Problem Set 4

AUTHOR
Cristian Bancayan & Sol Rivas Lopes

PUBLISHED
November 2, 2024

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

> "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **CB SCRL**

> "I have uploaded the names of anyone else other than my partner and I worked with on the problem set here" (1 point)

> Late coins used this pset: Each of us used **1** Late coins left after submission: Each of us still have **3**

Style Points (10 pts)

Submission Steps (10 pts)

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

Question 1

```
import warnings
import pandas as pd
import altair as alt
import datetime as dt
import numpy as np
import chardet
import os
import geopandas as gpd
import matplotlib.pyplot as plt
import time

alt.renderers.enable("png")

warnings.filterwarnings('ignore')

# Working directory
# Cris
base_path = r"C:\Users\Cristian\Documents\GitHub\ppha30538_fall2024"
# Sol
#base_path = r"C:\Users\solch\OneDrive\Documentos\2024 - autumn quarter\python II"
```

After reading the tasks for this problem set, I pulled the PRVDR_CTGRY_CD, PRVDR_CTGRY_SBTYP_CD, PRVDR_NUM, PGM_TRMNTN_CD, FAC_NAME, ZIP_CD variables

Question 2

Part (a)

```
# reading in file for 2016
path = os.path.join(
    base_path, "problem-set-4-sol-cristian\data\pos2016.csv")

# Sol
# df_2016 = pd.read_csv(path)

# Cris needs to read in his files this way otherwise they won't open
# Sol can't open them unless it's with the syntax above
df_2016 = pd.read_csv(path, sep=';')

# Subsetting to include only short-term hospitals
df_2016 = df_2016[(df_2016["PRVDR_CTGRY_CD"] == 1) & (df_2016["PRVDR_CTGRY_SBTYP_CD"] == 1)]

# Counting rows to get number of hospitals reported in data
row_count_2016 = df_2016.shape[0]

print(f"The number of hospitals reported in the data is {row_count_2016}")
```

```
The number of hospitals reported in the data is 7245
```

Part (b)

> According to the Kaiser Family Foundation, “there are nearly 5,000 short-term, acute care hospitals in the United States.” This number is considerably lower than the one we find in our data. There are two reasons that can explain this discrepancy. First, we know this datset will include hospitals that are no longer active, which would increase the number of facilities it is reporting. Second, this number references “acute care hospitals.” These facilities [“primarily provides short-term medical treatment for patients with severe or urgent health issues. These institutions — common in small towns and rural areas — deliver comprehensive and specialized medical care for conditions that require immediate attention.”] {https://www.kff.org/report-section/a-look-at-rural-hospital-closures-and-implications-for-access-to-care-three-case-studies-issue-brief/}. Although our data is subsetted for short-term facilities, it can be considering other categories of hospitals, such as psychiatric, research, or specialty hospitals.

Question 3

```
###   2017 DATASET    ###

# reading in file for 2017
path = os.path.join(
    base_path, "problem-set-4-sol-cristian\data\pos2017.csv")
```

```
# Sol
#df_2017 = pd.read_csv(path)
# Cris
df_2017 = pd.read_csv(path, sep=';')

# Subsetting to include only short-term hospitals
df_2017 = df_2017[(df_2017["PRVDR_CTGRY_CD"] == 1) &
                  (df_2017["PRVDR_CTGRY_SBTYP_CD"] == 1)]

# Counting rows to get number of hospitals reported in data
row_count_2017 = df_2017.shape[0]

print(f"The number of hospitals reported in the 2017 data is {row_count_2017}")
```

The number of hospitals reported in the 2017 data is 7260

```
###   2018 DATASET    ###
path = os.path.join(
    base_path, "problem-set-4-sol-cristian\data\pos2018.csv")

# Sol: When trying to read in file, it seems that the encoding is different than what python a
# It corrected my code to include the encoding:
#df_2018 = pd.read_csv(path, encoding="ISO-8859-1")
# Cris worked fine
df_2018 = pd.read_csv(path, sep=';')

# Subsetting to include only short-term hospitals
df_2018 = df_2018[(df_2018["PRVDR_CTGRY_CD"] == 1) &
                  (df_2018["PRVDR_CTGRY_SBTYP_CD"] == 1)]

row_count_2018 = df_2018.shape[0]

print(f"The number of hospitals reported in the 2018 data is {row_count_2018}")
```

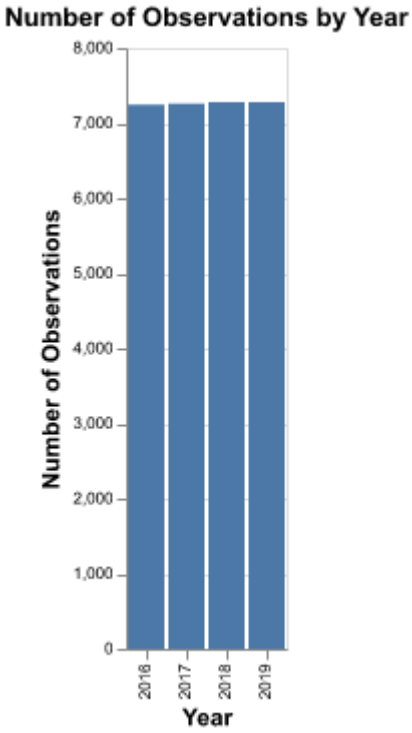
The number of hospitals reported in the 2018 data is 7277

```
###   2019 DATASET    ###

path = os.path.join(
    base_path, "problem-set-4-sol-cristian\data\pos2019.csv")

# Sol
#df_2019 = pd.read_csv(path, encoding="ISO-8859-1")
# Cris
df_2019 = pd.read_csv(path, sep=';')

# Subsetting to include only short-term hospitals
df_2019 = df_2019[(df_2019["PRVDR_CTGRY_CD"] == 1) &
                  (df_2019["PRVDR_CTGRY_SBTYP_CD"] == 1)]

row_count_2019 = df_2019.shape[0]
```

```
print(f"The number of hospitals reported in the 2019 data is {row_count_2019}")
```

```
The number of hospitals reported in the 2019 data is 7282
```

