\RecustomVerbatimEnvironment{verbatim}{Verbatim}{ showspaces = false, showtabs = false, breaksymbolleft={}, breaklines }

PS4: Spatial

AUTHOR
Alison Filbey and Claire Conzelmann

PUBLISHED
November 1, 2024

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

Style Points (10 pts)

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): Alison Filbey afilbey • Partner 2 (name and cnet ID): Claire Conzelmann cconzelmann
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: **CC AF**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set here" (1 point) **CC AF**
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)

```
#libraries
import pandas as pd
import altair as alt
alt.renderers.enable("png")
alt.data_transformers.enable("vegafusion")
import numpy as np
import shapely
from shapely import Polygon, Point
import geopandas as gpd
import matplotlib.pyplot as plt
import time
```

1. (Partner 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?

The variables I pulled as potentially useful are as follows: PRVDR_CTGRY_SBTYP_CD: Provider category and subcategory
PRVDR_CTGRY_CD: Provider category
FAC_NAME: Facility Name
PRVDR_NUM: CMS certification number
STATE_CD: State
PGM-TRMTN-CD: Termination Code
TRMNTN-EXPRTN-DT: Termination Date
ZIP_CD: Zip code

2. (Partner 1) 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?

```
#Reading in 2016 data and filtering for short-term hospitals
pos2016 = pd.read_csv('Data/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv')
pos2016 = pos2016[(pos2016["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2016["PRVDR_CTGRY_CD"] == 1)]
print(len(pos2016))
```

```
7245
```

The dataset has 7,245 observations. Upon first glance, this number seems reasonable. This is seemingly more than the number of registered hospitals in the 2018 American Hospital Association Statistics fact sheet (5,534), which uses data from the 2016 AHA Annual Survey. This discrepancy could come from differences in the definition of a hospital, which according to AHA must meet"AHA's criteria for registration as a hospital facility". This may differ from the CMS criteria for defining a short-term hospital.

3. (Partner 1) 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.

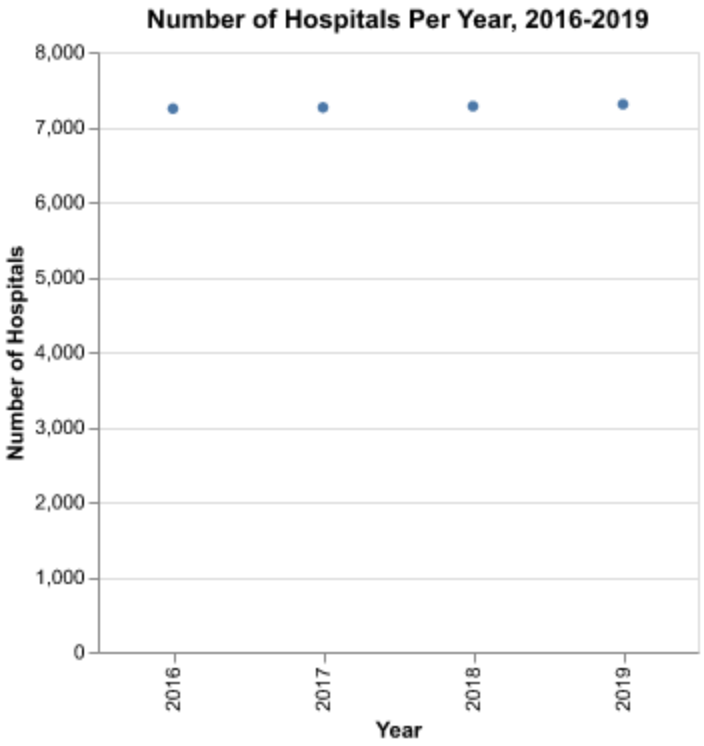
```
#Reading the datasets and creating a columns called year
pos2016['year']=2016
#2017
pos2017 = pd.read_csv('Data/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv')
pos2017 = pos2017[(pos2017["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2017["PRVDR_CTGRY_CD"] == 1)]
pos2017['year']=2017
#2018
file_2018 = 'Data/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv'
pos2018 = pd.read_csv(file_2018, encoding='ISO-8859-1')
pos2018 = pos2018[(pos2018["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2018["PRVDR_CTGRY_CD"] == 1)]
```

```
pos2018['year']=2018
#2019
file_2019 = 'Data/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv'
pos2019 = pd.read_csv(file_2019, encoding='ISO-8859-1')
pos2019 = pos2019[(pos2019["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2019["PRVDR_CTGRY_CD"] == 1)]
pos2019['year']=2019

#Appending the datasets
pos = pd.concat([pos2016, pos2017, pos2018, pos2019])
```

```
#Creating a plot of number of observations per year
plot1=alt.Chart(pos, width=300, height=300, title='Number of Hospitals Per Year, 2016-2019').mark_
).encode(
    alt.X('year:O', axis=alt.Axis(title= 'Year')),
    alt.Y('count()', axis=alt.Axis(title= 'Number of Hospitals'))
)

plot1
```

