American Marten Habitat Suitability

Goal

To identify if Pennsylvania suitable habitat areas for American marten.

Description

Suitable habitat areas will be determined using multiple natural resource rasters. Each raster will be given Habitat Suitability Indices (HSI) based off Tom Keller and other expert opinions. A sum of the rasters will provide areas of high suitability on the landscape. Using suitable areas, site visits will be conducted for field verification. If it is determined that suitability areas exist, reintroduction of the species could be discussed as an option.

Habitat Suitability Project Lead

Tom Keller

Furbearer Biologist, Pennsylvania Game Commission

Questions related to reintroduction information can be directed to: (717) 787-5529 or PAmarten@pa.gov

Habitat Suitability Geospatial Lead

Emily Clees

Geospatial Specialist, Northcentral Region, Pennsylvania Game Commission

Questions on the habitat suitability anslysis can be directed to: eclees@pa.gov

image credit: https://i2.wp.com/24.media.tumblr.com/tumblr_ma7nv7qdtg1rzpxtno1_400.jpg

Table of Contents

I. Data

Data Source Links

1. Land Cover

- **NLCD Land Cover** will be used as our land cover data
  - 2019 CONUS NLCD was used

2. Snow Cover

The National Weather Service, National Operational Hydrologic Remote Sensing Center, National Snowfall Analysis was used

- **NWS National Snowfall Analysis**
  - Downloaded yearly snowfall data from 2009 - 2019 then calculated an average over those years

3. Percent Canopy Cover

Land Fire's "Existing Vegetation Cover" was used for Percent Canopy Cover

- **Land Fire Data Download** will be used for percent canopy cover
  - LF 2020us_210 Existing Vegetation Cover was used
- **Existing Vegetation Cover Description**

