByAttribute_management("fcNHDFlowline","NEW_SELECTION",strWhere
)
    logger.info("certain NHDFlowline selected")

    # Select the wetland units that intersect intermittent or ephemeral stream(s)

arcpy.SelectLayerByLocation_management("fcWUflowPath1","INTERSECT","fcNHDFlowlin
e","#","NEW_SELECTION")
    logger.info("wetland units that intersect intermittent or ephemeral stream(s) selected")

    # Attribute the Wetland Units that intersect intermittent or ephemeral stream(s)
    arcpy.CalculateField_management("fcWUflowPath1","PerInt","'I'","PYTHON")
    logger.info("value assigned to wetland units that intersect intermittent or ephemeral
stream(s)")

    # Select by attribute, within attribute table of NHDFlowline
    arcpy.SelectLayerByAttribute_management("fcWUflowPath1", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNHDFlowline", "CLEAR_SELECTION")
    strWhere = """"FCode" IN (33400,46000,46006,55800)""""

arcpy.SelectLayerByAttribute_management("fcNHDFlowline","NEW_SELECTION",strWhere
)
    logger.info("certain NHDFlowline selected")

    # Select the Wetland Units that intersect perennial stream(s)
```

```
arcpy.SelectLayerByLocation_management("fcWUflowPath1","INTERSECT","fcNHDFlowlin
e","#","NEW_SELECTION")
    logger.info("wetland units that intersect perennial stream(s) selected")

    # Attribue the Wetland Units that intersect perennial stream(s)
    arcpy.CalculateField_management("fcWUflowPath1","PerInt","'P'","PYTHON")
    arcpy.SelectLayerByAttribute_management("fcWUflowPath1", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNHDFlowline", "CLEAR_SELECTION")
    logger.info("value assigned to wetland units that intersect perennial stream(s)")

    # Join Wetland Units to stream intersection points
    arcpy.MakeFeatureLayer_management(r"WU_WflowPath1","fcWUflowPath1")

arcpy.MakeFeatureLayer_management(r"NHDflowline_IntMultDiss","fcNHDflowline_IntMult
Diss")
    fmSJFP = arcpy.FieldMappings()
    fmSJFP.addTable("fcWUflowPath1")
    fmSJFP.addTable("fcNHDflowline_IntMultDiss")

arcpy.SpatialJoin_analysis("fcWUflowPath1","fcNHDflowline_IntMultDiss","WU_WflowPath
2","JOIN_ONE_TO_ONE","KEEP_ALL",fmSJFP,"INTERSECT")
    logger.info("spatial join Wetland Units to stream intersection points completed")
    fcWUflowPath2 = arcpy.mapping.Layer(r"WU_WflowPath2")

    # Attribute FlowPath (Throughflow, Outflow, Isolated)

arcpy.SelectLayerByAttribute_management(fcWUflowPath2,"NEW_SELECTION","""""Join_C
ount" > 1""")
    arcpy.CalculateField_management(fcWUflowPath2,"FlowPath","'TH'","PYTHON")
arcpy.SelectLayerByAttribute_management(fcWUflowPath2,"NEW_SELECTION","""""Join_C
ount" = 1""")
    arcpy.CalculateField_management(fcWUflowPath2,"FlowPath","'OU'","PYTHON")


arcpy.SelectLayerByAttribute_management(fcWUflowPath2,"NEW_SELECTION","""""Join_C
ount" = 0""")
    arcpy.CalculateField_management(fcWUflowPath2,"FlowPath","'IS'","PYTHON")
    logger.info("FlowPath attribute values assigned to Wetland Units")
    arcpy.SelectLayerByAttribute_management(fcWUflowPath2, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcNHDFlowline", "CLEAR_SELECTION")


##############################################################################
####
    # PART 2: Update FlowPath based on adjcent streams, rivers, and impoundments

##############################################################################
####
    # Update FlowPath from isolated to outflow for wetlands within 30m of a mapped stream

arcpy.SelectLayerByLocation_management(fcWUflowPath2,"WITHIN_A_DISTANCE","fcN
HDFlowline","30 Meters","NEW_SELECTION")


    strWhere = """"FlowPath" = 'IS'"""

arcpy.SelectLayerByAttribute_management(fcWUflowPath2,"SUBSET_SELECTION",strWhe
re)


    arcpy.CalculateField_management(fcWUflowPath2,"FlowPath","'OU'","PYTHON","#")
    arcpy.CalculateField_management(fcWUflowPath2,"PerInt","'I'","PYTHON","#")
    logger.info("values of FlowPath from isolated to outflow for wetlands within 30m of a
mapped stream updated")


    # Update FlowPath from isolate to outflow wetlands that contain an impoundment (NWI)
    arcpy.SelectLayerByAttribute_management(fcWUflowPath2, "CLEAR_SELECTION")


    strWhere = """"ATTRIBUTE" LIKE 'P%h%'""""
    arcpy.SelectLayerByAttribute_management("fcENWI","NEW_SELECTION",strWhere)

arcpy.SelectLayerByLocation_management(fcWUflowPath2,"CONTAINS","fcENWI","#","NE
W_SELECTION")


    strWhere = """"FlowPath" = 'IS'""""

arcpy.SelectLayerByAttribute_management(fcWUflowPath2,"SUBSET_SELECTION",strWhe
re)


    arcpy.CalculateField_management(fcWUflowPath2,"FlowPath","'OU'","PYTHON","#")
    arcpy.CalculateField_management(fcWUflowPath2,"PerInt","'I'","PYTHON","#")
    logger.info("values of FlowPath from isolate to outflow wetlands that contain an
impoundment (NWI) updated")


    # Join to add area of contributing watershed (CntrWshd)
    arcpy.SelectLayerByAttribute_management(fcWUflowPath2, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

arcpy.JoinField_management(fcWUflowPath2,"OBJECTID","fcDA27m","WUKey","WUKey;
CntrWshd")
    logger.info("area of contributing watershed joined")


    # Export join to feature class
    arcpy.CopyFeatures_management(fcWUflowPath2, "WU_WflowPath")
    arcpy.MakeFeatureLayer_management(r"WU_WflowPath", "fcWUflowPath")
    logger.info("joined feature exported")

    # Update isolated wetland to outflow intermittent if contributing watershed > 40 acres

arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION",""""Flow
Path" = 'IS' AND "CntrWshd" > 161874""")

    arcpy.CalculateField_management("fcWUflowPath","FlowPath","'OU'","PYTHON","#")
    arcpy.CalculateField_management("fcWUflowPath","PerInt","'I'","PYTHON","#")
    logger.info("values of isolated wetland to outflow intermittent if contributing watershed > 40
acres updated")
    arcpy.SelectLayerByAttribute_management("fcWUflowPath", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")

    # Set Flow Path to throughflow perennial for wetlands adjacent to NWI rivers
    # Select NWI rivers
    #
arcpy.SelectLayerByAttribute_management("fcENWI","NEW_SELECTION",""""WETLAND
_TYPE" = 'Riverine'""")

arcpy.SelectLayerByLocation_management("fcWUflowPath","INTERSECT","fcRivers","#","
NEW_SELECTION")
    logger.info("NWI rivers selected")

