PS4 v1.2: Spatial

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points. We use (*) to indicate a problem that we think might be time consuming.

Style Points (10 pts)

Please refer to the minilesson on code style **here**.

Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Tarini Dewan, tarinidewan
    - Partner 2 (name and cnet ID): Shreya Shravini, shreyashravini
3. Partner 1 will accept the `ps4` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: SS TD
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**": SS TD* (1 point)
6. Late coins used this pset: 1 Late coins left after submission: 3
7. Knit your `ps4.qmd` to an PDF file to make `ps4.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps4.qmd` and `ps4.pdf` to your github repo.
9. (Partner 1): submit `ps4.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

Important: Repositories are for tracking code. **Do not commit the data or shapefiles to your repo.** The best way to do this is with `.gitignore`, which we have covered in class. If you do accidentally commit the data, Github has a guide. The best course of action depends on whether you have pushed yet. This also means that both partners will have to download the initial raw data and any data cleaning code will need to be re-run on both partners' computers.

Important Note: We created separate datasets to work on our respective sections individually, rather than sharing the datasets. So, the names of dataframes and columns will vary between sections.

```
import pandas as pd
import altair as alt
alt.data_transformers.disable_max_rows()
alt.renderers.enable("png", ppi=250)
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

Download and explore the Provider of Services (POS) file (10 pts)

1. The variables are called:

- PRVDR_CTGRY_SBTYP_CD: Provider Category Subtype Code
- PRVDR_CTGRY_CD: Provider Category Code
- PRVDR_NUM: CCN (CMS Certification Number)
- PGM_TRMNTN_CD: Termination Code
- ZIP_CD: ZIP Code

2.

```
# import pos2016.csv
pos2016 =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tari

# create a year column
pos2016['YEAR'] = '2016'

# subset data to short-term hospitals
pos2016_hosp = pos2016[(pos2016['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2016['PRVDR_CTGRY_SBTYP_CD'] == 1)].reset_index(drop = True)
pos2016_hosp
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	FAC_NAME
0	1.0	1	SOUTHEAST ALABAMA MEDICAL CE
1	1.0	1	NORTH JACKSON HOSPITAL
2	1.0	1	MARSHALL MEDICAL CENTER SOUT
3	1.0	1	ELIZA COFFEE MEMORIAL HOSPITAI
4	1.0	1	MIZELL MEMORIAL HOSPITAL
...	...	...	...
7240	1.0	1	WEIMAR MEDICAL CENTER
7241	1.0	1	CLEVELAND EMERGENCY HOSPTIAI
7242	1.0	1	WISE HEALTH SYSTEM
7243	1.0	1	TEXAS GENERAL HOSPITAL- VZRMC
7244	1.0	1	FIRST TEXAS HOSPITAL

a. There are 7,245 hospitals reported in the data. This number seems greater than the actual number of hospitals.

```
pos2016_hosp.shape
pos2016_hosp['PRVDR_NUM'].nunique()
```

```
7245
```

b. According to the American Hospital Association, there were a total of 5,534 hospitals in FY 2016. The mismatch could be because the number in our data includes hospitals that have merged or closed (CMS = 01).

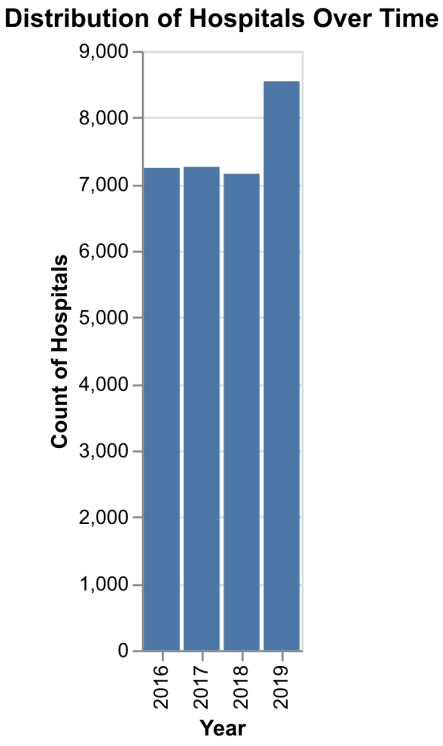
Attribution:
https://www.aha.org/system/files/2018-01/Fast%20Facts%202018%20pie%20charts.pdf

```
# check termination status
pos2016_hosp['PGM_TRMNTN_CD'].value_counts()
```

```
PGM_TRMNTN_CD
0    3423
1    1990
7    1481
4     223
5      75
3      29
6      13
2      11
Name: count, dtype: int64
```

