Problem Set 4

AUTHOR
Sitong Guo, Helen Liu

PUBLISHED
November 3, 2024

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): Sitong Guo (rehinkerg)
   - Partner 2 (name and cnet ID): Hailun Liu (hailunl)
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: ** SG ** ** HL **
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: **__** Late coins left after submission: **__**
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)

1. PRVDR_CTGRY_SBTYP_CD(Provider Category Subtype Code), PRVDR_CTGRY_CD(Provider Category Code), FAC_NAME(Facility Name), PRVDR_NUM(CMS Certification Number), PGM_TRMNTN_CD(Termination Code), ZIP_CD(Address: ZIP Code).

2. a. The file documented 7245 short term hospitals in 2016. This is likely to be overestimated for the acting quantity since that there is fraction of non-operating ones included.

   ::: {#6837d743 .cell execution_count=1} ``` {.python .cell-code} import pandas as pd

   pos2016 = pd.read_csv('E:/pos2016.csv',encoding='ISO-8859-1', dtype={'ZIP_CD': str})

   short_term_hospitals = pos2016[(pos2016['PRVDR_CTGRY_CD'] == 1) & (pos2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]

   num_hospitals = short_term_hospitals['PRVDR_NUM'].nunique()

   print(f"Number of short-term hospitals in 2016: {num_hospitals}") ```

   ::: {.cell-output .cell-output-stderr} ``` C:_18256\3682174362.py:3: DtypeWarning:

   Columns (3) have mixed types. Specify dtype option on import or set low_memory=False.

   ```
   :::
   ```

   ```
   ::: {.cell-output .cell-output-stdout}
   ```

Number of short-term hospitals in 2016: 7245 ``` ::: :::

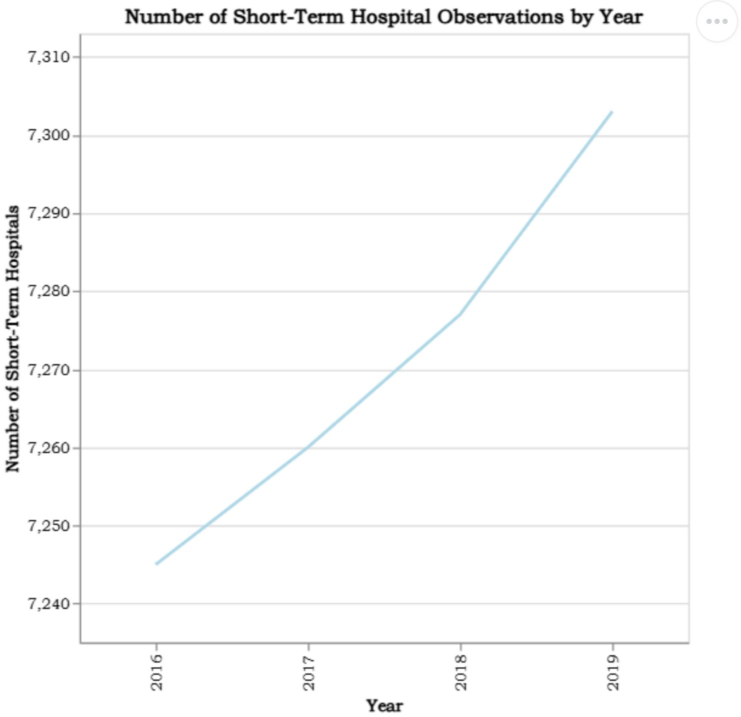
b. According to the KFF, as of Jul 07, 2016, there are nearly 5,000 short-term, acute care hospitals in the United States. As stated by 2016 CMS Statistics, on the other hand, Medicare short-term hospital was only 3436. This disvrepancy is due to that the file contains hospitals not acting and there might be other short-term hospitals besides acute care. (The Number of U.S. Hospitals by Type (Total 5534), FY2016 by American Hospital Association claimed that there were only 5534 hospitals in total in 2016, this is due to some inexplicable divergences on statistical caliber on definition of hospital.)

3.

```
import matplotlib.pyplot as plt
import altair as alt

pos2017 = pd.read_csv('E:/pos2017.csv', encoding='ISO-8859-1',dtype={'ZIP_CD': str})
pos2018 = pd.read_csv('E:/pos2018.csv', encoding='ISO-8859-1',dtype={'ZIP_CD': str})
pos2019 = pd.read_csv('E:/pos2019.csv', encoding='ISO-8859-1',dtype={'ZIP_CD': str})

def short_term(df):
    return df[(df['PRVDR_CTGRY_CD'] == 1) & (df['PRVDR_CTGRY_SBTYP_CD'] == 1)]

short_term_2016 = short_term(pos2016)
short_term_2017 = short_term(pos2017)
short_term_2018 = short_term(pos2018)
short_term_2019 = short_term(pos2019)

short_term_2016['Year'] = 2016
short_term_2017['Year'] = 2017
short_term_2018['Year'] = 2018
short_term_2019['Year'] = 2019

# append
all_years = pd.concat([short_term_2016, short_term_2017, short_term_2018, short_term_2019])

counts_by_year = all_years['Year'].value_counts().sort_index().reset_index()
counts_by_year.columns = ['Year', 'Count']

# set range for the y axis
y_min = counts_by_year['Count'].min() - 10
y_max = counts_by_year['Count'].max() + 10

alt.Chart(counts_by_year).mark_line(color='lightblue').encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Count:Q', title='Number of Short-Term Hospitals', scale=alt.Scale(domain=[y_min, y_max]))
).properties(
    title ='Number of Short-Term Hospital Observations by Year',
    height = 400,
    width = 400
)
```

```
C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\3356294368.py:16: 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-copy

C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\3356294368.py:17: 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-copy

C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\3356294368.py:18: 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-copy

C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\3356294368.py:19: 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-copy
```



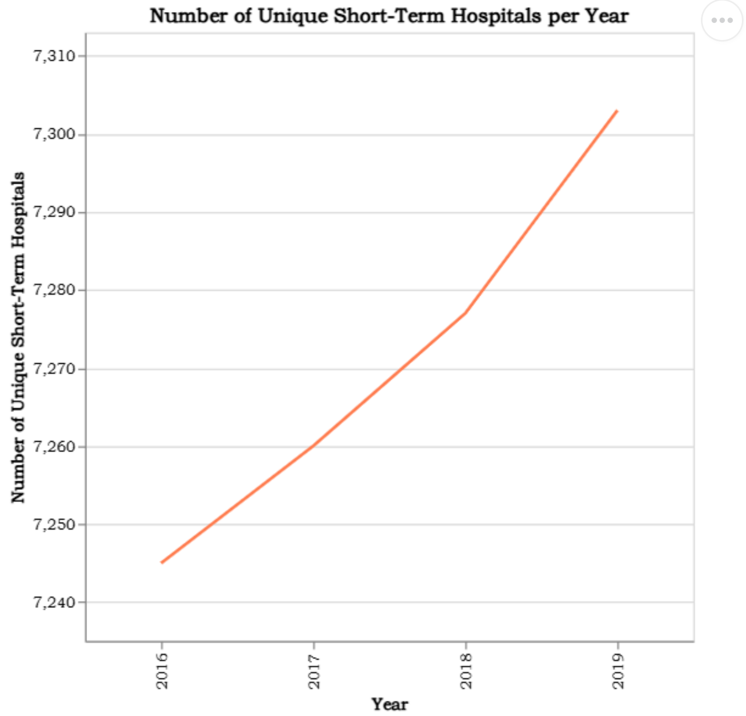
4. a.

