PS4: Spatial

AUTHOR
Peter Ganong, Maggie Shi, and Fatima Irfan

PUBLISHED
November 4, 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):
    - Partner 2 (name and cnet ID):
3. Partner 1 will accept the `ps4` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **__** **__**
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.

Policy Background

Hospital closures have been on the rise in recent years. A hospital closure can have significant public health implications as they can increase travel distances for patients and reduce access to critical care, especially for already-vulnerable populations. Before starting, please read this issue brief from the Kaiser Family Foundation (link) to understand the broader social and policy context. We will first look at closures nationally and then focus on Texas, a state with large urban metro areas but also with a substantial share of population living in relatively remote rural areas.

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

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

- First download the Provider of Services (POS) file for Q4 2016.
    - Data: data.cms.gov (link)
    - The data dictionary is located at the bottom of the page as "Provider of Services File - Hospital & Non-Hospital Facilities Data Dictionary"

1. (Partner 1) This is a fairly large dataset and we won't be using most of the variables. Read through the rest of the problem set and look through the data dictionary to identify which variables you will need to complete the exercise, and use the tool on data.cms.gov into restrict to those variables ("Manage Columns") before exporting ("Export"). Download this for 2016 and call it `pos2016.csv`. What are the variables you pulled?
    - ***Answer****: Variables to pull: provider category type (`prvdr_ctgry_cd`), provider category subtype (`prvdr_ctgry_sbtyp_cd`), zip code (`zip_cd`), termination code (`pgm_trmntn_cd`), facility name (`fac_name`) and CMS certification number (`prvdr_num`) are required.*

```
import altair as alt
```

