Problem Set 4

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

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): Zheng Cui; zhengcui
- Partner 2 (name and cnet ID): Xiaotian Tang; tangxiaotian

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: **ZC XT**
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: **1**

ZC Late coins left after submission: **2**

XT Late coins left after submission: **3**

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

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

8. (Partner 1): push ps4.qmd and ps4.pdf to your github repo.
9. (Partner 1): submit ps4.pdf via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

Style Points (10 pts)

Submission Steps (10 pts)

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

```
# reset -f
```

```
import pandas as pd
import altair as alt
import geopandas as gpd
import matplotlib.pyplot as plt
import time
```

1.

```
data = pd.read_csv('/Users/tang/Desktop/1Python II/ProblemSet4/pos2016.csv',
                    encoding='latin1',
                    dtype={'ZIP_CD': str})
data.columns
```

```
Index(['PRVDR_CTGRY_SBTYP_CD', 'PRVDR_CTGRY_CD', 'FAC_NAME', 'PRVDR_NUM',
       'PGM_TRMNTN_CD', 'ZIP_CD'],
      dtype='object')
```

I pulled six variables in total: 'PRVDR_CTGRY_SBTYP_CD', 'PRVDR_CTGRY_CD', 'FAC_NAME', 'PRVDR_NUM', 'PGM_TRMNTN_CD', 'ZIP_CD'.

2.

```
data =
↪ data.loc[data['PRVDR_CTGRY_SBTYP_CD']==1].loc[data['PRVDR_CTGRY_CD']==1]
```

a.

```
data.shape[0]
print(f'There are {data.shape[0]} short term hospitals within the united
↪ states, according to this data.')
```

```
There are 7245 short term hospitals within the united states, according to
this data.
```

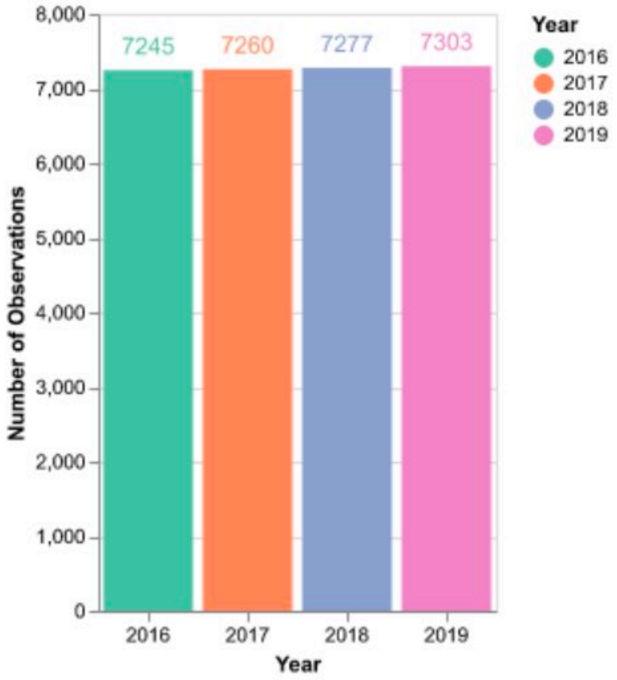
There are 7,245 short-term hospitals in the United States. This number may not make sense because it only considers how many short-term hospitals existed in 2016 without taking into account that some hospitals might have shut down that year.

b.

According to the article from the problem set, there are nearly 5,000 short-term, acute care hospitals in the United States. Our number (7,245 short-term hospitals) differs from what the article reported. Here is why we think there is a difference: our current reported data does not consider the fact that some hospitals might have shut down in 2016, so we might have over-reported the number of hospitals.

3.