::: {#a2ba149c .cell execution_count=3} ``` {.python .cell-code} unique_hospitals_per_year = all_years.groupby('Year')['PRVDR_NUM'].nunique().reset_index() unique_hospitals_per_year.columns = ['Year', 'Unique_Count']

y_min = unique_hospitals_per_year['Unique_Count'].min() - 10 y_max = unique_hospitals_per_year['Unique_Count'].max() + 10

alt.Chart(unique_hospitals_per_year).mark_line(color='coral').encode(x=alt.X('Year:O', title='Year'), y=alt.Y('Unique_Count:Q', title='Number of Unique Short-Term Hospitals', scale=alt.Scale(domain=[y_min, y_max]))).properties(title ='Number of Unique Short-Term Hospitals per Year', width = 400, height = 400) ```

::: {.cell-output .cell-output-display execution_count=3}



::: :::

b. They are identical, which means that the short-term hospitals were each having a unique CCN without redundancy, in the period of 2016-2019.

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

1.

```
pos20161 = pos2016[pos2016['PRVDR_CTGRY_CD'] == 1]
pos20161 = pos20161[pos20161['PRVDR_CTGRY_SBTYP_CD'] == 1]
pos20171=pos2017[pos2017['PRVDR_CTGRY_CD'] == 1]
pos20171=pos20171[pos20171['PRVDR_CTGRY_SBTYP_CD'] == 1]
pos20181=pos2018[pos2018['PRVDR_CTGRY_CD'] == 1]
pos20181=pos20181[pos20181['PRVDR_CTGRY_SBTYP_CD'] == 1]
pos20191=pos2019[pos2019['PRVDR_CTGRY_CD'] == 1]
pos20191=pos20191[pos20191['PRVDR_CTGRY_SBTYP_CD'] == 1]

pos2016_subset = pos20161[['FAC_NAME', 'PRVDR_NUM', 'PGM_TRMNTN_CD', 'ZIP_CD']]
pos2017_subset = pos20171[['FAC_NAME', 'PRVDR_NUM', 'PGM_TRMNTN_CD', 'ZIP_CD']]
pos2018_subset = pos20181[['FAC_NAME', 'PRVDR_NUM', 'PGM_TRMNTN_CD', 'ZIP_CD']]
pos2019_subset = pos20191[['FAC_NAME', 'PRVDR_NUM', 'PGM_TRMNTN_CD', 'ZIP_CD']]

pos2016_subset = pos2016_subset.add_suffix('_2016').rename(columns={'PRVDR_NUM_2016': 'PRVDR_NUM'})
pos2017_subset = pos2017_subset.add_suffix('_2017').rename(columns={'PRVDR_NUM_2017': 'PRVDR_NUM'})
pos2018_subset = pos2018_subset.add_suffix('_2018').rename(columns={'PRVDR_NUM_2018': 'PRVDR_NUM'})
pos2019_subset = pos2019_subset.add_suffix('_2019').rename(columns={'PRVDR_NUM_2019': 'PRVDR_NUM'})

merged = pos2016_subset.merge(pos2017_subset, on='PRVDR_NUM', how='outer') \
                            .merge(pos2018_subset, on='PRVDR_NUM', how='outer') \
                            .merge(pos2019_subset, on='PRVDR_NUM', how='outer')

merged_2016_0 = merged[merged['PGM_TRMNTN_CD_2016'] == 0]

suspected = merged_2016_0[
    ~((merged_2016_0['PGM_TRMNTN_CD_2016'] == 0) &
      (merged_2016_0['PGM_TRMNTN_CD_2017'] == 0) &
      (merged_2016_0['PGM_TRMNTN_CD_2018'] == 0) &
      (merged_2016_0['PGM_TRMNTN_CD_2019'] == 0))
]
suspected = suspected[['PRVDR_NUM', 'FAC_NAME_2016', 'ZIP_CD_2016', 'PGM_TRMNTN_CD_2016', 'PGM_TRMNTN_CD_

count = suspected['FAC_NAME_2016'].unique()
print(len(count))
```

```
174
```

There are 174 hospital that fit this definition(hospitals that were active in 2016 that were suspected to have closed by 2019)

2.

```python
import numpy as np
conditions = [
    (suspected['PGM_TRMNTN_CD_2017'] != 0) & (suspected['PGM_TRMNTN_CD_2018'] != 0) & (suspected['PGM_TRI
    (suspected['PGM_TRMNTN_CD_2017'] == 0) & (suspected['PGM_TRMNTN_CD_2018'] != 0) & (suspected['PGM_TRI
    (suspected['PGM_TRMNTN_CD_2017'] == 0) & (suspected['PGM_TRMNTN_CD_2018'] == 0) & (suspected['PGM_TRI
]
choices = [2017, 2018, 2019]
suspected['YearOfSuspectedClosure'] = np.select(conditions, choices, default=np.nan)

sorted_suspected = suspected.sort_values(by='FAC_NAME_2016')[['FAC_NAME_2016', 'YearOfSuspectedClosure',

top_10_hospitals = sorted_suspected.head(10)
print(top_10_hospitals)
```

```
                                       FAC_NAME_2016  YearOfSuspectedClosure  \
168                           ABRAZO MARYVALE CAMPUS                  2017.0
568         ADVENTIST MEDICAL CENTER - CENTRAL VALLEY                 2017.0
4852                           AFFINITY MEDICAL CENTER                2018.0
4356   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS                  2017.0
6273        ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE                  2017.0
3596                           ALLIANCE LAIRD HOSPITAL                2019.0
4990                          ALLIANCEHEALTH DEACONESS                2019.0
1384                     ANNE BATES LEACH EYE HOSPITAL                2019.0
1044          ARKANSAS VALLEY REGIONAL MEDICAL CENTER                 2017.0
3975               BANNER CHURCHILL COMMUNITY HOSPITAL                2017.0

     ZIP_CD_2016
168        85031
568        93230
4852       44646
4356       12208
6273       75662
3596       39365
4990       73112
1384       33136
1044       81050
3975       89406
```

3. a.

