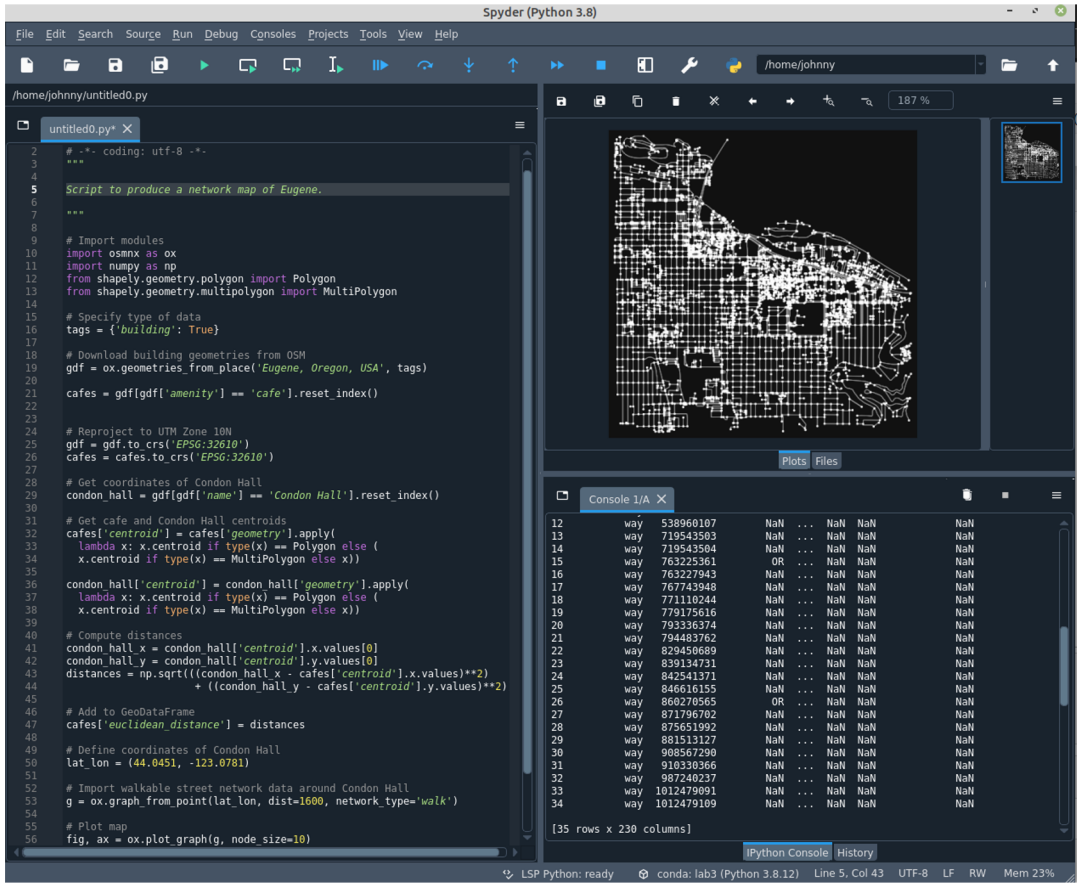
Geospatial Data Science Applications: GEOG 4/590

Lecture 2: Vector data



Johnny Ryan
jryan4@uoregon.edu
Office hours: Monday 15:00-17:00

Some good questions…

- How do we format the Canvas pdf submissions?
  - Right-click → Print… → Destination = Save as PDF
- Do we make a new notebook for our submission?
  - Yes
- Anything else?

