Working with Geospatial Data

Objectives

1. Understand the principles behind using geospatial (location) data as part of a broader analysis
2. Understand the similarities and differences between a geopandas dataframe and a pandas dataframe
3. Understand what a coordinate reference system (CRS) is
4. Be able to effectively perform spatial joins
5. Get excited about using Folium to make very cool maps

Why do a location analysis?

In 1854, London experienced a terrible cholera epidemic in the Soho district, with over 600 deaths. At the time, the leading theory for the cause of cholera was miasma (bad air).

Enter in physician John Snow, a skeptic of the miasma theory, who decided to take a more methodical approach to understanding the outbreak.

Why do a location analysis?

- Snow create a dot map to show the locations of cholera deaths, and ultimately identified the Broad Street water pump as the source of infection.
- Snow was able to convince authorities to shut down the pump, which led to the end of the epidemic.
- This work is cited as the start of epidemiology and also the start of using location plots to develop insight about spatial relationships.

NASHVILLE SOFTWARE SCHOOL

Geospatial Data Analysis Use Cases

- Marketing and Sales
  - Demographics
  - Customer segmentation
  - Site selection
- Transportation and Logistics
  - Route optimization
- Sociological
  - Crime tracking
  - Urban planning
- Epidemiology
  - Disease risk factors
  - Outbreak patterns
- Environmental
  - Risk assessment
  - Weather patterns

Geospatial Analysis in Python

Windows users: open Anaconda Navigator then launch Powershell Prompt

Mac users: launch a Terminal window

Switch to your geospatial environment:

```
conda activate geospatial
```

Launch Jupyter Notebook:

```
jupyter notebook
```

Leave Powershell Prompt/Terminal running in the background

GeoDataFrames

GeoPandas is the primary library we will use for geospatial analysis. This library introduces a new data structure: GeoDataFrames.

GeoDataFrames are very similar to pandas DataFrames, inheriting many of the same methods and attributes. There are two additional requirements for GeoDataFrames:

1. They must have a geometry column
2. They must have a CRS (coordinate reference system) attribute

There are some additional useful methods and attributes to know about for GeoDataFrames:

- `.area()`
- `.centroid`
- `.distance()`

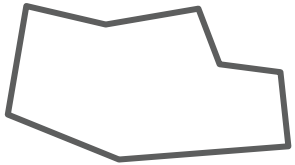
Geospatial Geometries

There are 3 basic types of geometries to know:

1. Point - a single longitude/latitude pair

2. Lines - 2 or more longitude/latitude pairs joined into a contiguous segment

3. Polygons - 3 or more longitude/latitude pairs joined to create an enclosed region



Coordinate Reference Systems

A CRS is a coordinate-based system that can be local, regional, or global and is used to locate geographical entities.

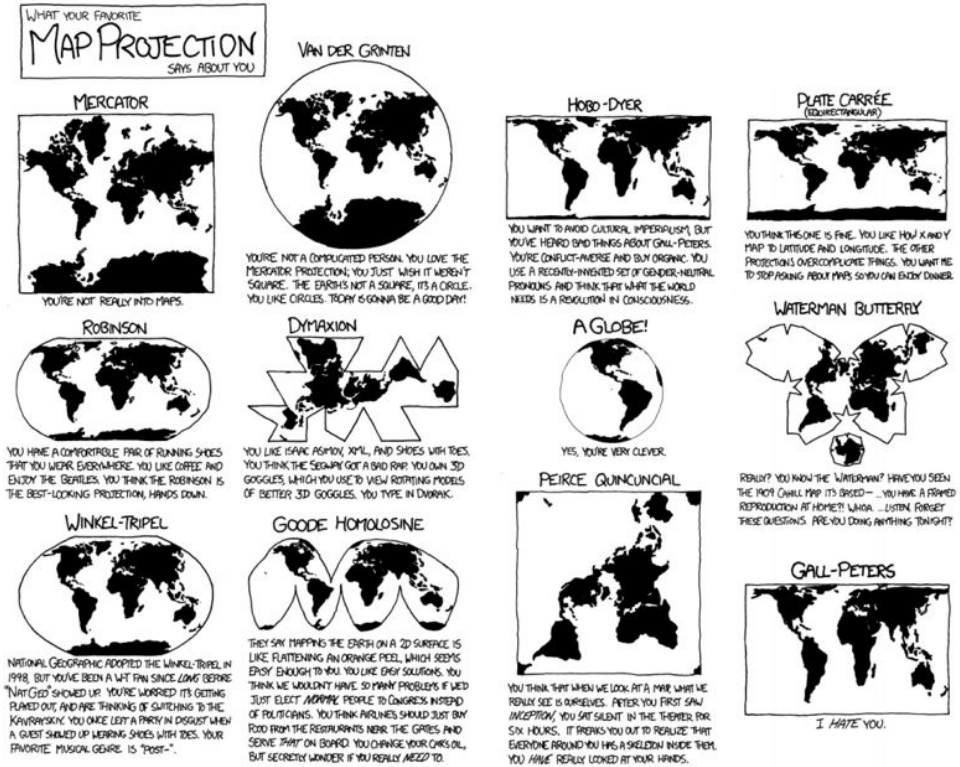
Essentially, the CRS is what tells us how a given set of coordinates in a geometry column relate to actual places on the earth.