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
pos2016 = pd.read_csv('pos2016.csv')
```

2. (Partner 1) Import your `pos2016.csv` file. We want to focus on short-term hospitals. These are identified as facilities with provider type code `01` and subtype code `01`. Subset your data to these facilities. How many hospitals are reported in this data? Does this number make sense? Cross-reference with other sources and cite the number you compared it to. If it differs, why do you think it could differ?
    - ***Answer****: there are 7245 in the CMS data, while the American Hospital Association reports about 6120 (https://www.aha.org/statistics/fast-facts-us-hospitals). The difference could be due to closures between 2016 and now, or it could be due to slight differences in definitions between the two sources*

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

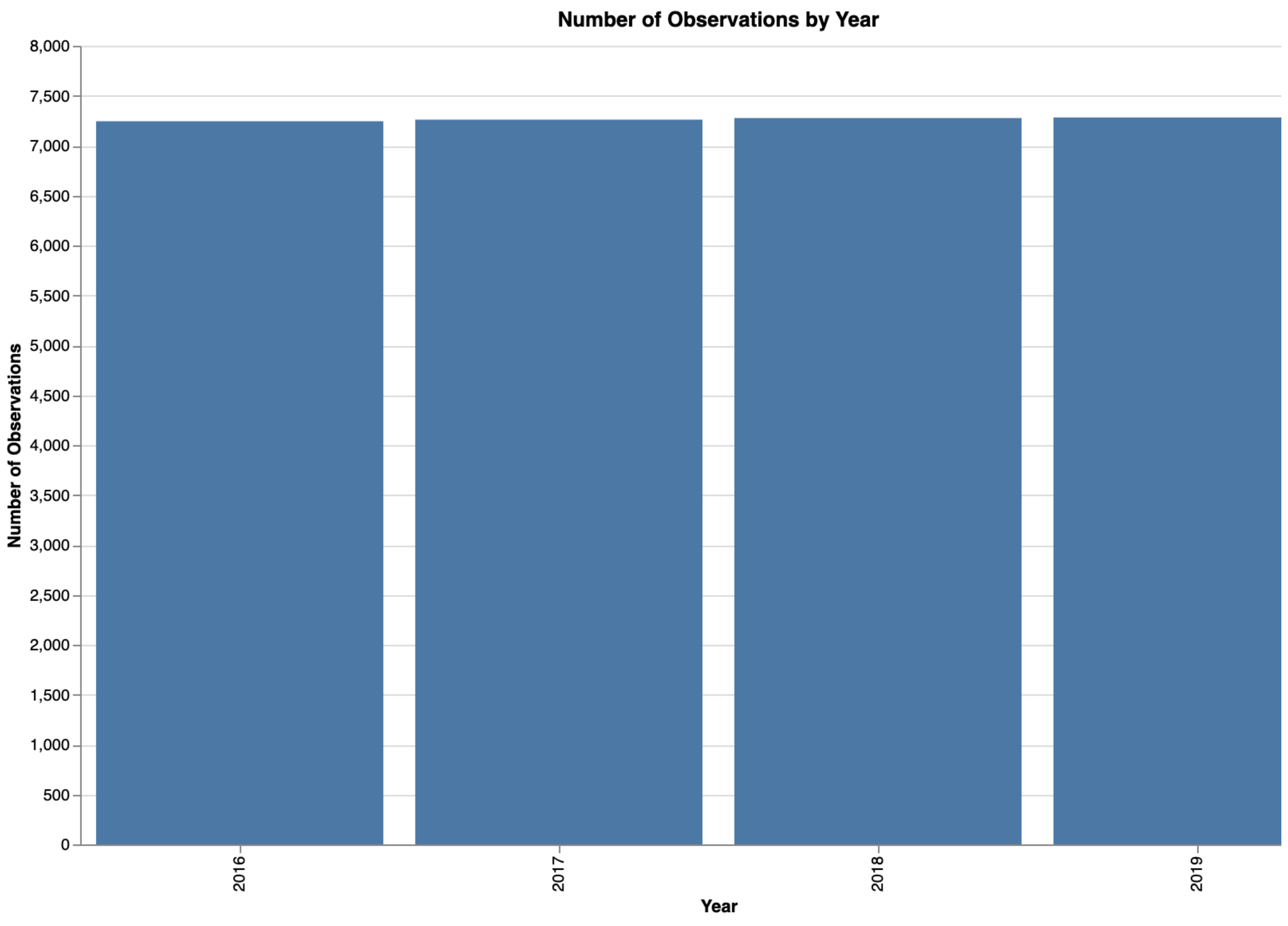
	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	FAC_NAME	INTRMDRY_CARR_CD	MDCD_VNDR_NUM	ORGNL_PRT
0	1.0	1	SOUTHEAST ALABAMA MEDICAL CENTER	10101.0	NaN	19660701
1	1.0	1	NORTH JACKSON HOSPITAL	10.0	NaN	19660701
2	1.0	1	MARSHALL MEDICAL CENTER SOUTH	10.0	NaN	19660701
3	1.0	1	ELIZA COFFEE MEMORIAL HOSPITAL	11.0	NaN	19660701
4	1.0	1	MIZELL MEMORIAL HOSPITAL	10.0	NaN	19660701
...	...	...	...	...	...	...
133526	1.0	1	WEIMAR MEDICAL CENTER	4411.0	NaN	20160226
133527	1.0	1	CLEVELAND EMERGENCY HOSPTIAL	4411.0	NaN	20160616
133528	1.0	1	WISE HEALTH SYSTEM	4411.0	NaN	20160726
133529	1.0	1	TEXAS GENERAL HOSPITAL-VZRMC LP	4411.0	NaN	20161007
133530	1.0	1	FIRST TEXAS HOSPITAL	4411.0	NaN	20161011

7245 rows × 461 columns

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

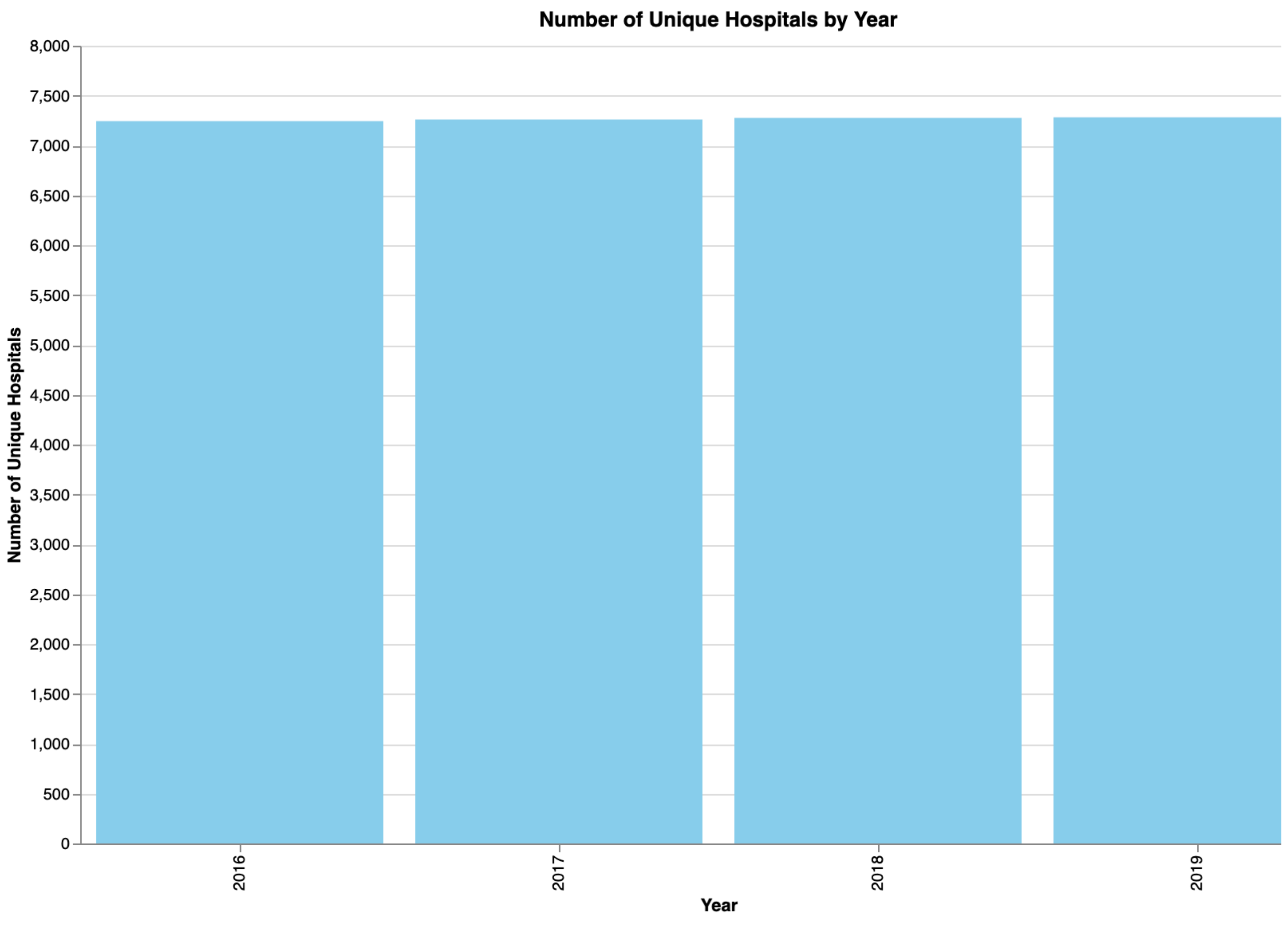
```
pos2017 = pd.read_csv('pos2017.csv')
pos2018 = pd.read_csv('pos2018.csv')
pos2019 = pd.read_csv('pos2019.csv')
```

```
pos2016['year'] = 2016

pos2017['BED_CNT'] = pos2017['BED_CNT'].astype(str).str.replace(',', '').astype(float)
pos2017 = pos2017[(pos2017['PRVDR_CTGRY_SBTYP_CD'] == 1) & (pos2017['PRVDR_CTGRY_CD'] == 1)]
pos2017['year'] = 2017

pos2018['BED_CNT'] = pos2018['BED_CNT'].astype(str).str.replace(',', '').astype(float)
pos2018 = pos2018[(pos2018['PRVDR_CTGRY_SBTYP_CD'] == 1) & (pos2018['PRVDR_CTGRY_CD'] == 1)]
pos2018['year'] = 2018

pos2019['BED_CNT'] = pos2019['BED_CNT'].astype(str).str.replace(',', '').astype(float)
pos2019 = pos2019[(pos2019['PRVDR_CTGRY_SBTYP_CD'] == 1) & (pos2019['PRVDR_CTGRY_CD'] == 1)]
pos2019['year'] = 2019

pos_all = pd.concat([pos2016, pos2017, pos2018, pos2019], ignore_index=True)

yearly_counts = pos_all['year'].value_counts().sort_index()

yearly_counts_df = yearly_counts.reset_index()
yearly_counts_df.columns = ['Year', 'Number of Observations']

chart = alt.Chart(yearly_counts_df).mark_bar().encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Number of Observations:Q', title='Number of Observations')
).properties(
    title='Number of Observations by Year',
    width=800,
    height=500
)

chart.show()
```



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

```python
unique_hospitals_per_year = pos_all.groupby('year')['PRVDR_NUM'].nunique()

unique_hospitals_df = unique_hospitals_per_year.reset_index()
unique_hospitals_df.columns = ['Year', 'Number of Unique Hospitals']

chart = alt.Chart(unique_hospitals_df).mark_bar(color='skyblue').encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Number of Unique Hospitals:Q', title='Number of Unique Hospitals')
).properties(
    title='Number of Unique Hospitals by Year',
    width=800,
    height=500
)

chart.show()
```



- ***Answer***: *If the two bar plots are similar, it means that the dataset already had mostly, if not all, unique rows.*

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

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

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