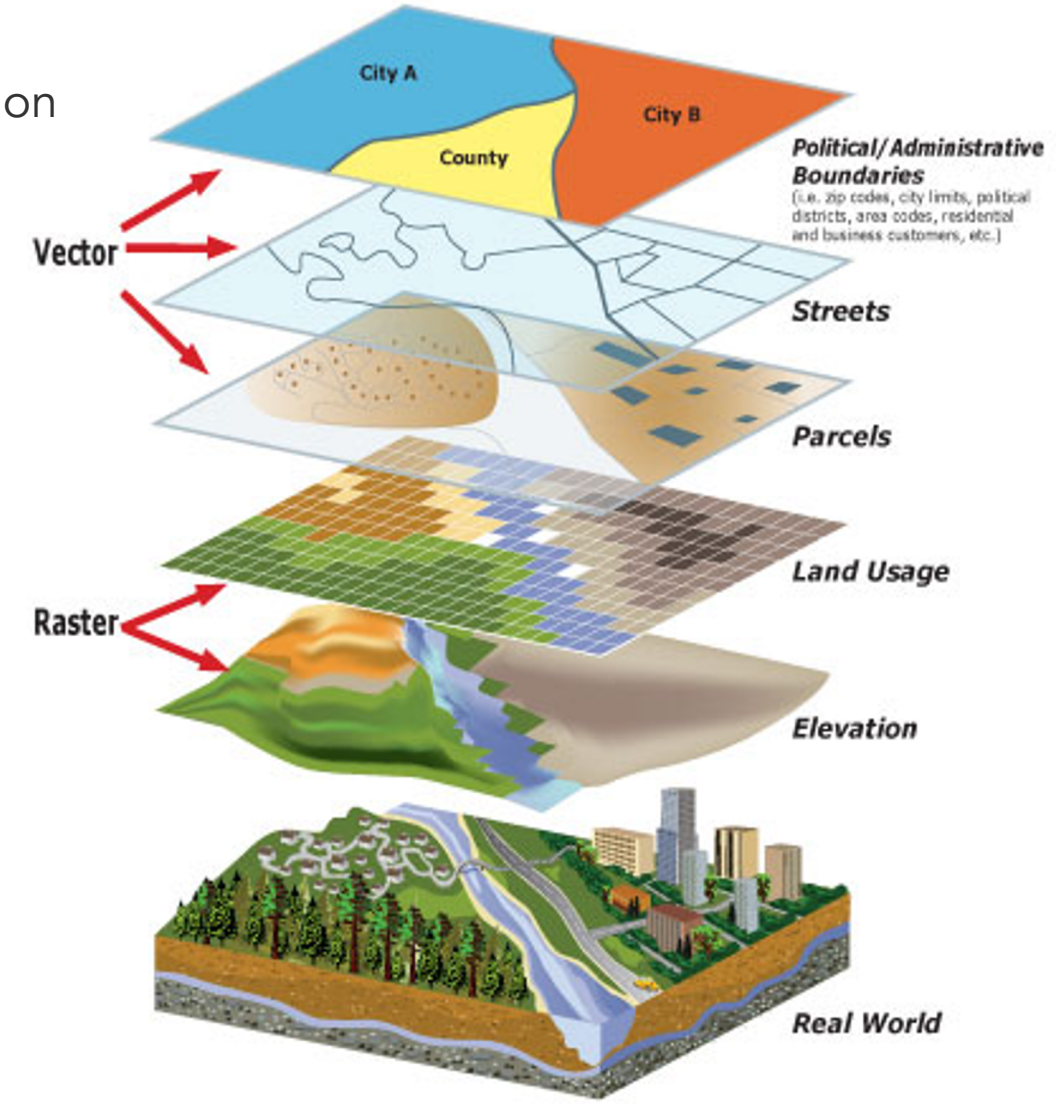
Overview

- Review basic characteristics of vector data
  - Object types and extent
  - Coordinate systems
  - Attributes
  - Formats
- Review common types of vector operations
- Introduce GeoPandas package
- This week's lab

Vector Geospatial Data

- "Vector" is a term commonly used in Geographical Information Systems (GIS) to refer to **discrete geometric entities**

- Vector data has several attributes:
  - Object type (e.g. point, line or polygon)
  - Extent
  - Coordinate reference system
  - Other attributes

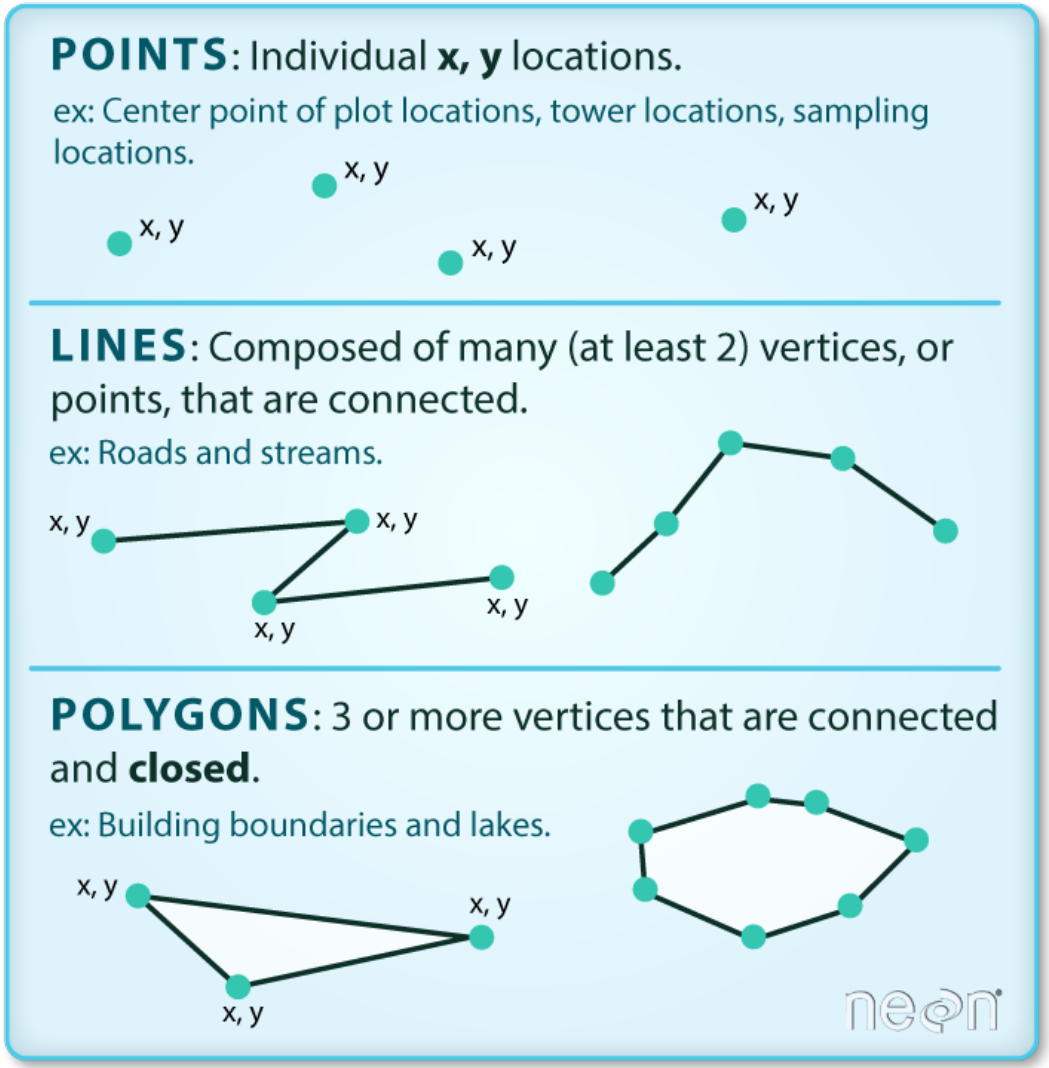


Object type and extent

Single entity:
- Point
- Line (formally known as a LineString)
- Polygon

Homogeneous entity collections:
- Multi-Point
- Multi-Line (MultiLineString)
- Multi-Polygon

Spatial extent of the vector data is represented by discrete geometric locations (x, y values)

