Problem Set 4

Sarah Morrison

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 (Sarah Morrison morrisons):
    - Partner 2 (Sarah Morrison morrisons):
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: **SM**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: **1** 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 geopandas as gpd
import pandas as pd
import altair as alt
import time
import shapely
from shapely import Point
from shapely.ops import nearest_points
```

1. PRVDR_CTGRY_CD, PRVDR_CTGRY_SBTYP_CD, FAC_NAME, PRVDR_NUM, ZIP_CD, CBSA-URBN-RRL-IND, CITY_NAME, STATE_CD, PGM_TRMNTN_CD
2.

```
# read in 2016 data
filepath =
 ↪  "/Users/sarahmorrison/Downloads/Github/problem-set-4-sarah/pos2016.csv"
pos2016 = gpd.read_file(filepath)
```

```
# filter 2016 data for short-term hospitals
pos2016_sth = pos2016[(pos2016['PRVDR_CTGRY_SBTYP_CD']=='01') &
 ↪  (pos2016['PRVDR_CTGRY_CD']=='01')]
pos2016_sth.shape
# add year columb
pos2016_sth['year'] = 2016
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/1983752376.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pos2016_sth['year'] = 2016
```

a. There are 7245 hospitals reported in this data. The article says there are nearly 5000 short-term hospitals, so this number seems big.

b. The number could be bigger than the article says because of duplicate hospitals. Some hospitals are listed in the data twice because of a name change.

3.

```
# read in 2017 data
filepath =
 ↪  "/Users/sarahmorrison/Downloads/Github/problem-set-4-sarah/pos2017.csv"
pos2017 = gpd.read_file(filepath)
```

```
# read in 2018 data
filepath2 =
 ↪  "/Users/sarahmorrison/Downloads/Github/problem-set-4-sarah/pos2018.csv"
pos2018 = gpd.read_file(filepath2, encoding='latin1') # helped by Ralph
 ↪  Valery Valiere ed discussion comment
```

```
# read in 2019 data
filepath =
 ↪  "/Users/sarahmorrison/Downloads/Github/problem-set-4-sarah/pos2019.csv"
pos2019 = gpd.read_file(filepath, encoding='latin1')
```

```
# filter 2019 data for short-term hospitals
pos2017_sth = pos2017[(pos2017['PRVDR_CTGRY_SBTYP_CD']=='01') &
 ↪  (pos2017['PRVDR_CTGRY_CD']=='01')]
pos2017_sth.shape
# add year column
pos2017_sth['year'] = 2017
# filter 2018 data for short-term hospitals
pos2018_sth = pos2018[(pos2018['PRVDR_CTGRY_SBTYP_CD']=='01') &
 ↪  (pos2018['PRVDR_CTGRY_CD']=='01')]
pos2018_sth.shape
# add year column
pos2018_sth['year'] = 2018
```

```
# filter 2019 data for short-term hospitals
pos2019_sth = pos2019[(pos2019['PRVDR_CTGRY_SBTYP_CD']=='01') &
 ↪  (pos2019['PRVDR_CTGRY_CD']=='01')]
pos2019_sth.shape
# add year column
pos2019_sth['year'] = 2019
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/2628018739.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pos2017_sth['year'] = 2017
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/2628018739.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pos2018_sth['year'] = 2018
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/2628018739.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pos2019_sth['year'] = 2019
```

```
# append all four years
combined_pos = pd.concat([pos2016_sth, pos2017_sth, pos2018_sth,
 ↪  pos2019_sth], ignore_index=True)
```

```
# create dataframe with number of observations per year
pos_yearly_observations = combined_pos.groupby('year').size().reset_index()
# rename number of observations column
pos_yearly_observations = pos_yearly_observations.rename(columns={0:
 ↪  'hospitals'})
```

```
# create plot
pos_yearly_observations_chart = alt.Chart(pos_yearly_observations,
 ↪  title='Number of Short Term Hospitals per Year from
 ↪  2016-2019').mark_bar().encode(
    alt.X('year:O', axis=alt.Axis(
        title='Year', labelAngle=0)),
    alt.Y('hospitals:Q', axis=alt.Axis(
        title='Number of Short Term Hospitals'
    ))
    ).properties(
        width=200
    )
#
 ↪  https://altair-viz.github.io/gallery/scatter_with_labels.html#gallery-scatter-with-labels
text =
 ↪  pos_yearly_observations_chart.mark_text(baseline='bottom').encode(text='hospitals')

pos_yearly_observations_chart + text
```

```
alt.LayerChart(...)
```

4. a.