::: {#92bb617b .cell execution_count=6} ``` {.python .cell-code} #hospitals active each year active_2016 = pos2016_subset[pos2016_subset['PGM_TRMNTN_CD_2016'] == 0][['ZIP_CD_2016', 'PRVDR_NUM']] active_2017 = pos2017_subset[pos2017_subset['PGM_TRMNTN_CD_2017'] == 0][['ZIP_CD_2017', 'PRVDR_NUM']] active_2018 = pos2018_subset[pos2018_subset['PGM_TRMNTN_CD_2018'] == 0] [['ZIP_CD_2018', 'PRVDR_NUM']] active_2019 = pos2019_subset[pos2019_subset['PGM_TRMNTN_CD_2019'] == 0][['ZIP_CD_2019', 'PRVDR_NUM']]

#by ZIP each year active_count_2016 = active_2016.groupby('ZIP_CD_2016').size().reset_index(name='ActiveCount_2016') active_count_2017 = active_2017.groupby('ZIP_CD_2017').size().reset_index(name='ActiveCount_2017') active_count_2018 = active_2018.groupby('ZIP_CD_2018').size().reset_index(name='ActiveCount_2018') active_count_2019 = active_2019.groupby('ZIP_CD_2019').size().reset_index(name='ActiveCount_2019')

active_count_2016 = active_count_2016.rename(columns={'ZIP_CD_2016': 'ZIP_CD'}) active_count_2017 = active_count_2017.rename(columns={'ZIP_CD_2017': 'ZIP_CD'}) active_count_2018 = active_count_2018.rename(columns={'ZIP_CD_2018': 'ZIP_CD'}) active_count_2019 = active_count_2019.rename(columns={'ZIP_CD_2019': 'ZIP_CD'})

# merge active counts to compare by ZIP zip_counts = active_count_2016.merge(active_count_2017, on='ZIP_CD', how='outer')
.merge(active_count_2018, on='ZIP_CD', how='outer')
.merge(active_count_2019, on='ZIP_CD', how='outer')

suspected = suspected.merge(zip_counts, left_on='ZIP_CD_2016', right_on='ZIP_CD', how='left')

#number of active did not decrease…questionable! suspected['IsMerger'] = np.where(((suspected['YearOfSuspectedClosure'] == 2017) & (suspected['ActiveCount_2017'] >= suspected['ActiveCount_2016'])) | ((suspected['YearOfSuspectedClosure'] == 2018) & (suspected['ActiveCount_2018'] >= suspected['ActiveCount_2017'])) | ((suspected['YearOfSuspectedClosure'] == 2019) & (suspected['ActiveCount_2019'] >= suspected['ActiveCount_2018'])), True, False)

#potential merger/acquisition corrected_closures = suspected[~suspected['IsMerger']] merger_count = suspected['IsMerger'].sum() print(f"potential mergers/acquisitions: {merger_count}") ```

::: {.cell-output .cell-output-stdout} `potential mergers/acquisitions: 6` ::: :::

b.

::: {#bbe9c8d7 .cell execution_count=7} `{.python .cell-code} #corrected closures corrected_closure_count = corrected_closures['FAC_NAME_2016'].nunique() print(f"hospitals correcting for m/a: {corrected_closure_count}")`

::: {.cell-output .cell-output-stdout} `hospitals correcting for m/a: 168` ::: :::

c.

::: {#145ce8fe .cell execution_count=8} `{.python .cell-code} sorted_corrected_closures = corrected_closures.sort_values(by='FAC_NAME_2016')[['FAC_NAME_2016', 'YearOfSuspectedClosure', 'ZIP_CD_2016']] print(sorted_corrected_closures.head(10))`

::: {.cell-output .cell-output-stdout} ``` FAC_NAME_2016 YearOfSuspectedClosure
4 ABRAZO MARYVALE CAMPUS 2017.0
10 ADVENTIST MEDICAL CENTER - CENTRAL VALLEY 2017.0
97 AFFINITY MEDICAL CENTER 2018.0
80 ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS 2017.0
140 ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE 2017.0
62 ALLIANCE LAIRD HOSPITAL 2019.0
101 ALLIANCEHEALTH DEACONESS 2019.0
26 ANNE BATES LEACH EYE HOSPITAL 2019.0
21 ARKANSAS VALLEY REGIONAL MEDICAL CENTER 2017.0
69 BANNER CHURCHILL COMMUNITY HOSPITAL 2017.0

```
ZIP_CD_2016
```

4 85031
10 93230
97 44646
80 12208
140 75662
62 39365
101 73112
26 33136
21 81050
69 89406
``` ::: :::

# Download Census zip code shapefile (10 pt)

1. a. (1).shp (Shape file): Main file that has feature geometrics, such as points, lines, or polygons that represent the shapes of geographic features inclluding ZIP code boundaries.

(2).shx (Shape index file) : Contains an positional index of the geometries in the shp file, accelerating access to geographic features.

(3).dbf (database file): a tabular file with attribute information, in dBASE format that stores attributes or additional data about each shape in the shp file.

(4).prj (projection file): Describes the Coordinate Reference System (CRS). Contains information about the system and projection used in the shp, ensuring that the data aligns correctly with other geographic data.
```

(5).xml: Detailed text information about the dataset like source, description, date, attribute definitions, and other information for understanding the data's context and structure.

b.

::: {#eb81ee47 .cell execution_count=9} ``` {.python .cell-code} import os directory = 'E:/SeriousBusiness/Applications/uchicago/python2'

for filename in os.listdir(directory): file_path = os.path.join(directory, filename) if os.path.isfile(file_path): file_size = os.path.getsize(file_path) print(f"{filename}: {file_size} KB") ```

::: {.cell-output .cell-output-stdout} `gz_2010_us_860_00_500k.dbf: 6425474 KB gz_2010_us_860_00_500k.prj: 165 KB gz_2010_us_860_00_500k.shp: 837544580 KB gz_2010_us_860_00_500k.shx: 265060 KB gz_2010_us_860_00_500k.xml: 15639 KB PS4.docx: 39891 KB PS4.pdf: 240979 KB pset4_template.qmd: 3234 KB ~$PS4.docx: 162 KB` ::: :::

2.

