**Submit to Moodle**

1. Project progress report. (Create the project progress report based on the instructions below. )
2. Your current Python script.

*You must submit an ArcGIS Project folder.  Inside, the .aprx and .atbx files should be at the top level.  Other files should be organized into****data, documentation****, and****code****subdirectories inside the project folder.  Be sure to test your script tool and code to make sure they work in this configuration.*

***usa\_rollercoasters (project folder)*** *|  
      usa\_rollercoasters.gdb (you can use this, but it’s not required)* ***usa\_rollercoasters.aprx******coaster\_speed.atbx******code (folder)*** *|  
               --main.py  
               --tracks.py  
               --speed.py* ***data (folder)*** *|  
              --usa\_coaster\_fan\_clubs.shp  
              --amusement\_parks.tif  
              --coaster\_lines.shp* ***documentation (folder)*** *|  
              --coaster\_writeup.docx  
              --coaster\_report.pptx  
              --coaster\_presentation\_part1.mp4  
              --coaster\_presentation\_part2.mp4  
              --thumbnail\_coaster\_pic.png*

1. A Toolbox (with your current GUI).

**How to create the progress report**

*Revise* original content based on any change of plans and based on comments in the feedback you received on your proposal. Highlight changes you make to the original submission (use a **bold font** for these parts). [If you were assigned a different project, submit a proposal for this project but there will be no bold sections]. As indicated below, you'll *add*three new sections to the proposal, one at the beginning on *how the instructor feedback from the first proposal submission was addressed* and two at the end to indicate your *current progress* and *how you'll be fulfilling the checklist requirements*. Also, move the batch processing into the checklist.  
  
This project report should have the following structure:

1. **Summarize the instructor feedback (and peer feedback, where applicable) from the first submission and how you addressed these.** (If you were assigned a new project, state that here and explain why.)
2. **Title:**North Carolina Woodpecker Species Distribution Modeling using Presence-Only Prediction
3. **Abstract:** A major challenge in understanding the distribution of species is that observational point data tends to be primarily recorded in locations easily accessible to people. One method of estimating the distribution in a broader area is through a spatial statistics technique called Presence Only Prediction using Maximum Entropy (MaxEnt)1. The purpose of this project is to use the MaxEnt approach to estimate woodpecker species’ distributions in North Carolina from crowd-sourced bird watching dataset.
4. **Input data:** The data used in this project are from three primary sources:
   1. Project FeederWatch2 – Bird observation data, compiled by The Cornell Lab of Ornithology and Birds Canada. There are two file formats for the data: (1) *.xlsx* and (2) *.csv*. The Excel file contains the “data dictionary”, and is primarily needed so that the common names of the woodpecker species can be joined with the *.csv* data (in which they are instead identified by a unique ID). The *.csv* data contains the relevant data needed for this project including the species, date observed, and the location observed. The data can be obtained directly from the Project FeederWatch website via the dataset requests page: <https://feederwatch.org/explore/raw-dataset-requests/>.
   2. National Land Cover Database3 – using data from the Multi-Resolution Land Characteristics (MRLC) Consortium4, North Carolina State University created the NLCD. The data used in this project is clipped/re-projected land cover data for the state of North Carolina. <https://gisdata.lib.ncsu.edu/fedgov/mrlc/nlcd2019/NC_NLCD2019only.zip>
   3. North Carolina Digital Elevation Model (DEM)5 – this data was acquired directly through the NCSU GIS library at: <https://gisdata.lib.ncsu.edu/DEM/nc250.zip>. The data is a raster image of North Carolina elevation at a 1:250,000 scale.

The observation data will serve as the input point features for the MaxEnt models (each species will be modeled separately), while the two raster datasets will be the explanatory rasters.

1. **Data Products:** The original bird observation data is sparse, and tends to be most prevalent in human-accessible areas where birdwatching is popular. The Presence-Only Prediction (MaxEnt) Algorithm will use explanatory variables (i.e., elevation and land cover) to estimate how likely it might be to observe the same birds across the entire study area. The function available through *arcpy* provides several outputs:

* Trained raster (*see Figure 1 of the example from the arcpy Presence-Only Prediction documentation*6)
* Trained features (background point data generated to aid in training the model against the explanatory features)
* Descriptive tables (response curve, sensitivity)

The primary output of concern in the scope of this project is the trained raster, since it can be used to easily visualize the estimated distribution of the input bird species across the study area.

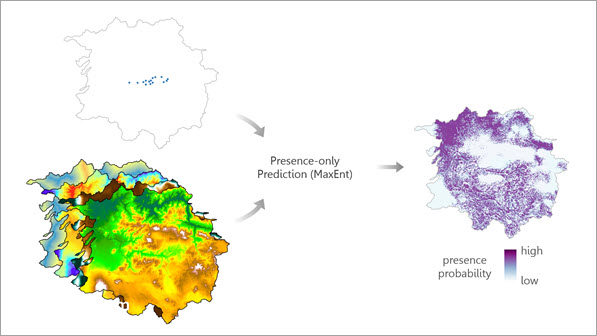
[](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/presence-only-prediction.htm)

Figure 1: Point data, explanatory raster(s), and other explanatory variables are used as inputs to the Presence-Only Prediction (MaxEnt) algorithm, and a presence probability raster is output for the defined study area. This image was taken from the arcpy Presence-Only Prediction documentation6

