Intermediate Spatial Data Science Lab Report

Title: Lab 1 - Using ETLs towards API interactions
Notice: Dr. Bryan Runck
Author: Andrew Arlt
Date: 10/08/2024

Project Repository: [andrewarlt/GIS5571/Lab1]
Google Drive Link: n/a
Time Spent: ~16 hours

Abstract

This lab activity focused on developing a functioning ETL setup that is able to interact with three different APIs. Data was downloaded through each API interaction, then transformed based on the specific data requirements. API interactions require different coding schemes to account for their individualized querying codes and systems. Some sites, like the NDAWN framework, do not have formal APIs, but use API-like querying that can be visualized in the URL. After data is transformed into usable datasets, such as shapefiles and dataframes/spatially-enabled dataframes, it can be projected into a common coordinate system before being spatially joined and exported as a geodatabase.

Problem Statement

In order to complete this project, it was necessary to determine the method for interfacing with three different APIs: 1) NDAWN, 2) CKAN (Minnesota Geospatial Commons), and 3) ArcGIS Online Rest API. Each of these APIs use slightly different methods for querying data and deriving the URL for calling the API request. The general task problem statement can be broken down into three parts:

1. Decomposing interfaces for spatial web APIs into conceptual models;
2. Compare web APIs using models and a custom extract, transform, and load (ETL) routine; and
3. Construct an ETL pipeline using a Notebook and integrate at least two datasets via spatial join.

This problem required a set of specific tasks that could establish a functional ETL routine path. Figure 1 (below) shows the breakdown of the specific tasks used to gather data from the APIs and then transform, join, and export the extracted datasets.

Figure 1. Table showing the required tasks and associated data and data information for each step of the ETL: extraction and initial transformations (blue), group transformations (yellow), and group loading actions (green).

#	Requirement	Defined As	(Spatial) Data	Attribute Data	Dataset	Preparation
1	**NDAWN API**	Download data via an AP, then transform to a common projection.	Point Data, Weather Station City Information	Lat, Long, Elevation, Year, Month, Day, Avg Max Temp, Total Solar Rad, Total Rainfall, Avg Dew Point	NDAWN to “Weather Data”	API request for CSV file; CSV remove extra rows and columns, then transform to an SEDF format
2	**CKAN API**	Download data via an AP, then transform to a common projection.	Polygon Data, Minnesota County Boundaries	Area, Perimeter, County ID, County Name, County Abbrev, County FIP, Shape Length, Shape Area	MN GeoCommons, CKAN API to “MN County”	API request for JSON, JSON queried for shapefile, shapefile projected to 4326, then transformed to an SEDF format
3	**ArcGIS Online REST API**	Download data via an AP, then transform to a common projection.	Polyline Data, Minnesota Roads	Object ID, Shape	ArcGIS Online REST API to “MN Streets”	API request for JSON, JSON to SEDF, projected to 4326, then transformed to shapefile

4	**NDAWN, MN Counties, and MN Streets**	Spatially join datasets.	Shapefiles, API Request Transformation Files	FID, Join Count, Target FID, Join Count 1, Target FID 1, ID, Area, Perimeter, Shape Area, ID 1, Station Name, Lat/Long, Elevation, Year, Month, Day, Avg Max Temp, Total Solar, Total Rain, Avg Dew Point, ID 12, Object ID, Shape	Weather Data, MN County, MN Streets to “Test Join”	Spatially join shapefiles, then transform to SEDF
5	**NDAWN, MN Counties, and MN Streets**	Spatially join datasets using spatial analysis.	Shapefiles, API Request Transformation Files	[Same as file descriptions in 1, 2, 3]	Weather Data, MN County, MN Streets to “Counties with Cities” and “Roads within <defined> Counties”	Spatially identify counties intersecting cities, output as shapefile (counties with cities) Use spatially defined shapefile above to identify roads intersecting those specific counties, output as shapefile (roads within counties)
6	**Joined Data**	Save integrated dataset to geodatabase.	SEDF, Test Join File	[Same as file description for 4]	Test Join	Define output path for geodatabase, transform SEDF as geodatabase
7	**Sorted and Joined Data**	Save integrated dataset to geodatabase.	Shapefiles, Weather Data, Counties with Cities, Roads within Counties	[Same as file descriptions in 1, 2, 3]	Join Query	Define output path for geodatabase, transform SEDF as geodatabase

Input Data

The goal output for this lab activity was to derive a data product that could produce a map with weather station locations, the counties that intersect those stations, and the roads that are located within the associated counties. To do this, NDAWN data for five Minnesota weather stations was queried for their basic station data (Elevation, Year, Month, Day, Avg Max Temp, Total Solar Rad, Total Rainfall, Avg Dew Point) as a CSV file.

