PS4

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 (Ruyu Zhang and ruyuzhang)
    - Partner 2 (Luyao Guo and gluyao)
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: ***R.Z L.G***
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** Late coins left after submission: **1**
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. I put six variables: PRVDR_CTGRY_SBTYP_CD (subtype of provider), PRVDR_CTGRY_CD (type of provider), FAC_NAME (name of provider), PRVDR_NUM (CMS Certification Number), PGM_TRMNTN_CD (current termination status), ZIP_CD(zipcode).
2.

```
import warnings
import altair as alt
import pandas as pd
import os

base_path = "/Users/ruyuzhang/Desktop/PPHA
 ↪  30538/problem-set-4-luyao-ruyu/dataset"
path2016 = os.path.join(base_path, "pos2016.csv")

df_2016 = pd.read_csv(path2016)

alt.renderers.enable("png")
warnings.filterwarnings('ignore')
```

a.

```
# subset facilities with provider type code 01 and subtype code 01
hospital_2016 = df_2016[(df_2016['PRVDR_CTGRY_CD'] == 1)
                        & (df_2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]

# the number of hospitals reported in this data
hospital_count_2016 = hospital_2016.shape[0]
print(f"There are {
      hospital_count_2016} short-term hospitals reported in this data.")
```

```
There are 7245 short-term hospitals reported in this data.
```

I think the number doesn't make sense since it may include closed hospitals.

b.

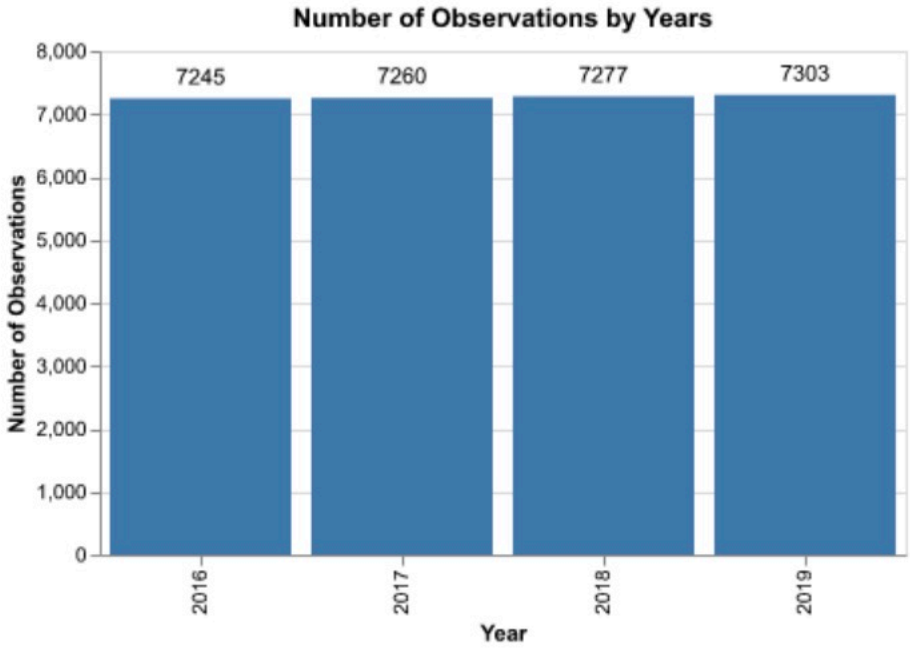
According to the article, there are nearly 5000 hospitals in the United States in 2016. Citation:https://www.kff.org/report-section/a-look-at-rural-hospital-closures-and-implications-for-access-to-care-three-case-studies-issue-brief/

Possible reasons: Variations in defining "short-term hospital" can lead to differences. For instance, some sources might exclude specialty hospitals or certain government-operated facilities. Also, the dataset might include facilities beyond the U.S., such as territories or military hospitals, which could inflate the count.

3.

```
path2017 = os.path.join(base_path, "pos2017.csv")
df_2017 = pd.read_csv(path2017, encoding="ISO-8859-1")

path2018 = os.path.join(base_path, "pos2018.csv")
df_2018 = pd.read_csv(path2018, encoding="ISO-8859-1")

path2019 = os.path.join(base_path, "pos2019.csv")
df_2019 = pd.read_csv(path2019, encoding="ISO-8859-1")

# subset facilities in files 2017-2019
hospital_2017 = df_2017[(df_2017['PRVDR_CTGRY_CD'] == 1)
                        & (df_2017['PRVDR_CTGRY_SBTYP_CD'] == 1)]
hospital_2017['Year'] = 2017
hospital_2018 = df_2018[(df_2018['PRVDR_CTGRY_CD'] == 1)
                        & (df_2018['PRVDR_CTGRY_SBTYP_CD'] == 1)]
hospital_2018['Year'] = 2018
hospital_2019 = df_2019[(df_2019['PRVDR_CTGRY_CD'] == 1)
                        & (df_2019['PRVDR_CTGRY_SBTYP_CD'] == 1)]
hospital_2019['Year'] = 2019
hospital_2016['Year'] = 2016

# append 4 datasets together
hospital_all = pd.concat(
    [hospital_2016, hospital_2017, hospital_2018, hospital_2019,],
 ↪  ignore_index=True)

# plot the number of observations by years
year_all = hospital_all['Year'].value_counts().reset_index()
year_all.columns = ['Year', 'Count']
```

