Lab Report

Title: Lab 1
Notice: Dr. Bryan Runck
Author: Alexander Edstrom
Date: 10/29/21

Project Repository: *AlexanderEdstrom/GIS5571 (github.com)*
Time Spent: 11

Abstract

This lab covers the basics of setting up a data pipeline in python using multiple APIs and using those pipelines to access data from three sources, Minnesota Geospatial Commons, Google Places, and NDAWN. The bulk of this report is focused on the main problem statement of the project, dissecting the differences and similarities between the three APIs. Specifically looking at the differences in their conceptual models and workflow processes to complete the same task. The data used is sourced from the MN Geospatial Commons, Google Places, and NDAWN. The methods section outlines the workflow used for each of the three pipeline constructions. The results section compares and contrasts each method as well as verifies the results via simple output comparison. The discussion includes my personal notes on how each of the APIs was to work with and learn.

Problem Statement

For the MN Geospatial Commons API, a Notebook will be created to act as a pipeline to pull two data sets, transform them to the same coordinate reference system, spatially join them, display the head of the attribute table, and then save the combined data to a geodatabase. For each of the other APIs, a pipeline will be made to pull data off of a database/URL. These three pipeline construction processes will be compared and contrasted to identify similarities and differences. Table 1 outlines these requirements.

Table 1. Requirements

#	Requirement	Defined As	(Spatial) Data	Attribute Data	Dataset	Preparation
1	“Data Sets”	Raw input data from each of the APIs databases’	Varies (Expanded in Data Section)	Varies		Pulled from a URL using each API

Input Data

The data used in this project varies based on the API used. For the CKAN API, the data comes from the MN Geospatial Commons. One dataset contains the road centerlines for the bus routes within the Minneapolis/St. Paul metro area. The other data set contains the trail centerlines for each of the MN State Trails. The data for the Google Places API comes from Google Maps, simpling pulling info about a particular restaurant. The land dataset comes from NDAWN, simple average temperature data from a given time frame at a given location. Table 2 outlines this data.

Table 2. Input Data Table

#	Title	Purpose in Analysis	Link to Source

1	Metro Area Bus Routes	Raw input dataset from The Metropolitan Council	Mn Geospatial Commons
2	State Trails of Minnesota	Raw input dataset from The MN DNR	Mn Geospatial Commons
3	Average Weekly Temp In Ayr ND	Raw input dataset from the NDAWN station in Ayr	NDAWN
4	Restaurant Data	Query result data from Google Places, info about a restaurant near a given location	Google Places