```
# ChatGPT Question: how to find the number of unique values in a column

# Find unique number of short term hospitals per year
unique_pos_2016 = pos2016_sth['PRVDR_NUM'].nunique()
unique_pos_2017 = pos2017_sth['PRVDR_NUM'].nunique()
unique_pos_2018 = pos2018_sth['PRVDR_NUM'].nunique()
unique_pos_2019 = pos2019_sth['PRVDR_NUM'].nunique()

# Create dataframe with years and the number of unique hospitals per year)
unique_pos_yearly = pd.DataFrame({
    'year': [2016, 2017, 2018, 2019],
    'unique_pos_count': [unique_pos_2016, unique_pos_2017, unique_pos_2018,
     ↪  unique_pos_2019]
})

# Create unique hospitals plot
unique_pos_yearly_chart = alt.Chart(unique_pos_yearly, title='Number of
 ↪  Unique Short Term Hospitals per Year from
 ↪  2016-2019').mark_bar(size=30).encode(
```

```
    alt.X('year:O', axis=alt.Axis(
        title='Year', labelAngle=0)),
    alt.Y('unique_pos_count:Q', axis=alt.Axis(
        title='Number of Unique Short Term Hospitals'
    ))
    ).properties(
        width=300
    )
#
 ↪  https://altair-viz.github.io/gallery/scatter_with_labels.html#gallery-scatter-with-labels
text =
 ↪  unique_pos_yearly_chart.mark_text(baseline='bottom').encode(text='unique_pos_count:Q')

unique_pos_yearly_chart + text
```

```
alt.LayerChart(...)
```

b. There are the same number of hospitals even when you edit for unique CMS certification numbers. This tells us that the data entered is accurate and each hospital only has one record per year.

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

1.

```
# filter 2016 data for only active hospitals
active_2016 = pos2016_sth[pos2016_sth['PGM_TRMNTN_CD']=='00']

# filter 2016 data for only the information needed for the question
active_2016_list = active_2016[['PGM_TRMNTN_CD', 'PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD']]

# ChatGPT Question: how to filter dataset based on value in another dataset
# create dataframes with all the active 2016 hospitals
# 2017
active_2017 =
 ↪  pos2017_sth[pos2017_sth['PRVDR_NUM'].isin(active_2016_list['PRVDR_NUM'])]
active_2017_list = active_2017[['PGM_TRMNTN_CD', 'PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD', 'year']]

# 2018
```

```
active_2018 =
 ↪  pos2018_sth[pos2018_sth['PRVDR_NUM'].isin(active_2016_list['PRVDR_NUM'])]
active_2018_list = active_2018[['PGM_TRMNTN_CD', 'PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD', 'year']]

# 2019
active_2019 =
 ↪  pos2019_sth[pos2019_sth['PRVDR_NUM'].isin(active_2016_list['PRVDR_NUM'])]
active_2019_list = active_2019[['PGM_TRMNTN_CD', 'PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD', 'year']]

# create a list to iterate over
active_list = [(2017, active_2017_list), (2018, active_2018_list), (2019,
 ↪  active_2019_list)]

# ChatGPT Question: how to make a function to find values that are not 00 in
 ↪  three dataframes
# create function
def find_inactives(active_2016_list, active_list, PGM_TRMNTN_CD='00'):
    inactive_facilities_list = []
    for year, df, in active_list:
        merge_active = active_2016_list[['PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD']].merge(
            df[['PRVDR_NUM', 'PGM_TRMNTN_CD']], on='PRVDR_NUM', how='left'
        )
        inactives = merge_active[merge_active['PGM_TRMNTN_CD'] !=
 ↪  PGM_TRMNTN_CD]
        inactives['year'] = year
        inactive_facilities_list.append(inactives[['PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD', 'year']])
    inactive_facilities =
 ↪  pd.concat(inactive_facilities_list).drop_duplicates(subset='PRVDR_NUM')
    return inactive_facilities

inactive_facilities = find_inactives(active_2016_list, active_list,
 ↪  PGM_TRMNTN_CD='00')
print(inactive_facilities)