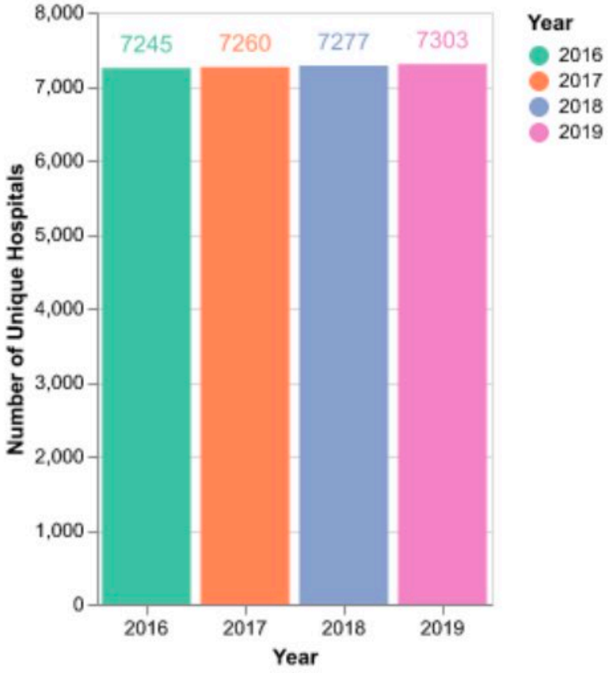
```
# import the data
data2017, data2018, data2019 = [pd.read_csv(f'/Users/tang/Desktop/1Python
↪ II/ProblemSet4/pos{year}.csv', encoding='latin1',dtype={'ZIP_CD': str})
↪ for year in [2017, 2018, 2019]]

# append the data, add a year column
df = pd.concat([df.assign(Year=year) for df, year in zip([data, data2017,
↪ data2018, data2019], [2016, 2017, 2018, 2019])], ignore_index=True)

# filter the data
df = df.loc[df['PRVDR_CTGRY_SBTYP_CD']==1].loc[df['PRVDR_CTGRY_CD']==1]

# Plot the number of obs
alt.data_transformers.disable_max_rows()

Chart = alt.Chart(df).mark_bar().encode(
    x=alt.X('Year:O', title='Year', axis=alt.Axis(labelAngle=0)),
    y=alt.Y('count():Q', title="Number of Observations"),
    color=alt.Color('Year:O', scale=alt.Scale(scheme='set2'))
).properties(width=200, height=300)

text = Chart.mark_text(
    align='center',
    baseline='bottom',
    dy=-5,
    fontSize=12
).encode(
    text='count():Q'
)

final_chart = Chart + text
final_chart.save('ps4_Q1_3.png')
```



4.

a.

```
Chart = alt.Chart(df).transform_aggregate(
    unique_hospitals='distinct(PRVDR_NUM)',
    groupby=['Year']
).mark_bar().encode(
    x=alt.X('Year:O', title='Year',axis=alt.Axis(labelAngle=0)),
    y=alt.Y('unique_hospitals:Q', title='Number of Unique Hospitals'),
    color=alt.Color('Year:O', scale=alt.Scale(scheme='set2'))
).properties(width=200, height=300)


text = Chart.mark_text(
    align='center',
    baseline='bottom',
    dy=-5,
    fontSize=12
).encode(
    text='unique_hospitals:Q'
)

final_chart = Chart+text
final_chart.save('ps4_Q1_4.png')
```



b.

This tell me that each observation has a unique CMS certification number in our dataset per year. The diagrams validate the data's structure and ensure that our conclusions are based on an accurate representation of the dataset, supporting the use of this data for further detailed healthcare analysis.

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

1.

```python
def check_closure(df):
    suspected_closure = []
    base_year = df[df["Year"]==2016]
    active = base_year[base_year["PGM_TRMNTN_CD"]==0][['FAC_NAME', 'ZIP_CD',
↪   'PRVDR_NUM']]

    for year in range(2017,2020):
        check = df[df["Year"]==year]
        merged = active.merge(check[
            ['PRVDR_NUM','PGM_TRMNTN_CD']
        ],
            on = 'PRVDR_NUM',
            how = 'left',
            indicator = True)

        closures = merged[
            (merged['_merge'] == 'left_only') | (merged['PGM_TRMNTN_CD'] !=
↪   0)
            ].copy()
        closures['suspected_closure_year'] = year
        suspected_closure.append(closures[['FAC_NAME',
↪   'ZIP_CD','suspected_closure_year']])

        active = merged[(merged['_merge'] == 'both') &
↪   (merged['PGM_TRMNTN_CD'] == 0)][
            ['FAC_NAME', 'ZIP_CD', 'PRVDR_NUM']]

    suspected = pd.concat(suspected_closure, ignore_index=True)
    return suspected

suspected = check_closure(df)

print(f'There are {suspected.shape[0]} hospitals that fit this definition.')
```

```
There are 174 hospitals that fit this definition.
```

