Problem Set 4: Spatial

```
# the following code will turn off any Python warnings in your code output
import warnings
warnings.filterwarnings("ignore")
```

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

Style Points (10 pts)

Submission Steps (10 pts)

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

(Note: Used ChatGPT to troubleshoot why pd.read_csv wasn't working- learned to specify a different encoding other than utf-8)

1. Which variables do we need to complete the exercise:

- Short-term hospitals -Certification Date (CRTFCTN_DT) -Change of ownership date (CHOW_DT) -Provider code 1 (PRVDR_CTGRY_CD) -subtype 1 (PRVDR_CTGRY_SBTYP_CD) -CMS Certification number (PRVDR_NUM) -Termination Code (PGM_TRMNTN_CD) -Termination Date (TRMNTN_EXPRTN_DT) -Facility Name (FAC_NAME) -Facility Zip (ZIP_CD)

2. Import the pos2016.csv file and subset it:

```
import pandas as pd
df = pd.read_csv('pos_2016.csv', encoding='Windows-1252', low_memory=False)
```

```python
import pandas as pd
df = pd.read_csv('pos_2016.csv', encoding='Windows-1252', low_memory=False)

df.columns = ["prvdr_subtype_code", "prvdr_category_code",
              "change_of_owner_date", "certification_date",
              "facility_name", "prvdr_num", "zip_code",
              "termination_code", "termination_date"]

# The date fields are numbers, so we need to change the numbers in the date
# fields to strings and then date objects. Attribution: Used ChatGPT to assist
# with regex issues
df['change_of_owner_date'] = pd.to_datetime(df['change_of_owner_date'].astype(
    str).str.replace('.0', '', regex=False), format='%Y%m%d')

df['certification_date'] = pd.to_datetime(df['certification_date'].astype(
    str).str.replace('.0', '', regex=False), format='%Y%m%d')

df['termination_date'] = pd.to_datetime(df['termination_date'].astype(
    str).str.replace('.0', '', regex=False), format='%Y%m%d')

# Convert zip code and termination codes to strings
df['termination_code'] = df['termination_code'].astype(str)
df['zip_code'] = df['zip_code'].astype(str)

df = df[(df["prvdr_category_code"] == 1) & (df["prvdr_subtype_code"] == 1)]
```

```
a. After subsetting to only short-term hospitals with these codes, we find 7245
hospitals in our data set.
b. The provided KFF article cites a number of "nearly 5000".This discrepancy
could be due to potential duplicates in our data (if a hospital was acquired
or merged into a newer hospital, it may be counted twice). Additionally, our
data may include hospitals that are short-term hospitals, but not specifically
acute-care hospitals providing emergency care/surgery and other interventions,
which the KFF article might exclude.
```

3. Repeat the previous steps with 2017Q4, 2018Q4, and 2019Q4 and then append them together:

```python
def read_clean_filter(pathname):
    """Reads the raw csv files from CMS, renames the columns, and filters to
    only short term hospitals.
    Also updates the date columns to pd.datetime format
```

```
    """
    df = pd.read_csv(pathname, encoding='Windows-1252', low_memory=False)

    df.columns = ["prvdr_subtype_code", "prvdr_category_code",
                  "change_of_owner_date", "certification_date",
                  "facility_name", "prvdr_num", "zip_code",
                  "termination_code", "termination_date"]

    df['change_of_owner_date'] = pd.to_datetime(df['change_of_owner_date'].astype(
        str).str.replace('.0', '', regex=False), format='%Y%m%d')

    df['certification_date'] = pd.to_datetime(df['certification_date'].astype(
        str).str.replace('.0', '', regex=False), format='%Y%m%d')

    df['termination_date'] = pd.to_datetime(df['termination_date'].astype(
        str).str.replace('.0', '', regex=False), format='%Y%m%d')

    df['termination_code'] = df['termination_code'].astype(str)
    df['zip_code'] = df['zip_code'].astype(str)

    df = df[(df["prvdr_category_code"] == 1) & (df["prvdr_subtype_code"] == 1)]

    return df


df_2017 = read_clean_filter("pos_2017.csv")
df_2018 = read_clean_filter("pos_2018.csv")
df_2019 = read_clean_filter("pos_2019.csv")

# Merge all dataframes from 2016-2019 together
# Use this for future pset problems

df_all = pd.concat([df, df_2017, df_2018, df_2019], ignore_index=True)
```