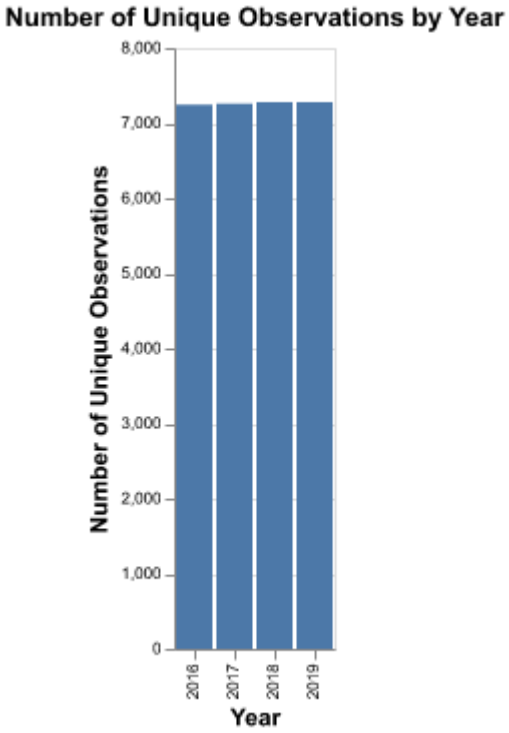
```
###     APPENDING     ###

# Adding a year column in each dataset to indentify it later on
df_2016["YEAR"] = 2016
df_2017["YEAR"] = 2017
df_2018["YEAR"] = 2018
df_2019["YEAR"] = 2019

# Appending datsets together
df_final = pd.concat([df_2016, df_2017, df_2018, df_2019],
                     ignore_index=True)

###     GRAPHING      ###

# Creating a small df with the number of observations each year
df_obs = {
    "Year": [2016, 2017, 2018, 2019],
    "Count": [row_count_2016, row_count_2017, row_count_2018, row_count_2019]
}
df_obs = pd.DataFrame(df_obs)


# Plotting
alt.Chart(df_obs, title="Number of Observations by Year").mark_bar().encode(
    alt.X("Year:N", title="Year"),
    alt.Y("Count:Q", title="Number of Observations")
).configure_axis(
    labelFontSize=8,
    titleFontSize=12
)
```



Question 4

Part (a)

```
# Getting unique provider numbers by year
df_unique = df_final.groupby("YEAR")["PRVDR_NUM"].nunique().reset_index()

# Plotting
alt.Chart(df_unique, title="Number of Unique Observations by Year").mark_bar().encode(
    alt.X("YEAR:N", title="Year"),
    alt.Y("PRVDR_NUM:Q", title="Number of Unique Observations")
).configure_axis(
    labelFontSize=8,
    titleFontSize=12
)
```



Part (b)

> The data features one unique hospital per year.

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

Question 1

```
# Creating a data set with active hospitals in 2016
df_active2016 = df_2016[df_2016["PGM_TRMNTN_CD"] == 0]

# Creating a data set with observations from other years...
df_closed_by2019 = df_final[df_final["YEAR"]!=2016]
# ... and subsetting to only include non-active hospitals
df_closed_by2019 = df_closed_by2019[df_closed_by2019["PGM_TRMNTN_CD"]!=0]

# Merging the two datasets
df_closed_by2019 = df_active2016.merge(df_closed_by2019[["PRVDR_NUM","PGM_TRMNTN_CD", "YEAR"]]

# Getting hospitals that were open in 2016 but closed in subsequent years
df_closed_by2019 = df_closed_by2019[df_closed_by2019["_merge"] == "both"]

# Sorting by provider number and then year to get all observations for same hospital together,
df_closed_by2019.sort_values(by=["PRVDR_NUM", "YEAR_y"], ascending=[False, True], inplace=True

# Keep the year in which the hospital has the status of not active for the first time.
df_closed_by2019 = df_closed_by2019.groupby("PRVDR_NUM").aggregate(
    FAC_NAME = ("FAC_NAME", "first"),
    ZIP_CD   = ("ZIP_CD", "first"),
    YEAR     = ("YEAR_y", "first")
```

```
).reset_index()

row_closed_by2019 = df_closed_by2019.shape[0]

print(f"Number of hospitals that were active in 2016 and were suspected to have closed by 2019
```

```
Number of hospitals that were active in 2016 and were suspected to have closed by 2019: 120
```

```
# Sorting by name
df_closed_by2019.sort_values(by=['FAC_NAME'], inplace=True)
df_closed_by2019.head(10)
df_closed_by2019.shape[0]
```

```
120
```

Question 3

```
# Creating a new dataset to work out of
df_active = df_final

# Creating a dummy column identifying active hospitals
df_active['ACTIVES'] = df_active['PGM_TRMNTN_CD'].apply(
    lambda x: 1 if x == 0 else 0)

# Creating a dummy column identifying non-active hospitals
df_active['NO_ACTIVE'] = df_active['PGM_TRMNTN_CD'].apply(
    lambda x: 1 if x != 0 else 0)

# Number of active, non active and total hospitals by zip and year
df_nactive_zip = df_active.groupby(["ZIP_CD", "YEAR"]).aggregate(
    ACTIVE=("ACTIVES", "sum"),
    NO_ACTIVE=("NO_ACTIVE", "sum"),
    TOTAL=("FAC_NAME", "count")
).reset_index()

# Merge to identify in which ZIP CODE ocurred hospital closures
df_zip_fail = df_closed_by2019.merge(
    df_nactive_zip, on=["ZIP_CD"], how="left", indicator=True)

# Keeping the year of change
df_zip_fail = df_zip_fail[(df_zip_fail['YEAR_x'] == df_zip_fail['YEAR_y']) | (
    df_zip_fail['YEAR_x'] - 1 == df_zip_fail['YEAR_y'])]

# This refers to year x
df_zip_fail['CHANGE_YEAR'] = df_active['PGM_TRMNTN_CD'].apply(
    lambda x: 1 if x != 0 else 0)

df_zip_fail['STATUS'] = 'Other'
df_zip_fail.loc[df_zip_fail['YEAR_x'] - 1 ==
                df_zip_fail['YEAR_y'], 'STATUS'] = 'BEFORE_CLOSURE'
df_zip_fail.loc[df_zip_fail['YEAR_x'] ==
```

```
                df_zip_fail['YEAR_y'], 'STATUS'] = 'CLOSED'

df_zip_fail = df_zip_fail[["ZIP_CD", "STATUS", "ACTIVE"]]

df_zip_fail = df_zip_fail.pivot(
    index="ZIP_CD", columns="STATUS", values="ACTIVE").reset_index()

df_zip_fail["INCREASE"] = df_zip_fail["BEFORE_CLOSURE"] - df_zip_fail["CLOSED"]

# Zip codes where the number of hospital do not decrease in the year of clousure
df_zip_fail = df_zip_fail[df_zip_fail["INCREASE"] <= 0]
```