Minnesota Geospatial Commons (CKAN API) was used to request a Minnesota county boundary shapefile, with basic area, shape, and official identification information of each county polygon. Minnesota rural roads data was acquired via the ArcGIS Online REST API through the MInnesota DNR to determine roads near the weather stations from the NDAWN dataset.

Figure 2. Data layer(s) used to perform the described processes.

#	Title	Purpose in Analysis	Link to Source
1	**NDAWN Station Data**	Used to establish weather station locations within MInnesota using 5 cities and their associated weather data for a subset of data values.	NDAWN
2	**MN County Boundaries**	Used to determine Minnesota county locations for each of the NDAWN weather stations used in the above dataset.	MN GeoCommons, CKAN API
3	**MN County Roads**	Used to locate rural (county) roads near the NDAWN weather stations used in the above dataset.	ArcGIS Online REST API

Methods

An ETL was established using the specific API interactions to extract the datasets described in the input data section. API interactions required different coding schemes to extract the input datasets. Below is a list of the APIs used and a comparison of the API URLs used to complete this lab activity. We can compare APIs using their request URLs, where the **Main URL API is orange, the Specific Service is purple, the Response Type is green, and the Function Query Inputs are blue**.

NDAWN API

NDAWN data is accessible through a query system using a GUI interface, but the query result in the URL for the landing page for the data download shows similarities to more traditional APIs: main URL, response type, and query inputs in a standardized format.

With more documentation, this website interaction could be more standardized and more easily used as an explicit API interaction. The formatting of the CSV dataset could be designed with a more useful datatable in mind, which does not require row modification for reading as a dataframe.

https://ndawn.ndsu.nodak.edu/table.csv?station=78&station=174&variable=hdt&variable=hdrh&variable=hdbst&variable=hdtst&variable=hdws&ttype=hourly&quick_pick=&begin_date=2024-10-06&end_date=2024-10-06

CKAN API

Minnesota Geospatial Commons uses the CKAN API format, which allows the user to request specific packages and file formats, and use common query sets for data requests. This data request did require opening a package and calling a second request for the specific file URL. We can see the API structure below, where there is an API URL, a service function, and then a query set. The green URL was the response in this request using a json search for the desired file link.

This API was relatively easy to interact with, but lacked a simple way to quickly interact with the desired dataset. If common file path nomenclature was used, then API interactions would be easier to code and request the direct files needed.

https://gisdata.mn.gov/api/3/action/package_search?q=minnesota%20counties
https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/bdry_counties_in_minnesota/shp_bdry_counties_in_minnesota.zip

ArcGIS Online REST API

The **ArcGIS Online REST API** was able to be accessed using an online query GUI from the desired data source feature layer. However, the URL structure for accessing the data from future layers with known source folders on ArcGIS Online would be relatively simple to request using the REST API URL structure. This structure contained the main API URL, service function information, querying functions, and then the response file type.

This seemed to be the API that was the most simple to interact with and then connect directly to the desired data sets. The API URL structure uses straightforward structures, but does require knowledge of the existing file within the dataset, which makes it less queryable.

https://arcgis.dnr.state.mn.us/host/rest/services/Hosted/DOT_Roads/FeatureServer/1/query?where=1%3D1&outFields=*&f=geojson

The data flow diagram shown in Figure 3 (below) shows the relatively simple set of steps used in the NDAWN API interaction and extraction compared to the CKAN API, which necessitated a dual API request to extract the actual ZIP data file. The diagram shows that the REST API interaction also required a few steps for extraction. The flow diagram includes the steps for transforming the files into dataframes and reprojecting into a common CRS 4326 coordinate reference system.