```python
# Start with the list of hospitals active in 2016
active_hospitals_2016 = pos_all[(pos_all['year'] == 2016) & (pos_all['PGM_TRMNTN_CD'] == 0)]
        [['FAC_NAME', 'ZIP_CD']].drop_duplicates()

suspected_closed_hospitals = []

# Loop through the years 2017 to 2019
for year in range(2017, 2020):

    # Get active hospitals in the current year
    active_hospitals_current_year = pos_all[(pos_all['year'] == year) & (pos_all['PGM_TRMNTN_CD'] ==
        0)][['FAC_NAME', 'ZIP_CD']].drop_duplicates()

    # Identify hospitals that were active in 2016 but are not active in the current year
    closed_hospitals =
        active_hospitals_2016[~active_hospitals_2016['FAC_NAME'].isin(active_hospitals_current_year['FA

    # Record the suspected closed hospitals with the first year they are not found active
    for _, row in closed_hospitals.iterrows():
        # Ensure we don't record the hospital again if already marked as closed in a prior year
        if not any(d['FAC_NAME'] == row['FAC_NAME'] for d in suspected_closed_hospitals):
            suspected_closed_hospitals.append({
                'FAC_NAME': row['FAC_NAME'],
                'ZIP_CD': row['ZIP_CD'],
                'year_of_suspected_closure': year
            })

# Convert the list of suspected closures to a DataFrame
suspected_closed_hospitals_df = pd.DataFrame(suspected_closed_hospitals)

suspected_closed_hospitals_df
```

	FAC_NAME	ZIP_CD	year_of_suspected_closure
0	WEDOWEE HOSPITAL	36278.0	2017
1	BROOKWOOD MEDICAL CENTER	35209.0	2017
2	ABRAZO MARYVALE CAMPUS	85031.0	2017
3	JOHN C. LINCOLN MEDICAL CENTER	85020.0	2017
4	SELLS INDIAN HOSPITAL	85634.0	2017
...	...	...	...
448	PROVIDENCE SACRED HEART MEDICAL CENTER	99204.0	2019
449	SUMMERSVILLE REGIONAL MEDICAL CENTER	26651.0	2019
450	TEXAS GENERAL HOSPITAL	75051.0	2019
451	LITTLE RIVER HEALTHCARE CAMERON HOSPITAL	76520.0	2019
452	PEARLAND MEDICAL CENTER	77584.0	2019

453 rows × 3 columns

```
- ***Answer**: There are 453 unique hospitals that fit this definition.*
```

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

```python
suspected_closed_hospitals_df = suspected_closed_hospitals_df.sort_values(by='FAC_NAME')
suspected_closed_hospitals_df[['FAC_NAME', 'year_of_suspected_closure']].head(10)
```

	FAC_NAME	year_of_suspected_closure
2	ABRAZO MARYVALE CAMPUS	2017
417	ADIRONDACK MEDICAL CENTER	2019
187	ADVENTIST MEDICAL CENTER	2018
9	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2017
8	ADVENTIST MEDICAL CENTER - REEDLEY	2017
305	AFFINITY MEDICAL CENTER	2018
73	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2017
138	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2017
410	ALLIANCE LAIRD HOSPITAL	2019
425	ALLIANCEHEALTH DEACONESS	2019

3. (Partner 2) However, not all suspected hospital closures are true closures. For example, in the case of a merger, a CMS certification number will be appear to be "terminated," but then the hospital re-appear under a similar name/address with a new CMS certification number in the next year. As a first pass to address this, remove any suspected hospital closures that are in zip codes where the number of active hospitals does not decrease in the year after the suspected closure.
   i. Among the suspected closures, how many hospitals fit this definition of potentially being a merger/acquisition? After correcting for this, how many hospitals do you have left?
   ii. Sort this list of corrected hospital closures by name and report the first 10 rows.

```
active_hospitals_count = pos_all[pos_all['PGM_TRMNTN_CD'] == 0].groupby(['year',
        'ZIP_CD']).size().reset_index(name='active_count')
merged_df = suspected_closed_hospitals_df.merge(active_hospitals_count, how='left', left_on=
        ['ZIP_CD', 'year_of_suspected_closure'], right_on=['ZIP_CD', 'year'])
merged_df['next_year'] = merged_df['year_of_suspected_closure'] + 1

next_year_counts = active_hospitals_count.rename(columns={'active_count':
        'next_year_count'}).drop('year', axis=1)
next_year_counts = next_year_counts.drop_duplicates(subset=['ZIP_CD']).groupby('ZIP_CD')
        ['next_year_count'].first().reset_index()
merged_df = merged_df.merge(next_year_counts, how='left', left_on='ZIP_CD', right_on='ZIP_CD')
merg_acq = merged_df[merged_df['next_year_count'] < merged_df['active_count']]

merged_hospitals = suspected_closed_hospitals_df.merge(merg_acq[['FAC_NAME', 'ZIP_CD']], on=
        ['FAC_NAME', 'ZIP_CD'], how='inner')