Now we plot the # of observations in the dataset by year of certification date:

```
import altair as alt

df_all['year'] = df_all['certification_date'].dt.year
summary = df_all.groupby('year').size().reset_index(name='count')

observations_by_year = alt.Chart(summary).mark_bar().encode(
```

```
    x=alt.X('year:O', title="Year"),
    y=alt.Y('count', title="Number of Observations")
)

observations_by_year.save('observations_by_year.png')
```

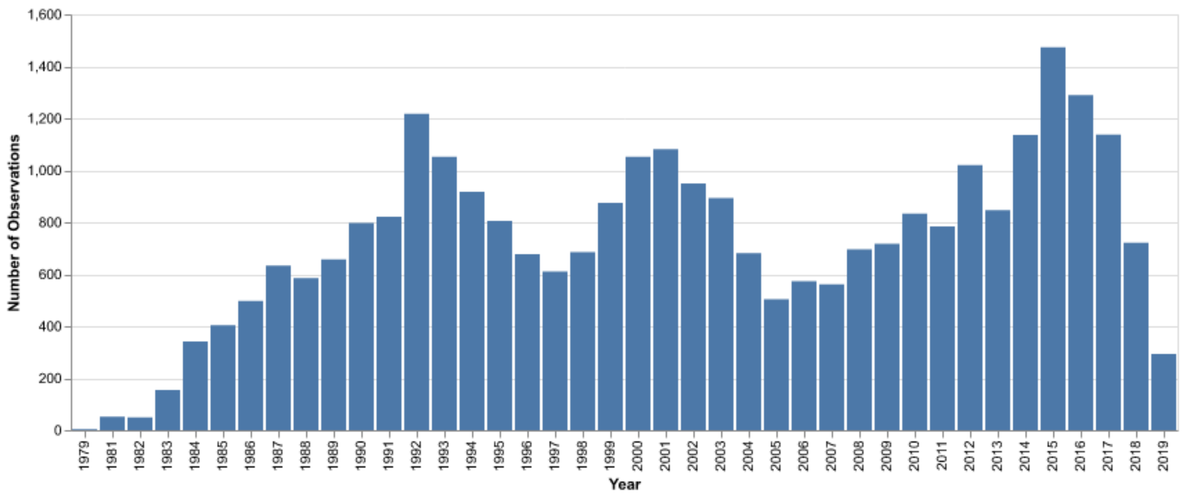


Figure 1: Observations per Year (2016-2019)

4.

a. Now we repeat the previous step but this time plot unique hospitals by year instead of just observations:

```
unique_hospitals = df_all.drop_duplicates(subset=['prvdr_num', 'year'])
unique_summary = unique_hospitals.groupby(
    'year').size().reset_index(name='count')

unique_hospitals_by_year = alt.Chart(unique_summary).mark_bar().encode(
    x=alt.X('year:O', title="Year"),
    y=alt.Y('count', title="Number of Unique Hospitals")
)

unique_hospitals_by_year.save('unique_hospitals_by_year.png')
```

