Connolly GIS 5571 Lab 3 Part 2

November 29, 2023

[75]:

```
import requests
import csv
import arcpy
import os
import io
from io import StringIO
import pandas as pd
from datetime import datetime, timedelta
```

[76]:

```
# Using the datetime tool, we can get today's date and format it as "YYYY-MM-DD"
today_date = datetime.now().strftime("%Y-%m-%d")
```

[77]:

```
#We use the requests tool to get the maximum temperature data as a CSV
#The end of the URL has been adjusted so that it takes in the current date
url = "https://ndawn.ndsu.nodak.edu/table.csv?
 ↪station=78&station=111&station=98&station=162&station=174&station=142&station=164&station=1

response = requests.get(url)
```

[79]:

```
#We then read in the CSV's text
#Since there were issues with all the entries on one row ending up in the same␣
 ↪cell, the following code breaks them up into columns
data = response.text
lines = data.strip().split('\n')[3:]
result = '\n'.join(lines)
```

[81]:

```
#We then can read the CSV data into a dataframe.
csv_file = StringIO(result)
dataframe = pd.read_csv(csv_file)
```

[82]:

```
#Since the first line after the column headers is units, we can drop that.
dataframe = dataframe.iloc[1:]
```

[ ]:

```
#First we convert the values of the temperature, latitude, and longitude␣
 ↪columns to numeric
dataframe['Max Temp'] = pd.to_numeric(dataframe['Max Temp'], errors='coerce')
dataframe['Latitude'] = pd.to_numeric(dataframe['Latitude'], errors='coerce')
```

```
dataframe['Longitude'] = pd.to_numeric(dataframe['Longitude'], errors='coerce')

# Then, we group by "Station Name" and calculate the mean for each group and␣
 ↪read that into a new dataframe
average_max_temp_df = dataframe.groupby("Station Name").agg({
    "Max Temp":'mean',
    'Latitude':'mean',
    'Longitude':"mean"
}).reset_index()
```

[84]:

```
#Then we perofrm the same procedure to gather the minimum temperature data

min_url = "https://ndawn.ndsu.nodak.edu/table.csv?
 ↪station=78&station=111&station=98&station=162&station=174&station=142&station=164&station=1
min_response = requests.get(min_url)

min_data= min_response.text

min_lines = min_data.strip().split('\n')[3:]

min_result = '\n'.join(min_lines)

min_csv_file = StringIO(min_result)

min_dataframe = pd.read_csv(min_csv_file)

min_dataframe = min_dataframe.iloc[1:]

min_dataframe['Min Temp'] = pd.to_numeric(min_dataframe['Min Temp'],␣
 ↪errors='coerce')
min_dataframe['Latitude'] = pd.to_numeric(min_dataframe['Latitude'],␣
 ↪errors='coerce')
min_dataframe['Longitude'] = pd.to_numeric(min_dataframe['Longitude'],␣
 ↪errors='coerce')

average_min_temp_df = min_dataframe.groupby("Station Name").agg({
    "Min Temp":'mean',
    'Latitude':'mean',
    'Longitude':"mean"
}).reset_index()
```

[92]:

```
# We then save the modified dataframes back to the CSV files
average_max_temp_df.to_csv('max_temp.csv', index=False)
average_min_temp_df.to_csv('min_temp.csv', index=False)
```

[93]:

```
#We are going to produe two sets of point features representing NDAWN stations.
#The first will be based off the maximum temperature data and use temperature␣
↪as the Z field
arcpy.management.XYTableToPoint(
    in_table="max_temp.csv",
    out_feature_class=r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS␣
↪5571 Lab3_2\GIS 5571 Lab3_2.gdb\output_modified_XYTableToPoint_max",
    x_field="Longitude",
    y_field="Latitude",
    z_field="Max Temp",
  ␣
↪coordinate_system='GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.
↪0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.
↪0174532925199433]],VERTCS["WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.
↪0,298.257223563]],PARAMETER["Vertical_Shift",0.0],PARAMETER["Direction",1.
↪0],UNIT["Meter",1.0]];-400 -400 1000000000;-100000 10000;-100000 10000;8.
↪98315284119521E-09;0.001;0.001;IsHighPrecision'
)
```

[93]:

```
<Result 'C:\\Users\\conno\\OneDrive\\Documents\\ArcGIS\\Projects\\GIS 5571
Lab3_2\\GIS 5571 Lab3_2.gdb\\output_modified_XYTableToPoint_max'>
```

[97]:

```
#We will create another set for the minimum temperature data.
arcpy.management.XYTableToPoint(
    in_table="min_temp.csv",
    out_feature_class=r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS␣
↪5571 Lab3_2\GIS 5571 Lab3_2.gdb\output_modified_XYTableToPoint_min",
    x_field="Longitude",
    y_field="Latitude",
    z_field="Min Temp",
  ␣
↪coordinate_system='GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.
↪0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.
↪0174532925199433]],VERTCS["WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.
↪0,298.257223563]],PARAMETER["Vertical_Shift",0.0],PARAMETER["Direction",1.
↪0],UNIT["Meter",1.0]];-400 -400 1000000000;-100000 10000;-100000 10000;8.
↪98315284119521E-09;0.001;0.001;IsHighPrecision'
)
```