    # Attribute Water Flow Path for Wetland Units Adjacent to NWI rivers
    arcpy.CalculateField_management("fcWUflowPath","FlowPath","'TH'","PYTHON","#")
    arcpy.CalculateField_management("fcWUflowPath","PerInt","'P'","PYTHON","#")
    logger.info("values assigned to Flow Path for Wetland Units intersecting NHD 24K rivers")
    arcpy.SelectLayerByAttribute_management("fcWUflowPath", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")
    # Select the NHD 24K Rivers and set intersecting wetland to throughflow perennial

arcpy.SelectLayerByLocation_management("fcWUflowPath","INTERSECT","fcNHDWB24k
Rivers","#","NEW_SELECTION")

    # Attribute Flow Path for Wetland Units intersecting NHD 24K rivers
    arcpy.CalculateField_management("fcWUflowPath","FlowPath","'TH'","PYTHON","#")
    arcpy.CalculateField_management("fcWUflowPath","PerInt","'P'","PYTHON","#")
    logger.info("values assigned to Flow Path for Wetland Units intersecting NHD 24K rivers")


################################################################################
###########
    # PART 3: Attribute Water Flow Path for Outflow, Throughflow and Isolated Wetland Units

################################################################################
###########

arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'TH' AND "PerInt" = 'I'""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'TI'","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'TH' AND "PerInt" = 'P'""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'TP'","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'OU' AND "PerInt" = 'I'""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'OI'","PYTHON","#")
arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION",""""Flow
Path" = 'OU' AND "PerInt" = 'P'"""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'OP'","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION",""""Flow
Path" = 'IS'"""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'IS'","PYTHON","#")
    logger.info("values assigned to Flow Path for Outflow, Throughflow and Isolated Wetland
Units")


################################################################################
###########
    # PART 4: Update Water Flow Path Based on adjacent lakes

################################################################################
###########
    arcpy.SelectLayerByAttribute_management("fcWUflowPath", "CLEAR_SELECTION")

#arcpy.SelectLayerByAttribute_management("fcENWI","NEW_SELECTION",""""WETLAN
D_TYPE" = 'Lake'"""")

arcpy.SelectLayerByLocation_management("fcWUflowPath","INTERSECT","fcLakes","#","N
EW_SELECTION")

    # Attribute Flow Path for Wetland Units adjacent to lakes
    arcpy.CalculateField_management("fcWUflowPath","FlowPath","'BI'","PYTHON","#")
    logger.info("values assigned to Flow Path for Wetland Units adjacent to lakes")

    # Attribute Water Flow Path for Wetland Units adjacent to lakes
    arcpy.SelectLayerByAttribute_management("fcWUflowPath", "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management("fcENWI", "CLEAR_SELECTION")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'BI' AND "WFlowPath" IN ( 'OP', 'OI')""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'BO'","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'BI' AND "WFlowPath" IN ( 'TP', 'TI')""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'TB'","PYTHON","#")


arcpy.SelectLayerByAttribute_management("fcWUflowPath","NEW_SELECTION","""""Flow
Path" = 'BI' AND "WFlowPath" = 'IS'""")
    arcpy.CalculateField_management("fcWUflowPath","WFlowPath","'IB'","PYTHON","#")
    logger.info("values assigned to WFlowPath field")
    arcpy.SelectLayerByAttribute_management("fcWUflowPath", "CLEAR_SELECTION")

    # Clean up
    if arcpy.Exists("WU_WflowPath1"):
        arcpy.Delete_management("WU_WflowPath1")
    if arcpy.Exists("NHDflowline_Intersect"):
        arcpy.Delete_management("NHDflowline_Intersect")
    if arcpy.Exists("NHDflowline_IntMult"):
        arcpy.Delete_management("NHDflowline_IntMult")
    if arcpy.Exists("NHDflowline_IntMultDiss"):
        arcpy.Delete_management("NHDflowline_IntMultDiss")
    if arcpy.Exists("WU_WflowPath2"):
        arcpy.Delete_management("WU_WflowPath2")
```

5.7.104 Water Quality Society

```
##############################################################################
#
# File Name: WQSociety.py
# Developer: Yibing Han
# Date: 12/13/2017
# Purpose:
#    This script handles the execution of all the Water Quality Value to Society metrics.
#
##############################################################################
import datetime
import logging
import traceback
import arcpy

from Variables import Fisheries, WaterSupply
from Factors import HUC12WQ, ImpairedOut, WQPlan, WQUse
from Aspects import WQSociety

def procWQSociety(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety")

    ############################################################
    ## 1. Run Variables
    ############################################################

    Fisheries.CalcFisheries(WetlandPoly)

    WaterSupply.DetermineWaterSupply(WetlandPoly)
```

```
##########################################################
## 2. Run Factors
##########################################################

HUC12WQ.DetermineHUC12WQ(WetlandPoly)

ImpairedOut.CalcImpairedOut(WetlandPoly)

WQPlan.CalcWQPlan(WetlandPoly)

WQUse.CalcWQUse(WetlandPoly)

##########################################################
## 3. Run Aspect
##########################################################

WQSociety.DetermineWQSociety(WetlandPoly)
```

5.7.105 Water Quality Society Aspects

```
##############################################################################
#
# File Name: WQSociety.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 5/26/2016 (modified 11/02/2017)
# Purpose:
#    Water Quality Function
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import arcpy
from utilities import actions
import logging

def DetermineWQSociety(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.WQSociety")

    # Setting python variables
    arcpy.MakeFeatureLayer_management(r"WU_HUC12WQ","fcHUC12WQ")
    arcpy.MakeFeatureLayer_management(r"WU_ImpairedOut","fcImpairedOut")
    arcpy.MakeFeatureLayer_management(r"WU_WQPlan","fcWQPlan")
    arcpy.MakeFeatureLayer_management(r"WU_WQUse","fcWQUse")
    logger.info("feature layers ready")
    # Clean up if needed
    if arcpy.Exists("WU_WQSociety1"):
        arcpy.Delete_management("WU_WQSociety1")
    if arcpy.Exists("WU_WQSociety2"):
        arcpy.Delete_management("WU_WQSociety2")
    if arcpy.Exists("WU_WQSociety3"):
        arcpy.Delete_management("WU_WQSociety3")
    if arcpy.Exists("WU_WQSociety4"):
        arcpy.Delete_management("WU_WQSociety4")
    if arcpy.Exists("WU_WQSociety"):
        arcpy.Delete_management("WU_WQSociety")

    # Create feature class to store WQSociety
    arcpy.CopyFeatures_management(WetlandPoly, "WU_WQSociety1")
    arcpy.MakeFeatureLayer_management(r"WU_WQSociety1","fcWUSociety1")
    logger.info("feature class WU_WQSociety1 created")

    # Add WQSociety field to Wetland Units and set initial point value to zero
    actions.DeleteField("fcWUSociety1","WQSociety")
    arcpy.AddField_management("fcWUSociety1", "WQSociety", "SHORT")
    arcpy.CalculateField_management("fcWUSociety1","WQSociety","0","VB","#")
    logger.info("WQSociety field added to Wetland Units and initial point values set to zero")

    # Spatial joins to bring in factor values

##################################################################################
#################
    # SJ: HUC12WQ
```

