PS4 Andy Fan Will Sigal

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

Style Points (10 pts)

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): Andy Fan, fanx
    - Partner 2 (name and cnet ID):
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:

AF WS

5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: (Andy:0); (Will:0) Late coins left after submission: (Andy: 3) ; (Will: 4)
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.

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

(This file records all providers certified to bill Medicare and Medicaid. Each provider has a unique CMS certification number, consistent across different years)

(The data dictionary is located at the bottom of the page as "Provider of Services File- Hospital & Non-Hospital Facilities Data Dictionary)

```
### SETUP
import pandas as pd
import altair as alt
import time
import os
import warnings
import geopandas as gpd
warnings.filterwarnings('ignore')
```

1. (Partner 1) This is a fairly large dataset and we won't be using most of the variables. Read through the rest of the problem set and look through the data dictionary to identify which variables you will need to complete the exercise, and use the tool on data.cms.gov into restrict to those variables ("Manage Columns") before exporting ("Export"). Download this for 2016 and call it pos2016.csv. What are the variables you pulled?

required variables:

provider type code: PRVDR_CTGRY_CD

subtype code: PRVDR_CTGRY_SBTYP_CD

CMS certification number: PRVDR_NUM

Termination code: PGM_TRMNTN_CD

zip code: ZIP_CD

Date of termination: TRMNTN_EXPRTN_DT

Facility name: FAC_NAME

```
### import data (default directory is just the repo)

#os.chdir('/Users/willsigal/Documents/GitHub/problem-set-4-andy-fan-will-sigal/Data')
 ↪  #will wd

os.chdir('d:\\UChicago\\Classes\\2024Qfall\\Programming
 ↪  Python\\problem-set-4-andy-fan-will-sigal\\Data') #andy wd

df_2016 = pd.read_csv('pos2016.csv', encoding="latin1", on_bad_lines="skip")
```

2. (Partner 1) Import your pos2016.csv file. We want to focus on short-term hospitals. These are identified as facilities with provider type code 01 and subtype code 01. Subset your data to these facilities. How many hospitals are reported in this data? Does this number make sense? Cross-reference with other sources and cite the number you compared it to. If it differs, why do you think it could differ?

```
###subset to short term hositals
df_2016 = df_2016[(df_2016['PRVDR_CTGRY_SBTYP_CD'] == 1) &
 ↪  (df_2016['PRVDR_CTGRY_CD'] == 1)]
```

a.There are 7275 short term hospitals reported in the data. This number at a glance makes sense nationwide.

b.Looking at the Article by Jane Wishner, Patricia Solleveld, Robin Rudowitz, Julia Paradise, and Larisa Antonisse, they found "nearly 5,000 short-term, acute care hospitals in the United States". This is less than the data we have. It is possibly because some of the short-term hospitals in our dataset closed (they are temporary after all, and could have only been setup for less than a year)

3. (Partner 1) Repeat the previous 3 steps with 2017Q4, 2018Q4, and 2019Q4 and then append them together. Plot the number of observations in your dataset by year.

```
### import data
df_2017 = pd.read_csv('pos2017.csv', encoding="latin1", on_bad_lines="skip")
df_2018 = pd.read_csv('pos2018.csv', encoding="latin1", on_bad_lines="skip")
df_2019 = pd.read_csv('pos2019.csv', encoding="latin1", on_bad_lines="skip")
```

```
###subset to short term hositals
df_2017 = df_2017[(df_2017['PRVDR_CTGRY_SBTYP_CD'] == 1) &
 ↪  (df_2017['PRVDR_CTGRY_CD'] == 1)]
df_2018 = df_2018[(df_2018['PRVDR_CTGRY_SBTYP_CD'] == 1) &
 ↪  (df_2018['PRVDR_CTGRY_CD'] == 1)]
df_2019 = df_2019[(df_2019['PRVDR_CTGRY_SBTYP_CD'] == 1) &
 ↪  (df_2019['PRVDR_CTGRY_CD'] == 1)]
```

The number of short-term hospitals in each of the years:

2017: 7260 hospitals

2018: 7277 hospitals

2019: 7303 hospitals

```
### create new year columns
df_2016['year'] = 2016
df_2017['year'] = 2017
df_2018['year'] = 2018
df_2019['year'] = 2019

### append
df = pd.concat([df_2016, df_2017, df_2018, df_2019], ignore_index=True)
```

```
### altair plot
df1_3 = df.groupby('year').size().reset_index(name='count')

chartQ1_3 = alt.Chart(df1_3).mark_bar().encode(
    x= alt.X('year:O', axis=alt.Axis(labelAngle=90)),
    y= alt.Y('count:Q', title='number of hospitals',
 ↪  scale=alt.Scale(domain=[7000, 7500], clamp=True))
).properties(
    width=400,
    height=200,
    title='Observations of Hospitals by Year'
)
chartQ1_3.save("chartQ1_3.png")
chartQ1_3
```

```
alt.Chart(...)
```