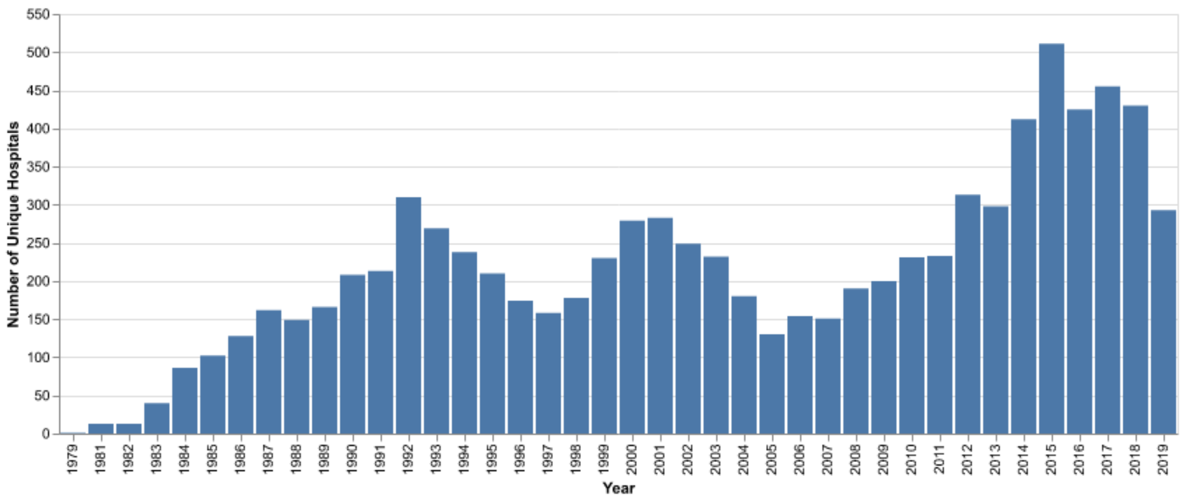


Figure 2: Unique Hospitals per Year (2016-2019)

b. Not only are the plots different, we can see two things:

1. The maximum number of hospitals is much smaller than the maximum number of observations, implying there are a lot of repeat rows in the data for the same facilities.
2. The graph of unique hospitals is much more skewed to the right than simply graphing observations, implying that either many new hospitals were certified from 2014 onwards, or that many old hospitals were merged/rebranded into new hospitals from 2014 onwards. Based on what we've read in the KFF article, it's highly likely that recent corporate/economic pressures caused hospitals to merge/rebrand leading to a rush of new certifications, rather than an actual increase in the amount of hospital capacity.

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

```
import geopandas
import numpy as np
import geopandas as gpd
import matplotlib as plt
```

1. Create a list of hospitals active in 2016, closed by 2019 (facility name, zip, year of closure). How many are there?

```
# Create a list of all hospitals active in 2016:
df_2016_active = df[df["termination_code"] == "0"]

# Now go back to original dataset and find zip codes where the number of
# active hospitals does not decrease in the year after the suspected closure.

# By definition, we don't have 2020 data for 2019 closures, and if we are
# looking at all 2016 active hospitals, 2017 is irrelevant.
# So we are looking only at hospitals that closed in 2017 and 2018.

closed_in_1718 = df_all[(df_all["termination_date"] >= "2017-01-01")
                        & (df_all["termination_date"] <= "2018-12-31")]

# Now remove duplicates:
closed_in_1718 = closed_in_1718.drop_duplicates(subset='prvdr_num')

# These are the hospitals in our dataset that closed in 2017-2018.
print(f"There are {len(closed_in_1718)
                   } hospitals that closed during this period.")
```

```
There are 94 hospitals that closed during this period.
```

There are 94 hospitals that fit this definition, but it's worth noting that some of the hospitals close and then reopen. Some of the reopenings are under different names but with the same provider code, and some are enitrely different.

2. Sort by name and report first 10.

```
# Limiting columns for rendering purposes
closed_in_1718_2 = closed_in_1718[['facility_name','zip_code','year']]
closed_in_1718_2.sort_values(by='facility_name', ascending=True, inplace=True)
closed_in_1718.sort_values(by='facility_name', ascending=True, inplace=True)
closed_in_1718_2.head(10)
```

	facility_name	zip_code	year
7413	ABRAZO MARYVALE CAMPUS	85031.0	2017
19343	AFFINITY MEDICAL CENTER	44646.0	2010
13485	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2017
26772	ALLIANCEHEALTH DEACONESS	73112.0	2016
14911	ARKANSAS CONTINUED CARE HOSPITAL OF JONESBORO	72401.0	2017

	facility_name	zip_code	year
21499	ASCENSION NE WISCONSIN MERCY CAMPUS	54904.0	2012
14697	BANNER PAYSON MEDICAL CENTER	85541.0	2018
21752	BAY AREA REGIONAL MEDICAL CENTER, LLC	77598.0	2016
29052	BAYLOR EMERGENCY MEDICAL CENTER	75087.0	2017
20642	BAYLOR SCOTT & WHITE MEDICAL CENTER GARLAND	75042.0	2010

