```
--- Jiaxuan Nie, Suyuan Fang
title: "PS4"
format:
  html: default
  pdf:
    keep-tex: true
    include-in-header:
       text: |
         \usepackage{fvextra}

\DefineVerbatimEnvironment{Highlighting}{Verbatim}{breaklines,commandchars=\
\\{\}}
include-before-body:
  text: |
    \RecustomVerbatimEnvironment{verbatim}{Verbatim}{
      showspaces = false,
      showtabs = false,
      breaksymbolleft={},
      breaklines
    }
---

**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
**[yes](https://uchicago.zoom.us/rec/share/pG_wQ-pHTQrJTmqNn4rcrw5V194M2H2s-2jdy8oVhWHkd_yZt9o162IWurpA-fxU.BIQlSgZLRYctvzp-)**.

## 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): Suyuan Fang - suyuanfang
    - Partner 2 (name and cnet ID): Jiaxuan nie - Jnie21
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: \*\*\SF\*\* \*\*\JN\*\*
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set 4
```

[yes](https://docs.google.com/forms/d/185usrCREQaUbvAXpWhChkjghdGgmAZXA3lPWpXLLsts/edit)" (1 point)
6. Late coins used this pset:1 Late coins left after submission:2
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](https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github#removing-files-from-a-repositorys-history). 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.

#Facility Name FAC_NAME
#short-term PRVDR_CTGRY_SBTYP_CD
#hospital PRVDR_CTGRY_CD
#CMS PRVDR_NUM
#termination PGM_TRMNTN_CD
#termination CRTFCTN_ACTN_TYPE_CD
#zip code ZIP_CD

2.

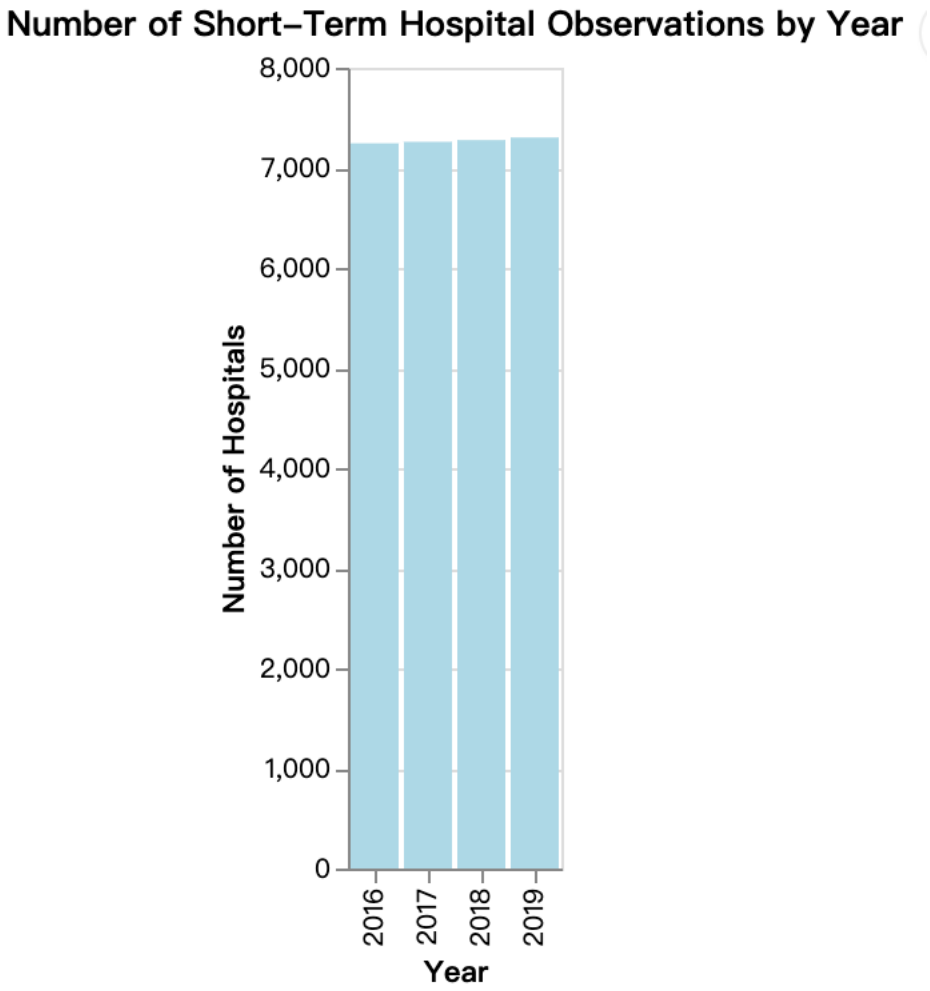
a.Number of short-term hospitals in 2016: 7245
b.The count of 7,245 short-term hospitals in 2016 may be higher than expected. Official sources such as the CMS or AHA typically report fewer facilities. Differences might result from data definitions or inclusion criteria in your dataset.