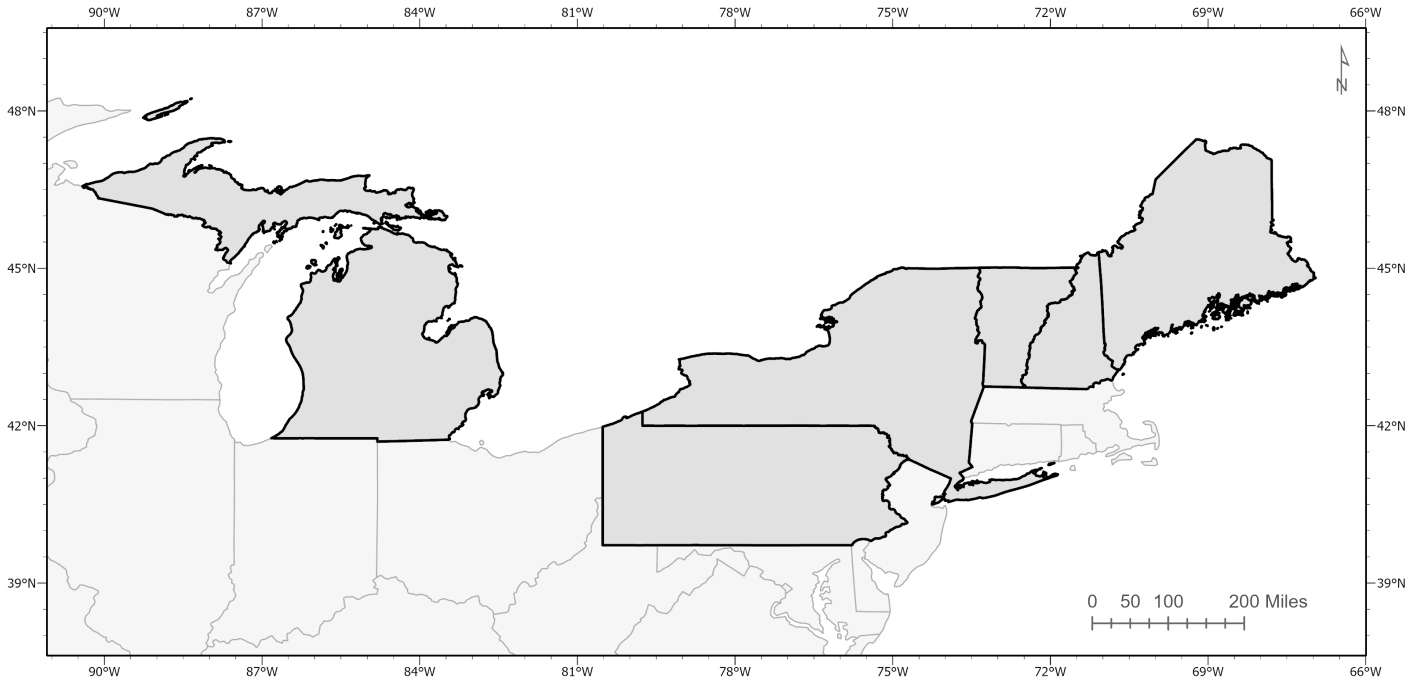
4. Stand Age/ Tree Height

Land Fire's "Existing Vegetation Height" was used for Percent Canopy Cover

- **Land Fire Data Download** will be used for percent canopy cover
  - LF 2020us_210 Existing Vegetation Height was used
  - Vegetation Height was used as a metric for stand age
- **Existing Vegetation Height Description**

Study Area

- What is the study area?
  - Pennsylvania, Maine, New Hampshire, Vermont, New York, and Michigan
  - States other than Pennsylvania were used to compaire existing marten populations to areas of high suitability in Pennslyvania



II. Data Prep

Merging Data

In [5]:
```
import os

    # In this step, I am setting up my folders and work spaces. By naming the folder
    locations and only using the name in the code below,
    # I can use this code in other projects and link to new datasets by changing the
    locations in 1 spot in the code.

    #Data Location - Where data used in the project was downloaded to
    data_location = r"P:\Projects\BWM_AmericanMartenHSI\Data"

    #Geodatabase Location - Where any feature classes for the project are saved
    gdb = r"P:\Projects\BWM_AmericanMartenHSI\AmMarten_HSI_Python.gdb"

    #Output Location - Where any outputs from this code are stored
    out_location = r"P:\Projects\BWM_AmericanMartenHSI\Output"

    print("os imported")
    print("Folders Assigned")
```

os imported

Folders Assigned

In [1]:
```
#List of all Rasters included in the Analysis

#Set Environment Workspace to data_location
arcpy.env.workspace = data_location

# Creating lists of each dataset allows me to call these lists later in code without
listing each layer one by one. The data was
# downloaded by state, so it will need to be combined to one raster in code below.

#List of all National Land Cover data
nlcd_list = arcpy.ListRasters("NLCD_2019*")

#Using "NLCD_2019*" looks for data in the workspace listed above that starts with
"NLCD_2019". Place the * on either side or both sides
# depending on what text is the same in each raster

#List of all Land fire Canopy Cover data
cover_list = arcpy.ListRasters("*_evc")

#List of all Land Fire Canopy Height data
height_list = arcpy.ListRasters("*_evh")
```

In [2]:
```
#By using the list Statements above, I am able to create Raster Mosaics to combine all
of the rasters by state into one layer

#Merge the National Land Cover data to one raster layer
am_nlcd = arcpy.MosaicToNewRaster_management(nlcd_list, out_location,
                                             "am_nlcd.tif", "", "8_BIT_UNSIGNED", "30",
"1", "LAST", "FIRST")

#This print statement shows the path of where the layer is saved
#print(am_nlcd)

#Merge the Percent Canopy Cover data to one raster layer
am_cover = arcpy.MosaicToNewRaster_management(cover_list, out_location,
                                              "am_cover.tif", "", "8_BIT_UNSIGNED", "30",
"1", "LAST", "FIRST")
#print(am_cover)

#Merge the Tree Height data to one raster layer
am_height = arcpy.MosaicToNewRaster_management(height_list, out_location,
                                              "am_height.tif", "", "8_BIT_UNSIGNED",
"30", "1", "LAST", "FIRST")
#print(am_height)

print("Mosaics are created")
```