# How many hospitals are there that fit this definition?
inactive_facilities.shape
```

```
     PRVDR_NUM                                          FAC_NAME ZIP_CD
     year
93      030001                             ABRAZO MARYVALE CAMPUS  85031
2017
299     050196          ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230
2017
381     050435                        FALLBROOK HOSPITAL DISTRICT  92028
2017
528     060036            ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050
2017
529     060043                            KEEFE MEMORIAL HOSPITAL  80810
2017
...        ...                                               ...    ...
...
3396    670083                             TEXAS GENERAL HOSPITAL  75051
2019
3398    670087  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613
2019
3404    670094           LITTLE RIVER HEALTHCARE CAMERON HOSPITAL  76520
2019
3407    670097                     BAYLOR EMERGENCY MEDICAL CENTER  75087
2019
3421    670117                   TEXAS GENERAL HOSPITAL- VZRMC LP  75140
2019

[174 rows x 4 columns]

/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/1829816529.py:33:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/1829816529.py:33:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/1829816529.py:33:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year

(174, 4)
```

2.

```
# sort hospitals by name
sorted_inactive_facilities =
 ↪  inactive_facilities.sort_values(by='FAC_NAME')[['FAC_NAME', 'year']]

# show first 10
sorted_inactive_facilities.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.

```
# fix function so that it removes codes 00 and 01
def find_closures(active_2016_list, active_list, closed_codes=['02', '03',
 ↪  '04', '07', '05', '06']):
    inactive_facilities_list = []
    for year, df, in active_list:
        merge_active = active_2016_list[['PRVDR_NUM', 'FAC_NAME',
 ↪  'ZIP_CD']].merge(
```

```
            df[['PRVDR_NUM', 'PGM_TRMNTN_CD']], on='PRVDR_NUM', how='left'
        )
        inactives =
↪   merge_active[merge_active['PGM_TRMNTN_CD'].isin(closed_codes)]
        inactives['year'] = year
        inactive_facilities_list.append(inactives[['PRVDR_NUM', 'FAC_NAME',
↪   'ZIP_CD', 'year']])
    inactive_facilities =
↪   pd.concat(inactive_facilities_list).drop_duplicates(subset='PRVDR_NUM')
    return inactive_facilities

closed_facilities = find_closures(active_2016_list, active_list,
↪   closed_codes=['02', '03', '04', '07', '05', '06'])
print(closed_facilities)

closed_facilities.shape
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	year
528	060036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050	2017
529	060043	KEEFE MEMORIAL HOSPITAL	80810	2017
1263	180153	CONTINUECARE HOSPITAL AT BAPTIST HEALTH PADUCAH	42003	2017
1801	290006	BANNER CHURCHILL COMMUNITY HOSPITAL	89406	2017
1908	320013	HOLY CROSS HOSPITAL A DIV OF TAOS HEALTH SYSTEMS	87571	2017
2373	370138	MEMORIAL HOSPITAL OF TEXAS COUNTY AUTHORITY	73942	2017
2601	400009	SANTA ROSA CLINIC	785	2017
2721	430081	PHS INDIAN HOSPITAL AT PINE RIDGE	57770	2017
2892	450178	PECOS COUNTY MEMORIAL HOSPITAL	79735	2017
2920	450293	FRIO REGIONAL HOSPITAL	78061	2017
2955	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662	2017
2979	450620	DIMMIT REGIONAL HOSPITAL	78834	2017
3096	460033	GARFIELD MEMORIAL HOSPITAL	84759	2017
3387	670072	HOPEBRIDGE HOSPITAL	77035	2017
3416	670111	CONTINUECARE HOSPITAL AT MEDICAL CENTER ODESSA	79761	2017
107	030033	BANNER PAYSON MEDICAL CENTER	85541	2018
517	060016	CENTURA HEALTH-ST THOMAS MORE HOSPITAL	81212	2018
1759	260228	MERCY MCCUNE BROOKS HOSPITAL	64836	2018
1955	330053	MEDINA MEMORIAL HOSPITAL	14103	2018
2063	330268	COBLESKILL REGIONAL HOSPITAL	12043	2018
2312	360274	MEDICAL CENTER AT ELIZABETH PLACE	45417	2018
2359	370080	SHARE MEDICAL CENTER	73717	2018
2407	370233	PINNACLE SPECIALTY HOSPITAL	74137	2018
2808	440181	BOLIVAR GENERAL HOSPITAL	38008	2018