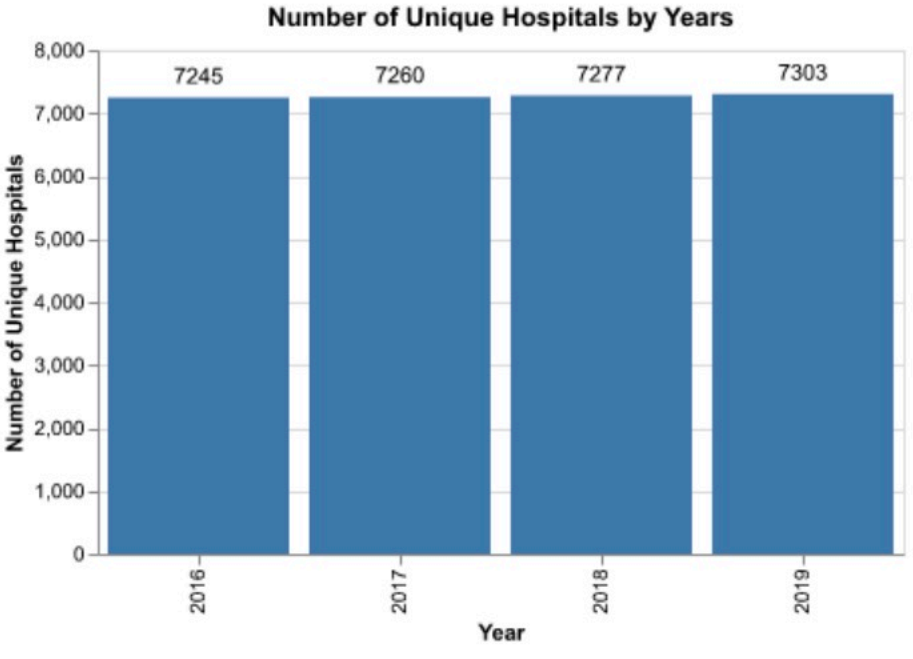
```
chart_raw = alt.Chart(year_all).mark_bar().encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Count:Q', title='Number of Observations',),
    tooltip=['Year', 'Count']
).properties(
    title='Number of Observations by Years',
    width=400,
    height=250
)

# add text labels
text_labels = chart_raw.mark_text(
    align='center',
    baseline='bottom',
    dy=-5,
    color='black'
).encode(
    text=alt.Text('Count:Q', format='.0f')
)

chart_all = chart_raw + text_labels

chart_all
```



4. a.

```
# plot number of unique hospitals per year
unique_hospitals = hospital_all.groupby(
    'Year')['PRVDR_NUM'].nunique().reset_index()
unique_hospitals.columns = ['Year', 'Unique_Hospitals']

chart_raw_2 = alt.Chart(unique_hospitals).mark_bar().encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Unique_Hospitals:Q', title='Number of Unique Hospitals',),
    tooltip=['Year', 'Unique_Hospitals']
).properties(
    title='Number of Unique Hospitals by Years',
    width=400,
    height=250
)

# add text labels
text_labels_2 = chart_raw_2.mark_text(
    align='center',
    baseline='bottom',
    dy=-5,
    color='black'
```

```
).encode(
    text=alt.Text('Unique_Hospitals:Q', format='.0f')
)

chart_unique = chart_raw_2 + text_labels_2

chart_unique
```



b. These two plots are identical, indicating that each hospital appears only once per year, with no duplicate entries. Also, the similarity in counts across years implies a stable and consistent reporting process, with minimal fluctuations in the number of short-term hospitals tracked each year.