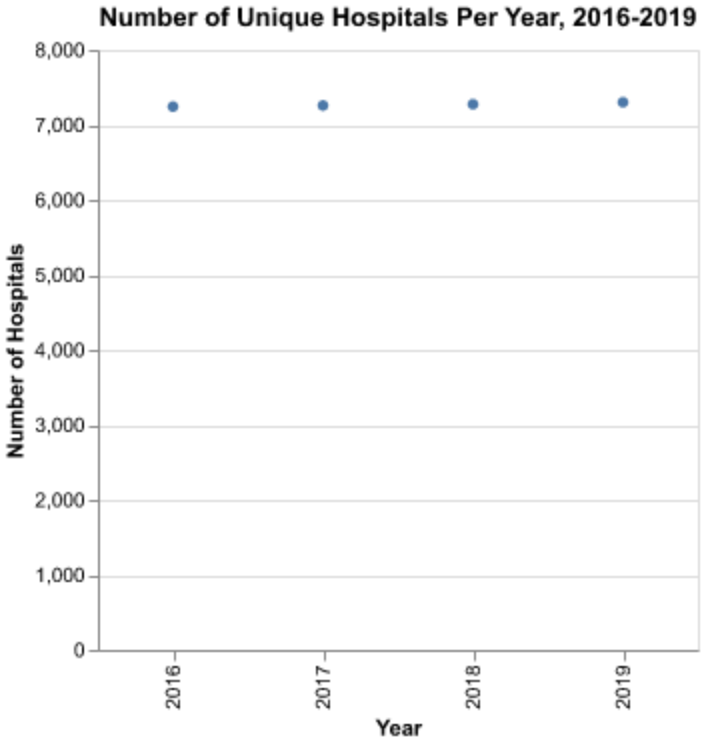


4. (Partner 1) Each hospital is identified by its CMS certification number. Plot the number of unique hospitals in your dataset per year. Compare this to your plot in the previous step. What does this tell you about the structure of the data?

```
#Creating a plot of number of observations per year
plot2 = alt.Chart(pos, width=300, height=300, title='Number of Unique Hospitals Per Year, \
2016-2019').mark_circle().transform_aggregate(
    groupby=['PRVDR_NUM', 'year'],
```

```
    unique_CMS='count():Q').encode(
    alt.X('year:O', axis=alt.Axis(title= 'Year')),
    alt.Y('count(unique_CMS):Q', axis=alt.Axis(title= 'Number of Hospitals'))
)

plot2
```



Comparing plot1 and plot2, it looks as if even after we restrict our number of hospitals to those with unique CMS certification numbers, the the number of hospitals per year remains constant. This tells us that the CMS certification number is a unique identifier of our data as every hospital has exactly one unique CMS number.

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?

```
#create indicator for hospitals active in 2016
pos["active_2016"] = np.where(
  (pos["year"]==2016) & (pos["PGM_TRMNTN_CD"]==0), 1, 0)
pos["active_2016"] = pos.groupby("PRVDR_NUM")["active_2016"].transform("max")

#find max year each hospital appears in dataset
pos["max_year"] = pos.groupby("PRVDR_NUM")["year"].transform("max")
```

```
#create indicator if hospital is ever not active
pos["ever_inactive"] = np.where(
  pos.groupby("PRVDR_NUM")["PGM_TRMNTN_CD"].transform("max")!=0, 1, 0)

#keep hospitals active in 2016 and no longer active
#or hospitals active in 2016 and disappear before 2019
inactive_hospitals = pos.loc[
  ((pos["active_2016"]==1) & (pos["ever_inactive"]==1)) |
  ((pos["active_2016"]==1) & (pos["max_year"]!=2019))]

#groupby hospital and status
grouped = inactive_hospitals.groupby(
  ["PRVDR_NUM", "PGM_TRMNTN_CD"])["year"].min().reset_index()

#drop active rows
grouped = grouped.loc[grouped["PGM_TRMNTN_CD"] != 0]

#rename year
grouped = grouped.rename({"year":"inactive_year"}, axis=1).drop(
  "PGM_TRMNTN_CD", axis=1)

#merge inactive year into inactive hospitals df
inactive_hospitals= pd.merge(
  inactive_hospitals, grouped, on="PRVDR_NUM", how="left")

#keep requested variables
inactive_hospitals = inactive_hospitals[[
  "FAC_NAME", "ZIP_CD", "inactive_year", "PRVDR_NUM", "year"]]

#count number of hospitals that went inactive
print(len(inactive_hospitals["PRVDR_NUM"].unique()))
```

```
174
```

There are 174 hospitals that became inactive at some point between 2016 and 2019.

