PS4

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points. We use (*) to indicate a problem that we think might be time consuming.

Style Points (10 pts)

Please refer to the minilesson on code style **here**.

Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Katherine Tu, katherinetu
    - Partner 2 (name and cnet ID): Kohan Chen, kohanchen
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: **KT KC**
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. Katherine: Late coins used this pset: **0** Late coins left after submission: **2**
    - Kohan: 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

Important: Repositories are for tracking code. **Do not commit the data or shapefiles to your repo.** The best way to do this is with `.gitignore`, which we have covered in class. If you do accidentally commit the data, Github has a guide. The best course of action depends on whether you have pushed yet. This also means that both partners will have to download the initial raw data and any data cleaning code will need to be re-run on both partners' computers.

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

```
import pandas as pd
import altair as alt

import warnings
warnings.filterwarnings('ignore')
```

1. What are the variables you pulled?

I pulled variables PRVDR_CTGRY_SBTYP_CD (provider subtype code), PRVDR_CTGRY_CD (provider category code), FAC_NAME (facility name), PRVDR_NUM (CMS Certification Number), PGM_TRMNTN_CD (termination code), and ZIP_CD (zipcode).

2.

a.

```
#import 2016Q4 data
pos2016 = pd.read_csv("/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/pos2016.csv")
```

```
#filter the data by short term and report number of observations
shrt_2016 = pos2016[(pos2016["PRVDR_CTGRY_SBTYP_CD"]== 1) &
 ↪  (pos2016["PRVDR_CTGRY_CD"]== 1)]
print(f"{shrt_2016.shape[0]} hospitals are are reported in this data.")
```

```
7245 hospitals are are reported in this data.
```

This number makes general sense since it reduced the total rows of over 141k in the total 2016 dataset to only over 7000, meaning that only a proportion of these hospitals fit the filter of short-term acute care hospitals. However, there could still be repetitions, since it was not guaranteed that each row represents one hospital.

b.

In the article provided at the beginning of the problem set, it is stated that there are around 5000 short term, acute care hospitals in the United States. This number is 2200 more than the number observed through subsetting the provider codes. The differences could be due to double counting in the data set, for instance from including hospitals that have terminated, or simply including multiple lines of the same data.

3.

```
#read data for 2017Q4, 2018Q4, 2019Q4
pos2017 = pd.read_csv("/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/pos2017.csv")
pos2018 = pd.read_csv("/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/pos2018.csv", encoding="ISO-8859-1")
pos2019 = pd.read_csv("/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/pos2019.csv",encoding="ISO-8859-1")
```

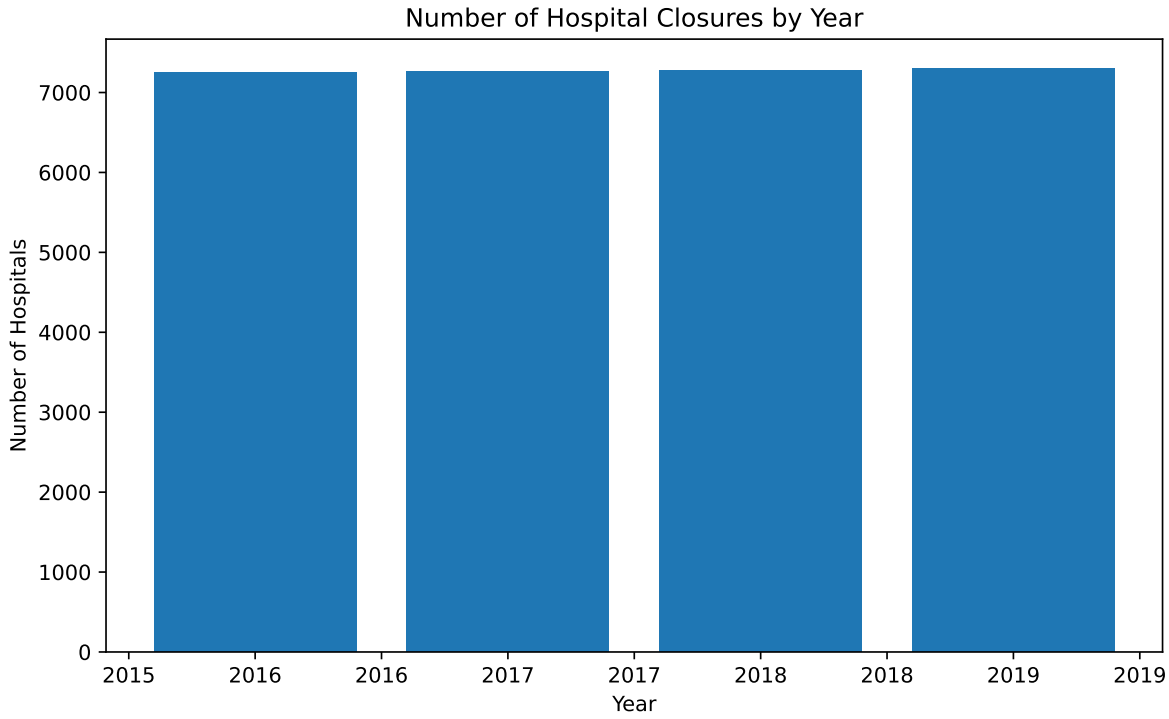
```
#filter the data for short-term hospitals
shrt_2017 = pos2017[(pos2017["PRVDR_CTGRY_SBTYP_CD"]== 1) &
 ↪  (pos2017["PRVDR_CTGRY_CD"]== 1)]
shrt_2018 = pos2018[(pos2018["PRVDR_CTGRY_SBTYP_CD"]== 1) &
 ↪  (pos2018["PRVDR_CTGRY_CD"]== 1)]
shrt_2019 = pos2019[(pos2019["PRVDR_CTGRY_SBTYP_CD"]== 1) &
 ↪  (pos2019["PRVDR_CTGRY_CD"]== 1)]
```

```
#create year column
shrt_2016["Year"] = 2016
shrt_2017["Year"] = 2017
shrt_2018["Year"] = 2018
shrt_2019["Year"] = 2019
```

```
#combine filtered data and count observations by year
shrt_combined = pd.concat([shrt_2016, shrt_2017, shrt_2018, shrt_2019])
observations_by_year = shrt_combined.groupby("Year").size().reset_index(name
 ↪  = "count")
```

```
#create a bar chart for the number of observations per year
import matplotlib.pyplot as plt
```