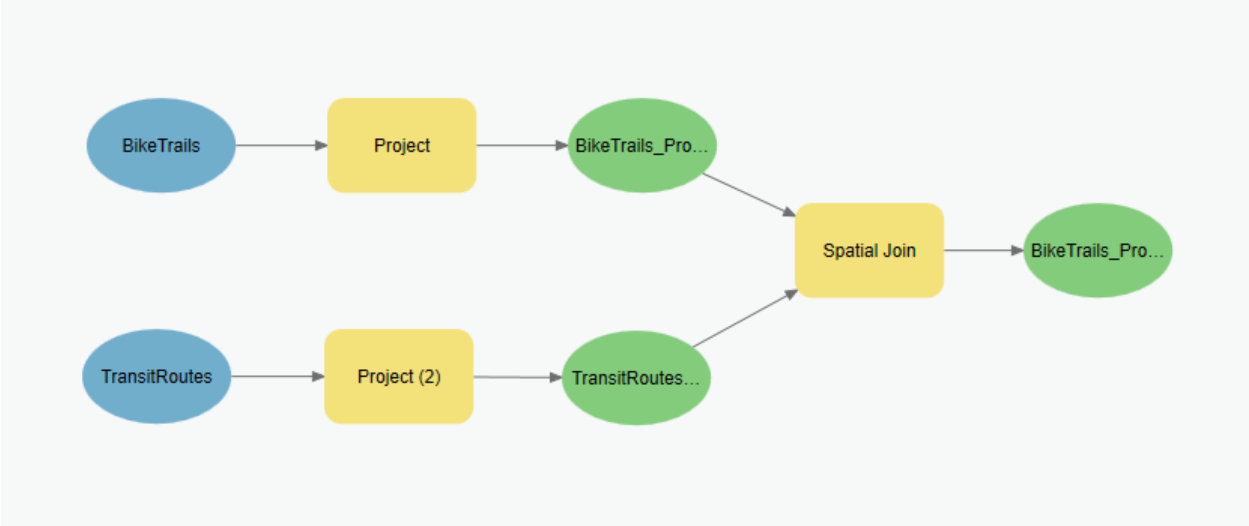
Methods

The process of getting the data imported was the largest portion of this lab. Each API method will be addressed as well as compared and contrasted with the others within the results, discussion and conclusion section below.

MN Geospatial Commons API (CKAN):

The CKAN API was the least straightforward of the group. Although the code required to make the data request is simple on paper, there was a lot more involved in finding the correct URL to pull the data that you are looking for. I ended up setting the packages, groups, and keys to allow me to output the list of URLs for a given keyword used as a query. After finding my shapefile URL using the key and print function, I was able to download my data and save it into my geodatabase after using the "zipfile" library to uncompress the file. The rest of the process was straightforward, as it only included the use of arcpy functions and ArcPro's toolset. This data flow is shown in figure 1. To reproject each of the datasets into the same coordinate system (NAD83 UTM Zone 15N), I used the "Project" tool to do this. The final step was to spatially join the two datasets with the "SpatialJoin" tool, allowing for the attribute table head to be printed, displaying each of the newly joined fields.

Figure 1. Data flow diagram.



Google Places API:

For the Google Places API, the process only included the construction of a pipeline to request data. The Google Places API required the creation of an account that gives you access to all of the credentials needed to receive an API key. With that key you can make data pulls just like with CKAN, the only difference was the way in which you had to access the data and store it. With the Google Places API, I was able to specify certain coordinates and search the surrounding area for restaurants that matched a search query. Once it found the nearest restaurant matching the

query, I had it return the name and address of the restaurant. The following was my URL for the API, the underlined section highlights the portion that returns only the name and address.

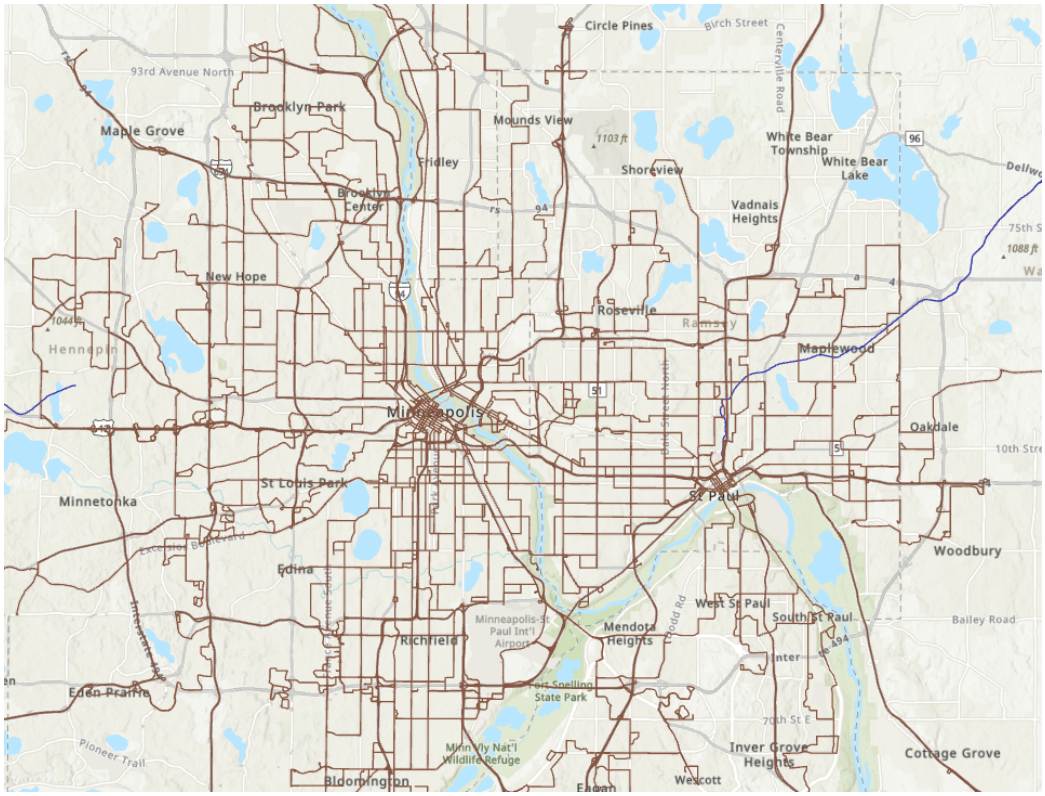
https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Olive&inputtype=textquery&locationbias=circle%3A2000%44.972083%2C-93.424556&fields=formatted_address%2Cname&key=AIzaSyC6YW1w_APjGv0Tf2dAWcKVq5DCcp-ADqg