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

1.

```
files = ["pos2016.csv", "pos2017.csv", "pos2018.csv", "pos2019.csv"]
years = [2016, 2017, 2018, 2019]

# Filter active short-term hospitals
hospital_2016 = df_2016[(df_2016['PRVDR_CTGRY_CD'] == 1) &
```

```
                        (df_2016['PRVDR_CTGRY_SBTYP_CD'] == 1) &
                        (df_2016['PGM_TRMNTN_CD'] == 0)]
active_2016 = hospital_2016[['FAC_NAME', 'ZIP_CD', 'PRVDR_NUM']]

hospital_status = active_2016.set_index('PRVDR_NUM').copy()
hospital_status['Closure_Year'] = None

# check  hospital closures
for year, df in zip([2017, 2018, 2019], [df_2017, df_2018, df_2019]):
    year_active = df[(df['PRVDR_CTGRY_CD'] == 1) &
                     (df['PRVDR_CTGRY_SBTYP_CD'] == 1) &
                     (df['PGM_TRMNTN_CD'] == 0)]['PRVDR_NUM']

    closed_in_year = hospital_status.index.difference(year_active)
    hospital_status.loc[closed_in_year,
                        'Closure_Year'] = hospital_status.loc[closed_in_year,
                         ↪  'Closure_Year'].fillna(year)
all_closed = hospital_status[hospital_status['Closure_Year'].notna(
)].reset_index()
num_closures = all_closed.shape[0]
print(f"Hospital closures: {num_closures}")
```

```
Hospital closures: 174
```

2.

```
sorted_closures = all_closed.sort_values(by='FAC_NAME')
sorted_closures[['FAC_NAME', 'Closure_Year']].head(10)
```

	FAC_NAME	Closure_Year
4	ABRAZO MARYVALE CAMPUS	2017
10	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2017
97	AFFINITY MEDICAL CENTER	2018
80	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2017
140	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2017
62	ALLIANCE LAIRD HOSPITAL	2019
101	ALLIANCEHEALTH DEACONESS	2019
26	ANNE BATES LEACH EYE HOSPITAL	2019
21	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	2017
69	BANNER CHURCHILL COMMUNITY HOSPITAL	2017

3.

```
# Create DataFrame
active_counts_by_year = {
    2016: df_2016,
    2017: df_2017,
    2018: df_2018,
    2019: df_2019
}

zip_counts = pd.DataFrame()
for year, df in active_counts_by_year.items():
    active_counts = df[(df['PRVDR_CTGRY_CD'] == 1) &
                       (df['PRVDR_CTGRY_SBTYP_CD'] == 1) &
                       (df['PGM_TRMNTN_CD'] ==
 ↪  0)].groupby('ZIP_CD')['PRVDR_NUM'].count().reset_index()
    active_counts.columns = ['ZIP_CD', f'Active_Count_{year}']
    if zip_counts.empty:
        zip_counts = active_counts
    else:
        zip_counts = zip_counts.merge(
            active_counts, on='ZIP_CD', how='outer').fillna(0)

# Identify potential mergers
potential_mergers = []

for _, row in all_closed.iterrows():
    zip_code = row['ZIP_CD']
    closure_year = row['Closure_Year']
    if closure_year < 2019:
        current_count = zip_counts.loc[zip_counts['ZIP_CD'] ==
                                       zip_code,
 ↪  f'Active_Count_{closure_year}'].values[0]
        next_year_count = zip_counts.loc[zip_counts['ZIP_CD'] == zip_code,
 ↪  f'Active_Count_{
            closure_year + 1}'].values[0]
        if next_year_count >= current_count:
            potential_mergers.append(row)

# Create DataFrame for potential merger
potential_mergers_df = pd.DataFrame(potential_mergers)
num_potential_mergers = potential_mergers_df.shape[0]
```

```
print(f"Potentially fit the definition of being a merger: {
      num_potential_mergers}")
```

```
Potentially fit the definition of being a merger: 97
```

97 hospitals potentially fit

b.

```
# Remove potential merger from the DataFrame
corrected_closures = all_closed[~all_closed['PRVDR_NUM'].isin(
    potential_mergers_df['PRVDR_NUM'])]
corrected_closures_count = corrected_closures.shape[0]
print(
    f"Number of hospitals left after correcting for potential
     ↪ mergers/acquisitions: {corrected_closures}")
```

```
Number of hospitals left after correcting for potential mergers/acquisitions:
PRVDR_NUM                                          FAC_NAME   ZIP_CD  \
0      010032                                WEDOWEE HOSPITAL  36278.0
1      010047                      GEORGIANA MEDICAL CENTER  36033.0
8      050018                PACIFIC ALLIANCE MEDICAL CENTER  90012.0
9      050153                             O'CONNOR HOSPITAL  95128.0
14     050688                 SAINT LOUISE REGIONAL HOSPITAL  95020.0
..        ...                                             ...      ...
165    670083                          TEXAS GENERAL HOSPITAL  75051.0
166    670087  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0
169    670094           LITTLE RIVER HEALTHCARE CAMERON HOSPITAL  76520.0
171    670097                   BAYLOR EMERGENCY MEDICAL CENTER  75087.0
173    670117                 TEXAS GENERAL HOSPITAL- VZRMC LP  75140.0

    Closure_Year
0           2019
1           2019
8           2019
9           2019
14          2019
..           ...
165         2019
166         2019
169         2019
171         2019
```

```
173         2019

[77 rows x 4 columns]
```

