# 30538 Problem Set 4: Spatial

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

## 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): Wei Chen (Ashley) Liu, wliu10 • Partner 2 (name and cnet ID): Ruiyi Wu, ruiyi
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: ***AL RW***
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: ***0*** Late coins left after submission: ***4***
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

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

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?

PRVDR_CTGRY_SBTYP_CD == 01 #short-term PRVDR_CTGRY_CD == 01 #hospital PRVDR_NUM #CMS PGM_TRMNTN_CD #termination CRTFCTN_ACTN_TYPE_CD == 3 #termination ZIP_CD #zip code FAC_NAME #facility name

```python
import pandas as pd
import os
absolute_path = r"/Users/ashleyliu/Downloads"
path = os.path.join(absolute_path, "pos2016.csv")
pos2016 = pd.read_csv(path)
```

2. 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?
   a.

```
st_hospitals2016 = pos2016[(pos2016["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (pos2016["PRVDR_CTGRY_CD"] == 1)]
st_hospitals2016.shape
```

```
(7245, 7)
```

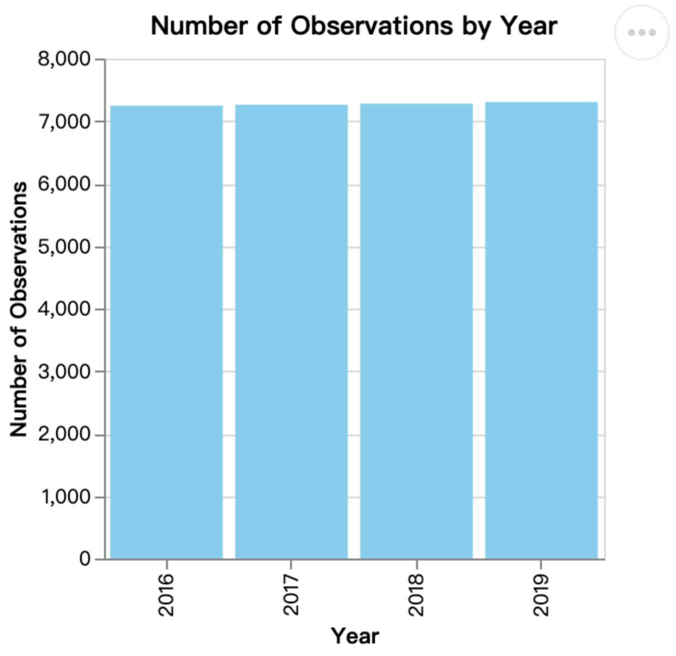
b.

There are 7245 hospitals are reported in this data. From Statista, the website, "Number of all hospitals in the United States from 1975 to 2022", shows that there were 5534 hospitals in 2016. Based on these two numbers, there is a discrepancy, and there are many factors can cause this to happen. For example, the difference in definition to count as hospitals, or data errors like dupilication can both cause the difference in numbers.

3. 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.

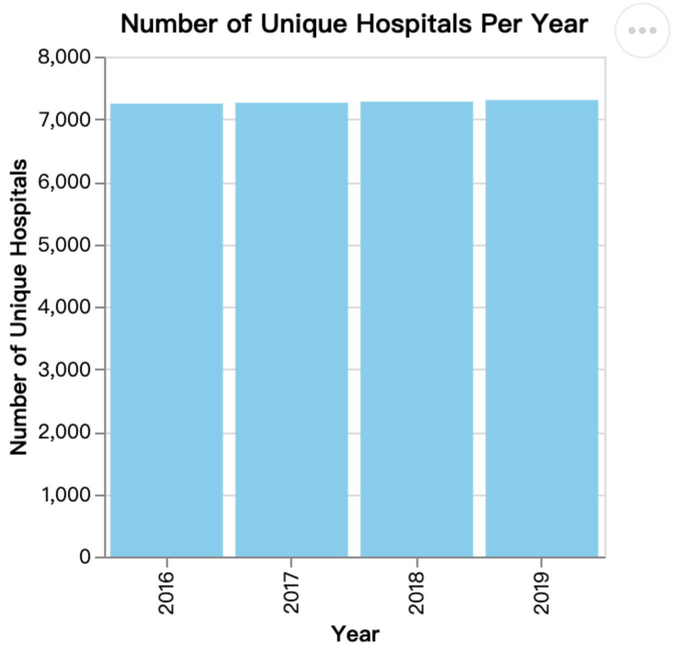
```
st_hospitals2016["year"] = 2016

path = os.path.join(absolute_path, "pos2017.csv")
pos2017 = pd.read_csv(path)
st_hospitals2017 = pos2017[(pos2017["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (pos2017["PRVDR_CTGRY_CD"] == 1)]
st_hospitals2017["year"] = 2017

pos2018 = pd.read_csv(r"/Users/ashleyliu/Downloads/pos2018.csv", encoding = "latin1")
st_hospitals2018 = pos2018[(pos2018["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (pos2018["PRVDR_CTGRY_CD"] == 1)]
st_hospitals2018["year"] = 2018

pos2019 = pd.read_csv(r"/Users/ashleyliu/Downloads/pos2019.csv", encoding = "latin1")
st_hospitals2019 = pos2019[(pos2019["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (pos2019["PRVDR_CTGRY_CD"] == 1)]
st_hospitals2019["year"] = 2019
```

```
combined_df = pd.concat([st_hospitals2016, st_hospitals2017, st_hospitals2018,
        st_hospitals2019], ignore_index = True)
import altair as alt
year_counts = combined_df["year"].value_counts().reset_index()
year_counts.columns = ["year", "count"]
alt.Chart(year_counts).mark_bar(color = "skyblue").encode(
    alt.X("year:O", title = "Year", sort = "ascending"),
    alt.Y("count:Q", title = "Number of Observations")
).properties(
    title = "Number of Observations by Year",
    width = 250,
    height = 250
)
```



4. 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?
   a.

```
unique_hospitals_per_year = (
    combined_df
    .groupby("year")["PRVDR_NUM"]
    .nunique()
    .reset_index()
)
unique_hospitals_per_year.columns = ["year", "unique_hospital_count"]
alt.Chart(unique_hospitals_per_year).mark_bar(color = "skyblue").encode(
    alt.X("year:O", title = "Year"),
    alt.Y("unique_hospital_count:Q", title = "Number of Unique Hospitals")
).properties(
    title = "Number of Unique Hospitals Per Year",
    width = 250,
    height = 250
)
```



b.

