

PSET4

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): Cedar Liu, cedarliu
 - Partner 2 (name and cnet ID): Shuping Wu, shupingw
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: **_CL_** **_SHW_**
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: **_1_**
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)

1. & 2.

```
import pandas as pd

# 1: Load the 2016 POS Data
pos2016 = pd.read_csv('data/Q4_2016.csv')
pos2017 = pd.read_csv('data/Q4_2017.csv')
pos2018 = pd.read_csv('data/Q4_2018.csv')
pos2019 = pd.read_csv('data/Q4_2019.csv')
pos2016['Year'] = '2016'
pos2017['Year'] = '2017'
pos2018['Year'] = '2018'
pos2019['Year'] = '2019'
# 2: Filter for Short-Term Hospitals
short_term_hospitals_6 = pos2016[(pos2016['PRVDR_CTGRY_CD'] == 1.0) &
    (pos2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]
num_hospitals_6 = short_term_hospitals_6.shape[0]

print(f"Number of short-term hospitals reported in 2016 data:
    {num_hospitals_6}")

unique_hospitals = short_term_hospitals_6.drop_duplicates(subset='FAC_NAME')
print(f"Number of unique hospitals: {len(unique_hospitals)}")
```

Number of short-term hospitals reported in 2016 data: 7245

Number of unique hospitals: 6770

For part I: Download and explore the Provider of Services (POS) file, we only need the variables including PRVDR_CTGRY_SBTYP_CD, PRVDR_CTGRY_CD, FAC_NAME, PRVDR_NUM.

In 2016, the United States had 4,840 community hospitals. These are defined as non-federal, short-term general and specialty hospitals whose facilities and services are

available to the public. The source is from AHA (American Hospital Association): <https://www.aha.org/system/files/2018-05/2018-chartbook-table-2-1.pdf>.

Why it differs: pos2016 from CMS (Centers for Medicare & Medicaid Services) might include facilities beyond the typical definition of short-term hospitals, such as specialty clinics or outpatient centers, leading to an inflated count. Also, the dataset is from 2016, it may not reflect hospital closures, mergers, or reclassifications that have occurred.

3.

```
short_term_hospitals_7 = pos2017[(pos2017['PRVDR_CTGRY_CD'] == 1) &
    (pos2017['PRVDR_CTGRY_SBTYP_CD'] == 1)]
num_hospitals_7 = short_term_hospitals_7.shape[0]

print(f"Number of short-term hospitals reported in 2017 data:
    {num_hospitals_7}")

short_term_hospitals_8 = pos2018[(pos2018['PRVDR_CTGRY_CD'] == 1) &
    (pos2018['PRVDR_CTGRY_SBTYP_CD'] == 1)]
num_hospitals_8 = short_term_hospitals_8.shape[0]

print(f"Number of short-term hospitals reported in 2018 data:
    {num_hospitals_8}")

short_term_hospitals_9 = pos2019[(pos2019['PRVDR_CTGRY_CD'] == 1) &
    (pos2019['PRVDR_CTGRY_SBTYP_CD'] == 1)]
num_hospitals_9 = short_term_hospitals_9.shape[0]

print(f"Number of short-term hospitals reported in 2019 data:
    {num_hospitals_9}")
```

Number of short-term hospitals reported in 2017 data: 7260
Number of short-term hospitals reported in 2018 data: 7277
Number of short-term hospitals reported in 2019 data: 7303