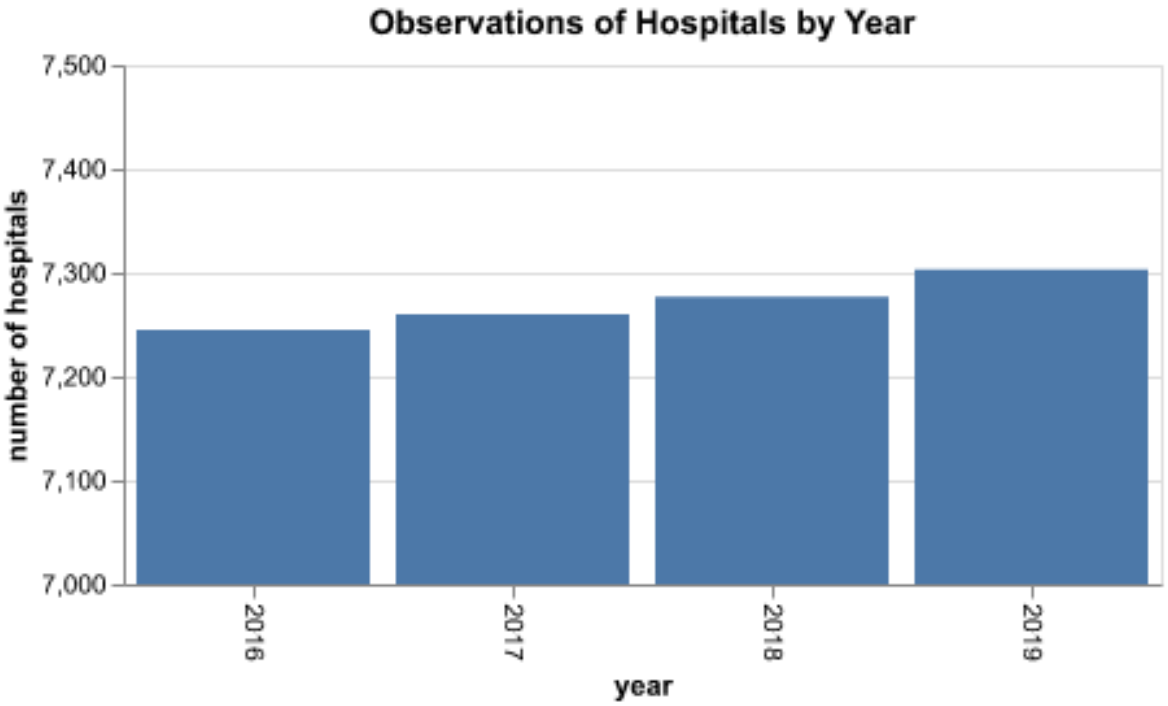


Figure 1: Chart Q1.3

4. (Partner 1) Each hospital is identified by its CMS certification number. Plot the number of unique hospitals in your dataset per year. Compare this to your plot in the previous step. What does this tell you about the structure of the data?

a.plot unique hospitals

```
### plot
df1_4 = df.groupby('year')['PRVDR_NUM'].nunique().reset_index()

chartQ1_4 = alt.Chart(df1_4).mark_bar().encode(
    x= alt.X('year:O', axis=alt.Axis(labelAngle=90)),
    y= alt.Y('PRVDR_NUM:Q', title='number of unique hospitals',
 ↪  scale=alt.Scale(domain=[7000, 7500], clamp=True))
).properties(
    width=400,
    height=200,
    title='Unique Hospitals by Year'
)
```

```
chartQ1_4.save("chartQ1_4.png")
chartQ1_4
```

```
alt.Chart(...)
```