Part (a)

```
df_closed_by2019 = df_closed_by2019.merge(df_zip_fail[["ZIP_CD","INCREASE"]], on = "ZIP_CD", h

df_closed_by2019["INCREASE"].unique()

df_closed_by2019["POT_MERGE"] = df_closed_by2019["INCREASE"].apply(
    lambda x: 1 if x == 0 else 0)

pot_merge = df_closed_by2019[df_closed_by2019["POT_MERGE"]==1]["POT_MERGE"].value_counts().ilo

print(f"Number of hospitals that fit in the definition of potentially being a merger/acquisiti

df_closed_by2019 = df_closed_by2019[df_closed_by2019["POT_MERGE"] != 1]
nhospital_corrected = df_closed_by2019.shape[0]

print(f"Number of hospitals left after correction for potential merge: {nhospital_corrected}")
```

```
Number of hospitals that fit in the definition of potentially being a merger/acquisition: 3
Number of hospitals left after correction for potential merge: 117
```

Part (b)

```
# Sorting by name
df_closed_by2019.sort_values(by=['FAC_NAME'], inplace=True)
df_closed_by2019.head(10)
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	YEAR	INCREASE	POT_MERGE
0	30001	ABRAZO MARYVALE CAMPUS	85031.0	2017.0	NaN	0
1	50196	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230.0	2017.0	NaN	0
2	360151	AFFINITY MEDICAL CENTER	44646.0	2018.0	NaN	0
3	330189	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	12208.0	2017.0	NaN	0
4	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2017.0	NaN	0
5	250159	ALLIANCE LAIRD HOSPITAL	39365.0	2019.0	NaN	0
6	370032	ALLIANCEHEALTH DEACONESS	73112.0	2019.0	NaN	0

	PRVDR_NUM	FAC_NAME	ZIP_CD	YEAR	INCREASE	POT_MERGE
7	60036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050.0	2017.0	NaN	0
8	290006	BANNER CHURCHILL COMMUNITY HOSPITAL	89406.0	2017.0	NaN	0
9	30033	BANNER PAYSON MEDICAL CENTER	85541.0	2018.0	NaN	0

Download Census zip code shapefile (10 pt)

Question 1

Part (a)

File Extension	Description
.dbf	Stores attribute data in a table format, similar to a database or spreadsheet.
.prj	Contains projection and coordinate system information for accurate representation of the Earth's sphere into a plane.
.shp	Holds spatial data like points, lines, and polygons.
.shx	An index file linking the .shp spatial data to the attribute data in .dbf
.xml	Stores metadata

All together, these files create the "shapefile," storing both attribute and spatial data.

Part (b)

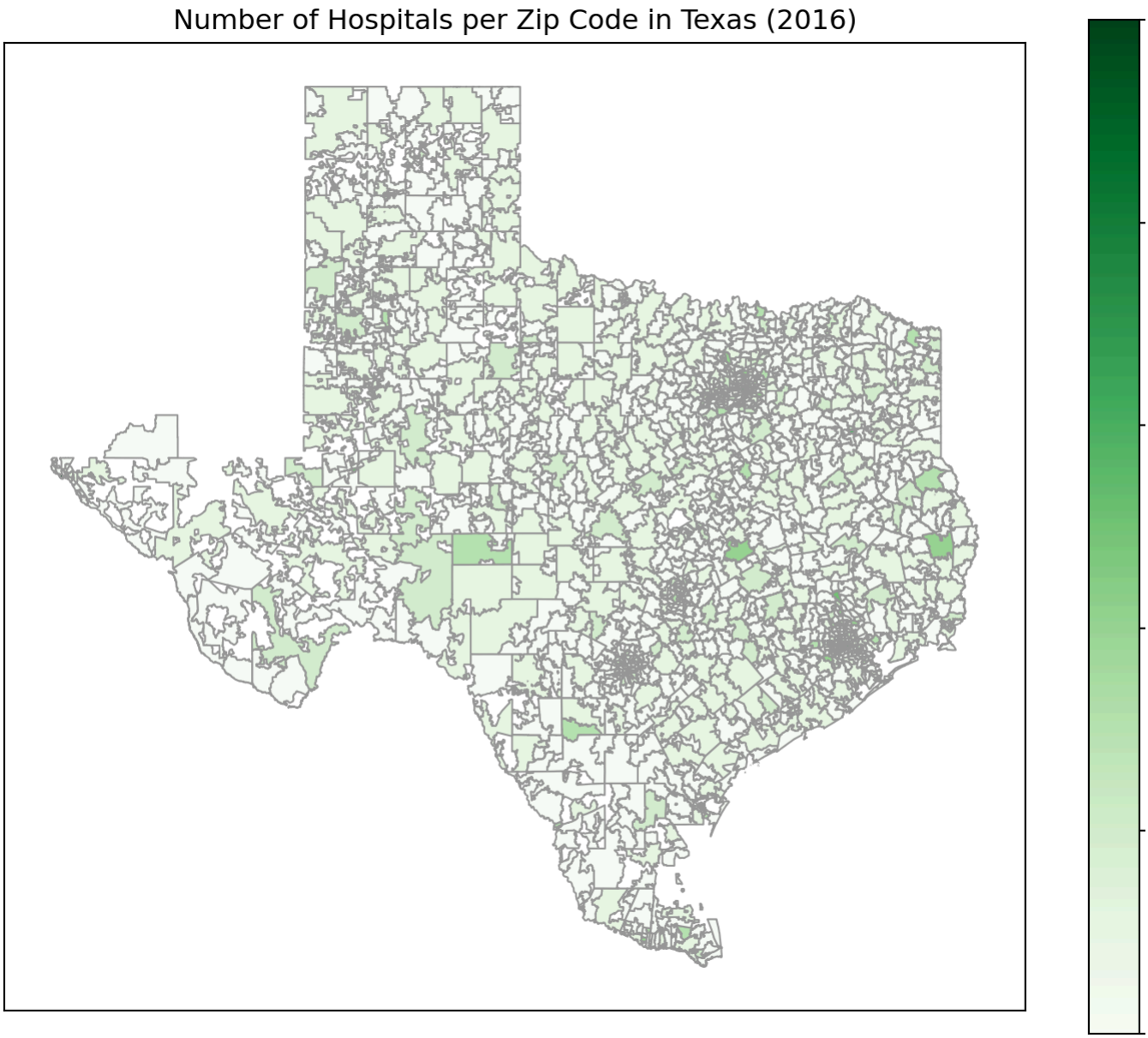
```
# Sol
# data_path = r"C:\Users\solch\OneDrive\Documentos\2024 - autumn quarter\python II\problem-set
# Cris
data_path = r"C:\Users\Cristian\Documents\GitHub\ppha30538_fall2024\problem-set-4-sol-cristian