3. Remove closures in ZIPs where number of active hospitals doesnt decrease

```
# Now for each closure, we need to check its zip code and see if it was a zip
# code where the number of active hospitals did not decrease the year after
# its closure.
# Each df_201x dataset is a snapshot of the state of hospitals in
# Q4-2017 or Q4-2018.

# How about this? Take the df_2017 data set only, and do a summary that counts
# the number of active hospitals by zip code. Then do the same for df_2018 and
# df_2019. Now we have three tables showing the breakdown of each zipcodes'
# active hospitals for each year.

# Attribution: Originally tried this with a for loop, but ChatGPT suggested
# using a dictionary to store the summary tables instead

active_hospital_summaries = {}

# Iterate through each DataFrame and generate the summary table
for year, df_x in zip(['2017', '2018', '2019'], [df_2017, df_2018, df_2019]):

    df_x['is_active'] = df_x['termination_code'] == "0"
    active_hospitals = df_x.groupby(
        'zip_code')['is_active'].sum().reset_index()

    # Store summary table in the dictionary
    active_hospital_summaries[f'active_hospitals_{year}'] = active_hospitals

# for readability
ah_2017 = active_hospital_summaries['active_hospitals_2017']
ah_2018 = active_hospital_summaries['active_hospitals_2018']
ah_2019 = active_hospital_summaries['active_hospitals_2019']


# Now we need to use these tables to create a way for us to check: when we
```

```
# provide our function a hospital closure, how can our function check to
# see if it was in a zip code where the number of active hospitals did not
# decrease the year after its closure?
# We can do this by taking our summary tables and merging them:

ah_2017 = ah_2017.rename(columns={'is_active': 'active_2017'})
ah_2018 = ah_2018.rename(columns={'is_active': 'active_2018'})

comp_2017_2018 = pd.merge(ah_2017, ah_2018, on="zip_code", how="outer")

# This handles any zip codes that were not present in 2017 or 2018 and now
# have a NA as part of the merge
comp_2017_2018.fillna(0, inplace=True)

# Now create a col that puts a boolean value if a zip code's active hospitals
# decreased from 2017-2018
comp_2017_2018['did_decrease'] = (
    comp_2017_2018['active_2017'] > comp_2017_2018['active_2018']).astype(int)

# And repeat for 2018-2019
ah_2019 = ah_2019.rename(columns={'is_active': 'active_2017'})
comp_2018_2019 = pd.merge(ah_2018, ah_2019, on="zip_code", how="outer")
comp_2018_2019.fillna(0, inplace=True)
comp_2018_2019['did_decrease'] = (
    comp_2018_2019['active_2017'] > comp_2018_2019['active_2018']).astype(int)

# Now we have our two comparison tables, so let's write a function that goes
# through closed_in_1718 that does the following:


def check_zip(closed_df, comp_2017_2018, comp_2018_2019):
    """
    # create a placeholder list for decreasing zipcodes (see below)
    # take the zipcode of the current row and check the termination_date's year
    # depending on whether the termination_date's year is 2017 or 2018, refer
    # to the comp_2017_2018 or the comp_2018_2019 table
    # Take the zipcode of the current row and check whether that zipcode has a
    # 0 or 1 in the "did_decrease" column
    # if that zipcode has a 1 in that column, add it to the decreasing zipcodes
    # list the function should return a list of all zipcodes that had decreasing
    # hospitals
    """
```

```python
    decreasing_zipcodes = []

    zip_codes = closed_df['zip_code']
    termination_years = closed_df['termination_date'].dt.year

    for zip_code, year in zip(zip_codes, termination_years):
        # Select the appropriate comparison table based on the termination year
        if year == 2017:
            comp_table = comp_2017_2018
        elif year == 2018:
            comp_table = comp_2018_2019
        else:
            # Skip if the termination year is not 2017 or 2018
            continue

        # Check if the zip code exists in the comparison table and if it
        # shows a decrease
        if zip_code in comp_table['zip_code'].values:
            did_decrease = comp_table.loc[comp_table['zip_code']
                                          == zip_code, 'did_decrease'].values[0]

            # If the zip code had a decrease in active hospitals, add it to the list
            if did_decrease == 1:
                decreasing_zipcodes.append(zip_code)

    return decreasing_zipcodes

decreasing_zipcodes = check_zip(closed_in_1718, comp_2017_2018, comp_2018_2019)
```

a. Now we have a list of decreasing zipcodes, these are likely mergers/acquisitions:

::: {.cell execution_count=11} `{.python .cell-code}` `print(len(decreasing_zipcodes))`

::: {.cell-output .cell-output-stdout} `12` ::: :::

There are 12 hospitals that fit this definition

b. After correcting for this, how many hospitals do you have left?