[97]:

```
<Result 'C:\\Users\\conno\\OneDrive\\Documents\\ArcGIS\\Projects\\GIS 5571
Lab3_2\\GIS 5571 Lab3_2.gdb\\output_modified_XYTableToPoint_min'>
```

[98]:

```
#From here, we can create our interpolations. First we will use IDW for the␣
↪maximum temperature dataset
with arcpy.EnvManager(scratchWorkspace=r"C:
↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
↪Lab3_2.gdb"):
    out_raster = arcpy.sa.Idw(
        in_point_features="output_modified_XYTableToPoint_max",
        z_field="Max_Temp",
        cell_size=0.0172421199999999,
        power=2,
        search_radius="VARIABLE 12",
        in_barrier_polyline_features=None
    )
    out_raster.save(r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS␣
 ↪5571 Lab3_2\GIS 5571 Lab3_2.gdb\Idw_output_m_max")
```

[95]:

```
#Then we do an EBK interpolation for the maximum temperature dataset
arcpy.ga.EmpiricalBayesianKriging(
    in_features="output_modified_XYTableToPoint_max",
    z_field="Max_Temp",
    out_ga_layer=None,
    out_raster=r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571␣
 ↪Lab3_2\GIS 5571 Lab3_2.gdb\EBK_Interpolation_raster_max",
    cell_size=0.0172421199999999,
    transformation_type="NONE",
    max_local_points=100,
    overlap_factor=1,
    number_semivariograms=100,
    search_neighborhood="NBRTYPE=StandardCircular RADIUS=3.1834293129079␣
 ↪ANGLE=0 NBR_MAX=15 NBR_MIN=10 SECTOR_TYPE=ONE_SECTOR",
    output_type="PREDICTION",
    quantile_value=0.5,
    threshold_type="EXCEED",
    probability_threshold=None,
    semivariogram_model_type="POWER"
)
```

[95]:

```
<Result ''>
```

[96]:

```
#Finally we create an NNI interpolation for the maximum temperature dataset
with arcpy.EnvManager(scratchWorkspace=r"C:
 ↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
 ↪Lab3_2.gdb"):
    Natural_outp1_max = arcpy.sa.NaturalNeighbor(
        in_point_features="output_modified_XYTableToPoint_max",
        z_field="Max_Temp",
        cell_size=0.0172421199999999
    )
    Natural_outp1_max.save(r"C:
 ↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
 ↪Lab3_2.gdb\Natural_outp1_max")
```

[96]:

```
<Result 'GPI_Output'>
```

[99]:

```
#Once we have those, we can perform the same interpolations for the minimum␣
 ↪temperature dataset
with arcpy.EnvManager(scratchWorkspace=r"C:
 ↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
 ↪Lab3_2.gdb"):
    out_raster = arcpy.sa.Idw(
        in_point_features="output_modified_XYTableToPoint_min",
        z_field="Min_Temp",
        cell_size=0.0172421199999999,
        power=2,
        search_radius="VARIABLE 12",
        in_barrier_polyline_features=None
    )
    out_raster_min.save(r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS␣
 ↪5571 Lab3_2\GIS 5571 Lab3_2.gdb\Idw_output_min")

    arcpy.ga.EmpiricalBayesianKriging(
    in_features="output_modified_XYTableToPoint_min",
    z_field="Min_Temp",
    out_ga_layer=None,
    out_raster=r"C:\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571␣
 ↪Lab3_2\GIS 5571 Lab3_2.gdb\EBK_Interpolation_raster_min",
    cell_size=0.0172421199999999,
    transformation_type="NONE",
    max_local_points=100,
    overlap_factor=1,
    number_semivariograms=100,
    search_neighborhood="NBRTYPE=StandardCircular RADIUS=3.1834293129079␣
 ↪ANGLE=0 NBR_MAX=15 NBR_MIN=10 SECTOR_TYPE=ONE_SECTOR",
    output_type="PREDICTION",
    quantile_value=0.5,
    threshold_type="EXCEED",
    probability_threshold=None,
    semivariogram_model_type="POWER"
)


with arcpy.EnvManager(scratchWorkspace=r"C:
 ↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
 ↪Lab3_2.gdb"):
    Natural_outp1_min = arcpy.sa.NaturalNeighbor(
        in_point_features="output_modified_XYTableToPoint_min",
        z_field="Min_Temp",
        cell_size=0.0172421199999999
    )
    Natural_outp1_min.save(r"C:
↪\Users\conno\OneDrive\Documents\ArcGIS\Projects\GIS 5571 Lab3_2\GIS 5571␣
↪Lab3_2.gdb\Natural_outp1_min")
```

```
[99]: <Result 'GPI_Output'>
```