# List of file paths
file_paths = ["gz_2010_us_860_00_500k.dbf ", "gz_2010_us_860_00_500k.prj ",
              "gz_2010_us_860_00_500k.shp", "gz_2010_us_860_00_500k.shx",
              "gz_2010_us_860_00_500k.xml"]

# Function to get file sizes
def get_file_sizes(paths):
    for file in paths:
        path = path = os.path.join(
            data_path, file)
        size = os.path.getsize(path)  # Size in bytes
        print(f"{os.path.basename(file)}: {size / 1024:.2f} KB")  # Size in KB
        # Got help from AI for conversion

# Measure and print file sizes
get_file_sizes(file_paths)
```

```
gz_2010_us_860_00_500k.dbf : 6274.88 KB
gz_2010_us_860_00_500k.prj : 0.16 KB
gz_2010_us_860_00_500k.shp: 817914.63 KB
gz_2010_us_860_00_500k.shx: 258.85 KB
gz_2010_us_860_00_500k.xml: 15.27 KB
```

Question 2

```
# Reading in shapefile
path = path = os.path.join(data_path, "gz_2010_us_860_00_500k.shp")
shp_file = gpd.read_file(path)

# Getting column names to identify where I can find the ZIP
shp_file.columns

# Subsetting Texas
texas_shp = shp_file[shp_file["ZCTA5"].str.startswith(
    ("75", "76", "77", "78", "79"))]

# Creating new df to work on
df_2016_tx = df_2016
# Turning zip column into a string for better data handling
df_2016_tx["ZIP_CD"] = df_2016_tx["ZIP_CD"] .astype(str)
# Subsetting to only TX
df_2016_tx = df_2016_tx[df_2016_tx["ZIP_CD"].str.startswith(
    ("75", "76", "77", "78", "79")) & (df_2016_tx["ZIP_CD"].str.len() == 7)]
# Cleaning strings
df_2016_tx["ZIP_CD"] = df_2016_tx["ZIP_CD"].str.replace(
    '.0', '', regex=False)
# Counting hospitals by ZIP
df_2016_tx = df_2016_tx.groupby("ZIP_CD").size().reset_index()
# Renaming columns to merge properly
df_2016_tx.columns = ["ZCTA5", "COUNT"]

gdf_2016_tx = texas_shp.merge(df_2016_tx, on="ZCTA5", how="left")
# Making NAs = 0
gdf_2016_tx["COUNT"] = gdf_2016_tx["COUNT"].fillna(0)
```

```
#  Plotting  #
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
gdf_2016_tx.plot(column="COUNT", cmap="Greens",
                    linewidth=0.8, ax=ax, edgecolor="0.6", legend=True)
ax.set_title("Number of Hospitals per Zip Code in Texas (2016)")
# Remove axis and ticks
ax.set_xticks([])
ax.set_yticks([])
plt.show()
```



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

Question 1

```
zips_all_centroids = shp_file.copy()
zips_all_centroids['centroid'] = shp_file.geometry.centroid

# Dimensions of the new GeoDataFrame:
zips_all_centroids.shape
zips_all_centroids.columns
```

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

Meaning of each column: + GEO_ID: Identifier for geographic area. + ZCTA5: ZIP Code Tabulation Area. + NAME: Description/Labe of an area. + LSAD: Legal/Statistical Area Description; typically used to describe the type of area. + CENSUSAREA: Area of the ZIP code region. + geometry: represent a polygon geometry of each ZIP code area. + centroid: centroid of each ZIP code area as a point geometry.

Question 2

```
# Zip codes in Texas
zips_texas_centroids = zips_all_centroids[zips_all_centroids["ZCTA5"].str.startswith(
    ("75", "76", "77", "78", "79"))]

# For subsetting all zip codes in Texas or a bordering state, we will use the following prefix
# Texas TX: 75, 76, 77, 78, 79 | New Mexico NM: 87, 88 | Oklahoma OK: 73, 74 | Arkansas : 716,

zips_texas_borderstates_centroids  = zips_all_centroids[zips_all_centroids["ZCTA5"].str.starts
    ("75", "76", "77", "78", "79",
     "87", "88",
     "73","74",
     "70", "71", "72"))]

nzip_tx = zips_texas_centroids["ZCTA5"].nunique()
nzip_tx_borders = zips_texas_borderstates_centroids["ZCTA5"].nunique()

print(f"Number of unique zip codes in zips_texas_centroids: {nzip_tx}")
print(f"Number of unique zip codes in nzip_tx_borders: {nzip_tx_borders}")
```

```
Number of unique zip codes in zips_texas_centroids: 1935
Number of unique zip codes in nzip_tx_borders: 4057
```

Question 3