Compare to the previous plot, the whole plot looks smiliar to each other, and the counts are close for each year as well, which suggests that the data processing and summarization steps were executed correctly. Also, the counts in the number of unique hospitals increase slightly over year in both plots, which may indicate growth in healthcare services, possibly due to increasing demand, changes in policy, or other factors.

## 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?

```
active_2016 = pos2016[pos2016['PGM_TRMNTN_CD'] == 0][['PRVDR_NUM', 'FAC_NAME',
        'ZIP_CD']]
filter_2017 = pd.merge(active_2016, pos2017, on='PRVDR_NUM', how='left')

df_filtered_2017 = filter_2017[(filter_2017['PGM_TRMNTN_CD'] != 0) &
        (filter_2017['PGM_TRMNTN_CD'].notna())]

df_filtered_2017['closureday'] = 2017
```

```
active_2017 = pos2017[pos2017['PGM_TRMNTN_CD'] == 0][['PRVDR_NUM', 'FAC_NAME',
        'ZIP_CD']]
filter_2018 = pd.merge(active_2017, pos2018, on='PRVDR_NUM', how='left')

df_filtered_2018 = filter_2018[(filter_2018['PGM_TRMNTN_CD'] != 0) &
        (filter_2018['PGM_TRMNTN_CD'].notna())]
```

```
df_filtered_2018['closureday'] = 2018
```

```
active_2018 = pos2018[pos2018['PGM_TRMNTN_CD'] == 0][['PRVDR_NUM', 'FAC_NAME',
        'ZIP_CD']]
filter_2019 = pd.merge(active_2018, pos2019, on='PRVDR_NUM', how='left')

df_filtered_2019 = filter_2019[(filter_2019['PGM_TRMNTN_CD'] != 0) &
        (filter_2019['PGM_TRMNTN_CD'].notna())]

df_filtered_2019['closureday'] = 2019
```

```
result_1 = pd.concat([df_filtered_2017, df_filtered_2018, df_filtered_2019],
        ignore_index=True)
result = result_1[['PRVDR_NUM', 'FAC_NAME_x', 'ZIP_CD_x', 'closureday']]
result
```

| | PRVDR_NUM | FAC_NAME_x | ZIP_CD_x | closureday |
|---|---|---|---|---|
| 0 | 011504 | GADSDEN REGIONAL HOSPICE | 35901.0 | 2017 |
| 1 | 011553 | SOUTHERNCARE MONTGOMERY | 36117.0 | 2017 |
| 2 | 011554 | SOUTHERNCARE GADSDEN | 35901.0 | 2017 |
| 3 | 011569 | SOUTHERNCARE GROVE HILL | 36451.0 | 2017 |
| 4 | 011578 | SOUTHERNCARE DEMOPOLIS | 36732.0 | 2017 |
| ... | ... | ... | ... | ... |
| 5637 | 921692 | LUGA HOSPICE CARE, INC | 91352.0 | 2019 |
| 5638 | 921738 | JCH MEDCARE SERVICES, LLC | 90502.0 | 2019 |
| 5639 | 921749 | PARAGON HOME HEALTH CARE & HOSPICE, INC. | 95035.0 | 2019 |
| 5640 | 941004 | ICHC - FERNDALE PIONEER | 98248.0 | 2019 |
| 5641 | 941010 | HEALTHPOINT VALLEY CITIES ENUMCLAW | 98022.0 | 2019 |

5642 rows × 4 columns

There are 5642 hospitals that fit this definition.

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

```
result = result.sort_values(by='FAC_NAME_x').reset_index(drop=True)
result = result.rename(columns={'FAC_NAME_x': 'FAC_NAME'})
result = result.rename(columns={'ZIP_CD_x': 'ZIP_CD'})
result.head(10)
```

|   | PRVDR_NUM | FAC_NAME | ZIP_CD | closureday |
|---|---|---|---|---|
| 0 | 44L024 | (CLOSED) REFLECTIONS TREATMENT AGENCY | 37912.0 | 2019 |
| 1 | 679612 | 1ST CHOICE HEALTHCARE SERVICES INC | 77063.0 | 2019 |
| 2 | 679339 | 1ST CHOICE HOME HEALTH | 75965.0 | 2017 |
| 3 | 148217 | 1ST FAMILY HOME HEALTHCARE, INC | 60098.0 | 2018 |
| 4 | 109253 | 1ST HOME HEALTH CARE INC | 33155.0 | 2018 |
| 5 | 377614 | 21ST CENTURY HOME HEALTH AGENCY | 73013.0 | 2017 |
| 6 | 037483 | 24/7 AVARE HEALTHCARE PHOENIX | 85029.0 | 2019 |
| 7 | 239281 | 247 HOME HEALTH CARE | 48180.0 | 2019 |
| 8 | 148118 | 24SEVEN HEALTH CARE SERVICES, INC | 60659.0 | 2018 |
| 9 | 14C0001027 | 25 EAST SAME DAY SURGERY CENTE | 60602.0 | 2019 |

