Lab Report

Title: Comparing GIS API Models and Basic Arcpy Geoprocessing
Notice: Dr. Bryan Runck
Author: Holly Leaf
Date: 10/6/21

Project Repository: https://github.com/hleaf/GIS5571/tree/main/Lab1
Google Drive Link:
Time Spent: *9.5 hours*

Abstract
The purpose of this lab is to compare and contrast the conceptual models for the APIs for three websites: MN Geospatial Commons, Google Places, and NDAWN. Additionally, Jupyter Notebooks were used to create a data pipeline to query, import, and spatially join two datasets from MN Geospatial Commons. These two datasets included a polygon layer and a point layer. They were joined using the "within" method to filter for points that were within the county polygons.

Problem Statement
This lab will compare and contrast the conceptual models for the API's: MN Geospatial Commons, Google Places, and NDAWN. It will also use Jupyter notebooks to create scripts which can download data sets and spatially join them using an API.

Table 1. Data used in part 2 of the lab

#	Requirement	Defined As	(Spatial) Data	Attribute Data	Dataset	Preparation
1	County and CTU boundaries	Counties and Cities & Townships 2020, Twin Cities Metropolitan Area	Polygon boundaries		Mn GeoSpatial Commons	
2	MPCA – What's in my Neighborhood Sites	Sites identified by the MN Pollution Control Agency	Point and Polygon data		MN GeoSpatial Commons	

Input Data
The input data used (table 2) was imported from MN GeoSpatial Commons. Dataset 1 was used for boundary polygons for the counties in the Twin Cities metro. The MPCA "What's in my Neighborhood" dataset contains point and attribute data for MPCA-identified sites of various varieties. These datasets were spatially joined (see Methods).

Table 2. Input datasets

#	Title	Purpose in Analysis	Link to Source
1	County and CTU boundaries	Metro county shapefiles for spatial join	Mn GeoSpatial Commons

2	MPCA – What's in my Neighborhood Sites	MPCA point data for spatial join	MN GeoSpatial Commons

Methods

To begin a comparison, each API was reviewed using provided documentation (MnGeo, CKAN, Google Places, NDAWN). Methods for searching datasets using each API were compared. A Google Developer account was created to generate an API key, as was an MN Geo account for API credentials. The NDAWN site was navigated to discover the URL parameter format for requesting CSV files. Chrome developer tools were used for NDAWN as well.

Arcpy was used in Jupyter Notebooks within ArcGIS Pro to create a pipeline to download datasets, check/transform the coordinate systems, spatially join two datasets, print the merged attribute fields, and save the feature class to a geodatabase. The notebook file for this pipeline was saved in the project repository.

Results

API Models

MN Geospatial Commons

MnGeo uses the CKAN API. CKAN uses JSON and follows the RESTful API style. It can be accessed to return data in JSON format using HTTP requests. These requests can be performed using a web browser, terminal (curl), or using a scripting language such as Python or Javascript. For this lab, Python was used in Jupyter Notebooks within ArcGIS to access the MnGeo database. The API can be used to search the database for datasets based on a search query. It can also be used to load a dataset so it can be imported into a Feature Dataset or Geodatabase in ArcGIS.

Google Places

The Google Places API also uses HTTP requests to access the Google Maps platform. The API allows for the following requests: Place Search, Place Details, Place Photos, Place Autocomplete, and Query Autocomplete. Returns are either in JSON or XML format. An API key is required to access the Google Places API, although there is a $300 free trial period. Google uses Place IDs to identify specific places in the Google database. A Place ID is included in the returned JSON or XML data.

NDAWN

NDAWN (North Dakota Agricultural Weather Network) is a service to provide data from 155 weather stations in North Dakota. Data is accessible via the website UI or using HTTP requests. While there is not an API documentation, HTTP request formats can be deciphered from the website, including the ability to download CSVs. In this way, station data can be accessed using a script similar to the MnGeo and Google Places APIs. Each weather station has a station ID, and HTTP requests can be formatted to access data for a particular station, date, and monitoring period (e.g., weekly).

The website has a link to export CSV files, which can be used in HTTP requests, for instance: https://ndawn.ndsu.nodak.edu/table.csv?station=98&begin_date=2021-09-28&count=1&quick_pick=&ttype=weekly&variable=wdsr. This URL will download data for the weather station ID 98 for the weekly average data for the week beginning on September 28, 2021.

Jupyter Notebooks – spatial join

The input datasets (table 1) were used to create a spatial join pipeline using Jupyter Notebooks. While I was able to use the MnGeo API to find the datasets, I had difficulty importing them as feature layers using Python. In order to continue with the lab, I opted to download the shapefiles and import them manually. More work will need to be done to do this task programmatically in the future.