```
df_2016_tx_border = df_2016[df_2016["PGM_TRMNTN_CD"]==0]
# Turning zip column into a string for better data handling
df_2016_tx_border["ZIP_CD"] = df_2016_tx_border["ZIP_CD"].astype(str)

df_2016_tx_border  = df_2016_tx_border[df_2016_tx_border["ZIP_CD"].str.startswith(
    ("75", "76", "77", "78", "79",
     "87", "88",
     "73","74",
     "70", "71", "72"))]

df_2016_tx_border["ZIP_CD"] = df_2016_tx_border["ZIP_CD"].str.replace(
    '.0', '', regex=False)

# Hospitals per ZIP
df_2016_tx_border = df_2016_tx_border.groupby("ZIP_CD").size().reset_index()
# Renaming columns to merge properly
df_2016_tx_border.columns = ["ZCTA5", "COUNT"]
# ZIPS with at least one hospital
df_2016_tx_border = df_2016_tx_border[df_2016_tx_border["COUNT"]>0]
```

```
# Merging
zips_withhospital_centroids= zips_texas_borderstates_centroids.merge(df_2016_tx_border, on="ZC
zips_withhospital_centroids = zips_withhospital_centroids[zips_withhospital_centroids["COUNT"]
```

I am merging the previously created GeoDataFrame, zips_texas_borderstates_centroids, which contains all ZIP codes in Texas and bordering states, with the dataset which includes the number of hospitals per ZIP code. Since the later dataset contains the ZIP codes of interest (with at leat 1 hospital in 2016), we are using a right merge to retain these ZIP codes for analysis. The merge is performed on the ZCTA5 variable, and we are preserving all geographical variables: GEO_ID, LSAD, CENSUSAREA, geometry, and centroid. Additionally, we are adding the number of hospitals in each ZIP code area.

Question 4

Part (a)

```
# Subseting with 10 zip codes
zips_texas_with_distance_10 = zips_texas_centroids[:10][["GEO_ID", "ZCTA5", "geometry", "centr

zips_texas_with_distance_10
# To estimate the time for this exercise
start_time = time.time()
zips_texas_with_distance_10 = gpd.sjoin_nearest(
    zips_texas_with_distance_10,
    zips_withhospital_centroids,
    how = "left",
    distance_col = "distance"
)
end_time = time.time()
time_spent = end_time - start_time

print(f"Time of this excercise: {time_spent}s")
estimation = zips_texas_centroids.shape[0]/10*time_spent
print(f"Estimated time for the whole GeoDataFrame: {estimation}s")
```

```
Time of this excercise: 2.2256484031677246s
Estimated time for the whole GeoDataFrame: 430.6629660129547s
```

Part (b)

```
zips_texas_with_distance = zips_texas_centroids[["GEO_ID", "ZCTA5", "geometry", "centroid"]]

# To estimate the time for this exercise
start_time = time.time()
zips_texas_with_distance = gpd.sjoin_nearest(
    zips_texas_with_distance,
    zips_withhospital_centroids,
    how = "inner",
    distance_col = "distance"
)
```

```
end_time = time.time()
time_spent = end_time - start_time
time_spent

print(f"It takes: {time_spent} seconds")
```

```
It takes: 492.65690994262695 seconds
```

My estimation was lower than the actual value.

Part (c)

```
path = path = os.path.join(data_path, "gz_2010_us_860_00_500k.prj")

with open(path, "r") as prj_file:
    prj_content = prj_file.read()
    print("Projection content:", prj_content)
```

```
Projection content:
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298
.257222101]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]]
```

The unit shown in the .prj file is "Degree." One degree is approximately equivalent to 69 miles.

Question 5

```
# Average distance to the nearest hospital for each zip code
zips_texas_avg_distance = zips_texas_with_distance.groupby("ZCTA5_left").aggregate(
    avg_distance = ("distance", "mean"),
    geometry = ("geometry", "first"),
    centroid = ("centroid_left", "first")
).reset_index()

# Average distance in miles per zip code
zips_texas_avg_distance["avg_distance"] = zips_texas_avg_distance["avg_distance"]*69
```

Part (a)

```
zips_texas_avg_distance_gdf = gpd.GeoDataFrame(
    zips_texas_avg_distance,
    geometry="geometry",
    crs="EPSG:4269"
)