77 hospitals are left as suspected true closures.

c.

```
# Sort suspected closures by facility name
sorted_remaining_closures = corrected_closures.sort_values(
    by='FAC_NAME')[['FAC_NAME', 'ZIP_CD', 'Closure_Year']]
print("Sorted list of the first 10 corrected hospital closures by name:")
print(sorted_remaining_closures.head(10))
```

```
Sorted list of the first 10 corrected hospital closures by name:
                                                FAC_NAME   ZIP_CD Closure_Year
62                                ALLIANCE LAIRD HOSPITAL  39365.0         2019
101                              ALLIANCEHEALTH DEACONESS  73112.0         2019
26                         ANNE BATES LEACH EYE HOSPITAL  33136.0         2019
115                        BARIX CLINICS OF PENNSYLVANIA  19047.0         2019
171                      BAYLOR EMERGENCY MEDICAL CENTER  75087.0         2019
166  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0         2019
98                            BELMONT COMMUNITY HOSPITAL  43906.0         2019
67                               BIG SKY MEDICAL CENTER  59716.0         2019
65                  BLACK RIVER COMMUNITY MEDICAL CENTER  63901.0         2019
142                        CARE REGIONAL MEDICAL CENTER  78336.0         2019
```

Download Census zip code shapefile (10 pt)

1. a. .shp is Shapefile. This file contains the geometry data (points, lines, or polygons) that represent spatial features. .dbf is Database File. This file stores attribute data in tabular form, linked to each spatial feature in the .shp file. .shx is Shape Index Format. This file holds an index of the geometry in the .shp file, allowing for faster access to spatial features. .prj is Projection Format. This file contains coordinate system and projection information, which defines how the spatial data aligns with the Earth's surface. .xml is Metadata. This file holds metadata about the shapefile dataset, describing its source, creation details, and other contextual information.

   b. .shp file: 837.5 MB .dbf file: 6.4 MB .shx file: 265 KB .prj file: 165 bytes .xml file: 16 KB

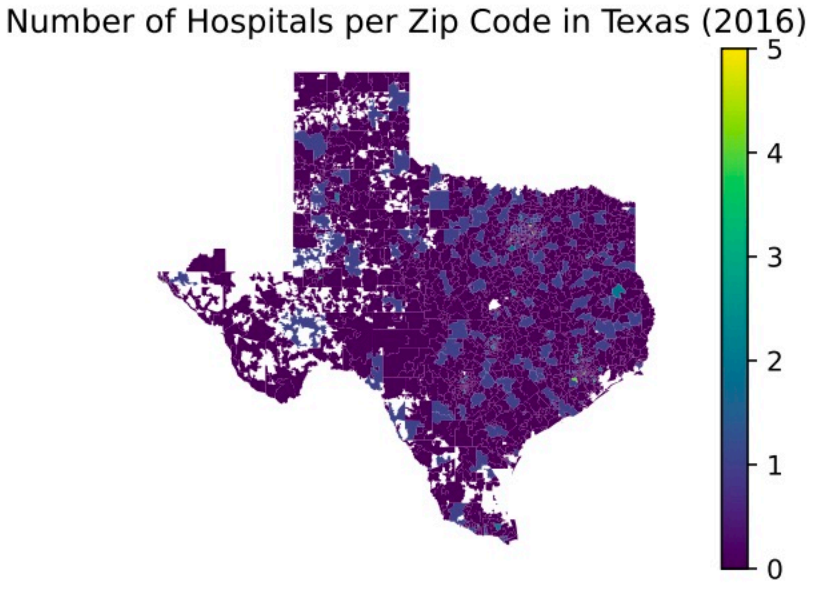
   The .shp file is the largest, containing the spatial data, followed by the .dbf file for attribute data. The .shx, .prj, and .xml files are much smaller, primarily containing indexing, projection, and metadata, respectively.