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

#number of hospitals per ZIP
hospitals_per_zip = active_2016.groupby('ZIP_CD_2016').size().reset_index(name='Hospital_Count')
hospitals_per_zip.columns = ['ZIP_CD', 'Hospital_Count']
```

```
zip_codes = gpd.read_file('E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.shp')
```

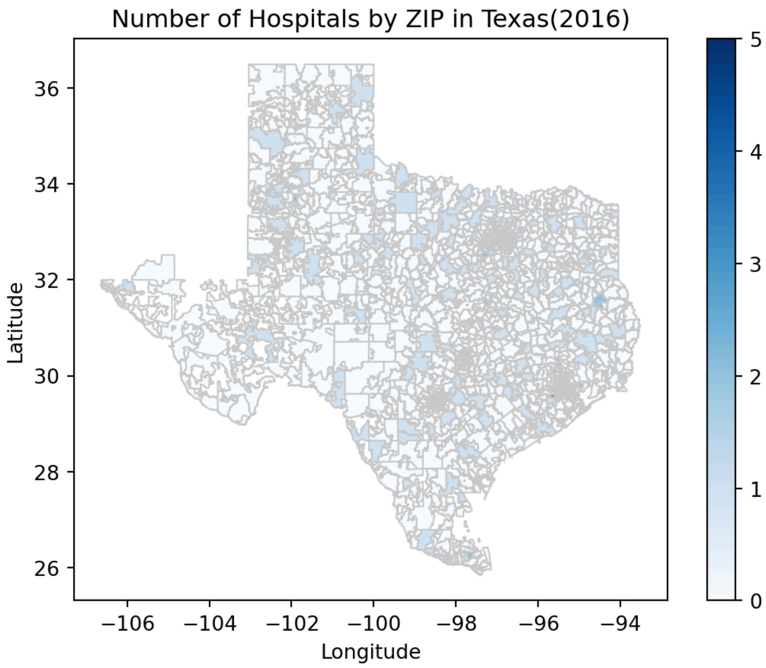
```
#print(zip_codes.columns)
# Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry'], dtype='object')
#Texas
texas_zip_codes = zip_codes[zip_codes['ZCTA5'].str.startswith(('75', '76', '77', '78', '79'))]
```

```
texas_zip_hospitals = texas_zip_codes.merge(hospitals_per_zip, left_on='ZCTA5', right_on='ZIP_CD', how='
#fill missing hospital counts with 0!
texas_zip_hospitals['Hospital_Count'] = texas_zip_hospitals['Hospital_Count'].fillna(0)
texas_zip_hospitals.head(5)
```

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry	ZIP_CD	Hospital_Count
0	8600000US78624	78624	78624	ZCTA5	708.041	POLYGON ((-98.96423 30.49848, -98.96416 30.498...	78624	1.0
1	8600000US78626	78626	78626	ZCTA5	93.046	POLYGON ((-97.60944 30.57185, -97.61688 30.568...	NaN	0.0
2	8600000US78628	78628	78628	ZCTA5	73.382	POLYGON ((-97.69285 30.57122, -97.69286 30.571...	NaN	0.0
3	8600000US78631	78631	78631	ZCTA5	325.074	POLYGON ((-99.13053 30.36555, -99.13065 30.365...	NaN	0.0
4	8600000US78632	78632	78632	ZCTA5	96.278	POLYGON ((-97.40946 29.75929, -97.40947 29.758...	NaN	0.0

```
plt.figure(figsize=(18, 15))
texas_zip_hospitals.plot(column='Hospital_Count', cmap='Blues', linewidth=0.8, edgecolor='0.8', legend=Tr
plt.title('Number of Hospitals by ZIP in Texas(2016)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

```
<Figure size 1728x1440 with 0 Axes>
```



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

1.

```
shp = gpd.read_file('E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.shp')
shx = gpd.read_file('E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.shx')
zips_all_centroids = shp.copy()
zips_all_centroids['geometry'] = shp.geometry.centroid
```

```
C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\65589225.py:4: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()'
to re-project geometries to a projected CRS before this operation.
```

Dimensions: The dimensions output from the GeoDataFramel created for the centroid of each zip code nationally gives the number of rows and columns which is equal to the number of unique ZIP codes in the data. Each row represents a unique ZIP Code area, and the columns represent the . The coordinate position of the centroids(each ZIP Code).

Columns meaning: 1.GEO_ID: A unique identifier for each geographic region, helping to distinguish each ZIP Code area. 2.ZCTA5:This is the 5-digit ZIP Code Tabulation Area (ZCTA), a representation of ZIP Code regions used in census data. 3.NAME: Typically represents the ZIP Code for the area. 4.LSAD: Legal/Statistical Area Description, identifying the type of area, such as city or rural. 5.geometry: Contains the centroid point of each ZIP Code area.This is a Point geometry representing the geometric center of each ZIP Code area.

2. In subsets of zips_texas_centroids,there are 1935 unique zip codes. In subsets of zips_texas_borderstates_centroids,there are 4057 unique zip codes.

```
import geopandas as gpd
zips_all_centroids['ZIP_INT'] = zips_all_centroids['ZCTA5'].astype(int)
texas_condition = (
    (zips_all_centroids['ZIP_INT'] >= 75000) & (zips_all_centroids['ZIP_INT'] <= 79999)
)

border_states_condition = (
    ((zips_all_centroids['ZIP_INT'] >= 87000) & (zips_all_centroids['ZIP_INT'] <= 88499)) |
    ((zips_all_centroids['ZIP_INT'] >= 70000) & (zips_all_centroids['ZIP_INT'] <= 72999)) |
    ((zips_all_centroids['ZIP_INT'] >= 73000) & (zips_all_centroids['ZIP_INT'] <= 79999))
)
zips_texas_centroids = zips_all_centroids[texas_condition]
zips_texas_borderstates_centroids = zips_all_centroids[texas_condition | border_states_condition]
```

```python
unique_texas_zips = zips_texas_centroids['ZCTA5'].nunique()
unique_texas_border_zips = zips_texas_borderstates_centroids['ZCTA5'].nunique()
print("Unique ZIP Codes in Texas:", unique_texas_zips)
print("Unique ZIP Codes in Texas and bordering states:", unique_texas_border_zips)
```

```
Unique ZIP Codes in Texas: 1935
Unique ZIP Codes in Texas and bordering states: 4057
```