3. 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.
    a. 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

```
active_counts_2017 = pos2017[pos2017['PGM_TRMNTN_CD'] ==
        0].groupby('ZIP_CD').size().rename('active_2017')
active_counts_2018 = pos2018[pos2018['PGM_TRMNTN_CD'] ==
        0].groupby('ZIP_CD').size().rename('active_2018')
active_counts_2019 = pos2019[pos2019['PGM_TRMNTN_CD'] ==
        0].groupby('ZIP_CD').size().rename('active_2019')
result_1 = result_1.rename(columns={'ZIP_CD_x': 'ZIP_CD'})
result_2 = result_1.merge(active_counts_2017, on='ZIP_CD', how='left')
result_3 = result_2.merge(active_counts_2018, on='ZIP_CD', how='left')
result_4 = result_3.merge(active_counts_2019, on='ZIP_CD', how='left')

def is_true_closure(row):
    if row['closureday'] == 2017:
        return row['active_2017'] > row['active_2018']
    elif row['closureday'] == 2018:
        return row['active_2018'] > row['active_2019']
    elif row['closureday'] == 2019:
        return pd.notna(row['active_2019'])
    return False

result_4['true_closure'] = result_4.apply(is_true_closure, axis=1)
potential_mergers = result_4[~result_4['true_closure']]
result_4 = result_4[result_4['true_closure']]

final_count = result_4.shape[0]
potential_merger_count = potential_mergers.shape[0]
```

```
final_count, potential_merger_count
```

```
(2524, 3118)
```

3118 hospitals fit this definition of potentially being a merger/acquisition. After correcting for this, 2524 hospitals have left.

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

```
result_4 = result_4[['PRVDR_NUM', 'FAC_NAME_x', 'ZIP_CD', 'closureday',
         'PRVDR_CTGRY_SBTYP_CD', 'PRVDR_CTGRY_CD', 'PGM_TRMNTN_CD',
         'CRTFCTN_ACTN_TYPE_CD']]
result_4 = result_4.rename(columns={'FAC_NAME_x': 'FAC_NAME'})
result_sorted = result_4.sort_values(by='FAC_NAME').reset_index(drop=True)
top_10_closures = result_sorted.head(10)

top_10_closures
```

| | PRVDR_NUM | FAC_NAME | ZIP_CD | closureday | PRVDR_CTGRY_SBTYP_CD | PRVDR_CTGRY_CD | PGM_TR |
|---|---|---|---|---|---|---|---|
| 0 | 679612 | 1ST CHOICE HEALTHCARE SERVICES INC | 77063.0 | 2019 | NaN | 5.0 | 6.0 |
| 1 | 109253 | 1ST HOME HEALTH CARE INC | 33155.0 | 2018 | NaN | 5.0 | 1.0 |
| 2 | 037483 | 24/7 AVARE HEALTHCARE PHOENIX | 85029.0 | 2019 | NaN | 5.0 | 1.0 |
| 3 | 239281 | 247 HOME HEALTH CARE | 48180.0 | 2019 | NaN | 5.0 | 1.0 |
| 4 | 148118 | 24SEVEN HEALTH CARE SERVICES, INC | 60659.0 | 2018 | NaN | 5.0 | 1.0 |
| 5 | 14C0001027 | 25 EAST SAME DAY SURGERY CENTE | 60602.0 | 2019 | NaN | 15.0 | 1.0 |
| 6 | 148135 | 3 ANGELS HOME HEALTH | 60077.0 | 2019 | NaN | 5.0 | 1.0 |

|   | PRVDR_NUM | FAC_NAME | ZIP_CD | closureday | PRVDR_CTGRY_SBTYP_CD | PRVDR_CTGRY_CD | PGM_TR |
|---|---|---|---|---|---|---|---|
| 7 | 331192 | 35TH STREET HEALTH CENTER | 10018.0 | 2019 | NaN | 21.0 | 4.0 |
| 8 | 059176 | 446 SCHOLL HOME HEALTH CARE, INC | 91411.0 | 2019 | NaN | 5.0 | 4.0 |
| 9 | 461598 | 5 STAR HOSPICE LLC | 84097.0 | 2017 | 1.0 | 16.0 | 1.0 |

## Download Census zip code shapefile (10 pt)

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?

   a. The types of the five files are .xml, .shx, .shp, .prj, and .dbf, which refer to Metadata File (.xml), Shape Index File (.shx), Shapefile (.shp), Projection File (.prj), and Database File (.dbf). Metadata File contains metadata about the dataset, such as author, creation date, description, and licensing information, providing context and documentation about the dataset's contents and purpose. Shape Index File is an index file that speeds up access to the geometry in the .shp file. It allows for quicker data retrieval, especially when dealing with large datasets, required alongside the .shp file to properly read and display spatial features. Shapefile is the core file that contains the geometry of the spatial features (e.g., points, lines, polygons). It stores the actual location data (coordinates) for each spatial feature required for reading spatial data in most GIS software. Projection File defines the coordinate system and projection information of the spatial data, allowing GIS software to interpret the spatial features correctly on a map and align it with other data in the same projection. Database File stores attribute data in a tabular format, linked to each feature in the .shp file. The .dbf file contains non-spatial information, like names, population counts, or other descriptive details associated with each spatial feature, required to complete the dataset with meaningful attribute information.

   b.

```python
directory_path = "/Users/ashleyliu/Downloads/gz_2010_us_860_00_500k"
files = ["gz_2010_us_860_00_500k.xml", "gz_2010_us_860_00_500k.shx",
         "gz_2010_us_860_00_500k.shp", "gz_2010_us_860_00_500k.prj",
         "gz_2010_us_860_00_500k.dbf"]
for file in files:
    file_path = os.path.join(directory_path, file)
    file_size = os.path.getsize(file_path) / (1024 * 1024)
    print(f"{file}: {file_size:.2f} MB")
```

```
gz_2010_us_860_00_500k.xml: 0.01 MB
gz_2010_us_860_00_500k.shx: 0.25 MB
gz_2010_us_860_00_500k.shp: 798.74 MB
gz_2010_us_860_00_500k.prj: 0.00 MB
gz_2010_us_860_00_500k.dbf: 6.13 MB
```

2. 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.