94 - 12 hospitals = 82 hospitals left.

c. Sort this list of corrected hospital closures by name and report the first 10 rows.

```
# Attribution: ChatGPT- Using ~ is the "not" operator (originally tried to
# do this with an != sign)
corrected_closures = closed_in_1718[
    ~closed_in_1718['zip_code'].isin(decreasing_zipcodes)]

corrected_closures.sort_values(
    by='facility_name', ascending=True, inplace=True)

# Limiting columns for rendering purposes
corrected_closures_2 = corrected_closures[['facility_name','zip_code','year']]
corrected_closures_2.head(10)
```

	facility_name	zip_code	year
7413	ABRAZO MARYVALE CAMPUS	85031.0	2017
19343	AFFINITY MEDICAL CENTER	44646.0	2010
13485	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2017
26772	ALLIANCEHEALTH DEACONESS	73112.0	2016
14911	ARKANSAS CONTINUED CARE HOSPITAL OF JONESBORO	72401.0	2017
21499	ASCENSION NE WISCONSIN MERCY CAMPUS	54904.0	2012
14697	BANNER PAYSON MEDICAL CENTER	85541.0	2018
21752	BAY AREA REGIONAL MEDICAL CENTER, LLC	77598.0	2016
29052	BAYLOR EMERGENCY MEDICAL CENTER	75087.0	2017
20642	BAYLOR SCOTT & WHITE MEDICAL CENTER GARLAND	75042.0	2010

Download Census zip code shapefile (10 pt)

1. a. The five file types are:

    1. .shp - The file that contains the actual geoms (points, lines, polygons) that will be displayed
    2. .dbf - Stores attribute data for each shape in the .shp file, like population, income, temperature, etc.
    3. .shx - An index file that points to specific shapes in the .shp file for easier navigation
    4. .prj - A projection file that stores the coordinate information for latitude/longitude
    5. .xml - Metadata file that is mostly used for documentation/readme purposes, including data sources, attributes, etc.

    b. The .shp file is the largest at 837.5 MB, while the other files are: .dbf - 6.4 MB .shx - 0.26 MB .xml - 16 KB .prj - 0.16 KB

2. To open the shapefile, we will install and use geopandas:

```
zip_codes = gpd.read_file(
    "/Users/jakub/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp")

# Texas zip codes always start with 75, 76, or 77, and ZCTA5 is the column with
# the zip codes Used ChatGPT to find a function that can identify strings by
# the first few characters
tx_zip_codes = zip_codes[
    zip_codes["ZCTA5"].str.startswith(('75', '76', '77', '78', '79'))]
```

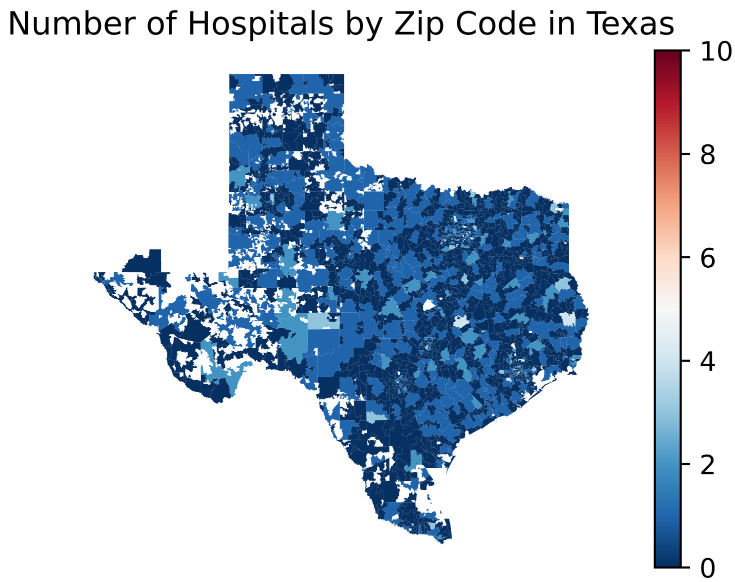
Now we calculate the # of hospitals per zip code in 2016 based on the previous step: (note: we are using the entire dataset of hospitals short term and active in 2016- this is the initial df file from Q4 2016)

```
# Note: This counts only short term, active hospitals in Texas

texas_df_2016 = df[df['zip_code'].astype(str).str.startswith(
    ('75', '76', '77', '78', '79'))]


texas_summary_2016 = texas_df_2016.groupby('zip_code').size().reset_index()

# Remove trailing decimal points from zip code
texas_summary_2016['zip_code'] = texas_summary_2016['zip_code'].astype(
    str).str.split('.').str[0].str.zfill(5)

texas_summary_2016.columns = ['zip_code', 'hospital_count']
print(texas_summary_2016.sort_values(by="hospital_count", ascending=False))
```

```
    zip_code  hospital_count
524    79902              10
239    77054               7
135    76104               6
52     75235               6
236    77030               6
..       ...             ...
10     75041               1
13     75057               1
15     75063               1
18     75071               1
515    79756               1

[531 rows x 2 columns]
```