```
2825    440232             HOUSTON COUNTY COMMUNITY HOSPITAL  37061  2018
15      010032                              WEDOWEE HOSPITAL  36278  2019
480     050751                  MIRACLE MILE MEDICAL CENTER  90036  2019
503     050781                   SONOMA WEST MEDICAL CENTER  95472  2019
899     130067                       IDAHO DOCTORS HOSPITAL  83221  2019
1213    180021                 PINEVILLE COMMUNITY HOSPITAL  40977  2019
1230    180053                      FLEMING COUNTY HOSPITAL  41041  2019
1496    230040                      SPECTRUM HEALTH PENNOCK  49058  2019
1656    250079         SHARKEY ISSAQUENA COMMUNITY HOSPITAL  39159  2019
1773    270089                       BIG SKY MEDICAL CENTER  59716  2019
2351    370048                  MCCURTAIN MEMORIAL HOSPITAL  74745  2019
2779    440083               TENNOVA HEALTHCARE - JAMESTOWN  38556  2019
2895    450192                       HILL REGIONAL HOSPITAL  76645  2019
3020    450746                         KNOX COUNTY HOSPITAL  79529  2019
3023    450754                    HAMILTON GENERAL HOSPITAL  76531  2019
3269    510082         SUMMERSVILLE REGIONAL MEDICAL CENTER  26651  2019
3396    670083                       TEXAS GENERAL HOSPITAL  75051  2019

/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/3740686694.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/3740686694.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/3740686694.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  inactives['year'] = year
```

(41, 4)

a. Since there were 174 hospitals in the first list and 41 in the second, we can assume that 133 hospitals were actually mergers/acquisitions

b. After the correction, there are only 41 hospitals left.

c.

```
# sort hospitals by name
sorted_closed_facilities = closed_facilities.sort_values(by='FAC_NAME')

# show first 10
sorted_closed_facilities.head(10)
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	yea
2955	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662	20
528	060036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050	20
1801	290006	BANNER CHURCHILL COMMUNITY HOSPITAL	89406	20
107	030033	BANNER PAYSON MEDICAL CENTER	85541	20
1773	270089	BIG SKY MEDICAL CENTER	59716	20
2808	440181	BOLIVAR GENERAL HOSPITAL	38008	20
517	060016	CENTURA HEALTH-ST THOMAS MORE HOSPITAL	81212	20
2063	330268	COBLESKILL REGIONAL HOSPITAL	12043	20
1263	180153	CONTINUECARE HOSPITAL AT BAPTIST HEALTH PADUCAH	42003	20
3416	670111	CONTINUECARE HOSPITAL AT MEDICAL CENTER ODESSA	79761	20

Download Census zip code shapefile (10 pt)

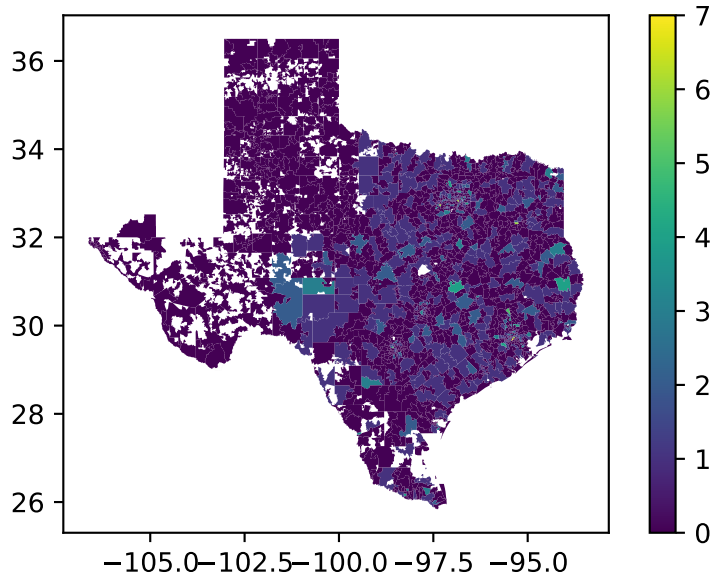
1. a. In the shapfile folder, there is a dBase database file, a projection definition file, a shapefile, a shape index format file, and an extensible markup language file. The shapefile stores the geometric features. The shape index file has the positional index. The database file has the attribute information. The projection defintion file describes that coordinate reference system.