3.

```{python}
import pandas as pd
import altair as alt

def process_hospital_data(file_path):
    df = pd.read_csv(file_path, encoding='ISO-8859-1')
```

```
    df_filtered = df[(df['PRVDR_CTGRY_CD'] == 1) &
(df['PRVDR_CTGRY_SBTYP_CD'] == 1)].copy()
    year = file_path[-8:-4]
    df_filtered.loc[:, 'Year'] = year
    hospital_count = df_filtered.shape[0]
    print(f'Number of short-term hospitals in {year}: {hospital_count}')
    return df_filtered

pos2016 = process_hospital_data('/Users/suyuanfang/Desktop/Pyhton/problem-
set-4-nelly-alex/pos2016.csv')
pos2017 = process_hospital_data('/Users/suyuanfang/Desktop/Pyhton/problem-
set-4-nelly-alex/pos2017.csv')
pos2018 = process_hospital_data('/Users/suyuanfang/Desktop/Pyhton/problem-
set-4-nelly-alex/pos2018.csv')
pos2019 = process_hospital_data('/Users/suyuanfang/Desktop/Pyhton/problem-
set-4-nelly-alex/pos2019.csv')

data_combined = pd.concat([pos2016, pos2017, pos2018, pos2019])

output_path = '/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-
alex/combined_pos.csv'
data_combined.to_csv(output_path, index=False)

plot_data = data_combined['Year'].value_counts().reset_index()
plot_data.columns = ['Year', 'Count']
plot_data = plot_data.sort_values('Year')

chart = alt.Chart(plot_data).mark_bar(color='lightblue').encode(
    x=alt.X('Year:N', title='Year'),
    y=alt.Y('Count:Q', title='Number of Hospitals')
).properties(
    title='Number of Short-Term Hospital Observations by Year'
)
chart.display()
```

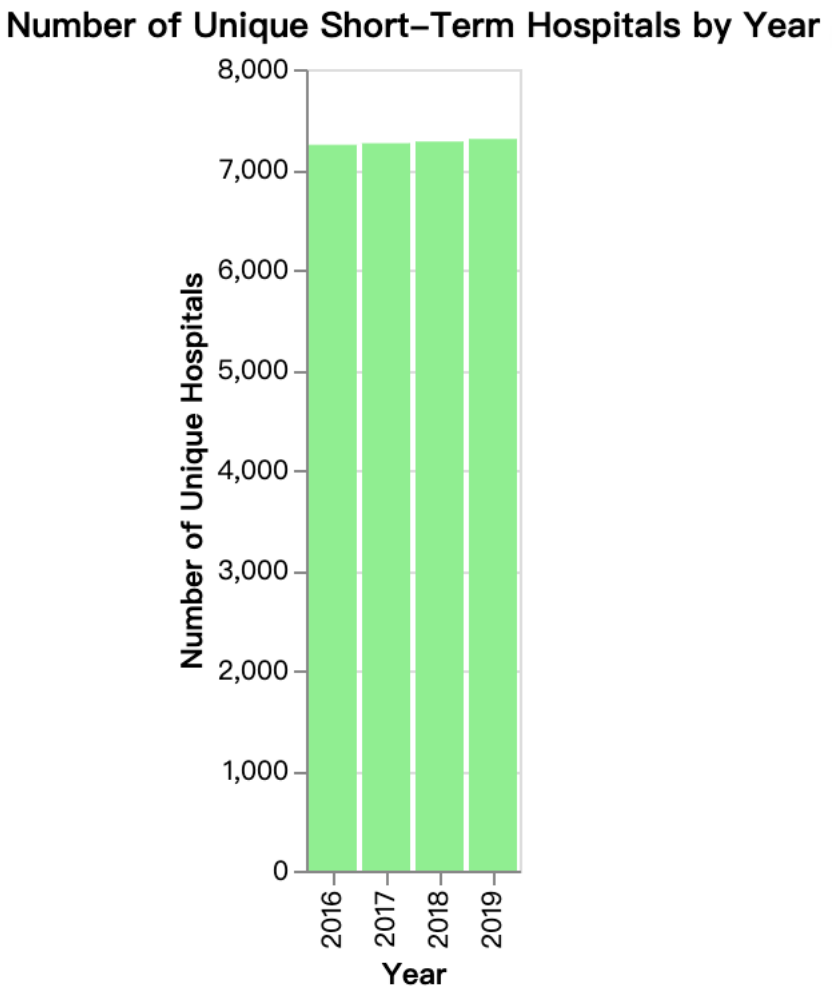
```
Number of short-term hospitals in 2016: 7245
Number of short-term hospitals in 2017: 7260
Number of short-term hospitals in 2018: 7277
Number of short-term hospitals in 2019: 7303
```