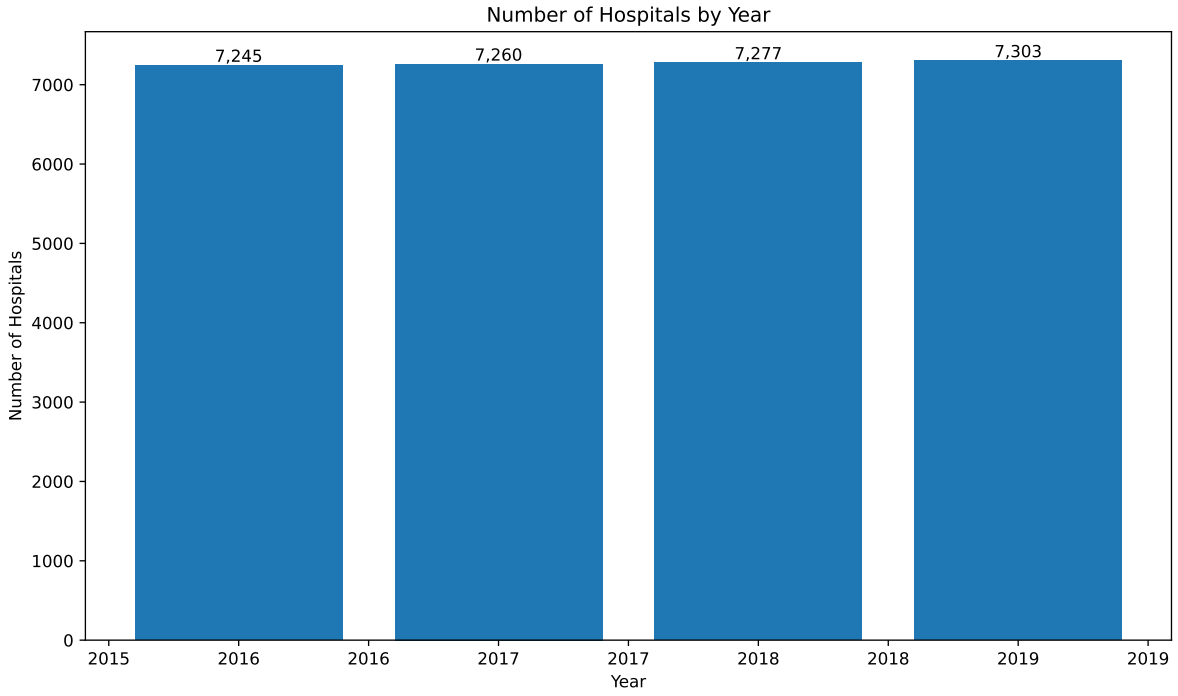
```
plt.figure(figsize=(8, 5))
plt.bar(observations_by_year['Year'].astype(int),
 ↪  observations_by_year['count'])  # Convert to int here
plt.title('Number of Hospital Closures by Year')
plt.xlabel('Year')
plt.ylabel('Number of Hospitals')

plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p:
 ↪  f'{int(x)}'))

plt.tight_layout()
plt.show()
```



