problem set IV

Jenny Zhong & Summer Negahdar

"This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **SN JZ**

"I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point) Late coins used this pset: **1 this pset** Late coins left after submission: **\0** Knit your `ps4.qmd` to an PDF file to make `ps4.pdf`,

Download and explore the Provider of Services (POS) file

1. PRVDR_CTGRY_SBTYP_CD: Provider subtype for reporting categories.

PRVDR_CTGRY_CD: Provider type in Medicare/Medicaid programs.

PGM_TRMNTN_CD: Provider's current termination status code.

TRMNTN_EXPRTN_DT: Provider termination or certificate expiration date.

FAC_NAME: Certified facility or hospital's official name.

ZIP_CD: Provider's five-digit ZIP code.

CHOW_CNT: Number of provider ownership changes.

CHOW_DT: Date of latest ownership change.

CITY_NAME: Provider's physical location city.

PRVDR_NUM: CMS-certified provider's unique identification number.

2.

```
import pandas as pd
import geopandas as gpd
import shapely
import altair as alt
import sys
import pyproj
import os
from pyproj import CRS, Transformer

file_path = '/Users/samarnegahdar/Desktop/untitled folder/problem-set-4-summer-jenny'

pos2016 = pd.read_csv('pos2016.csv')
```

```
pos2016.head(3)

#converting to string
pos2016['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  pos2016['PRVDR_CTGRY_SBTYP_CD'].astype(str).str.zfill(2)

pos2016['PRVDR_CTGRY_CD'] = pos2016['PRVDR_CTGRY_CD'].astype(str).str.zfill(2)
pos2016.head(3)

st_hospitals2016 = pos2016[(pos2016['PRVDR_CTGRY_SBTYP_CD'] == '1.0') &
 ↪  (pos2016['PRVDR_CTGRY_CD'] == '01')]

print(st_hospitals2016.head(3))
st_hospitals2016.shape
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD
0               NaN  36301.0
1       20010831.0  35740.0
2               NaN  35957.0

(7245, 10)
```

a. 7,245 hospitals are reported in this data.
b. Compared to sources that provide counts of short-term hospitals in the US, such as the Kaiser Family Foundation, we know that there were nearly 5,000 short term, acute care hospitals in the US in recent years (half of which are rural and half urban). This could be higher than the reported number because of definition differences, for example the dataset could define short-term hospitals different to the Kaiser Family Foundation's definition. adding the year 2016

```
st_hospitals2016.loc[:, 'YEAR'] = 2016
print(st_hospitals2016.head(3))
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
```

```
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD  YEAR
0               NaN  36301.0  2016
1        20010831.0  35740.0  2016
2               NaN  35957.0  2016

/var/folders/j5/rv933w1173s068kbzq0kp2xh0000gn/T/ipykernel_9517/3669868829.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  st_hospitals2016.loc[:, 'YEAR'] = 2016
```

3. for 2017:

```
#importing pos2017
pos2017 = pd.read_csv("pos2017.csv", encoding='latin1')
#converting to string
pos2017['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  pos2017['PRVDR_CTGRY_SBTYP_CD'].astype(str).str.zfill(2)

pos2017['PRVDR_CTGRY_CD'] = pos2017['PRVDR_CTGRY_CD'].astype(str).str.zfill(2) ##filling
 ↪  numbers with a zero before integer
pos2017.head(3)
#then focusing on short term hospitals
st_hospitals2017 = pos2017[(pos2017['PRVDR_CTGRY_SBTYP_CD'] == '1.0') &
 ↪  (pos2017['PRVDR_CTGRY_CD'] == '01')]

print(st_hospitals2017.head(3))
print(st_hospitals2017.tail(3))
st_hospitals2017.shape
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD
0               NaN  36301.0
1        20010831.0  35740.0
2               NaN  35957.0
       PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT      CITY_NAME  \
135473                  1.0             01         0      NaN        EL PASO
```

```
135474                  1.0             01         0      NaN          HURST
135475                  1.0             01         0      NaN  THE WOODLANDS

                                                FAC_NAME PRVDR_NUM  \
135473  THE HOSPITALS OF PROVIDENCE TRANSMOUNTAIN CAMPUS    670120
135474                      SAINT CAMILLUS MEDICAL CENTER    670121
135475          HOUSTON METHODIST THE WOODLANDS HOSPITAL    670122

        PGM_TRMNTN_CD  TRMNTN_EXPRTN_DT   ZIP_CD
135473              0               NaN  79911.0
135474              0               NaN  76054.0
135475              0               NaN  77385.0

(7260, 10)
```