Now we will create a choropleth of the hospitals in Texas by zip code:

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

# We need to merge our previous hospital data with the shapefile zip_codes data:

tx_zip_hospitals = tx_zip_codes.merge(
    texas_summary_2016, left_on="ZCTA5", right_on="zip_code", how="left")

# Replace NaN hospital counts with 0
tx_zip_hospitals['hospital_count'] = tx_zip_hospitals[
    'hospital_count'].fillna(0).astype(int)

# Attribution: ChatGPT suggested defining custom bins to make it easier
# to differentiate regions

tx_zip_hospitals.plot(
    column='hospital_count',
    cmap='RdBu_r',
    legend=True
)
plt.title("Number of Hospitals by Zip Code in Texas")
plt.axis('off')
plt.savefig("texas_zip_map.png", format='png', bbox_inches='tight', dpi=300)
```



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

1. GeoDataFrame for centroid of each zip code nationally, dimensions, columns.

```
zips_all_centroids = zip_codes.copy()
zips_all_centroids['centroid'] = zip_codes['geometry'].centroid
zips_all_centroids.total_bounds
print(zips_all_centroids.columns)
```

```
Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry',
       'centroid'],
      dtype='object')
```

The total bounds (dimensions) of the resulting GeoDataFrame are [-176.684744, 17.910817, -65.224638, 71.341324]. The columns are: - GEO_ID - a unique identifier for some geographic area. The last 5 numbers appear to the be zip code. - ZCTA5 - Zip Code Tabulation Area, represents 1 or a group of zip codes. - NAME - name of the area, appears to be the same as zip code. - geometry - identifies the shape (polygon or multipolygon), as well as the specific points it uses to create the shape. - centroid - identifies the centroid of the shape as a single point.

2. Subset above into texas & texas or bordering state.

```
# Convert Texas zips into list & cross-reference
tx_zip_codes_list = tx_zip_codes['ZCTA5'].tolist()
zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].isin(
    tx_zip_codes_list)]
print(len(zips_texas_centroids))

# Oklahoma, Arkansas, Louisiana, New Mexico
tx_border_zip_codes = zip_codes[zip_codes["ZCTA5"].str.startswith(
    ('7', '87', '88'))]
tx_border_zip_codes_list = tx_border_zip_codes['ZCTA5'].tolist()
zips_texas_borderstates_centroids = zips_all_centroids[zips_all_centroids[
    'ZCTA5'].isin(tx_border_zip_codes_list)]
print(len(zips_texas_borderstates_centroids))
```

```
1935
4057
```

There are 1,935 unique zip codes in Texas and 4,057 unique zip codes in Texas or a bordering state.

3. Subset that contains at least 1 hospital