4. a.

```
#filter unique hospitals by CMS number in each year
shrt_unique = shrt_combined.drop_duplicates(subset = ["Year","PRVDR_NUM"])
observations_unique = shrt_unique.groupby("Year").size().reset_index(name =
 ↪  "count")
```

```
plt.figure(figsize=(10, 6))
plt.bar(observations_unique['Year'].astype(int),
 ↪  observations_unique['count'])  # Convert to int here

# Customize the chart
plt.title('Number of Hospitals by Year')
plt.xlabel('Year')
plt.ylabel('Number of Hospitals')

# Format x-axis to show integers
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p:
 ↪  f'{int(x)}'))

for i, v in enumerate(observations_unique['count']):
    plt.text(observations_unique['Year'].iloc[i].astype(int), v, format(v,
 ↪  ','),
             ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



b. There was no change after the dataset was fiiltered by the CMS certification number. This means that there were no replications and the data sets were already structured by CMS certification number.

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

1.

```
#filter for active hospitals each year
active_2016 = shrt_2016[shrt_2016["PGM_TRMNTN_CD"]==0]
active_2017 = shrt_2017[shrt_2017["PGM_TRMNTN_CD"]==0]
active_2018 = shrt_2018[shrt_2018["PGM_TRMNTN_CD"]==0]
active_2019 = shrt_2019[shrt_2019["PGM_TRMNTN_CD"]==0]

#standardize the zip
def standardize_zip(df):
    df = df.copy()
    df['ZIP_CD'] =
 ↪  df['ZIP_CD'].astype(float).astype(int).astype(str).str.zfill(5)
    return df

active_2016 = standardize_zip(active_2016)
active_2017 = standardize_zip(active_2017)
active_2018 = standardize_zip(active_2018)
active_2019 = standardize_zip(active_2019)
```

```
#merge 2016 active hospitals with 2017 active hospitals to find ones that are
 ↪  not included in 2017 active.
active_2017_cms = active_2017["PRVDR_NUM"]
merged_1617 = active_2016.merge(active_2017_cms, on="PRVDR_NUM", how="left",
 ↪  indicator=True)
suspected_closed_17 = merged_1617[merged_1617["_merge"] ==
 ↪  "left_only"][["PRVDR_NUM","FAC_NAME", "ZIP_CD"]]
suspected_closed_17["Year"] = 2017
```

```
#do the same for 2018 active hospitals
active_2018_cms = active_2018["PRVDR_NUM"]
merged_1618 = active_2016.merge(active_2018_cms, on="PRVDR_NUM", how="left",
 ↪  indicator=True)
```

```
suspected_closed_18 = merged_1618[merged_1618["_merge"] ==
 ↪  "left_only"][["PRVDR_NUM","FAC_NAME", "ZIP_CD"]]
suspected_closed_18["Year"] = 2018
```

```
#do the same for 2019 active hospitals
active_2019_cms = active_2019["PRVDR_NUM"]
merged_1619 = active_2016.merge(active_2019_cms, on="PRVDR_NUM", how="left",
 ↪  indicator=True)
suspected_closed_19 = merged_1619[merged_1619["_merge"] ==
 ↪  "left_only"][["PRVDR_NUM","FAC_NAME", "ZIP_CD"]]
suspected_closed_19["Year"] = 2019
```

```
#combine the list of closed hospitals and drop duplicates in case they are
 ↪  included in multiple years
sus_closed_combined = pd.concat([suspected_closed_17,
 ↪  suspected_closed_18,suspected_closed_19])
sus_closed_unique = sus_closed_combined.drop_duplicates(subset=["PRVDR_NUM",
 ↪  "FAC_NAME", "ZIP_CD"], keep="first")
print(f"{sus_closed_unique.shape[0]} hospitals fit this definition.")
```

```
174 hospitals fit this definition.
```

2.

```
#sort list by names and print out the top 10 rows
sorted_sus_closed = sus_closed_unique.sort_values(by = "FAC_NAME")
print(sorted_sus_closed[["FAC_NAME","Year"]].head(10))
```

```
                                        FAC_NAME  Year
93                          ABRAZO MARYVALE CAMPUS  2017
299       ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  2017
2276                        AFFINITY MEDICAL CENTER  2018
2019  ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS  2017
2955       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE  2017
1682                        ALLIANCE LAIRD HOSPITAL  2019
2345                      ALLIANCEHEALTH DEACONESS  2019
720                   ANNE BATES LEACH EYE HOSPITAL  2019
528         ARKANSAS VALLEY REGIONAL MEDICAL CENTER  2017
1801            BANNER CHURCHILL COMMUNITY HOSPITAL  2017
```

3.