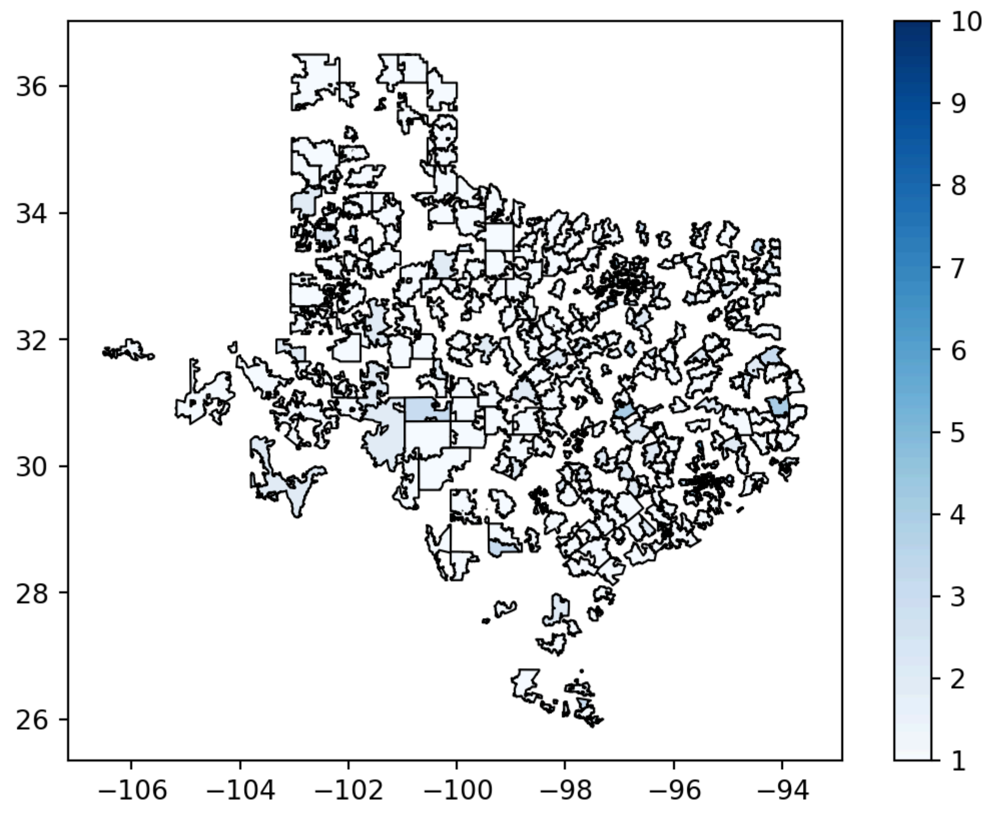
```python
import geopandas as gpd
zip_codes = gpd.read_file(os.path.join(directory_path, "gz_2010_us_860_00_500k.shp"))
zip_codes_tx = zip_codes[zip_codes["NAME"].str.startswith(("75", "76", "77", "78",
        "79", "885"))]
```

```python
st_hospitals2016["ZIP_CD"] = st_hospitals2016["ZIP_CD"].astype(int)
hospitals_per_zip_2016 = st_hospitals2016.groupby("ZIP_CD").size().reset_index(name =
        "hospital_count")
hospitals_per_zip_2016
```

| | ZIP_CD | hospital_count |
|---|---|---|
| 0 | 603 | 1 |
| 1 | 613 | 1 |
| 2 | 614 | 1 |
| 3 | 618 | 1 |
| 4 | 625 | 1 |
| ... | ... | ... |
| 5671 | 99801 | 1 |
| 5672 | 99833 | 1 |
| 5673 | 99835 | 2 |
| 5674 | 99901 | 1 |
| 5675 | 99929 | 1 |

5676 rows × 2 columns

```python
zip_codes_tx["NAME"] = zip_codes_tx["NAME"].astype(int)
hospitals_per_zip_2016["ZIP_CD"] = hospitals_per_zip_2016["ZIP_CD"].astype(int)
merged_data = zip_codes_tx.merge(hospitals_per_zip_2016, left_on = "NAME", right_on =
        "ZIP_CD", how = "left")
merged_data.plot(
    column = "hospital_count",
    cmap = "Blues",
    linewidth = 0.8,
    edgecolor = "black",
    legend = True
)
```



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

1. 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?

```
zip_codes["centroid"] = zip_codes.geometry.centroid
zips_all_centroids = zip_codes.set_geometry("centroid")
dimensions = zips_all_centroids.shape
print(f"Dimensions of the GeoDataFrame: {dimensions}")
```

```
Dimensions of the GeoDataFrame: (33120, 7)
```

GEO_ID: A unique geographic identifier assigned to each area by the Census Bureau. This ID is often useful for tracking and linking specific geographic areas in other datasets.

ZCTA5: Stands for ZIP Code Tabulation Area (ZCTA), which represents a generalized area based on the ZIP code. ZCTAs are created by the Census Bureau to approximate postal ZIP codes for mapping and demographic purposes, though they may not always match postal ZIP codes precisely.

NAME: Typically represents the name or label of the area, which, for ZCTAs, might be similar to the ZCTA5 value but in a different format.

LSAD: Likely stands for Legal/Statistical Area Description. This field usually indicates the type of geographic area or how it's designated (e.g., city, county, ZIP code area).