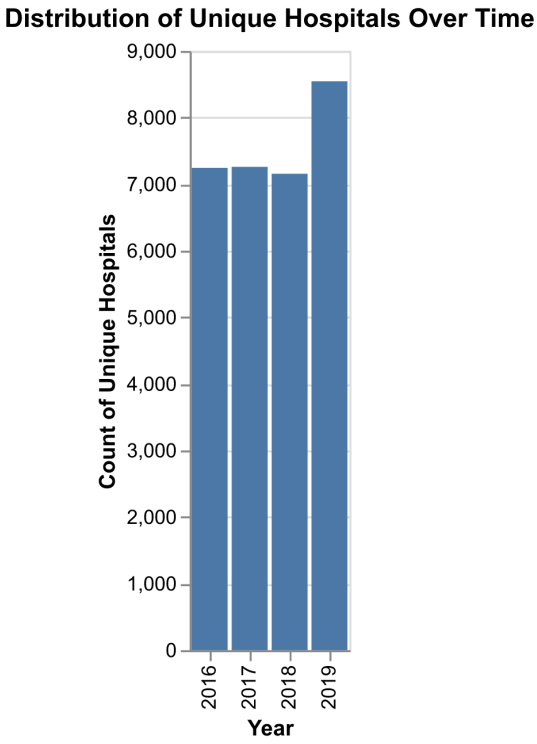
3.

```
# pos2017.csv
pos2017 =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tari
pos2017['YEAR'] = '2017' # year column
pos2017_hosp = pos2017[(pos2017['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2017['PRVDR_CTGRY_SBTYP_CD'] == 1)] # subsetting data

# pos2018.csv
pos2018 =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tari
 ↪  encoding = 'latin1')
pos2018['YEAR'] = '2018' # year column
pos2018_hosp = pos2018[(pos2017['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2018['PRVDR_CTGRY_SBTYP_CD'] == 1)] # subsetting data

# pos2019.csv
pos2019 =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tari
 ↪  encoding = 'latin1')
pos2019['YEAR'] = '2019' # year column
pos2019_hosp = pos2019[(pos2017['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2019['PRVDR_CTGRY_SBTYP_CD'] == 1)] # subsetting data

# append datasets
pos_df = pd.concat([pos2016_hosp, pos2017_hosp, pos2018_hosp, pos2019_hosp])

# plot the number of observations in your dataset by year
alt.Chart(pos_df).mark_bar().encode(
    x = alt.X('YEAR:N', title = 'Year'),
    y = alt.Y('count(PRVDR_CTGRY_SBTYP_CD):Q', title = 'Count of Hospitals')
).properties(
    title = 'Distribution of Hospitals Over Time'
)
```



4. a.

```
# dropping duplicate CMS numbers across years
pos_df_uniq = pos_df.drop_duplicates(subset = ['PRVDR_NUM', 'YEAR'], keep =
 ↪  'first')

# plot the number of unique hospitals in your dataset by year
alt.Chart(pos_df_uniq).mark_bar().encode(
    x = alt.X('YEAR:N', title = 'Year'),
    y = alt.Y('count(PRVDR_CTGRY_SBTYP_CD):Q', title = 'Count of Unique
 ↪  Hospitals')
).properties(
    title = 'Distribution of Unique Hospitals Over Time'
)
```



b. The plots are the same which tells us that the dataset is structured such that each row represents a unique hospital per year, with no repeated entries for any hospital within the same year.

Identify hospital closures in POS file (15 pts) (*)

IGNORE

```
dfdict = {2016:pos2016, 2017:pos2017, 2018:pos2018, 2019:pos2019}
lofdf = []
for year, df in dfdict.items():
    tempdf=df.copy()
    tempdf[f'active_{year}'] = 0
    tempdf.loc[(tempdf['PGM_TRMNTN_CD'] == 0), f'active_{year}'] = 1
    tempdf[f'ZIP_CD_{year}'] = tempdf['ZIP_CD']
    tempdf[f'FAC_NAME_{year}'] = tempdf['FAC_NAME']
    tempdf = tempdf[['PRVDR_NUM', f'ZIP_CD_{year}',f'active_{year}'
 ↪  ,f'FAC_NAME_{year}' ]]
    lofdf.append(tempdf)
```

```
df2016 = lofdf[0]
df2017 = lofdf[1]
df2018 = lofdf[2]
df2019 = lofdf[3]

track_df = pd.DataFrame()
for df in lofdf:
    if len(track_df) == 0:
        track_df = df2016
        continue
    track_df = pd.merge(track_df,df, how='left', on= 'PRVDR_NUM')
track_df = track_df.fillna(0)


# Filter hospitals active in 2016
active_2016 = track_df[track_df['active_2016'] == 1]

# Create an empty list to store hospitals suspected to have closed by 2019
suspected_closed_hospitals = []

# Iterate through the 2016 active hospitals
for index, row in active_2016.iterrows():
    # Check if the hospital became inactive or disappeared in 2017, 2018, or
    ↪ 2019
    if (row['active_2017'] == 0 or pd.isnull(row['active_2017'])):
        suspected_closed_hospitals.append({
            'Facility Name': row['FAC_NAME_2016'],
            'ZIP': row['ZIP_CD_2016'],
            'Year of Suspected Closure': 2017
        })
    elif (row['active_2018'] == 0 or pd.isnull(row['active_2018'])):
        suspected_closed_hospitals.append({
            'Facility Name': row['FAC_NAME_2016'],
            'ZIP': row['ZIP_CD_2016'],
            'Year of Suspected Closure': 2018
        })
    elif (row['active_2019'] == 0 or pd.isnull(row['active_2019'])):
        suspected_closed_hospitals.append({
            'Facility Name': row['FAC_NAME_2016'],
            'ZIP': row['ZIP_CD_2016'],
            'Year of Suspected Closure': 2019
        })
```

```
# Convert the list to a DataFrame for easier analysis
closed_hospitals_df = pd.DataFrame(suspected_closed_hospitals)

# Count the number of hospitals that fit this definition
num_suspected_closed_hospitals = len(closed_hospitals_df)

print(f"Number of hospitals suspected to have closed by 2019:
 ↪  {num_suspected_closed_hospitals}")
closed_hospitals_df.head()

# Calculate the number of active hospitals by zip code for each year
zip_counts = {
    2017: track_df[track_df['active_2017'] ==
     ↪  1].groupby('ZIP_CD_2017').size(),
    2018: track_df[track_df['active_2018'] ==
     ↪  1].groupby('ZIP_CD_2018').size(),
    2016: track_df[track_df['active_2016'] ==
     ↪  1].groupby('ZIP_CD_2016').size(),
    2019: track_df[track_df['active_2019'] ==
     ↪  1].groupby('ZIP_CD_2019').size(),
}

# Filter suspected closures where the zip code count does not decrease in the
 ↪  next year
filtered_hospitals = []

for hospital in suspected_closed_hospitals:
    zip_code = hospital['ZIP']
    year_of_closure = hospital['Year of Suspected Closure']

    # Check if active hospital count in zip code decreased in the following
     ↪  year
    if year_of_closure in zip_counts and (year_of_closure + 1) in zip_counts:
        if zip_code in zip_counts[year_of_closure] and zip_code in
         ↪  zip_counts[year_of_closure + 1]:
            # Only keep the hospital if the count decreased
            if zip_counts[year_of_closure][zip_code] >
             ↪  zip_counts[year_of_closure + 1][zip_code]:
                filtered_hospitals.append(hospital)

# Convert the filtered list to a DataFrame
```

```
filtered_hospitals_df = pd.DataFrame(filtered_hospitals)

# Sort by Facility Name and get the first 10 hospitals
first_10_filtered_hospitals = filtered_hospitals_df.sort_values(by='Facility
 ↪  Name')[['Facility Name', 'Year of Suspected Closure']].head(10)

# Display the result
print(first_10_filtered_hospitals)
```

```
Number of hospitals suspected to have closed by 2019: 5492
                          Facility Name  Year of Suspected Closure
168             1ST HOME HEALTH CARE INC                       2018
247    24SEVEN HEALTH CARE SERVICES, INC                       2018
738                  5 STAR HOSPICE LLC                        2017
968                  A & A HOSPICE, INC                        2018
788          A & L HOSPICE SERVICES, INC                       2018
869  A & T MULTI-HEALTHCARE SERVICES LLC                       2018
531                        A C L D, INC                        2018
529                        A C L D, INC                        2018
431        A&A RELIABLE HOME HEALTH CARE                       2018
963             A&M MIRACLE HOSPICE, INC                       2018
```