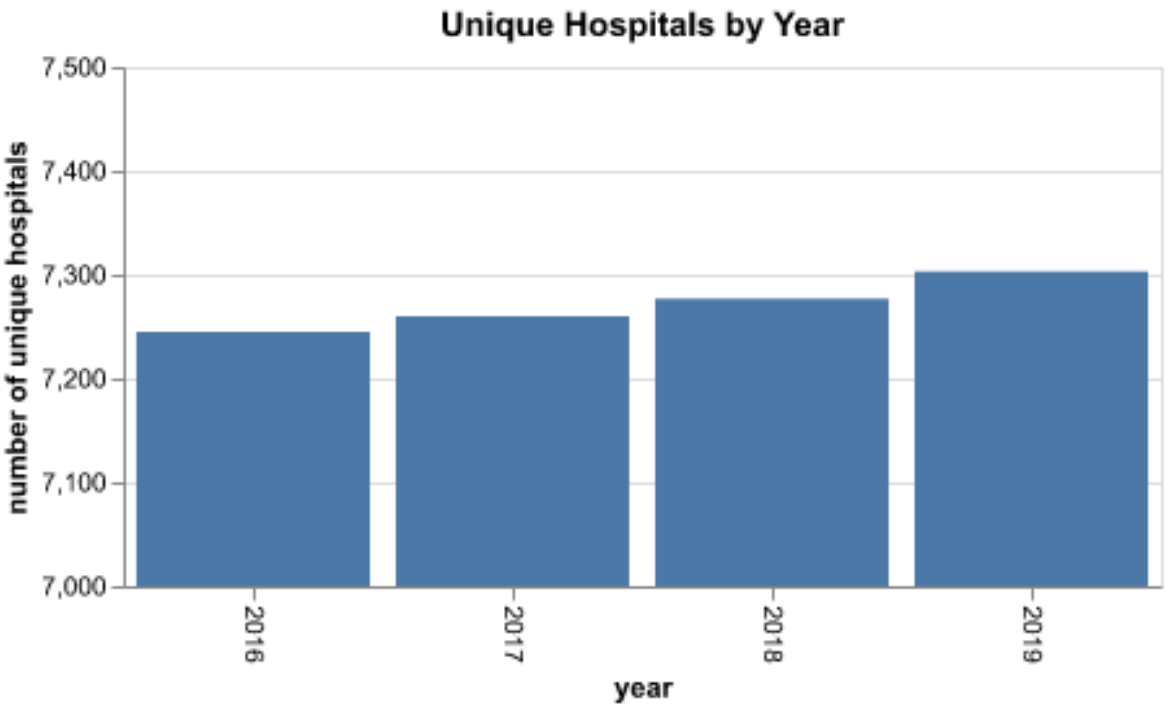


Figure 2: Chart Q1.4

```
b.compare to previous step, what does this tell about data structure:
```

The plots are identical. Which means each hospital is a unique observation, tallied each year.

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

We will now use the 2016-2019 files to identify hospital closures nationally. A hospital issuspected to have closed if its Termination Code in the POS file lists them as an "Active Provider" in 2016 and then they are either not active or do not appear in the data at all in a subsequent year.

1. (Partner 2) Use this definition to create a list of all hospitals that were active in 2016 that were suspected to have closed by 2019. Record the facility name and zip of each hospital as well as the year of suspected closure (when they become terminated or disappear from the data). How many hospitals are there that fit this definition?

```
active_in_2016 = df[(df['year'] == 2016) & (df['PGM_TRMNTN_CD'] ==
↪  0)][['PRVDR_NUM', 'FAC_NAME', 'ZIP_CD']].copy()

last_active_years = df[df['PGM_TRMNTN_CD'] ==
↪  0].groupby('PRVDR_NUM')['year'].max().reset_index(name='last_active_year')

active_in_2016 = active_in_2016.merge(last_active_years, on='PRVDR_NUM',
↪  how='left')

active_hospitals_count = active_in_2016.shape[0]

suspected_closures = active_in_2016[active_in_2016['last_active_year'] <
↪  2019].copy()

closure_count = suspected_closures.shape[0]
suspected_closures[['FAC_NAME', 'ZIP_CD', 'last_active_year']], closure_count

closure_count
```

```
174
```

2. (Partner 2) Sort this list of hospitals by name and report the names and year of suspected closure for the first 10 rows.

```
first_ten_closures =
↪  suspected_closures.sort_values(by='FAC_NAME').head(10)[['FAC_NAME',
↪  'last_active_year']]

first_ten_closures
```

	FAC_NAME	last_active_year
93	ABRAZO MARYVALE CAMPUS	2016
299	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2016
2276	AFFINITY MEDICAL CENTER	2017
2019	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2016

	FAC_NAME	last_active_year
2955	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2016
1682	ALLIANCE LAIRD HOSPITAL	2018
2345	ALLIANCEHEALTH DEACONESS	2018
720	ANNE BATES LEACH EYE HOSPITAL	2018
528	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	2016
1801	BANNER CHURCHILL COMMUNITY HOSPITAL	2016

3. (Partner 2) However, not all suspected hospital closures are true closures. For example, in the case of a merger, a CMS certification number will be appear to be "terminated," but then the hospital re-appear under a similar name/address with a new CMS certifi cation number in the next year. As a first pass to address this, remove any suspected hospital closures that are in zip codes where the number of active hospitals does not decrease in the year after the suspected closure.