```
#find the size of each zip code by year
zip_active_2016 = active_2016.groupby("ZIP_CD").size().reset_index(name =
 ↪  "count_2016")
zip_active_2017 = active_2017.groupby("ZIP_CD").size().reset_index(name =
 ↪  "count_2017")
zip_active_2018 = active_2018.groupby("ZIP_CD").size().reset_index(name =
 ↪  "count_2018")
zip_active_2019 = active_2019.groupby("ZIP_CD").size().reset_index(name =
 ↪  "count_2019")
```

```
#find if the number in each zipcode decreased the year after and filter those
 ↪  that are in zipcodes that decreased for 2017
zip_1617 = zip_active_2016.merge(zip_active_2017, on='ZIP_CD',
 ↪  how='outer').fillna(0)
zip_1617["decrease"] = zip_1617["count_2017"] < zip_1617["count_2016"]
suspected_closed_17_filtered =
 ↪  suspected_closed_17[suspected_closed_17["ZIP_CD"].isin(zip_1617[zip_1617["decrease"]]["Z
```

```
#do the same for 2018
zip_1718 = zip_active_2017.merge(zip_active_2018, on='ZIP_CD',
 ↪  how='outer').fillna(0)
zip_1718["decrease"] = zip_1718["count_2018"] < zip_1718["count_2017"]
suspected_closed_18_filtered =
 ↪  suspected_closed_18[suspected_closed_18["ZIP_CD"].isin(zip_1718[zip_1718["decrease"]]["Z
```

```
#do the same for 2019
zip_1819 = zip_active_2018.merge(zip_active_2019, on='ZIP_CD',
 ↪  how='outer').fillna(0)
zip_1819["decrease"] = zip_1819["count_2019"] < zip_1819["count_2018"]
suspected_closed_19_filtered =
 ↪  suspected_closed_19[suspected_closed_19["ZIP_CD"].isin(zip_1819[zip_1819["decrease"]]["Z
```

```
#combine the filtered datasets
filtered_sus_closed_combined = pd.concat([suspected_closed_17_filtered,
 ↪  suspected_closed_18_filtered,suspected_closed_19_filtered])
```

a.

```
#find the amount changed from before the zipcode filter
number_fitting_merge = sus_closed_unique.shape[0] -
 ↪  filtered_sus_closed_combined.shape[0]
print(f"{number_fitting_merge} hospitals fit this definition of potentially
 ↪  being a merger/acquisition.")
```

```
6 hospitals fit this definition of potentially being a merger/acquisition.
```

b.

```
#print the current number of hospitals
print(f"After correcting this, I have {filtered_sus_closed_combined.shape[0]}
 ↪  hospitals left.")
```

```
After correcting this, I have 168 hospitals left.
```

c.

```
#sort by name and print the top 10 rows
sort_filtered_sus_closed = filtered_sus_closed_combined.sort_values(by =
 ↪  "FAC_NAME")
print (sort_filtered_sus_closed.head(10))
```

```
     PRVDR_NUM                                    FAC_NAME ZIP_CD  Year
93      030001                      ABRAZO MARYVALE CAMPUS  85031  2017
299     050196  ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230  2017
2276    360151                      AFFINITY MEDICAL CENTER  44646  2018
2019    330189  ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS  12208  2017
2955    450488   ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE  75662  2017
1682    250159                      ALLIANCE LAIRD HOSPITAL  39365  2019
2345    370032                     ALLIANCEHEALTH DEACONESS  73112  2019
720     100240              ANNE BATES LEACH EYE HOSPITAL  33136  2019
528     060036    ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050  2017
1801    290006         BANNER CHURCHILL COMMUNITY HOSPITAL  89406  2017
```

Download Census zip code shapefile (10 pt)

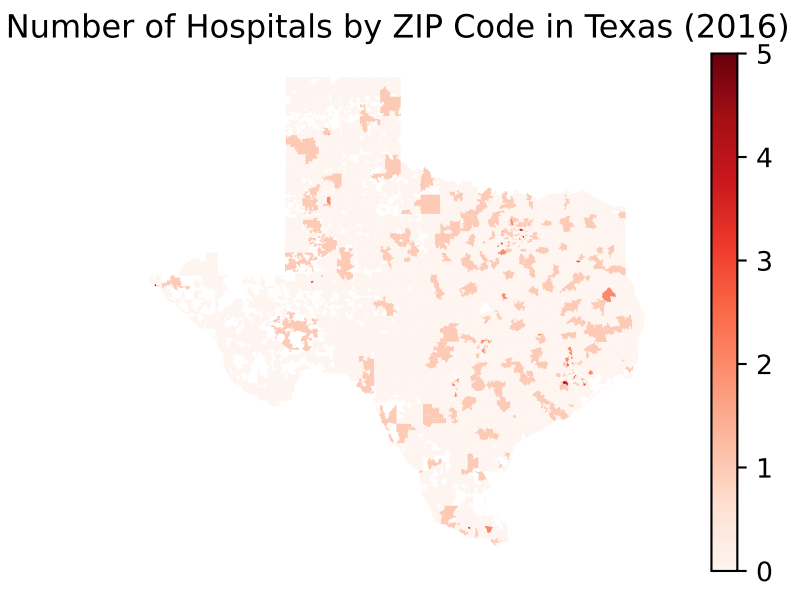
1. a. dbf, prj, shp, shx, and xml files. dbf files contain attribute information, prj describes the Coordinate Reference System, shp has the feature geometrics, shx has the position indices, and xml is a code that contains tags on how the how the file should be structured, stored, and transported.

b. dbf:6.4mb, prj:165 bytes, shp:837.5mb, shx:265kb, xml:16kb

2.