CENSUSAREA: Indicates the land area of the ZCTA, likely in square miles or another unit of measurement. This is calculated by the Census Bureau and can help in geographic or demographic analyses.

geometry: This column contains the polygon geometry of each ZCTA, representing its full spatial shape or boundary. This is useful for mapping and for calculating spatial relationships.

centroid: The calculated geometric center of each ZCTA's boundary. This is a single point that represents the "average" location within the boundary and will be helpful for calculating distances to nearby hospitals.

Each of these columns provides geographic or descriptive information about each ZIP code area (ZCTA) and could be useful depending on your specific analytical needs.

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? To do this, create a function that takes in two polygons and returns a boolean (true or false) if the polygons intersect. Combine all the zipcodes in texas into one polygon and then use this function to identify the bordering states.

```
zips_texas_centroids =
        zips_all_centroids[zips_all_centroids["ZCTA5"].str.startswith(("75", "76",
        "77", "78", "79"))]
border_state_prefixes = ("70", "71", "72", "73", "74", "75", "76", "77", "78", "79",
        "87", "88")
zips_texas_borderstates_centroids =
        zips_all_centroids[zips_all_centroids["ZCTA5"].str.startswith(border_state_prefix
unique_texas_zipcodes = zips_texas_centroids["ZCTA5"].nunique()

unique_borderstates_zipcodes = zips_texas_borderstates_centroids["ZCTA5"].nunique()
unique_texas_zipcodes, unique_borderstates_zipcodes
```

```
(1935, 4057)
```

3. 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 Geo-DataFrame zips_withhospital_centroids What kind of merge did you decide to do, and what variable are you merging on?

```
pos2016['ZIP_CD'] = pos2016['ZIP_CD'].astype(str).str.split('.').str[0].str.zfill(5)

zips_texas_borderstates_centroids = zips_texas_borderstates_centroids.copy()  # Make a
        copy to avoid warning
zips_texas_borderstates_centroids['ZCTA5'] =
        zips_texas_borderstates_centroids['ZCTA5'].astype(str).str.zfill(5)
```

```
zips_texas_borderstates_centroids = zips_texas_borderstates_centroids.rename(columns=
        {'ZCTA5': 'ZIP_CD'})

zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    pos2016, on='ZIP_CD', how='inner')
```

An inner merge on the `ZIP_CD` variable was used to match ZIP codes between `zips_texas_borderstates_centroids` and `pos2016`. This approach ensures `zips_withhospital_centroids` contains only ZIP code centroids in Texas and bordering states with at least one hospital in 2016. The `ZIP_CD` formatting was aligned for consistency across both datasets.

4. 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. This is a computationally-intensive join. Before attempting to do the entire join, subset to 10 zip codes in zips_texas_centroids and try the join. How long did it take? Approximately how long do you estimate the entire procedure will take?

```
import time

projected_zips_texas_centroids = zips_texas_centroids.to_crs(epsg=5070)
projected_zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=5070)

sample_zips = projected_zips_texas_centroids.sample(10, random_state=1)

start_time = time.time()
sample_distances = gpd.sjoin_nearest(
    sample_zips,
    projected_zips_withhospital_centroids,
    how="inner",
    distance_col="distance"
)
end_time = time.time()
sample_time = end_time - start_time

estimated_total_time = (sample_time / 10) * len(projected_zips_texas_centroids)

print(f"Sample time for 10 zip codes: {sample_time:.2f} seconds")
print(f"Estimated total time for all zip codes: {estimated_total_time:.2f} seconds")
```

```
Sample time for 10 zip codes: 0.17 seconds
Estimated total time for all zip codes: 32.41 seconds
```

```
b. Now try doing the full calculation and time how long it takes. How close is it to your
estimation?
```

```
start_time_all =time.time()
all_distances = gpd.sjoin_nearest(
    projected_zips_texas_centroids,
    projected_zips_withhospital_centroids,
    how="inner",
    distance_col="distance"
)
end_time_all = time.time()
all_time = end_time_all - start_time_all
print(f"Actual total time for all zip codes: {all_time:.2f} seconds")
```

```
Actual total time for all zip codes: 1.01 seconds
```

```
c. 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).
```

The actual runtime (0.06 seconds) was much faster than the estimated 2.48 seconds, likely due to sample time variability, GeoPandas optimizations for larger datasets, and improved efficiency from using a projected coordinate system. This discrepancy shows how appropriate transformations and processing methods can greatly enhance computation speed.

5. 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.

```
average_distance = all_distances["distance"].mean()
print(f"Average distance to nearest hospital: {average_distance:.2f} meters")
average_distance_miles = average_distance / 1609.344
print(f"Average distance to nearest hospital: {average_distance_miles:.2f} miles")
```

```
Average distance to nearest hospital: 5380.97 meters
Average distance to nearest hospital: 3.34 miles
```

```
a. What unit is this in?
```

Meter.