4.

```
import altair as alt

years = ['2016', '2017', '2018', '2019']
dataframes = []

for year in years:
```

```

df = eval(f'pos{year}')
df['Year'] = year
dataframes.append(df)

all_data = pd.concat(dataframes)

short_term_hospitals = all_data[(all_data['PRVDR_CTGRY_CD'] == 1) &
                                (all_data['PRVDR_CTGRY_SBTYP_CD'] == 1)]

total_hospitals_per_year =
    short_term_hospitals['Year'].value_counts().reset_index()
total_hospitals_per_year.columns = ['Year', 'Total Hospitals']
chart_unique_hospitals =
    alt.Chart(total_hospitals_per_year).mark_bar().encode(
        x=alt.X('Year:O', title='Year'),
        y=alt.Y('Total Hospitals', title='Number of Total Hospitals',),
        color=alt.value('blue')
    ).properties(
        title='Number of Total Short-Term Hospitals by Year'
    ).transform_calculate(
        adjusted_value='datum["Total Hospitals"]-7000'
    ).encode(
        y=alt.Y('adjusted_value:Q', title='Number of Total Hospitals (Adjusted)'),
        scale=alt.Scale(domain=[0, 350]), axis=alt.Axis(labelExpr='datum.value + 7000'))
)

chart_unique_hospitals.display()

unique_hospitals_per_year =
    short_term_hospitals.groupby('Year')['PRVDR_NUM'].nunique().reset_index()
unique_hospitals_per_year.columns = ['Year', 'Unique Hospitals']

chart_unique_hospitals =
    alt.Chart(unique_hospitals_per_year).mark_bar().encode(
        x=alt.X('Year:O', title='Year'),
        y=alt.Y('Unique Hospitals', title='Number of Unique Hospitals',),
        color=alt.value('skyblue')
    ).properties(
        title='Number of Unique Short-Term Hospitals by Year'
    ).transform_calculate(
        adjusted_value='datum["Unique Hospitals"]-7000'
)

```

```

).encode(
    y=alt.Y('adjusted_value:Q', title='Number of Unique Hospitals
        (Adjusted)', scale=alt.Scale(domain=[0, 350]),
    axis=alt.Axis(labelExpr='datum.value + 7000'))
)

chart_unique_hospitals.display()

```

D:\Anaconda Installing\Lib\site-packages\altair\utils\core.py:395:
FutureWarning:

the convert_dtype parameter is deprecated and will be removed in a future
version. Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.

alt.Chart(...)

D:\Anaconda Installing\Lib\site-packages\altair\utils\core.py:395:
FutureWarning:

the convert_dtype parameter is deprecated and will be removed in a future
version. Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.

alt.Chart(...)

The number of observations equals the number of unique hospitals each year, it implies that each hospital is represented by a single unique entry in each dataset, with no repeated entries per hospital within a given year, confirming a straightforward data structure.

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

1.

```

active_2016 = short_term_hospitals[short_term_hospitals['Year'] == '2016']
active_2016_hospitals = active_2016[['PRVDR_NUM', 'FAC_NAME',
    'ZIP_CD']].drop_duplicates()

suspected_closures = []

for year, df in zip(['2017', '2018', '2019'], [pos2017, pos2018, pos2019]):
    year_hospitals = df[(df['PRVDR_CTGRY_CD'] == 1) &
    (df['PRVDR_CTCR_Y_CD'] == 1)]

```

```

closed_hospitals = active_2016_hospitals[
    ~active_2016_hospitals['PRVDR_NUM'].isin(year_hospitals[year_hospitals['PGM_TRMN'
    == 0]['PRVDR_NUM'])
]

closed_hospitals = closed_hospitals.assign(Suspected_Closure_Year=year)
suspected_closures.append(closed_hospitals)

suspected_closures_df =
    pd.concat(suspected_closures).drop_duplicates(['PRVDR_NUM'])

closures_count = suspected_closures_df.shape[0]
print(f"Number of suspected closures: {closures_count}")
print(suspected_closures_df[['FAC_NAME', 'ZIP_CD',
    'Suspected_Closure_Year']])

```

Number of suspected closures: 3996

	FAC_NAME	ZIP_CD	\
1	NORTH JACKSON HOSPITAL	35740.0	
6	HARTSELLE MEDICAL CENTER	35640.0	
7	MARSHALL MEDICAL CENTER NORTH	35976.0	
10	SOUTHWEST ALABAMA MEDICAL CENTER	36784.0	
12	FAIRVIEW MEDICAL CTR	36108.0	
...
133498	TEXAS GENERAL HOSPITAL	75051.0	
133501	BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER	78613.0	
133508	LITTLE RIVER HEALTHCARE CAMERON HOSPITAL	76520.0	
133511	BAYLOR EMERGENCY MEDICAL CENTER	75087.0	
133529	TEXAS GENERAL HOSPITAL- VZRMCP LP	75140.0	

	Suspected_Closure_Year
1	2017
6	2017
7	2017
10	2017
12	2017
...	...
133498	2019
133501	2019
133508	2019
133511	2019