print(zips_texas_avg_distance_gdf.crs)
```

```
EPSG:4269
```

The units are in degree. However en the distances were multiplied by 69 to convert them to miles.

Part (b)

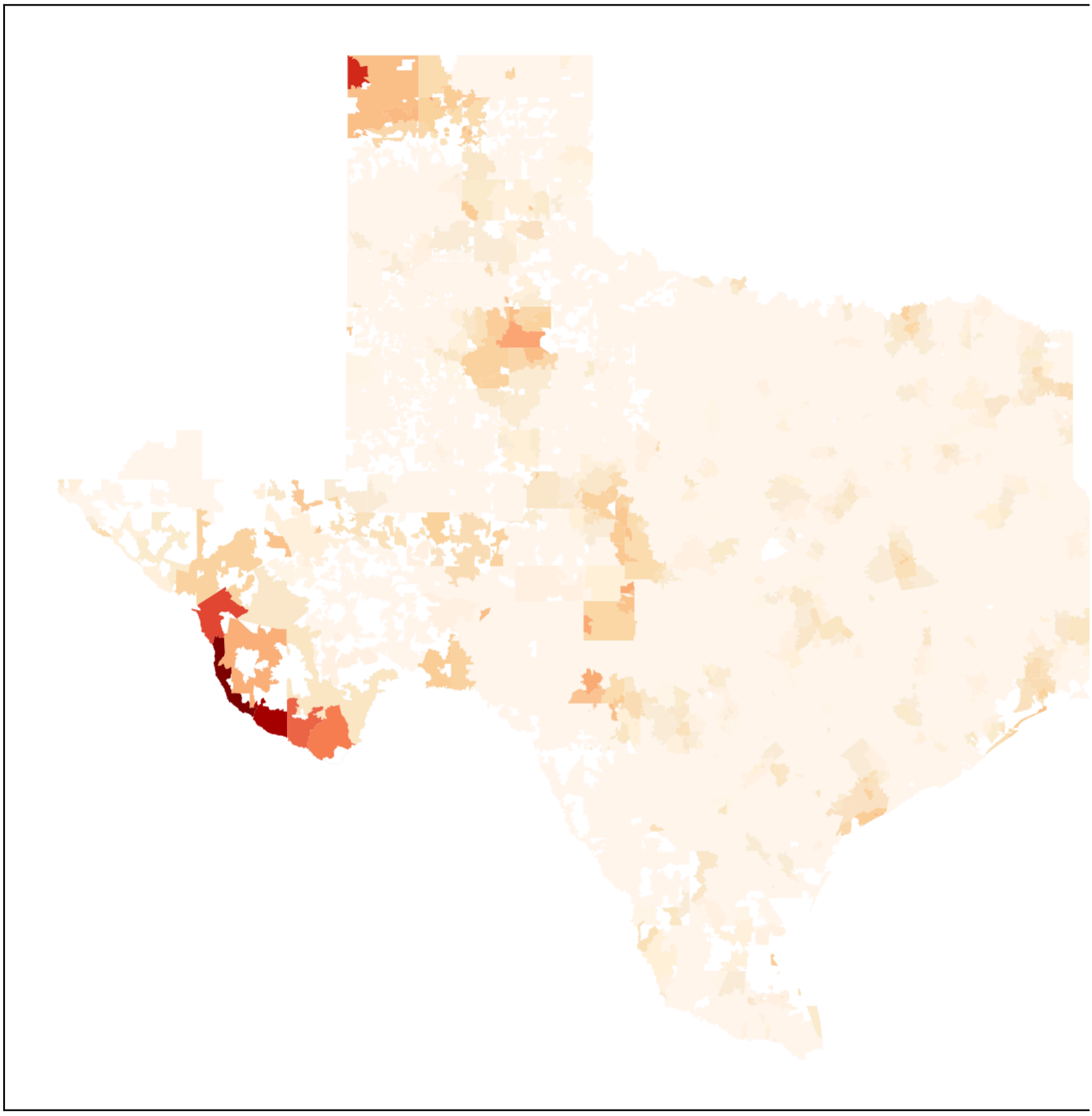
```
# Average distance in miles
zips_texas_avg_distance_gdf["avg_distance"].mean()
```

```
2.754165545455319
```

The average distance is 2.75 miles. In the dataset we can see that there are many zeros, which make the average to be lower. A zero distance indicate that there is a hospital in the zip code. The value then make sense.

Part (c)

```
fig, ax = plt.subplots(1, 1, figsize=(12, 12))
zips_texas_avg_distance_gdf.plot(
    column="avg_distance",
    cmap="OrRd",
    legend=True,
    legend_kwds={
        'label': "Average Distance to Nearest Hospital (miles)",
        'orientation': "vertical"
    },
    ax=ax
)

ax.set_xticks([])
ax.set_yticks([])

plt.show()
```



From the graph, we can see that some areas in the borders have the greater distance to hospitals.

Effects of closures on access in Texas (15 pts)

Question 1

```
# Turning ZIP_CD into a string
df_closed_by2019["ZIP_CD"] = df_closed_by2019["ZIP_CD"].astype(str)
# Selecting TX zipcodes
df_closed_tx = df_closed_by2019[df_closed_by2019["ZIP_CD"].str.startswith(
    ("75", "76", "77", "78", "79")) & (df_closed_by2019["ZIP_CD"].str.len() == 7)]
# Cleaning for aesthetic purposes
df_closed_tx["ZIP_CD"] = df_closed_tx["ZIP_CD"].str.replace(r'\.0$', '', regex=True)

# Counting closures by
df_closed_zip_tx = df_closed_tx.groupby("ZIP_CD").size().reset_index()
df_closed_zip_tx.columns = ["ZIP", "hospital_closures"]