```
import geopandas as gpd
import matplotlib.pyplot as plt

zip_shp = gpd.read_file("/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp")
texas_zips = zip_shp[zip_shp['ZCTA5'].str.startswith(('75', '76', '77', '78',
 ↪  '79'))]

hospital_counts =
 ↪  active_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')
hospital_counts['ZIP_CD'] =
 ↪  hospital_counts['ZIP_CD'].astype(float).astype(int).astype(str).str.zfill(5)

texas_hospitals = texas_zips.merge(hospital_counts,
                                   left_on='ZCTA5',
                                   right_on='ZIP_CD',
                                   how='left')

texas_hospitals['hospital_count'] =
 ↪  texas_hospitals['hospital_count'].fillna(0)

texas_hospitals.plot(column='hospital_count',
 ↪  cmap='Reds',legend=True).set_axis_off()
plt.title('Number of Hospitals by ZIP Code in Texas (2016)')
```

```
Text(0.5, 1.0, 'Number of Hospitals by ZIP Code in Texas (2016)')
```



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

1.

```
zips_all_centroids = zip_shp.copy()
zips_all_centroids['geometry'] = zip_shp['geometry'].centroid
print("Dimensions:", zips_all_centroids.shape)
print("\nColumns:")
for col in zips_all_centroids.columns:
    print(f"- {col}")
```

```
Dimensions: (33120, 6)

Columns:
- GEO_ID
- ZCTA5
- NAME
- LSAD
- CENSUSAREA
- geometry
```

GEO_ID: Unique geographic identifier for each ZIP code area ZCTA5: The 5-digit ZIP Code Tabulation Area code NAME: Text name of the ZCTA LSAD: Legal/Statistical Area Descrip-

tion CENSUSAREA: Area measurement in square meters/units geometry: The centroid point geometry (x,y coordinates) of each ZIP code area"

2.

```
zips_texas_centroids = zips_all_centroids[
    zips_all_centroids['ZCTA5'].str.startswith(('75', '76', '77', '78',
 ↪  '79'))
]
zips_texas_borderstates_centroids = zips_all_centroids[
    zips_all_centroids['ZCTA5'].str.startswith(('75', '76', '77', '78', '79',
        # Oklahoma
        '73', '74',
        # New Mexico
        '87', '88',
        # Arkansas
        '71', '72',
        # Louisiana
        '70'))
]
print("length of unique zips in texas:",
 ↪  len(zips_texas_centroids['ZCTA5'].unique()))
print("length of unique zips in texas and border states:",
 ↪  len(zips_texas_borderstates_centroids['ZCTA5'].unique()))
```

```
length of unique zips in texas: 1935
length of unique zips in texas and border states: 4057
```

3.

```
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospital_counts,
    left_on='ZCTA5',
    right_on='ZIP_CD',
    how='inner'
)

zips_withhospital_centroids = zips_withhospital_centroids[
    zips_withhospital_centroids['hospital_count'] >= 1
]
```

I did the inner merge because we only want the zip codess that appear in both datasets. I merged on ZCTA5 and ZIP_CD because they are ZIP codes. 4. a.

```
import time
test_zips = zips_texas_centroids.head(10).copy()
start_time = time.time()

distances = []
for i, row in test_zips.iterrows():
    distances_to_hospitals =
 ↪  zips_withhospital_centroids.geometry.distance(row.geometry)
    min_distance = distances_to_hospitals.min()
    distances.append(min_distance)

estimated_time = time.time() - start_time
print(f"Estimated time to calculate distances: {estimated_time:.2f} seconds")

total_zips = len(zips_texas_centroids)
estimated_time = (estimated_time / 10) * total_zips
print(f"Estimated time for full calculation: {estimated_time:.2f} seconds")
```

```
Estimated time to calculate distances: 0.03 seconds
Estimated time for full calculation: 6.42 seconds
```

b.

```
start_time = time.time()

distances = []
for i, row in zips_texas_centroids.iterrows():
    distances_to_hospitals =
 ↪  zips_withhospital_centroids.geometry.distance(row.geometry)
    min_distance = distances_to_hospitals.min()
    distances.append(min_distance)

estimated_time = time.time() - start_time
print(f"Full calculation time: {estimated_time:.2f} seconds")
```

```
Full calculation time: 6.67 seconds
```

It is faster then my estimate. Full calculation time: 11.75 seconds and estimated time for full calculation: 18.32 seconds. c.