133529 2019

[3996 rows x 3 columns]

2.

```
sorted_closures_top10 =
    ↪ suspected_closures_df.sort_values('FAC_NAME').head(10)

print(sorted_closures_top10[['FAC_NAME', 'Suspected_Closure_Year']])
```

	FAC_NAME Suspected_Closure_Year
106772	(CLOSED) BAPTIST HICKMAN COMMUNITY HOSPITAL
2017	
106779	(CLOSED) BAPTIST HOSPITAL OF ROANE COUNTY
2017	
106844	(CLOSED) BAPTIST MEMORIAL HOSPITAL LAUDERDALE
2017	
106924	(CLOSED) BAPTIST MEMORIAL HOSPITAL-COLLIERVILLE
2017	
106875	(CLOSED) BLEDSOE COUNTY HOSPITAL
2017	
106864	(CLOSED) CAMDEN GENERAL HOSPITAL
2017	
106914	(CLOSED) COLUMBIA CHEATHAM MEDICAL CENTER
2017	
106831	(CLOSED) COLUMBIA WHITWELL MEDICAL CENTER INC
2017	
106891	(CLOSED) COLUMBIA EAST RIDGE HOSP
2017	
106795	(CLOSED) COPPER BASIN MEDICAL CENTER
2017	

3. a. Among the suspected closures, 995 hospitals fit this definition of potentially being a merger/acquisition.
b. After correcting for this, there are 3001 hospitals we've got left.

```
suspected_closures_df =
    ↪ pd.concat(suspected_closures).drop_duplicates(['PRVDR_NUM'])
suspected_closures_df
active_hospitals_zip_year =
    ↪ short_term_hospitals[short_term_hospitals['PGM_TRMNTN_CD'] ==
    ↪ 0].groupby(['ZIP_CD', 'Year']).size().unstack(fill_value=0)
```

```

merger_candidates = []

for _, row in suspected_closures_df.iterrows():
    zip_code = row['ZIP_CD']
    closure_year = row['Suspected_Closure_Year']

    if closure_year in ['2016', '2017', '2018']:
        next_year = str(int(closure_year) + 1)
        if zip_code in active_hospitals_zip_year.index:
            if active_hospitals_zip_year.loc[zip_code, next_year] >=
                active_hospitals_zip_year.loc[zip_code, closure_year]:
                merger_candidates.append(row)

merger_candidates_df = pd.DataFrame(merger_candidates)

merger_count = merger_candidates_df.shape[0]
print(f"Number of potential mergers/acquisitions: {merger_count}")
print(merger_candidates_df[['FAC_NAME', 'ZIP_CD', 'Suspected_Closure_Year']])

```

	FAC_NAME	ZIP_CD	Suspected_Closure_Year
21 2017	COLUMBIA REGIONAL MEDICAL CENTER	36420.0	
25 2017	PHENIX REGIONAL HOSPITAL	36867.0	
35 2017	CHILTON MEDICAL CENTER	35045.0	
44 2017	ATMORE COMMUNITY HOSPITAL	36502.0	
45 2017	DECATUR MORGAN HOSPITAL-PARKWAY CAMPUS	35601.0	
...
...
126562 2018	WHEATON FRANCISCAN HEALTHCARE FRANKLIN	53132.0	
129056 2018	MOUNTAIN VIEW REGIONAL HOSPITAL	82605.0	
133442 2018	NORTH CYPRESS MEDICAL CENTER	77429.0	

```
133473          EMERUS HOSPITAL 77479.0
2018
133510  BAY AREA REGIONAL MEDICAL CENTER, LLC 77598.0
2018
```

[995 rows x 3 columns]

c.

```
corrected_closures_df =
    ↪ suspected_closures_df[~suspected_closures_df['PRVDR_NUM'].isin(merger_candidates_df['PRVDR_NUM'])]

corrected_closures_count = corrected_closures_df.shape[0]

sorted_corrected_closures_top10 =
    ↪ corrected_closures_df.sort_values('FAC_NAME').head(10)

print(f"Number of corrected hospital closures: {corrected_closures_count}")
print(sorted_corrected_closures_top10[['FAC_NAME', 'ZIP_CD',
    ↪ 'Suspected_Closure_Year']])
```