2.

```
suspected[['FAC_NAME','suspected_closure_year']].sort_values(by='FAC_NAME').head(10)
```

	FAC_NAME	suspected_closure_year
0	ABRAZO MARYVALE CAMPUS	2017
1	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2017
73	AFFINITY MEDICAL CENTER	2018
15	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2017
29	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2017
132	ALLIANCE LAIRD HOSPITAL	2019
147	ALLIANCEHEALTH DEACONESS	2019
109	ANNE BATES LEACH EYE HOSPITAL	2019
3	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	2017
11	BANNER CHURCHILL COMMUNITY HOSPITAL	2017

3.

```
def find_merger(df):
    potential_merger_list = []
    corrected_list = []
    # since we are gonna compare the number of active hospitals in the
     ↪ suspected year of closing to the number of active hospitals in the
     ↪ year after.
    # the first year that has suspected closure hospital is 2017
    # the following three steps help obtain the number of active hospital in
     ↪ each zipcode.
    sus_year = df[df["Year"]==2017]
    active = sus_year[sus_year["PGM_TRMNTN_CD"]==0]
    sus_agg=active.groupby('ZIP_CD').size().reset_index(name = 'count')

    for year in range(2018,2020):
        check = df[df["Year"]==year]
        check_active = check[check['PGM_TRMNTN_CD']==0]
        check_agg =
↪   check_active.groupby('ZIP_CD').size().reset_index(name='count')

        # merge sus_agg & check_agg on ZIP_CD to check whether decrease
        merged = sus_agg.merge(check_agg,
                               on='ZIP_CD',
                               how='left',
                               suffixes=('_sus', '_check'))

        # if doesn't decrease, the hospital will be selected on the
         ↪ remove_list
        # active providers also included in this list
        merged['count_check'].fillna(0, inplace=True)
        remove_list = merged[merged['count_check'] >= merged['count_sus']]

        # compare the remove_list with suspected list (we are in the loop of
         ↪ the year after the suspected year)
        suspected_that_year =
 ↪  suspected[suspected['suspected_closure_year']==(year-1)]
        merged_susp = suspected_that_year.merge(remove_list['ZIP_CD'],
                                                on='ZIP_CD',
                                                how='inner')


        saved = merged_susp[['FAC_NAME','ZIP_CD','suspected_closure_year']]
        potential_merger_list.append(saved)

        # This year's check_agg is next year's sus_agg
        sus_agg = check_agg

    # obtain the result
    merger_acq = pd.concat(potential_merger_list, ignore_index=True)

    return merger_acq

result = find_merger(df)
```

```
/var/folders/ql/ctwb9j8x6rb315jvsg9ht2140000gn/T/ipykernel_17169/1027238835.py:24:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves
as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  merged['count_check'].fillna(0, inplace=True)
/var/folders/ql/ctwb9j8x6rb315jvsg9ht2140000gn/T/ipykernel_17169/1027238835.py:24:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves
as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  merged['count_check'].fillna(0, inplace=True)
```

a.

```python
print(f"There are {result.shape[0]} hospitals fit this definition.")
```

```
There are 25 hospitals fit this definition.
```

b.

```python
number_left = suspected.shape[0]-result.shape[0]
print(f"After correction, {number_left} hospitals have left.")
```

```
After correction, 149 hospitals have left.
```

c.

```python
corr_list = suspected.merge(result,
                            how = 'left',
                            indicator = True)
corr_list =
↪  corr_list[corr_list['_merge']=='left_only'].drop(columns='_merge')
↪

corr_list.sort_values(by='FAC_NAME').head(10)
```

	FAC_NAME	ZIP_CD	suspected_closure_year
0	ABRAZO MARYVALE CAMPUS	85031	2017
73	AFFINITY MEDICAL CENTER	44646	2018

	FAC_NAME	ZIP_CD	suspected_closure_year
29	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662	2017
132	ALLIANCE LAIRD HOSPITAL	39365	2019
147	ALLIANCEHEALTH DEACONESS	73112	2019
109	ANNE BATES LEACH EYE HOSPITAL	33136	2019
3	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050	2017
11	BANNER CHURCHILL COMMUNITY HOSPITAL	89406	2017
42	BANNER PAYSON MEDICAL CENTER	85541	2018
153	BARIX CLINICS OF PENNSYLVANIA	19047	2019

3 Download Census zip code shapefile (10 pt)

1.

a.