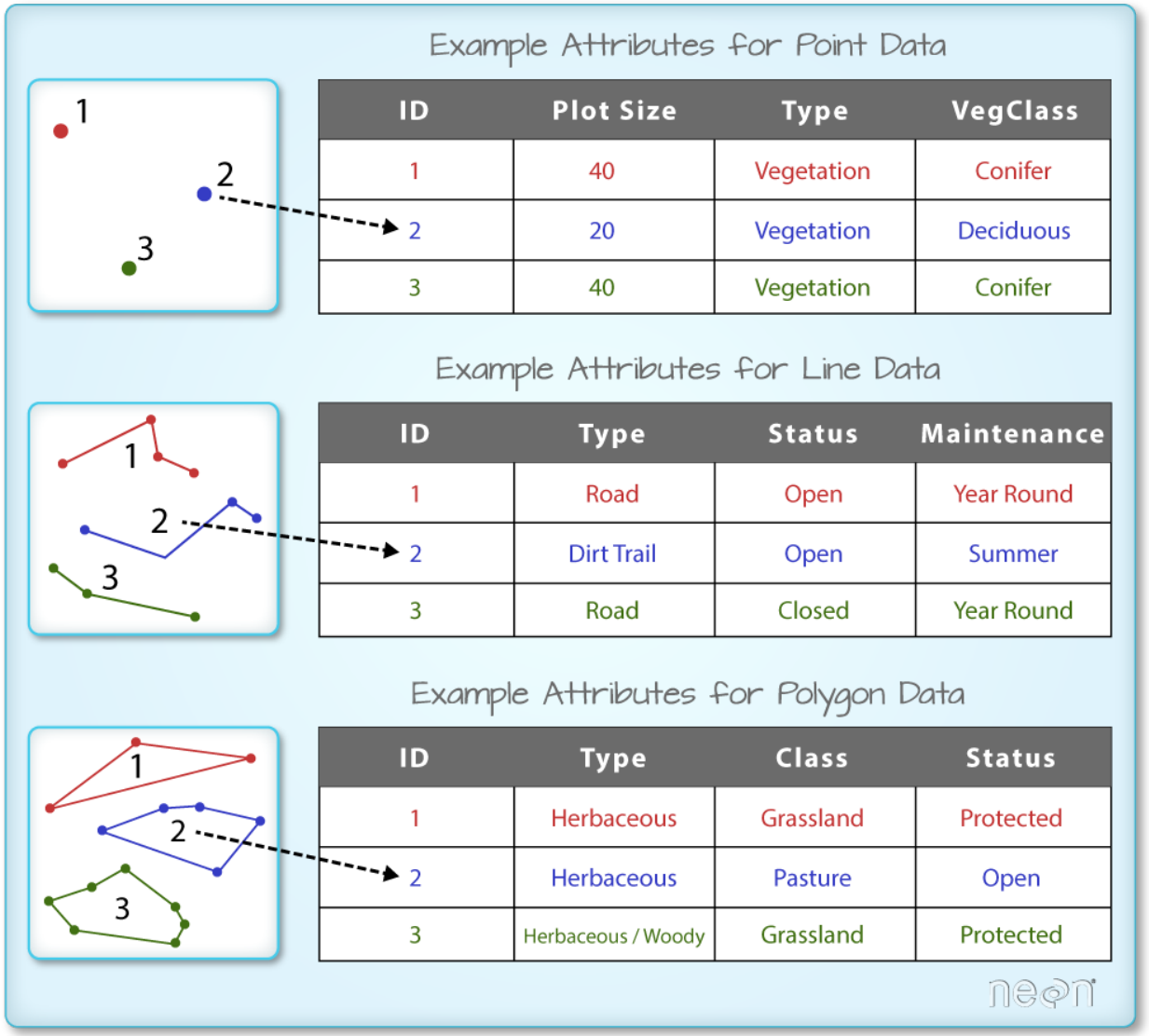


Coordinate reference system (CRS)

- CRS information connects data to the Earth's surface using a mathematical model
- A data structure cannot be considered geospatial unless it is accompanied by CRS information

Other attributes

- Vector attributes are like columns in a spreadsheet.
- For example, a line object that contains the locations of streams, might contain the name of each stream.



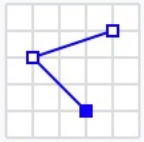
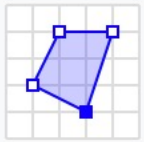
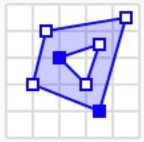
ID	Plot Size	Type	VegClass
1	40	Vegetation	Conifer
2	20	Vegetation	Deciduous
3	40	Vegetation	Conifer

ID	Type	Status	Maintenance
1	Road	Open	Year Round
2	Dirt Trail	Open	Summer
3	Road	Closed	Year Round

ID	Type	Class	Status
1	Herbaceous	Grassland	Protected
2	Herbaceous	Pasture	Open
3	Herbaceous / Woody	Grassland	Protected

Formats: Well-Known Text (WKT)

- A text markup language for representing vector geometry objects

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Formats: GeoJSON

- An open standard format designed for representing simple geographical features **and** their non-spatial attributes.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Formats: Shapefile

- A digital vector storage format for storing geographic location and associated attribute information developed by ESRI

Three mandatory files:

- **.shp:** contains the geometry for all features
- .**shx:** indexes the geometry
- **.dbf:** is a standard database file used to store attribute data and object IDs.

These files need to have the **same name** and to be stored in the **same directory** (folder) to open with Python.

Formats: Shapefile

- A digital vector storage format for storing geographic location and associated attribute information developed by ESRI

Three mandatory files:

- **.shp:** contains the geometry for all features
- .**shx:** indexes the geometry
- **.dbf:** is a standard database file used to store attribute data and object IDs.