print(df_closed_zip_tx)
```

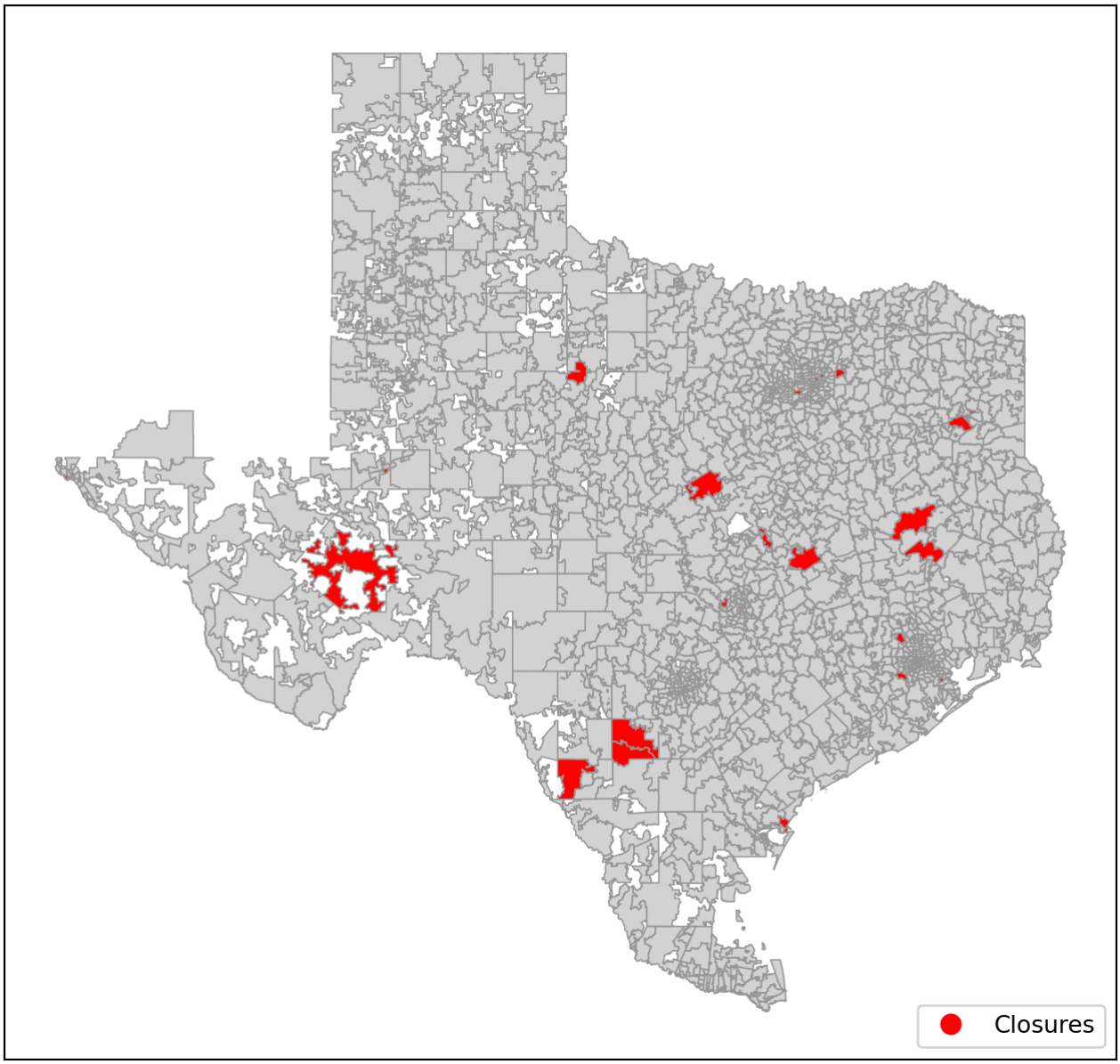
```
      ZIP  hospital_closures
0   75042                  1
1   75051                  1
2   75087                  1
3   75231                  1
4   75601                  1
5   75662                  1
6   75835                  1
7   75862                  1
8   76502                  1
9   76520                  1
10  76531                  1
11  77035                  1
12  77054                  1
13  77429                  1
14  77479                  1
15  77598                  1
16  78017                  1
17  78061                  1
18  78336                  1
19  78734                  1
20  78834                  1
21  79553                  1
22  79735                  1
23  79761                  1
24  79902                  1
```

Question 2

```
df_closed_zip_tx.columns = ["ZCTA5", "hospital_closures"]
gdf_tx = gdf_2016_tx.merge(df_closed_zip_tx, on="ZCTA5", how= "left" )
# Replace NA values with 0
gdf_tx["hospital_closures"] = gdf_tx["hospital_closures"].fillna(0)
```

```
# Setting red color for zips with closures
# Chat GPT helped with this. Query: Since my variable is binary, can my scale be binary as wel
```

```
from matplotlib.colors import ListedColormap
from matplotlib.lines import Line2D

binary_cmap = ListedColormap(["lightgray", "red"])

# Making chart
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
gdf_tx.plot(column= "hospital_closures", cmap= binary_cmap,
           linewidth=0.6, ax=ax, edgecolor='0.6')
# I also asked AI to help me make this plot prettier. The following lines were coded # with so

# Title
ax.set_title("Texas ZIP Codes with Hospital Closures (2017-2019)", fontsize=12, weight='bold',

# Legend
closure_marker = [Line2D([0], [0], marker='o', color='w', label='Closures',
                         markersize=10, markerfacecolor='red')]
ax.legend(handles=closure_marker, loc='lower right', fontsize=10)

# Remove axis and ticks
ax.set_xticks([])
ax.set_yticks([])

plt.show()
```



Question 3

```
# Filter geodf to only include those with hospital closures
gdf_tx_closures = gdf_tx[gdf_tx["hospital_closures"] > 0]

# Creating a 10 mi buffer (Projection is in meters, mile is 1609.34m, so 10 miles =  1609.34 *
gdf_tx_closures["buffer"] = gdf_tx_closures.geometry.buffer(16093.4)

# Renaming column to idenitfy ZCTA of indirectly affected better after merging
gdf_tx_closures = gdf_tx_closures.rename(columns={"ZCTA5": "ZCTA5_buffer"})

# Spatial join to merge geodfs together
# AI inadvertedly helped on this while I did a Google search.
# Google search: spatial join on python, gemini showed me some code
gdf_tx_indirect_closures = gpd.sjoin(
    gdf_tx, gdf_tx_closures, how="inner", predicate="intersects")

# Subsetting buffer Zips so they don't get counted twice
```

```
buffer_zips = gdf_tx_indirect_closures["ZCTA5_buffer"].unique()

# Get indirectly affected zips
indirectly_affected_zips = gdf_tx_indirect_closures["ZCTA5"].unique()
indirectly_affected_zips = indirectly_affected_zips[~np.isin(
    indirectly_affected_zips, buffer_zips)]

# Counting indirectly affected ZIP codes
indirectly_affected_zip_count = len(indirectly_affected_zips)

print("Number of indirectly affected ZIP codes:", indirectly_affected_zip_count)
```