# Now filter out these matched rows from suspected_closed_hospitals_df
suspected_closed_hospitals_df_filtered =
        suspected_closed_hospitals_df[~suspected_closed_hospitals_df[['FAC_NAME',
        'ZIP_CD']].apply(tuple, 1).isin(merged_hospitals[['FAC_NAME', 'ZIP_CD']].apply(tuple, 1))]

suspected_closed_hospitals_df_filtered
```

	FAC_NAME	ZIP_CD	year_of_suspected_closure
2	ABRAZO MARYVALE CAMPUS	85031.0	2017
417	ADIRONDACK MEDICAL CENTER	12983.0	2019
187	ADVENTIST MEDICAL CENTER	93230.0	2018
9	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230.0	2017
8	ADVENTIST MEDICAL CENTER - REEDLEY	93654.0	2017
...	...	...	...
242	WOMAN'S HOSPITAL	70817.0	2018
405	WOMEN'S AND CHILDREN'S HOSPITAL	70508.0	2019
30	WUESTHOFF MEDICAL CENTER-MELBOURNE	32935.0	2017
27	WUESTHOFF MEDICAL CENTER-ROCKLEDGE	32955.0	2017
21	YAMPA VALLEY MEDICAL CENTER	80487.0	2017

451 rows × 3 columns

- ***Answer****: 2 hospitals fit the definition of potentially being a merger/aquisition. After correcting for these, I have 451 suspected closures.*

Download Census zip code shapefile (10 pt)

Navigate to the Census shapefiles (link), select "gz_2010_us_860_00_500k.zip" and download the resulting shapefile.

Note: If you have difficulty downloading or working with the file (the size is too large for your computer), reach out to the instruction team on Ed so that we can give them a smaller initial shapefile to work with.

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

- ***Answer****: The file types are xml (16kb), dbf (6.4 mb, has attribute information), shp (837mb, has feature geometrics), shx (265kb, has a positional index), and prj (which describes the coordinate reference system).*

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

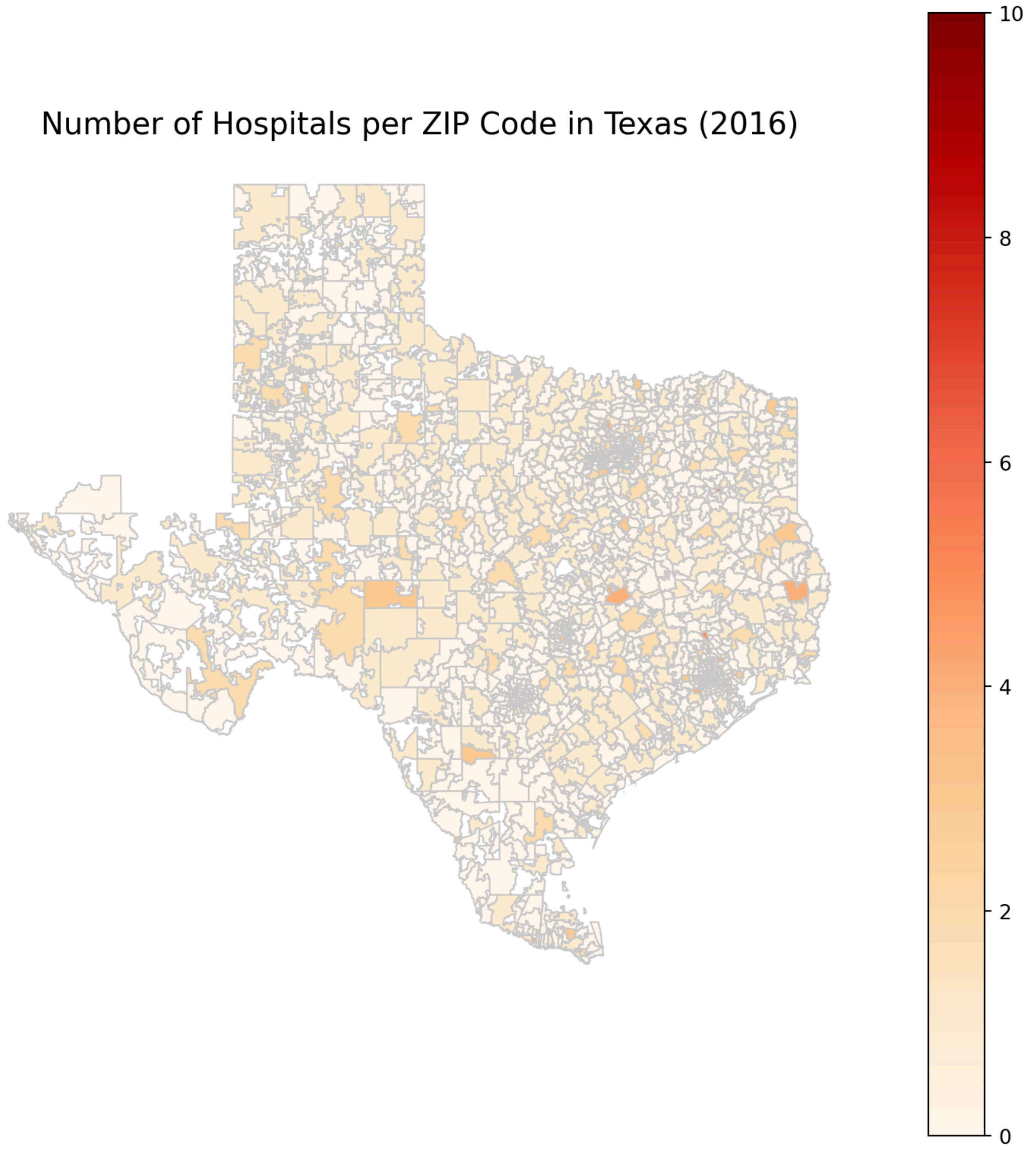
```
import pyproj
pyproj.datadir.set_data_dir('/opt/homebrew/opt/proj/share/proj')

shapefile_path = "gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"
gdf = gpd.read_file(shapefile_path, crs="EPSG:3081")
pos_all
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	FAC_NAME	INTRMDRY_CARR_CD	MDCD_VNDR_NUM	ORGNL_PR
0	1.0	1	SOUTHEAST ALABAMA MEDICAL CENTER	10101.0	NaN	19660701.0
1	1.0	1	NORTH JACKSON HOSPITAL	10.0	NaN	19660701.0
2	1.0	1	MARSHALL MEDICAL CENTER SOUTH	10.0	NaN	19660701.0
3	1.0	1	ELIZA COFFEE MEMORIAL HOSPITAL	11.0	NaN	19660701.0

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	FAC_NAME	INTRMDRY_CARR_CD	MDCD_VNDR_NUM	ORGNL_PR
4	1.0	1	MIZELL MEMORIAL HOSPITAL	10.0	NaN	19660701.0
...	...	...	...	...	...	...
29059	1.0	1	HOUSTON METHODIST THE WOODLANDS HOSPITAL	NaN	NaN	NaN
29060	1.0	1	THE HOSPITALS OF PROVIDENCE HORIZON CITY CAMPUS	NaN	NaN	NaN
29061	1.0	1	TEXAS CENTER FOR INFECTIOUS DISEASE	NaN	NaN	NaN
29062	1.0	1	CROCKETT MEDICAL CENTER	NaN	NaN	NaN
29063	1.0	1	BAYLOR SCOTT & WHITE MEDICAL CENTER PFLUGERVILLE	NaN	NaN	NaN