Mosaics are created

Average Snow Accumulation

- We downloaded a 10 year window of snow accumulation layers, but an average is needed
  - Average all of the snow accumulation layers using Cell Statistics

In [3]:
```
import arcpy
```

```
import os

#Set Environment Workspace to data_location
arcpy.env.workspace = data_location

#List of all Snow data
snow_list = arcpy.ListRasters("sfav2_CONUS_*")

#print(snow_list)

#Average the snow tifs from the list above
snow_avg = arcpy.sa.CellStatistics(snow_list, "MEAN")

print("Snow Average Complete")
```

Snow Average Complete

III. Defining HSI Values

What is Habitat Suitability Index (HSI)?

- HSI gives variables a numerical index
  - The numerical index is used to represent the ability for that variable to support a selected species
  - An example of the methods used to assign HSI can be seen below

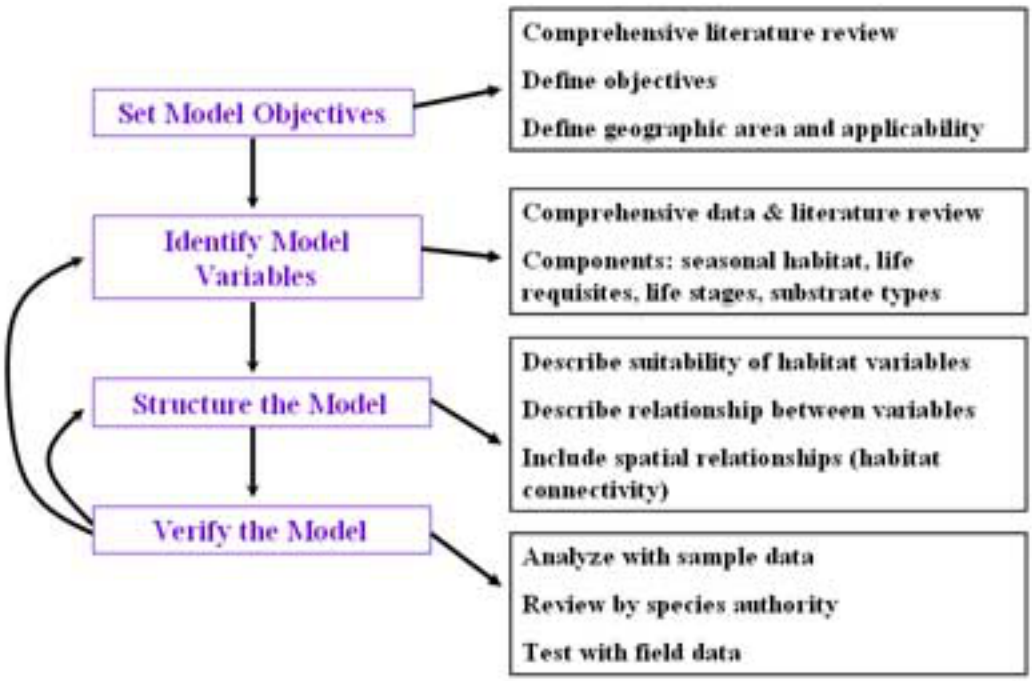


Figure 1. Steps for constructing the HSI.