The five file types are .dbf .prj .shp .shx .xml respectively.

(1).shp (Shape Format) Type: Geometry data
Content: Stores the geometric information of spatial features, such as the coordinates for points, lines, and polygons. This is the primary file in a Shapefile and contains the actual spatial data.

(2).shx (Shape Index Format) Type: Positional index
Content: Acts as an index for the .shp file, allowing quick access to the geometric features. It speeds up the process of locating and reading specific features within the .shp file.

(3).dbf (Attribute Format) Type: Attribute data
Content: Stores attribute data in a tabular format, similar to a database table. This file contains non-spatial information for each feature, such as names, population, or other characteristics, corresponding to the geometry in the .shp file.

(4).prj (Projection Format) Type: Projection and Coordinate Reference System (CRS)
Content: Defines the coordinate system used by the Shapefile, such as WGS84. This file describes how the spatial data is projected, allowing it to align properly with other geographic data.

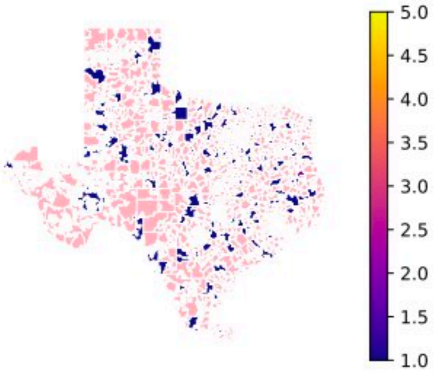
(5).xml (Metadata) Type: Metadata
Content: An optional file containing metadata about the Shapefile, such as the source, creation date, description, and other information about the dataset. It serves as documentation for the data.

b.

(1).shp: 837.5MB; (2).shx:265KB; (3).dbf:6.4MB; (4).prj:165 bytes; (5).xml 16KB

2.

```
file_path = '/Users/tang/Desktop/1Python
 ↪ II/ProblemSet4/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp'

geodata = gpd.read_file(file_path)
# create texas range
texas_range = list(range(75000, 80000)) + list(range(88500, 88600))
# filter for texas data
texas_geodata = geodata[geodata['ZCTA5'].astype(int).isin(texas_range)]


# obtain the num of hospital per zipcode in 2016
df_2016 = df[df['Year']==2016] # these are short-term hospitals.
df_2016 = df_2016[df_2016['PGM_TRMNTN_CD']==0] # these are active hospitals.

df_2016_agg = df_2016.groupby('ZIP_CD').size().reset_index(name='count')

# merge
texas_geodata_merged = texas_geodata.merge(df_2016_agg,
                                    left_on = 'ZCTA5',
                                    right_on = 'ZIP_CD',
                                    how = 'left')


# plot
texas_geodata_merged.plot(column="count",
 ↪ edgecolor="white",linewidth=0.2,legend=True,cmap="plasma",
missing_kwds={"color": "lightpink", "label": "No Data"})
plt.axis("off")
plt.show()
```



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

1.

```python
zips_all_centroids = geodata.copy()
zips_all_centroids['centroid']=zips_all_centroids.centroid
zips_all_centroids = zips_all_centroids[['ZCTA5','geometry','centroid']]
zips_all_centroids.head(3)
```

```
/var/folders/ql/ctwb9j8x6rb315jvsg9ht2140000gn/T/ipykernel_17169/4212993397.py:2:
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.

  zips_all_centroids['centroid']=zips_all_centroids.centroid
```

	ZCTA5	geometry	centroid
0	01040	POLYGON ((-72.62734 42.16203, -72.62764 42.162...	POINT (-72.64107 42.21257)
1	01050	POLYGON ((-72.95393 42.34379, -72.95385 42.343...	POINT (-72.86985 42.28786)
2	01053	POLYGON ((-72.68286 42.37002, -72.68287 42.369...	POINT (-72.71162 42.35349)

This GeoDataFrame has 33120 rows and 3 columns:

The first column is the 5-digit zip code;

The second column is the geometry, which contains the geometric representation of each zip code in the form of polygons or multipolygons which describes the spatial footprint of each ZCTA.