Download Census zip code shapefile (10 pt)

1. a. The five file types are:

   - .dbf: The dBASE table that stores the attribute information of features; data is stored in an array with multiple records and fields (has attribute info)
   - .prj: The file that stores the coordinate system information
   - .shp: The main file that stores the feature geometry (points, lines, polygons)
   - .shx: The index file that stores the index of the feature geometry (has positional index)
   - .xml: file format that stores information about the shapefile

   # Attribution: https://desktop.arcgis.com/en/arcmap/latest/manage-data/shapefiles/shapefile-file-extensions.htm

   b. The size for each of the files is given in the output below.

```
import os

# define the folder path
folder_path =
 ↪  '/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tarini/gz_2010_us

# get file size
for file_name in os.listdir(folder_path):
    file_path = os.path.join(folder_path, file_name)
    file_size_mb = os.path.getsize(file_path) / (1024 * 1024)  # Convert size
 ↪  to MB
    print(f'File Name: {file_name}, File Size: {file_size_mb:.2f} MB')
```

```
File Name: gz_2010_us_860_00_500k.prj, File Size: 0.00 MB
File Name: gz_2010_us_860_00_500k.shx, File Size: 0.25 MB
File Name: gz_2010_us_860_00_500k.shp, File Size: 798.74 MB
File Name: gz_2010_us_860_00_500k.dbf, File Size: 6.13 MB
File Name: gz_2010_us_860_00_500k.xml, File Size: 0.01 MB
```

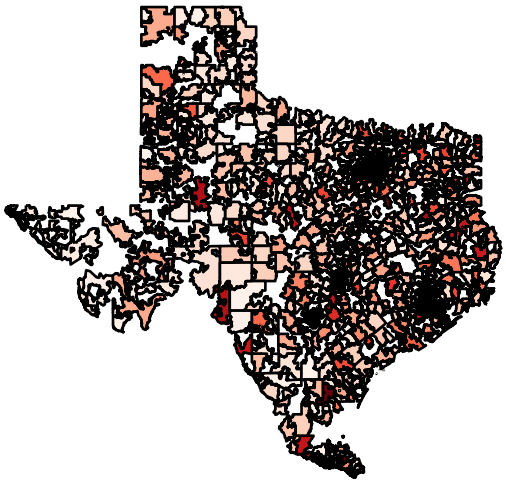
2. The number of hospitals per zip code is given in the output below.

```
import geopandas as gpd
import matplotlib.pyplot as plt

# loading .shp file
filepath =
 ↪  '/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tarini/gz_2010_us
texas_df = gpd.read_file(filepath)

# filtering data to Texas zip codes
texas_df['Texas'] = texas_df['ZCTA5'].apply(lambda x: 1 if
 ↪  x.startswith(('75', '76', '77', '78', '79')) else 0)
texas_df = texas_df[texas_df['Texas'] == 1]

# convert zip codes in both datasets to string
texas_df['ZCTA5'] = texas_df['ZCTA5'].astype(str)
pos2016['ZIP_CD'] = pos2016['ZIP_CD'].fillna(0) # account for NAs in pos2016
 ↪  zip code data
pos2016['ZIP_CD'] = pos2016['ZIP_CD'].astype(int).astype(str)

# merge both datasets by zip code
hosp_tex_2016 = pd.merge(texas_df, pos2016, how = 'inner', left_on = 'ZCTA5',
 ↪  right_on = 'ZIP_CD')
```

```
# number of hospitals in Texas per zip code 2016
print(hosp_tex_2016['ZCTA5'].value_counts())
hosp_per_zip = hosp_tex_2016.groupby('ZIP_CD').size().reset_index(name =
 ↪  'hospital_count')

# merge with spatial geometry data for zip codes
hosp_geo = hosp_tex_2016[['ZIP_CD', 'geometry']].drop_duplicates().merge(
    hosp_per_zip, on = 'ZIP_CD', how = 'left').fillna(0)

# choropleth of hospitals by zip code
plt.figure(figsize=(20, 16), dpi=1000)
hosp_geo.plot(column='hospital_count',
            cmap='Reds',
              edgecolor='black',
              vmin=0,  # Start color scale at 0
              vmax=50   # Adjust this value to increase contrast
              ).set_axis_off()
plt.title('Number of Hospitals per ZIP Code in Texas 2016')
plt.show()

#fig, ax = plt.subplots(figsize=(10, 10))
#hosp_geo.plot(column='hospital_count', legend=True, ax=ax).set_axis_off()
#plt.title('Hospitals per ZIP Code in Texas')
#plt.show()
```

```
ZCTA5
77036    195
78229    148
77074    112
76104    108
78501     97
        ...
76155      1
78248      1
79036      1
77622      1
75757      1
Name: count, Length: 1194, dtype: int64

<Figure size 20000x16000 with 0 Axes>
```