Adding the 2017 column

```
st_hospitals2017.loc[:, 'YEAR'] = 2017

print(st_hospitals2017.head(3))
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD  YEAR
0               NaN  36301.0  2017
1        20010831.0  35740.0  2017
2               NaN  35957.0  2017

/var/folders/j5/rv933w1173s068kbzq0kp2xh0000gn/T/ipykernel_9517/196776328.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  st_hospitals2017.loc[:, 'YEAR'] = 2017
```

for 2018:

```
#importing pos2018
pos2018 = pd.read_csv('pos2018.csv', encoding='latin1')
#converting to string
pos2018['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  pos2018['PRVDR_CTGRY_SBTYP_CD'].astype(str).str.zfill(2)
```

```
pos2018['PRVDR_CTGRY_CD'] = pos2018['PRVDR_CTGRY_CD'].astype(str).str.zfill(2)
pos2018.head(3)
#then focus on st hospitals
st_hospitals2018 = pos2018[(pos2018['PRVDR_CTGRY_SBTYP_CD'] == '1.0') &
 ↪  (pos2018['PRVDR_CTGRY_CD'] == '01')]

print(st_hospitals2018.head(3))
print(st_hospitals2018.tail(3))
st_hospitals2018.shape
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD
0               NaN  36301.0
1        20010831.0  35740.0
2               NaN  35957.0
       PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT     CITY_NAME  \
137569                  1.0             01         0      NaN  HORIZON CITY
137570                  1.0             01         0      NaN   SAN ANTONIO
137571                  1.0             01         0      NaN      CROCKETT

                                           FAC_NAME PRVDR_NUM  \
137569  THE HOSPITALS OF PROVIDENCE HORIZON CITY CAMPUS    670124
137570              TEXAS CENTER FOR INFECTIOUS DISEASE    670125
137571                          CROCKETT MEDICAL CENTER    670126

        PGM_TRMNTN_CD  TRMNTN_EXPRTN_DT   ZIP_CD
137569              0               NaN  79928.0
137570              0               NaN  78223.0
137571              0               NaN  75835.0

(7277, 10)
```

Adding the 2018 column

```
#column had to be extended
st_hospitals2018.loc[:, 'YEAR'] = 2018

print(st_hospitals2018.head(3))
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
```

```
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD  YEAR
0               NaN  36301.0  2018
1        20010831.0  35740.0  2018
2               NaN  35957.0  2018

/var/folders/j5/rv933w1173s068kbzq0kp2xh0000gn/T/ipykernel_9517/471435673.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  st_hospitals2018.loc[:, 'YEAR'] = 2018
```

for 2019:

```
#importing pos2019
pos2019 = pd.read_csv('pos2019.csv', encoding='latin1')

#converting to string
pos2019['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  pos2019['PRVDR_CTGRY_SBTYP_CD'].astype(str).str.zfill(2)

pos2019['PRVDR_CTGRY_CD'] = pos2019['PRVDR_CTGRY_CD'].astype(str).str.zfill(2)
pos2019.head(3)
#then focus on st hospitals
st_hospitals2019 = pos2019[(pos2019['PRVDR_CTGRY_SBTYP_CD'] == '1.0') &
 ↪  (pos2019['PRVDR_CTGRY_CD'] == '01')]

print(st_hospitals2019.head(3))
print(st_hospitals2019.tail(3))
st_hospitals2019.shape
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                              FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0       SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1                 NORTH JACKSON HOSPITAL    010004              1
2  MARSHALL MEDICAL CENTERS SOUTH CAMPUS    010005              0
```

```
   TRMNTN_EXPRTN_DT   ZIP_CD
0               NaN  36301.0
1        20010831.0  35740.0
2               NaN  35957.0
       PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT     CITY_NAME  \
139516                  1.0             01         0      NaN  PFLUGERVILLE
139517                  1.0             01         0      NaN       HOUSTON
139518                  1.0             01         0      NaN   SAN ANTONIO

                                            FAC_NAME PRVDR_NUM  \
139516  BAYLOR SCOTT & WHITE MEDICAL CENTER   PFLUGERV...    670128
139517                              THE HEIGHTS HOSPITAL    670129
139518                               SOUTHCROSS HOSPITAL    670130

        PGM_TRMNTN_CD  TRMNTN_EXPRTN_DT   ZIP_CD
139516              0               NaN  78660.0
139517              0               NaN  77008.0
139518              0               NaN  78222.0

(7303, 10)
```

Adding the 2019 column

```
st_hospitals2019.loc[:, 'YEAR'] = 2019

print(st_hospitals2019.head(3))
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                             FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0       SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1                 NORTH JACKSON HOSPITAL    010004              1
2  MARSHALL MEDICAL CENTERS SOUTH CAMPUS    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD  YEAR
0               NaN  36301.0  2019
1        20010831.0  35740.0  2019
2               NaN  35957.0  2019

/var/folders/j5/rv933w1173s068kbzq0kp2xh0000gn/T/ipykernel_9517/1295493316.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  st_hospitals2019.loc[:, 'YEAR'] = 2019
```