a. Among the suspected closures, how many hospitals fit this definition of potentially being a merger/acquisition?

```
active_hospitals_count = df[df['PGM_TRMNTN_CD'] == 0].groupby(['ZIP_CD',
 ↪  'year']).size().reset_index(name='active_count')
zip_year_counts = active_hospitals_count.pivot(index='ZIP_CD',
 ↪  columns='year', values='active_count').fillna(0)

suspected_closures['closure_year_count'] = suspected_closures.apply(
    lambda row: zip_year_counts.at[row['ZIP_CD'], row['last_active_year']] if
     ↪  row['ZIP_CD'] in zip_year_counts.index else 0, axis=1
)
suspected_closures['next_year_count'] = suspected_closures.apply(
    lambda row: zip_year_counts.at[row['ZIP_CD'], row['last_active_year'] +
     ↪  1] if row['ZIP_CD'] in zip_year_counts.index else 0, axis=1
)

potential_mergers =
 ↪  suspected_closures[suspected_closures['closure_year_count'] <=
 ↪  suspected_closures['next_year_count']]

merger_count = potential_mergers.shape[0]

merger_count
```

```
8
```

b. After correcting for this, how many hospitals do you have left?

```
true_closures = suspected_closures[suspected_closures['closure_year_count'] >
 ↪  suspected_closures['next_year_count']]

remaining_closures_count = true_closures.shape[0]

remaining_closures_count
```

```
166
```

c. Sort this list of corrected hospital closures by name and report the first 10 rows.

```
correct_closure_df =
 ↪  true_closures.sort_values(by='FAC_NAME').head(10)[['FAC_NAME', 'ZIP_CD',
 ↪  'last_active_year']]

correct_closure_df.head(10)
```

	FAC_NAME	ZIP_CD	last_active_year
93	ABRAZO MARYVALE CAMPUS	85031.0	2016
299	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230.0	2016
2276	AFFINITY MEDICAL CENTER	44646.0	2017
2019	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	12208.0	2016
2955	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2016
1682	ALLIANCE LAIRD HOSPITAL	39365.0	2018
2345	ALLIANCEHEALTH DEACONESS	73112.0	2018
720	ANNE BATES LEACH EYE HOSPITAL	33136.0	2018
528	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050.0	2016
1801	BANNER CHURCHILL COMMUNITY HOSPITAL	89406.0	2016

Download Census zip code shapefile (10 pt)

Navigate to the Census shapefiles (link), select "gz_2010_us_860_00_500k.zip" and download the resulting shapefile. Note: If you have difficulty downloading or working with the file because the size is too large for your computer, reach out to the instruction team on Ed so that we can give them a smaller initial shapefile to work with.

1. (Partner 1) This is non-tabular data. 1a. What are the five file types and what type of information is in each file? 1b. It will be useful going forward to have a sense going forward of which files are big versus small. After unzipping, how big is each of the datasets?

a.The 5 file types are:
1.dbf file--which is a table that stores attributes of the geometries.
2.prj file--stores the coordinate system information; used by ArcGIS.
3.shp file--The main file that stores the feature geometry (by far the largest, >800mb in this instance)
4.shx file--the index file for the geometries
5.xml file--is metadata for ArcGIS that has info about the shapefile itself

b. The size of the files:
1.dbf file--6275kb
2.prj file--1kb
3.shp file--817,915kb
4.shx file--259kb
5.xml file--16kb