Number of Hospitals per ZIP Code in Texas 2016



Effects of closures on access in Texas (15 pts)

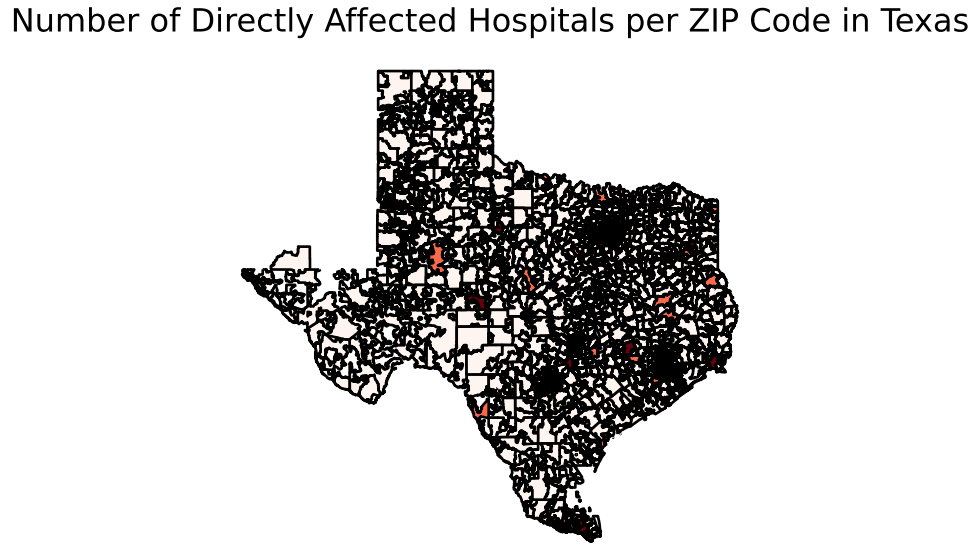
1. The number of hospital closures by ZIP is given in the output below.

```
# convert zip code in corrected hospital data to string
filtered_hospitals_df['ZIP'] =
 ↪  filtered_hospitals_df['ZIP'].astype(int).astype(str)

# merge texas data and closed hospital data only on closed hospital data
clos_tex = pd.merge(texas_df, filtered_hospitals_df, how = 'inner', left_on =
 ↪  'ZCTA5', right_on = 'ZIP')

# create a list of directly affected zip codes in Texas
affected_zips = clos_tex['ZIP'].unique().tolist()

# table of the number of closures by zipcode
clos_zips = clos_tex.groupby('ZIP').size().reset_index(name =
 ↪  'closure_count')
```

2. There are 103 directly affected ZIP codes in Texas.

```
from matplotlib.colors import LogNorm

clos_zips.shape

# merge texas data and closed hospital data only on all Texas hospital data
clos_tex_all = pd.merge(texas_df, filtered_hospitals_df, how = 'left',
 ↪  left_on = 'ZCTA5', right_on = 'ZIP')

# calculate the number of closed hospitals in Texas in the merged dataframe
clos_tex_all['Closed'] = clos_tex_all['Year of Suspected Closure'].fillna(0)
clos_tex_all['Closed'] = clos_tex_all['Closed'].apply(lambda x: 0 if x == 0
 ↪  else 1)

# group by zip codes and count the number of closed hospitals per zip
clos_df_agg = clos_tex_all.groupby(['ZCTA5'])['Closed'].sum().reset_index()

# merge with spatial geometry data for zip codes
clos_geo = pd.merge(clos_df_agg, texas_df, how = 'right', left_on = 'ZCTA5',
 ↪  right_on = 'ZCTA5')
clos_geo = gpd.GeoDataFrame(clos_geo, geometry = 'geometry')

# choropleth of closed hospitals by zip code
plt.figure(figsize=(20, 16), dpi=1000)
clos_geo.plot(column='Closed',
            cmap='Reds',
              edgecolor='black',
              vmin=0,  # Start color scale at 0
              vmax=2   # Adjust this value to increase contrast
              ).set_axis_off()
plt.title('Number of Directly Affected Hospitals per ZIP Code in Texas')
plt.show()
#fig, ax = plt.subplots(figsize=(10, 10))
#clos_geo.plot(column='Closed', cmap='OrRd', legend=True,
 ↪  ax=ax).set_axis_off()
#plt.title('Number of Directly Affected Hospitals per ZIP Code in Texas')
#plt.show()
```

```
<Figure size 20000x16000 with 0 Axes>
```