```
################################################################################
#################
    fmSJHUC = arcpy.FieldMappings()
    fmSJHUC.addTable("fcWUSociety1")
    fmSJHUC.addTable("fcHUC12WQ")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WQSociety","HUC12WQ"]

    for field in fmSJHUC.fields:
        if field.name not in keepers:
            fmSJHUC.removeFieldMap(fmSJHUC.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUSociety1","fcHUC12WQ","WU_WQSociety2","JOIN_ONE_
TO_ONE","KEEP_ALL",fmSJHUC,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQSociety2","fcWUSociety2")
    logger.info("Spatial Join completed to add variable HUC12WQ")


################################################################################
#################
    # SJ: ImpairedOut
################################################################################
#################
    fmSJIO = arcpy.FieldMappings()
    fmSJIO.addTable("WU_WQSociety2")
    fmSJIO.addTable("fcImpairedOut")
    keepers = []
    keepers = 
["WUKey","Shape_Length","Shape_Area","WQSociety","HUC12WQ","ImpairedOut"]


    for field in fmSJIO.fields:
        if field.name not in keepers:
            fmSJIO.removeFieldMap(fmSJIO.findFieldMapIndex(field.name))



arcpy.SpatialJoin_analysis("WU_WQSociety2","fcImpairedOut","WU_WQSociety3","JOIN_O
NE_TO_ONE","KEEP_ALL",fmSJIO,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQSociety3","fcWUSociety3")
    logger.info("Spatial Join completed to add variable ImpairedOut")



##############################################################################
#################
    # SJ: WQPlan

##############################################################################
#################
    fmSJWQP = arcpy.FieldMappings()
    fmSJWQP.addTable("fcWUSociety3")
    fmSJWQP.addTable("fcWQPlan")


    keepers = []
    keepers = 
["WUKey","Shape_Length","Shape_Area","WQSociety","HUC12WQ","ImpairedOut","WQPla
n"]


    for field in fmSJWQP.fields:
        if field.name not in keepers:
            fmSJWQP.removeFieldMap(fmSJWQP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUSociety3","fcWQPlan","WU_WQSociety4","JOIN_ONE_TO
_ONE","KEEP_ALL",fmSJWQP,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQSociety4","fcWUSociety4")
    logger.info("Spatial Join completed to add variable WQPlan")


##################################################################################
#################
    # SJ: WQUse
##################################################################################
#################
    fmSJWQP = arcpy.FieldMappings()
    fmSJWQP.addTable("fcWUSociety4")
    fmSJWQP.addTable("fcWQUse")

    keepers = []
    keepers =
["WUKey","Shape_Length","Shape_Area","WQSociety","HUC12WQ","ImpairedOut","WQPla
n","WQUse"]

    for field in fmSJWQP.fields:
        if field.name not in keepers:
            fmSJWQP.removeFieldMap(fmSJWQP.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUSociety4","fcWQUse","WU_WQSociety","JOIN_ONE_TO_
ONE","KEEP_ALL",fmSJWQP,"CONTAINS")
```

```
    arcpy.MakeFeatureLayer_management("WU_WQSociety","fcWUSociety")
    logger.info("Spatial Join completed to add variable WQUse")


    # Sum the factor points
    arcpy.CalculateField_management("fcWUSociety","WQSociety","[HUC12WQ] +
[ImpairedOut] + [WQPlan] + [WQUse]","VB","#")
    logger.info("points summed for WQSociety")


    # Reduce values that exceed the maximum allowable points

arcpy.SelectLayerByAttribute_management("fcWUSociety","NEW_SELECTION","""""WQSoc
iety" > 4""")
    arcpy.CalculateField_management("fcWUSociety","WQSociety","4","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWUSociety", "CLEAR_SELECTION")
    logger.info("points reduced to maximum allowed for WQSociety")


    # Clean up
    if arcpy.Exists("WU_WQSociety1"):
        arcpy.Delete_management("WU_WQSociety1")
    if arcpy.Exists("WU_WQSociety2"):
        arcpy.Delete_management("WU_WQSociety2")
    if arcpy.Exists("WU_WQSociety3"):
        arcpy.Delete_management("WU_WQSociety3")
    if arcpy.Exists("WU_WQSociety4"):
        arcpy.Delete_management("WU_WQSociety4")
```

5.7.106 Huc12WQ: Water Quality Society

```
##############################################################################
#
# File Name: HUC12WQ.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date 3/18/2016 (modified 12/05/2017)
# Purpose:
#    Water Quality Function, Value to Society
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def DetermineHUC12WQ(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.HUC12WQ")

    # Clean up if needed
    if arcpy.Exists("WU_HUC12WQ"):
        arcpy.Delete_management("WU_HUC12WQ")

    # Setting python variables
    shpHUC12s = arcpy.mapping.Layer(globalvars.srcHUC12s)
    PublicFishingLakes = arcpy.mapping.Layer(globalvars.srcPublicFishingLakes)
    fcAlgalStreams = arcpy.mapping.Layer(globalvars.srcAlgalStreams)
    fcAlgalLakes = arcpy.mapping.Layer(globalvars.srcAlgalLakes)
    shpImpairedStreams = arcpy.mapping.Layer(globalvars.srcImpairedStreams)
    shpLimestone = arcpy.mapping.Layer(globalvars.srcLimeStone)
    shpDolostone = arcpy.mapping.Layer(globalvars.srcDolostone)

    # Create feature class to store ImpairedOut variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_HUC12WQ")
    WUHUC12WQ = arcpy.mapping.Layer(r"WU_HUC12WQ")
    logger.info("feature layers ready")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(WUHUC12WQ,"HUC12WQ")
    arcpy.AddField_management(WUHUC12WQ, "HUC12WQ", "SHORT")
    arcpy.CalculateField_management(WUHUC12WQ,"HUC12WQ","0","VB","#")
    logger.info("field HUC12WQ added and initial value set to 0")

    # Select lakes with power boat use
    strWHERE = """"BoatType" NOT LIKE 'No%'"""
    arcpy.SelectLayerByAttribute_management(PublicFishingLakes, "NEW_SELECTION",
strWHERE)
    logger.info("lakes with power boat use selected")

    # Select HUC12s that contain lakes with power boat use

arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",PublicFishingLakes,"#
","NEW_SELECTION")
    logger.info("HUC12s that contain lakes with power boat use selected")

    # Select HUC12s that contain algal lakes
```

```
arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",fcAlgalLakes,"#","AD
D_TO_SELECTION")
    logger.info("HUC12s that contain algal lakes selected")

    # Select HUC12s that contain algal streams

arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",fcAlgalStreams,"#","
ADD_TO_SELECTION")
    logger.info("HUC12s that contain algal streams selected")

    # Select HUC12s that contain impaired stream reaches

arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",shpImpairedStreams,"
#","ADD_TO_SELECTION")
    logger.info("HUC12s that contain impaired stream reaches selected")

    # Select HUC12s that contain karst

arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",shpLimestone,"#","A
DD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(shpHUC12s,"INTERSECT",shpDolostone,"#","A
DD_TO_SELECTION")
    logger.info("HUC12s that contain karst selected")

    #Select Wetland Units with water quality issues

arcpy.SelectLayerByLocation_management(WUHUC12WQ,"INTERSECT",shpHUC12s,"#","
NEW_SELECTION")
    logger.info("Wetland Units with water quality issues selected")