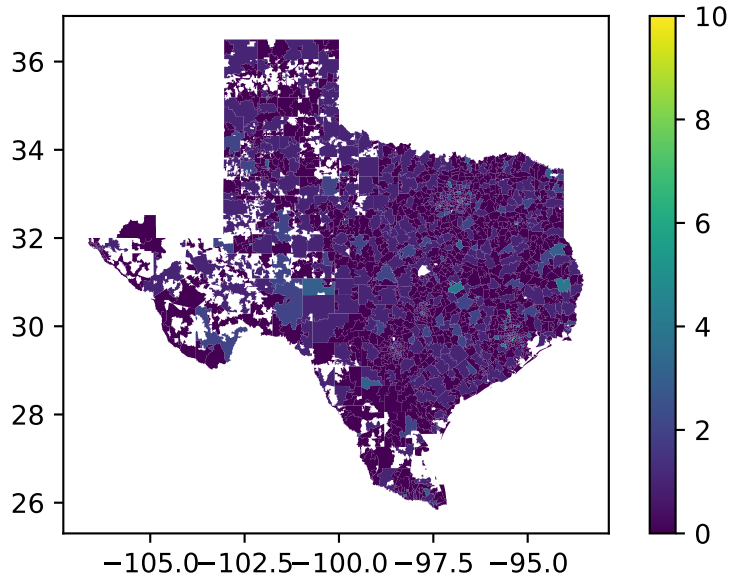
2. (Partner 1) Load the zip code shapefile and restrict to Texas zip codes. (Hint: you can identify which state a zip code is in using the first 2-3 numbers in the zip code (Wikipedia link). Then calculate the number of hospitals per zip code in 2016 based on the cleaned POS file from the previous step. Plot a choropleth of the number of hospitals by zip code in Texas.

```
import geopandas as gpd
#os.chdir('/Users/willsigal/Documents/GitHub/problem-set-4-andy-fan-will-sigal/Shapefiles')
 ↪  #will wd shapefiles

os.chdir('d:\\UChicago\\Classes\\2024Qfall\\Programming
 ↪  Python\\problem-set-4-andy-fan-will-sigal\\Shapefiles') #andy wd
 ↪  shapefiles

zips_all = gpd.read_file('gz_2010_us_860_00_500k.shp')
gpd_texas = pd.DataFrame(zips_all)
#texas zips: 75xxx-79xxx, 885xx, (also 73301 but for IRS only)
gpd_texas =
 ↪  gpd_texas[gpd_texas['ZCTA5'].str.startswith(('75','76','77','78','79','885'))]
```

```
### number of hospitals per zip:
df_2016_tx = df_2016
```

```
df_2016_tx['ZIP_CD'] = df_2016_tx['ZIP_CD'].astype(int)
df_2016_tx['ZIP_CD'] = df_2016_tx['ZIP_CD'].astype(str)
df_2016_tx = df_2016_tx[df_2016_tx['ZIP_CD'].isin(gpd_texas['ZCTA5'])]

tx_zip =
 ↪  df_2016_tx['ZIP_CD'].value_counts().reindex(gpd_texas['ZCTA5']).fillna(0).astype(int)
tx_zip = tx_zip.reset_index()

#asked chatgpt how to find the number of observations for each zip code in
 ↪  df1 using observations from df2
```

```
###Plot a choropleth of the number of hospitals by zip code in Texas.
#append count to gpd
zips_texas =
 ↪  zips_all[zips_all['ZCTA5'].str.startswith(('75','76','77','78','79','885'))]
zips_texas = pd.merge(zips_texas, tx_zip, on='ZCTA5', how='left')

#create area column in km2
zips_texas["area_km2"] = zips_texas.area / 1000000
zips_texas[['ZCTA5', 'count', 'area_km2', 'geometry']].head()

zips_texas.plot(column="count", legend=True)
```



*ZCTA5 is the Census 5 digit zip code tabulation area, and will be considered the zip code as no other variable is avaliable.

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

1. (Partner 2) Create a GeoDataFrame for the centroid of each zip code nationally: zips_all_centroids. What are the dimensions of the resulting GeoDataFrame and what do each of the columns mean?

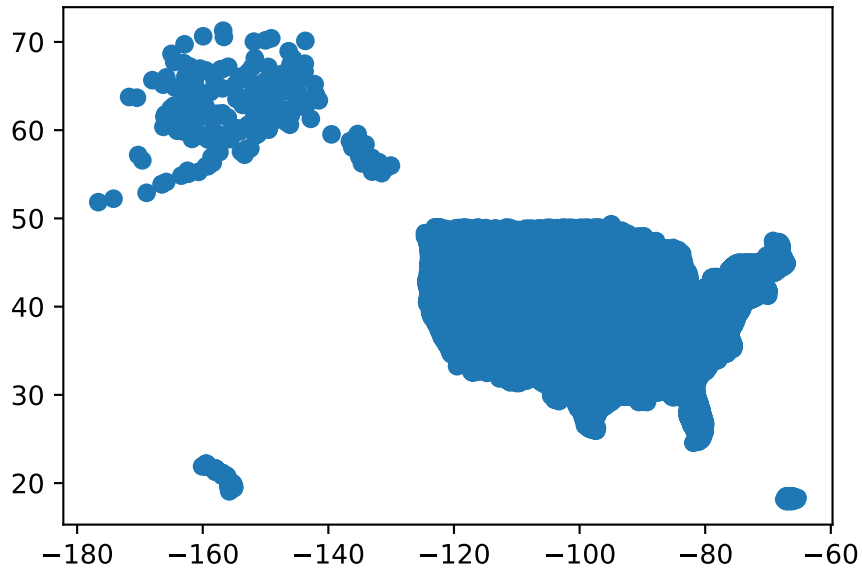
```
zips_all_centroids = zips_all
zips_all_centroids['geometry'] = zips_all['geometry'].centroid
zips_all_centroids.shape
```

```
(33120, 6)
```

```
zips_all_centroids.columns
```

```
Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry'],
dtype='object')
```

```
zips_all_centroids.plot()
```



Geo_id is the unique identifier, ZCTA5 is the xip code, Name is the name of the area, censusarea is the area of the zcta in square kilometers and geometry is the centroid point geomotry for each ZCTA.