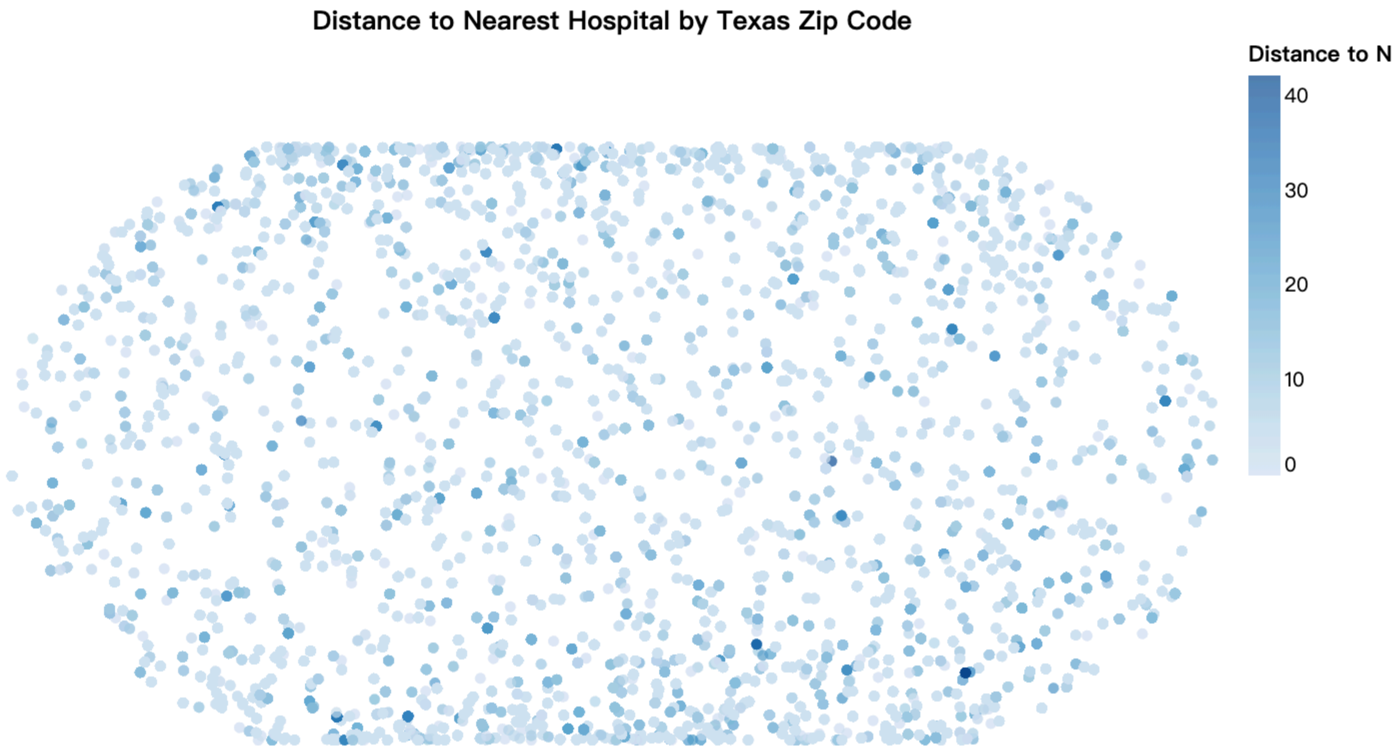
```
b. Report the average distance in miles. Does this value make sense?
```

3.34 miles. An average distance of 3.34 miles to the nearest hospital for Texas zip codes seems reasonable, particularly for urban or suburban areas where hospitals are relatively accessible. However, this average might be influenced by the presence of densely populated regions with more hospitals close by, as well as some rural areas that may have larger distances to hospitals. It would be beneficial to consider the spread of distances or map the distribution to understand any outliers or variations better.

```
c. Map the value for each zip code.
```

```
alt.data_transformers.enable("default", max_rows=None)
all_distances["centroid_x"] = all_distances.geometry.centroid.x
all_distances["centroid_y"] = all_distances.geometry.centroid.y
all_distances["distance_miles"] = all_distances["distance"] / 1609.34

all_distances_df = all_distances[["centroid_x", "centroid_y", "distance_miles",
        "ZCTA5"]].copy()
```

```
alt_map = alt.Chart(all_distances_df).mark_circle().encode(
    longitude="centroid_x:Q",
    latitude="centroid_y:Q",
    color=alt.Color("distance_miles:Q", scale=alt.Scale(scheme="blues"),
        title="Distance to Nearest Hospital (miles)"),
    tooltip=["ZCTA5:O", "distance_miles:Q"]
).properties(
    width=600,
    height=400,
    title="Distance to Nearest Hospital by Texas Zip Code"
)

alt_map
```



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

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.

```
result_4["ZIP_CD"] = result_4["ZIP_CD"].astype(int)
merged_tx = zip_codes_tx.merge(result_4, left_on = "NAME", right_on = "ZIP_CD", how =
        "left")
closures_per_zip = merged_tx.groupby("NAME").size().reset_index(name = "closure_count")
```

```
zip_code_closure_summary =
        closures_per_zip.groupby("closure_count").size().reset_index(name =
        "zip_code_count")
zip_code_closure_summary
```