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

```
#drop duplicates
inactive_hospitals_nodup = inactive_hospitals.drop_duplicates(subset="PRVDR_NUM")

#sort by facility name and print first 10 rows
print(inactive_hospitals_nodup[["FAC_NAME", "inactive_year"]].sort_values(
  "FAC_NAME").head(10))
```

```
                                     FAC_NAME  inactive_year
4                          ABRAZO MARYVALE CAMPUS           2017
10       ADVENTIST MEDICAL CENTER - CENTRAL VALLEY           2017
97                         AFFINITY MEDICAL CENTER           2018
```

```
80   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS          2017
140       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE          2017
62                          ALLIANCE LAIRD HOSPITAL          2019
101                        ALLIANCEHEALTH DEACONESS          2019
26                    ANNE BATES LEACH EYE HOSPITAL          2019
21          ARKANSAS VALLEY REGIONAL MEDICAL CENTER          2017
69              BANNER CHURCHILL COMMUNITY HOSPITAL          2017
```

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

```
#calculate number of active hospitals in each zipcode by year
active_count = pos[pos["PGM_TRMNTN_CD"] == 0]
active_count = active_count.groupby(["ZIP_CD", "year"]).size().reset_index(
  name="n_fac_zip")

#merge the counts back into the original df and the inactive df
pos = pd.merge(pos, active_count, on=["ZIP_CD", "year"], how="left")
inactive_hospitals = pd.merge(
  inactive_hospitals, active_count, on=["ZIP_CD", "year"], how="left")

#replace missing n active hospitals with zero
pos["n_fac_zip"].fillna(0, inplace=True)
inactive_hospitals["n_fac_zip"].fillna(0, inplace=True)

#create lag for number of hospitals in each zipcode
#I googled how to create a lag of a variable and followed this example
#https://discuss.datasciencedojo.com/t/how-to-create-lags-and-leads-of-a-column/1023
inactive_hospitals = inactive_hospitals.sort_values(by=["PRVDR_NUM", "year"])
inactive_hospitals["lag_n_fac_zip"] = inactive_hospitals.groupby(
  "PRVDR_NUM")["n_fac_zip"].shift(1)

#keep hospitals in zipcodes that saw decrease in number of hospitals after closure
inactive_hospitals = inactive_hospitals.loc[
  (inactive_hospitals["year"]==inactive_hospitals["inactive_year"]) &
  (inactive_hospitals["lag_n_fac_zip"] > inactive_hospitals["n_fac_zip"])]
```

```
C:\Users\aliso\AppData\Local\Temp\ipykernel_33628\2235526038.py:12: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value},
inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on
the original object.


  pos["n_fac_zip"].fillna(0, inplace=True)
C:\Users\aliso\AppData\Local\Temp\ipykernel_33628\2235526038.py:13: FutureWarning: A value is
```

```
trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value},
inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on
the original object.


  inactive_hospitals["n_fac_zip"].fillna(0, inplace=True)
```

a Before this correction, there were 174 hospitals that had a suspected closure between 2016 and 2019. After this correction, there are 167 hospitals that had a suspected closure, meaning there are 7 suspected mergers/aquisitions.

b There are 167 hospitals with a suspected closure after correcting for this.

c

```
#sort by facility name and print first 10 rows
print(inactive_hospitals[["FAC_NAME", "inactive_year"]].sort_values(
  "FAC_NAME").head(10))
```

```
                                      FAC_NAME  inactive_year
178                        ABRAZO MARYVALE CAMPUS           2017
184     ADVENTIST MEDICAL CENTER - CENTRAL VALLEY           2017
445                        AFFINITY MEDICAL CENTER           2018
254  ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS           2017
314       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE           2017
584                        ALLIANCE LAIRD HOSPITAL           2019
623                       ALLIANCEHEALTH DEACONESS           2019
195        ARKANSAS VALLEY REGIONAL MEDICAL CENTER           2017
504            ASCENSION NE WISCONSIN MERCY CAMPUS           2018
610                   ATRIUM HEALTH KINGS MOUNTAIN           2019
```

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?

The first file type is a DBF. This is in a database file format, which is used to store tabular attribute data. It is 6,725KB. The second file type is a PRJ file, which stores Coordinate Reference System information. It is 1 KB. The third file is a SHP file, which is a feature geometry. The fourth file is a SHX file, which is a positional index of the feature geomtery. It is 817,915 KB. The fifth file type is an XML file that consists of International Organization for Standardization (ISO 191) metadata in Extensible Markup Language (XML) format and entity and attribute of ISO 191 metadata in XML format. It is 16 KB.

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.