Number of corrected hospital closures: 3001

		FAC_NAME	ZIP_CD	\
106772	(CLOSED) BAPTIST HICKMAN COMMUNITY HOSPITAL		37033.0	
106779	(CLOSED) BAPTIST HOSPITAL OF ROANE COUNTY		37854.0	
106844	(CLOSED) BAPTIST MEMORIAL HOSPITAL LAUDERDALE		38063.0	
106924	(CLOSED) BAPTIST MEMORIAL HOSPITAL-COLLIERVILLE		38017.0	
106875	(CLOSED) BLEDSOE COUNTY HOSPITAL		37367.0	
106864	(CLOSED) CAMDEN GENERAL HOSPITAL		38320.0	
106914	(CLOSED) COLUMBIA CHEATHAM MEDICAL CENTER		37015.0	
106831	(CLOSED) COLUMBIA WHITWELL MEDICAL CENTER INC		37397.0	
106891	(CLOSED) COLUMBIA EAST RIDGE HOSP		37412.0	
106795	(CLOSED) COPPER BASIN MEDICAL CENTER		37317.0	

	Suspected_Closure_Year
106772	2017
106779	2017
106844	2017
106924	2017
106875	2017
106864	2017
106914	2017
106831	2017

106891	2017
106795	2017

Download Census zip code shapefile (10 pt)

1.
 - a. -rw-r-r- 1 15535 197609 6.2M Oct 30 23:58 gz_2010_us_860_00_500k.dbf
 - b. -rw-r-r- 1 15535 197609 165 Oct 30 23:58 gz_2010_us_860_00_500k.prj
 - c. -rw-r-r- 1 15535 197609 799M Oct 30 23:58 gz_2010_us_860_00_500k.shp
 - d. -rw-r-r- 1 15535 197609 259K Oct 30 23:58 gz_2010_us_860_00_500k.shx
 - e. -rw-r-r- 1 15535 197609 16K Oct 30 23:58 gz_2010_us_860_00_500k.xml
- a. .dbf (Attribute Data File): Contains attribute data for each shape (e.g., names, values, categories), stored in a tabular format. This file links each shape's attributes, providing additional context to the geometries stored in the .shp file. Size: 6.2 MB
- b. .prj (Projection File): Stores the coordinate system and projection information. This file defines how the geometrical data is projected on the Earth's surface, essential for accurate spatial analysis and alignment with other datasets. Size: 165 bytes
- c. .shp (Shape File): Contains the geometry of the features, such as points, lines, or polygons. This is the core file of the shapefile dataset and is necessary for drawing the shapes on a map. Size: 799 MB
- d. .shx (Shape Index File): Stores the positional index of the feature geometry to allow quick access to the geometrical data in the .shp file. It acts as a spatial index that accelerates data retrieval. Size: 259 KB
- e. .xml (Metadata File): Contains metadata about the shapefile, such as the data's source, description, and other attributes. Metadata provides context for the dataset's creation and use. Size: 16 KB

2.

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

# Load the shapefile
zip_shapefile = gpd.read_file('data/gz_2010_us_860_00_500k.shp')
```

```

# texas_choropleth

# Filter for Texas ZIP codes (first two digits between 75 and 79)
texas_zip_codes = zip_shapefile[zip_shapefile['ZCTA5'].str[:3].isin(['733']) |
    | zip_shapefile['ZCTA5'].str[:2].isin(['75', '76', '77', '78', '79'])]

# Filter and count hospitals per ZIP code in Texas
texas_hospitals = short_term_hospitals_6[
    short_term_hospitals_6['ZIP_CD'].astype(str).str[:3].isin(['733']) |
    short_term_hospitals_6['ZIP_CD'].astype(str).str[:2].isin(['75', '76',
    | '77', '78', '79'])]
]

hospitals_per_zip = texas_hospitals['ZIP_CD'].value_counts().reset_index()
hospitals_per_zip.columns = ['ZIP_CD', 'Hospital_Count']