3. There are 1009 indirectly affected ZIP codes in Texas.

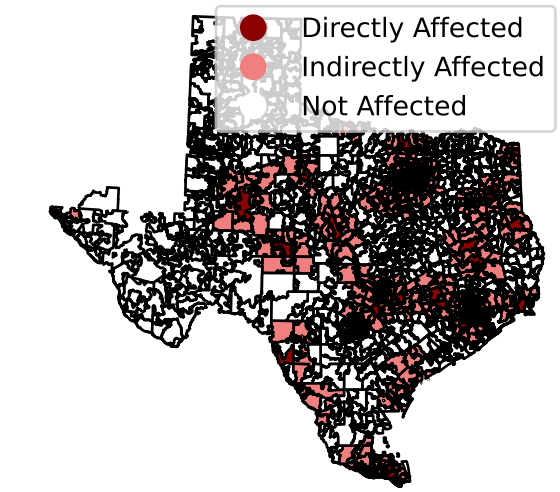
```
clos_geo['geometry'].area

# convert square degrees to square meters
clos_geo = clos_geo.to_crs('EPSG:32614')

# Attribution: ChatGPT
# Query: "Here is the content of my .prj file. What unit is the area in?
# Response: In your .prj file, the area units are in degrees. If you are
 ↪  calculating areas using these coordinates, the results will likely be in
 ↪  square degrees, which is not a standard unit for area. To get area
 ↪  measurements in more conventional units like square meters or square
 ↪  kilometers, you would need to project your data to a suitable projected
 ↪  coordinate system, such as NAD83 / UTM Zone appropriate for your region.
# Query: Help me convert this unit to meters.

# create a 10-mile buffer around each directly affected zip code (10 miles =
 ↪  16093.4 meters)
clos_geo2 = clos_geo[clos_geo['Closed'] >= 1]
buffer_10 = clos_geo2.buffer(16093.4)

# convert buffer back to a GeoDataFrame
buffer_gdf = gpd.GeoDataFrame(geometry = buffer_10, crs = clos_geo2.crs)
```

```
texas_df = texas_df.to_crs('EPSG:32614')

# spatial join with original Texas shapefile
ind_affected = gpd.sjoin(texas_df, buffer_gdf, how = 'inner', predicate =
 ↪  'intersects')
print(ind_affected['ZCTA5'].nunique())
```

```
1009
```

4.

```
from matplotlib.colors import ListedColormap

# classifying zip codes as affected, not affected, indirectly affected in
 ↪  main Texas shapefile
texas_df['status'] = 'Not Affected'
texas_df.loc[texas_df['ZCTA5'].isin(ind_affected['ZCTA5']), 'status'] =
 ↪  'Indirectly Affected'
texas_df.loc[texas_df['ZCTA5'].isin(clos_tex['ZCTA5']), 'status'] = 'Directly
 ↪  Affected'

# choropleth
colors = ['darkred', 'lightcoral', 'white']
custom_cmap = ListedColormap(colors)
texas_df.plot(column = 'status', cmap = custom_cmap, legend = True, edgecolor
 ↪  = 'black').set_axis_off()
plt.title('Hospital Closure Effects by ZIP Code')
plt.show()
```



Reflecting on the exercise (10 pts)

1.

a. Issues with Current Method:

- The method only checks if the total number of hospitals in a ZIP code decreases, but doesn't verify if it's the same hospital that closed
- A ZIP code could have two changes at once (one closure and one opening) that would mask the closure
- Different hospitals in the same ZIP code could swap CMS numbers due to administrative changes

b. Ways to Improve:

- Look at facility names and addresses to track hospitals across years even if CMS numbers change
- Compare hospital characteristics (like bed size, services) between years to better identify if a "new" hospital is actually the same facility
- Track staffing levels or patient volume to distinguish between true closures and administrative changes
- Consider tracking changes in nearby ZIP codes too, as mergers might involve facilities in adjacent areas

```
---
title: "PS4 v1.2: Spatial"
format:
  pdf:
    keep-tex: true
    include-in-header:
      text: |
        \usepackage{fvextra}
        \DefineVerbatimEnvironment{Highlighting}{Verbatim}{breaklines,commandchars=\\\{\}}
        \usepackage{hyperref}
include-before-body:
  text: |
    \RecustomVerbatimEnvironment{verbatim}{Verbatim}{
      showspaces = false,
      showtabs = false,
      breaksymbolleft={},
      breaklines
    }
---
```

****PS4:**** Due Sat Nov 2 at 5:00PM Central. Worth 100 points.

We use (`*`) to indicate a problem that we think might be time consuming.

Style Points (10 pts)

Please refer to the minilesson on code style

****[here](**https://uchicago.zoom.us/rec/share/pG_wQ-pHTQrJTmqNn4rcrw5V194M2H2s-2jdy8oVhWHkd_yZt9o162IWurpA-fxU.BIQlSgZLRYctvzp-**)****.

Submission Steps (10 pts)

1. This problem set is a paired problem set.

2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.

- Partner 1 (name and cnet ID): Tarini Dewan, tarinidewan

- Partner 2 (name and cnet ID): Shreya Shravini, shreyashravini

3. Partner 1 will accept the `ps4` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.

4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: SS TD

5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **[here](https://docs.google.com/forms/d/185usrCREQaUbvAXpWhChkjghdGgmAZXA3lPWpXLLsts/edit)**": SS TD* (1 point)

6. Late coins used this pset: 1 Late coins left after submission: 3

7. Knit your `ps4.qmd` to an PDF file to make `ps4.pdf`,

* The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.