4.

a.

```{python}
unique_hospitals_per_year = data_combined.groupby('Year')['PRVDR_NUM'].nunique().reset_index()
unique_hospitals_per_year.columns = ['Year', 'Unique Count']

chart_unique = alt.Chart(unique_hospitals_per_year).mark_bar(color='lightgreen').encode(
    x=alt.X('Year:N', title='Year'),
    y=alt.Y('Unique Count:Q', title='Number of Unique Hospitals')
).properties(
    title='Number of Unique Short-Term Hospitals by Year'
)
chart_unique.display()
print("\nUnique hospital counts per year:\n", unique_hospitals_per_year)
```



Unique hospital counts per year:

	Year	Unique Count
0	2016	7245
1	2017	7260
2	2018	7277
3	2019	7303

b.The two plots show that the total number of observations and unique hospitals per year are nearly identical, indicating that each hospital is consistently reported only once per year. This suggests the data structure is stable, with no significant duplication or variation across the years analyzed.

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

1. 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?

2. Number of hospitals in 2016: 7245
3. Number of hospitals in 2017: 7260
4. Number of hospitals in 2018: 7277
5. Number of hospitals in 2019: 7303
6. Number of suspected closures: 174

```{python}
!pip install pandas
import pandas as pd

def hospital_data(file_path):
    return pd.read_csv(file_path, encoding='latin1')
pos2016 = hospital_data('/Users/jxn/Documents/GitHub/Alex-Nelly/pos2016.csv')
pos2017 = hospital_data('/Users/jxn/Documents/GitHub/Alex-Nelly/pos2017.csv')
pos2018 = hospital_data('/Users/jxn/Documents/GitHub/Alex-Nelly/pos2018.csv')
pos2019 = hospital_data('/Users/jxn/Documents/GitHub/Alex-Nelly/pos2019.csv')
print("Sample of active hospitals in 2016 (hospital_active):")
print(hospital_active.head())

print("Columns in closures_df:", closures_df.columns)
print("Columns in potential_mergers_df:", potential_mergers_df.columns)
hospital_active = pos2016[pos2016['PGM_TRMNTN_CD'] == '0']
data_years = {2017: pos2017, 2018: pos2018, 2019: pos2019}
print("List of hospitals active in 2016 and suspected to have closed by 2019:")
print(final_closures_df[['FAC_NAME', 'ZIP_CD', 'Year_of_Closure']])
num_suspected_closures = final_closures_df.shape[0]
print(f"Number of hospitals suspected to have closed by 2019: {num_suspected_closures}")
num_suspected_closures = hospital_active.shape[0]
print(f"Number of hospitals suspected to have closed by 2019: {num_suspected_closures}")


2. First 10 rows of the cleaned 2016 dataset:
   PRVDR_CTGRY_SBTYP_CD  PRVDR_CTGRY_CD                          FAC_NAME  \
0                   1.0               1  SOUTHEAST ALABAMA MEDICAL CENTER
1                   1.0               1            NORTH JACKSON HOSPITAL
2                   1.0               1     MARSHALL MEDICAL CENTER SOUTH
3                   1.0               1    ELIZA COFFEE MEMORIAL HOSPITAL
```

```
4                   1.0               1          MIZELL MEMORIAL HOSPITAL
5                   1.0               1       CRENSHAW COMMUNITY HOSPITAL
6                   1.0               1          HARTSELLE MEDICAL CENTER