# Merge hospital count with Texas ZIP shapefile
texas_zip_codes['ZIP_CD'] =
    |> texas_zip_codes['ZCTA5'].astype(int).astype(str).str.zfill(5)
hospitals_per_zip['ZIP_CD'] =
    |> hospitals_per_zip['ZIP_CD'].astype(int).astype(str).str.zfill(5)
texas_choropleth = texas_zip_codes.merge(hospitals_per_zip, on='ZIP_CD',
    |> how='left').fillna(0)

```

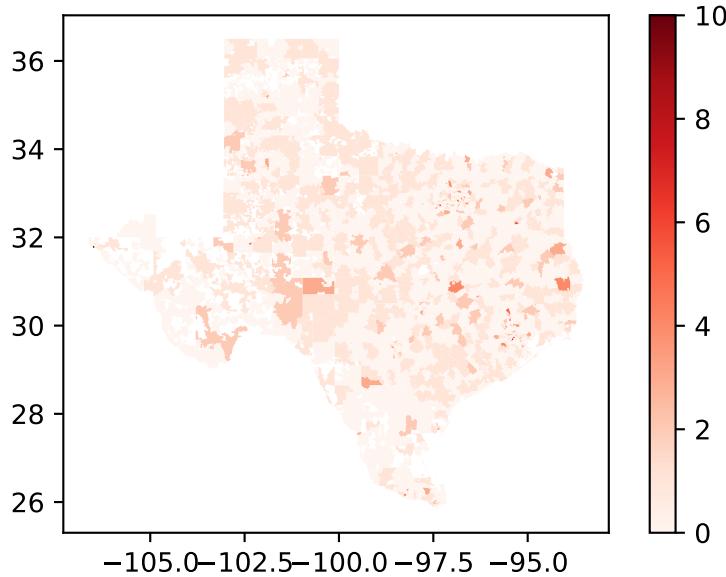
C:\Users\think
book\AppData\Roaming\Python\Python311\site-packages\geopandas\geodataframe.py:1819:
SettingWithCopyWarning:

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

```
texas_choropleth.plot(column="Hospital_Count", cmap='Reds', legend=True)
```



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

- Dimensions are:

```

zips_all_centroids = zip_shapefile.copy()
zips_all_centroids['geometry'] = zips_all_centroids.centroid

dimensions = zips_all_centroids.shape
columns_info = zips_all_centroids.columns

dimensions, columns_info

((33120, 6),
 Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry'],
      dtype='object'))

```

- (texas_zip_count, texas_borderstates_zip_count)

```

texas_prefixes = ['75', '76', '77', '78', '79', '733']
bordering_states_prefixes = texas_prefixes + ['73', '74', '70', '71', '87',
    ↪ '88']

zips_texas_centroids =
    ↪ zips_all_centroids[zips_all_centroids['ZCTA5'].str[:3].isin(['733'])] |
    ↪ zips_all_centroids['ZCTA5'].str[:2].isin(['75', '76', '77', '78', '79'])]

```

```

zips_texas_borderstates_centroids =
    zips_all_centroids[zips_all_centroids['ZCTA5'].str[:2].isin(bordering_states_prefixes)]

texas_zip_count = zips_texas_centroids['ZCTA5'].nunique()
texas_borderstates_zip_count =
    zips_texas_borderstates_centroids['ZCTA5'].nunique()

texas_zip_count, texas_borderstates_zip_count

```

(1935, 3596)

3. Merge on ZIP_CD, inner merge.

```

zips_texas_borderstates_centroids['ZIP_CD'] =
    zips_texas_borderstates_centroids['ZCTA5'].astype(int).astype(str).str.zfill(5)
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospitals_per_zip[hospitals_per_zip['Hospital_Count'] > 0],
    on='ZIP_CD',
    how='inner'
)
print(f"Number of ZIP codes with at least one hospital:
    {zips_withhospital_centroids.shape[0]}")
zips_withhospital_centroids

```

Number of ZIP codes with at least one hospital: 490

```

C:\Users\think
book\AppData\Roaming\Python\Python311\site-packages\geopandas\geodataframe.py:1819:
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

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry	ZIP_CD
0	8600000US78624	78624	78624	ZCTA5	708.041	POINT (-98.87707 30.2816)	78624
1	8600000US78626	78626	78626	ZCTA5	93.046	POINT (-97.59733 30.66535)	78626