2. (Partner 2) Create two GeoDataFrames as subsets of zips_all_centroids. First, create all zip codes in Texas: zips_texas_centroids. Then, create all zip codes in Texas or a bordering state: zips_texas_borderstates_centroids, using the zip code prefixes to make these subsets. How many unique zip codes are in each of these subsets?

```
texas_prefixes = ['75', '76', '77', '78', '79']

border_states_prefixes = {'New Mexico': ['87', '88'],'Oklahoma': ['73',
 ↪  '74'],'Arkansas': ['71', '72'],'Louisiana': ['70', '71']}

zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str[:2].isin(texas_prefixes)]

all_prefixes = texas_prefixes + [prefix for state_prefixes in
 ↪  border_states_prefixes.values() for prefix in state_prefixes]

zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str[:2].isin(all_prefixes)]

texas_unique_zips = zips_texas_centroids['ZCTA5'].nunique()
border_states_unique_zips =
 ↪  zips_texas_borderstates_centroids['ZCTA5'].nunique()


print(f'there are {texas_unique_zips} unique zipcodes in Texas')

print(f'there are {border_states_unique_zips} unique zips in Texas and
 ↪  bordering states')
```

```
there are 1935 unique zipcodes in Texas
there are 4057 unique zips in Texas and bordering states
```

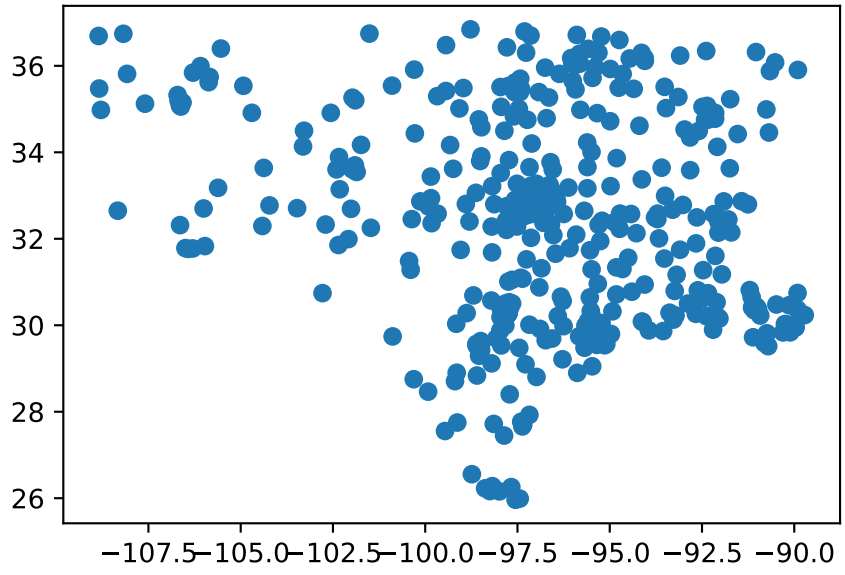
3.(Partner 2) Then create a subset of zips_texas_borderstates_centroids that contains only the zip codes with at least 1 hospital in 2016. Call the resulting GeoDataFrame zips_withhospital_centroids What kind of merge did you decide to do, and what variable are you merging on?

```
zips_texas_borderstates_centroids['ZCTA5'].head
```

```
<bound method NDFrame.head of 8870     70003
8871     70030
```

```
8872     70032
8873     70036
8874     70038
         ...
32917    78261
32918    78368
32919    78412
32920    78557
32921    78586
Name: ZCTA5, Length: 4057, dtype: object>
```

```
hospitals_2016 = df[(df['year'] == 2016) & (df['PGM_TRMNTN_CD'] == 0)]
hospitals_2016['ZIP_CD'] = hospitals_2016['ZIP_CD'].astype(int).astype(str)

zips_texas_borderstates_centroids['ZCTA5'] =
 ↪  zips_texas_borderstates_centroids['ZCTA5']

zip_codes_with_hospitals = hospitals_2016['ZIP_CD'].unique()

zips_withhospital_centroids =
 ↪  zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zip_co

zips_withhospital_centroids.plot()
```



```
len(zips_withhospital_centroids)
```

```
448
```

I decided to first filter for active hospitals in the 2016 df, then I made the zipswithhospitals by filtering the zips_texas_border_state_centroid df zips by whether the zips are in the active hospital zip.

