

The background features a dark blue gradient with a subtle pattern of white dots. On the left side, there are several overlapping circular elements. A prominent one is a large circle with a scale around its perimeter, ranging from 140 to 260 in increments of 10. Other circles of varying sizes and styles (solid, dashed, dotted) are scattered across the scene, some with arrows indicating direction or rotation.

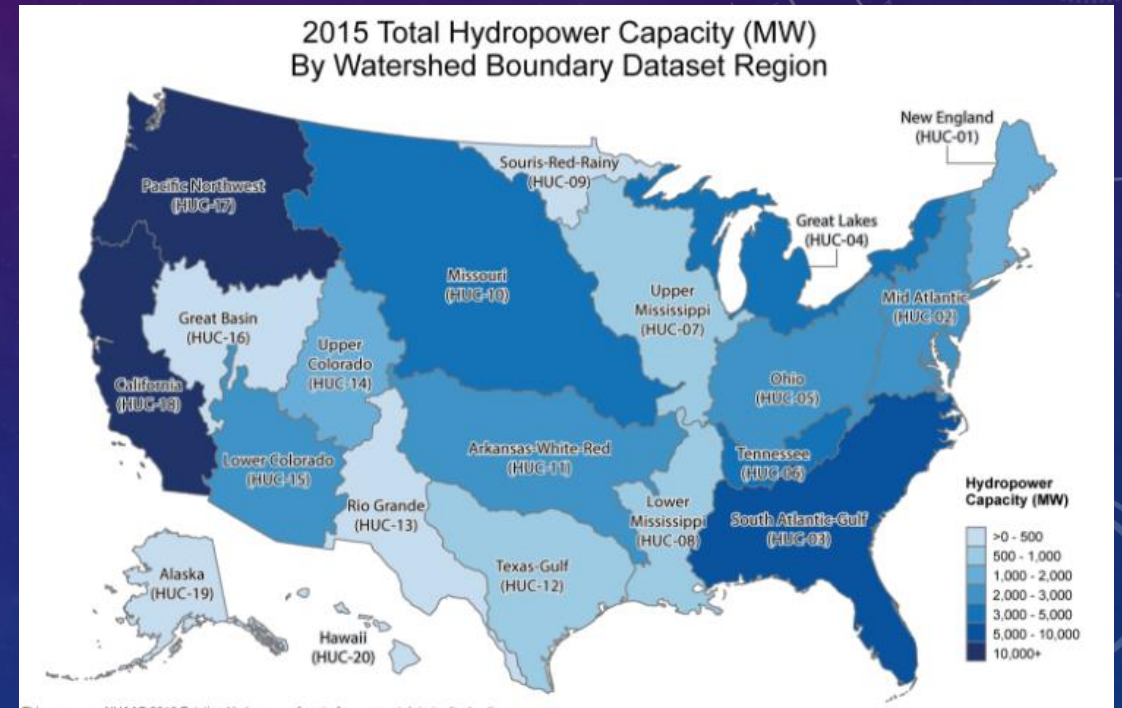
FIND A POLYGON BY A LOCATION

GIS SERIES 1

JONG LEE

TERMS

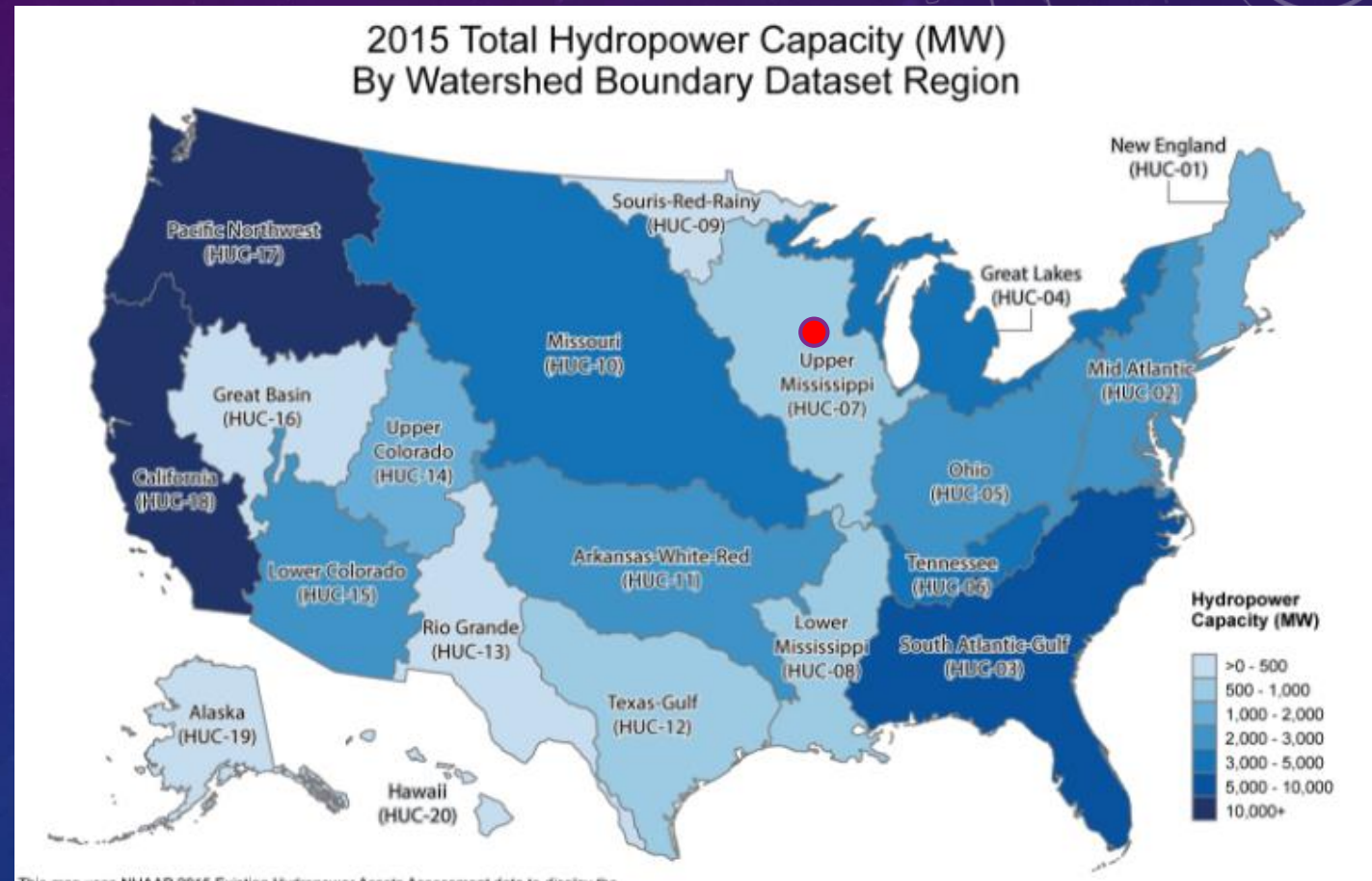
- HUC (Hydrologic Unit Code): Code for watershed
- HUC has levels: HUC2, HUC4, HUC6, HUC8, HUC10, HUC12
 - HUC2, HUC4, HUC6, HUC8 are popular for general use
 - HUC10, HUC12 are used by local watershed/water management