The third row is the centroid, which contains the geometric centroid of each zip code area represented as a point.

2.

```
# recall that the prefixes list of Texas is texas_range
# filter for Texas data
zips_texas_centroids =
↪ zips_all_centroids[zips_all_centroids['ZCTA5'].astype(int).isin(texas_range)]

# create range for Texas and bordering state
broader_range = list(range(70000, 80000)) + list(range(87000, 88600))
# filter for the broader area
zips_texas_borderstates_centroids =
↪ zips_all_centroids[zips_all_centroids['ZCTA5'].astype(int).isin(broader_range)]

# calculate the number of unique zip codes
unique_texas_zipcodes = zips_texas_centroids['ZCTA5'].nunique()
unique_texas_borderstates_zipcodes =
↪ zips_texas_borderstates_centroids['ZCTA5'].nunique()

print(f'There are {unique_texas_zipcodes} unique zip codes in the Texas
↪ subset; and {unique_texas_borderstates_zipcodes} in the second subset')
```

```
There are 1935 unique zip codes in the Texas subset; and 4057 in the second
subset
```

3.

```
# recall that the active hospital data was stored in df_2016_agg

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    df_2016_agg,
    left_on = 'ZCTA5',
    right_on = 'ZIP_CD',
    how = 'inner'
).drop(columns=['ZIP_CD'])

zips_withhospital_centroids.head()
```

	ZCTA5	geometry	centroid	count
0	70043	POLYGON ((-89.98424 29.96844, -89.97689 29.965...	POINT (-89.96276 29.94804)	1
1	70127	POLYGON ((-89.98738 30.05384, -89.98714 30.053...	POINT (-89.97675 30.02501)	1
2	70301	POLYGON ((-90.95365 29.85292, -90.94858 29.856...	POINT (-90.74089 29.8141)	1
3	70360	POLYGON ((-90.79653 29.53538, -90.80386 29.535...	POINT (-90.81028 29.58819)	2
4	70403	POLYGON ((-90.57342 30.44929, -90.57691 30.450...	POINT (-90.48388 30.48002)	2

I decide to use inner join; and merge on the zip code number.

4.

a.

```
# sample test

sample_zips = zips_texas_centroids.sample(10)

start_time = time.time()

# use spatial join to calculate the nearest distance
nearest_hospitals = gpd.sjoin_nearest(
    sample_zips, zips_withhospital_centroids, how="left",
 ↪  distance_col="distance"
)

end_time = time.time()
sample_time = end_time - start_time

print(f'It takes {round(sample_time,2)} seconds.')
```

```
/Users/tang/Desktop/1Python
II/ve/.venv/lib/python3.9/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.

  warnings.warn(

It takes 1.13 seconds.
```

Since the population size is around 200 times of the sample size, if I assume a linear consumption of time, I would estimate that it takes me 128s, around 2 minutes.

b.

```
# full
start_time = time.time()

# use spatial join to calculate the nearest distance
nearest_hospitals_full = gpd.sjoin_nearest(
    zips_texas_centroids, zips_withhospital_centroids, how="left",
 ↪  distance_col="distance"
)

end_time = time.time()
full_time = end_time - start_time

print(f'It takes {round(full_time,2)} seconds.')
```

```
/Users/tang/Desktop/1Python
II/ve/.venv/lib/python3.9/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.

  warnings.warn(

It takes 115.53 seconds.
```

This result is very much close to my estimation.

c.

```
with open('/Users/tang/Desktop/1Python
↪ II/ProblemSet4/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.prj', 'r')
↪ as file:
    prj_content = file.read()
    print(prj_content)
```

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,2
```

It is in the unit "Degree".

degree = (2 * Math.Pi * 6371004) / 360 * 0.000621371 = 69.09 miles.

5.

a.