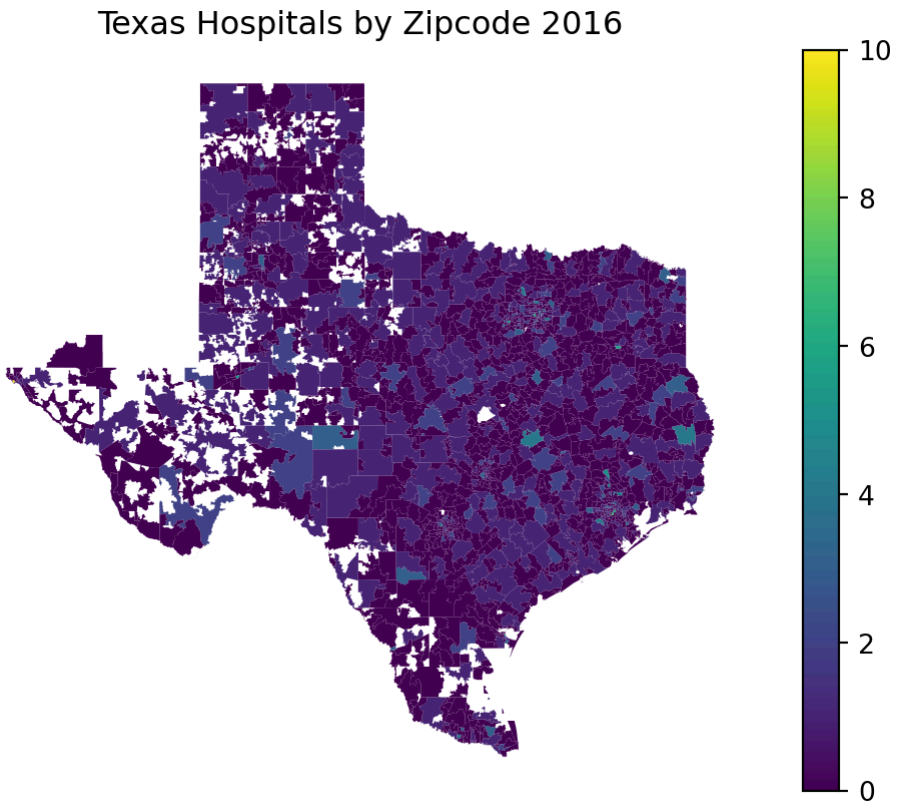
```
#Loading the shapefile
shapefile = "data/gz_2010_us_860_00_500k.shp"
census_data = gpd.read_file(shapefile)

#Filter by shapefile zipcode
texas_zipcodes = ("75", "76", "77", "78", "79")
texas_census_data = census_data[census_data['ZCTA5'].str.startswith(texas_zipcodes)]

#Filter hospital data by zipcode
texas_pos_data_2016 = pos[(pos['STATE_CD']== 'TX') & (pos["year"] == 2016)]
texas_pos_data_2016_zip = texas_pos_data_2016.groupby('ZIP_CD')['PRVDR_NUM'].count(\
  ).reset_index()
texas_pos_data_2016_zip['zip_string'] = texas_pos_data_2016_zip['ZIP_CD'].astype(int\
  ).astype(str)

#Creating a choropleth
#Used source:https://towardsdatascience.com/lets-make-
#a-map-using-geopandas-pandas-and
#-matplotlib-to-make-a-chloropleth-map-dddc31c1983d
texas_pos_data_2016_map = texas_census_data.set_index("NAME" \
  ).join(texas_pos_data_2016_zip.set_index('zip_string'))
texas_pos_data_2016_map['PRVDR_NUM'] = texas_pos_data_2016_map['PRVDR_NUM'\
  ].fillna(0)

texas_pos_data_2016_map.plot(column='PRVDR_NUM', legend=True)
plt.title("Texas Hospitals by Zipcode 2016")
plt.axis("off")
plt.show()
```



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 and what do each of the columns mean?

```
#create geodataframe that is centroids of all zipcodes nationally
zips_all_centroids = census_data.centroid.reset_index(name="centroid")
zips_all_centroids.head()
```

```
C:\Users\aliso\AppData\Local\Temp\ipykernel_33628\3703024821.py:2: UserWarning: Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a projected CRS before this operation.

  zips_all_centroids = census_data.centroid.reset_index(name="centroid")
```

	index	centroid
0	0	POINT (-72.64107 42.21257)
1	1	POINT (-72.86985 42.28786)
2	2	POINT (-72.71162 42.35349)
3	3	POINT (-72.45805 42.19215)
4	4	POINT (-72.3243 42.09165)

The resulting GeoDataFrame has 33,120 rows and 2 columns. The first column is the index. The second column is the point geometry of the centroid which contains the longitude and latitude coordinates of each zip code centroid.

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?

```
#create geodataframe that is centroids of Texas zipcodes
zips_texas_centroids = texas_census_data.centroid.reset_index(
  name="centroid")
len(zips_texas_centroids)
```

```
C:\Users\aliso\AppData\Local\Temp\ipykernel_33628\2570516493.py:2: UserWarning: Geometry is in a geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a projected CRS before this operation.

  zips_texas_centroids = texas_census_data.centroid.reset_index(

1935
```

There are 1,935 zipcodes in Texas.