1. **Revised pseudocode:** Update based on feedback and add new sections. Be sure to include user input (GET) statements at the beginning of the pseudocode. Also include output statements (ADD data to the map, CAPTURE screenshot, INSERT screenshot in HTML, ADD tables to HTML, etc.)
2. **Keywords:**species distribution, raster, point data, digital elevation model, land cover, presence-only prediction, MaxEnt, statistics, birds
3. **Progress report:**Copy the code you have already written for the project and briefly explain what other progress has been made, such as data preparation or testing of Geoprocessing tools. (You will be evaluated on making at least some progress in the code and we will provide feedback on what you submit, so the more you can submit, the better.)
4. **Respond briefly to the requirement checklist:**

**Category I (Complete Cat. I a-f)**

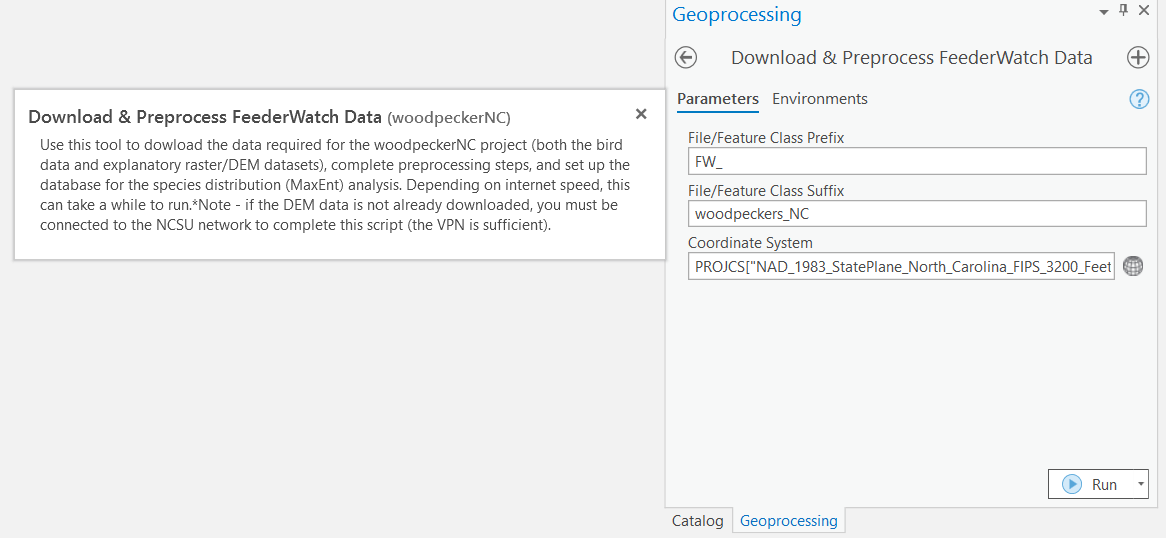
* 1. **Python geoprocessing:**Name one or two of the tools you will use.
  2. **Python batch processing (FOR or WHILE loops):**State the processing problem you are handling with Python.
  3. **Functions:** List descriptive names of 3 user-defined reusable functions along with a docstring for each. Remember not to use functions presented as examples and not to create functions that are simply wrappers for geoprocessing tools.
  4. **Class:**List a descriptive name and a docstring for a class you will use and one or more properties.
  5. **Mapping:** One or two sentences explaining how you'll use the mapping module and what you'll be automatically adding to the map.
  6. **GUI:**Submit a screenshot of your graphical user interface ([See Figure 1 for an example](https://lgtateos.github.io/gis540/project/proposalExtended.html#figure1)) and a table with a list of parameters ([should have the same columns as Table 1 below](https://lgtateos.github.io/gis540/project/proposalExtended.html#table1)). The parameter table should contain a list of the parameters you are using along with their data type and their properties (Data Type, Direction, Default value, Filter, and any other properties that you are setting). ALL parameters must have DEFAULT VALUES. The project should use at least 3 user inputs.  Use meaningful labels for each input.

**Category II  (Pick one or more--tell us which one(s) you’ll do and briefly describe)**

* 1. Reading and/or writing files with standard file objects or with cursors.
  2. Reading and/or writing HTML to automatically generate an output report.
  3. Reading and/or writing KML with BeautifulSoup.
  4. Downloading and uncompressing data from the web.

**Category III  (Pick one or more--tell us which one(s) you’ll do)**

* 1. Use the tool validator to create dynamic behavior for the script tool.
  2. Provide a portable toolbar button to launch the script tool.
  3. Inform the user of progress, using messages.
  4. Add the arcpy progressor to present messages.
  5. In the script tool, use the parameter filter property, the parameter "obtained from" property, the parameter symbology property, or the Schema property.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Data Type** | **Type** | **Direction** | **Default Value** | **Filter** |
| \_prefix | String | Optional | Input | FW\_ |  |
| \_suffix | String | Optional | Input | woodpeckers\_NC |  |
| coord\_sys | Coordinate System | Required | Input | NAD\_1983\_StatePlane\_North\_Carolina\_FIPS\_3200\_Feet |  |

**References**

1. <https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/presence-only-prediction.htm>
2. <https://feederwatch.org/about/project-overview/>
3. <https://www.lib.ncsu.edu/gis/nlcd>
4. <https://www.mrlc.gov/>
5. <https://www.lib.ncsu.edu/gis/dem>
6. <https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/presence-only-prediction.htm>