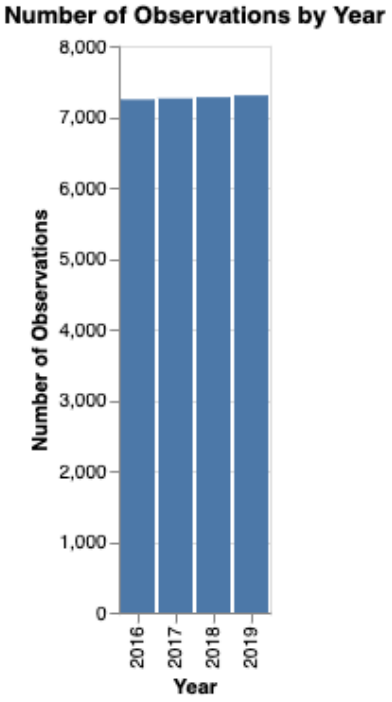
Appending them together

```
all_years_data = pd.concat([st_hospitals2016, st_hospitals2017, st_hospitals2018,
 ↪  st_hospitals2019], ignore_index=True)

print(all_years_data.head(3))
all_years_data.shape
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD  CHOW_CNT  CHOW_DT   CITY_NAME  \
0                  1.0             01         1      NaN      DOTHAN
1                  1.0             01         0      NaN  BRIDGEPORT
2                  1.0             01         0      NaN        BOAZ

                          FAC_NAME PRVDR_NUM  PGM_TRMNTN_CD  \
0  SOUTHEAST ALABAMA MEDICAL CENTER    010001              0
1            NORTH JACKSON HOSPITAL    010004              1
2     MARSHALL MEDICAL CENTER SOUTH    010005              0

   TRMNTN_EXPRTN_DT   ZIP_CD  YEAR
0               NaN  36301.0  2016
1        20010831.0  35740.0  2016
2               NaN  35957.0  2016

(29085, 11)
```

Plotting by observations

```
observations_by_year = all_years_data.groupby('YEAR').size().reset_index(name='count')

chart13 = alt.Chart(observations_by_year).mark_bar().encode(
    x=alt.X('YEAR:O', title='Year'),
    y=alt.Y('count:Q', title='Number of Observations')
).properties(
    title='Number of Observations by Year'
)
```



4. a.

```
unique_hospitals_by_year =
 ↪  all_years_data.groupby('YEAR')['PRVDR_NUM'].nunique().reset_index(name='unique_count')

unique_hospital_chart = alt.Chart(unique_hospitals_by_year).mark_bar().encode(
    x=alt.X('YEAR:O', title='Year'),
    y=alt.Y('unique_count:Q', title='Number of Unique Hospitals')
).properties(
    title='Number of Unique Hospitals by Year'
)
unique_hospital_chart.display()
```

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

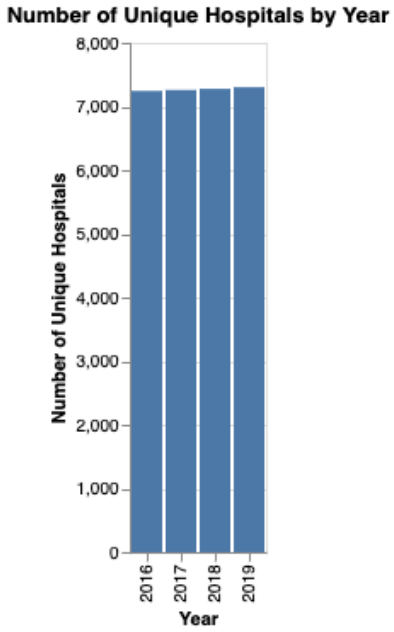


Figure 1: unique hospitals

b. The number of unique hospitals and total observations charts are identical, which means that each hospital has only one record in the dataset for each year. The dataset is structured as a snapshot of unique hospitals, with no intra-year variations or multiple records per hospital within each year.

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

```
unique_termination_codes = all_years_data['PGM_TRMNTN_CD'].unique()
print(unique_termination_codes)
```

```
[0 1 7 4 6 5 3 2]
```

1.

```
all_years_data['PGM_TRMNTN_CD'] = all_years_data['PGM_TRMNTN_CD'].astype(str)
all_years_data['PGM_TRMNTN_CD'] = all_years_data['PGM_TRMNTN_CD'].fillna('1')
print("Check for missing values in PGM_TRMNTN_CD column after filling:")
print(all_years_data['PGM_TRMNTN_CD'].isnull().sum())
suspected_closures = []
active_2016 = all_years_data[(all_years_data['YEAR'] == 2016) &
 ↪  (all_years_data['PGM_TRMNTN_CD'] == '0')]

for _, hospital in active_2016.iterrows():
    facility_name = hospital['FAC_NAME']
    zip_code = hospital['ZIP_CD']

    closed_2019 = all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
```

```
        (all_years_data['YEAR'] == 2019) &
        (all_years_data['PGM_TRMNTN_CD'].isin(['1', '2', '3', '4']))
    ]

    termination_code_2018 = all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
        (all_years_data['YEAR'] == 2018)
    ]['PGM_TRMNTN_CD'].iloc[0] if not all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
        (all_years_data['YEAR'] == 2018)
    ].empty else None

    termination_code_2017 = all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
        (all_years_data['YEAR'] == 2017)
    ]['PGM_TRMNTN_CD'].iloc[0] if not all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
        (all_years_data['YEAR'] == 2017)
    ].empty else None

    if not closed_2019.empty:
        suspected_closures.append({
            'FAC_NAME': facility_name,
            'ZIP_CD': zip_code,
            'year_of_closure': 2019,
            'PGM_TRMNTN_CD_2018': termination_code_2018,
            'PGM_TRMNTN_CD_2017': termination_code_2017
        })

suspected_closures_df =
 ↪  pd.DataFrame(suspected_closures).drop_duplicates(subset=['FAC_NAME', 'ZIP_CD'])

num_unique_suspected_closures = len(suspected_closures_df)
print(f"Number of unique hospitals suspected to have closed by 2019:
 ↪  {num_unique_suspected_closures}")
```

```
Check for missing values in PGM_TRMNTN_CD column after filling:
0
Number of unique hospitals suspected to have closed by 2019: 145
```