```
import os
from pyproj import Geod
prj_file = "/Users/kohanchen/Documents/Fall
 ↪  2024/student30538/problem_sets/ps4/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.prj"
if os.path.exists(prj_file):
    with open(prj_file, 'r') as f:
        print("Projection file contents:")
        print(f.read())
print("Miles:",0.017453292519943295*69)
```

```
Projection file contents:
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,29
Miles: 1.2042771838760873
```

The .prj file indicates that the unit is in degrees. 1 degree 69 miles.

5. a.

```
zips_texas_centroids['distance_to_nearest_hospital'] = distances
zips_texas_centroids['distance_miles'] =
 ↪  zips_texas_centroids['distance_to_nearest_hospital'] * 69
```

Original unit is in degrees. b.

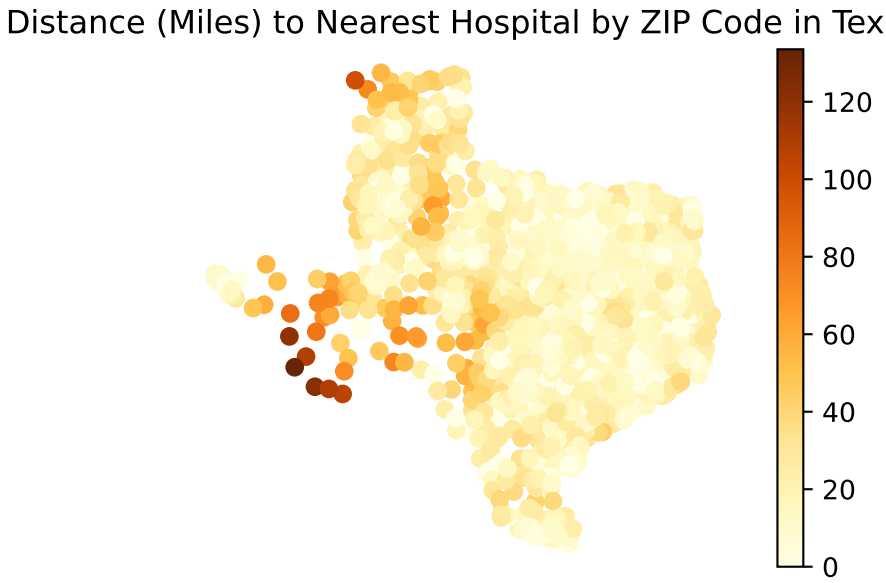
```
average_distance = zips_texas_centroids['distance_miles'].mean()
print(f"\nb. Average distance to nearest hospital: {average_distance:.2f}
 ↪  miles")
print(zips_texas_centroids['distance_miles'].describe())
```

```
b. Average distance to nearest hospital: 14.56 miles
count    1935.000000
mean       14.560207
std        14.278557
min         0.000000
25%         4.128051
50%        11.902677
75%        19.995055
max       133.539995
Name: distance_miles, dtype: float64
```

Yes, these values make sense for Texas c.

```
zips_texas_centroids.plot(
    column='distance_miles',
    cmap='YlOrBr',
    legend=True
).set_axis_off()
plt.title('Distance (Miles) to Nearest Hospital by ZIP Code in Texas')
```

```
Text(0.5, 1.0, 'Distance (Miles) to Nearest Hospital by ZIP Code in Texas')
```



Effects of closures on access in Texas (15 pts)

1.

```
# Filter for Texas ZIP codes (75xxx-79xxx) and years 2016-2019
texas_closures = sort_filtered_sus_closed[
    (sort_filtered_sus_closed['ZIP_CD'].astype(str).str.startswith(('75',
 ↪  '76', '77', '78', '79'))) &
    (sort_filtered_sus_closed['Year'].between(2016, 2019))
]
closure_counts =
 ↪  texas_closures.groupby('ZIP_CD').size().reset_index(name='number_of_closures')
```

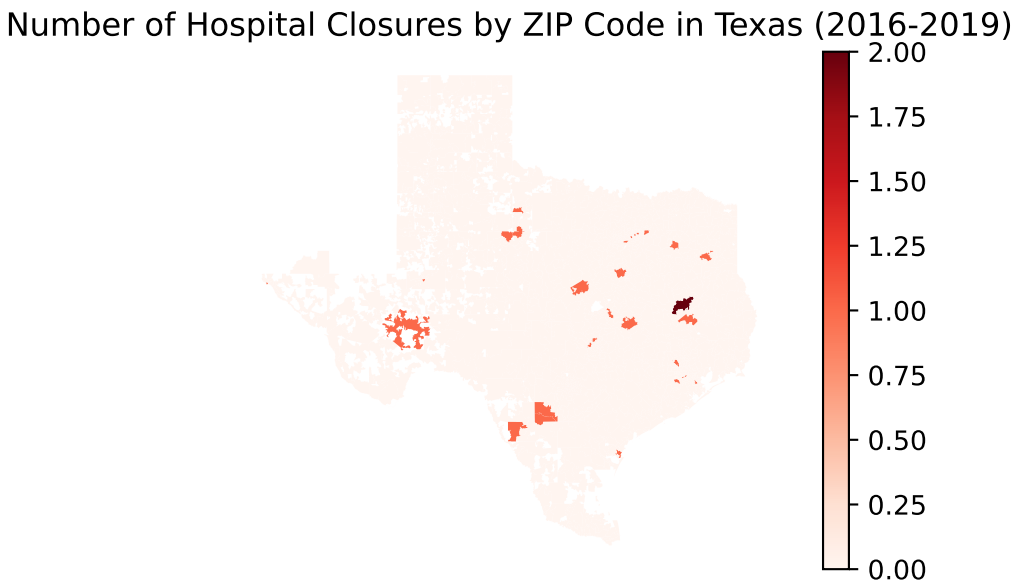
```
closure_counts = closure_counts.sort_values('number_of_closures',
 ↪  ascending=False)
closure_counts['ZIP_CD'] =
 ↪  closure_counts['ZIP_CD'].astype(float).astype(int).astype(str).str.zfill(5)
print("Hospital Closures by ZIP Code in Texas (2016-2019):")
print(closure_counts.to_string(index=False))
```