2.

```
import matplotlib.pyplot as plt
import geopandas as gpd
import shapely
from shapely import Polygon, Point

pathshp = os.path.join(base_path, "gz_2010_us_860_00_500k.shp")
df_shp = gpd.read_file(pathshp)

# Filter Texas zip codes that start with '75', '76', '77', '78', or '79'
df_texas = df_shp[df_shp["ZCTA5"].str.startswith(
    ('75', '76', '77', '78', '79'))]
df_texas['ZCTA5'] = df_texas['ZCTA5'].astype(str)

# calculate the number of hospitals per zip code in 2016
hospital_zip = hospital_2016.groupby(
    'ZIP_CD').size().reset_index(name='hospital_count')
hospital_zip['ZIP_CD'] = hospital_zip['ZIP_CD'].astype(
    str).str.replace(r'\.0$', '', regex=True).str.zfill(5)

print(hospital_zip)

# plot a choropleth
map_texas = df_texas.merge(
    hospital_zip, left_on='ZCTA5', right_on='ZIP_CD', how='left')
map_texas['hospital_count'] = map_texas['hospital_count'].fillna(0)

map_texas.plot(column='hospital_count', legend=True).set_axis_off()
plt.title('Number of Hospitals per Zip Code in Texas (2016)', fontsize=12)

plt.show()
```

```
     ZIP_CD  hospital_count
0     00603               1
1     00613               1
2     00614               1
3     00631               1
4     00641               1
...     ...             ...
3062  99559               1
3063  99645               1
```

```
3064   99669                1
3065   99701                1
3066   99801                1

[3067 rows x 2 columns]
```



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

1.

```
from shapely.geometry import Point, Polygon
from shapely.ops import nearest_points
import matplotlib.pyplot as plt
import time

if df_shp.crs.is_geographic:
    df_shp = df_shp.to_crs("EPSG:3857")

# Calculate centroids for all ZIP codes
df_shp['centroid'] = df_shp.geometry.centroid
zips_all_centroids = gpd.GeoDataFrame(
    df_shp[['ZCTA5']], geometry=df_shp['centroid'])
zips_all_centroids.columns = ['ZIP_CD', 'geometry']
```

```
print(f"Dimensions of zips_all_centroids: {zips_all_centroids.shape}")
print("Columns: ZIP_CD (ZIP code identifier), geometry (centroid of the ZIP
 ↪  code)")
```

```
Dimensions of zips_all_centroids: (33120, 2)
Columns: ZIP_CD (ZIP code identifier), geometry (centroid of the ZIP code)
```

Dimensions: The GeoDataFrame contains 33,120 rows and 2 columns. Columns: ZIP_CD: This column represents the ZIP code identifier for each area. geometry: This column contains the centroid point (latitude and longitude coordinates) of each ZIP code area.

2.

```
# Create a subset of zips_all_centroids
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZIP_CD'].str.startswith(
    ('75', '76', '77', '78', '79'))]

# Define ZIP code prefixes for Texas and its bordering states
border_state_prefixes = (
    # New Mexico
    '870', '871', '872', '873', '874', '875', '876', '877', '878', '879',
     ↪  '880', '881', '882', '883', '884',
    # Oklahoma
    '73', '74',
    # Arkansas
    '716', '717', '718', '719', '720', '721', '722', '723', '724', '725',
     ↪  '726', '727', '728', '729',
    # Louisiana
    '700', '701', '702', '703', '704', '705', '706', '707', '708', '709',
     ↪  '710', '711', '712', '713', '714', '715',
    # Texas prefixes
    '75', '76', '77', '78', '79'
)

zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZIP_CD'].str.startswith(
    border_state_prefixes)]
print(f"Number of unique ZIP codes in Texas: {
      zips_texas_centroids['ZIP_CD'].nunique()}")
print(f"Number of unique ZIP codes in Texas and bordering states: {
      zips_texas_borderstates_centroids['ZIP_CD'].nunique()}")
```

```
Number of unique ZIP codes in Texas: 1935
Number of unique ZIP codes in Texas and bordering states: 4057
```

Number of unique ZIP codes in Texas: 1935 Number of unique ZIP codes in Texas and bordering states: 4057

3.

```
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospital_zip, on='ZIP_CD', how='left'
)

# Check and filter based on the correct column name
zips_withhospital_centroids = zips_withhospital_centroids[
    zips_withhospital_centroids['hospital_count'] > 0]
```

I used a left merge to create the zips_withhospital_centroids GeoDataFrame. The merge was based on the ZIP_CD column to ensure that all ZIP codes from zips_texas_borderstates_centroids were included, with matches to hospital_zip data indicating ZIP codes that had at least one hospital in 2016. After the merge, I filtered the GeoDataFrame to keep only those ZIP codes where the hospital_count was greater than 0.

4. a.

```
import time

# Subset for 10 ZIP codes and time the calculation
start_time = time.time()
sample_zips = zips_texas_centroids.head(10)
sample_zips['Nearest_Hospital_Distance'] = sample_zips.apply(
    lambda row: row.geometry.distance(
        nearest_points(
            row.geometry, zips_withhospital_centroids.unary_union)[1]
    ),
    axis=1
)
end_time = time.time()

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

# Estimate time for the full calculation
```

```
estimated_time_full = time_taken * (zips_texas_centroids.shape[0] / 10)
print(f"Estimated time for full calculation: {
      estimated_time_full:.2f} seconds")
```

```
Time taken for 10 ZIP codes: 0.01 seconds
Estimated time for full calculation: 1.72 seconds
```

b.

```
start_time_full = time.time()
zips_texas_centroids['Nearest_Hospital_Distance'] =
 ↪  zips_texas_centroids.apply(
    lambda row: row.geometry.distance(
        nearest_points(
            row.geometry, zips_withhospital_centroids.unary_union)[1]
    ),
    axis=1
)
end_time_full = time.time()

print(f"Time taken for full calculation: {
      end_time_full - start_time_full:.2f} seconds")
```

```
Time taken for full calculation: 1.13 seconds
```

c.

```
print(df_shp.crs)
zips_texas_centroids['Distance_Miles'] =
 ↪  zips_texas_centroids['Nearest_Hospital_Distance'] * 0.000621371
```

```
EPSG:3857
```