7                   1.0               1     MARSHALL MEDICAL CENTER NORTH
8                   1.0               1                 ST VINCENT'S EAST
9                   1.0               1    DEKALB REGIONAL MEDICAL CENTER
3.

hospital_closures = []
for _, row in hospital_active.iterrows():
    prvid = row['PRVDR_NUM']
    facility_name = row['FAC_NAME']
    zip_code = row['ZIP_CD']

    closed = False
    for year, data in data_years.items():
        if prvid in data['PRVDR_NUM'].values:
            status = data[data['PRVDR_NUM'] ==
prvid]['PGM_TRMNTN_CD'].values[0]
            if status != '0':
                hospital_closures.append((facility_name, zip_code, year))
                closed = True
                break
        else:
            hospital_closures.append((facility_name, zip_code, year))
            closed = True
            break
closures_df = pd.DataFrame(hospital_closures, columns=['FAC_NAME', 'ZIP_CD',
'Year_of_Closure']).drop_duplicates()
3.

    a. .Number of remaining closures after correcting for
mergers/acquisitions: 77
    b.Cleaned 2016 dataset saved as 'cleaned_2016_pos.csv' without suspected
closures
    c.
                                              FAC_NAME   ZIP_CD  \
62                           ALLIANCE LAIRD HOSPITAL  39365.0
101                         ALLIANCEHEALTH DEACONESS  73112.0
26                     ANNE BATES LEACH EYE HOSPITAL  33136.0
115                    BARIX CLINICS OF PENNSYLVANIA  19047.0
171                  BAYLOR EMERGENCY MEDICAL CENTER  75087.0
166  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0
98                         BELMONT COMMUNITY HOSPITAL  43906.0
```

```
67                              BIG SKY MEDICAL CENTER  59716.0
65                BLACK RIVER COMMUNITY MEDICAL CENTER  63901.0
142                         CARE REGIONAL MEDICAL CENTER  78336.0

if not potential_mergers_df.empty:
    final_closures_df = pd.merge(
        closures_df,
        potential_mergers_df[['FAC_NAME', 'ZIP_CD', 'Year_of_Closure']],
        on=['FAC_NAME', 'ZIP_CD', 'Year_of_Closure'],
        how='left',
        indicator=True
    )
    final_closures_df = final_closures_df[final_closures_df['_merge'] ==
'left_only'].drop(columns=['_merge'])
else:
    # If potential_mergers_df is empty, use closures_df directly as
final_closures_df
    final_closures_df = closures_df.copy()
# Output the number of remaining closures after correcting for
mergers/acquisitions
num_final_closures = final_closures_df.shape[0]
print(f"Number of remaining closures after correcting for
mergers/acquisitions: {num_final_closures}")
cleaned_2016_pos = pos2016.merge(final_closures_df[['FAC_NAME', 'ZIP_CD']],
on=['FAC_NAME', 'ZIP_CD'], how='left', indicator=True)
cleaned_2016_pos = cleaned_2016_pos[cleaned_2016_pos['_merge'] ==
'left_only'].drop(columns=['_merge'])
cleaned_2016_pos.to_csv("cleaned_2016_pos.csv", index=False)
print("Cleaned 2016 dataset saved as 'cleaned_2016_pos.csv' without
suspected closures.")
print("First 10 rows of the cleaned 2016 dataset:")
print(cleaned_2016_pos.head(10))