8. (Partner 1): push `ps4.qmd` and `ps4.pdf` to your github repo.

9. (Partner 1): submit `ps4.pdf` via Gradescope. Add your partner on Gradescope.

10. (Partner 1): tag your submission in Gradescope

Important: Repositories are for tracking code. **Do not commit the data or shapefiles to your repo.** The best way to do this is with `.gitignore`, which we have covered in class. If you do accidentally commit the data, Github has a [guide](https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github#removing-files-from-a-repositorys-history). The best course of action depends on whether you have pushed yet. This also means that both partners will have to download the initial raw data and any data cleaning code will need to be re-run on both partners' computers.

Note: We created separate datasets to work on our respective sections individually, rather than sharing the datasets. So, the names of dataframes and columns will vary between sections.

```{python}
import pandas as pd

import altair as alt

alt.data_transformers.disable_max_rows()

alt.renderers.enable("png", ppi=250)

import numpy as np


import warnings
```

warnings.filterwarnings('ignore')

```

**## Identify hospital closures in POS file (15 pts) (*)**

1.

```{python}

import pandas as pd

# Load the POS data for each year

pos2016 = pd.read_csv(r'C:\Users\Shreya Work\OneDrive\Documents\GitHub\problem-set-4-shreya-and-tarini\pos2016.csv', encoding='ISO-8859-1')

pos2017 = pd.read_csv(r'C:\Users\Shreya Work\OneDrive\Documents\GitHub\problem-set-4-shreya-and-tarini\pos2017.csv', encoding='ISO-8859-1')

pos2018 = pd.read_csv(r'C:\Users\Shreya Work\OneDrive\Documents\GitHub\problem-set-4-shreya-and-tarini\pos2018.csv', encoding='ISO-8859-1')

pos2019 = pd.read_csv(r'C:\Users\Shreya Work\OneDrive\Documents\GitHub\problem-set-4-shreya-and-tarini\pos2019.csv', encoding='ISO-8859-1')

# Add a 'Year' column to each DataFrame

pos2016['Year'] = 2016

pos2017['Year'] = 2017

pos2018['Year'] = 2018

pos2019['Year'] = 2019

# Concatenate all years into a single DataFrame

combined_data = pd.concat([pos2016, pos2017, pos2018, pos2019], ignore_index=True)

# Filter data for hospitals that were active in 2016
```
```

```
active_2016 = combined_data[(combined_data['Year'] == 2016) &
(combined_data['PGM_TRMNTN_CD'] == 'A')]

# Initialize an empty list to record suspected closures
suspected_closures = []

# Loop over the active hospitals in 2016
for index, hospital in active_2016.iterrows():
    # Get the CMS Certification Number to track across years
    cms_cert_num = hospital['PRVDR_NUM']
    facility_name = hospital['FAC_NAME']
    zip_code = hospital['ZIP_CD']

    # Track if the hospital appears as active in each subsequent year
    active_in_later_years = False
    closure_year = None

    # Check 2017, 2018, and 2019
    for year in [2017, 2018, 2019]:
        # Filter for this hospital in the given year
        hospital_year_data = combined_data[(combined_data['Year'] == year) &
(combined_data['PRVDR_NUM'] == cms_cert_num)]

        if hospital_year_data.empty:
            # If hospital does not appear at all, record the closure year
            closure_year = year
            break
        elif not any(hospital_year_data['PGM_TRMNTN_CD'] == 'A'):
            # If hospital appears but is not active, record the closure year
            closure_year = year
            break
        else:
```

```
        # If hospital is active, continue to next year
        active_in_later_years = True

    # If hospital was not active through 2019, add to suspected closures
    if not active_in_later_years:
        suspected_closures.append({
            'Facility Name': facility_name,
            'ZIP Code': zip_code,
            'Year of Suspected Closure': closure_year
        })

# Convert suspected closures to a DataFrame for easy viewing and analysis
closures_df = pd.DataFrame(suspected_closures)

# Output the number of suspected closures
num_closures = len(closures_df)
print(f"Number of hospitals suspected to have closed by 2019: {num_closures}")
closures_df
```

2.

```{python}
sorted_closures_df = closures_df.sort_values(by='FAC_NAME')

# Display the names and year of suspected closure for the first 10 rows
first_10_closures = sorted_closures_df[['FAC_NAME', 'Year of Suspected Closure']].head(10)
first_10_closures
```

3.

a.

```{python}
# Step 1: Count the number of active hospitals in each ZIP code per year
active_hospitals_per_zip_year = all_years_data[all_years_data['PGM_TRMNTN_CD'] == 'A'] \
                    .groupby(['ZIP_CD', 'Year'])['PRVDR_NUM'] \
                    .nunique().reset_index(name='Active_Hospital_Count')

# Step 2: Merge the suspected closures with the active hospital counts to get the count for the year of suspected closure and the following year
closures_with_counts = closures_df.merge(
    active_hospitals_per_zip_year,
    left_on=['ZIP_CD', 'Year of Suspected Closure'],
    right_on=['ZIP_CD', 'Year'],
    how='left'
).rename(columns={'Active_Hospital_Count': 'Closure_Year_Count'})

closures_with_counts = closures_with_counts.merge(
    active_hospitals_per_zip_year,
    left_on=['ZIP_CD', 'Year of Suspected Closure + 1'],
    right_on=['ZIP_CD', 'Year'],
    how='left'
).rename(columns={'Active_Hospital_Count': 'Following_Year_Count'})

# Step 3: Filter suspected closures where the count of active hospitals does not decrease in the following year
potential_mergers = closures_with_counts[
    (closures_with_counts['Following_Year_Count'] >= closures_with_counts['Closure_Year_Count'])
]
```

Output the number of potential mergers

num_potential_mergers = potential_mergers.shape[0]

num_potential_mergers

```

b.