29064 rows × 465 columns

```python
texas_zip_gdf = gdf[gdf['ZCTA5'].str.startswith(('75', '76', '77', '78', '79'))]
pos_all['ZIP_CD'] = pos_all['ZIP_CD'].astype(str).str.replace('.0', '')
texas_hospitals_2016 = pos_all[(pos_all['year'] == 2016) &
        (pos_all['ZIP_CD'].astype(str).str.startswith(('75', '76', '77', '78', '79')))]
hospital_counts = texas_hospitals_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')
texas_zip_gdf = texas_zip_gdf.rename(columns={'ZCTA5': 'ZIP_CD'})
texas_zip_gdf = texas_zip_gdf.merge(hospital_counts, on='ZIP_CD', how='left')
texas_zip_gdf['hospital_count'] = texas_zip_gdf['hospital_count'].fillna(0)

# Plot a choropleth of the number of hospitals per ZIP code
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_gdf.plot(column='hospital_count', cmap='OrRd', linewidth=0.8, ax=ax, edgecolor='0.8',
        legend=True)

# Add title and labels
ax.set_title('Number of Hospitals per ZIP Code in Texas (2016)', fontsize=15)
ax.set_axis_off()

# Show the plot
plt.show()
```



Calculate zip code's distance to the nearest hospital (25 pts) (*)

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

- ***Answer:*** *Use Geopandas'* `centroid` *command. The resulting dataframe has 33120 rows, one for each zip code, and 2 columns, one for zip code and another for the centroid point of that zip code.*

```
gdf['centroid'] = gdf.centroid
gdf['ZIP_CD'] = pd.to_numeric(gdf['ZCTA5'])

zips_all_centroids = gpd.GeoDataFrame(gdf[['ZIP_CD', 'centroid']])
zips_all_centroids = zips_all_centroids.rename(columns={'ZCTA5': 'ZIP_CD'})
zips_all_centroids.set_geometry('centroid', inplace=True)
zips_all_centroids.shape
```

```
(33120, 2)
```

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

```
zips_all_centroids['ZIP_CD'] = zips_all_centroids['ZIP_CD'].astype(str).str.zfill(5)
zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZIP_CD'].str[:2].isin(['75', '76',
        '77', '78', '79'])]

zips_texas_borderstates_centroids =
        zips_all_centroids[zips_all_centroids['ZIP_CD'].str[:2].isin(['75', '76', '77', '78', '79',
        '73', '74', '72', '70', '87', '88', '90', '91', '92', '93'])]

zips_texas_borderstates_centroids
zips_texas_centroids
```

	ZIP_CD	centroid
9207	78624	POINT (-98.87707 30.2816)
9208	78626	POINT (-97.59733 30.66535)
9209	78628	POINT (-97.75112 30.64108)
9210	78631	POINT (-99.30528 30.33772)
9211	78632	POINT (-97.47045 29.69633)
…	…	…
32917	78261	POINT (-98.40189 29.6918)
32918	78368	POINT (-97.8103 28.10544)

	ZIP_CD	centroid
32919	78412	POINT (-97.34297 27.70383)
32920	78557	POINT (-98.24337 26.10675)
32921	78586	POINT (-97.6311 26.10507)

1935 rows × 2 columns

- ***Answer***: *There are 1935 zip codes in Texas and 4654 in Texas and bordering states.*

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

```
zip_codes_with_hospitals = active_hospitals_count['ZIP_CD'].unique()
zip_codes_with_hospitals_df = pd.DataFrame(zip_codes_with_hospitals, columns=['ZIP_CD'])
zip_codes_with_hospitals_df['ZIP_CD'] =
        zip_codes_with_hospitals_df['ZIP_CD'].astype(str).str.replace('.0', '').str.zfill(5)

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    zip_codes_with_hospitals_df,
    how='inner',  # Inner merge to keep only ZIP codes that exist in both DataFrames
    on='ZIP_CD'   # Merge on the 'ZIP_CD' column
)

zips_withhospital_centroids
```

	ZIP_CD	centroid
0	70043	POINT (-89.96276 29.94804)
1	70127	POINT (-89.97675 30.02501)
2	70301	POINT (-90.74089 29.8141)
3	70360	POINT (-90.81028 29.58819)
4	70403	POINT (-90.48388 30.48002)
...	...	...
584	77375	POINT (-95.5894 30.09551)
585	74361	POINT (-95.30665 36.30295)
586	75035	POINT (-96.77269 33.1553)
587	78028	POINT (-99.15819 30.03424)
588	78412	POINT (-97.34297 27.70383)

589 rows × 2 columns

- ***Answer:*** *It should be a non-spatial, inner join merge because we want to find the intersection between the two dataframes. We should merge on zip code.*

4. (Partner 2) For each zip code in `zips_texas_centroids`, calculate the distance to the nearest zip code with at least one hospital in `zips_withhospital_centroids`.
   a. How long did it take?

```
import time

start_time = time.time()

nearest_hospital_df = gpd.sjoin_nearest(
zips_texas_centroids,
zips_withhospital_centroids,
distance_col='nearest_hospital_distance',  # Column to store distance
how='left'  # Keep all Texas ZIP codes
)
elapsed_time = time.time() - start_time
```

- ***Answer:*** *Use Geopandas `sjoin_nearest` command with the `distance_col` parameter and calculate the distance between `zips_withhospital_centroids` and `zips_texas_centroids` (0.022 seconds).*

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

- ***Answer:*** *The average conversion from degrees to miles would be about 61.8 (may vary)*