    # Assign 1 point to Wetland Units in HUC12s with water quality issues
```

```
arcpy.CalculateField_management(WUHUC12WQ,"HUC12WQ","1","VB","#")
logger.info("1 point assigned to Wetland Units in HUC12s with water quality issues")

arcpy.SelectLayerByAttribute_management(WUHUC12WQ, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(shpHUC12s, "CLEAR_SELECTION")
```

5.7.107 ImpairedOut: Water Quality Society

```
##############################################################################
#
# File Name: ImpairedOut.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/16/2016 (modified 12/06/2017)
# Purpose:
#    Water Quality Function
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcImpairedOut(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.ImpairedOut")

    # Clean up if needed
    if arcpy.Exists("WU_ImpairedOut"):
        arcpy.Delete_management("WU_ImpairedOut")

    PublicFishingLakes = arcpy.mapping.Layer(globalvars.srcPublicFishingLakes)
    fcAlgalStreams = arcpy.mapping.Layer(globalvars.srcAlgalStreams)
    fcAlgalLakes = arcpy.mapping.Layer(globalvars.srcAlgalLakes)
    shpImpairedStreams = arcpy.mapping.Layer(globalvars.srcImpairedStreams)
    #shpEPAOverlist24KNHD = arcpy.mapping.Layer(globalvars.srcEPAOverlist)
    shpLimestone = arcpy.mapping.Layer(globalvars.srcLimeStone)
    shpDolostone = arcpy.mapping.Layer(globalvars.srcDolostone)

    # Create feature class to store ImpairedOut variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_ImpairedOut")
    WUImpairedOut = arcpy.mapping.Layer(r"WU_ImpairedOut")
    logger.info("feature layers ready")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(WUImpairedOut,"ImpairedOut")
    arcpy.AddField_management(WUImpairedOut, "ImpairedOut", "SHORT")
    arcpy.CalculateField_management(WUImpairedOut,"ImpairedOut","0","VB","#")
    logger.info("field ImpairedOut added and initial value set to 0")

    # Select lakes with power boat use
    strWHERE = """"BoatType" NOT LIKE 'No%'"""
    arcpy.SelectLayerByAttribute_management(PublicFishingLakes, "NEW_SELECTION", 
strWHERE)
    logger.info("lakes with power boat use selected")

    # Select Wetland Units < 1km from lakes with power boat use
    
arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",PublicFishingLak
es,"1000 Meters","NEW_SELECTION")
    logger.info("Wetland Units < 1km from lakes with power boat use selected")

    # Select Wetland Units < 1km from algal lakes
```

```
arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",fcAlgalLakes,"10
00 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units < 1km from algal lakes selected")

    # Select Wetland Units < 1km from algal streams

arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",fcAlgalStreams,"
1000 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units < 1km from algal streams selected")

    # Select Wetland Units < 1km from impaired stream reaches

arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",shpImpairedStrea
ms,"1000 Meters","ADD_TO_SELECTION")

#arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",shpEPAOverlist
24KNHD,"1000 Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units < 1km from impaired stream reaches selected")

    # Select Wetland Units on karst

arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",shpLimestone,"#"
,"ADD_TO_SELECTION")

arcpy.SelectLayerByLocation_management(WUImpairedOut,"INTERSECT",shpDolostone,"#"
,"ADD_TO_SELECTION")
    logger.info("Wetland Units on karst selected")

    # Assign 1 point to Wetland Units that dishcharge to impaired waters
    arcpy.CalculateField_management(WUImpairedOut,"ImpairedOut","1","VB","#")
    logger.info("1 point assigned to Wetland Units that discharge to impaired waters")
```

```
arcpy.SelectLayerByAttribute_management(PublicFishingLakes, "CLEAR_SELECTION")
arcpy.SelectLayerByAttribute_management(WUImpairedOut, "CLEAR_SELECTION")
```

5.7.108 WQPlan: Water Quality Society

```
##############################################################################
#
# File Name: WQPlan.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/18/2016 (modified 11/02/2017)
# Purpose:
#    Water Quality Function
#
##############################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcWQPlan(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.WQPlan")

    #clean up if needed
    if arcpy.Exists("WU_WQPlan"):
        arcpy.Delete_management("WU_WQPlan")

    # Setting python variables
    fcWatershedPlan = arcpy.mapping.Layer(globalvars.srcWatershedPlan)
    fcTDML = arcpy.mapping.Layer(globalvars.srcTMDL)
    fcNatStrProAct_HUC10 = arcpy.mapping.Layer(globalvars.srcNatStrPreAct)
    fcNF = arcpy.mapping.Layer(globalvars.srcNF)
    fcNP = arcpy.mapping.Layer(globalvars.srcNP)
    fcNWR = arcpy.mapping.Layer(globalvars.srcNWR)

    # Create feature class to store WQPlan factor
    arcpy.CopyFeatures_management(WetlandPoly, "WU_WQPlan")
    WUWQPlan = arcpy.mapping.Layer(r"WU_WQPlan")
    logger.info("feature layers ready")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(WUWQPlan,"WQPlan")
    arcpy.AddField_management(WUWQPlan, "WQPlan", "SHORT")
    arcpy.CalculateField_management(WUWQPlan,"WQPlan","0","VB","#")
    logger.info("WQPlan field added to Wetland Units and initial point values set to zero")

    # Select Wetland Units with Watershed Plan

arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcWatershedPlan,"#",
"NEW_SELECTION")
    logger.info("Wetland Units with Watershed Plan selected")

    # Select Wetland Units with TMDL Plan

arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcTDML,"#","ADD_
TO_SELECTION")
    logger.info("Wetland Units with TMDL Plan selected")

    # Select Wetland Units in watershed drained by National Streams Preservation Act streams
```

```
arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcNatStrProAct_HUC
10,"#","ADD_TO_SELECTION")
    logger.info("Wetland Units with TMDL Plan selected")

    # Select National Forest owned by Forest Service
    arcpy.SelectLayerByAttribute_management(fcNF,"NEW_SELECTION","""Ownership" =
'Forest Service'""")
    logger.info("National Forest owned by Forest Service selected")

    # Select Wetland Units in Nation Parks, Forets, and Wildlife Refuges

arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcNF,"#","ADD_TO_
SELECTION")

arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcNP,"#","ADD_TO_
SELECTION")

arcpy.SelectLayerByLocation_management(WUWQPlan,"INTERSECT",fcNWR,"#","ADD_T
O_SELECTION")
    logger.info("Wetland Units in Nation Parks, Forets, and Wildlife Refuges selected")

    # Assign 2 points to Wetland Units that are included in a watershed plan of some kind
    arcpy.CalculateField_management(WUWQPlan,"WQPlan","2","VB","#")
    logger.info("2 points assigned to Wetland Units that are included in a watershed plan of some
kind")

    arcpy.SelectLayerByAttribute_management(WUWQPlan, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(fcNF, "CLEAR_SELECTION")
```

5.7.109 WQuse: Water Quality Society

```
###########################################################################
#
# File Name: WQUse.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/18/2016 (modified 11/02/2017)
# Purpose:
#    Water Quality Function, Value to Society
#
###########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcWQUse(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.WQUse")