2.

```
print(suspected_closures_df.head(3))
```

```
                  FAC_NAME   ZIP_CD  year_of_closure PGM_TRMNTN_CD_2018  \
0  GEORGIANA MEDICAL CENTER  36033.0             2019                   0
```

```
1          RMC JACKSONVILLE  36265.0             2019                  1
2             UAB HIGHLANDS  35205.0             2019                  1

  PGM_TRMNTN_CD_2017
0                  0
1                  0
2                  1
```

3. a.

```
properly_closed_hospitals = []
merge_hospitals = []

for _, hospital in suspected_closures_df.iterrows():
    facility_name = hospital['FAC_NAME']
    zip_code = hospital['ZIP_CD']
    termination_code_2018 = hospital['PGM_TRMNTN_CD_2018']
    termination_code_2017 = hospital['PGM_TRMNTN_CD_2017']

    if termination_code_2018 != '0':
        properly_closed_hospitals.append({
            'FAC_NAME': facility_name,
            'ZIP_CD': zip_code,
            'PGM_TRMNTN_CD_2018': termination_code_2018,
            'PGM_TRMNTN_CD_2017': termination_code_2017
        })
    else:
        if termination_code_2017 == '0':
            properly_closed_hospitals.append({
                'FAC_NAME': facility_name,
                'ZIP_CD': zip_code,
                'PGM_TRMNTN_CD_2018': termination_code_2018,
                'PGM_TRMNTN_CD_2017': termination_code_2017
            })
        else:
            merge_hospitals.append({
                'FAC_NAME': facility_name,
                'ZIP_CD': zip_code,
                'PGM_TRMNTN_CD_2018': termination_code_2018,
                'PGM_TRMNTN_CD_2017': termination_code_2017
            })
properly_closed_hospitals_df = pd.DataFrame(properly_closed_hospitals)
merge_hospitals_df = pd.DataFrame(merge_hospitals)

print("Properly closed hospitals:")
print(properly_closed_hospitals_df.head(3))

print("\nHospitals suspected of merger:")
print(merge_hospitals_df.head(3))

num_properly_closed = properly_closed_hospitals_df.shape[0]
```

```
print(f"Number of properly closed hospitals: {num_properly_closed}")

num_suspected_mergers = merge_hospitals_df.shape[0]
print(f"Number of suspected mergers: {num_suspected_mergers}")
```

```
Properly closed hospitals:
                  FAC_NAME   ZIP_CD PGM_TRMNTN_CD_2018 PGM_TRMNTN_CD_2017
0  GEORGIANA MEDICAL CENTER  36033.0                 0                 0
1          RMC JACKSONVILLE  36265.0                 1                 0
2             UAB HIGHLANDS  35205.0                 1                 1

Hospitals suspected of merger:
Empty DataFrame
Columns: []
Index: []
Number of properly closed hospitals: 145
Number of suspected mergers: 0
```

```
properly_closed_hospitals = []
merge_hospitals = []

for _, hospital in suspected_closures_df.iterrows():
    facility_name = hospital['FAC_NAME']
    zip_code = hospital['ZIP_CD']

    hospital_data = all_years_data[
        (all_years_data['FAC_NAME'] == facility_name) &
        (all_years_data['ZIP_CD'] == zip_code) &
        (all_years_data['YEAR'].isin([2017, 2018]))
    ]

    termination_2018 = hospital_data[hospital_data['YEAR'] ==
↪   2018]['PGM_TRMNTN_CD'].iloc[0] if not hospital_data[hospital_data['YEAR'] ==
↪   2018].empty else '1'
    termination_2017 = hospital_data[hospital_data['YEAR'] ==
↪   2017]['PGM_TRMNTN_CD'].iloc[0] if not hospital_data[hospital_data['YEAR'] ==
↪   2017].empty else '1'

    if termination_2018 != '0':
        properly_closed_hospitals.append({
            'FAC_NAME': facility_name,
            'ZIP_CD': zip_code,
            'PGM_TRMNTN_CD_2017': termination_2017,
            'PGM_TRMNTN_CD_2018': termination_2018
        })
    elif termination_2018 == '0':
        if termination_2017 == '0':
            properly_closed_hospitals.append({
                'FAC_NAME': facility_name,
                'ZIP_CD': zip_code,
```

```
                'PGM_TRMNTN_CD_2017': termination_2017,
                'PGM_TRMNTN_CD_2018': termination_2018
            })
        else:
            # Otherwise, add to suspected mergers
            merge_hospitals.append({
                'FAC_NAME': facility_name,
                'ZIP_CD': zip_code,
                'PGM_TRMNTN_CD_2017': termination_2017,
                'PGM_TRMNTN_CD_2018': termination_2018
            })

# Convert results to DataFrames for inspection
properly_closed_hospitals_df = pd.DataFrame(properly_closed_hospitals)
merge_hospitals_df = pd.DataFrame(merge_hospitals)
# Number of properly closed hospitals
num_properly_closed = properly_closed_hospitals_df.shape[0]
print(f"Number of properly closed hospitals: {num_properly_closed}")

# Number of suspected mergers
num_suspected_mergers = merge_hospitals_df.shape[0]
print(f"Number of suspected mergers: {num_suspected_mergers}")
```

```
Number of properly closed hospitals: 145
Number of suspected mergers: 0
```