```
Download Census zip code shapefile (10 pt)
1.
   a.
   .shp (Shapefile): Holds geometric data (e.g., ZIP code boundaries).
   .shx (Index File): Provides quick access to the .shp data.
   .dbf (Database File): Contains attribute data for each shape (e.g., ZIP
codes).
   .prj (Projection File): Specifies the coordinate system and projection.
   .xml (Metadata File): Describes the dataset, including sources and
attributes.
```
```

   b.
   .shp 837.5 MB
   .shx 265KB
   .dbf 6.4MB
   .prj 165B
   .xml 16KB
2.
```{python}
import pandas as pd
import geopandas as gpd
import altair as alt

cleaned_pos2016 = pd.read_csv('/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-alex/cleaned_2016_pos.csv')

zip_shapefile_path = '/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-alex/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp'
gdf = gpd.read_file(zip_shapefile_path)

print(gdf.columns)
gdf['ZIP_CODE'] = gdf['ZCTA5'].astype(str)
texas_prefixes = ['75', '76', '77', '78', '79']
gdf_texas = gdf[gdf['ZIP_CODE'].str[:2].isin(texas_prefixes)]
hospitals_per_zip = cleaned_pos2016['ZIP_CD'].value_counts().reset_index()
hospitals_per_zip.columns = ['ZIP_CODE', 'Hospital_Count']
hospitals_per_zip['ZIP_CODE'] = hospitals_per_zip['ZIP_CODE'].astype(str)
gdf_texas['ZIP_CODE'] = gdf_texas['ZIP_CODE'].astype(str)
gdf_texas = gdf_texas.merge(hospitals_per_zip, on='ZIP_CODE', how='left')

plot_data = gdf_texas[['ZIP_CODE', 'geometry', 'Hospital_Count']]
plot_data['Hospital_Count'] = plot_data['Hospital_Count'].fillna(0)

chart = alt.Chart(plot_data).mark_geoshape().encode(
   color=alt.Color('Hospital_Count:Q', scale=alt.Scale(scheme='blues'), title='Hospital Count'),
   tooltip=['ZIP_CODE', 'Hospital_Count']
).properties(
   width=800,
   height=600,
   title='Number of Hospitals by ZIP Code in Texas (2016)'
)
chart.display()
```

...

Calculate zip code's distance to the nearest hospital (20 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 and what do each of the columns mean?

```{python}
import geopandas as gpd
!pip install matplotlib
import pandas as pd
import matplotlib.pyplot as plt
```

Dimensions of zips_all_centroids: (33120, 7):there are 33,120 rows and 7 columns. Each row represents a ZIP code area across the U.S., and each column contains geographic or identifying information.

Columns in zips_all_centroids:

GEO_ID: A unique identifier for each ZIP code area in the dataset.

ZCTA5: approximates the U.S. Postal Service ZIP code.
NAME: Another identifier
LSAD: which can indicate the type or classification of the area.
CENSUSAREA: the geographic area (likely in square miles or square kilometers) of each ZIP code region.
geometry: The original polygon geometry that outlines each ZIP code boundary.
centroid: A point geometry representing the centroid (center) of each ZIP code area, calculated from the polygon in the geometry column.

```
zips_all = gpd.read_file('/Users/jxn/Desktop/gz_2010_us_860_00_500k')
zips_all_centroids = zips_all.copy()
zips_all_centroids['centroid'] = zips_all.geometry.centroid
zips_all_centroids = zips_all_centroids.set_geometry('centroid')
```

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?
Unique ZIP codes in Texas: 1935
Unique ZIP codes in Texas and bordering states: 3486
Number of ZIP codes with at least one hospital in 2016: 0

```
texas_and_border_prefixes = ['75', '76', '77', '78', '79', '73', '88', '89', '84', '85', '86', '87']
pos2016_with_hospitals = pos2016[pos2016['PGM_TRMNTN_CD'] == '0']
pos2016_with_hospitals['ZIP_CD'] = pos2016_with_hospitals['ZIP_CD'].astype(str).str.zfill(5)
pos2016_with_hospitals = pos2016_with_hospitals[pos2016_with_hospitals['ZIP_CD'].str[:2].isin(texas_and_border_prefixes)]
pos2016_with_hospitals = pos2016_with_hospitals[['ZIP_CD']].drop_duplicates()
pos2016_with_hospitals.columns = ['ZCTA5']
print("Unique ZIP codes in Texas:", zips_texas_centroids['ZCTA5'].nunique())
print("Unique ZIP codes in Texas and bordering states:", zips_texas_borderstates_centroids['ZCTA5'].nunique())
```