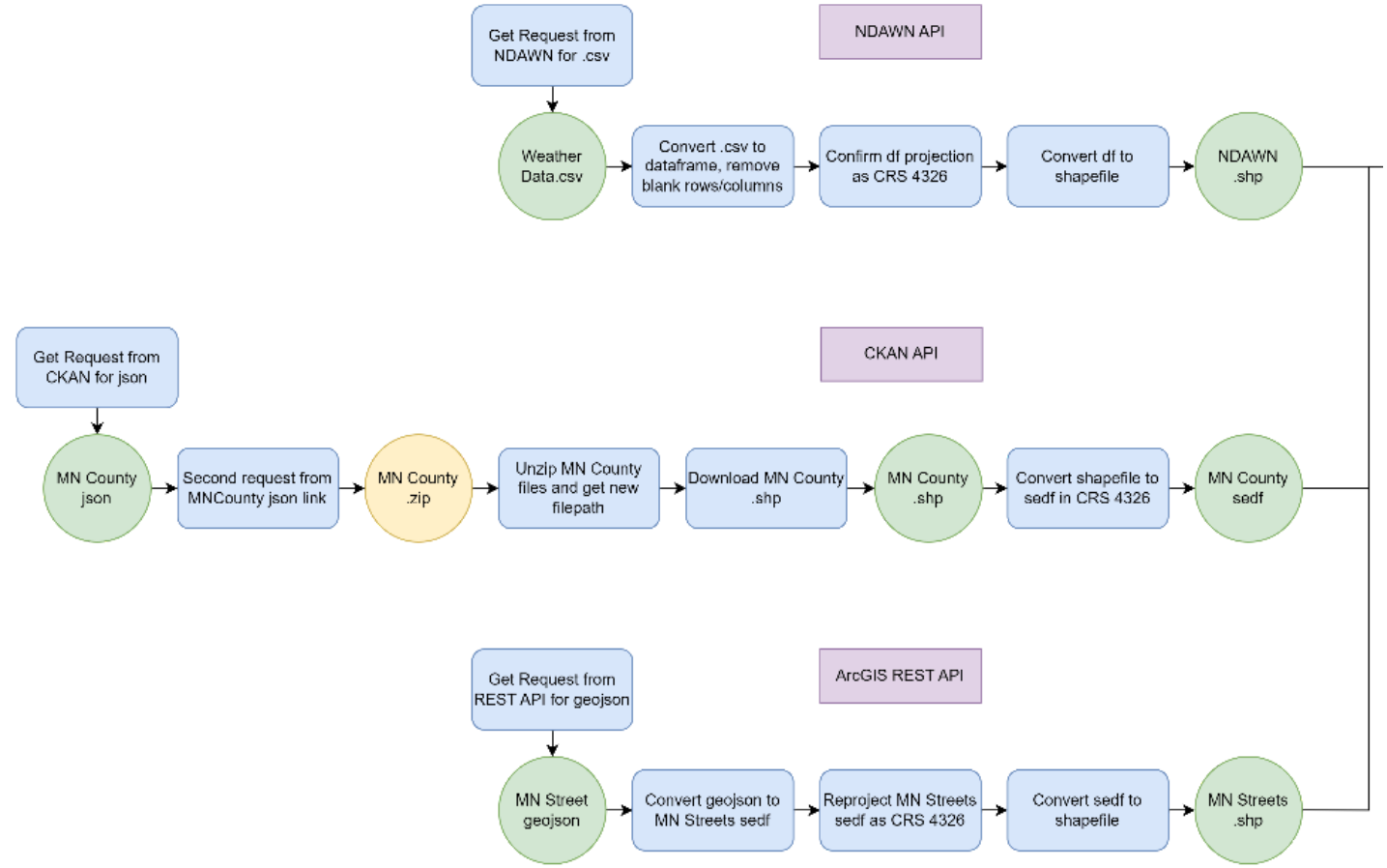

Figure 3. Data flow diagram showing the API interactions used in this ETL.