3. There are 468 rows in zips_withhospital_centroids .

We'll use an inner join since we only want the zip codes that appear in both datasets (those in zips_texas_borderstates_centroids that are also in merged_gdf).The merge will be based on the zip code column, typically named something like 'ZIP_CD_2016' in both GeoDataFrames. Ensure both columns are named consistently for the merge.

```python
dbf = gpd.read_file('E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.dbf')
merged_2016_0['ZIP_INT'] = pd.to_numeric(merged_2016_0['ZIP_CD_2016'], errors='coerce')
border_states_condition = (
    ((merged_2016_0['ZIP_INT'] >= 87000) & (merged_2016_0['ZIP_INT'] <= 88499)) |
    ((merged_2016_0['ZIP_INT'] >= 70000) & (merged_2016_0['ZIP_INT'] <= 72999)) |
    ((merged_2016_0['ZIP_INT'] >= 73000) & (merged_2016_0['ZIP_INT'] <= 79999))
)
border_states_hopital_2016 = merged_2016_0[border_states_condition]
border_states_hopital_2016 = border_states_hopital_2016.drop(columns=['ZIP_INT'])

texas_borderstates_hopital = border_states_hopital_2016.groupby('ZIP_CD_2016').size().reset_index(name='(
texas_borderstates_hopital['ZIP_CD_2016'] = texas_borderstates_hopital['ZIP_CD_2016'].astype(int)
texas_borderstates_hopital['ZIP_CD_2016'] = texas_borderstates_hopital['ZIP_CD_2016'].astype(str)

df = pd.DataFrame(zips_texas_borderstates_centroids.drop(columns='geometry'))
df = df.rename(columns={'ZCTA5': 'ZIP_CD_2016'})
df['ZIP_CD_2016'].astype(str)
```

```
C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\2794894124.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-copy

8870     70003
8871     70030
8872     70032
8873     70036
8874     70038
         ...
32917    78261
32918    78368
32919    78412
32920    78557
32921    78586
Name: ZIP_CD_2016, Length: 4057, dtype: object
```

```python
merged_gdf = df.merge(
    texas_borderstates_hopital,
    on='ZIP_CD_2016',
    how='right'
)
print(f"There are {len(merged_gdf)} rows in zips_withhospital_centroids .")
```

```
There are 468 rows in zips_withhospital_centroids .
```

```python
zips_withhospital_centroids = gpd.GeoDataFrame(merged_gdf)
```

4. a.It will take about 0.0099 seconds for subset to 10 zip codes.And the whole process will take about 1.93 seconds.

```python
import time
#shp = gpd.read_file('E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.shp')
zips_texas_centroids = shp[shp['ZCTA5'].astype(str).str[:2].isin(['75', '76', '77', '78', '79'])]
zips_withhospital_centroids = shp[shp['ZCTA5'].isin(merged_gdf['ZIP_CD_2016'])]

zips_texas_centroids['geometry'] = zips_texas_centroids.geometry.centroid
zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.geometry.centroid

zips_texas_subset = zips_texas_centroids.head(10)
start_time = time.time()
subset_join_result = gpd.sjoin_nearest(
    zips_texas_subset,
    zips_withhospital_centroids,
    how="inner",
    distance_col="distance"
)
time_taken = time.time() - start_time
subset_join_result, time_taken
```

```
C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\196238175.py:6: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()'
to re-project geometries to a projected CRS before this operation.


E:\anaconda3\Lib\site-packages\geopandas\geodataframe.py:1819: 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-copy

C:\Users\RedthinkerDantler\AppData\Local\Temp\ipykernel_18256\196238175.py:7: UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()'
to re-project geometries to a projected CRS before this operation.


E:\anaconda3\Lib\site-packages\geopandas\array.py:403: UserWarning:

Geometry is in a geographic CRS. Results from 'sjoin_nearest' are likely incorrect. Use
'GeoSeries.to_crs()' to re-project geometries to a projected CRS before this operation.


(         GEO_ID_left ZCTA5_left NAME_left LSAD_left  CENSUSAREA_left  \
 9207  8600000US78624      78624     78624     ZCTA5          708.041
 9208  8600000US78626      78626     78626     ZCTA5           93.046
 9209  8600000US78628      78628     78628     ZCTA5           73.382
 9210  8600000US78631      78631     78631     ZCTA5          325.074
 9211  8600000US78632      78632     78632     ZCTA5           96.278
 9212  8600000US78633      78633     78633     ZCTA5           82.269
 9213  8600000US78634      78634     78634     ZCTA5           63.656
 9214  8600000US78635      78635     78635     ZCTA5           15.940
 9215  8600000US78636      78636     78636     ZCTA5          349.689
 9216  8600000US78638      78638     78638     ZCTA5          114.562

                        geometry  index_right    GEO_ID_right ZCTA5_right  \
 9207   POINT (-98.87707 30.2816)         9207  8600000US78624       78624
 9208  POINT (-97.59733 30.66535)        24856  8600000US78664       78664
 9209  POINT (-97.75112 30.64108)         9231  8600000US78681       78681
 9210  POINT (-99.30528 30.33772)        32903  8600000US78028       78028
 9211  POINT (-97.47045 29.69633)        32565  8600000US78629       78629
 9212  POINT (-97.75426 30.74197)         9231  8600000US78681       78681
 9213  POINT (-97.54471 30.55908)        24856  8600000US78664       78664
 9214  POINT (-98.55961 30.21086)         9207  8600000US78624       78624
 9215  POINT (-98.41885 30.30504)         9223  8600000US78654       78654
 9216   POINT (-97.79495 29.6569)        26359  8600000US78155       78155

      NAME_right LSAD_right  CENSUSAREA_right  distance
 9207      78624      ZCTA5           708.041  0.000000
```

```
9208       78664      ZCTA5           16.562  0.167651
9209       78681      ZCTA5           21.727  0.110844
9210       78028      ZCTA5          250.675  0.337251
9211       78629      ZCTA5          425.389  0.219909
9212       78681      ZCTA5           21.727  0.210633
9213       78664      ZCTA5           16.562  0.114657
9214       78624      ZCTA5          708.041  0.325246
9215       78654      ZCTA5          200.189  0.340902
9216       78155      ZCTA5          354.566  0.189651  ,
0.0866093635559082)
```

```
b.As for the full calculation,it takes about 0.07 seconds,which is faster than l estimated.
```

```
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=3857)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=3857)
start_time = time.time()
subset_join_result = gpd.sjoin_nearest(
    zips_texas_centroids,
    zips_withhospital_centroids,
    how="inner",
    distance_col="distance"
)
time_taken = time.time() - start_time
subset_join_result, time_taken
```

```
(          GEO_ID_left ZCTA5_left NAME_left LSAD_left  CENSUSAREA_left  \
 9207   8600000US78624      78624     78624     ZCTA5          708.041
 9208   8600000US78626      78626     78626     ZCTA5           93.046
 9209   8600000US78628      78628     78628     ZCTA5           73.382
 9210   8600000US78631      78631     78631     ZCTA5          325.074
 9211   8600000US78632      78632     78632     ZCTA5           96.278
 ...               ...        ...       ...       ...              ...
 32917  8600000US78261      78261     78261     ZCTA5           29.865
 32918  8600000US78368      78368     78368     ZCTA5          216.341
 32919  8600000US78412      78412     78412     ZCTA5            8.798
 32920  8600000US78557      78557     78557     ZCTA5           11.653
 32921  8600000US78586      78586     78586     ZCTA5          176.313

                                geometry  index_right     GEO_ID_right  \
 9207   POINT (-11006945.524 3539798.214)         9207   8600000US78624
 9208   POINT (-10864484.611 3589364.081)        24856   8600000US78664
 9209   POINT (-10881605.454 3586223.598)         9231   8600000US78681
 9210   POINT (-11054613.303 3547034.688)        32903   8600000US78028
 9211   POINT (-10850361.016 3464574.867)        32565   8600000US78629
 ...                                  ...          ...              ...
 32917  POINT (-10954048.141 3463994.843)        10809   8600000US78258
 32918   POINT (-10888193.07 3262273.761)        26302   8600000US78102
 32919   POINT (-10836170.333 3211684.17)        32919   8600000US78412
 32920  POINT (-10936401.874 3012307.913)        10839   8600000US78503
 32921  POINT (-10868244.776 3012099.803)        10847   8600000US78550

       ZCTA5_right NAME_right LSAD_right  CENSUSAREA_right      distance
 9207        78624      78624      ZCTA5           708.041      0.000000
 9208        78664      78664      ZCTA5            16.562  21443.710461
 9209        78681      78681      ZCTA5            21.727  14228.011802
 9210        78028      78028      ZCTA5           250.675  42374.908087
 9211        78629      78629      ZCTA5           425.389  28125.820177
 ...           ...        ...        ...               ...           ...
 32917       78258      78258      ZCTA5            15.874  12854.585280
 32918       78102      78102      ZCTA5           469.003  39136.894333
 32919       78412      78412      ZCTA5             8.798      0.000000
 32920       78503      78503      ZCTA5            17.036   7258.416063
 32921       78550      78550      ZCTA5            95.609  19219.631838

 [1935 rows x 13 columns],
 0.026610136032104492)
```

```
c.The .prj file specifies that the unit is in "Degree" (angular unit)
```