5.

a. 1 meter=0.000621371 miles. The unit used in the calculation is meters

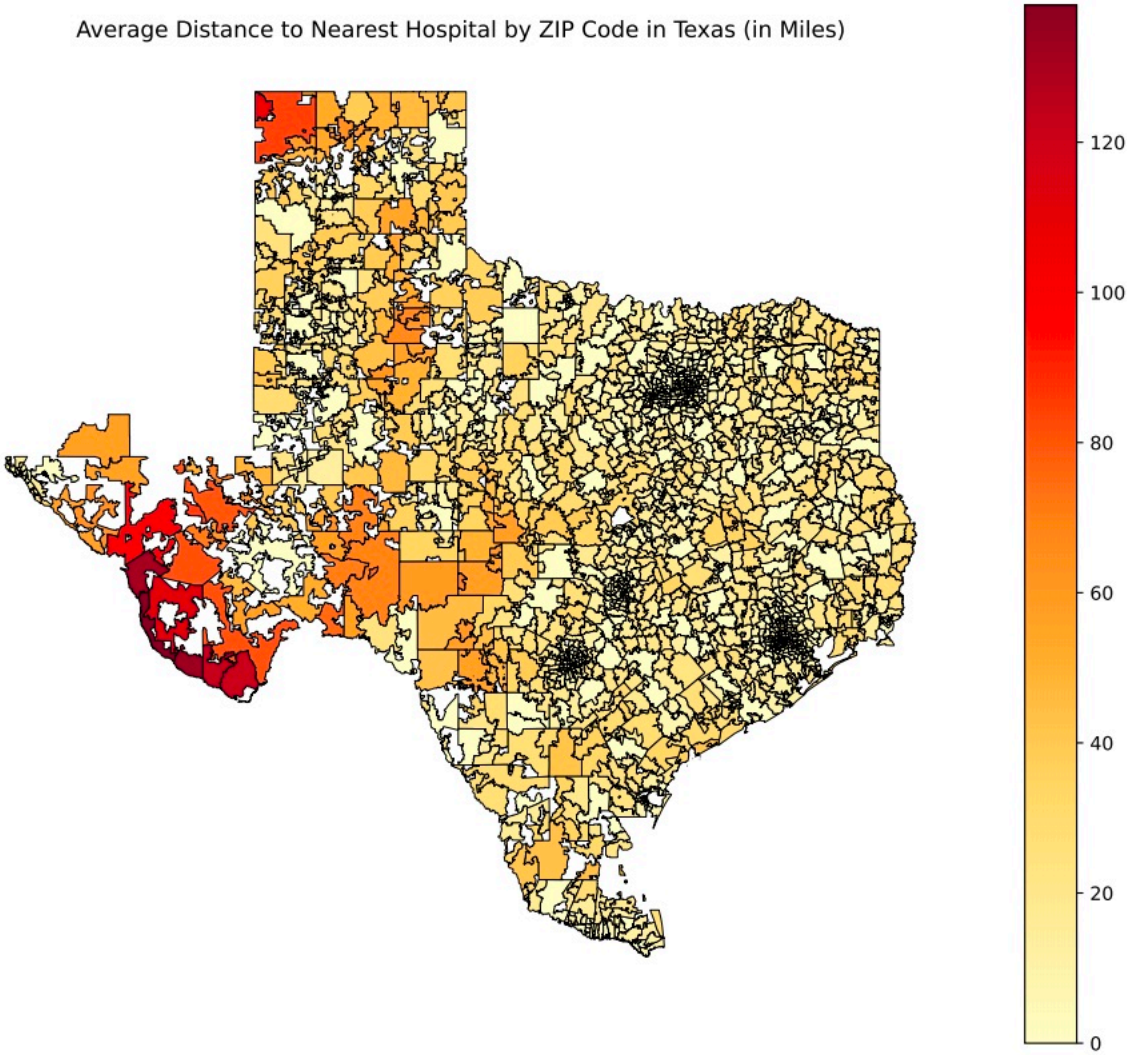
b.

```
average_distance = zips_texas_centroids['Distance_Miles'].mean()
print(f"Average distance to the nearest hospital in miles: {
      average_distance:.2f}")
```

```
Average distance to the nearest hospital in miles: 15.77
```