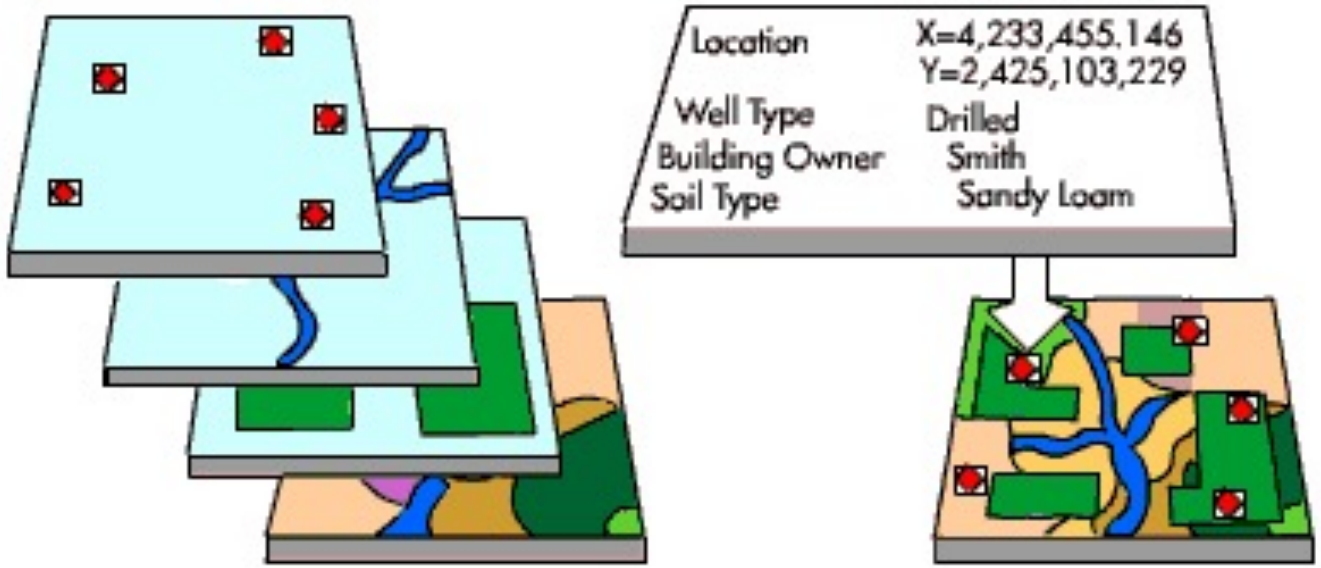
These files need to have the **same name** and to be stored in the **same directory** (folder) to open with Python.

Sometimes, a shapefile will have other associated files including:

- **.prj:** contains information on projection format including the coordinate system and projection information
- **.sbn** and **.sbx**: the files that are a spatial index of the features
- **.xml:** contains the metadata associated with the shapefile
- **.cpg:** optional plain text files that describes the encoding applied to create the Shapefile.

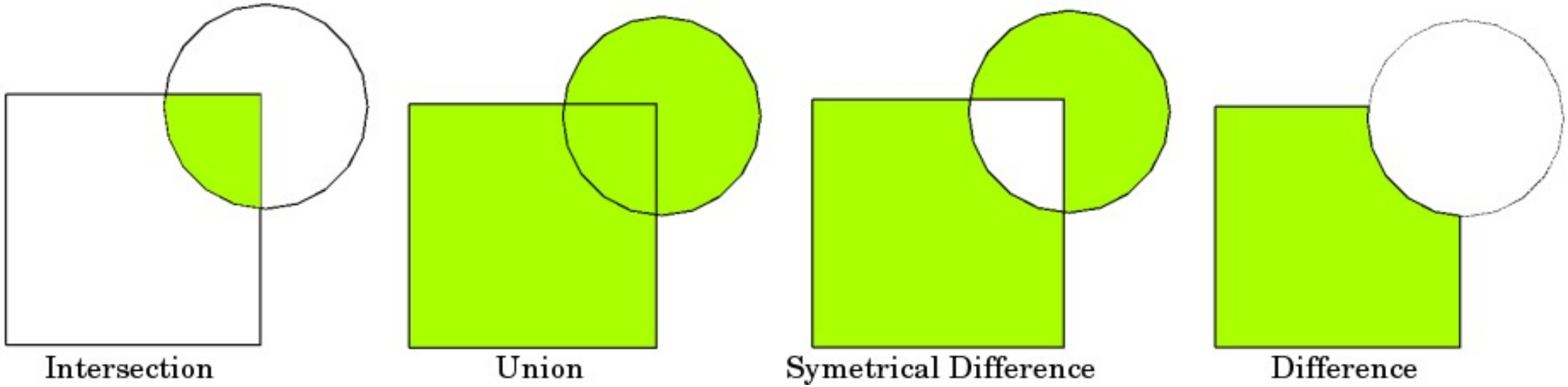
Relationships

- The vector data spatial model is accompanied by a group of natural language relationships between geometric objects – contains, intersects, overlaps, touches, etc. –
- These are termed **binary predicates** (e.g. **covers, crosses, touches, within, intersects**)
- Returns either **True** or **False**