After importing the datasets (table 1) into ArcGIS as feature classes, Arcpy was used to ensure they are using the same spatial reference system. A spatial join was then performed to create a feature class for the MPCA sites which were within the 7-county metro area using Arcpy:

```
arcpy.analysis.SpatialJoin('my_neighborhood_sites', 'Census2020Counties', 'MPCA_in_metro_counties',
'JOIN_ONE_TO_MANY', 'KEEP_COMMON', field_mapping = 'count', match_option="WITHIN")
```

Afterwards, Arcpy was used to print the merged attribute headers:

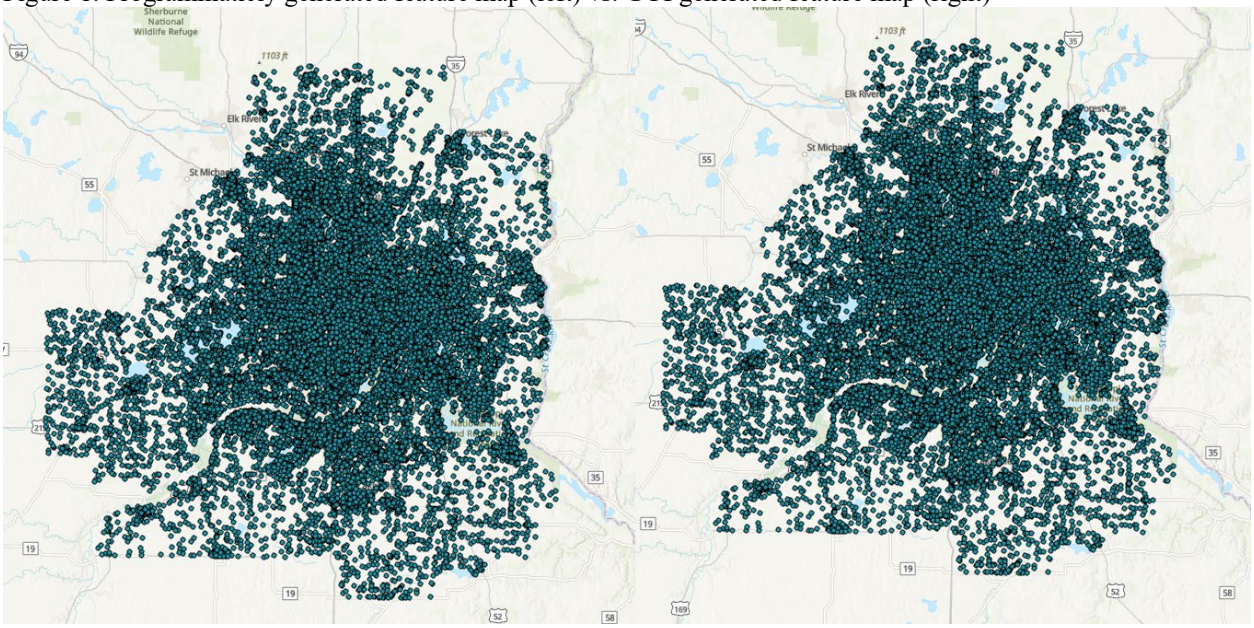
```
desc = arcpy.Describe("MPCA_in_metro_counties")
for field in desc.fields:
    print(field.name)
```

The feature was saved to a geodatabase entitled “Lab1.gdb” within the project repository.

Results Verification

Results were verified qualitatively by comparing the output merged layer map using the programmatic method to a map generated using the GUI spatial join tool (Figure 1).

Figure 1. Programmaticly generated feature map (left) vs. GUI generated feature map (right)



Discussion and Conclusion

I learned how to perform basic functions using the APIs for MnGeo, Google Places, and NDAWN. I learned how to crawl a website using Chrome Developer Tools to scrape a website for data. I learned how to use API requests on these platforms to query and import data into ArcGIS using Arcpy. MnGeo and Google Places used similar types of HTTP API calls, but NDAWN had to be done a bit more manually by analyzing the site’s URL structure. All of these API models can be used to automate GIS tasks and optimize workflow (especially compared to using GUI functionality for these sites and ArcGIS).

I learned how to use Arcpy to search and request datasets, and to perform a basic geoprocessing such as a spatial join and coordinate system conversion. I discovered that I still have work to do to perform all of these functions programmatically instead of relying on the GUI versions of these capabilities. This will be necessary to optimize workflow and automate GIS functionality in the future.

Overall, I discovered several shortcomings in my current skillset related to coding methods for GIS. Previous GIS coursework did not adequately prepare me for the level of coding I would be doing in this course, so I will need to play catch-up to not fall further behind as we continue in the semester.

References

MN Geospatial Commons. https://gisdata.mn.gov/content/?q=help/api

CKAN API documentation. https://docs.ckan.org/en/ckan-2.1.5/api.html

Google Places documentation. https://developers.google.com/maps/documentation/places/web-service/overview

NDAWN. https://ndawn.ndsu.nodak.edu/

ArcGIS Pro Python Reference. https://pro.arcgis.com/en/pro-app/latest/arcpy/main/arcgis-pro-arcpy-reference.htm

Self-score

Category	Description	Points Possible	Score
Structural Elements	All elements of a lab report are included **(2 points each)**: Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score	28	**24**
Clarity of Content	Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5 minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**.	24	**16**
Reproducibility	Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified.	28	**20**
Verification	Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**.	20	**15**
		100	**75**