   b. The .dbf file is 6.4 MB, the .prj file is 165 B, the .shp file is 837.5 MB, the .shx file is 265 KB, and the .xml file is 16 KB. The .shp is the biggest file.

2.

```
# read in shapefile
zip_shapefile = gpd.read_file("gz_2010_us_860_00_500k.zip")
```

```
# filter shapefile for Texas zipcodes
texas_shapefile =
 ↪  zip_shapefile[zip_shapefile['ZCTA5'].str.startswith(('733','75', '76',
 ↪  '77', '78', '79'))]

# rename zipcode column to match pos file
texas_shapefile = texas_shapefile.rename(columns={'ZCTA5': 'ZIP_CD'})

# filter 2016 hospital for Texas zipcodes
pos2016_tx = pos2016_sth[pos2016_sth['ZIP_CD'].str.startswith(('733','75',
 ↪  '76', '77', '78'))]

# find number of hospitals per zipcode
hospital_count =
 ↪  pos2016_tx.groupby('ZIP_CD').size().reset_index(name='hospital_count')

# merge files
hospital_count_map = texas_shapefile.merge(hospital_count, on='ZIP_CD',
 ↪  how='left')

# preview data
hospital_count_map

# fill NAs with 0
hospital_count_map['hospital_count'] =
 ↪  hospital_count_map['hospital_count'].fillna(0)

hospital_count_map.plot(column='hospital_count', legend=True)
```



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

1.

```
# create centroid column
zip_shapefile['centroid'] = zip_shapefile['geometry'].centroid

# rename zipcode column
zip_shapefile = zip_shapefile.rename(columns={'ZCTA5': 'ZIP_CD'})

# create geodataframe
zips_all_centroids = gpd.GeoDataFrame(zip_shapefile[['ZIP_CD', 'centroid']],
 ↪  geometry='centroid')

# find dimensions
zips_all_centroids.shape
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/3749194253.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.

  zip_shapefile['centroid'] = zip_shapefile['geometry'].centroid
```

```
(33120, 2)
```

The dimensions of the GeoDataFrame are (33120, 2). The ZIP_CD column is all the zipcodes in the census data. The centroid column is the arithmetic mean position of all the zipcodes points in the polygon.

2.

```
# create texas centroid geodataframe
zips_texas_centroids =
 ↪  zip_shapefile[zip_shapefile['ZIP_CD'].str.startswith(('733','75', '76',
 ↪  '77', '78', '79'))]

# Unique zipcodes in Texas subset
print(zips_texas_centroids['ZIP_CD'].nunique())

# create texas and border states centroid dataframe
zips_texas_borderstates_centroids =
 ↪  zip_shapefile[zip_shapefile['ZIP_CD'].str.startswith(('7', '87', '88'))]

# Unique zipcodes in Texas and Border States subset
print(zips_texas_borderstates_centroids['ZIP_CD'].nunique())
```

```
1935
4057
```

3.

```
# filter 2016 hospitals Texas in texas for only active facilities
active_2016_tx = pos2016_tx[pos2016_tx['PGM_TRMNTN_CD']=='00']

# find number of active hospitals per zipcode
active_hospital_count =
 ↪  active_2016_tx.groupby('ZIP_CD').size().reset_index(name='hospital_count')

zips_withhospital_centroids = gpd.GeoDataFrame(
    active_hospital_count.merge(zips_texas_centroids, on='ZIP_CD',
 ↪  how='left'),
    geometry='geometry')
```

I did a left merge and merged the texas zipcode centroids on the list of Texas zipcodes with at least one hospital. This resulted in only keeping the zipcodes that the latter list identifies as having a hospital and added in the centroid information for each of those zipcodes.