    # Clean up if needed
    if arcpy.Exists("WU_WQUse1"):
        arcpy.Delete_management("WU_WQUse1")
    if arcpy.Exists("WU_WQUse2"):
        arcpy.Delete_management("WU_WQUse2")
    if arcpy.Exists("WU_WQUse"):
        arcpy.Delete_management("WU_WQUse")
    # Setting python variables
    arcpy.MakeFeatureLayer_management(globalvars.srcSwimmingAreas,"fcSwimmingAreas")
    arcpy.MakeFeatureLayer_management(r"WU_WaterSupply","fcWaterSupply")
    arcpy.MakeFeatureLayer_management(r"WU_Fisheries","fcFisheries")
    logger.info("feature layers ready")

    # Create feature class to store WQPlan factor
    arcpy.CopyFeatures_management(WetlandPoly, "WU_WQUse1")
    arcpy.MakeFeatureLayer_management(r"WU_WQUse1","fcWUWQUse1")
    logger.info("feature class WU_WQUse1 created")

    # Add fields to Wetland Units and set initial point value to zero
    actions.DeleteField("fcWUWQUse1","WQUse")
    arcpy.AddField_management("fcWUWQUse1", "WQUse", "SHORT")
    arcpy.CalculateField_management("fcWUWQUse1","WQUse","0","VB","#")

    actions.DeleteField("fcWUWQUse1","Swim")
    arcpy.AddField_management("fcWUWQUse1", "Swim", "SHORT")
    arcpy.CalculateField_management("fcWUWQUse1","Swim","0","VB","#")
    logger.info("WQUse and Swim fields added to Wetland Units and initial point values set to
zero")

    # Select Wetland Units within 1km of Swimming Area

arcpy.SelectLayerByLocation_management("fcWUWQUse1","INTERSECT","fcSwimmingAr
eas","1000 Meters","NEW_SELECTION")
    logger.info("Wetland Units within 1km of Swimming Area selected")

    # Assign 1 point to selected Wetland Units
    arcpy.CalculateField_management("fcWUWQUse1","Swim","1","VB","#")
    logger.info("1 point assigned to selected Wetland Units")

    # Select Wetland Units within 50m of Swimming Area

arcpy.SelectLayerByLocation_management("fcWUWQUse1","INTERSECT","fcSwimmingAr
eas","50 Meters","NEW_SELECTION")
    logger.info("Wetland Units within 50m of Swimming Area selected")

    # Assign 2 points to selected Wetland Units
    arcpy.CalculateField_management("fcWUWQUse1","Swim","2","VB","#")
    logger.info("2 points assigned to selected Wetland Units")

    arcpy.SelectLayerByAttribute_management("fcWUWQUse1", "CLEAR_SELECTION")

    # Spatial join to bring together the factor values

###############################################################################
#################
    # SJ: WaterSupply

###############################################################################
#################
    fmSJWS = arcpy.FieldMappings()
    fmSJWS.addTable("fcWUWQUse1")
    fmSJWS.addTable("fcWaterSupply")

    keepers = []
    keepers = ["WUKey","Shape_Length","Shape_Area","WQUse","Swim","WaterSupply"]

    for field in fmSJWS.fields:
        if field.name not in keepers:
            fmSJWS.removeFieldMap(fmSJWS.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUWQUse1","fcWaterSupply","WU_WQUse2","JOIN_ONE_T
O_ONE","KEEP_ALL",fmSJWS,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQUse2","fcWUWQUse2")
    logger.info("Spatial Join completed to add variable WaterSupply")


################################################################################
#################
    # SJ: Fisheries

################################################################################
#################
    fmSJF = arcpy.FieldMappings()
    fmSJF.addTable("fcWUWQUse2")
    fmSJF.addTable("fcFisheries")

    keepers = []
    keepers = 
["WUKey","Shape_Length","Shape_Area","WQUse","Swim","WaterSupply","Fisheries"]

    for field in fmSJF.fields:
        if field.name not in keepers:
            fmSJF.removeFieldMap(fmSJF.findFieldMapIndex(field.name))


arcpy.SpatialJoin_analysis("fcWUWQUse2","fcFisheries","WU_WQUse","JOIN_ONE_TO_O
NE","KEEP_ALL",fmSJF,"CONTAINS")
    arcpy.MakeFeatureLayer_management("WU_WQUse","fcWUWQUse")
    logger.info("Spatial Join completed to add variable Fisheries")


    # Sum the points for water quality use
    arcpy.CalculateField_management("fcWUWQUse","WQUse","[Swim] + [WaterSupply] + 
[Fisheries]","VB","#")
    logger.info("points summed for WQUse")


    # Reduce points to maximum allowed for WQUse

arcpy.SelectLayerByAttribute_management("fcWUWQUse","NEW_SELECTION","""""WQUs
e" > 2""")
    arcpy.CalculateField_management("fcWUWQUse","WQUse","2","VB","#")
    arcpy.SelectLayerByAttribute_management("fcWUWQUse", "CLEAR_SELECTION")
    logger.info("points reduced to maximum allowed for WQUse")


    ## Clean up
    if arcpy.Exists("WU_WQUse1"):
        arcpy.Delete_management("WU_WQUse1")
    if arcpy.Exists("WU_WQUse2"):
        arcpy.Delete_management("WU_WQUse2")
```

5.7.110 Fisheries: Water Quality Society

```
##########################################################################
#
# File Name: Fisheries.py
# Developer: Chad Ashworth (modified by Yibing Han)
# Date: 3/16/2016 (modified 12/04/2017)
# Purpose:
#    Water Quality Function
#
##########################################################################
#!/usr/bin/python
import sys
sys.path.append("../../..")

import logging
import arcpy
from globalvars import globalvars
from utilities import actions

def CalcFisheries(WetlandPoly):
    logger = logging.getLogger("WFA.WQuality.WQSociety.Fisheries")

    # Clean up if needed
    if arcpy.Exists("WU_Fisheries"):
        arcpy.Delete_management("WU_Fisheries")

    # Setting python variables
    TroutUpdate = arcpy.mapping.Layer(globalvars.srcTrout)
    HQSFisheries = arcpy.mapping.Layer(globalvars.srcHQSFisheries)
    TroutStreams = arcpy.mapping.Layer(globalvars.srcTroutStreams)

    # Create feature class to store Fisheries variable
    arcpy.CopyFeatures_management(WetlandPoly, "WU_Fisheries")
    WUFisheries = arcpy.mapping.Layer(r"WU_Fisheries")
    logger.info("feature layers ready")

    # Add field to Wetland Units and set initial point value to zero
    actions.DeleteField(WUFisheries,"Fisheries")
    arcpy.AddField_management(WUFisheries, "Fisheries", "SHORT")
    arcpy.CalculateField_management(WUFisheries,"Fisheries","0","VB","#")
    logger.info("field Fisheries added and initial value set to 0")

    # Select Wetland Units within 1km of perennial trout stream

arcpy.SelectLayerByLocation_management(WUFisheries,"INTERSECT",TroutUpdate,"1000
Meters","NEW_SELECTION")
    logger.info("Wetland Units within 1km of perennial trout stream selected")

    # Select Wetland Units within 1km of high quality stream fisheries

arcpy.SelectLayerByLocation_management(WUFisheries,"INTERSECT",HQSFisheries,"1000
Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units within 1km of high quality stream fisheries selected")