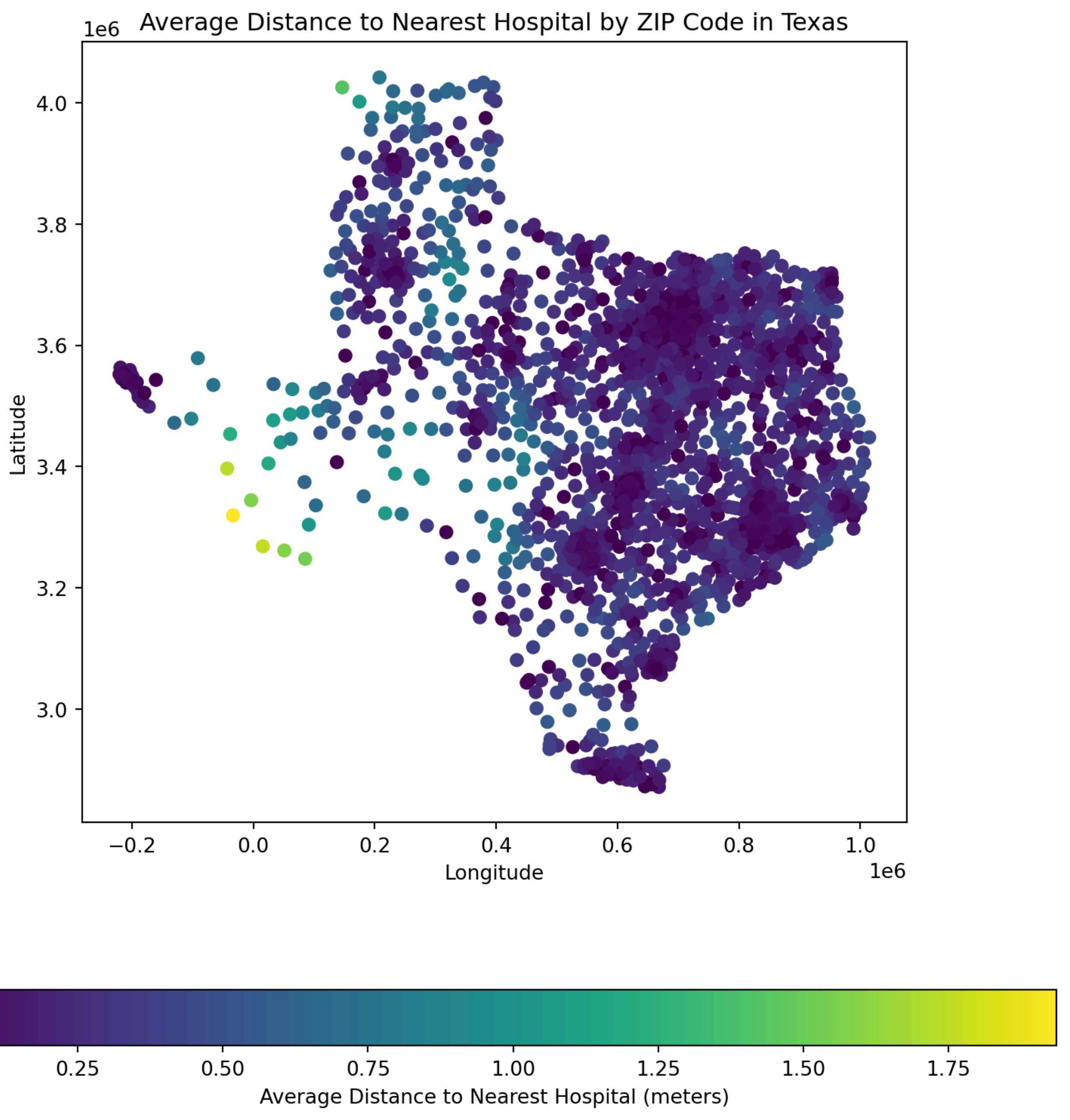
5. (Partner 2) Calculate the average distance to the nearest hospital for each zip code in Texas. What unit is this in? Report the average distance in miles. Does this value make sense? Map the value for each zip code.

```
if nearest_hospital_df.crs is None:
    print("CRS is missing; please set it to the appropriate EPSG.")
else:
    print(nearest_hospital_df.crs)
    # If needed, reproject:
    nearest_hospital_df = nearest_hospital_df.to_crs(epsg=32614)  # UTM Zone 14N, suitable for Texas
```

```
EPSG:4269
```

```
nearest_hospital_df['nearest_hospital_distance'] =
        pd.to_numeric(nearest_hospital_df['nearest_hospital_distance'], errors='coerce')
average_distances = nearest_hospital_df.groupby('ZIP_CD_left')
        ['nearest_hospital_distance'].mean().reset_index()
```

```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
nearest_hospital_df.boundary.plot(ax=ax, linewidth=1, color='black')  # Plot the boundaries
nearest_hospital_df.plot(column='nearest_hospital_distance', ax=ax, legend=True,
                         cmap='viridis', legend_kwds={'label': "Average Distance to Nearest Hospital
        (meters)",
                                                      'orientation': "horizontal"})
plt.title('Average Distance to Nearest Hospital by ZIP Code in Texas')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```


Average Distance to Nearest Hospital by ZIP Code in Texas
1e6
4.0
3.8
3.6
3.4
3.2
3.0
Latitude
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Longitude
1e6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Distance to Nearest Hospital (meters)