4. (Partner 2) For each zip code in zips_texas_centroids, calculate the distance to the nearest zip code with at least one hospital in zips_withhospital_centroids.

a. This is a computationally-intensive join. Before attempting to do the entire join, subset to 10 zip codes in zips_texas_centroids and try the join. How long did it take? Approximately how long do you estimate the entire procedure will take?

```
import time
from shapely import Point
subset_texas_centroids = zips_texas_centroids.head(10)

start_time = time.time()

distances = []
for _, texas_zip in subset_texas_centroids.iterrows():
    min_distance =
 ↪  zips_withhospital_centroids.distance(texas_zip.geometry).min()
    distances.append(min_distance)


end_time = time.time()

time_taken_subset = end_time - start_time
print(f'Time taken for 10 ZIP codes was {time_taken_subset:.2f} seconds')

full_data_estimate = (time_taken_subset / 10) * len(zips_texas_centroids)

print(f'it will take approx. {full_data_estimate} seconds to join the whole
 ↪  df')
```

```
Time taken for 10 ZIP codes was 0.09 seconds
it will take approx. 16.603901624679565 seconds to join the whole df
```

b. Now try doing the full calculation and time how long it takes. How close is it to your estimation?

```
start_time_full = time.time()

full_distances = []
for _, texas_zip in zips_texas_centroids.iterrows():
    min_distance =
 ↪  zips_withhospital_centroids.distance(texas_zip.geometry).min()
    full_distances.append(min_distance)


end_time_full = time.time()

actual_time_full = end_time_full - start_time_full

print(f'the whole calculation took {actual_time_full} this is very close to
 ↪  my calculation')

zips_texas_centroids['nearest_hospital_distance'] = full_distances
```

```
the whole calculation took 15.393359661102295 this is very close to my
calculation
```

c. Look into the .prj file and report which unit it is in. Convert the given unit to miles, using an appropriate conversion you find online (estimates are okay).

```
with open("gz_2010_us_860_00_500k.prj", 'r') as file:
    prj_text = file.read()

print(prj_text) ## units are in degrees

zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪  zips_texas_centroids['nearest_hospital_distance'] * 69
```

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,29
```

5.(Partner 2) Calculate the average distance to the nearest hospital for each zip code in Texas.

a. What unit is this in

```
#Its in degrees
```

b. Report the average distance in miles. Does this value make sense?

```
average_distance_miles =
 ↪  zips_texas_centroids['nearest_hospital_distance_miles'].mean()

print(f'The average distance is {average_distance_miles:.2f} miles.')
```

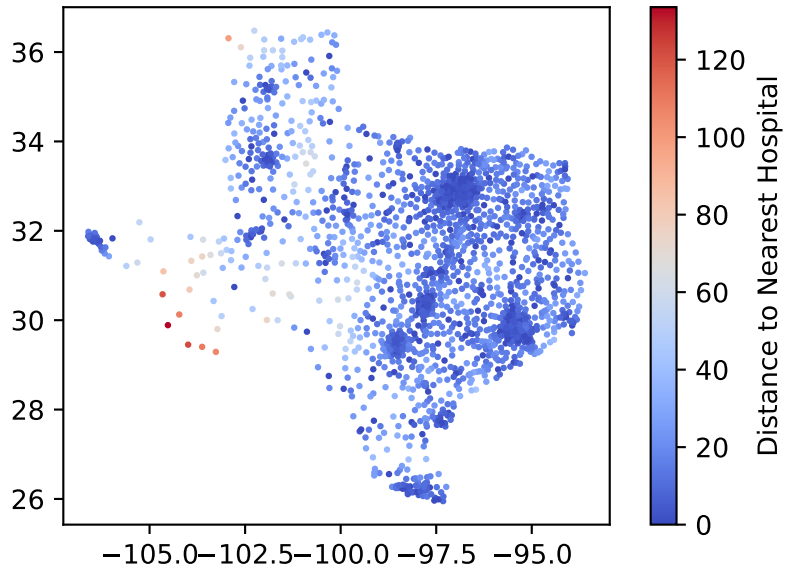
```
The average distance is 14.56 miles.
```

c. Map the value for each zip code

```
zips_texas_centroids['nearest_hospital_distance']
```

```
9207     0.000000
9208     0.167651
9209     0.110844
9210     0.337251
9211     0.219909
           ...
32917    0.110636
32918    0.313242
32919    0.000000
32920    0.058567
32921    0.155733
Name: nearest_hospital_distance, Length: 1935, dtype: float64
```

```
zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪  pd.to_numeric(zips_texas_centroids['nearest_hospital_distance_miles'],
 ↪  errors='coerce')

zips_texas_centroids.plot(
    column='nearest_hospital_distance_miles',
    cmap='coolwarm', markersize = 2,
    legend=True,legend_kwds={'label': "Distance to Nearest Hospital"})
```



Effects of closures on access in Texas (15 pts)

1. (Partner 1) Using the corrected hospital closures dataset from the first section, create a list of directly affected zip codes in Texas– that is, those with at least one closure in 2016-2019. Display a table of the number of zip codes vs. the number of closures they experienced.