    # Select Wetland Units within 1km of stocked trout streams

arcpy.SelectLayerByLocation_management(WUFisheries,"INTERSECT",TroutStreams,"1000
Meters","ADD_TO_SELECTION")
    logger.info("Wetland Units within 1km of stocked trout streams selected")
    # Assign 1 point to Wetland Units that disharge to economic fisheries
    arcpy.CalculateField_management(WUFisheries,"Fisheries","1","VB","#")
    logger.info("1 point assigned to Wetland Units that discharge to economic fisheries")
    arcpy.SelectLayerByAttribute_management(WUFisheries, "CLEAR_SELECTION")

    # Select special fisheries
    strWHERE = """"StockCode" NOT LIKE 'NS'"""
    arcpy.SelectLayerByAttribute_management(TroutStreams, "NEW_SELECTION", 
strWHERE)
    strWHERE = """"RegType" = 'Catch-and-Release' OR "RegType" = 'Children and Class Q' 
OR "RegType" = 'Fly-fishing-Only'"""

arcpy.SelectLayerByAttribute_management(TroutStreams,"ADD_TO_SELECTION",strWHE
RE)
    logger.info("special fisheries selected")

    # Select Wetland Units within 1 km of special fisheries

arcpy.SelectLayerByLocation_management(WUFisheries,"INTERSECT",TroutStreams,"1000 
Meters","NEW_SELECTION")
    logger.info("Wetland Units within 1 km of special fisheries selected")

    # Assign 2 points to Wetland Units within 1km of special fisheries
    arcpy.CalculateField_management(WUFisheries,"Fisheries","2","VB","#")
    logger.info("2 points assigned to Wetland Units within 1km of special fisheries")
    arcpy.SelectLayerByAttribute_management(WUFisheries, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(TroutStreams, "CLEAR_SELECTION")
```

5.7.111 WaterSupply: Water Quality Society

```
##############################################################################
#
# File Name: WaterSupply.py
# Developer: Chad Ashworth, Nate Gunn (modified by Yibing Han)
# Date: 5/18/2016 (modified 12/08/2017)
# Purpose:
#    This script contains functions which score wetlands based upon spatial
#    relationships to water supply #intakes.  The "variable" (feature class)
#    output is referred to as WaterSupply in accompanying documentation
#
##############################################################################
#!/usr/bin/python
import sys, os
sys.path.append("../../..")

import arcpy
from globalvars import globalvars
from datetime import datetime
import collections
from utilities import actions
import logging

logger = logging.getLogger("WFA.WQuality.WQSociety.WaterSupply")
# Setting Python variables
RATIO = "contribution"
KEY = "wetland"
PSWI = "intake"
ITEMS_TO_REMOVE = ["WU_WaterSupply", "Surface_Intake_Drainage_Area_Intersect"]
```

```
#if __name__ == "__main__":
def DetermineWaterSupply(WetlandPoly):
    # Clean up if needed
    if arcpy.Exists("WU_WaterSupply"):
        arcpy.Delete_management("WU_WaterSupply")
    if arcpy.Exists("Surface_Intake_Drainage_Area_Intersect"):
        arcpy.Delete_management("Surface_Intake_Drainage_Area_Intersect")
    if arcpy.Exists("pswi_watersheds_with_out_of_state_drainage"):
        arcpy.Delete_management("pswi_watersheds_with_out_of_state_drainage")

    # Reading layers
    wetlands = WetlandPoly
    public_water_intakes_orig = globalvars.PSWI_WATERSHEDS
    arcpy.CopyFeatures_management(public_water_intakes_orig,
"pswi_watersheds_with_out_of_state_drainage")
    public_water_intakes =
arcpy.mapping.Layer("pswi_watersheds_with_out_of_state_drainage")
    zpc_5hr = arcpy.mapping.Layer(globalvars.ZPC_5HR)
    protection_areas = arcpy.mapping.Layer(globalvars.PROTECTION_AREAS)
    logger.info("layers ready")

    items = PublicWaterSupplyCheck(wetlands, public_water_intakes, zpc_5hr, protection_areas,
public_water_intakes)
    #sanity_check(items, wetlands)

    # Clean up
    if arcpy.Exists("Surface_Intake_Drainage_Area_Intersect"):
        arcpy.Delete_management("Surface_Intake_Drainage_Area_Intersect")
    if arcpy.Exists("pswi_watersheds_with_out_of_state_drainage"):
        arcpy.Delete_management("pswi_watersheds_with_out_of_state_drainage")


######################################################################
################ OPTIMIZED CODE STARTS HERE
######################################################################
#Calculate each wetland's max contribution to a surface intake drainage area
def get_wetland_unit_contribution(wetland_unit, surface_intake_drainage_area, output):
    ## Create Original Area Field to the Surface Intake Drainage Area
    actions.DeleteField(surface_intake_drainage_area, "OrigArea")
    arcpy.AddField_management(surface_intake_drainage_area, "OrigArea", "DOUBLE")
    arcpy.CalculateField_management(surface_intake_drainage_area, "OrigArea",
"!SHAPE_Area!", "PYTHON_9.3")
    logger.info("field OrigArea created and calculated")


    ## Intersect the Supplied Wetland Layer and the Surface Intake Drainage Area
    lyrs = [wetland_unit, surface_intake_drainage_area]

arcpy.Intersect_analysis(lyrs,"Surface_Intake_Drainage_Area_Intersect","ALL","#","INPUT")
    logger.info("intersect complete for Supplied Wetland Layer and the Surface Intake Drainage
Area")


    #delete original area
    #arcpy.DeleteField_management(in_table=surface_intake_drainage_area,
drop_field="OrigArea")


## This will help keep things reliable if run in immediate mode.  It also speeds testing.
def cleanup_workspace():
    # get the TOC by way of the map document and current frame
    if "ArcMap" in sys.executable:
        doc = arcpy.mapping.MapDocument("CURRENT")
        frame = doc.activeDataFrame
        # clear the TOC or Delete attempts will fail
        for lyr in arcpy.mapping.ListLayers(doc):
            if lyr.name in ITEMS_TO_REMOVE:
                arcpy.mapping.RemoveLayer(frame, lyr)

    # clear the FCs used in intermediate steps and prior run results.
    for item in ITEMS_TO_REMOVE:
        if arcpy.Exists(item):
            #print("Deleting {}".format(item,))
            arcpy.Delete_management(item)
    logger.info("workspace cleaned up")
#area ratio based scoring
#NOTE: the documentation uses percentage scoring.  Be careful to use percentage
#in results if
def score(ratio):
    retval = 0
    if ratio:
        if ratio > 0.001:
            retval = 1
        if ratio > 0.01:
            retval = 2
    return retval

def classify_wetlands_by_intersection(wetlands, pswi_watersheds):
    #a range would be faster, but requires no gaps in OIDs for this class
    sc = arcpy.da.SearchCursor(wetlands, ["OID@"])
    items = {}
    for item in sc:
        items[item[0]] = {KEY: item[0], RATIO: 0.0}
    del sc
    sc = arcpy.da.SearchCursor("Surface_Intake_Drainage_Area_Intersect", ['FID_' +
wetlands.name, 'FID_' + pswi_watersheds.name, 'SHAPE@AREA' , 'OrigArea'])
    #store the max contribution towards a drainage area for any wetland unit which intersects
    for item in sc:
        #print("listing item {}".format(item,))
        wlu = item[0]
        #print("Already stored? {}".format(key in items))
        if wlu in items and items[wlu]:
            testval = {PSWI: item[1], RATIO: item[2] / item[3]} # new (key, ratio intersection area
to intake area)
            if testval[RATIO] > items[wlu][RATIO]:
                items[wlu] = testval
        else:
            items[wlu] = {PSWI:item[1], RATIO: item[2] / item[3]} # store dictionary of (pwsi,
ratio intersection area to intake area)
    zeroes = 0
    ones = 0
    twos = 0
    for item in items:
        item_score = score(items[item][RATIO]) #score based on the stored ratio
        if item_score == 0:
            zeroes += 1
        if item_score ==1:
            ones += 1
        if item_score == 2:
            twos += 1
        items[item]['score'] = item_score
        #if item_score > 0:
            #print("wetland {} contains item: {}".format(item, items[item]))
    #print "Classification results immediately after intersection scoring"
    #print "zeroes: {}".format(zeroes,)
    #print "ones  : {}".format(ones,)
    #print "twos  : {}".format(twos,)
    #print ""
    return items
    logger.info("the max contribution towards a drainage area for any wetland unit which 
intersects stored")