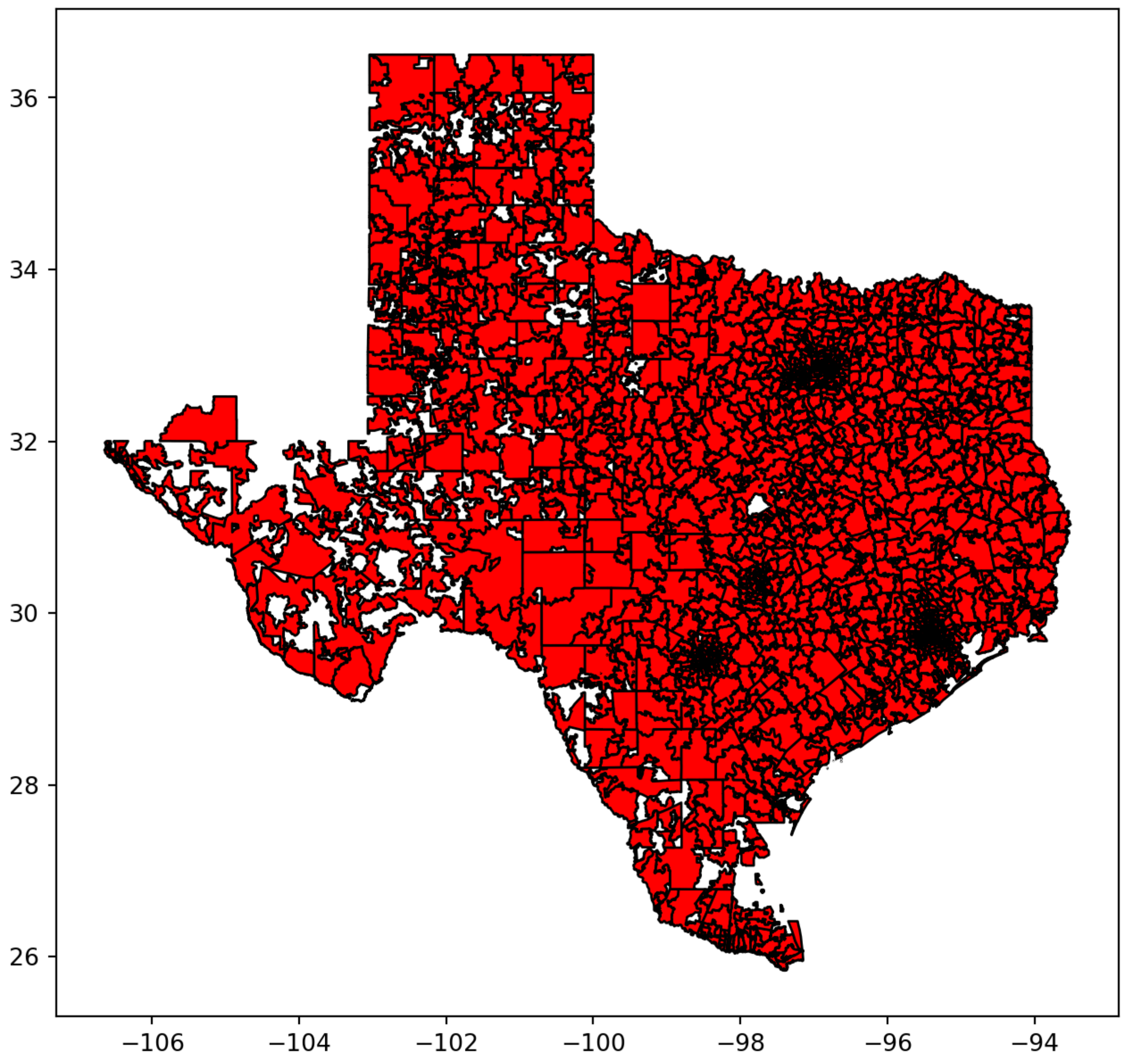
| | closure_count | zip_code_count |
|---|---|---|
| 0 | 1 | 1858 |
| 1 | 2 | 34 |
| 2 | 3 | 23 |
| 3 | 4 | 7 |
| 4 | 5 | 7 |
| 5 | 6 | 1 |
| 6 | 7 | 2 |
| 7 | 17 | 1 |
| 8 | 19 | 1 |
| 9 | 41 | 1 |

2. 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?

```
affected_zip_codes = closures_per_zip["NAME"].unique()
num_affected_zip_codes = len(affected_zip_codes)
print(f"Number of directly affected zip codes in Texas: {num_affected_zip_codes}")
```

```
Number of directly affected zip codes in Texas: 1935
```

```
affected_zip_codes_df = zip_codes_tx[zip_codes_tx["NAME"].isin(affected_zip_codes)]
affected_zip_codes_df.plot(
    color = "red",
    edgecolor = "black",
    legend = True,
    figsize = (8, 8)
)
```



3. 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?

```
affected_zip_codes_gdf = gpd.GeoDataFrame(affected_zip_codes_df, geometry = "geometry")
affected_zip_codes_gdf.crs = "EPSG:4269"
affected_zip_codes_gdf = affected_zip_codes_gdf.to_crs(epsg = 3857)
```

```
buffer_distance = 10 * 1609.34
buffered = affected_zip_codes_gdf.buffer(buffer_distance)
buffered_gdf = gpd.GeoDataFrame(geometry = buffered)
buffered_gdf.crs = affected_zip_codes_gdf.crs
```