```
average_distances = nearest_hospital_df.groupby('ZIP_CD_left')
        ['nearest_hospital_distance'].mean().reset_index()
zips_texas_centroids = zips_texas_centroids.merge(average_distances, left_on='ZIP_CD',
        right_on='ZIP_CD_left', how='left')
zips_texas_centroids
```

	ZIP_CD	centroid	ZIP_CD_left	nearest_hospital_distance
0	78624	POINT (-98.87707 30.2816)	78624	0.000000
1	78626	POINT (-97.59733 30.66535)	78626	0.167651
2	78628	POINT (-97.75112 30.64108)	78628	0.110844
3	78631	POINT (-99.30528 30.33772)	78631	0.337251
4	78632	POINT (-97.47045 29.69633)	78632	0.219909
...	...	...	...	...
1930	78261	POINT (-98.40189 29.6918)	78261	0.110636
1931	78368	POINT (-97.8103 28.10544)	78368	0.313242
1932	78412	POINT (-97.34297 27.70383)	78412	0.000000
1933	78557	POINT (-98.24337 26.10675)	78557	0.058567
1934	78586	POINT (-97.6311 26.10507)	78586	0.155733

1935 rows × 4 columns

- ***Answer:*** *The average distance to the nearest hospital is 0.209 (in degrees) or about 13 (in miles)*

Effects of closures on access in Texas (20 pts)

We next want to identify the zip codes in Texas that were affected by a closure in 2016-2019.

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

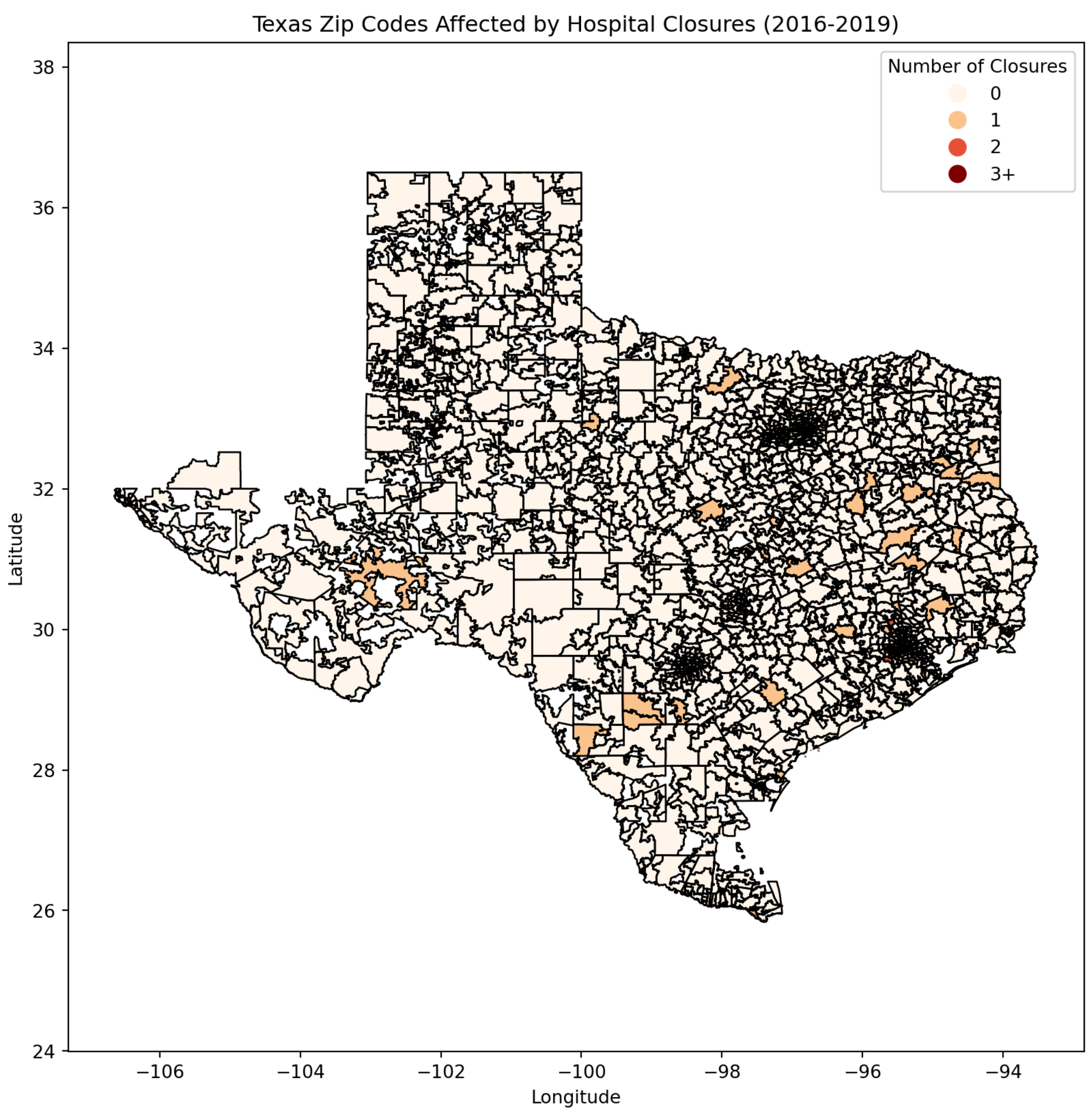
```
texas_zip_prefixes = ['75', '76', '77', '78', '79']
suspected_closed_hospitals_df_filtered['ZIP_CD'] =
        suspected_closed_hospitals_df_filtered['ZIP_CD'].astype(str).str.replace('.0',
        '').str.zfill(5)
texas_closures =
        suspected_closed_hospitals_df_filtered[suspected_closed_hospitals_df_filtered['ZIP_CD'].str.sta
texas_closures_zips = texas_closures['ZIP_CD'].tolist()
# Count the number of closures per zip code
closure_counts = texas_closures.groupby('ZIP_CD').size()
closure_counts
```

```
ZIP_CD
75035    1
75039    1
75042    1
75051    1
75218    1
        ..
78834    1
79553    1
79735    1
79761    1
79902    2
Length: 66, dtype: int64
```

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

```
import numpy as np

# Define bins and labels for discrete color mapping
bins = [0, 1, 2, 3, np.inf]  # The last bin includes any values >= 10
labels = ['0', '1', '2', '3+']  # Adjust labels to match bins

texas_zip_gdf['closure_count'] = texas_zip_gdf['ZIP_CD'].map(closure_counts).fillna(0)
# Create a new column with binned closure counts
texas_zip_gdf['closure_bins'] = pd.cut(texas_zip_gdf['closure_count'], bins=bins, labels=labels,
        include_lowest=True, right=False)

# Check the distribution in the binned data
closure_bins_count = texas_zip_gdf['closure_bins'].value_counts().sort_index()
print(closure_bins_count)

# Plotting
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_gdf.boundary.plot(ax=ax, linewidth=1, color='black')  # Plot boundaries
plot = texas_zip_gdf.plot(column='closure_bins', ax=ax, legend=True,
                          cmap='OrRd', missing_kwds={'color': 'lightgrey'})

# Set the title for the color bar
plot.get_legend().set_title("Number of Closures")

# Title and labels
plt.title('Texas Zip Codes Affected by Hospital Closures (2016–2019)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.axis('equal')
plt.show()
```

```
closure_bins
0     1871
1       59
2        4
3+       1
Name: count, dtype: int64
```



- ***Answer:*** *There are 64 affected zip codes.*

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