There are no potential merged hospitals! a. there are no merged hospitals! b.

```
# Sort the properly closed hospitals by FAC_NAME and display the first 10 rows
```

sorted_properly_closed_hospitals = properly_closed_hospitals_df.sort_values(by='FAC_NAME').head(3)

Display the sorted list

print("First 10 corrected hospital closures sorted by name:") print(sorted_properly_closed_hospitals.head(3))

```
## Download Census zip code shapefile (10 pt)

1.
    a. There are five types of file types here.
**.shp (shape file)**: file contains the geometric shapes. It stores the spatial
coordinates and shapes of objects like points, lines or polygons.
**.shx (shape index format)**: file is an index of the geometry in the .shp file, and
provides quick access to the geometric shapes.
**.dbf (attribute format)**: file contains attribute data in tabular format, with each
row corresponding to a feature and each column containing attributes associated with that
feature.
```

.prj (projection format): file contains information about the coordinate sysetm and projection, and defines how shapes are mapped onto Earth's surface.
.xml (metadata format): file contains metadata, contains descriptive information about dataset such as source, creation date, projection dates.

    b.

File Extension	Size (MB)
.shp	837.5
.dbf	6.4
.prj	0.000165
.shx	0.000265
.xml	0.016

2.

```{python}
import geopandas as gpd
```

```
shapefile_path = '/Users/samarnegahdar/Desktop/untitled
 ↪  folder/problem-set-4-summer-jenny/shapefiles/gz_2010_us_860_00_500k.shp'

try:
    zip_shapes = gpd.read_file(shapefile_path)
    print("Shapefile loaded successfully with SHX restoration.")
except Exception as e:
    print("Error loading shapefile:", e)
```

```
#filtering for texas zip codes
texas_zip_shapes = zip_shapes[zip_shapes['ZCTA5'].astype(str).str.startswith(('75', '76',
 ↪  '77', '78', '79')) | (zip_shapes['ZCTA5'] == '733')]

print(texas_zip_shapes.head(3))
print("Number of Texas ZIP codes:", texas_zip_shapes.shape[0])
```

```
#pos2016
pos2016['ZIP_CD'] = pos2016['ZIP_CD'].astype(str)

texas_hospitals_2016 = pos2016[pos2016['ZIP_CD'].str.startswith(('75', '76', '77', '78',
 ↪  '79')) | (pos2016['ZIP_CD'] == '733')]

print(texas_hospitals_2016.head(3))
print("Number of Texas hospital records in 2016:", texas_hospitals_2016.shape[0])

hospitals_per_zip =
 ↪  texas_hospitals_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')
```

```
texas_zip_shapes['ZCTA5'] = texas_zip_shapes['ZCTA5'].astype(str)
hospitals_per_zip['ZIP_CD'] =
 ↪  hospitals_per_zip['ZIP_CD'].astype(float).astype(int).astype(str).str.zfill(5)
print("Formatted ZIP codes in hospitals_per_zip:",
 ↪  hospitals_per_zip['ZIP_CD'].unique()[:10])
hospitals_per_zip = hospitals_per_zip[['ZIP_CD', 'hospital_count']]
hospitals_per_zip = hospitals_per_zip.rename(columns={'ZIP_CD': 'ZIP_CD_hp',
 ↪  'hospital_count': 'hospital_count_hp'})
texas_zip_shapes = texas_zip_shapes.merge(hospitals_per_zip, left_on='ZCTA5',
 ↪  right_on='ZIP_CD_hp', how='left')
texas_zip_shapes['hospital_count_hp'] = texas_zip_shapes['hospital_count_hp'].fillna(0)
print("Number of ZIP codes with hospital_count as 0:",
 ↪  (texas_zip_shapes['hospital_count_hp'] == 0).sum())
print("Total ZIP codes in Texas:", texas_zip_shapes.shape[0])
print(texas_zip_shapes[['ZCTA5', 'hospital_count_hp']].head(3))
```