#Perform the selections by attributes and intersections to generate scores of 1
def score_one_by_selection(wetlands, zpc_5hr, protection_areas, items):
    ## Select Secondary Protection Areas with surface water connections.
    strWhere1Pt = """"P_TYPE" in (NULL, 'Secondary Protection Area') AND "FAC_SRC" IN 
('GU', 'SW')"""
    arcpy.SelectLayerByAttribute_management(protection_areas, "NEW_SELECTION", 
strWhere1Pt)
    logger.info("Secondary Protection Areas with surface water connections selected")


    ## Select Wetland units within a Secondary Protection Area.
    arcpy.SelectLayerByLocation_management(wetlands, "INTERSECT", protection_areas, "", 
"NEW_SELECTION")
    logger.info("Wetland units within a Secondary Protection Area selected")


    ## Select Wetland Units within the 5-hour travel distance in a Zone of Peripheral Concern.
    arcpy.SelectLayerByLocation_management(wetlands, "INTERSECT", zpc_5hr, "", 
"ADD_TO_SELECTION")
    logger.info("Wetland Units within the 5-hour travel distance in a Zone of Peripheral Concern 
added to selection")


    with arcpy.da.SearchCursor(wetlands, ["OID@"]) as sc:
        for record in sc:
            oid = record[0]
            items[oid]['score']  = max(items[oid]['score'], 1)
    del sc
    logger.info("1 point scores generated for wetland units")

    arcpy.SelectLayerByAttribute_management(protection_areas, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(wetlands, "CLEAR_SELECTION")

#Perform the selections by attributes and intersections to generate scores of 2
def score_two_by_selection(wetlands, protection_areas, items):
    #print("Scoring twos by selection...")
    ## Select Protection Areas and Wellhead Critical Areas with surface water connections
    strWhere2Pt = """"P_TYPE" IN ('Protection Area', 'rotection Area', 'Wellhead Critical Area')
AND "FAC_SRC" IN ('GU', 'SW') """
    arcpy.SelectLayerByAttribute_management(protection_areas, "NEW_SELECTION",
strWhere2Pt)
    logger.info("Protection Areas and Wellhead Critical Areas with surface water connections
selected")

    ## Select Wetland Units within Protection Areas or Wellhead Critical Areas with surface
connections.
    arcpy.SelectLayerByLocation_management(wetlands, "INTERSECT", protection_areas, "",
"NEW_SELECTION")
    logger.info("Wetland Units within Protection Areas or Wellhead Critical Areas with surface
connections selected")

    ## Select Wetland Units within Zone of Critical Concern.
    arcpy.SelectLayerByLocation_management(wetlands, "INTERSECT", globalvars.ZCC_WV,
"", "ADD_TO_SELECTION")
    logger.info("Wetland Units within Zone of Critical Concern added to selection")
    with arcpy.da.SearchCursor(wetlands, ["OID@"]) as sc:
        for record in sc:
            oid = record[0]
            items[oid]['score'] = max(items[oid]['score'], 2)
    del sc
    logger.info("2 points scores generated for wetland units")

    arcpy.SelectLayerByAttribute_management(protection_areas, "CLEAR_SELECTION")
    arcpy.SelectLayerByAttribute_management(wetlands, "CLEAR_SELECTION")

#do a timed run of the wetland process
def PublicWaterSupplyCheck(wetlands, intakes, zpcs, protection_areas, pswi_watersheds):
    items = None

    fcPublicWaterIntakeIntersect = "Surface_Intake_Drainage_Area_Intersect"

    cleanup_workspace()

    #perform the intersection used for scoring
    get_wetland_unit_contribution(wetlands, intakes, "WU_WaterSupply")

    #score based on the wetland
    items = classify_wetlands_by_intersection(wetlands, pswi_watersheds)

    #1s
    score_one_by_selection(wetlands, zpcs, protection_areas, items)

    #2s
    score_two_by_selection(wetlands, protection_areas, items)
    output_results(items, wetlands)


    return items
    logger.info("completed output WU_WaterSupply")
'''
def sanity_check(items, wetlands):
    #print("\nSanity check for results: ")
    check = len(items) == int(arcpy.GetCount_management("WU_WaterSupply").getOutput(0))
    #print "Number of Wetlands in FC is consistent with number of results: {}\n".format(check,)
    #print("Total wetland counts per score:\n")
    zeroes = 0
    ones = 0
    twos = 0
    for item in items:
        val = items[item]['score']
        if val == 2:
            twos += 1
        elif val == 1:
            ones += 1
        else:
            zeroes += 1
    #print("zeroes : {}".format(str(zeroes).rjust(8),))
    #print("ones   : {}".format(str(ones).rjust(8),))
    #print("twos   : {}".format(str(twos).rjust(8),))
    total = ones + twos + zeroes
    #print("-" * 17)
    #print("Total  : {}".format(str(total).rjust(8),))
    logger.info("Completed sanity check for results")
```

```
'''
def output_results(items, wetlands):
    ws = arcpy.env.workspace
    template = globalvars.PSWI_TEMPLATE
    target_path = os.path.join(ws, "WU_WaterSupply")
    if arcpy.Exists(template):
        if arcpy.Exists(target_path):
            arcpy.Delete_management(target_path)


        sr = arcpy.SpatialReference(26917)
        arcpy.CreateFeatureclass_management(arcpy.env.workspace, "WU_WaterSupply",
"POLYGON", template, "DISABLED", "DISABLED", sr)
        logger.info("empty feature class created for WU_WaterSupply")
        #
        arcpy.SelectLayerByAttribute_management(wetlands,"CLEAR_SELECTION")
        with arcpy.da.SearchCursor(wetlands, ["OID@", "Shape@", "WUKey"]) as sc,
arcpy.da.InsertCursor(target_path, ["OID@", "Shape@", "WUKey", "PctInt", "WaterSupply"])
as ic:
            for record in sc:
                # [oid, geometry, ratio, score]
                wlu = record[0]
                row = [wlu, record[1], record[2], items[wlu][RATIO] * 100, items[wlu]['score']]
                ic.insertRow(row)
            del sc
            del ic
        logger.info("stored records loaded into WU_WaterSupply")
    else:
        logger.warn("The supplied output feature class {} template is missing.".format(template,))
```

5.7.112 Wetland Functional Assessment Tool Launch Script