- An example of an HSI can be seen below
  - Typically the numberical values are decimals from 0 to 1.0
  - For our analysis, I assigned values from 0 to 100 for our HSI values

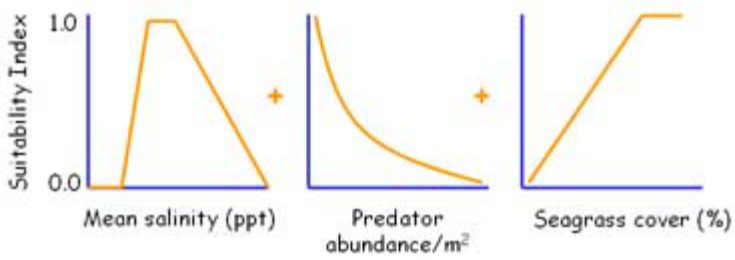


Figure 2. The Habitat Suitability Index (HSI) represents the combined interactions of all species-habitat relationships. The output is a numerical index that scores the capacity of a habitat to support the selected species (ranges from 0-1)

HSI Image Source: https://archive.epa.gov/aed/html/research/scallop/web/html/hsi.html

- Examples of HSI Models
  - **American Marten HSI** some parameters from this paper were used in our analysis
  - **Understanding HSI** More options for HSI analyses
  - **Water Management HSI Example** Another example of HSI used in South Florida

HSI Values for American marten parameters

Land Cover HSI

Attribute	HSI Value
All other Land Classes	NO DATA
Shrub	25
Deciduous	50
Mixed	75
Coniferous	100

Snow Cover HSI

Attribute	HSI Value
0 cm	NO DATA
10 - 34 cm	25
35 - 59 cm	50
60 - 90 cm	75
> 91cm	100