```{python}

# Step 1: Remove potential mergers from the suspected closures

corrected_closures =
closures_df[~closures_df['PRVDR_NUM'].isin(potential_mergers['PRVDR_NUM'])]

# Step 2: Count the remaining hospitals in the corrected closures list

num_corrected_closures = corrected_closures.shape[0]

num_corrected_closures

```

c.

```{python}

# Step 1: Sort the corrected closures DataFrame by 'Facility Name'

sorted_corrected_closures = corrected_closures.sort_values(by='FAC_NAME')

# Step 2: Display the first 10 rows of the sorted DataFrame

first_10_corrected_closures = sorted_corrected_closures[['FAC_NAME', 'Year of Suspected
Closure']].head(10)

# Display the results

first_10_corrected_closures

```
```

Calculate zip code’s distance to the nearest hospital (20 pts) (*)

1.

```{python}
import geopandas as gpd

# Load the national zip code shapefile (assuming the same zip file as before)
national_zips_filepath = "C:/Users/Shreya Work/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"
national_zips_df = gpd.read_file(national_zips_filepath)

# Calculate centroids for each zip code geometry
national_zips_df['centroid'] = national_zips_df.geometry.centroid

# Create a new GeoDataFrame for the centroids
zips_all_centroids = gpd.GeoDataFrame(national_zips_df[['ZCTA5', 'centroid']])
zips_all_centroids = zips_all_centroids.set_geometry('centroid')

# Rename columns for clarity
zips_all_centroids.columns = ['ZIP_Code', 'geometry']

# Display the dimensions of the GeoDataFrame
dimensions = zips_all_centroids.shape
print(f"Dimensions of zips_all_centroids GeoDataFrame: {dimensions}")

# Display the first few rows of the GeoDataFrame to understand the columns
print(zips_all_centroids.head())
```

2.

```{python}

import geopandas as gpd


# Create a list of zip code prefixes for Texas and its bordering states

texas_prefixes = ['75']

border_states_prefixes = ['73', '74', '76', '77', '78', '79']  # Oklahoma, New Mexico, Arkansas, Louisiana


# Create GeoDataFrame for Texas zip codes

zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZIP_Code'].str.startswith(tuple(texas_prefixes))]


# Create GeoDataFrame for Texas and bordering states zip codes

border_states_prefixes_combined = texas_prefixes + border_states_prefixes

zips_texas_borderstates_centroids = zips_all_centroids[zips_all_centroids['ZIP_Code'].str.startswith(tuple(border_states_prefixes_combined))]


# Count unique zip codes in each subset

unique_texas_zip_codes = zips_texas_centroids['ZIP_Code'].nunique()

unique_borderstates_zip_codes = zips_texas_borderstates_centroids['ZIP_Code'].nunique()


# Display results

print(f"Number of unique zip codes in Texas: {unique_texas_zip_codes}")

print(f"Number of unique zip codes in Texas or a bordering state: {unique_borderstates_zip_codes}")


```


3.


```{python}
```

```
import geopandas as gpd

import pandas as pd

import matplotlib.pyplot as plt


# Load the zip code shapefile for national ZIP codes

filepath = "C:/Users/Shreya
Work/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"

zips_all_centroids = gpd.read_file(filepath)


# Create a GeoDataFrame for Texas zip codes

zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(('75', '76', '77',
'78', '79'))]


# Create a GeoDataFrame for zip codes in Texas or bordering states

border_states = ['Oklahoma', 'Arkansas', 'Louisiana', 'New Mexico']  # Include bordering states

# You will need a way to filter these based on their zip prefixes

zips_texas_borderstates_centroids =
zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(('75', '76', '77', '78', '79', '73', '71', '70',
'88'))]


# Assuming hosp_per_zip is already defined from previous analysis

# hosp_per_zip should look like:

# hosp_per_zip = hosp_tex_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')


# Ensure hosp_per_zip has the correct column for merging

hosp_per_zip = hosp_per_zip.rename(columns={'ZIP_CD': 'ZCTA5'})


# Merge to find ZIP codes in Texas border states with at least 1 hospital

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(

    hosp_per_zip,

    how='inner',  # Use inner merge to keep only those with at least 1 hospital
```

```
    on='ZCTA5'  # Merging on the 'ZCTA5' column which represents ZIP codes
)

# Display the resulting GeoDataFrame
print(zips_withhospital_centroids.head())
print(f"Unique ZIP Codes with Hospitals: {zips_withhospital_centroids['ZCTA5'].nunique()}")
```

4.