make sense, It is convenient for local citizens to find the hospitals which average distance shorter than 20 miles

```
pathshp = os.path.join(base_path, "gz_2010_us_860_00_500k.shp")
df_shp = gpd.read_file(pathshp)

# Filter Texas zip codes that start with '75', '76', '77', '78', or '79'
df_texas = df_shp[df_shp["ZCTA5"].str.startswith(
    ('75', '76', '77', '78', '79'))]
df_texas['ZCTA5'] = df_texas['ZCTA5'].astype(str)

# Merge the distance data for plotting
map_texas = df_texas.merge(zips_texas_centroids[[
                           'ZIP_CD', 'Distance_Miles']], left_on='ZCTA5',
                            ↪ right_on='ZIP_CD', how='left')
map_texas['Distance_Miles'] = map_texas['Distance_Miles'].fillna(0)

# Plot the choropleth map with consistent styling
fig, ax = plt.subplots(figsize=(12, 10))
map_texas.plot(column='Distance_Miles', cmap='YlOrRd',
               legend=True, linewidth=0.5, ax=ax, edgecolor='black')

# Set title and remove axes for a unified style
ax.set_title(
    'Average Distance to Nearest Hospital by ZIP Code in Texas (in Miles)',
     ↪ fontsize=12)
ax.set_axis_off()

# Adjust legend properties for consistency
legend = ax.get_legend()
if legend:
    # Move the legend to the side for better visualization
    legend.set_bbox_to_anchor((1.05, 1))
    legend.set_title('Distance (Miles)')

plt.show()
```



Effects of closures on access in Texas (15 pts)

1.

```
# Filter closures for Texas zip codes that start with '75', '76', '77', '78',
 ↪  or '79'
corrected_closures['ZIP_CD'] = corrected_closures['ZIP_CD'].astype(
    str).str.replace(r'\.0$', '', regex=True).str.zfill(5)
closures_texas =
 ↪  corrected_closures[corrected_closures['ZIP_CD'].str.startswith(
    ('75', '76', '77', '78', '79'))]
```

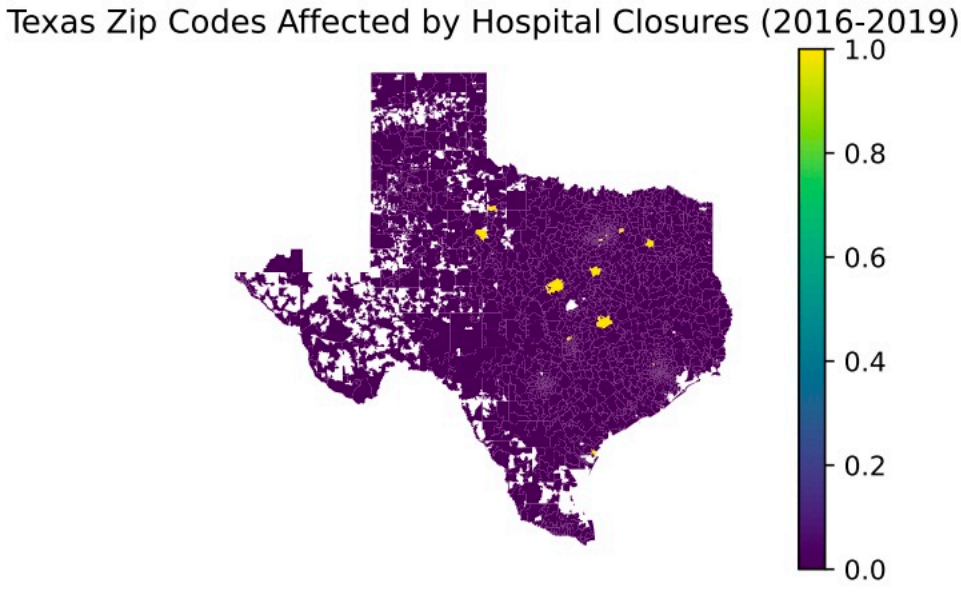
```
# Count the number of closures per Texas zip code
closures_texas_count = closures_texas.groupby(
    'ZIP_CD').size().reset_index(name='closure_count')

closures_texas_count
```

	ZIP_CD	closure_count
0	75051	1
1	75087	1
2	75140	1
3	75235	1
4	75390	1
5	76520	1
6	76531	1
7	76645	1
8	77065	1
9	78336	1
10	78613	1
11	79520	1
12	79529	1
13	79902	1

2.

```
# Merge shapefile data with closure counts
map_closures_texas = df_texas.merge(
    closures_texas_count, left_on='ZCTA5', right_on='ZIP_CD', how='left')
map_closures_texas['closure_count'] =
 ↪  map_closures_texas['closure_count'].fillna(
    0)

# plot a choropleth
map_closures_texas.plot(column='closure_count', legend=True).set_axis_off()
plt.title('Texas Zip Codes Affected by Hospital Closures (2016-2019)',
 ↪  fontsize=12)
plt.show()

# number of directly affected zip codes in Texas
directly_zip_count = closures_texas_count.shape[0]
```

```
print(f"The number of directly affected zip codes in Texas is {
      directly_zip_count}.")
```




```
The number of directly affected zip codes in Texas is 14.
```

3.