Percent Canopy Cover HSI

Attribute	HSI Value
0 - 25%	NO DATA
N/A	25
26 - 50%	50
N/A	75
> 50%	100

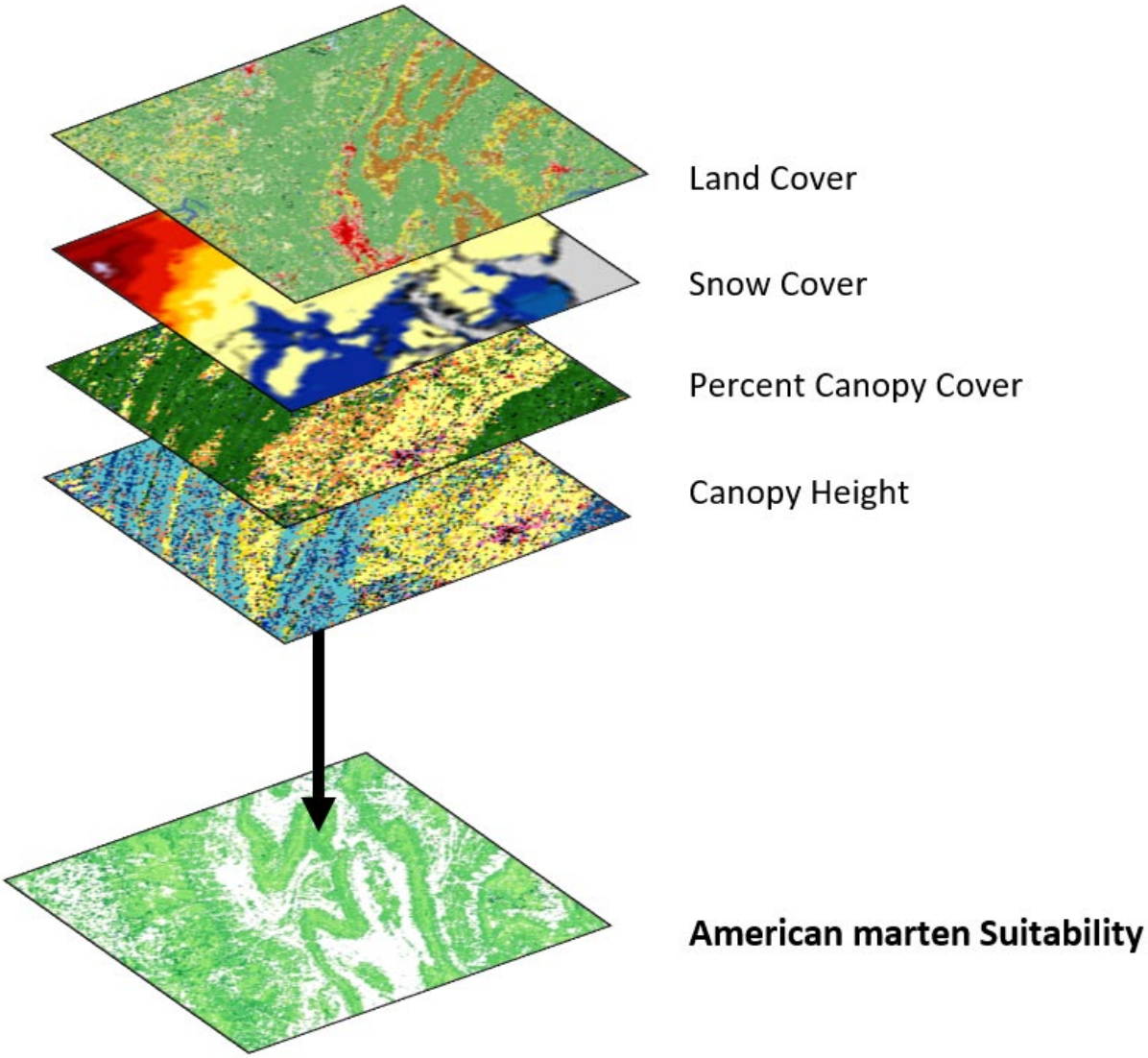
Tree Height HSI

Attribute	HSI Value
< 1m	NO DATA
1 - 7m	25
8 - 13m	50
14 - 20m	75
> 20m	100

IV. Analyses

Analysis

In analysis 2, coarse woody debris was taken out before the summation of the layers. The layer may not be accurate to what is present in the field, so an analysis was run, leaving the layer out.



Analyses

In [6]:
```
import arcpy

    # Set workspace and Environments
    # This is important to make sure all layers are aligned

    gdb = r"P:\Projects\BWM_AmericanMartenHSI\AmMarten_HSI_Python.gdb"
    arcpy.env.workspace = gdb
    arcpy.env.outputCoordinateSystem = am_nlcd[0]
    arcpy.env.mask = am_nlcd[0]

    print(arcpy.env.outputCoordinateSystem.name)

    #Set up link to study area
    study_states = os.path.join(gdb, "BufferedStudyArea")

    # Extract by mask used to clip all rasters to the study states and sets the environments
```

```
to align all rasters

#Snow Extract to Study States
am_snow_states = arcpy.sa.ExtractByMask(snow_avg, study_states)
# am_snow_states.save(os.path.join(out_location, "am_snow_states.tif"))

# Using "os.path.join" allows you to save the output to a specified location in Step 1
above. You will not need to change these for
# future projects then

#NLCD Extract to Study States
am_nlcd_states = arcpy.sa.ExtractByMask(am_nlcd, study_states)
# am_nlcd_states.save(os.path.join(out_location, "am_nlcd_states.tif"))

#Height Extract to Study States
am_height_states = arcpy.sa.ExtractByMask(am_height, study_states)
# am_height_states.save(os.path.join(out_location, "am_height_states.tif"))

#Canopy Cover Extract to Study States
am_cover_states = arcpy.sa.ExtractByMask(am_cover, study_states)
# am_cover_states.save(os.path.join(out_location, "am_cover_states.tif"))

print("Data Extracted to Study States")
```

```
Albers_Conical_Equal_Area
Data Extracted to Study States
```