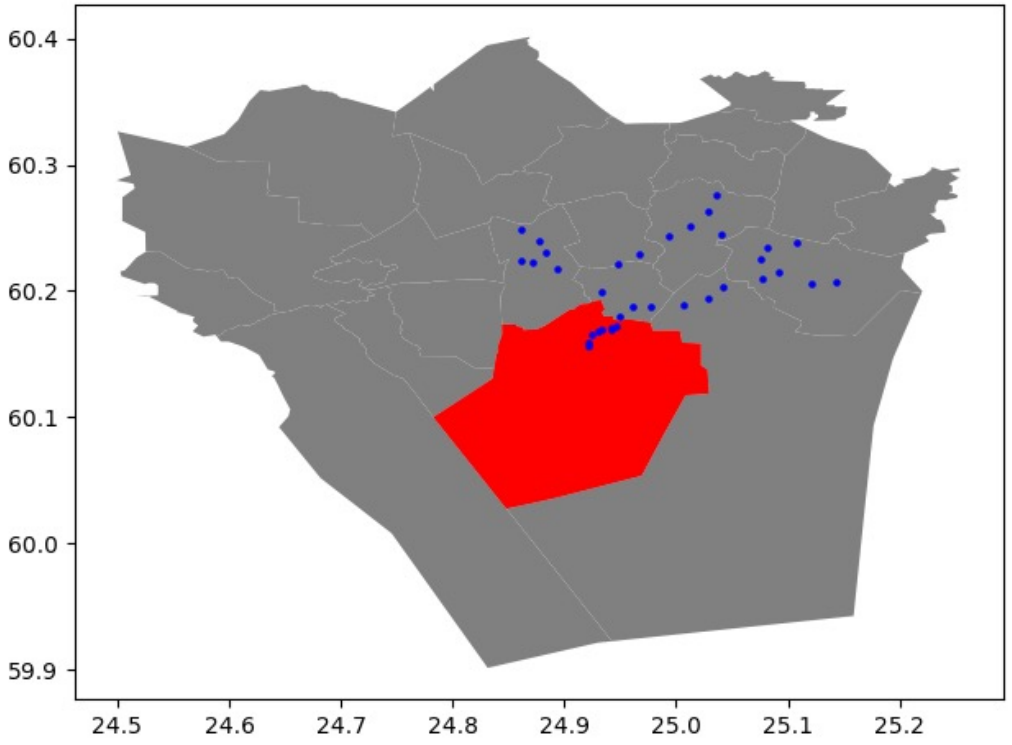
Operations

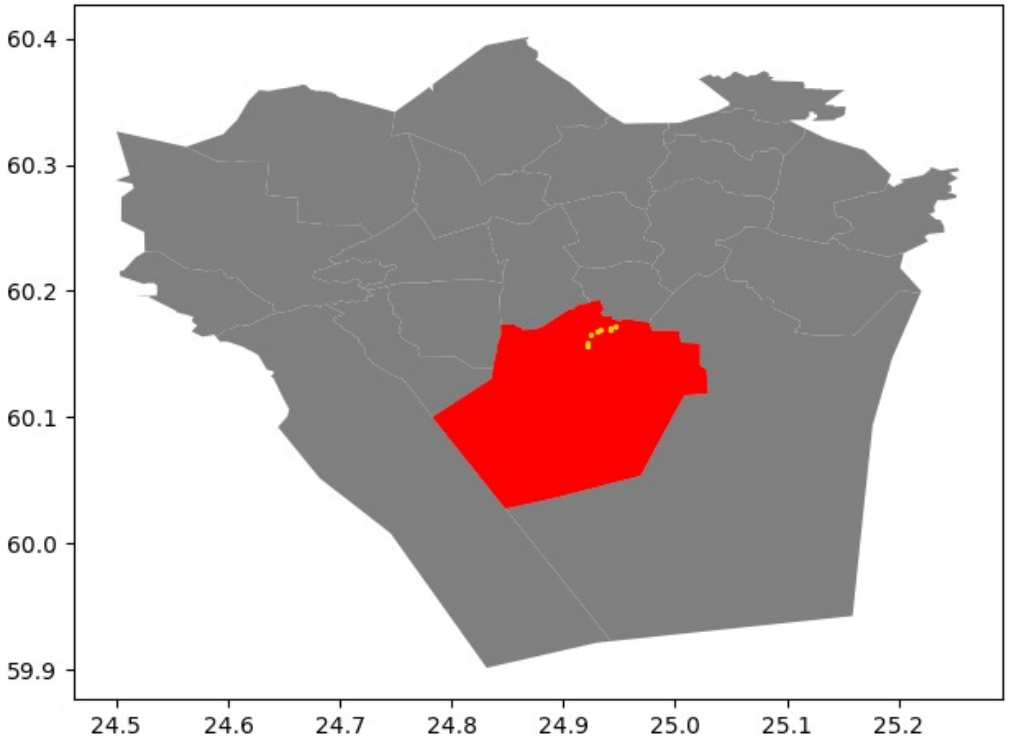
- As well as boolean attributes and methods, we can use spatial analysis methods to combine two or more vector datasets based on the spatial relationship between their geometries.



Spatial joins

- Many uses in geospatial analysis (e.g. points within polygon)

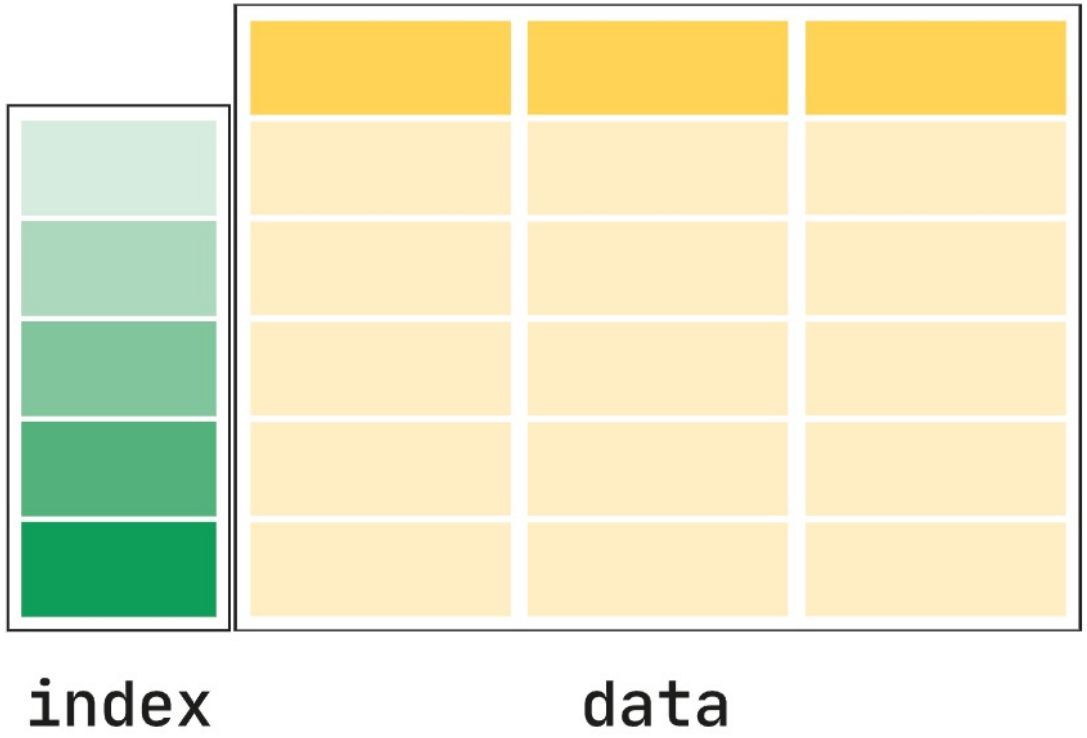




GeoPandas

- Extends **pandas** to allow spatial operations on geometric types.