```
Hospital Closures by ZIP Code in Texas (2016-2019):
ZIP_CD  number_of_closures
 75835                   2
 75042                   1
 77479                   1
 77598                   1
 78017                   1
 78061                   1
 78336                   1
 78613                   1
 78734                   1
 77065                   1
 78834                   1
 79520                   1
 79529                   1
 79553                   1
 79735                   1
 79761                   1
 77429                   1
 77054                   1
 75051                   1
 77035                   1
 76645                   1
 76531                   1
 76520                   1
 76502                   1
 75862                   1
 75662                   1
 75601                   1
 75390                   1
 75235                   1
 75231                   1
 75140                   1
 75087                   1
 79902                   1
```

2.

```
texas_zips = texas_zips.merge(
    closure_counts[['ZIP_CD', 'number_of_closures']],
    left_on='ZCTA5',
    right_on='ZIP_CD',
    how='left'
)
texas_zips['number_of_closures'] = texas_zips['number_of_closures'].fillna(0)

texas_zips.plot(
    column='number_of_closures',
    legend=True,
    cmap='Reds'
).set_axis_off()
plt.title('Number of Hospital Closures by ZIP Code in Texas (2016-2019)')

n_affected = len(closure_counts)
print(f"Number of directly affected ZIP codes: {n_affected}")
```

```
Number of directly affected ZIP codes: 33
```



3.

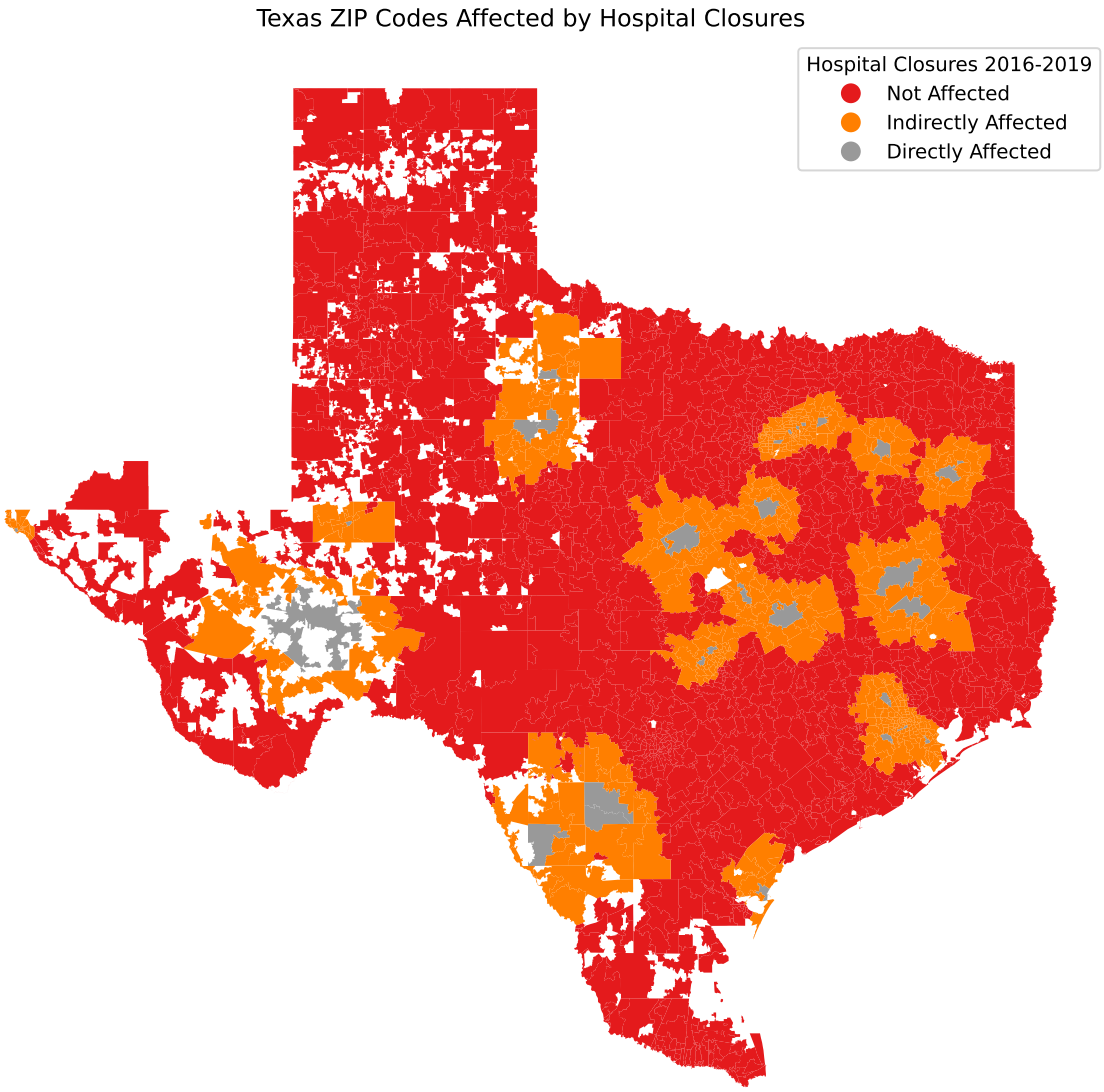
```
#directly affected ZIP codes
directly_affected =
 ↪  texas_zips[texas_zips['ZCTA5'].isin(closure_counts['ZIP_CD'])]