I had already written the code to use a function to check if zipcodes bordered texas before this question was changed to be simpler. Based off Professor Shi's comment on Ed stating we could get extra credit if we had done this already, I am leaving my code the way it is instead of changing it based off the new question.

```
#dissolve zipcodes within texas so it's a singular polygon
texas_dissolved = texas_census_data.dissolve()

#determine which zipcodes border texas
def check_intersections(poly1, poly2):
    """
    Determine if poly1 and poly2 intersect

    Parameters:
    -poly1 and poly2 are singular polygons

    Returns:
    -a boolean (True or False) indicating if the polygons intersect
    """

    intersects = poly1.intersects(poly2)[0]
    return intersects

#create an empty dictionary to store results
tx_bordering_zips = {"ZCTA5": [], "borders_tx": []}
```

```
for row in range(len(census_data)):
  #iterate through each polygon in census data and check if it intersects texas
  intersects = check_intersections(
    texas_dissolved["geometry"], census_data["geometry"][row])

  #get zip code of current polygon
  zipcode = census_data["ZCTA5"][row]

  #store results
  tx_bordering_zips["ZCTA5"].append(zipcode)
  tx_bordering_zips["borders_tx"].append(intersects)

tx_bordering_zips = pd.DataFrame(tx_bordering_zips)

#keep only zipcodes that border texas and extract first two digits to find state prefix
tx_bordering_zips = tx_bordering_zips.loc[tx_bordering_zips["borders_tx"] == 1]
tx_bordering_zips["state_prefix"] = tx_bordering_zips["ZCTA5"].str[:2]

#find state prefixes
border_zips = tx_bordering_zips["state_prefix"].unique()

#using Wikipedia page from Pset, make sure all prefixes for border states are in list
#need to do this because counties that border TX do not account for all prefixes in given border s
border_zips = np.append(border_zips, [
  "87", "72", "880", "881", "882", "883", "884"])
border_zips = np.delete(border_zips, 9)
border_zips = tuple(border_zips)

#create centroids of texas and borderstate zipcodes
zips_texas_borderstates_centroids = census_data[
  census_data["ZCTA5"].str.startswith(border_zips)].centroid.reset_index()
len(zips_texas_borderstates_centroids)
```

```
C:\Users\aliso\AppData\Local\Temp\ipykernel_33628\2711199785.py:52: UserWarning: Geometry is in a
geographic CRS. Results from 'centroid' are likely incorrect. Use 'GeoSeries.to_crs()' to re-
project geometries to a projected CRS before this operation.

  census_data["ZCTA5"].str.startswith(border_zips)].centroid.reset_index()

4057
```

There are 4,057 zipcodes in Texas or in states that border Texas.

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 GeoDataFrame zips_withhospital_centroids What kind of merge did you decide to do, and what variable are you merging on?

```
#turn zipcode in hospital data to string
active_count["ZCTA5"] = active_count["ZIP_CD"].astype(int).astype(str)
```

```
#keep only zipcodes with active hospitals in 2016
active_count_2016 = active_count.loc[active_count["year"]==2016]

#keep polygons with active hospitals in 2016
hosp_active_2016 = pd.merge(census_data, active_count_2016, on="ZCTA5", how="inner")

#spatial join centroids with kept polygons
#I googled geopandas spatial join syntax and used this website as a guide
#https://geopandas.org/en/stable/docs/reference/api/geopandas.sjoin.html
zips_withhospital_centroids = gpd.sjoin(zips_texas_borderstates_centroids,
  hosp_active_2016,
  how="inner", predicate="within")
```

I first merged the list of zipcodes with active hospitals in 2016 from the previous sections to the census zipcode shapefile, merging on zipcode ("ZCTA5"). Then I performed a spatial join of the file containing polygons for zipcodes with active hospitals with the file containing Texas and border state centroids. The spatial join keeps centroids that are within the polygons in the active hosptial zipcode GeoDataFrame.

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?

```
#subset to 10 zipcodes in Texas first
zips_texas_centroids_subset = zips_texas_centroids.head(10)

#reset index to loop through indices sequentially
zips_texas_centroids_subset = zips_texas_centroids_subset.reset_index(drop=True)
zips_withhospital_centroids = zips_withhospital_centroids.reset_index(drop=True)

#set start time
start_time = time.time()

#calculate nearest distance
for geom1 in range(len(zips_texas_centroids_subset)):
  #create empty list to hold distances for given centroid/zipcode
  distances = []

  for geom2 in range(len(zips_withhospital_centroids)):
    #loop through zips with hospitals and calcualate distance from current tx zipcode
    #I googled how to calculate distance between two points and used this resource
    #https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.distance.html
    distance = zips_texas_centroids_subset.geometry[geom1].distance(
      zips_withhospital_centroids.geometry[geom2])
    distances.append(distance)

  #calculate minimum distance to zip with hospital for this tx zip
```

```
  min_dist = min(distances)
  zips_texas_centroids_subset.loc[geom1, "min_dist"] = min_dist

#set end time
end_time = time.time()
end_time - start_time
```

```
0.19080114364624023
```

It took 0.07 seconds for this code to run on a subset of 10 texas zipcodes. There are 1,935 total Texas zipcodes, so on the full dataset, it should take approximately 13.54 seconds.

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

```
#reset index to loop through indices sequentially
zips_texas_centroids = zips_texas_centroids.reset_index(drop=True)
zips_withhospital_centroids = zips_withhospital_centroids.reset_index(drop=True)

#set start time
start_time = time.time()

#calculate nearest distance
for geom1 in range(len(zips_texas_centroids)):
  #create empty list to hold distances for given centroid/zipcode
  distances = []

  for geom2 in range(len(zips_withhospital_centroids)):
    #loop through zips with hospitals and calcualate distance from current tx zipcode
    #I googled how to calculate distance between points and used this example
    #https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.distance.html
    distance = zips_texas_centroids.geometry[geom1].distance(
      zips_withhospital_centroids.geometry[geom2])
    distances.append(distance)

  #calculate minimum distance to zip with hospital for this tx zip
  min_dist = min(distances)
  zips_texas_centroids.loc[geom1, "min_dist"] = min_dist

#set end time
end_time = time.time()
end_time - start_time
```

```
36.712746381759644
```

It took 11.7 seconds for this to run on the full sample of Texas zipcodes. This is somewhat close but 3 seconds off of my initial estimate of how long it would take.