NDAWN API:
The NDAWN API was the more barebones and simple API of the group. I ended up using the exact same template that I used for the Google Places API, only switching the URL. I used the NDAWN database to pull the average temperate over a given week from a given station in Ayn ND.

Results

Figure 2 shows the output of the MN GeoSpatial Commons API data request, project, and spatial join, I simply added the final data layer to a map to more easily visualize the dataset. While each of the APIs accomplished pretty much the same goal, to download data from an online data source, the execution of each was somewhat different. The Google Places API seemed to have a much smaller use case than the CKAN API, which makes since as CKAN is used all over the web. The Google Places API was highly optimized for interacting with other Google APIs and services such as Google Maps and Google Earth, I saw countless references to other Google services within the documentation. This made it much easier to use with Google data, where CKAN wasn't optimized specifically for MN GeoSpatial Commons. The MNGeoSpatial Commons application of the API seemed less streamlined than the Google counterpart. NDAWN was by far the least complex of the set, I felt like I could pull any csv or json data from any website exactly like that. There did not seem to be any specific methods or tricks for the NDAWN database in particular that wouldn't also be applicable to other similar databases. While it was easy to retrieve the data, the response seemed to have a lot of extra data that wasn't exactly the details I was looking for, so extra work would have to be done if I were to do analysis on that data like I did with the MN GeoSpatial Commons datasets.

Figure 2. Both datasets displayed on a map.



Results Verification

The MN GeoSpatial Commons API data pipeline had a visual component that allowed for easy verification. Simply looking at the data on the map allowed me to see that the data was successfully retrieved, unzipped, and imported. The spatial join was also obvious as I could see the combined fields using the final print statement. The other two APIs allowed me to see the data that I pulled without downloading it, so in both cases a simple print statement of the response in text form allowed me to see that the data I was attempting to retrieve was sucssesfully accessed.

Discussion and Conclusion

Directly working with the variety of APIs allowed me to understand how the general conceptual model of an API works. I don't think I'd have as comprehensive of an understanding having only worked with one API. Going into the work, I had some misconceptions about how acquiring an API key worked, it was easier than I thought it was going to be for the MN GeoSpatial Commons API, as we didn't need to verify our request and didn't need a key at all. The process of creating an account and getting a Google Places key was also easier than I thought, Google seemed to want people to explore their service in hopes that they are more liekley to use it in the future.

References

No external sources cited

Self-score

Fill out this rubric for yourself and include it in your lab report. The same rubric will be used to generate a grade in proportion to the points assigned in the syllabus to the assignment.

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	**28**
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	**22**
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	**28**
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	**18**
		100	**96**