```
# Create temporary, smaller dataframes for this purpose
df_2016_temp = df[['facility_name', 'prvdr_num',
                   'zip_code', 'termination_code']]
df_2017_temp = df_2017[['facility_name',
                        'prvdr_num', 'zip_code', 'termination_code']]
df_2018_temp = df_2018[['facility_name',
                        'prvdr_num', 'zip_code', 'termination_code']]
df_2019_temp = df_2019[['facility_name',
                        'prvdr_num', 'zip_code', 'termination_code']]
# Merge them on provider number to have all the term codes together by year
df_terms = df_2016_temp.merge(df_2017_temp, on=['prvdr_num'],
                              how='outer', suffixes=('', '_2017')
                              ).merge(df_2018_temp, on=['prvdr_num'],
                                      how='outer', suffixes=('', '_2018')
                                      ).merge(df_2019_temp, on=['prvdr_num'],
                                              how='outer', suffixes=('', '_2019'))
# Rename 2016 column & filter for only those active in 2016
df_terms = df_terms.rename(
    columns={'termination_code': 'termination_code_2016',
```

```
    'zip_code': 'zip_code_2016'})
df_terms = df_terms[df_terms['termination_code_2016'] == '0']
```

```
# Make sure they're the same type
active_2016 = df_terms['zip_code_2016'].str.replace(
    '.0', '', regex=False).astype(int)
zips_texas_borderstates_centroids['ZCTA5'] = zips_texas_borderstates_centroids[
    'ZCTA5'].astype(int)

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    active_2016, left_on='ZCTA5', right_on='zip_code_2016', how='inner')
print(len(zips_withhospital_centroids))
```

```
556
```

We did an inner merge because we only want to keep the rows from zips_texasborderstates_centroids that have an active hospital in them and retain the geodata. We don't need any of the other information from the hospitals dataset. We merged left on ZCTA5 and right on zip_code_2016 because both of these variables contain zip codes. The result is 556 Texas or bordering states zip codes with hospitals out of the 4,057 total zip codes we found earlier.

4. Calculate distance to nearest zip code w hospital.
    a. Subset to 10 & time it. Estimate total time.

```
# Subset only 10 observations. data_for_join only zcta5 and geometry
zips_texas_centroids_10 = zips_texas_centroids.head(10)
data_for_join_10 = zips_texas_centroids_10[["ZCTA5", "geometry"]]

# Spatial join based on lecure slides
# Asked ChatGPT how to time a function to get that part
start_time_10 = time.time()
join_to_hospital_10 = gpd.sjoin_nearest(
    data_for_join_10,
    zips_withhospital_centroids,
    how='inner',
    distance_col="distance_hospital")
end_time_10 = time.time()

elapsed_time_10 = end_time_10 - start_time_10
print(elapsed_time_10)
full_estimate = elapsed_time_10 * (556/10)
print(full_estimate)
```

```
0.6405501365661621
35.61458759307862
```

It took about 0.6 seconds to run this subset of 10 zip codes on my machine, so I would estimate that the full data would take about 55.6 times longer, or about 35 seconds (the exact times are printed above).

b. Full calculation

```
data_for_join = zips_texas_centroids[["ZCTA5", "geometry"]]

start_time = time.time()
join_to_hospital = gpd.sjoin_nearest(
    data_for_join,
    zips_withhospital_centroids,
    how='inner',
    distance_col="distance_hospital")
end_time = time.time()

elapsed_time = end_time - start_time
print(elapsed_time)
```

```
316.89579582214355
```

The full calculation took 310s, which is almost 10 times larger than my estimate.

c. Units

```
join_to_hospital['distance_hospital_miles'] = join_to_hospital[
    'distance_hospital'] * 69
```

The .prj file reports the units as UNIT[“Degree”,0.017453292519943295]. According to the USGS, one degree of latitude is about 69 miles.

5. Average Distances within zipcodes in Texas.

   a. What unit?

```python
avg_dist_hospital = join_to_hospital.groupby(
    'ZCTA5_left')['distance_hospital'].mean(
).reset_index()

# Sort by descending for rendering purposes
avg_dist_hospital_2 = avg_dist_hospital.sort_values(
    by='distance_hospital', ascending=False)
avg_dist_hospital_2.head()
```

	ZCTA5_left	distance_hospital
1900	79845	1.247494
1901	79846	1.115968
1673	79087	0.959087
1908	79854	0.862142
1906	79852	0.781770

The first 10 results are printed above. The units are degrees (of latitude).

`b. Convert to miles. Makes sense?`