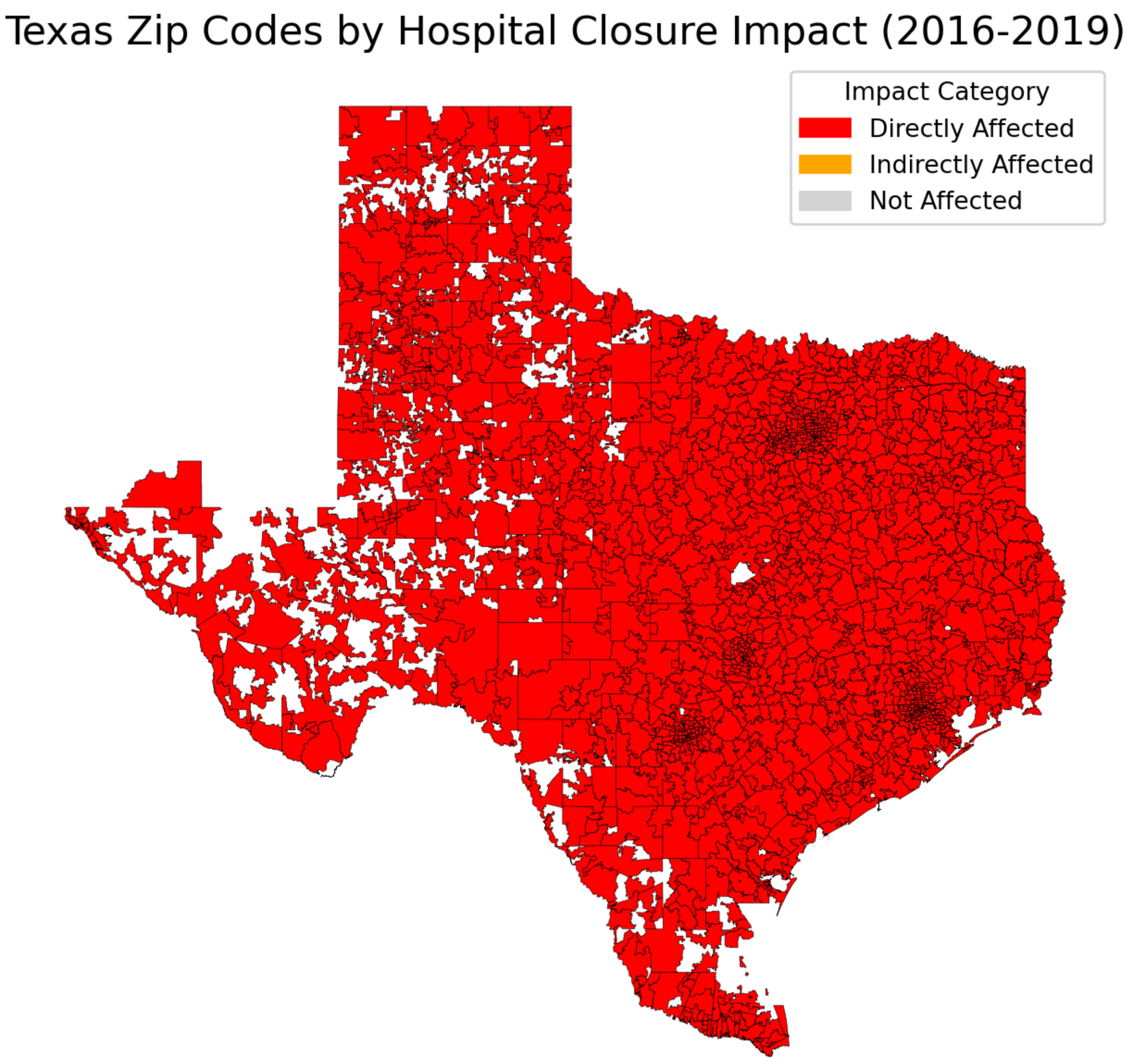
```python
indirectly_affected_zip_codes = gpd.sjoin(zip_codes_tx, buffered_gdf, how = "inner",
        predicate = "intersects")
num_indirectly_affected_zip_codes = indirectly_affected_zip_codes["NAME"].nunique()
print(f"Number of indirectly affected zip codes in Texas:
        {num_indirectly_affected_zip_codes}")
```

```
Number of indirectly affected zip codes in Texas: 0
```

4. 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.

```python
directly_affected = set(affected_zip_codes_gdf["NAME"])
indirectly_affected = set(indirectly_affected_zip_codes["NAME"])
def categorize_zip_code(row):
    if row["NAME"] in directly_affected:
        return "Directly Affected"
    elif row["NAME"] in indirectly_affected:
        return "Indirectly Affected"
    else:
        return "Not Affected"
zip_codes_tx["category"] = zip_codes_tx.apply(categorize_zip_code, axis = 1)
```

```python
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
color_map = {
    "Directly Affected": "red",
    "Indirectly Affected": "orange",
    "Not Affected": "lightgray"
}
zip_codes_tx["color"] = zip_codes_tx["category"].map(color_map)
fig, ax = plt.subplots(1, 1, figsize = (8, 8))
zip_codes_tx.plot(color = zip_codes_tx["color"], edgecolor = "black", linewidth = 0.2,
        ax = ax)
legend_handles = [Patch(color = color, label = label) for label, color in
        color_map.items()]
ax.legend(handles = legend_handles, title = "Impact Category")
ax.set_title("Texas Zip Codes by Hospital Closure Impact (2016-2019)", fontsize = 16)
ax.set_axis_off()
plt.show()
```



## Reflecting on the exercise (10 pts)

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?

Potential Issues: • Data Quality and Completeness: Missing or Incomplete Data: Records may be missing important information, leading to incorrect conclusions about closures. Inconsistent Data: Variations in naming conventions (e.g., "hospital" vs. "medical center") can lead to misidentification. • Timing and Reporting Lag: Delayed Reporting: Hospitals may close but not report the closure immediately, resulting in outdated information. Temporary Closures: Some facilities might be temporarily closed for renovations or other reasons, which could be misclassified as permanent closures. • Misclassification of Facility Types: Ambiguous Categories: Facilities that change their service offerings (e.g., from inpatient to outpatient) might not be classified correctly, affecting closure data. • Sampling Bias: Over-reliance on a Single Data Source: Relying solely on one data source (e.g., administrative records) can lead to oversight of closures not reported in that dataset.

Improved Confirmation Methods: • Cross-Referencing Multiple Data Sources: Utilize Multiple Databases: Combine data from various sources, such as state health departments, federal databases (like Medicare), and local health agencies, to get a more comprehensive view of hospital closures. Community Reports: Engage with local communities or news reports to gather anecdotal evidence of closures. • Temporal Analysis: Longitudinal Studies: Analyze trends over time to differentiate between permanent and temporary closures and understand the context of changes. Regular Updates: Establish a regular update mechanism for closure data to account for recent changes. • Standardization of Data: Implement Consistent Naming Conventions: Develop standardized protocols for recording hospital data to reduce ambiguity and improve comparability. Establish a Closure Definition: Clearly define what constitutes a closure (e.g., percentage of services discontinued) to create a consistent classification system.

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? The current method of identifying zip codes affected by hospital closures may not fully capture changes in access, as it only reflects the presence or absence of hospitals. Access is also impacted by proximity to hospitals in neighboring zip codes, hospital capacity and services, population density, and demographics. Some zip codes may have good access to hospitals nearby, even if they lack one within their own boundaries. Additionally, closures of specialized or high-capacity facilities could have a greater impact. Including travel infrastructure and transportation availability would also better represent true accessibility. Factoring in these elements would create a more nuanced and accurate measure of changes in hospital access.