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry	ZIP_C
2	8600000US78636	78636	78636	ZCTA5	349.689	POINT (-98.41885 30.30504)	78636
3	8600000US78640	78640	78640	ZCTA5	91.662	POINT (-97.82814 29.99496)	78640
4	8600000US78643	78643	78643	ZCTA5	704.140	POINT (-98.69472 30.69029)	78643
...
485	8600000US75035	75035	75035	ZCTA5	23.643	POINT (-96.77269 33.1553)	75035
486	8600000US78028	78028	78028	ZCTA5	250.675	POINT (-99.15819 30.03424)	78028
487	8600000US78222	78222	78222	ZCTA5	21.890	POINT (-98.37787 29.36619)	78222
488	8600000US78412	78412	78412	ZCTA5	8.798	POINT (-97.34297 27.70383)	78412
489	8600000US78586	78586	78586	ZCTA5	176.313	POINT (-97.6311 26.10507)	78586

4. a.

```
from shapely.geometry import Point
import time

# Select a subset of 10 ZIP codes for testing
zips_texas_centroids_subset = zips_texas_centroids.head(10)

# Start timer
start_time = time.time()

# Calculate distance to the nearest ZIP code with a hospital for each ZIP in
# the subset
zips_texas_centroids_subset['nearest_hospital_distance'] =
    zips_texas_centroids_subset.geometry.apply(
        lambda x: zips_withhospital_centroids.distance(x).min()
    )

# End timer and calculate duration
duration = time.time() - start_time
print(f"Time taken for 10 ZIP codes: {duration:.2f} seconds")
```

Time taken for 10 ZIP codes: 0.21 seconds

b.

```
# Start timer
start_time_full = time.time()

# Calculate distance to the nearest ZIP code with a hospital for each ZIP in
# the full dataset
```

```

zips_texas_centroids['nearest_hospital_distance'] =
    zips_texas_centroids.geometry.apply(
        lambda x: zips_withhospital_centroids.distance(x).min()
    )

# End timer and calculate full duration
duration_full = time.time() - start_time_full
print(f"Time taken for full calculation: {duration_full:.2f} seconds")

```

Time taken for full calculation: 36.95 seconds

c.

5. a. Warning says it is in a geographic CRS, specifically degree.
- b. This quite makes sense

```

zips_texas_centroids = zips_texas_centroids.to_crs(epsg=32614)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=32614)

# Calculate distance to the nearest ZIP code with a hospital for each ZIP in
# Texas
start_time = time.time()
zips_texas_centroids['nearest_hospital_distance_meters'] =
    zips_texas_centroids.geometry.apply(
        lambda x: zips_withhospital_centroids.distance(x).min()
    )
duration = time.time() - start_time
print(f"Time taken for full calculation: {duration:.2f} seconds")

# Convert distances from meters to miles
zips_texas_centroids['nearest_hospital_distance_miles'] =
    zips_texas_centroids['nearest_hospital_distance_meters'] * 0.000621371
zips_texas_centroids_distance_mean =
    zips_texas_centroids['nearest_hospital_distance_miles'].mean()
print(f"The average distance is: {zips_texas_centroids_distance_mean:.2f}
    miles")
# Verify results
print(zips_texas_centroids.head())