```
Number of indirectly affected ZIP codes: 136
```

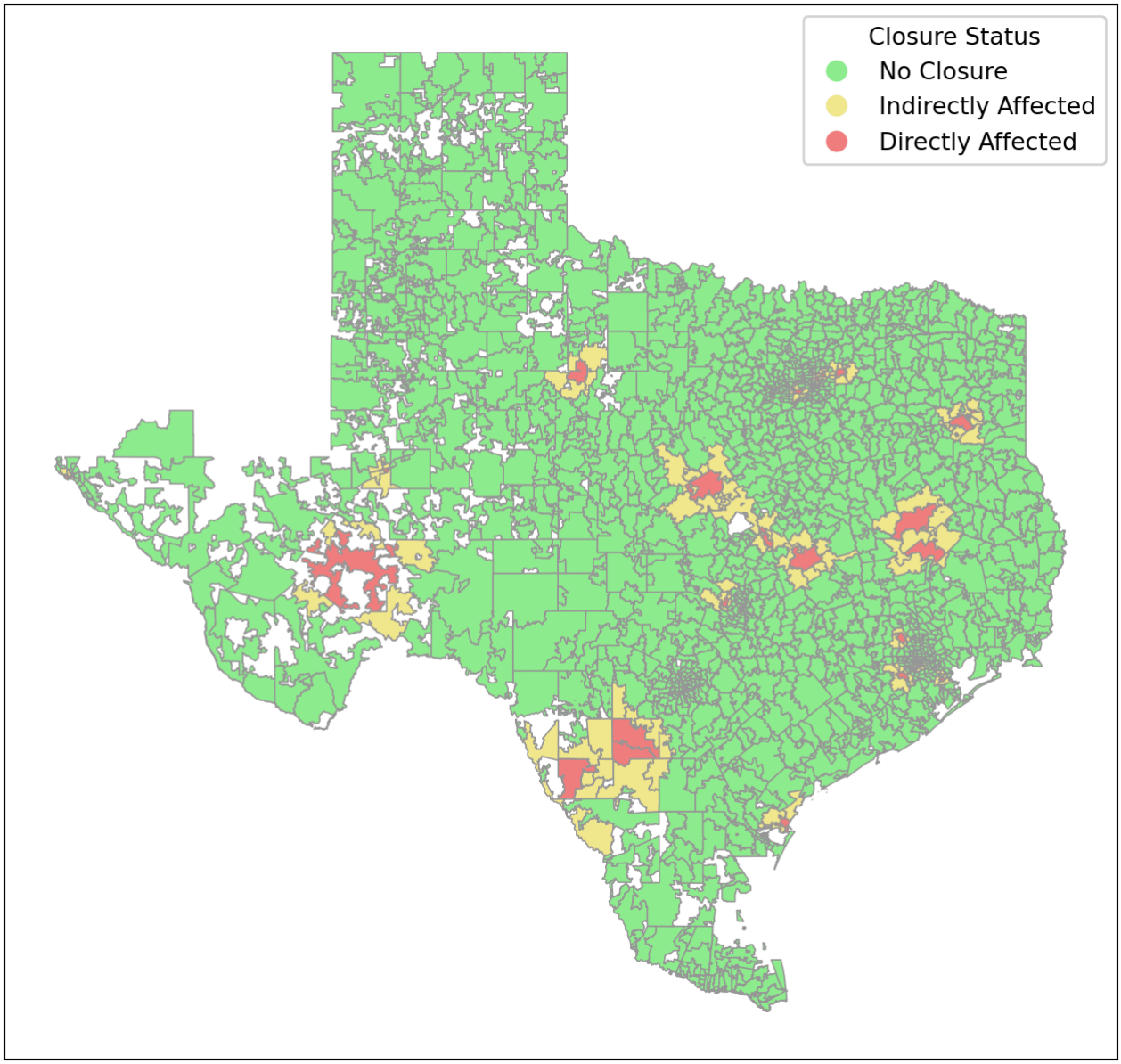
Question 4

```
# Creating a dummy that reflects hospital closures
gdf_tx["affected"] = np.where(gdf_tx["hospital_closures"] > 0, 1, 0)

# Re-assigning 0 as 2 when ZCTA is indirectly affected
gdf_tx["affected"] = np.where(
    (gdf_tx["ZCTA5"].isin(indirectly_affected_zips)), 0.5, gdf_tx["affected"]
)

# Check the distribution of the affected variable to make sure this worked properly
closure_status_distribution = gdf_tx["affected"].value_counts()
print(closure_status_distribution)
```

```
affected
0.0    1774
0.5     136
1.0      25
Name: count, dtype: int64
```

```
# Choosing colors
closure_cmap = ListedColormap(["lightgreen", "khaki", "lightcoral"])

fig, ax = plt.subplots(1, 1, figsize=(10, 8))
gdf_tx.plot(column= "affected", cmap= closure_cmap,
           linewidth=0.6, ax=ax, edgecolor='0.6')

# I also asked AI to help me make this plot prettier. The following lines were coded # with so
ax.set_title("Texas ZIP Codes affected by Hospital Closures (2017-2019)", fontsize=12, weight=
handles = [
    plt.Line2D([0], [0], marker="o", color="w", markerfacecolor= "lightgreen", markersize=10,
    plt.Line2D([0], [0], marker="o", color="w", markerfacecolor= "khaki", markersize=10, label
    plt.Line2D([0], [0], marker="o", color="w", markerfacecolor= "lightcoral", markersize=10,
]

ax.legend(handles=handles, title="Closure Status")

ax.set_xticks([])
```

```
ax.set_yticks([])

plt.show()
```



Reflecting on the exercise (10 pts)

Part (a)

One potential problem with the "first-pass" approach is that it does not consider hospitals that are opening. For instance, the net number of hospitals in a ZIP code will remain unchanged if one closes and a brand new one opens, leading to an inaccurate assessment of the healthcare landscape. Additionally, ZIP codes can change, and this might flag an area for having a decreased number of hospitals when a facility has simply relocated to a different ZIP code. Similarly, if a hospital moves to a new campus serving the same area but is now located in a neighboring ZIP code, this could lead to an erroneous count.

To address these issues, a more effective administrative approach would be to change the way mergers and acquisitions are recorded. For the purpose of accurately determining the number of operating hospitals, it is not necessary to consider whether facilities are under new administration. Instead, these cases should be recorded as "active" with the same CMS code, regardless of name changes. Implementing a tracking system for facility transitions would help ensure that data analysts can maintain accurate records of operational hospitals. While this administrative choice remains unchanged, we must acknowledge that data analysts are often limited by existing practices and rely heavily on approaches such as the first-pass method.

Part (b)

The approach used to identify ZIP codes affected by closures considers only a single dimension: a 10-mile distance. It does not account for factors such as road networks or transportation infrastructure, population density in each area, health needs, or the presence of other healthcare facilities.

An improvement to this approach would be to consider additional variables. The number of people living in the area and the availability of other healthcare facilities are especially important. For instance, if a hospital closes in a densely populated area but there are many other healthcare services available, that area may be less affected compared to other ZIP codes with fewer healthcare options. In addition, we can include in the map the presence of roads which is very important for access to healh facilities.