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

I decided to merge on zipcodes since it is one of the major variable we display in 2016 hospital data.In this case, we merge on ZATC5.

```
pos2016_with_hospitals = pos2016[pos2016['PGM_TRMNTN_CD'] ==
'0'][['ZIP_CD']].drop_duplicates()
pos2016_with_hospitals.columns = ['ZCTA5']
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    pos2016_with_hospitals, on='ZCTA5', how='inner'
)
print("Number of ZIP codes with at least one hospital in 2016:",
zips_withhospital_centroids['ZCTA5'].nunique())
```

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.

```
  zips_all_centroids['centroid'] = zips_all.geometry.centroid
Unique ZIP codes in Texas: 1935
Unique ZIP codes in Texas and bordering states: 3486
Number of ZIP codes with at least one hospital in 2016: 0
Average distance to nearest hospital (in miles): nan
```

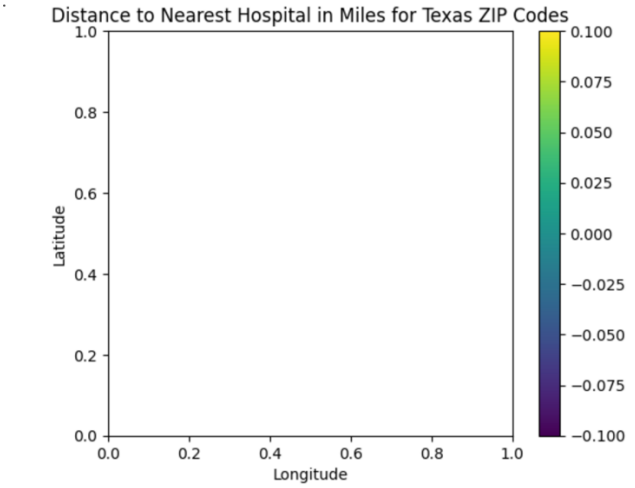
```
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=5070)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=5070)
zips_texas_centroids['nearest_hospital_distance'] =
zips_texas_centroids.geometry.apply(
    lambda x: zips_withhospital_centroids.distance(x).min()
)
zips_texas_centroids['nearest_hospital_distance_miles'] =
zips_texas_centroids['nearest_hospital_distance'] * 0.000621371
print("Average distance to nearest hospital (in miles):",
zips_texas_centroids['nearest_hospital_distance_miles'].mean())
```

```
test_zips = zips_texas_centroids.sample(10)
start_time = time.time()
test_zips['nearest_hospital_distance'] = test_zips.geometry.apply(
```

```
    lambda x: zips_withhospital_centroids.distance(x).min()
)
end_time = time.time()
print("Time taken for 10 ZIP codes:", end_time - start_time, "seconds")
estimated_time = (end_time - start_time) * (len(zips_texas_centroids) / 10)
print("Estimated time for full dataset:", estimated_time, "seconds")
```

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

```
zips_all_centroids['centroid'] = zips_all.geometry.centroid
Unique ZIP codes in Texas: 1935
Unique ZIP codes in Texas and bordering states: 3486
Number of ZIP codes with at least one hospital in 2016: 0
Time taken for 10 ZIP codes: 0.00074291229248046888 seconds
Estimated time for full dataset: 0.1437535285949707 seconds
Actual time for full calculation: 0.07631802558898926 seconds
```




```
start_time_full = time.time()
zips_texas_centroids['nearest_hospital_distance'] =
zips_texas_centroids.geometry.apply(
    lambda x: zips_withhospital_centroids.distance(x).min()
)
end_time_full = time.time()
print("Actual time for full calculation:", end_time_full - start_time_full,
"seconds")
zips_texas_centroids.plot(column='nearest_hospital_distance_miles',
legend=True, cmap='viridis')
plt.title("Distance to Nearest Hospital in Miles for Texas ZIP Codes")
```