```
print(zips_texas_centroids.crs)
```

```
EPSG:4269
```

It is in the "degree" unit.

b.

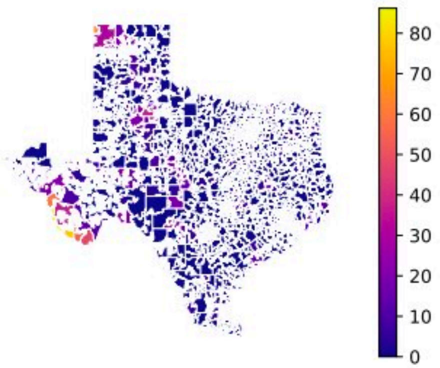
```
# convert unit degree into miles
nearest_hospitals_full['dis_miles'] = nearest_hospitals_full['distance'] *
↪ 69.09

# aggregate, calculate the avg distance.
avgdis_nearest =
↪ nearest_hospitals_full.groupby('ZCTA5_left')['dis_miles'].mean().reset_index(name
↪ = 'distance')

print(avgdis_nearest.head(10))

avg_distance_in_miles = avgdis_nearest['distance'].mean()

print(f'The average distance is {round(avg_distance_in_miles,2)} in miles.')
```

```
  ZCTA5_left  distance
0      75001       0.0
1      75002       0.0
2      75006       0.0
3      75007       0.0
4      75009       0.0
5      75010       0.0
6      75013       0.0
7      75019       0.0
8      75020       0.0
9      75021       0.0
The average distance is 2.76 in miles.
```

This value is smaller than my estimation. It seems that hospitals are very dense in Texas. Given that Texas is a very large state with relatiely low population density in rural areas, it doesn't seem to make sense.

c.

```
# merge

texas_dis_geodata = texas_geodata.merge(avgdis_nearest,
                                    left_on = 'ZCTA5',
                                    right_on = 'ZCTA5_left',
                                    how = 'left')

# plot
texas_dis_geodata.plot(column="distance",
 ↪  edgecolor="white",linewidth=0.2,legend=True,cmap="plasma",
missing_kwds={"color": "green", "label": "No Data"})
plt.axis("off")
plt.show()
```



5 Effects of closures on access in Texas (15 pts)

1.

```
# aggregate and count num of hospital by zip code
closure_by_zips = corr_list.groupby('ZIP_CD').size().reset_index(name =
 ↪  'count')


# filter for Texas; recall that the filter index for Texas is texas_range

closure_by_zips_texas =
 ↪  closure_by_zips[closure_by_zips['ZIP_CD'].astype(int).isin(texas_range)]

# display
closure_by_zips_texas[['ZIP_CD','count']]
```

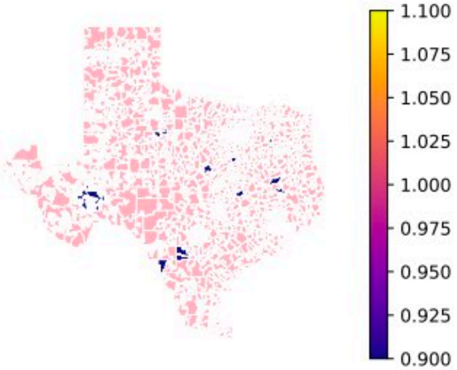
	ZIP_CD	count
95	75042	1
96	75051	1
97	75087	1
98	75140	1

	ZIP_CD	count
99	75235	1
100	75390	1
101	75601	1
102	75662	1
103	75835	1
104	75862	1
105	76502	1
106	76520	1
107	76531	1
108	76645	1
109	77035	1
110	77065	1
111	77429	1
112	78017	1
113	78061	1
114	78336	1
116	78613	1
117	78734	1
118	78834	1
119	79520	1
120	79529	1
121	79553	1
122	79735	1
123	79902	1

2.

```
# merge
texas_closure = texas_geodata.merge(closure_by_zips_texas,
                                    left_on = 'ZCTA5',
                                    right_on = 'ZIP_CD',
                                    how = 'left')


# plot
texas_closure.plot(column="count",
 ↪  edgecolor="white",linewidth=0.2,legend=True,cmap="plasma",
missing_kwds={"color": "lightpink", "label": "No Data"})
plt.axis("off")
plt.show()
```

```python
# calculate the number
num_dir = texas_closure['count'].notna().sum()
print(f'There are {num_dir} directy affected zipcodes in Texas.')
```