A given CRS will be associated with a specific map projection. Different projections handle the challenges of creating a 2-dimensional map of a 3-dimensional earth in different ways.

If you find yourself needing to choose a CRS, consider 2 main questions:

- Which projection?
- Which units of measure (degrees, meters, etc.)?

Map Projections (click image for a large list)



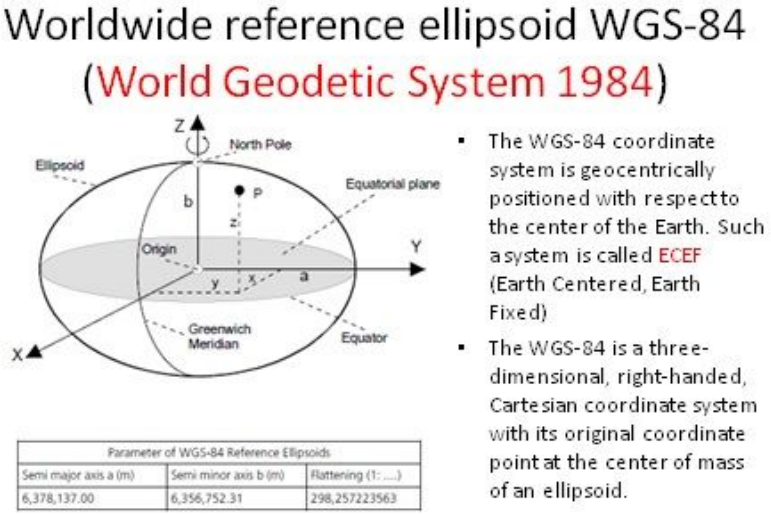
*What's that? You think I don't like the Peters map because I'm uncomfortable with having my cultural assumptions challenged? Are you sure you're not... ::puts on sunglasses:: ... **projecting**?* http://xkcd.com/977/ - http://bit.ly/explainxkcd-977

NASHVILLE SOFTWARE SCHOOL

Projection/CRS - Common Example

One common projection is the Web Mercator Projection aka WGS 84

- A variant of the Mercator projection
- Developed in 1984 by the World Geodetic Society
- Rose to prominence when Google Maps adopted it in 2005
- Now used by all major online map providers
- Used in 2 major CRSs
  - EPSG:3857 used by Google Maps
  - EPSG:4326 used by Google Earth



Working with Geospatial Data - Reading in a File

Geojson is one common format for geospatial data. Here is the result of

- reading in a geojson file of Nashville neighborhoods using the geopandas read_file() method,
- printing the crs, and
- looking at the first 5 rows with the .head() method.