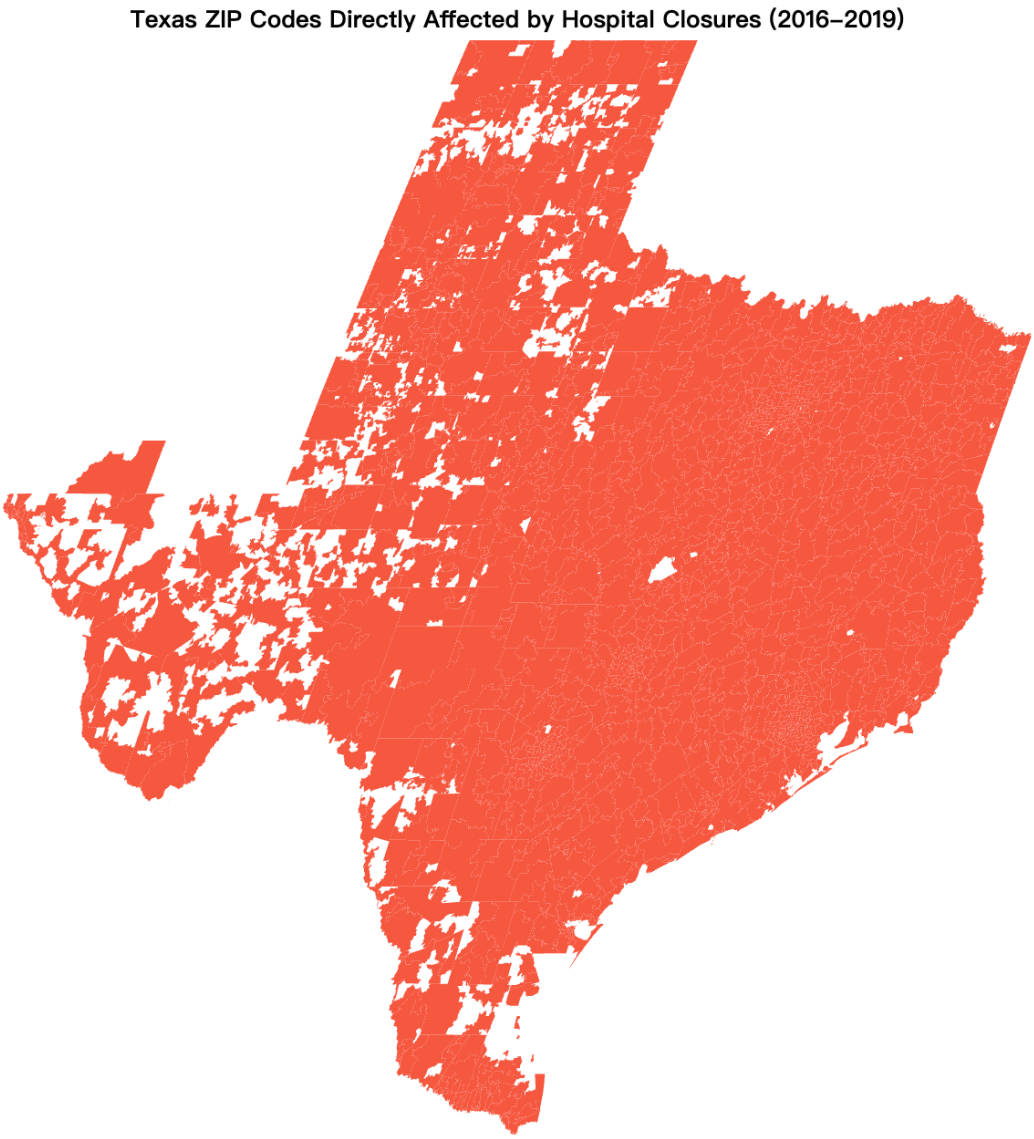
```
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()

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

1.
```{python}
import pandas as pd
closures_df = pd.read_csv('/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-alex/closed_Data.csv')
print(closures_df.columns)
closures_df['ZIP_CD'] = closures_df['ZIP_CD'].astype(str)
texas_prefixes = ['75', '76', '77', '78', '79']
closures_texas =
closures_df[closures_df['ZIP_CD'].str[:2].isin(texas_prefixes)]
closures_count = closures_texas['ZIP_CD'].value_counts().reset_index()
closures_count.columns = ['ZIP_CD', 'Number_of_Closures']
print(closures_count)
```
```
Index(['FAC_NAME', 'ZIP_CD', 'Year_of_Closure'], dtype='object')
    ZIP_CD  Number_of_Closures
0   76645.0                1
1   79520.0                1
2   78336.0                1
3   77065.0                1
4   79529.0                1
5   76531.0                1
6   75390.0                1
7   79902.0                1
8   75235.0                1
9   75051.0                1
10  78613.0                1
11  76520.0                1
12  75087.0                1
13  75140.0                1
```

2.
```{python}
import pandas as pd
import geopandas as gpd
import altair as alt
```