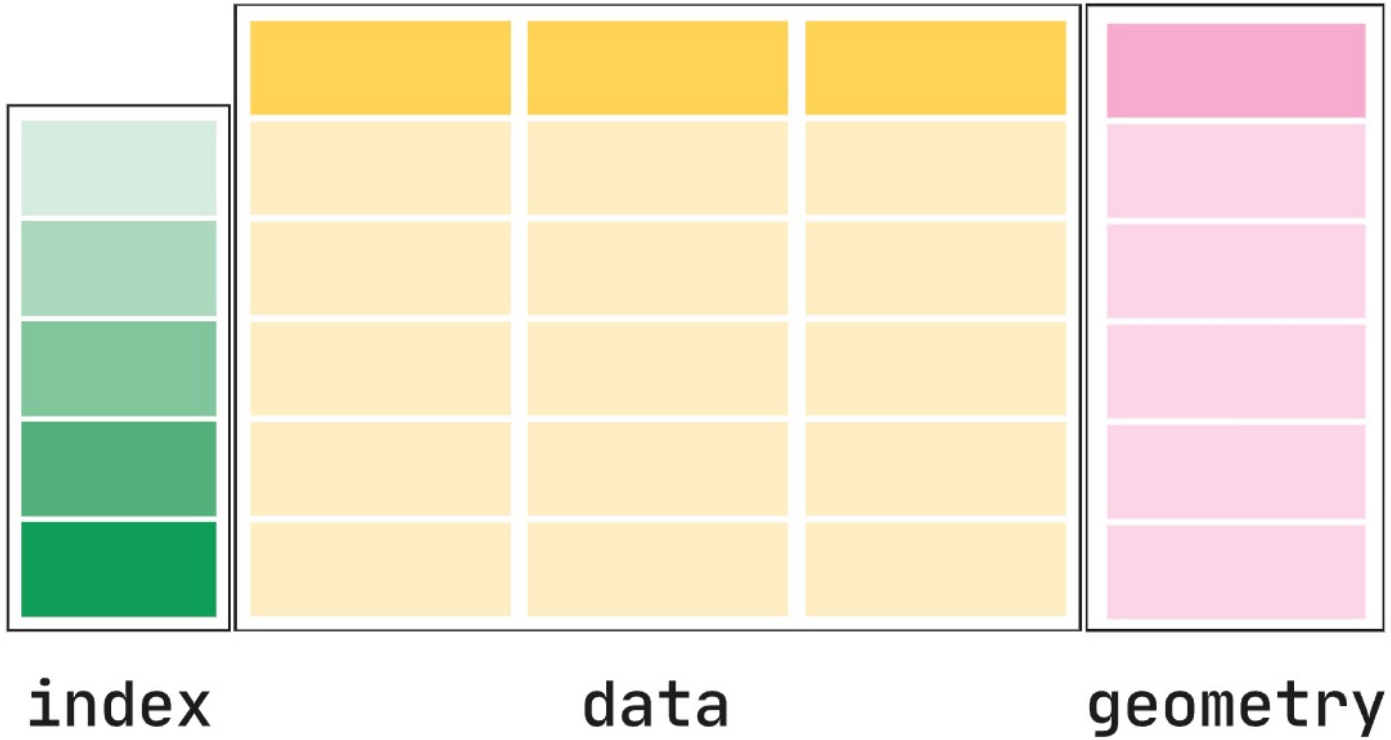
- Builds on several other libraries including **shapely** for geometric operations, **fiona** for file access and **matplotlib** for plotting.



GeoPandas

- Extends **pandas** to allow spatial operations on geometric types.

- Builds on several other libraries including **shapely** for geometric operations, **fiona** for file access and **matplotlib** for plotting.



Follow along!

- Go to the course Dropbox link → Lecture2 → Download the **us_mainland_states** shapefile
- Open up Spyder or a Jupyter Notebook
- Run the code in the next few slides

Reading geospatial data

[1]:
```
import geopandas

path_to_data = geopandas.datasets.get_path("nybb")
gdf = geopandas.read_file(path_to_data)

gdf
```

[1]:

	BoroCode	BoroName	Shape_Leng	Shape_Area	geometry
0	5	Staten Island	330470.010332	1.623820e+09	MULTIPOLYGON (((970217.022 145643.332, 970227....
1	4	Queens	896344.047763	3.045213e+09	MULTIPOLYGON (((1029606.077 156073.814, 102957...
2	3	Brooklyn	741080.523166	1.937479e+09	MULTIPOLYGON (((1021176.479 151374.797, 102100...
3	1	Manhattan	359299.096471	6.364715e+08	MULTIPOLYGON (((981219.056 188655.316, 980940....
4	2	Bronx	464392.991824	1.186925e+09	MULTIPOLYGON (((1012821.806 229228.265, 101278...

Measuring area

[3]:
```
gdf = gdf.set_index("BoroName")
```

[4]:
```
gdf["area"] = gdf.area
gdf["area"]
```

[4]:
```
BoroName
Staten Island    1.623822e+09
Queens           3.045214e+09
Brooklyn         1.937478e+09
Manhattan        6.364712e+08
Bronx            1.186926e+09
Name: area, dtype: float64
```

Compute polygon boundary and centroid

[5]:
```
gdf['boundary'] = gdf.boundary
gdf['boundary']
```

[5]:
```
BoroName
Staten Island    MULTILINESTRING ((970217.022 145643.332, 97022...
Queens           MULTILINESTRING ((1029606.077 156073.814, 1029...
Brooklyn         MULTILINESTRING ((1021176.479 151374.797, 1021...
Manhattan        MULTILINESTRING ((981219.056 188655.316, 98094...
Bronx            MULTILINESTRING ((1012821.806 229228.265, 1012...
Name: boundary, dtype: geometry
```

[6]:
```
gdf['centroid'] = gdf.centroid
gdf['centroid']
```

[6]:
```
BoroName
Staten Island     POINT (941639.450 150931.991)
Queens           POINT (1034578.078 197116.604)
Brooklyn          POINT (998769.115 174169.761)
Manhattan         POINT (993336.965 222451.437)
Bronx            POINT (1021174.790 249937.980)
Name: centroid, dtype: geometry
```

Compute distances

[7]:
```
first_point = gdf['centroid'].iloc[0]
gdf['distance'] = gdf['centroid'].distance(first_point)
gdf['distance']
```

[7]:
```
BoroName
Staten Island         0.000000
Queens           103781.535276
Brooklyn          61674.893421
Manhattan         88247.742789
Bronx            126996.283623
Name: distance, dtype: float64
```