```

Time taken for full calculation: 0.39 seconds

The average distance is: 8.22 miles

GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	\

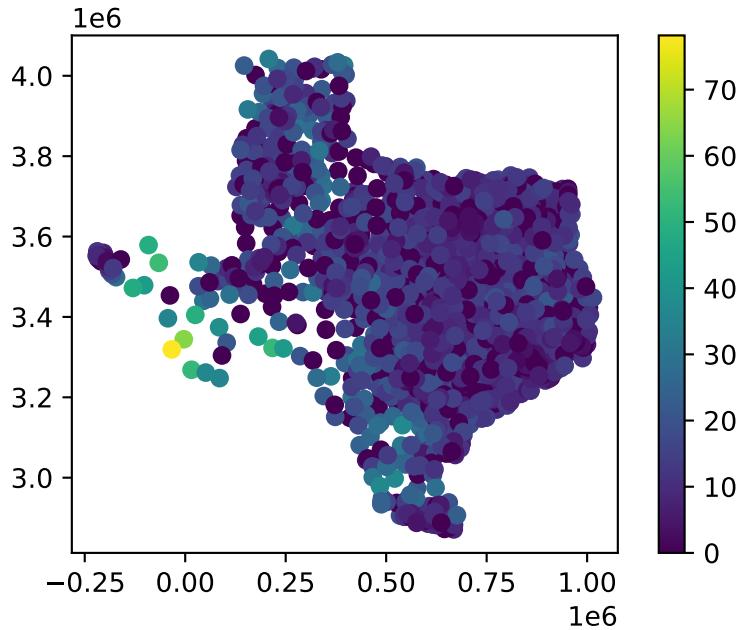
9207	8600000US78624	78624	78624	ZCTA5	708.041
9208	8600000US78626	78626	78626	ZCTA5	93.046
9209	8600000US78628	78628	78628	ZCTA5	73.382
9210	8600000US78631	78631	78631	ZCTA5	325.074
9211	8600000US78632	78632	78632	ZCTA5	96.278

	geometry	nearest_hospital_distance	\
9207	POINT (511822.321 3349995.734)	0.000000	
9208	POINT (634379.614 3393354.111)	0.000000	
9209	POINT (619673.979 3390490.352)	0.110844	
9210	POINT (470656.501 3356247.742)	0.337251	
9211	POINT (647977.513 3286115.301)	0.164115	

	nearest_hospital_distance_meters	nearest_hospital_distance_miles
9207	0.000000	0.000000
9208	0.000000	0.000000
9209	12197.888778	7.579414
9210	36489.177324	22.673317
9211	15883.323167	9.869436

c.

```
zips_texas_centroids.plot(column='nearest_hospital_distance_miles',
                           legend=True)
```



Effects of closures on access in Texas (15 pts)

1.

```
suspected_closures_df['ZIP_CD'] =
    suspected_closures_df['ZIP_CD'].astype(int).astype(str).str.zfill(5)

closure_table =
    suspected_closures_df[suspected_closures_df['ZIP_CD'].str[:3].isin(['733'])]
    | suspected_closures_df['ZIP_CD'].str[:2].isin(['75', '76', '77', '78',
    '79'])]

closure_table = closure_table['ZIP_CD'].value_counts().reset_index()
closure_table.columns = ['ZIP_CD', 'Total Closure']
closure_table
```

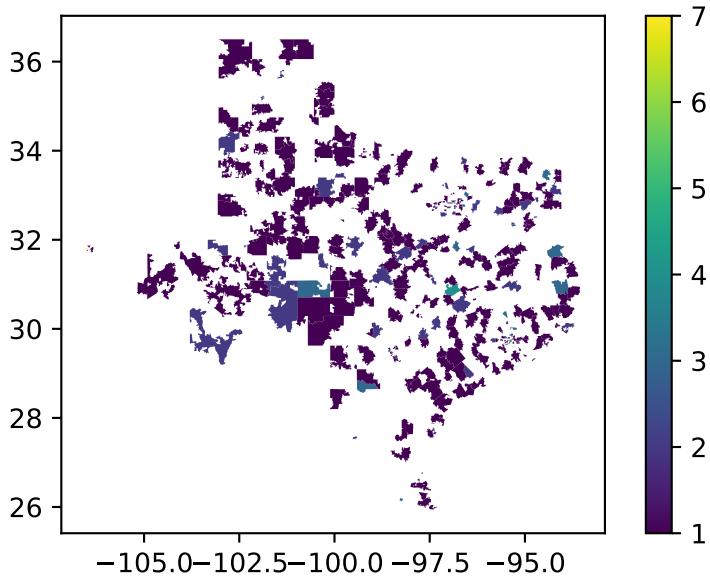
	ZIP_CD	Total Closure
0	79902	7
1	75235	5
2	77054	5
3	76520	4
4	75570	3

	ZIP_CD	Total Closure
...
338	79744	1
339	79546	1
340	77063	1
341	79901	1
342	75087	1

2.

```
closure_choropleth = texas_zip_codes.merge(closure_table, on='ZIP_CD',
                                         how='right')
```

```
closure_choropleth.plot(column = 'Total Closure', legend = True)
```