HUC2

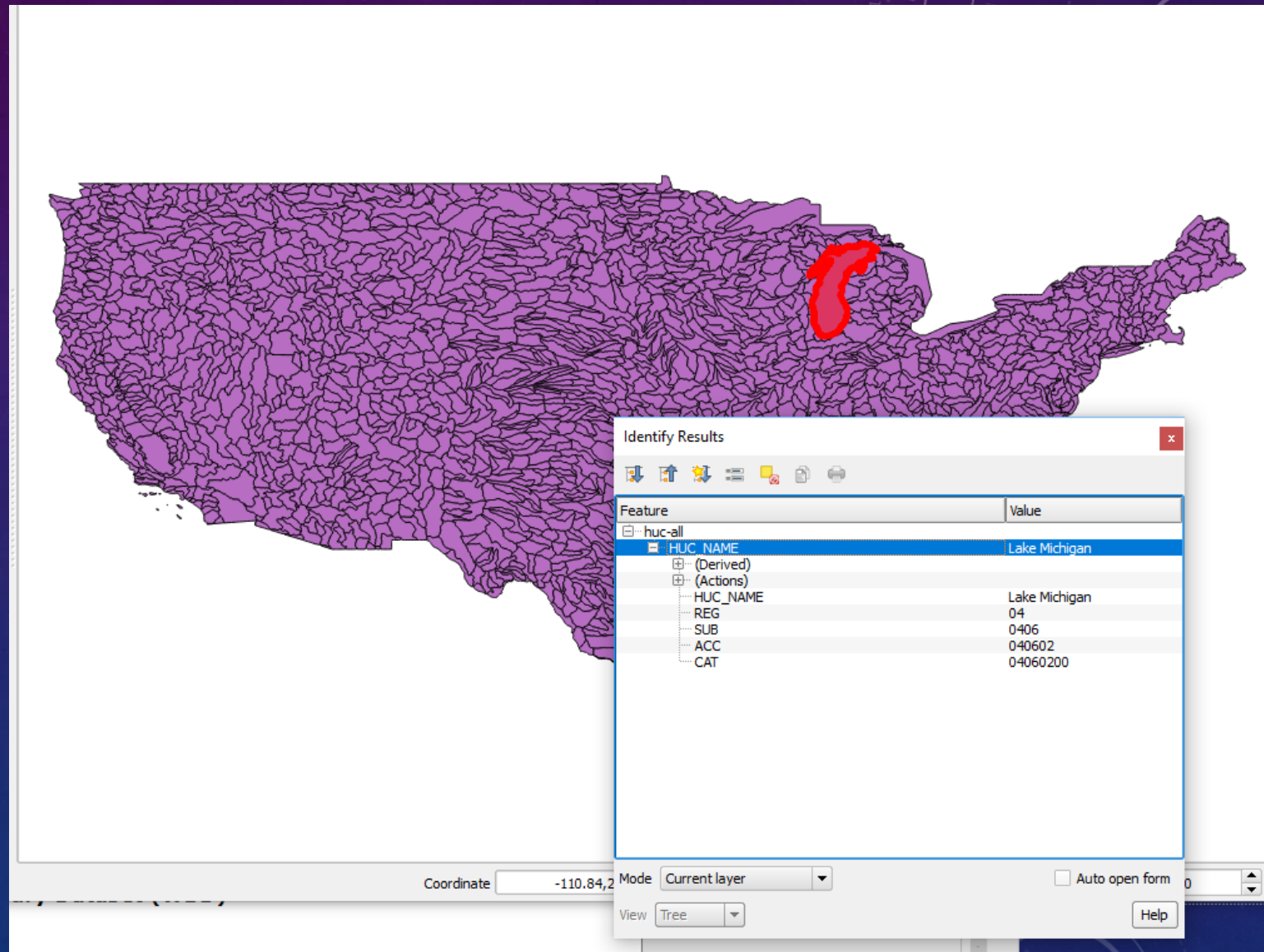
PROBLEM

- Need to find which HUC a sensor is belong to.
 - A sensor has a location with lat/lon
 - HUC2, HUC4, HUC6, HUC8



DATA

- USGS published (updating) a shapefile for HUC polygons.
 - NAD83 (EPSG:4269)
 - Contains information about HUC2, HUC4, HUC6, HUC8
 - 2158 polygons
 - 51 MB



PYTHON LIBRARY

- Pandas
 - Data analysis library on top of Numpy, Scipy, etc.
- Geopandas
 - Pandas with geospatial capability by using Fiona, pyproj, shapely
- Shapely
 - Manipulation and analysis of Geometry objects (2D)

CODE – HUC.PY

- DataFrame: 2-D labeled data structure
- GeoDataFrame: DataFrame with Geometry
- sjoin: spatial join
 - two geometry objects are merged based on their spatial relationship to one another

```
1 import geopandas as gpd
2 import pandas as pd
3 import shapely.wkt
4 from geopandas.tools import sjoin
5
6 class HucFinder:
7
8     def __init__(self, huc_data_file):
9         # initialize with shapefile
10        self.hucData = gpd.GeoDataFrame.from_file(huc_data_file)
11
12    def getHuc(self, lat, lon):
13        # create a geodataframe with lat/lon
14        wkt = 'POINT('+str(lon)+' '+ str(lat)+')'
15        geometry = [shapely.wkt.loads(wkt)]
16        crs = {'init': 'epsg:4269'}
17        point = gpd.GeoDataFrame(pd.DataFrame({'id': [0]}), crs=crs,
18                                 geometry=geometry)
19
20    try:
21        # find a huc polygon contains the point
22        huc = sjoin(point, self.hucData, how='inner', op='intersects')
23
24        # if there is no huc reutrnr empty dictionary
25        if(len(huc.index) == 0):
26            return {}
27
28        # if there is a huc, create a dictionary
29        result = {'huc_name': huc['HUC_NAME'][0],
30                'huc2': huc['REG'][0],
31                'huc4': huc['SUB'][0],
32                'huc6': huc['ACC'][0],
33                'huc8': huc['CAT'][0] }
34        return result
35    except ValueError:
36        # if there is no huc reutrnr empty dictionary
37        return {}
```

CODE – TEST-HUC.PY

```
1 from huc import HucFinder
2
3 if __name__ == '__main__':
4     hucfinder = HucFinder('huc-all.shp')
5     print(hucfinder.getHuc(lat=38, lon=-85.74))
6     print(hucfinder.getHuc(lat=39, lon=-85.74))
7     print(hucfinder.getHuc(lat=99, lon=-85.74))
8     print(hucfinder.getHuc(lat=37, lon=-84.00))
9     print(hucfinder.getHuc(lat=40, lon=-84.55))
10    print(hucfinder.getHuc(lat=36, lon=-82.74))
11    print(hucfinder.getHuc(lat=38, lon=-83.74))
12
```


LARGER DATASET

- Use Spatial database such as Postgis

QUESTION

- What are two biggest assumptions in this geospatial operation?