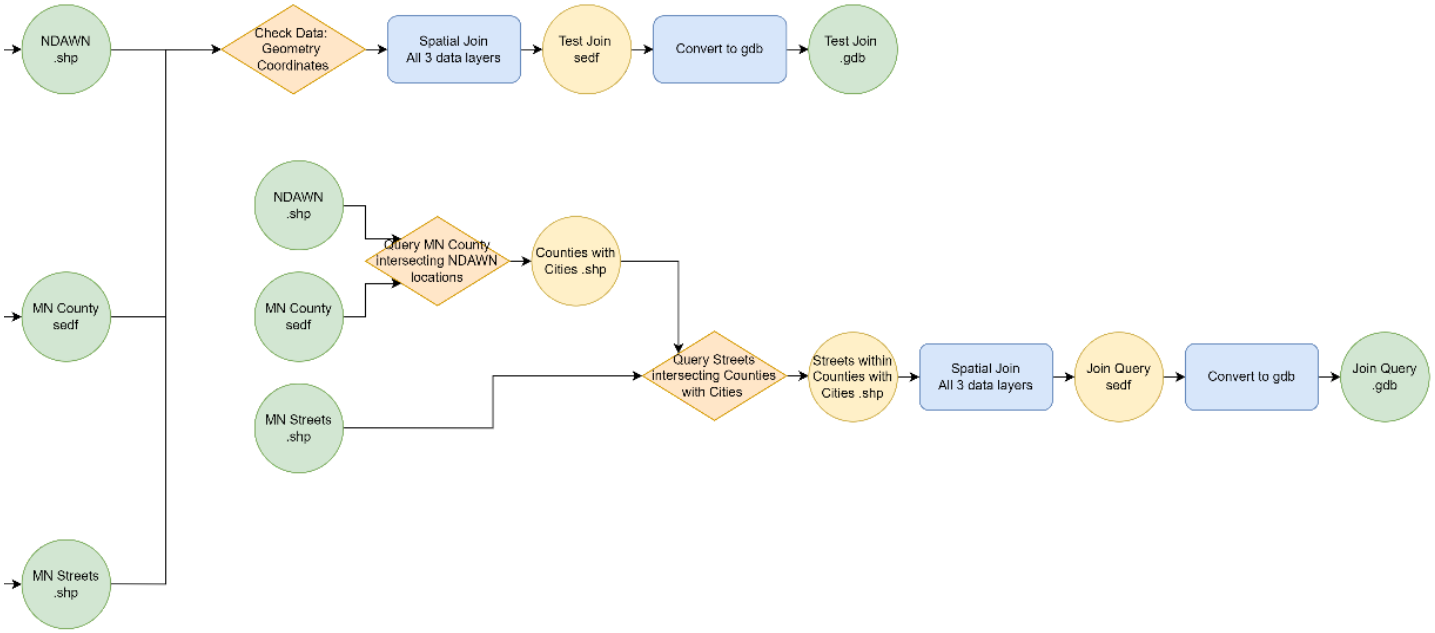

Figure 4. Data flow diagram showing the methods for transforming the data layers through spatial joins and conversion to a geodatabase.

Figure 4 (above) shows the method used for spatially joining the datasets used in the transformation and loading portion of the ETL. Because the datasets were projected during the initial extraction steps, the spatially enabled dataframes (SEDFs) could simply be joined using Pandas spatial join features functions. This spatially joined file could be transformed into a geodatabase using arcpy functions.

A second spatial join action set was completed using spatial analysis features to define the data files in relation to the NDAWN weather station cities. Figure 4 (above) shows that a query was used to define the counties in the original Minnesota county dataset that contained (intersected) with the NDAWN station locations. The counties with NDAWN stations were defined as “counties with cities” and used for a second spatial intersection query using the Minnesota roads data, which became “Streets within Counties”. These files were converted to SEDFs and joined in a method similar to the first spatial join described, and then also exported as a geodatabase.

Results

The ETL structure resulted in requesting three unique datasets and then transforming them into usable formats. The ETL was able to convert files into dataframes that could be modified into similar structures, remove excess or missing data, and reproject them into common coordinate systems. Files were modified using spatial joins and analysis and then exported to geodatabases. These files were able to be visualized on a map projection for verification, where only counties and roads that existed within a geographic intersection relating to the NDAWN stations would show up. Figure 5, below, shows the map produced by the final files created using the described ETL.