In [7]:
```
#Reclassify Layers to attach HSI Values to each raster cell. See tables above for list
of hsi values.

# Coarse Woody Debris Reclassify
am_cwd_hsi_rr = arcpy.sa.RemapRange([[91, 182, "NODATA"],
                                     [183, 189, 100],
                                     [201, 204, 100]])

am_cwd_hsi = arcpy.sa.Reclassify(am_cwd_states, "VALUE", am_cwd_hsi_rr)

#National Land Cover Reclassify
am_nlcd_hsi_rr = arcpy.sa.RemapRange([[-1, 31, "NODATA"],
                                      [41, 41, 50],
                                      [42, 42, 100],
                                      [43, 43, 75],
                                      [51, 51, "NODATA"],
                                      [52, 52, 25],
                                      [71, 95, "NODATA"]])

am_nlcd_hsi = arcpy.sa.Reclassify(am_nlcd_states, "VALUE", am_nlcd_hsi_rr)

#Adding in areas that are mixed as a value of 100 as well


## Redo Reclassify values from here down
#Tree Height Reclassify
am_height_hsi_rr = arcpy.sa.RemapRange([[-1, 100, "NODATA"],
                                        [101, 107, 25],
```

```
                                        [108, 113, 50],
                                        [114, 120, 75],
                                        [121, 199, 100],
                                        [201, 310, "NODATA"]])

    am_height_hsi = arcpy.sa.Reclassify(am_height_states, "VALUE", am_height_hsi_rr)

    #Percent Canopy Cover Reclassify
    am_cover_hsi_rr = arcpy.sa.RemapRange([[-1, 125, "NODATA"],
                                        [126, 150, 50 ],
                                        [151, 199, 100],
                                        [201, 399, "NODATA"]])

    am_cover_hsi = arcpy.sa.Reclassify(am_cover_states, "VALUE", am_cover_hsi_rr)

    #Snow Average Reclassify failed - does not have valid statistics
    am_snow_hsi_rr = arcpy.sa.RemapRange([[-100000, 0.999999, "NODATA"],
                                       [3.93, 13.78, 25],
                                       [13.78, 23.62, 50],
                                       [23.62, 36, 75],
                                       [36, 280, 100]])

    am_snow_hsi = arcpy.sa.Reclassify(am_snow_states, "VALUE", am_snow_hsi_rr)


    print("HSI Values Assigned")
```

HSI Values Assigned

In [4]:

```
    # Set up datalists for the analysis
    # Setting up the environments as well

    gdb = r"P:\Projects\BWM_AmericanMartenHSI\AmMarten_HSI_Python.gdb"
    arcpy.env.workspace = gdb
    arcpy.env.outputCoordinateSystem = am_nlcd[0]
    arcpy.env.mask = am_nlcd[0]
    arcpy.env.cellSize = am_nlcd[0]

    # Below are the two options for running the final analysis. We chose to omit the coarse
    woody debris, cwd, data
    # in the final analysis because it did not seem to give us the results we wanted.

    # Option 1 - Including Coarse Woody Debris
    #am_hs_cwd_data = [am_cwd_hsi, am_nlcd_hsi, am_height_hsi, am_snow_hsi, am_cover_hsi]

    # Analysis - Coarse Woody Debris, Land Cover, Tree Height, % Canopy Cover, Snow added
    together
    #am_hs_cwd = arcpy.sa.CellStatistics(am_hs_cwd_data, "SUM")
    #am_hs_cwd.save(os.path.join(out_location, "am_hs_cwd.tif"))


    #Option 2 - Removing Coarse Woody Debris. This is the option we chose.
    am_hs_no_cwd_data = [am_nlcd_hsi, am_height_hsi, am_snow_hsi, am_cover_hsi]

    # Analysis - Land Cover, Tree Height, % Canopy Cover, Snow added together
```

```
am_hs_no_cwd = arcpy.sa.CellStatistics(am_hs_no_cwd_data, "SUM")
am_hs_no_cwd.save(os.path.join(out_location, "am_hs_no_cwd.tif"))

print("Analysis Complete")
```

```
Analysis Complete
```