```
closures_df = pd.read_csv('/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-alex/closed_Data.csv')

print(closures_df.columns)
closures_df['ZIP_CD'] = closures_df['ZIP_CD'].astype(str)
texas_prefixes = ['75', '76', '77', '78', '79']
closures_texas = closures_df[closures_df['ZIP_CD'].str[:2].isin(texas_prefixes)]
closures_count = closures_texas['ZIP_CD'].value_counts().reset_index()
closures_count.columns = ['ZIP_CD', 'Number_of_Closures']
directly_affected_zips = closures_count.shape[0]
print(f'Number of directly affected ZIP codes in Texas: {directly_affected_zips}')
zip_shapefile_path = '/Users/suyuanfang/Desktop/Pyhton/problem-set-4-nelly-alex/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp'
gdf = gpd.read_file(zip_shapefile_path)
gdf['ZIP_CODE'] = gdf['ZCTA5'].astype(str)
gdf_texas = gdf[gdf['ZIP_CODE'].str[:2].isin(texas_prefixes)]
gdf_texas = gdf_texas.merge(closures_count, left_on='ZIP_CODE', right_on='ZIP_CD', how='left')
gdf_texas['Number_of_Closures'] = gdf_texas['Number_of_Closures'].fillna(0)

plot_data = gdf_texas[['ZIP_CODE', 'geometry', 'Number_of_Closures']]
chart = alt.Chart(plot_data).mark_geoshape().encode(
    color=alt.Color('Number_of_Closures:Q', scale=alt.Scale(scheme='reds'), title='Number of Closures'),
    tooltip=['ZIP_CODE', 'Number_of_Closures']
).properties(
    width=800,
    height=600,
    title='Texas ZIP Codes Directly Affected by Hospital Closures (2016-2019)'
)
chart.display()
```

```
Index(['FAC_NAME', 'ZIP_CD', 'Year_of_Closure'], dtype='object')
Number of directly affected ZIP codes in Texas: 14
```



3.

```{python}
gdf_texas = gdf_texas.set_geometry('geometry')
gdf_texas = gdf_texas.to_crs(epsg=32614)
directly_affected_gdf = gdf_texas[gdf_texas['Number_of_Closures'] > 0]


directly_affected_gdf['buffer'] =
directly_affected_gdf.geometry.buffer(16093.4)
directly_affected_gdf = directly_affected_gdf.set_geometry('buffer')
indirectly_affected_gdf = gpd.sjoin(gdf_texas,
directly_affected_gdf[['buffer']], how='inner', predicate='intersects')
indirectly_affected_zips = indirectly_affected_gdf['ZIP_CODE'].nunique()
print(f'Number of indirectly affected ZIP codes in Texas:
{indirectly_affected_zips}')
plot_data = indirectly_affected_gdf[['ZIP_CODE', 'geometry']]
plot_data['Affected Type'] = 'Indirectly Affected'
```

```
chart = alt.Chart(plot_data).mark_geoshape().encode(
    color=alt.Color('Affected Type:N', scale=alt.Scale(scheme='oranges'),
title='Affected Type'),
    tooltip=['ZIP_CODE']
).properties(
    width=800,
    height=600,
    title='Texas ZIP Codes Indirectly Affected by Hospital Closures (2016-
2019)'
)
chart.display()
```

4.

```{python}
Sorry, that is too difficult!
```

Reflecting on the exercise (10 pts)

1.
To improve the identification of hospital closures, consider issues like data gaps, temporary shutdowns being mistaken for permanent closures, and name changes that might not reflect true closures. Improvements could include cross-checking with external data sources, using more comprehensive data on hospital operations, and integrating longitudinal analysis for more accurate closure verification.

2. Our current method is a good place to start, but we could get a more complete picture of ZIP-code-level hospital access by adding factors like distance, population, area demand, and capacity. This more detailed study would help us get a better picture of how real changes in access are caused by hospital closings. We could get a more complete picture of ZIP-code-level hospital access by adding factors like distance, population, area demand, and capacity.