# 10-mile buffer (convert to projected CRS first)
# Texas State Mapping System (EPSG:3081)
directly_affected_proj = directly_affected.to_crs(epsg=3081)
# Create buffer (10 miles = 16093.4 meters)
buffer = directly_affected_proj.geometry.buffer(16093.4)

# Create GeoDataFrame from buffer
buffer_gdf = gpd.GeoDataFrame(geometry=buffer,
 ↪  crs=directly_affected_proj.crs)

texas_zips_proj = texas_zips.to_crs(epsg=3081)

# Spatial join to find indirectly affected ZIP codes
indirectly_affected = gpd.sjoin(
    texas_zips_proj,
    buffer_gdf,
    how='inner',
    predicate='intersects'
)

# Remove ZIPs that had closures from indirect list
indirectly_affected = indirectly_affected[
    ~indirectly_affected['ZCTA5'].isin(closure_counts['ZIP_CD'])
]

n_indirect = len(indirectly_affected['ZCTA5'].unique())
print(f"Number of indirectly affected ZIP codes: {n_indirect}")
```

```
Number of indirectly affected ZIP codes: 577
```

4.

```
texas_zips['impact'] = 0  # Default: Not Affected
texas_zips.loc[texas_zips['ZCTA5'].isin(indirectly_affected['ZCTA5']),
 ↪  'impact'] = 1  # Indirectly
texas_zips.loc[texas_zips['ZCTA5'].isin(closure_counts['ZIP_CD']), 'impact']
 ↪  = 2  # Directly
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
texas_zips.plot(
    column='impact',
    categorical=True,
    cmap='Set1',
    legend=True,
    legend_kwds={
        'labels': ['Not Affected', 'Indirectly Affected', 'Directly
        ↪  Affected'],
        'title': 'Hospital Closures 2016-2019'
    },
    ax=ax
)

plt.title('Texas ZIP Codes Affected by Hospital Closures')
plt.axis('off')
plt.show()
```



Reflecting on the exercise (10 pts)

1. The "first pass" method to identify true closures that are not mergers by removing 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 is imperfect, because it is only based of the zipcodes, and assumes that any ZIP code with a stable or increasing number of hospitals has no true closures. It assumes that any hospital that opened in the same zip code could be due to a merger, when it could be a completely new hospital, and the closure was filtered out from the dataset. A potential way to address that is through

matching the facility names or addresses to identify if there are new hospitals built in the area.

2. We have identified affected zipcodes through calculating if there were any closures within a zipcode, and calculated the 10 miles zone that would be indirectly affected by the closures. However, this approach treats every individual in the zipcode equally, when people vary in their income level, their proximity to hospitals, and more. Hospitals also provide different level of services, and there could be different densities of populations within zipcodes, and the impact of the closure could vary significantly due to these factors. A few ways to improve this measure would be to incorporate population density information to assess what is the access to hospitals per 1000 people.