In [13]:
```
#Adding focal moving window analysis

gdb = r"P:\Projects\BWM_AmericanMartenHSI\AmMarten_HSI_Python.gdb"
arcpy.env.workspace = gdb
arcpy.env.outputCoordinateSystem = am_nlcd[0]
arcpy.env.mask = am_nlcd[0]
arcpy.env.cellSize = am_nlcd[0]

#Defining where each state's buffer is located for the extract by mask

pa_state = os.path.join(gdb, "Buffer10mi_PA_State")
mi_state = os.path.join(gdb, "Buffer10mi_MI_State")
me_state = os.path.join(gdb, "Buffer10mi_ME_State")
nh_state = os.path.join(gdb, "Buffer10mi_NH_State")
vt_state = os.path.join(gdb, "Buffer10mi_VT_State")
ny_state = os.path.join(gdb, "Buffer10mi_NY_State")

am_hs_no_cwd_final = (os.path.join(out_location, "am_hs_no_cwd.tif"))

#Extracting by mask the HSI tif for each state to run the focal statistic by state.
There is not enough room to run the entire HSI
# output in one round

am_hs_no_cwd_pa = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, pa_state)
am_hs_no_cwd_mi = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, mi_state)
am_hs_no_cwd_me = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, me_state)
am_hs_no_cwd_nh = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, nh_state)
am_hs_no_cwd_vt = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, vt_state)
am_hs_no_cwd_ny = arcpy.sa.ExtractByMask(am_hs_no_cwd_final, ny_state)


# Running Focal Statistics for each State and saving the tif to my output location

am_hs_no_cwd_pa_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_pa, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_pa_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_pa_fs_norc.tif"))

am_hs_no_cwd_mi_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_mi, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_mi_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_mi_fs_norc.tif"))

am_hs_no_cwd_me_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_me, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_me_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_me_fs_norc.tif"))

am_hs_no_cwd_nh_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_nh, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_nh_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_nh_fs_norc.tif"))
```

```
am_hs_no_cwd_vt_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_vt, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_vt_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_vt_fs_norc.tif"))

am_hs_no_cwd_ny_fs_norc = arcpy.sa.FocalStatistics(am_hs_no_cwd_ny, "Circle 54 CELL",
"MEAN", "NODATA")
am_hs_no_cwd_ny_fs_norc.save(os.path.join(out_location, "am_hs_no_cwd_ny_fs_norc.tif"))
```