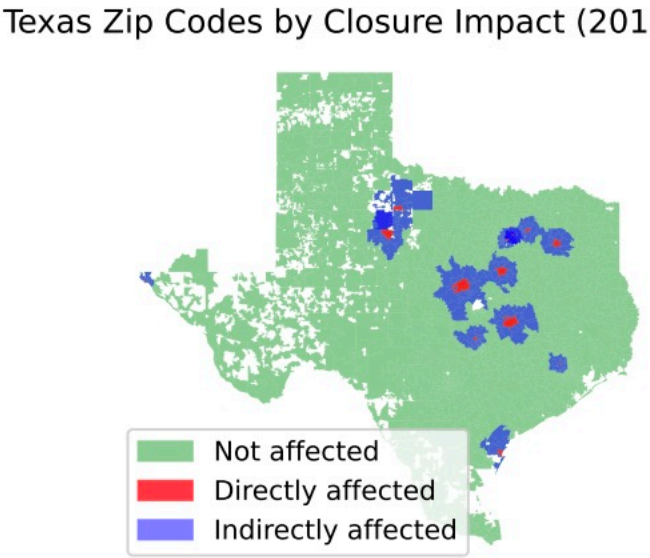
```
# create a GeoDataFrame of the directly affected zip codes
direct_closures_texas_gdf = df_texas.merge(
    closures_texas_count, left_on='ZCTA5', right_on='ZIP_CD', how='inner')

# create a 10-mile buffer around them (approximately 16093 meters)
closures_buffer = direct_closures_texas_gdf.copy()
closures_buffer['geometry'] = closures_buffer.geometry.buffer(10/69)

# do a spatial join with the overall Texas zip code shapefile.
indirectly_zip = gpd.sjoin(df_texas, closures_buffer,
                           how="inner", predicate="intersects")

# Exclude zip codes that are directly affected
indirectly_zip = indirectly_zip[~indirectly_zip['ZCTA5_left'].isin(
    direct_closures_texas_gdf['ZCTA5'])]

# number of indirectly affected zip codes in Texas
```

```
indirectly_zip_count = indirectly_zip.shape[0]
print(f"The number of indirectly affected zip codes in Texas is {
      indirectly_zip_count}.")
```

```
The number of indirectly affected zip codes in Texas is 420.
```

4.

```
import matplotlib.patches as mpatches

fig, ax = plt.subplots()
# Plot all Texas zip codes as "Not affected"
df_texas.plot(ax=ax, color="green", alpha=0.4, label="Not affected")
# Plot directly affected zip codes
direct_closures_texas_gdf.plot(
    ax=ax, color="red", alpha=0.7, label="directly affected").set_axis_off()
# Plot indirectly affected zip codes
indirectly_zip.plot(ax=ax, color="blue", alpha=0.5,
                    label="indirectly affected").set_axis_off()

# add legend and title
not_affected_patch = mpatches.Patch(
    color="green", alpha=0.4, label="Not affected")
directly_affected_patch = mpatches.Patch(
    color="red", alpha=0.7, label="Directly affected")
indirectly_affected_patch = mpatches.Patch(
    color="blue", alpha=0.5, label="Indirectly affected")
plt.legend(handles=[not_affected_patch, directly_affected_patch,
           indirectly_affected_patch], loc="lower left")
plt.title('Texas Zip Codes by Closure Impact (2016-2019)', fontsize=12)


plt.show()
```



Reflecting on the exercise (10 pts)

a. The current "first-pass" method of checking closures by filtering ZIP codes where the number of hospitals doesn't decrease has its flaws. Just because the total number of hospitals stays the same doesn't mean there weren't closures—it could be that some hospitals shut down and others opened separately. One way to improve this is to look at the names of the hospitals. If a new hospital appears with the same or very similar name as a previously closed one, it's likely a merger or a rebranding, not a true closure. This approach would help us better identify real closures versus situations where hospitals are just changing names or merging.

b. The current 10-mile radius method is a useful approximation but has limitations:

   1. Affects urban and rural areas differently, as rural residents often have fewer alternatives within 10 miles.
   2. Population density should be considered, since closures in high-density areas may affect more people.