a.

```{python}
import geopandas as gpd
import pandas as pd
import time

# Load the zip code shapefile for Texas ZIP codes
filepath = "C:/Users/Shreya Work/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"
zips_texas_centroids = gpd.read_file(filepath)
zips_texas_centroids = zips_texas_centroids[zips_texas_centroids['ZCTA5'].str.startswith(('75', '76', '77', '78', '79'))]

# Subset to 10 zip codes for testing
subset_zips_texas = zips_texas_centroids.head(10)

# Ensure that the coordinate reference system (CRS) is the same for both GeoDataFrames
subset_zips_texas = subset_zips_texas.to_crs(epsg=4326)  # Assuming your zips_withhospital_centroids is in EPSG:4326
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)
```

```
# Start timing the distance calculation
start_time = time.time()

# Calculate distances to the nearest zip code with at least one hospital
distances = subset_zips_texas.geometry.apply(lambda x:
zips_withhospital_centroids.geometry.distance(x).min())

# End timing
end_time = time.time()

# Display the results
subset_zips_texas['Distance to Nearest Hospital'] = distances
print(subset_zips_texas[['ZCTA5', 'Distance to Nearest Hospital']])

# Calculate time taken
time_taken = end_time - start_time
print(f"Time taken for the join with 10 ZIP codes: {time_taken:.2f} seconds")

# Estimate total time for the entire procedure
total_zip_codes = len(zips_texas_centroids)
estimated_time = (time_taken / len(subset_zips_texas)) * total_zip_codes
print(f"Estimated total time for the entire procedure: {estimated_time:.2f} seconds")
```

b.

```{python}
import geopandas as gpd
import pandas as pd
import time
```

```
# Load the zip code shapefile for all ZIP codes in the U.S.

filepath = "C:/Users/Shreya Work/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"


zips_all_centroids = gpd.read_file(filepath)


# Create a subset of Texas zip codes

zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(('75', '76', '77', '78', '79'))]


# Create a subset of border states (example: Louisiana, Arkansas, Oklahoma, New Mexico)

border_states_prefixes = ['70', '71', '72', '73', '74', '73', '74', '87']

zips_texas_borderstates_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(tuple(border_states_prefixes))]


# Load the cleaned POS data (make sure this DataFrame contains hospital information)

pos2016 = pd.read_csv('path_to_your_pos_data.csv')  # Load your cleaned POS data


# Create zips_withhospital_centroids

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(

    pos2016.groupby('ZIP_CD').size().reset_index(name='hospital_count'),

    how='inner',

    left_on='ZCTA5',  # This assumes 'ZCTA5' corresponds to ZIP_CD in pos2016

    right_on='ZIP_CD'

)


# Ensure that the coordinate reference system (CRS) is the same for both GeoDataFrames

zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)  # Assuming EPSG:4326 is your CRS

zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)


# Start timing the full distance calculation
```

```
start_time_full = time.time()

# Calculate distances to the nearest zip code with at least one hospital
distances_full = zips_texas_centroids.geometry.apply(lambda x:
zips_withhospital_centroids.geometry.distance(x).min())

# End timing
end_time_full = time.time()

# Add distances to the GeoDataFrame
zips_texas_centroids['Distance to Nearest Hospital'] = distances_full

# Display the first few results
print(zips_texas_centroids[['ZCTA5', 'Distance to Nearest Hospital']].head())

# Calculate time taken for the full calculation
time_taken_full = end_time_full - start_time_full
print(f"Time taken for the full calculation: {time_taken_full:.2f} seconds")

# Optionally, compare with estimated time
total_zip_codes = len(zips_texas_centroids)
estimated_time_full = (time_taken / len(subset_zips_texas)) * total_zip_codes
print(f"Estimated total time for the entire procedure: {estimated_time_full:.2f} seconds")
```

c.

```{python}
# Specify the path to the .prj file
prj_file_path = 'C:/Users/Shreya
Work/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.prj'
```

```
# Read the contents of the .prj file
with open(prj_file_path, 'r') as file:
    prj_contents = file.read()

print(prj_contents)
```

5.

a.

```{python}
# Calculate the average distance to the nearest hospital for each zip code
average_distances = zips_texas_centroids.groupby('ZIP_CD')['distance_to_nearest_hospital'].mean().reset_index()

# Rename the column for clarity
average_distances.columns = ['ZIP_CD', 'Average_Distance_to_Hospital']

# Display the average distances
print(average_distances)
```

b.

```{python}
# Convert average distances from meters to miles
average_distances['Average_Distance_to_Hospital_miles'] = average_distances['Average_Distance_to_Hospital'] / 1609.34
```

```
# Display the average distance in miles
print(average_distances[['ZIP_CD', 'Average_Distance_to_Hospital_miles']])
```

```
```

c.

```{python}
import geopandas as gpd
import matplotlib.pyplot as plt

# Merge the average distances into the GeoDataFrame
zips_texas_centroids = zips_texas_centroids.merge(average_distances[['ZIP_CD',
'Average_Distance_to_Hospital_miles']],
                                                  left_on='ZCTA5',
                                                  right_on='ZIP_CD',
                                                  how='left')

# Plotting the choropleth map
plt.figure(figsize=(12, 10))
zips_texas_centroids.plot(column='Average_Distance_to_Hospital_miles',
                          cmap='OrRd',
                          legend=True,
                          edgecolor='black',
                          missing_kwds={"color": "lightgrey", "label": "Missing values"})

plt.title("Average Distance to Nearest Hospital by Zip Code in Texas", fontsize=16)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.axis('off')  # Turn off the axis
plt.show()
```

```

**## Reflecting on the exercise (10 pts)**

2.

a. Limitations of Current Method:

ZIP Code Focus: May overlook changes in access when multiple hospitals exist in a ZIP code.

Mergers and Transfers: Doesn't account for hospitals that merge or relocate, misrepresenting access.

b. Improvement Suggestions:

Spatial Analysis: Use GIS to analyze distances from residents to hospitals.

Neighborhood-Level Analysis: Assess access changes at a neighborhood level.

Patient Data: Gather demographics to understand community impacts.

Surveys: Collect resident feedback on access changes.

Health Outcomes: Monitor health outcomes after closures.
```