```
texas_zip_gdf_direct = texas_zip_gdf[texas_zip_gdf['ZIP_CD'].isin(texas_closures_zips)]

buffered_gdf = texas_zip_gdf_direct.copy()
buffered_gdf['geometry'] = buffered_gdf.geometry.buffer(0.161)  # 10 miles in degrees

indirectly_affected = gpd.sjoin(texas_zip_gdf, buffered_gdf, predicate='intersects')
len(set(indirectly_affected['ZIP_CD_left']))
```

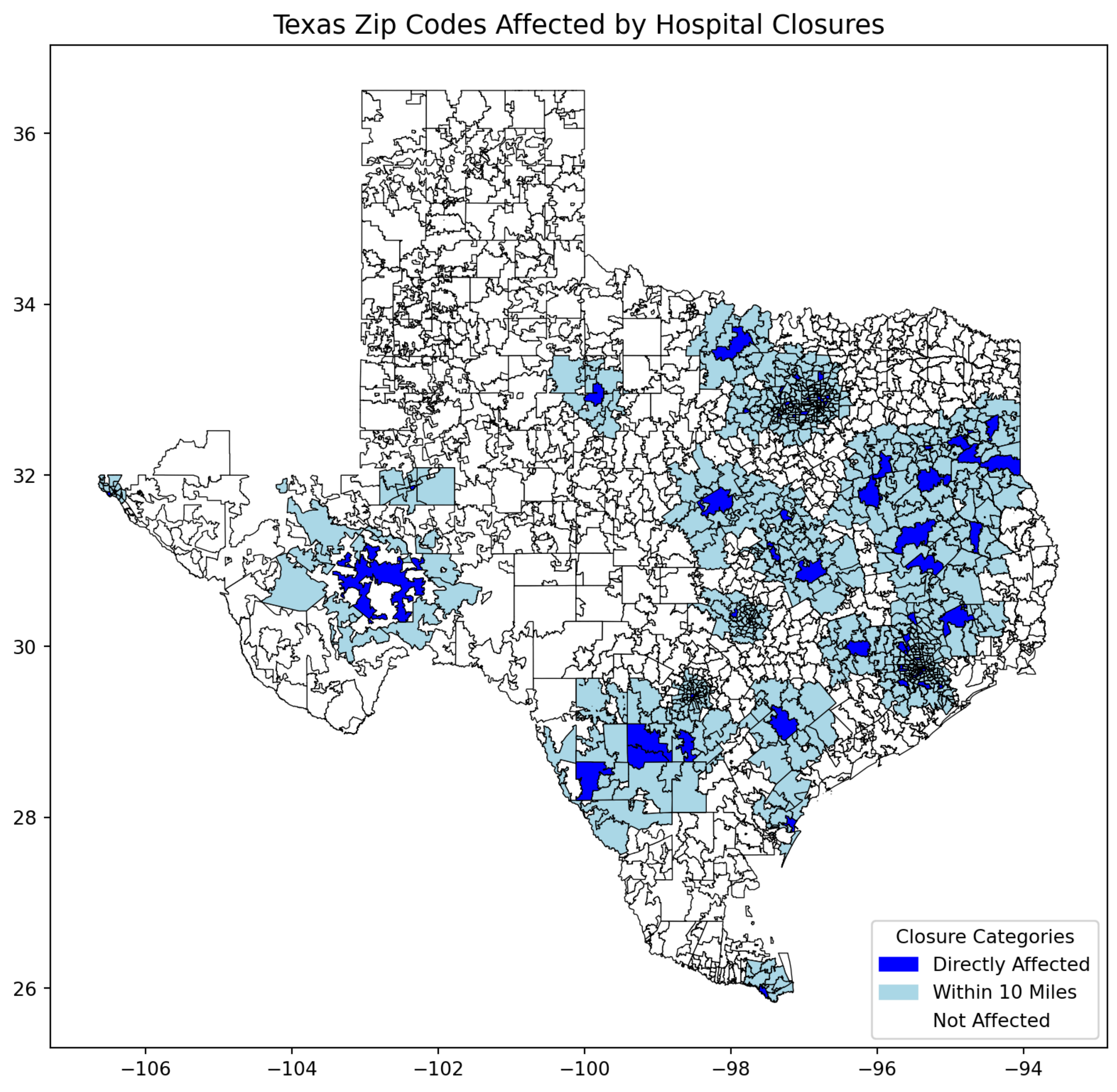
```
933
```

- ***Answer:*** *There are 933 indirectly affected zip codes*

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

```
import matplotlib.patches as mpatches

not_affected = texas_zip_gdf[~texas_zip_gdf['ZIP_CD'].isin(texas_closures_zips)]
not_affected.shape

texas_zip_gdf_direct['category'] = 'Directly Affected'
indirectly_affected['category'] = 'Within 10 Miles'
not_affected['category'] = 'Not Affected'

combined_gdf = gpd.GeoDataFrame(pd.concat([texas_zip_gdf_direct, indirectly_affected, not_affected],
        ignore_index=True))

# Reorder combined_gdf so 'Directly Affected' is drawn last
combined_gdf['category'] = pd.Categorical(combined_gdf['category'], categories=['Not Affected',
        'Within 10 Miles', 'Directly Affected'], ordered=True)
combined_gdf = combined_gdf.sort_values('category')

# Define colors for each category
colors = {'Directly Affected': 'blue', 'Within 10 Miles': 'lightblue', 'Not Affected': 'white'}

# Map colors to the combined GeoDataFrame
combined_gdf['color'] = combined_gdf['category'].map(colors)

# Plotting with enhanced edge line width for visibility
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
```

```python
combined_gdf.plot(ax=ax, color=combined_gdf['color'], edgecolor='black', linewidth=0.5)

# Create legend patches
legend_patches = [mpatches.Patch(color=color, label=category) for category, color in colors.items()]
plt.legend(handles=legend_patches, title='Closure Categories', loc='lower right')

# Title
plt.title('Texas Zip Codes Affected by Hospital Closures', fontsize=14)
plt.show()
```



Reflecting on the exercise (10 pts)

- (Partner 1) The "first-pass" method we're using to address incorrectly identified closures in the data is imperfect. Can you think of some potential issues that could arise still and ways to do a better job at confirming hospital closures?
    - ***Example answer****: This could accidentally eliminate true hospital closures in zip codes where another hospital entered in the next year – the net change in number of hospitals in that zip code would be 0. A better way to do this would be to pull in hospital address or hospital name and check if the hospital's name or address remained the same. Since there could be slight deviations in both of these from year to year, we could do fuzzy matching using Levenshtein distances or Jaro-Winkler distances.*
- (Partner 2) Consider the way we are identifying zip codes affected by closures. How well does this reflect changes in zip-code-level access to hospitals? Can you think of some ways to improve this measure?
    - ***Example answer****: this measure does not take into account the total number of hospitals that the population in a zip code has access to, but just looks at whether they were affected by a hospital closure. A hospital closure in a large metropolitan area like Dallas probably won't have much of an impact on hospital access in the surrounding areas because there are many other hospitals in that area. A better measure would take into account the total number of hospitals or hospital-beds a zip code has access to.*