Find projection

```
[23]: gdf.crs
```

```
[23]: <Projected CRS: EPSG:2263>
Name: NAD83 / New York Long Island (ftUS)
Axis Info [cartesian]:
- X[east]: Easting (US survey foot)
- Y[north]: Northing (US survey foot)
Area of Use:
- name: United States (USA) - New York - counties of Bronx; Kings; Nassau; New Yor
- bounds: (-74.26, 40.47, -71.8, 41.3)
Coordinate Operation:
- name: SPCS83 New York Long Island zone (US Survey feet)
- method: Lambert Conic Conformal (2SP)
Datum: North American Datum 1983
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

Convert projection

[24]:
```
gdf = gdf.set_geometry("geometry")
boroughs_4326 = gdf.to_crs("EPSG:4326")
```

[25]:
```
boroughs_4326.crs
```

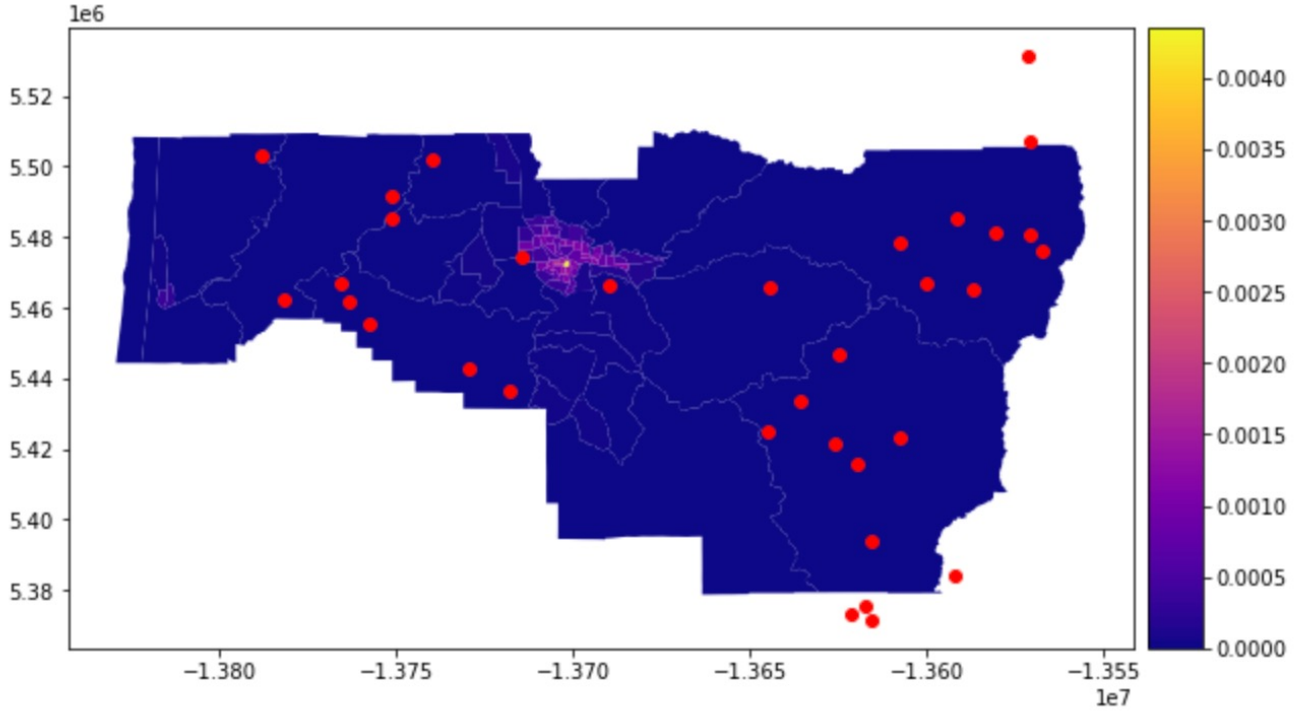
[25]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Write data to file

[2]:
```
gdf.to_file("my_file.geojson", driver="GeoJSON")
```

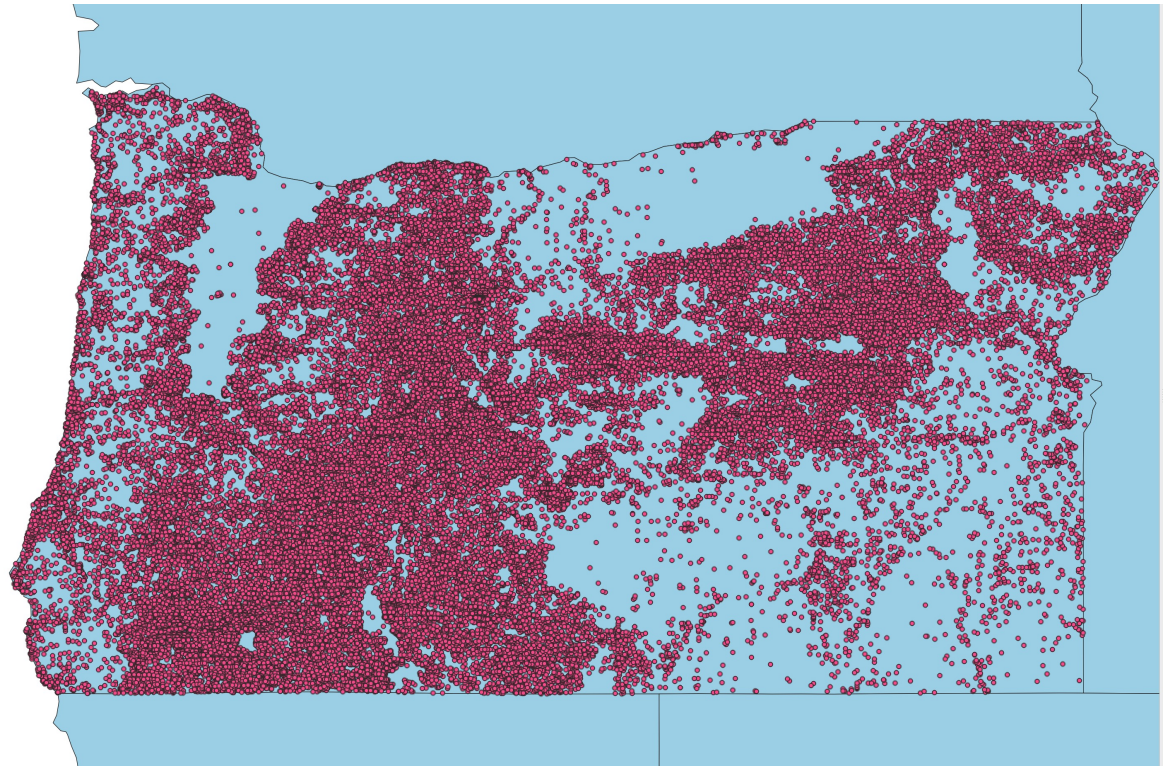
Lab #2

- Explore a large wildfire dataset using **GeoPandas**.
- Introduce **Census Bureau** data data variables and geographic units using **cenpy**.



Lab #2: Wildfire dataset

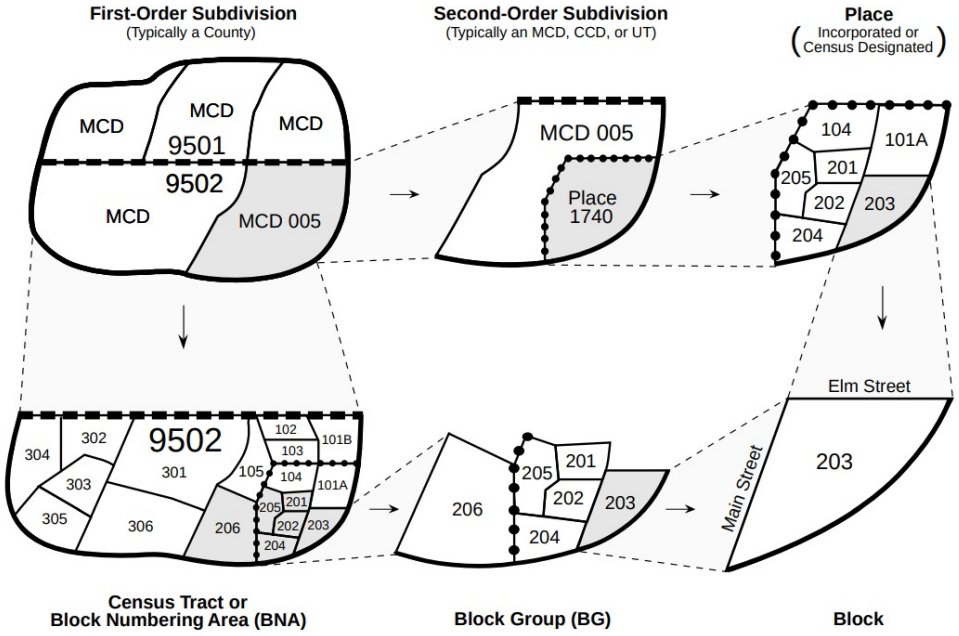
- Spatial database of wildfires that occurred in the Oregon from 1992 to 2018.
- Records of 67,402 fires



	DISCOVERY_	DISCOVER_1	DISCOVER_2	NWCG_CAUSE	NWCG_GENER	NWCG_CAU_1	CONT_DATE	CONT_DOY	CONT_TIME	FIRE_SIZE	FIRE_SIZE_	LATITUDE	LONGITUDE	OWNER_DESC	STATE	COUNTY	FIPS_CODE	FIPS_NAME
1	12/07/08 0...	190	1800	Natural	Natural	NULL	2012/07/30 0...	212.0000000...	0830	558198.300...	G	42.3918940...	-117.893687...	BLM	OR	Malheur	41045	Malheur Cou...
2	02/07/13 0...	194	1608	Natural	Natural	NULL	2002/11/08 0...	312.0000000...	1800	499945.000...	G	42.0388888...	-123.911666...	USFS	OR	NULL	NULL	NULL
3	14/07/14 0...	195	0146	Natural	Natural	NULL	2014/09/03 0...	246.000000...	1800	280141.0000...	G	43.3282999...	-118.224400...	BLM	OR	Malheur	41045	Malheur Cou...