Truthing the Model Against Known Marten Populations

In [7]:
```
import os

# Setting Envrionments and Workspace for tools below

gdb = r"P:\Projects\BWM_AmericanMartenHSI\AmMarten_HSI_Python.gdb"
arcpy.env.workspace = gdb
arcpy.env.outputCoordinateSystem = am_nlcd[0]
arcpy.env.mask = am_nlcd[0]
arcpy.env.cellSize = am_nlcd[0]

#iNaturalist Point Analysis

#Converting iNaturalist points to rasters so that I can exctract the HSI value for each
raster cell where a sighting occured.

arcpy.conversion.PointToRaster("MartenObservationsINat", "id", os.path.join(gdb,
"MartenObservationsINat_Raster"), "MOST_FREQUENT", "NONE", 30, "BUILD")

#Extracting HSI values for each known marten sighting from iNaturalist

me_inat_hsi =
arcpy.sa.ExtractByMask(os.path.join(out_location,"am_hs_no_cwd_me_fs_norc.tif"),
"MartenObservationsINat_Raster")
me_inat_hsi.save(os.path.join(out_location, "me_inat_hsi.tif"))

mi_inat_hsi = arcpy.sa.ExtractByMask(os.path.join(out_location,
"am_hs_no_cwd_mi_fs_norc.tif"), "MartenObservationsINat_Raster")
mi_inat_hsi.save(os.path.join(out_location, "mi_inat_hsi.tif"))

nh_inat_hsi = arcpy.sa.ExtractByMask(os.path.join(out_location,
"am_hs_no_cwd_nh_fs_norc.tif"), "MartenObservationsINat_Raster")
nh_inat_hsi.save(os.path.join(out_location, "nh_inat_hsi.tif"))

vt_inat_hsi = arcpy.sa.ExtractByMask(os.path.join(out_location,
"am_hs_no_cwd_vt_fs_norc.tif"), "MartenObservationsINat_Raster")
vt_inat_hsi.save(os.path.join(out_location, "vt_inat_hsi.tif"))

ny_inat_hsi = arcpy.sa.ExtractByMask(os.path.join(out_location,
"am_hs_no_cwd_ny_fs_norc.tif"), "MartenObservationsINat_Raster")
ny_inat_hsi.save(os.path.join(out_location, "ny_inat_hsi.tif"))

#Converting the Rasters back to points to be able to run statistics on the feature
layer. This allowed me to average all of the HSI
# values from each sighting point together
```

```
arcpy.conversion.RasterToPoint("me_inat_hsi", os.path.join(gdb, "me_inat_rasterpoint",
"Value")

arcpy.conversion.RasterToPoint("mi_inat_hsi", os.path.join(gdb,"mi_inat_rasterpoint",
"Value")

arcpy.conversion.RasterToPoint("nh_inat_hsi", os.path.join(gdb,"nh_inat_rasterpoint",
"Value")

arcpy.conversion.RasterToPoint("vt_inat_hsi", os.path.join(gdb,"vt_inat_rasterpoint",
"Value")

arcpy.conversion.RasterToPoint("ny_inat_hsi", os.path.join(gdb,"ny_inat_rasterpoint",
"Value")


# Michigan Collared Marten Home Ranges

#Extracted the HSI layer by each of the marten home ranges

out_raster = arcpy.sa.ExtractByMask("am_hs_no_cwd.tif", "All_MI_MartenPops95")
#all_mi_marten95_hsi.save(os.path.join(out_location, "all_mi_marten95_hsi")

# Converted each of the rasters within each marten home range to individual points
arcpy.conversion.RasterToPoint("all_mi_marten95_hsi", os.path.join(out_location,
"all_mi_marten95"), "Value")

# Summarized all of the points within each of the marten home ranges to get the average
HSI within each known marten home range
arcpy.analysis.SummarizeWithin("All_MI_MartenPops95", "all_mi_marten95 avg HSI: 329.4",
os.path.join(out_location, "All_MI_MartenPops95_SummarizeWithin", "KEEP_ALL", "grid_code
Mean;grid_code Max;grid_code Min", "ADD_SHAPE_SUM", '', None, "NO_MIN_MAJ",
"NO_PERCENT", None)

#Converted each of the summary polygons to a point so that I could add each homerange to
the point layer to average all known marten HSIs together
arcpy.management.FeatureToPoint("All_MI_MartenPops95_SummarizeWithin 336",
os.path.join(gdb, "All_MI_MartenPops95_SumPts", "INSIDE")
```

V. Documentation References

Cell Statistics

- https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/cell-statistics.htm

Environment Settings

- https://pro.arcgis.com/en/pro-app/2.8/tool-reference/environment-settings/an-overview-of-geoprocessing-environment-settings.htm

Euclidean Distance

https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/euclidean-distance.htm

Extract by Mask

- https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/extract-by-mask.htm

Focal Statistics

- https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/focal-statistics.htm

ListRasters

- https://pro.arcgis.com/en/pro-app/2.8/arcpy/functions/listrasters.htm

Mosaic to New Raster

- https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/mosaic-to-new-raster.htm

Reclassify

- https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/reclassify.htm

Remap Range

- https://pro.arcgis.com/en/pro-app/latest/arcpy/spatial-analyst/an-overview-of-transformation-classes.htm