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 .prj file tells us the unit is in degrees. The true conversion from degrees to miles takes into account the degree of latitude, but we will approximate and use the formula that 1 mile is equal to 69.17 degrees. (I used this source for the conversion)

5. Calculate the average distance to the nearest hospital for each zip code in Texas. **a** What unit is this in?

```
#calculate average minimum distance to hospital
mean_distance_val = zips_texas_centroids["min_dist"].mean()
mean_distance_val
```

```
np.float64(0.20807054502865063)
```

The average distance for zipcodes in Texas to a zipcode with a hospital is 0.208 degrees.

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

```
#calculate average minimum distance to hospital
mean_distance_mi = zips_texas_centroids["min_dist"].mean()*69.17
mean_distance_mi
```

```
np.float64(14.392239599631765)
```

This is in degrees. To calculate the average distance in miles, we multiply by 69.17. So the average distance is 14.392 miles. This value represents the average distance for each zipcode in Texas to a zipcode with at least one hospital. This seems fairly reasonable to me. It actually is lower than what I was expecting, given that Texas is so large and has very rural counties. I would have expected some rural counties to be farther away from hospitals.

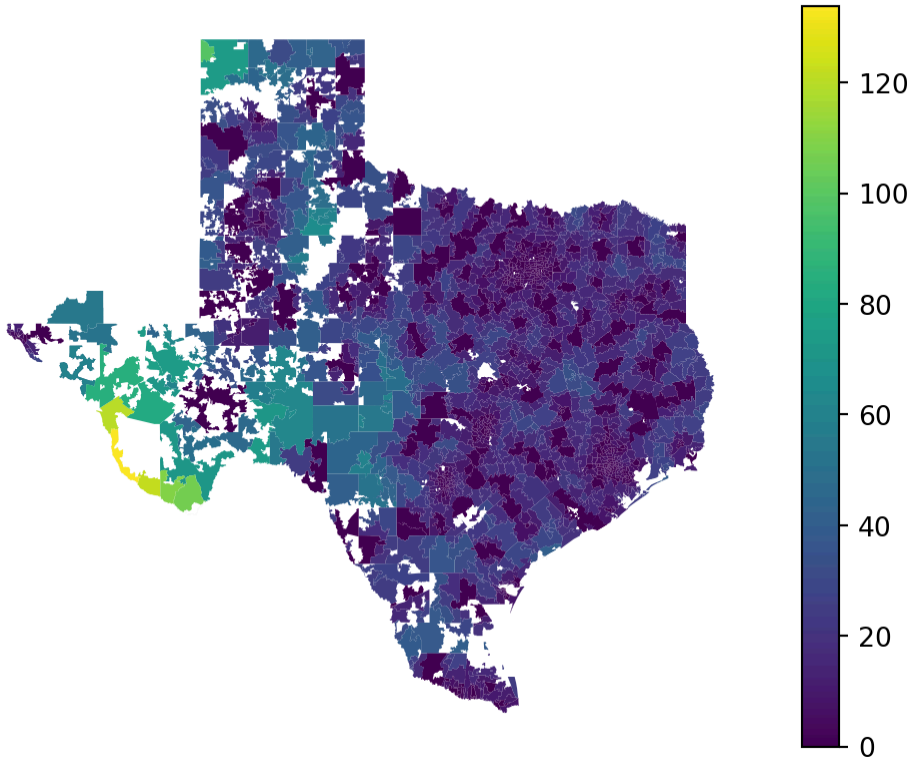
c Map the value for each zip code.

```
#convert distance to nearest hospital to miles
zips_texas_centroids["min_dist_mi"] = zips_texas_centroids["min_dist"]*69.17

#merge centroids back to zipcode data
zips_texas_w_centroids = gpd.sjoin(texas_census_data, zips_texas_centroids,
  how="left", predicate="contains")

#map average minimum distance to hosptial for each zipcode
zips_texas_w_centroids.plot(column="min_dist_mi", legend=True)
plt.title("Average Distance (in miles) to Nearest Hospital by Texas Zipcode")
plt.axis("off")
plt.show()
```