```
There are 28 directy affected zipcodes in Texas.
```

3.

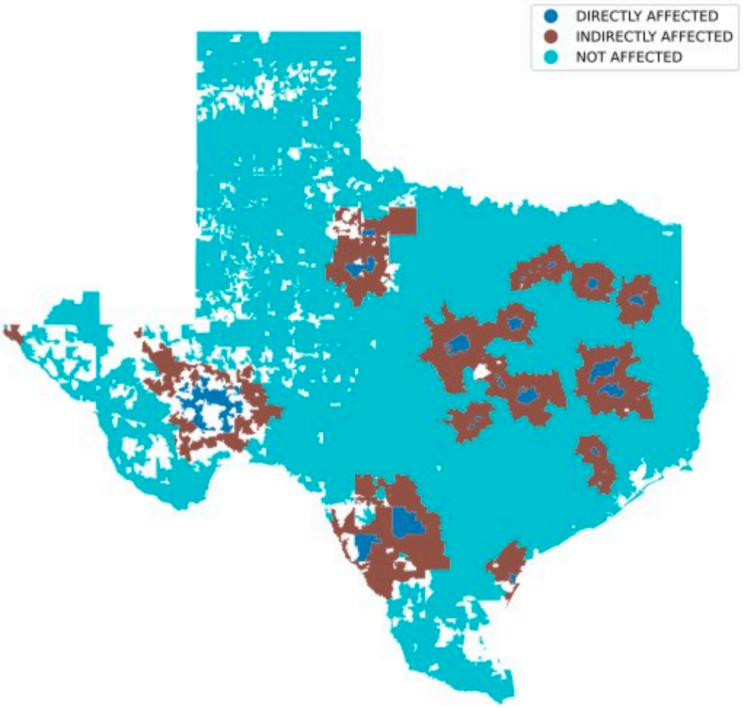
```python
# select the directly affected GeoDataFrame
dir_affct_gdf = texas_geodata.merge(closure_by_zips_texas,
                                    left_on = 'ZCTA5',
                                    right_on = 'ZIP_CD',
                                    how = 'inner')

# Project to meters
dir_affct_projected = dir_affct_gdf.to_crs(epsg=3857)
texas_geodata_projected = texas_geodata.to_crs(epsg=3857)


# set 10 miles (16093.4 meters) buffer
dir_affct_buffer = dir_affct_projected.copy()
dir_affct_buffer['geometry'] = dir_affct_buffer['geometry'].buffer(16093.4)

# spacial join
ind_affct_gdf = gpd.sjoin(texas_geodata_projected, dir_affct_buffer,
↪ how='inner', predicate='intersects')
```

```
# count the number of zipcode directly or indirectly affected.
num_ind = ind_affct_gdf['ZCTA5_left'].nunique()
print(f'There are {num_ind} indirecty affected zipcodes in Texas.')
```

```
There are 505 indirecty affected zipcodes in Texas.
```

4.

```
# create filter index
dir_zips_set = set(dir_affct_gdf['ZCTA5'])
ind_zips_set = set(ind_affct_gdf['ZCTA5_left']) - dir_zips_set

# create a projection function
def determine_status(zip_code):
    if zip_code in dir_zips_set:
        return "DIRECTLY AFFECTED"
    elif zip_code in ind_zips_set:
        return "INDIRECTLY AFFECTED"
    else:
        return "NOT AFFECTED"

texas_status = texas_geodata_projected.copy()
texas_status['status'] = texas_status['ZCTA5'].apply(determine_status)

# plot
dissolved = texas_status.dissolve(by='status').reset_index()


fig, ax = plt.subplots(1,1, figsize = (10,10))
dissolved.plot(column="status", legend=True, ax=ax)
plt.axis("off")
plt.show()
```



Reflecting on the exercise (10 pts)

Partner1: The 'first-path' method could be imperfect in several cases. For instance, a new hospital was constructed in the specific zip code where an old hospital was suspected closure. In another example if a hospital moved from one zip code to another, but the distance is not too far, like from 57th st Hyde Park to 53rd st Hyde Park and zip code changes from 60637 to 60615, should not be considered as closure. To do a better job, we could probably develop the variable 'PGM_TRMNT_CD' more and see closer to the termination status, or we could use a more detailed variable 'ST_ADR' (street address) to analysis closure and use zip codes only for clustering purpose.

Partner2: The reflection of changes in zip-code-level access to hospitals is generally good, but could be considered as a 'naive' reflection. In real life, whether the closure of a hospital affects residents is depend on variables such as distance and commute convenience. To improve this measure, we can calculate directly the distance between each hospital and use KNN method to find out the 'influential region' of each hospital. After that can we analysis the impacts of hospital closures.