In [4]:
```
import geopandas as gpd

neighborhoods = gpd.read_file('neighborhoods.geojson')
print(neighborhoods.crs)
neighborhoods.head()
```

```
epsg:4326
```

Out[4]:

	name	geometry
0	Historic Buena Vista	MULTIPOLYGON (((-86.79511 36.17576, -86.79403 ...
1	Charlotte Park	MULTIPOLYGON (((-86.87460 36.15758, -86.87317 ...
2	Hillwood	MULTIPOLYGON (((-86.87614 36.13554, -86.87583 ...
3	West Meade	MULTIPOLYGON (((-86.90384 36.12554, -86.90328 ...
4	White Bridge	MULTIPOLYGON (((-86.86321 36.12886, -86.86321 ...

Working with Geospatial Data - Spatial Joins

Spatial joins of 2 GeoDataFrames allow you to do things like:

- Find points that fall within polygons
- Find polygons that overlap each other
- Find line segments that intersect each other
- And much more!

Think back to when doing joins in SQL. Spatial joins are very similar, but rather than working with 2 tables from a database, we are working with 2 GeoDataFrames we have read into a Jupyter notebook, each of which contain information about geometric objects (points, lines, or polygons).

It's important to think about which type of geometric object each GeoDataFrame contains as you set up the code for your spatial join. Let's take a look at the syntax.

Working with Geospatial Data - Spatial Joins

This is the syntax for performing a spatial join with GeoPandas:

```
gpd.sjoin(leftgdf_name, rightgdf_name, how = 'join_type', predicate = 'predicate_type')
```

- Join type options:
  - `'inner'` (the default)
  - `'left'`
  - `'right'`
- There are many predicate options, but a few common ones include:
  - `'contains'`
  - `'intersects'`
  - `'within'`

You will read the spatial join code much as you would a sentence, for example:

```
gpd.sjoin(schools_gdf, school_districts_gdf, predicate = 'within')
```

says find the **schools** ***within*** the **school districts**.

NASHVILLE SOFTWARE SCHOOL

Working with Geospatial Data - Spatial Joins

Here's another example:

We want to spatially joins a GeoDataFrame containing **polygons** of Nashville neighborhoods (called neighborhoods) and another containing **points** of Nashville bus stops (bus_stops). We want to identify which bus stops are located in each neighborhood, and we only want to keep bus stops located in neighborhoods and neighborhoods that have at least one bus stop.

There are two equally good ways we could set up this spatial join:

```
gpd.sjoin(neighborhoods, bus_stops, predicate = 'contains')
```
Think "Neighborhoods that **contain** bus stops"

```
gpd.sjoin(bus_stops, neighborhoods, predicate = 'within')
```
Think "Bus stops that are **within** neighborhoods"

We would choose one over the other based on which geometry column we want to keep (**the left GeoDataFrame's geometry is the one that will be kept**).

Adding context with Folium

Folium is a Python package built on the leaflet javascript library.

It gives you a ton of options for making really amazing map-based visuals. You can add features like:

- Interactive street maps with markers and marker clusters
- The ability to easily customize maker popups
- A way to create special maps called choropleths
- A built in way to save your interactive maps as HTML
    - Pro Tip: Google Sites are a quick and easy way to share your maps publicly

We'll go through a sample notebook to learn more about Folium and some of its major features.

Additional Resources

- Geometric Objects:
  https://automating-gis-processes.github.io/2016/Lesson1-Geometric-Objects.html
  https://autogis-site.readthedocs.io/en/latest/lessons/lesson-1/geometry-objects.html
- Understanding projection and CRSs:
  https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/intro-to-coordinate-reference-systems/
  https://docs.qgis.org/3.22/en/docs/gentle_gis_introduction/coordinate_reference_systems.html
- CRS in GeoPandas:
  https://geopandas.org/en/stable/docs/user_guide/projections.html
  https://www.practicaldatascience.org/html/gis_crs_geopandas.html
- GeoPandas spatial joins:
  https://geopandas.org/en/stable/docs/reference/api/geopandas.sjoin.html
- GIS StackExchange: https://gis.stackexchange.com/
- Folium docs: https://python-visualization.github.io/folium/modules.html

Geospatial Individual Project

1. Go to data.nashville.gov and in the search bar, type GIS to search for geospatial datasets.
2. Create a new Jupyter notebook, and use the example notebook as a guide to explore the new geospatial dataset.
3. Play with spatial joins using one dataset that contains polygons and another that has points.
4. Explore Folium and/or choropleth maps if you want to!