Average Distance (in miles) to Nearest Hospital by Texas Zipcode



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 closures by zipcode.

```
#Creating a Texas dataset that only contains hospitals that were ever inactive
inactive_hospitals['zip_string'] = inactive_hospitals['ZIP_CD'].astype(int).astype(str)
texas_pos_data_inactive = inactive_hospitals[inactive_hospitals['zip_string'].str.startswith(\
  texas_zipcodes)]

#Grouping this by zipcode
texas_pos_data_group = texas_pos_data_inactive.groupby('zip_string')['PRVDR_NUM'].count(\
  ).reset_index()
print(texas_pos_data_group)
```

```
   zip_string  PRVDR_NUM
0       75042          1
1       75051          1
2       75087          1
3       75140          1
4       75231          1
5       75235          1
6       75390          1
7       75601          1
```

```
8       75662          1
9       75835          1
10      75862          1
11      76502          1
12      76520          1
13      76531          1
14      76645          1
15      77035          1
16      77054          1
17      77065          1
18      77429          1
19      77479          1
20      77598          1
21      78017          1
22      78061          1
23      78336          1
24        785          1
25      78613          1
26      78734          1
27      78834          1
28      79520          1
29      79529          1
30      79553          1
31      79735          1
32      79761          1
33      79902          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?

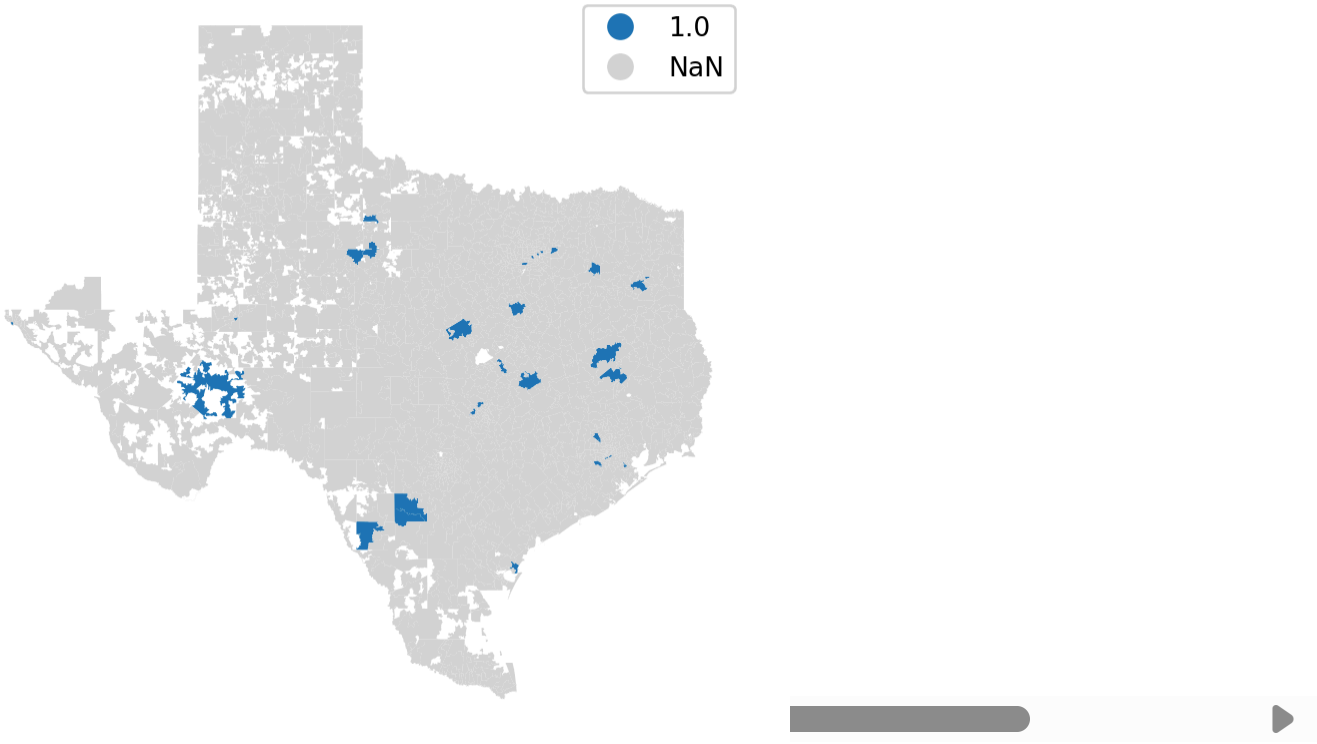
```
#Filtering by directly affected zipcodes
texas_inactive_zipcodes = texas_pos_data_group['zip_string'].astype(int).astype(\
  str).tolist()


#Joining the shapefile
texas_pos_data_group_map = texas_census_data.set_index("NAME").join(\
  texas_pos_data_group.set_index('zip_string'))

#Creating the choropleth
#Used this website for reference on how to set the outline for zipcodes with missing values
#https://stackoverflow.com/questions/73848143/using-missing-kwds-with-geopandas-changes-the-shape
texas_pos_data_group_map.plot(column='PRVDR_NUM', legend=True, missing_kwds=dict(\
  color="lightgray"), categorical=True)
plt.title("Texas Hospitals That Were Ever Inactive by Zipcode")
plt.axis("off")
plt.show()
```

```
#Number of directly affected zipcodes
len(texas_inactive_zipcodes)
```

Texas Hospitals That Were Ever Inactive by Zipcode



34

There are 34 directly affected zipcodes.

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?