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(12, 10))
texas_zip_shapes.plot(column='hospital_count_hp', cmap='OrRd', linewidth=0.8, ax=ax,
 ↪  edgecolor='0.8', legend=True)
ax.set_title("Number of Hospitals per ZIP Code in Texas (2016)")
ax.set_axis_off()
plt.show()
```

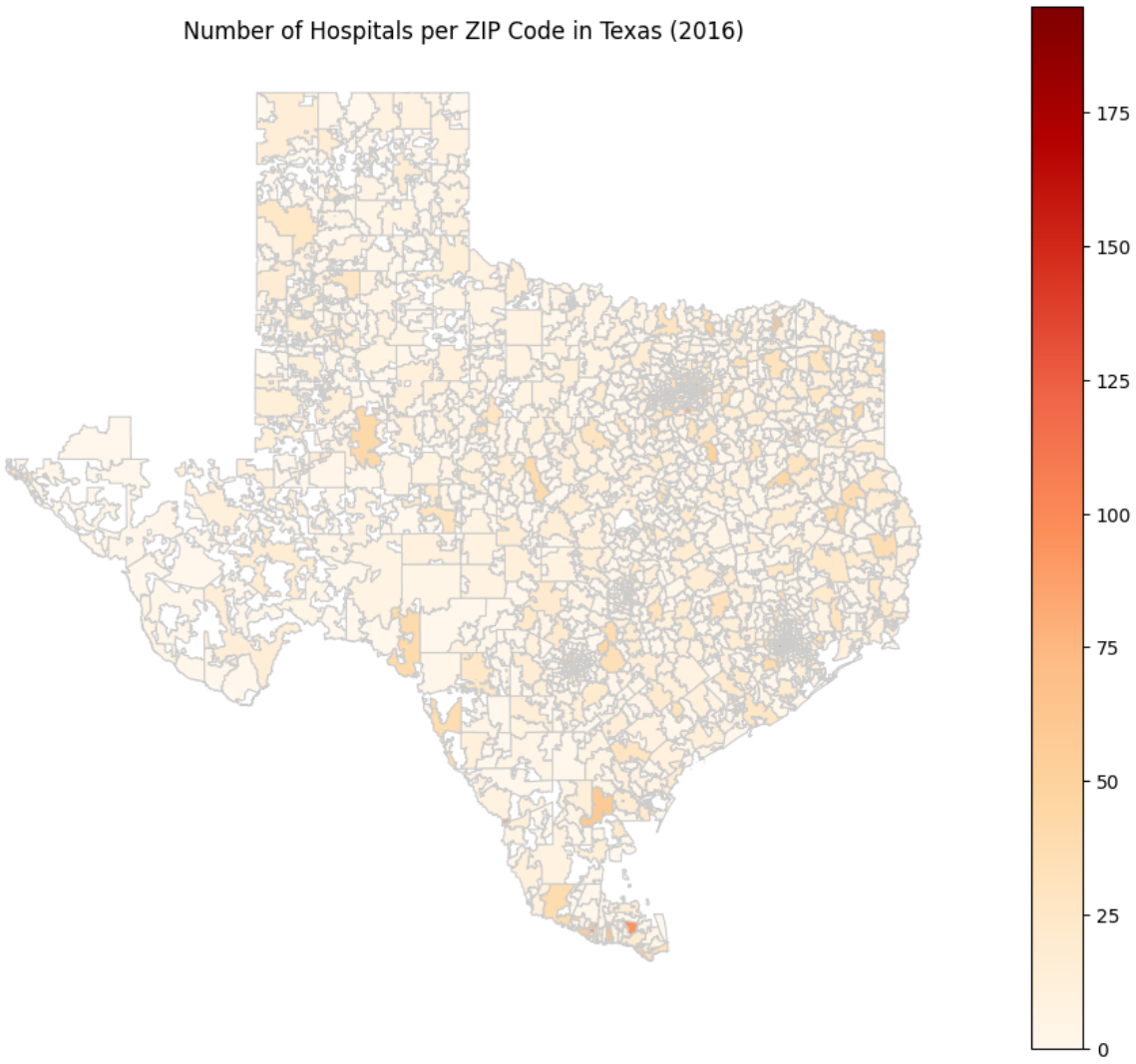


Figure 2: hospitals/zipcode

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

1.

```
zip_shapes['centroids']= zip_shapes.geometry.centroid
all_zip_centroid= zip_shapes.copy().set_geometry('centroids')
print(all_zip_centroid.shape.head(3))
print(all_zip_centroid.columns.head(3))
print(all_zip_centroid.head(3))
```

GEO_ID: it is ta code to identify the grographic type: state, county, zipcode

ZCTA5:it is 5 digit zip code tabulation area(approximation of postal codes)

NAME: is the same as ZCTA5 only more user friendly.

LSDA: legal/statistical area description categorizing geo area type(state, county,zipcode(in this case our description is zip code ZCTA5))

CENSUSAREA:The land area of the ZCTA in square miles as calculated by the Census Bureau. It measures the physical size of each area, excluding water bodies.

2.