In this context, “Degree” represents degrees of latitude and longitude. On the Earth’s surface, 1 degree of latitude is approximately equal to 69 miles. The distance represented by 1 degree of longitude varies by latitude, but in mid-latitude regions like the United States, 1 degree of longitude is roughly 53 miles. So, the approximate conversions are: 1 degree of latitude ≈ 69 miles 1 degree of longitude ≈ 53 miles (suitable for mid-latitude areas in the U.S.)

```
from pyproj import CRS
prj_file = "E:/SeriousBusiness/Applications/uchicago/python2/gz_2010_us_860_00_500k.prj"
with open(prj_file, 'r') as file:
    prj_text = file.read()
crs = CRS.from_wkt(prj_text)
print(crs)
```

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101
]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]]
```

5. a.Unit is inclueded distance.

```
subset_join_result = gpd.sjoin_nearest(
    zips_texas_centroids,
    zips_withhospital_centroids,
    how="inner",
    distance_col="distance"
)
```

```
b.the average distance is 0.21101748566398393.
```

the average distance in miles is 14.49miles

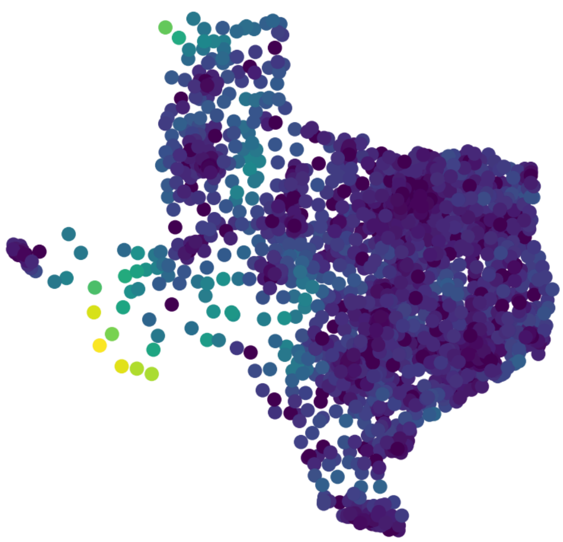
```
dd = subset_join_result.drop(columns='geometry')
mean_distance = dd['distance'].mean()
print(mean_distance)
```

```
25385.83933214372
```

```
c.
```

```
import matplotlib.pyplot as plt
subset_join_result.plot(column = 'distance').set_axis_off()
plt.axis("off")
```

```
(-11939710.352867601,
 -10348254.934308909,
 2924394.112609078,
 4435184.103883993)
```



Effects of closures on access in Texas (15 pts)

1.

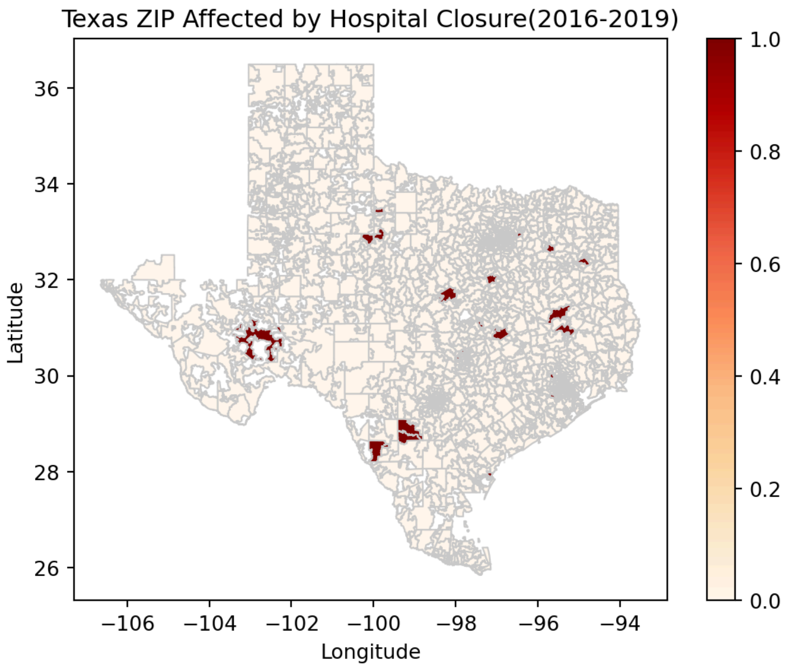
```
texas_closures = sorted_corrected_closures[sorted_corrected_closures['ZIP_CD_2016'].str.startswith(('75'
closures_by_zip = texas_closures.groupby('ZIP_CD_2016').size().reset_index(name='Number_of_Closures')
closures_by_zip.columns = ['ZIP Code', 'Number of Closures']
print(closures_by_zip)
```

```
   ZIP Code  Number of Closures
0     75042                   1
1     75051                   1
2     75087                   1
3     75140                   1
4     75231                   1
5     75235                   1
6     75390                   1
7     75601                   1
8     75662                   1
9     75835                   1
10    75862                   1
11    76502                   1
12    76520                   1
13    76531                   1
14    76645                   1
15    77035                   1
16    77054                   1
17    77065                   1
18    77429                   1
19    77479                   1
20    77598                   1
21    78017                   1
22    78061                   1
23    78336                   1
24      785                   1
25    78613                   1
26    78734                   1
27    78834                   1
28    79520                   1
29    79529                   1
30    79553                   1
31    79735                   1
32    79761                   1
33    79902                   1
```

2.