```
################################################################################
#
# File Name: wfa.py
# Developer: Yibing Han @ West Virginia GIS Tech Center
# Methodology: Elizabeth Byers @ WV DEP
# Date: 02/2018
# Purpose:
#    This is the main launching script that initiates the Wetland Functional Assessment.
# For questions or concerns, please contact:
#     Email: Yibing Han (yibing.han@mail.wvu.edu) / Kurt Donaldson
(kurt.donaldson@mail.wvu.edu)
#     Mailing Address:
#     WV GIS Technical Center
#     WVU Department of Geology & Geography
#     330 Brooks Hall
#     P.O. Box 6300
#     Morgantown, WV 26506
#     Phone: (304) 293-0557
#
################################################################################
#!/usr/bin/python
import sys, os
sys.path.append(os.path.abspath("."))

import gc
gc.enable()

import datetime
import time
import logging
import traceback
import arcpy

from globalvars import globalvars
from utilities import initRequest, actions, calcFunction, calcAllResults
from wquality import WQuality
from floodattn import FloodAttn
from habeco import HabEco

#scriptPath = os.path.abspath(__file__)
#thisFolder = os.path.dirname(scriptPath)

arcpy.env.overwriteOutput = True

#Set logger
logger = logging.getLogger("WFA")
logger.setLevel(logging.INFO)
#loggerFilePath = os.path.join(thisFolder, "logs")
loggerFile = os.path.join(globalvars.srcLogFolder, "WFA_" +
str(time.strftime("%Y%m%d_%H%M%S")) + ".txt")
fh = logging.FileHandler(loggerFile)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
fh.setFormatter(formatter)
logger.addHandler(fh)

ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
ch.setFormatter(formatter)
logger.addHandler(ch)
logger.info("program started")

uid = "WetlandFunction"
strUploadFile = sys.argv[1]
globalvars.srcInput = strUploadFile
fcWetlandInput = arcpy.mapping.Layer(strUploadFile)

arcpy.AddMessage("Input Feature Class: " + str(strUploadFile))

# Geodatabase/Workspace
fcGeodbFCName = uid + "_" + str(time.strftime("%Y%m%d_%H%M%S"))
arcpy.CreateFileGDB_management(globalvars.srcGDBFolder, fcGeodbFCName)
strWorkspace = globalvars.srcGDBFolder + "\\" + fcGeodbFCName + ".gdb"
arcpy.env.workspace = strWorkspace

arcpy.AddMessage("Target Geodatabase: " + str(fcGeodbFCName))

# initial run to create variables used across the sections
logger.info("Running initial functions to create variables used across the sections")
arcpy.AddMessage("Running initial functions to create variables used across the sections")
try:
fcWU = arcpy.mapping.Layer(initRequest.MainUtil(fcWetlandInput))
arcpy.AddMessage("Initial run completed to create variables used across the sections")
except arcpy.ExecuteError:
msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
arcpy.AddError(msgs)
logger.error(msgs)
arcpy.AddMessage("Initial run to created variables failed")
```

```
sys.exit(1)
for handler in logger.handlers:
            logger.removeHandler(handler)
except:
tb = sys.exc_info()[2]
tbinfo = traceback.format_tb(tb)[0]
pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

arcpy.AddError(pymsg)
arcpy.AddError(msgs)
logger.error(pymsg)
logger.error(msgs)
arcpy.AddMessage("Initial run to created variables failed")
sys.exit(1)
for handler in logger.handlers:
            logger.removeHandler(handler)

# run the Water Quality Assessment
logger.info("Running Water Quality Assessment")
arcpy.AddMessage("Running Water Quality Assessment")
try:
    WQuality.RunWaterQuality(fcWU)
    arcpy.AddMessage("Water Quality Assessment completed")
except arcpy.ExecuteError:
    msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
    arcpy.AddError(msgs)
    logger.error(msgs)
    arcpy.AddMessage("Habitat and Ecological Integrity Assessmemt failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)
except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]
    pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
    msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg)
    arcpy.AddError(msgs)
    logger.error(pymsg)
    logger.error(msgs)
    arcpy.AddMessage("Water Quality Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)

# run Flood Attenuation Assessment
logger.info("Running Flood Attenuation Assessment")
arcpy.AddMessage("Running Flood Attenuation Assessment")
try:
    FloodAttn.RunFloodAttn(fcWU)
    arcpy.AddMessage("Flood Attenuation Assessment completed")
except arcpy.ExecuteError:
    msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
    arcpy.AddError(msgs)
    logger.error(msgs)
    arcpy.AddMessage("Flood Attenuation Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)
except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]
    pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
    msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg)
    arcpy.AddError(msgs)
    logger.error(pymsg)
    logger.error(msgs)
    arcpy.AddMessage("Flood Attenuation Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)

# run Habitat and Ecological Integrity Assessment
logger.info("Running Habitat and Ecological Integrity Assessment")
arcpy.AddMessage("Running Habitat and Ecological Integrity Assessment")
try:
    HabEco.RunHabEco(fcWU)
    arcpy.AddMessage("Habitat and Ecological Integrity Assessment completed")
except arcpy.ExecuteError:
    msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
    arcpy.AddError(msgs)
    logger.error(msgs)
    arcpy.AddMessage("Habitat and Ecological Integrity Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)
except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]
    pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
    msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg)
    arcpy.AddError(msgs)
    logger.error(pymsg)
    logger.error(msgs)
    arcpy.AddMessage("Habitat and Ecological Integrity Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)

# run Final Assessment that aggregates all three previous sections
logger.info("Running final assessment that aggregates all three previous sections")
arcpy.AddMessage("Running final assessment that aggregates all three previous sections")
try:
    calcFunction.CalcFunction()
    arcpy.AddMessage("Final Assessment (Pt.1) completed")
except arcpy.ExecuteError:
    msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
    arcpy.AddError(msgs)
    logger.error(msgs)
    arcpy.AddMessage("Final Assessment failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)
except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]
    pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
    msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg)
    arcpy.AddError(msgs)
    logger.error(pymsg)
    logger.error(msgs)
    arcpy.AddMessage("Final Assessment (Pt.1) failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)

# run Final Assessment that aggregates all scoreing fields
logger.info("Running final assessment that aggregates all three previous sections")
arcpy.AddMessage("Running final assessment that aggregates all three previous sections")
try:
    calcAllResults.CalcAllResults(fcWU)
    arcpy.AddMessage("Final Assessment (Pt.2) completed")
except arcpy.ExecuteError:
    msgs = "EXECUTE ERRORS:\n" + arcpy.GetMessages(2) + "\n"
    arcpy.AddError(msgs)
    logger.error(msgs)
    arcpy.AddMessage("Final Assessment (Pt.2) failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)
except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]
    pymsg = "PYTHON ERRORS:\nTraceback info:\n" + tbinfo + "\nError Info:\n" +
str(sys.exc_info()[1])
    msgs = "ArcPy ERRORS:\n" + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg)
    arcpy.AddError(msgs)
    logger.error(pymsg)
    logger.error(msgs)
    arcpy.AddMessage("Final Assessment (Pt.2) failed")
    sys.exit(1)
    for handler in logger.handlers:
        logger.removeHandler(handler)

logger.info("program ended")
for handler in logger.handlers:
    logger.removeHandler(handler)
```