```
from shapely.ops import unary_union

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

zips_texas_borderstates_centroids =
↪  all_zip_centroid[all_zip_centroid['ZCTA5'].str.startswith(
    ('75', '76', '77', '78', '79', '70', '71', '72', '73', '74', '80', '81', '88', '87',
↪  '86')
)]

num_texas_zips = zips_texas_centroids['ZCTA5'].nunique()
num_bordering_zips = zips_texas_borderstates_centroids['ZCTA5'].nunique()
print(f"Number of unique Texas ZIP codes: {num_texas_zips}")
print(f"Number of unique ZIP codes in Texas and bordering states: {num_bordering_zips}")
def intersects_texas(texas_polygon, other_polygon):
    return texas_polygon.intersects(other_polygon)

texas_polygon = unary_union(zips_texas_centroids.geometry)

zips_texas_borderstates_centroids['borders_texas'] =
↪  zips_texas_borderstates_centroids.geometry.apply(
    lambda geom: intersects_texas(texas_polygon, geom)
)

unique_texas_zips = zips_texas_centroids['ZCTA5'].nunique()
unique_bordering_zips =
↪  zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['borders_texas']]['ZCTA5']

print(f"Unique Texas ZIP codes: {unique_texas_zips}")
print(f"Unique ZIP codes in Texas and bordering states: {unique_bordering_zips}")
```

3.

```
texas_hospitals_2016 = pos2016[pos2016['ZIP_CD'].str.startswith(('75', '76', '77', '78',
↪  '79')) | (pos2016['ZIP_CD'] == '733')]
hospitals_per_zip =
↪  texas_hospitals_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')
hospitals_per_zip['ZIP_CD'] = hospitals_per_zip['ZIP_CD'].astype(str).str.zfill(5)

zips_texas_borderstates_centroids['ZCTA5'] =
↪  zips_texas_borderstates_centroids['ZCTA5'].astype(str).str.zfill(5)

print("Sample ZIP codes in hospitals_per_zip (should be string with zero-padding):")
print(hospitals_per_zip['ZIP_CD'].head(3))
```

```
print("Sample ZIP codes in zips_texas_borderstates_centroids (should be string with
 ↪  zero-padding):")
print(zips_texas_borderstates_centroids['ZCTA5'].head(3))

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospitals_per_zip,
    left_on='ZCTA5',
    right_on='ZIP_CD',
    how='inner'
)

zips_withhospital_centroids =
 ↪  zips_withhospital_centroids[zips_withhospital_centroids['hospital_count'] > 0]

print("zips_withhospital_centroids with at least 1 hospital:")
print(zips_withhospital_centroids.head(3))
num_unique_zip_codes = zips_withhospital_centroids['ZCTA5'].nunique()
print(f"Number of unique ZIP codes with at least 1 hospital in 2016:
 ↪  {num_unique_zip_codes}")
```

I inner merged on zip code but had osme problems because they did not match (zip code in geo file had decimals and had to be converted)

4. a.

```
import numpy as np
import time
start_time_T= time.time()
LAT_TO_MILES = 69.0
LON_TO_MILES = 55.0

def degree_to_miles_distance(point1, point2):
    lat_diff = (point1.y - point2.y) * LAT_TO_MILES
    lon_diff = (point1.x - point2.x) * LON_TO_MILES
    return np.sqrt(lat_diff**2 + lon_diff**2)

def calculate_nearest_distance(point, centroids):
    if centroids.empty:
        return float('inf')  # Return infinity if there are no centroids
    return min(degree_to_miles_distance(point, centroid) for centroid in centroids)

Q4a_subset = zips_texas_centroids.head(3)
start_time = time.time()

Q4a_subset['nearest_hospital_distance'] = Q4a_subset['centroids'].apply(
    lambda x: calculate_nearest_distance(x, zips_withhospital_centroids['centroids'])
)

end_time = time.time()
```

```
subset_duration = end_time - start_time
total_zips_count = len(zips_texas_centroids)
estimated_total_duration = (subset_duration / 10) * total_zips_count

print(f"Time taken for 10 ZIP codes: {subset_duration:.2f} seconds")
print(f"Estimated time for entire dataset: {estimated_total_duration / 60:.2f} minutes")
```

It will take approximatelty 18 seconds to run the whole dataset! b. look at part C where I calculated the actual distance. I have mentioned how much it will take to run the whole thing there. it is 36 seconds! c. the unit is degree, which is approximately 69 miles or 111 km. here is convertion:

```
start_time_T= time.time()

zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪  zips_texas_centroids['centroids'].apply(
    lambda centroid: min(
        degree_to_miles_distance(centroid, hospital_centroid)
        for hospital_centroid in zips_withhospital_centroids['centroids']
    )
)

average_distance_miles = zips_texas_centroids['nearest_hospital_distance_miles'].mean()
end_time_T=time.time()
total_time= end_time_T - start_time_T
print(f"The average distance to the nearest hospital in Texas (in miles) is:
 ↪  {average_distance_miles:.2f}", "miles")
print("the total time required to calculate the prompt above is", total_time, "seconds")
```

5.