```
df5 = df[df['TRMNTN_EXPRTN_DT'].notna()]
df5['ZIP_CD'] = df5['ZIP_CD'].astype(int)
df5['ZIP_CD'] = df5['ZIP_CD'].astype(str)
Q5_1_closures = df5['ZIP_CD'].value_counts().rename("number_closures")
Q5_1_closures
```

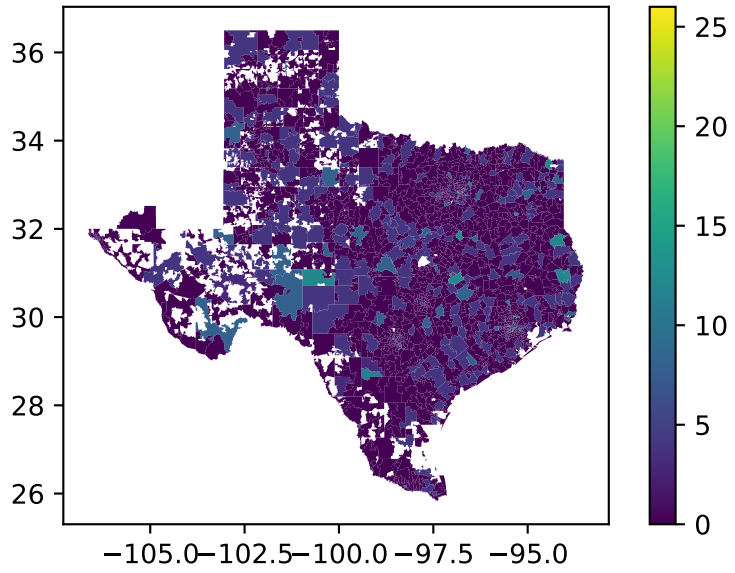
```
ZIP_CD
79902    26
70115    24
71101    24
80218    24
90301    20
         ..
41041     1
70503     1
20707     1
49058     1
```

```
77090     1
Name: number_closures, Length: 3403, dtype: int64
```

2.(Partner 1) Plot a choropleth of which Texas zip codes were directly affected by a closure in 2016-2019– there was at least one closure within the zip code. How many directly affected zip codes are there in Texas?

```
Q5_1_closures = Q5_1_closures.reset_index()
Q5_1_closures.columns = ['ZCTA5', 'number_closures']

zips_texas5_2 = zips_texas
zips_texas5_2 = zips_texas5_2.merge(Q5_1_closures, on='ZCTA5', how='left')
zips_texas5_2['number_closures'] = zips_texas5_2['number_closures'].fillna(0)
print((zips_texas5_2['number_closures'] > 0).sum())

###plot
zips_texas5_2.plot(column="number_closures", legend=True)
```

```
334
```



There are 334 Zip codes in Texas that were affected by at least 1 closure between 2017-2019

3. (Partner 1) Then identify all the indirectly affected zip codes: Texas zip codes within a 10-mile radius of the directly affected zip codes. To do so, first create a GeoDataFrame of the directly affected zip codes. Then create a 10-mile buffer around them. Then, do a spatial join with the overall Texas zip code shapefile. How many indirectly affected zip codes are there in Texas?

```
import matplotlib.pyplot as plt
### gdf of directly affected zips
zips_texas5_3 = zips_texas5_2[zips_texas5_2['number_closures'] > 0]
### create buffer
zips_texas5_3.crs #check distance units.   ### 1 degree latitude =69 miles,
 ↪  therefore 1 mile =  1/69 degrees = 0.01449
zips_texas5_3['geometry'] = zips_texas5_3.geometry.buffer(0.01449)

###join
zips_texas5_3_joined = gpd.sjoin(zips_texas5_2, zips_texas5_3, how='inner')

zips_texas5_3_joined['ZCTA5_left'].nunique()

#asked chatgpt "after conducting a spacial left join, how to find number of
 ↪  overlapping areas, given a buffer on the right geometries"
```

```
1518
```

There are 1518 total affected zip codes, and 1184 directly affected zip codes in Texas.

4.(Partner 1) Make a choropleth plot of the Texas zip codes with a different color for each of the 3 categories: directly affected by a closure, within 10 miles of closure but not directly affected, or not affected.

```
#df_test = pd.DataFrame(zips_texas5_3_joined)
```

Reflecting on the exercise (10 pts)

(Partner 1) The "first-pass" method we're using to address incorrectly identified closures in the data is imperfect. Can you think of some potential issues that could arise still and ways to do a better job at confirming hospital closures?

(Partner 2) Consider the way we are identifying zip codes affected by closures. How well does this reflect changes in zip-code-level access to hospitals? Can you think of some ways to improve this measure?