4	17/07/12 0...	193	1645	Natural	Natural	NULL	2017/07/12 0...	193.000000...	1701	191125.0000...	G	42.2966666...	-123.953611...	USFS	OR	015	41015	Curry County
5	12/07/08 0...	190	1602	Natural	Natural	NULL	2012/11/16 00...	321.000000...	1300	160801.0000...	G	42.8186000...	-119.174999...	BLM	OR	Harney	41025	Harney County
6	96/08/10 0...	223	1605	Human	Equipment an...	NULL	1996/09/19 0...	263.000000...	1400	118230.0000...	G	44.98319999...	-121.351200...	BIA	OR	NULL	NULL	NULL
7	15/08/12 0...	224	0330	Natural	Natural	NULL	2015/11/01 0...	305.000000...	1200	101028.0000...	G	44.31779999...	-118.929199...	USFS	OR	Grant	41023	Grant County
8	18/06/21 0...	172	0748	Natural	Natural	NULL	2018/07/12 0...	193.000000...	1518	100226.000...	G	45.15970000...	-121.070499...	PRIVATE	OR	Wasco	41065	Wasco County
9	12/08/06 0...	219	1634	Natural	Natural	NULL	2012/09/17 0...	261.000000...	1800	92977.0000...	G	42.11166666...	-120.801388...	USFS	OR	037	41037	Lake County
10	18/07/16 0...	197	0143	Natural	Natural	NULL	NULL	NULL	NULL	87701.00000...	G	42.3694444...	-123.860277...	USFS	OR	033	41033	Josephine C...
11	14/07/14 0...	195	0553	Natural	Natural	NULL	2014/09/03 0...	246.000000...	1800	87141.00000...	G	43.61610000...	-118.425600...	BLM	OR	Harney	41025	Harney County

Lab #2: Census data

- We will use data from the American Community Survey (ACS) product
- Provides social and economic information every year but, unlike the Decenniel Survey, only represents a sample (about 3.5 million) of US households.




```
In [1]: from cenpy import products
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: chicago = products.ACS(2017).from_place('Chicago, IL', level='tract',
                                                variables=['B00002*', 'B01002H_001E'])

        Matched: Chicago, IL to Chicago city within layer Incorporated Places

In [3]: f, ax = plt.subplots(1,1,figsize=(20,20))
        chicago.dropna(subset=['B00002_001E'], axis=0).plot('B00002_001E', ax=ax, cmap='plasma')
        ax.set_facecolor('k')
```