3.

```
closure_choropleth = closure_choropleth.to_crs(epsg=32614)

closure_choropleth['buffer_10_miles'] =
    closure_choropleth.geometry.buffer(16093.4)
```

```

# Create a GeoDataFrame containing just the buffered areas
buffers = closure_choropleth[['ZIP_CD', 'buffer_10_miles']]
buffers = buffers.set_geometry('buffer_10_miles')

# Perform a spatial join between the Texas ZIP codes and the buffers to find
# indirectly affected ZIP codes
indirectly_affected_zips = gpd.sjoin(zips_texas_centroids, buffers,
                                       how='inner', predicate='intersects')

# Count unique indirectly affected ZIP codes
indirectly_affected_zip_count = indirectly_affected_zips['ZIP_CD'].nunique()

# Display results
print(f"Number of indirectly affected ZIP codes in Texas:
      {indirectly_affected_zip_count}")

```

Number of indirectly affected ZIP codes in Texas: 333

4.

```

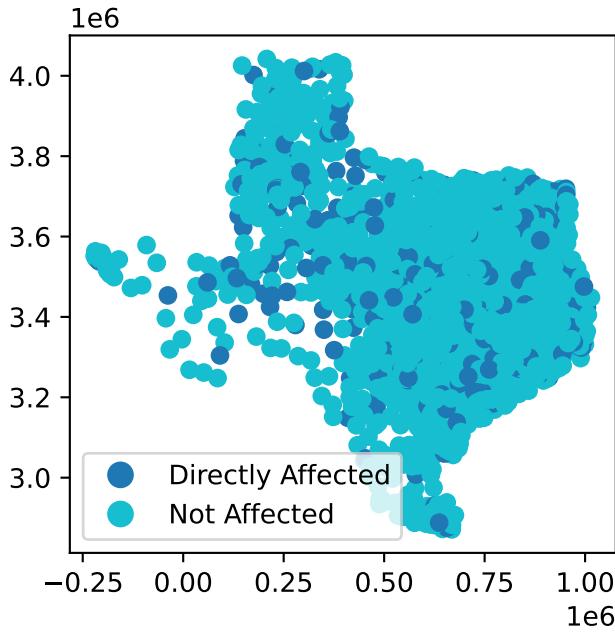
zips_texas_centroids['Category'] = 'Not Affected'

# Label directly affected ZIP codes
directly_affected_zips = closure_choropleth['ZIP_CD'].tolist()
zips_texas_centroids.loc[zips_texas_centroids['ZCTA5'].isin(directly_affected_zips),
                         'Category'] = 'Directly Affected'

# Label indirectly affected ZIP codes (within 10 miles, but not directly
# affected)
indirectly_affected_zips_only =
    indirectly_affected_zips[~indirectly_affected_zips['ZIP_CD'].isin(directly_affected_zips)]
indirectly_affected_zip_codes =
    indirectly_affected_zips_only['ZIP_CD'].tolist()
zips_texas_centroids.loc[zips_texas_centroids['ZCTA5'].isin(indirectly_affected_zip_codes),
                         'Category'] = 'Indirectly Affected'

zips_texas_centroids.plot(
    column='Category',
    legend=True
)

```



Reflecting on the exercise (10 pts)

1. The potential issue here is accurately identifying hospital closures and openings year-over-year, given that simply looking at changes in hospital names might lead to incorrect conclusions. Some hospitals may appear to have “closed” or “opened” due to name changes or administrative updates, rather than actual closures or openings. To address this, the approach suggested is filtering hospitals by unique identifiers of address and zipcode. By verifying changes based on location data rather than names alone, we can more reliably confirm actual closures and openings. For instance, if two hospitals with specific addresses and zipcodes are missing in a given year, they are likely closed. Conversely, new addresses and zipcodes indicate genuine openings.
2. The two approaches provide a basic approximation of hospital access. There are limitations such as the 10-mile radius might not accurately capture true accessibility. Some rural areas may have fewer transportation options, making a 10-mile distance a significant barrier, while urban areas may have better transit or multiple hospitals nearby. Also, we are measuring the data based on zip code areas, which in reality it is subject to change and is not all accurate w.r.t. boundary measurements. There are potential ways to improve this, such as adding GIS information of transportation times into the system, considering hospital deisity and population.