```python
avg_dist_hospital['distance_hospital_miles'] = avg_dist_hospital[
    'distance_hospital'] * 69
avg_dist_hospital = avg_dist_hospital.rename(columns={'ZCTA5_left': 'ZCTA5'})

np.mean(avg_dist_hospital['distance_hospital_miles'])
```

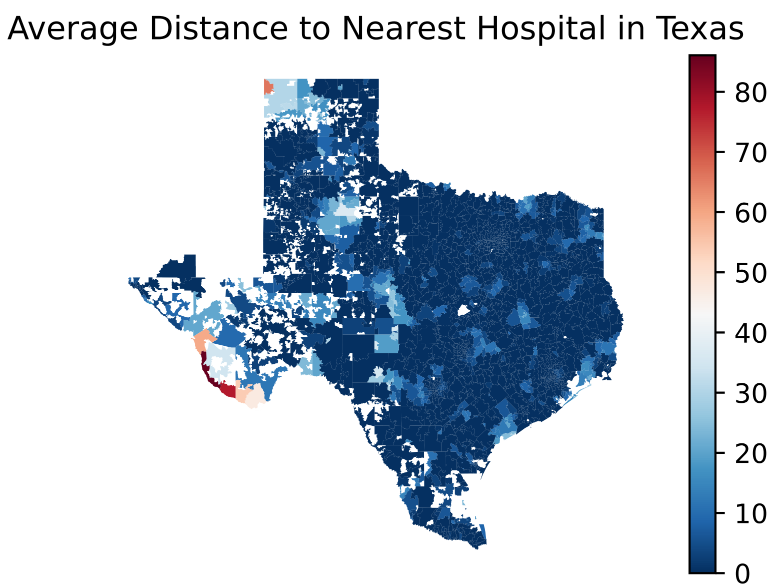
```
np.float64(2.754165545455319)
```

The average distance to a hospital shows as 2.75 miles. This value does not make sense and is not entirely correct because we are assuming that within zip code distance is 0 miles. Zip code areas can range from small to large and it could be the case that someone lives 10 miles away, but in the same zip code, and they are still being classified as 0 miles. This number is grossly underprojected.

`c. Map the value for each zip.`

```
avg_per_zip = zips_texas_centroids.merge(
    avg_dist_hospital,
    on = 'ZCTA5',
    how='left')
```

```
# Plot Texas. ChatGPT was used for assistance on the colors and
# how to save as png
avg_per_zip.plot(column='distance_hospital_miles', cmap='RdBu_r', legend=True)
plt.title("Average Distance to Nearest Hospital in Texas")
plt.axis('off')
plt.savefig("texas_map.png", format='png', bbox_inches='tight', dpi=300)
```



Effects of closures on access in Texas (15 pts)

1.
2.
3.
4.

Reflecting on the exercise (10 pts)

1a. Potential issues: we can only measure "true closure", by that definition above, for hospitals who were suspected to close in 2017 or 2018. This is because we restricted the data to hospitals active in 2016 (so we're not measuring their potential closings), and we only have data up until 2019 (so we can't use 2020 data to determine if the closings of 2019 were true by the zip codes definition). Besides being limited to 2 years, another issue is that we are looking at total active hospitals in a zip code and using that to determine whether a closure was real. Hypothetically, if a hospital closed and a new one opened, that closure would be counted as fake. Futher, if a hospital closed in one zip code and moved to another (or two from different zips merged into one), we would be counting that as a "true" closure, when it was just a relocation that could have been hypothetically down the street, but in a different zip code. Lastly, if a hospital temporarily closed and was then reopened a year later, our methods would count this as a true closure, when that may no the case. Better way: One solution is to look at clusters of zip codes. For example, instead of just looking at closures in 1 zip code, we can merge each zip code with each bordering zip code to create clusters which could avoid the issues of mergers or short relocations. We could also use the other hospital types to avoid a potential issue of a hospital reclasifying from short-term to something else, but still existing.

2. Classifying zip codes as 'directly affected', 'indirectly affected', and 'not affected' with a threshold of 10 miles does do the a good job of reflecting changes in zip-code-level access to hospitals. From a google search, it's apparent that the average distance to a hospital in the United States is about 10 miles in rural areas (which is likely how this number was derived). To improve this measure, we would have to calculate the average and max distance to a hospital in each zip code and make those the parameters (average or less is directly affected, average to max is indirectly affected, max or more is not affected). This would be tailored to each zip code and would give us a better representation of the changes in access per zip code while taking into account their differences.