Results Verification

The map below shows that the data files are properly projected (in a common coordinate system) and that the spatial defined features were correctly queried and exported.

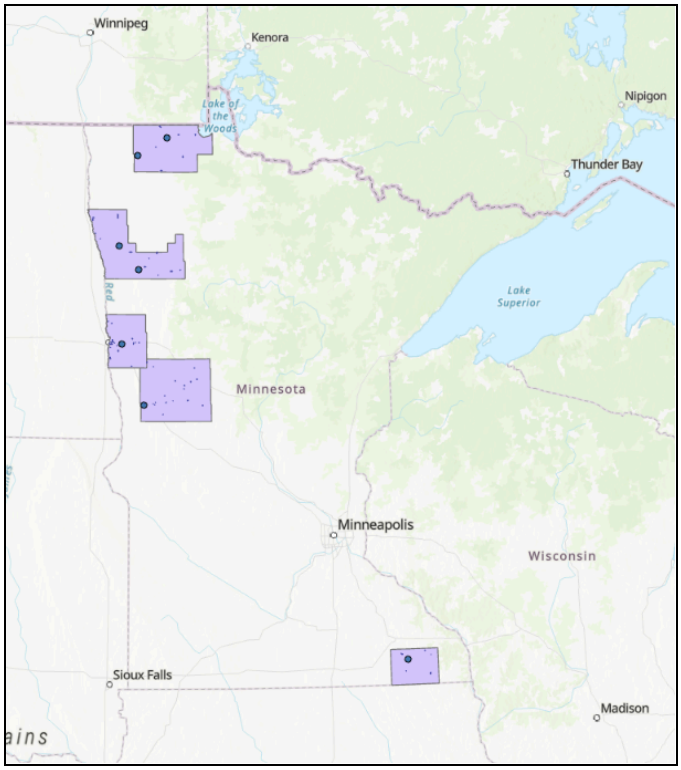

Figure 5. Map of the spatially joined features using spatial analysis to determine the intersecting features between the 3 data layers.

Discussion and Conclusion

The lab activity involved learning about the structure and role of APIs. Data requesting using coding is quite different from directly downloading datasets from a source website, after searching within that site. The benefit of calling data directly from a site seems obvious, since often there can be many files that are needed for a specific area, or data that is frequently updated or has unique versions of data over time. API interactions seem like a convenient way to efficiently obtain these data without having to spend a lot of time downloading individual datasets or files.

The idea of utilizing ETL frameworks for workflows is also nice, since the data is already being stored in local or common file path for the extraction (API) portion of the workflow. This means that the data can easily be called and modified for each of the files that need to be transformed. This would make working with remote sensing data a lot easier, since these data often need transformations and processing before analysis.

Some of the challenges in using this framework include which of the notebook setups to use, and which functions are available for different packages. For example, I worked within the ArcGIS Pro desktop notebooks, which meant that data could be easily visualized to a map when files were created. But this also meant that geopandas was not able to be utilized for spatial joins, making the spatial join functions a bit harder to code and complete successfully.

Difference in file structures also posed a challenge and required trial and error to determine which functions produced necessary outcomes. This occurred when trying to complete the spatial analysis transformation, where the match option could be defined as “Intersect” or “Within” for the particular query. Only the “Intersect” option seemed to produce the desired query function for the given datasets.

References

Build powerful apps with ArcGIS services. (n.d.). Esri Developer REST APIs Documentation. Retrieved October 8, 2024, from https://developers.arcgis.com/rest/

Introduction to the Spatially Enabled DataFrame. (n.d.). ArcGIS API for Python. Retrieved October 8, 2024, from https://developers.arcgis.com/python/latest/guide/introduction-to-the-spatially-enabled-dataframe/

Spatial joins | ArcGIS GeoAnalytics Engine | Esri Developer. (n.d.). ArcGIS GeoAnalytics Engine. Retrieved October 8, 2024, from https://developers.arcgis.com/geoanalytics/core-concepts/spatial-joins/

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	**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	**24**
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	**20**
		100	**100**