```
#Used the following sources first to understand how to create a 10 mile
# buffer and second looked up the EPSG
#https://epsg.io/3081
#https://stackoverflow.com/questions/51263138/how-to-create-an-accurate-buffer-of-5-miles-around-
#Creating a 10 mile buffer
texas_census_data_inactive = texas_census_data[texas_census_data['NAME'].isin(\
  texas_inactive_zipcodes)]
texas_census_data_inactive = texas_census_data_inactive.to_crs(epsg=3081)
buffer_10_miles = (10 * 1000) * 1.60934
texas_census_data_inactive['geometry_10_miles'\
  ] = texas_census_data_inactive.geometry.buffer(buffer_10_miles)

#Creating a GeoPandas dataframe with the 10 miles surrounding
#
```

```
ten_mile_radius = texas_census_data_inactive.set_geometry('geometry_10_miles')

ten_mile_radius = gpd.GeoDataFrame(geometry=texas_census_data_inactive['geometry_10_miles'])

#Ensuring the Texas census data uses the same EPSG
texas_census_data = texas_census_data.to_crs(epsg=3081)

texas_census_data_ten = gpd.sjoin_nearest(
ten_mile_radius,
texas_census_data,
how='inner',
distance_col="distance"
)

texas_dist_miles_zipcodes = texas_census_data_ten['ZCTA5'].unique().tolist()
len(texas_dist_miles_zipcodes)
```

```
610
```

There are 610 zipcodes, and since there are 34 directly affected areas included there are 576 indirectly affected zipcodes.

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.

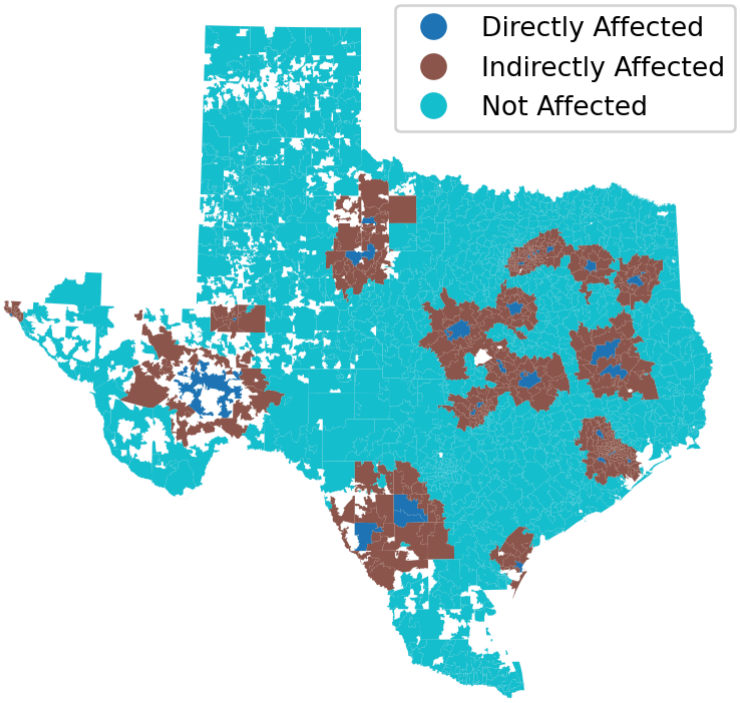
```
#Creating a Texas dataset
texas_pos_data = pos[(pos['STATE_CD']== 'TX')]

#Grouping by zip code and creating a string variable of zipcode
all_texas_zipcodes = texas_pos_data.groupby('ZIP_CD')['ever_inactive'].sum().reset_index()
all_texas_zipcodes['zip_string'] = all_texas_zipcodes['ZIP_CD'].astype(int).astype(str)

#Joining with the geo data and creating an indicator variable of
# whether a zipcode was directly affected, indirectly affected, or not affected.
texas_pos_data_all_map = texas_census_data.set_index("NAME").join(all_texas_zipcodes.set_index('z
texas_pos_data_all_map['category'] = np.where(texas_pos_data_all_map['ZCTA5'].isin(texas_inactive_
texas_pos_data_all_map['category'] = np.where((texas_pos_data_all_map['category'] != \
   "Directly Affected") & (texas_pos_data_all_map['ZCTA5'].isin(texas_dist_miles_zipcodes\
    )), "Indirectly Affected", texas_pos_data_all_map['category'])
len(texas_pos_data_all_map[texas_pos_data_all_map['category']==\
  "Indirectly Affected"])

#Creating a choropleth
texas_pos_data_all_map.plot(column='category', legend=True)
plt.title("Texas Zipcodes by Hospital Closure Impact")
plt.axis("off")
plt.show()
```

Texas Zipcodes by Hospital Closure Impact



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?

A potential issue with this “first pass” method where we 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 is that new hospitals could have opened in these zipcodes in the years after the suspected closures or hospitals, leading to more hospitals in the zipcode. A better way to find hospital closures would be to match on name or addresses in each dataset to determine whether any of the hospitals marked as terminated are in the dataset in the next year.

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?

Having at least one closure or being near a zipcode with at least one closure could measure some level of affectedness, but it’s possible that if one zipcode had multiple hospitals and only one of them closed, the remaining hospitals were still able to provide adequate care for the surrounding population. Perhaps it would be better to classify zipcodes as affected if they were in or near a zipcode that had a closure of its only hospital. Or we could look at why the hospitals closed. If a hospital closed because there was population decline in the area and the services were no longer needed, then that zipcode probably wouldn’t be greatly affected. But if there was no population change and a hospital closed, then perhaps we

could assume that zipcode would be more affected, as the population would not be able to receive the medical services they need.