```
average_distance = zips_texas_centroids['nearest_hospital_distance_miles'].mean()
print(f"Average distance to the nearest hospital for each ZIP code in Texas:
 ↪  {average_distance:.2f} miles")

fig, ax = plt.subplots(1, 1, figsize=(12, 10))
zips_texas_centroids.plot(column='nearest_hospital_distance_miles', cmap='YlOrRd',
 ↪  linewidth=0.8, ax=ax, edgecolor='0.8', legend=True)
ax.set_title("Distance to the Nearest Hospital for Each ZIP Code in Texas (Miles)")
ax.set_axis_off()
plt.show()
```

Effects of closures on access in Texas (15 pts)

1. There are 20 hospitals that have had at least one closure between 2016-2019.

```
corrected_closures_df = pd.DataFrame(properly_closed_hospitals)

corrected_closures_df['ZIP_CD'] = corrected_closures_df['ZIP_CD'].astype(str)
texas_closures =
 ↪  corrected_closures_df[corrected_closures_df['ZIP_CD'].str.startswith(('75', '76',
 ↪  '77', '78', '79', '733'))]
closures_by_zip =
 ↪  texas_closures.groupby('ZIP_CD').size().reset_index(name='closure_count')

print("Number of hospital closures by ZIP code in Texas (2016-2019):")
print(closures_by_zip.head(3))
```

2.

```
print(texas_zip_shapes.columns.head(3))
print(corrected_closures_df.columns.head(3))

closures_by_zip =
 ↪  corrected_closures_df.groupby('ZIP_CD').size().reset_index(name='closure_count')
texas_zip_shapes = texas_zip_shapes.drop(columns=['closure_count_x', 'closure_count_y'],
 ↪  errors='ignore')
texas_zip_shapes = texas_zip_shapes.merge(closures_by_zip, on='ZIP_CD', how='left',
 ↪  suffixes=('', '_closure'))
texas_zip_shapes['closure_count'] = texas_zip_shapes['closure_count'].fillna(0)

print(texas_zip_shapes.columns.head(3))
print(texas_zip_shapes[['ZIP_CD', 'closure_count']].head(3))
print(len(texas_zip_shapes))
```

3.

```
affected_zip_codes = affected_zip_codes[affected_zip_codes['geometry'].notnull()]
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(12, 10))
affected_zip_codes.plot(column='closure_count', cmap='OrRd', linewidth=0.8, ax=ax,
 ↪  edgecolor='0.8', legend=True)
ax.set_title("Texas ZIP Codes Directly Affected by Hospital Closures (2016-2019)")
ax.set_axis_off()
plt.show()
```

4.

```
texas_zip_shapes['affected_status'] = 'Not Affected'
directly_affected = texas_zip_shapes[texas_zip_shapes['closure_count'] > 0]
texas_zip_shapes.loc[texas_zip_shapes['closure_count'] > 0, 'affected_status'] =
 ↪  'Directly Affected'

texas_zip_shapes = texas_zip_shapes.to_crs(epsg=5070)
```

```
directly_affected = directly_affected.to_crs(epsg=5070)
buffer_10_miles = directly_affected.buffer(16093.4).unary_union

within_10_miles =
 ↪  texas_zip_shapes[texas_zip_shapes['geometry'].intersects(buffer_10_miles) &
 ↪  (texas_zip_shapes['affected_status'] == 'Not Affected')]
texas_zip_shapes.loc[within_10_miles.index, 'affected_status'] = 'Within 10 Miles of
 ↪  Closure'

fig, ax = plt.subplots(1, 1, figsize=(12, 10))
texas_zip_shapes.plot(column='affected_status', cmap='Set1', linewidth=0.8, ax=ax,
 ↪  edgecolor='0.8', legend=True)
ax.set_title("Texas ZIP Codes by Closure Impact (2016-2019)")
ax.set_axis_off()
plt.show()
```

Reflecting on the exercise (10 pts)

Partner 1: The "first-pass" method for identifying hospital closures has limitations that could lead to inaccuracies. Capturing data only annually risks misidentifying closures, as hospitals that close and reopen within a year may still appear as closed. More frequent data collection (e.g., monthly) and cross-referencing with reliable healthcare databases like CMS or Medicare could improve accuracy. Additionally, hospitals that merge or reclassify, such as converting to outpatient centers, may seem closed but still offer services; tracking such changes would help clarify true closures. Finally, factoring in geographical and demographic differences, especially between urban and rural areas, would provide a more realistic view of how closures impact access, better reflecting actual community needs.

Partner 2: Identifying affected ZIP codes by proximity to closures is helpful but might not fully reflect changes in hospital accessibility. ZIP codes vary in size and infrastructure, making it overly simplistic to define access by ZIP code alone. Using real travel distances or public transit times would better capture access limitations. Additionally, hospital closures impact communities differently depending on the services provided—an emergency center's closure has a greater effect on access than a specialized facility. Finally, rural areas often rely on a single hospital, unlike urban areas with multiple options within a short distance. Adjusting thresholds for rural and urban settings could improve the measure's accuracy.