```
texas_zip_closures = texas_zip_codes.merge(closures_by_zip, left_on='ZCTA5', right_on='ZIP Code', how='l
#Fill nan with 0..
texas_zip_closures['Number of Closures'] = texas_zip_closures['Number of Closures'].fillna(0)

affected_zip_count = texas_zip_closures[texas_zip_closures['Number of Closures'] > 0]['ZCTA5'].nunique()
print(f"Directly affected ZIPs in Texas: {affected_zip_count}")

plt.figure(figsize=(12, 10))
texas_zip_closures.plot(column='Number of Closures', cmap='OrRd', linewidth=0.8, edgecolor='0.8', legend=
plt.title('Texas ZIP Affected by Hospital Closure(2016-2019)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

```
Directly affected ZIPs in Texas: 33

<Figure size 1152x960 with 0 Axes>
```



3.

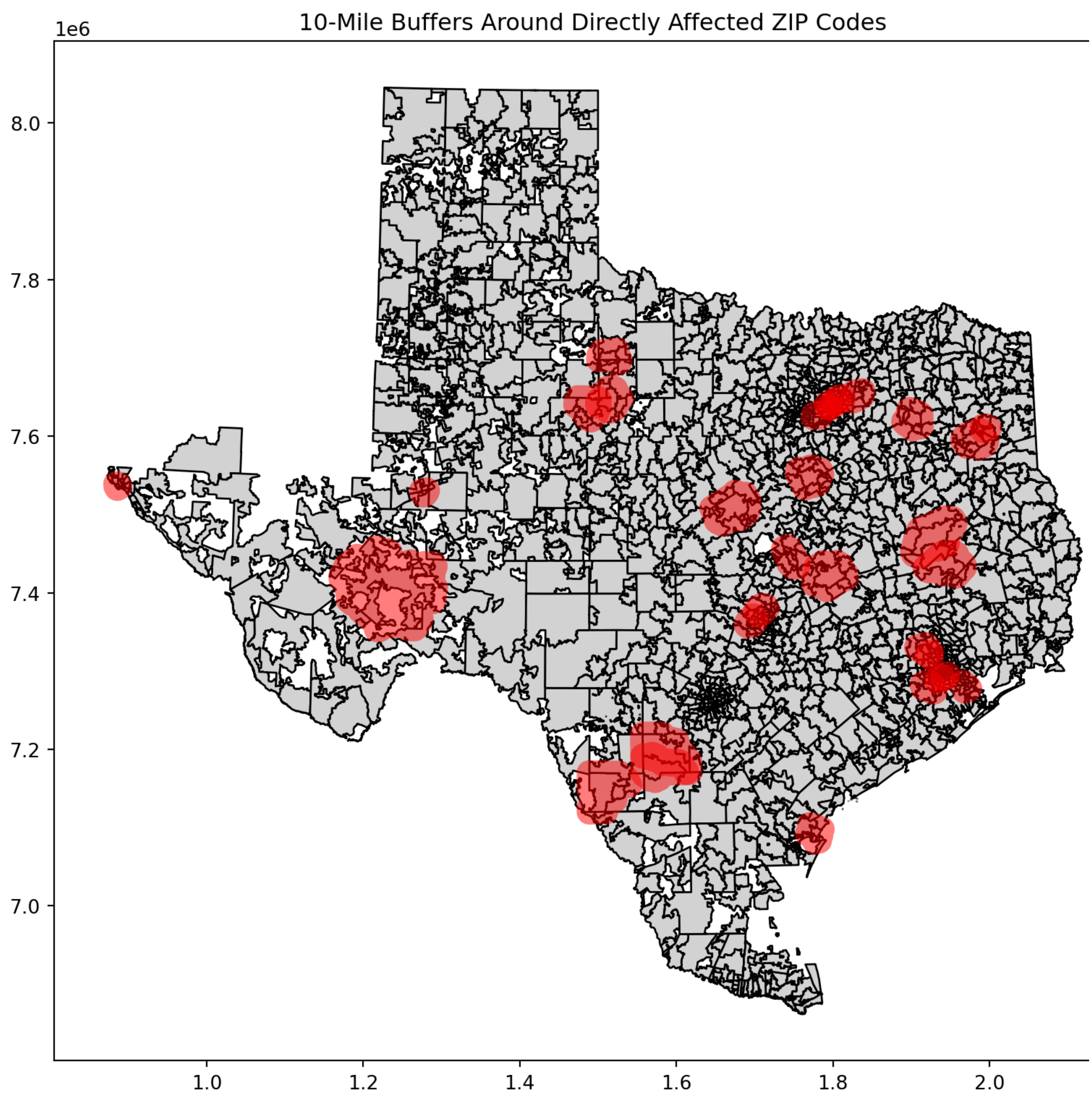
```
# 16093.44 meters
directly_affected_zips = texas_zip_closures[texas_zip_closures['Number of Closures'] > 0]
'''
 IMPORTANT!: print(directly_affected_zips.crs) to see the CRS! IT IS NOT METER.
'''
directly_affected_zips = directly_affected_zips.to_crs("EPSG:3083")
texas_zip_codes = texas_zip_codes.to_crs("EPSG:3083")
# seems like this question doesn't need buffer around centroid.
'''
directly_affected_zips['centroid'] = directly_affected_zips.geometry.centroid
directly_affected_zips['buffer'] = directly_affected_zips['centroid'].buffer(16093.4)
directly_affected_zips.set_geometry('buffer', inplace=True)
'''
directly_affected_zips['geometry'] = directly_affected_zips.geometry.buffer(16093.4)
affected_zips = gpd.sjoin(texas_zip_codes, directly_affected_zips, how='inner', predicate='intersects')

# sift out those directly; Must use ZCTA5_left, since the zcta5_right and ZIP Code is introduced from dir
indirectly_affected_zips = affected_zips[~affected_zips['ZCTA5_left'].isin(directly_affected_zips['ZCTA5
indirect_zip_count = indirectly_affected_zips['ZCTA5_left'].nunique()
print("Number of indirectly affected ZIP codes:", indirect_zip_count)
```

```
Number of indirectly affected ZIP codes: 576
```

```
print(affected_zips.columns)
```