4. a.

```
# ChatGPT Question: how to find the distance for 10 zipcodes to the nearest
 ↪  zipcode with a hospital

# get sample Texas zipcodes
sampled_zips_texas = zips_texas_centroids.sample(n=10, random_state=1)

# create timer
start_time_sample = time.time()

# run function with the sample zipcodes
sample_distances = sampled_zips_texas.geometry.apply(
    lambda x: zips_withhospital_centroids.geometry.distance(x).min()
)

#end timer
end_time_sample = time.time()

# time taken for sample
sample_duration = end_time_sample - start_time_sample
print("Time taken for sampled calculation:", sample_duration, "seconds")

# find time estimation for full sample
print("Estimated time for ful calculation:",
 ↪  ((sample_duration/10)*len(zips_texas_centroids))/60, "minutes")
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/2725698882.py:11:
UserWarning: Geometry is in a geographic CRS. Results from 'distance' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a
projected CRS before this operation.

  lambda x: zips_withhospital_centroids.geometry.distance(x).min()

Time taken for sampled calculation: 4.400004148483276 seconds
Estimated time for ful calculation: 14.190013378858566 minutes
```

b.

```
# ChatGPT Question: how to find the distance for 10 zipcodes to the nearest
 ↪  zipcode with a hospital

# create timer
```

```
start_time_full = time.time()

# run function with the sample zipcodes
zips_texas_centroids['distances'] = zips_texas_centroids.geometry.apply(
    lambda x: zips_withhospital_centroids.geometry.distance(x).min()
)

#end timer
end_time_full = time.time()

# time taken for sample
full_duration = end_time_full - start_time_full
print("Time taken for sampled calculation:", (full_duration/60), "minutes")


# How close was the actual duration to my estimated duration?
seconds_off = full_duration -
 ↪  ((sample_duration/10)*len(zips_texas_centroids))
print("I was off by:", seconds_off, 'seconds')
```

```
/var/folders/9h/vrgs4_l97tl4bm5k_7x33hb40000gn/T/ipykernel_18695/4074092989.py:8:
UserWarning: Geometry is in a geographic CRS. Results from 'distance' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a
projected CRS before this operation.

  lambda x: zips_withhospital_centroids.geometry.distance(x).min()

Time taken for sampled calculation: 16.289838965733846 minutes
I was off by: 125.98953521251678 seconds

/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/geopandas/ge
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  super().__setitem__(key, value)

c.
```

```
# read in prj file
with open('gz_2010_us_860_00_500k.prj', 'r') as file:
    prj_info = file.read()

print(prj_info)

# correct crs for measuring distance https://epsg.io/5070
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=5070)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=5070)

# convert meters to miles
# ChatGPT Question: How do I convert {prj_info} to units in miles
lat_conversion_factor = 69  # miles per degree of latitude
lon_conversion_factor = 59.2  # miles per degree of longitude

# Average latitude for Texas
avg_latitude = 31
# Apply conversion
# For simplicity, we will assume a simple average here:
import numpy as np
zips_texas_centroids['distances_miles'] = zips_texas_centroids['distances'] *
 ↪  np.sqrt(lat_conversion_factor**2 + lon_conversion_factor**2)
```

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,29
```

The unit is measured in degrees.

5. a.

```
#zips_texas_centroids['distance_hospital'] =
 ↪  zips_texas_centroids.geometry.apply(
 #   lambda x: active_2016_tx.geometry.distance(x).min()
  #  )
```

The unit is in degrees b.

c.

Effects of closures on access in Texas (15 pts)

1.
2.
3.
4.

Reflecting on the exercise (10 pts)

Partner 1: An issue with the "first pass" method is that, according to the data dictionary, 01 is for a merge or a closed hospital, so getting rid of 01 does not guarantee we are getting rid of only merged hospitals. A better way would potentially be to use strings to find similar hospital names in the same zipcode and look at how that changes across the years.

Partner 2: I think the way we find affected zipcodes is effective. Since we calculate distance to nearest hospital, we get an accurate sense of how difficult healthcare access is for different zipcodes. Looking at quality of hospitals might give a better sense of inequality in the healthcare system. A zipcode could have a close hospital that is not great and they are still not as well off as others.