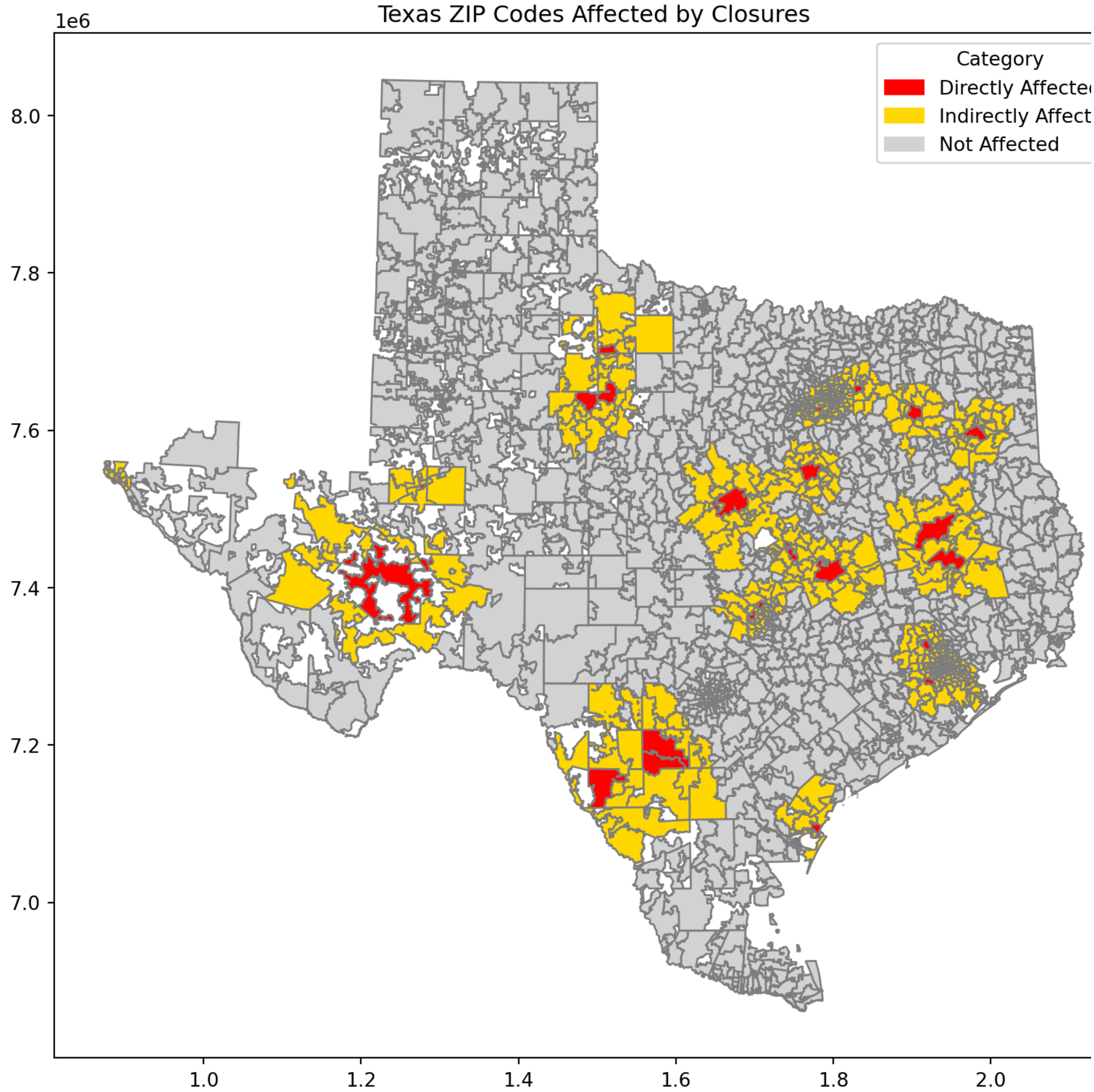
```
Index(['GEO_ID_left', 'ZCTA5_left', 'NAME_left', 'LSAD_left',
       'CENSUSAREA_left', 'geometry', 'index_right', 'GEO_ID_right',
       'ZCTA5_right', 'NAME_right', 'LSAD_right', 'CENSUSAREA_right',
       'ZIP Code', 'Number of Closures'],
      dtype='object')
```

```
# buffered areas
fig, ax = plt.subplots(figsize=(10, 10))
texas_zip_codes.plot(ax=ax, color='lightgrey', edgecolor='black')
directly_affected_zips.plot(ax=ax, color='red', alpha=0.5)
plt.title("10-Mile Buffers Around Directly Affected ZIP Codes")
plt.show()
```



4.

```
def categorize_zip(zip):
    if zip['ZCTA5'] in directly_affected_zips['ZIP Code'].values:
        return 'Directly Affected'
    elif zip['ZCTA5'] in indirectly_affected_zips['ZCTA5_left'].values:
        return 'Indirectly Affected'
    else:
        return 'Not Affected'
# apply function onto dataframe, use .apply(func, axis=1):
texas_zip_codes['Category'] = texas_zip_codes.apply(categorize_zip, axis=1)
color_mapping = {
    'Directly Affected': 'red',
    'Indirectly Affected': 'gold',
    'Not Affected': 'lightgray'
}
texas_zip_codes['Color'] = texas_zip_codes['Category'].map(color_mapping)

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_codes.plot(ax=ax, color=texas_zip_codes['Color'], edgecolor='gray',legend = True) #simply addi
ax.set_title('Texas ZIP Codes Affected by Closures')

#legend with custom patches
import matplotlib.patches as mpatches
legend_patches = [mpatches.Patch(color=color, label=label) for label, color in color_mapping.items()]
ax.legend(handles=legend_patches, title="Category")
plt.show()
```



Reflecting on the exercise (10 pts)

1.This is indeed doubtable. We are given the knowledge that when a TorF closure takes place, that will be reflected in the year's file. Thus, for an actual closure, the number of active hospital next year shall remain unaltered, and for false closure(merger/acquisition), the number next year should rise by 1. This indicates that the "non-decrease" criteria will fail to winnow out the false closures, they are all non-decrease! In a nutshell, the number we calculated in section2 is still the aggregate number of those 'simply non-active shown by termination code'. A remedy could be changing from 'not decrease' to 'increase', or sifting out those with more than 1 unique CMS number in the four years.

2.When we are identifying zip codes affected by closures,it may have duplicated date if the hopital is merged accross difference zip codes. For the hospital which located near the froniter of a zipcode may happens that sitation. I think the best way to improve this sitation is by identifying the merge hopital with a 'merger_number',which may help us better identify the true closure.Instead of calculating distance alone, measure average travel time to the nearest hospital, considering public transportation, road quality, and traffic patterns. This would give a more realistic view of accessibility.Adjust the measure to account for population density and demographics. High-population or high-need areas could be weighted more heavily, reflecting a higher demand for healthcare services.