Problem Set 4

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): Katika Klinkaew, katikak
    - Partner 2 (name and cnet ID): Liujun Hua, liujunh
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: ***KK** ***LH**
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: ***0** Late coins left after submission: **___**
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.

```
import pandas as pd
import altair as alt
import os
import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

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

1.

```
PRVDR_CTGRY_SBTYP_CD = Sub-type of Provider
PRVDR_CTGRY_CD = Provider Category Code
PRVDR_NUM = CMS Certification Number
PGM_TRMNTN_CD = Termination Code
TRMNTN_EXPRTN_DT = Date the provider was terminated
FAC_NAME = Facility Name
ZIP_CD = Zip code
STATE_CD = State Abbreviation
```

2. a.

```
# set path and read in the pos2016.csv file
pset_path =
 ↪  '/Users/katikaklinkaew/Documents/GitHub/problem-set-4-katika-and-liujun/data/'

# create a function to load, filter, and count number of short-term hospitals
def read_short_term_hospitals(year, pset_path):
    """
    read the provider-of-service data for a given year, filters for
 ↪  short-term hospitals,
    and returns the filtered DataFrame.
```

```
    Parameters:
        year (int): The year of the data (e.g., 2017).
        pset_path (str): The path to the directory containing the CSV files.
    """
    file_path = os.path.join(pset_path, f'pos{year}.csv')
    df = pd.read_csv(file_path, encoding='latin1')
    df = df[(df['PRVDR_CTGRY_CD'] == 1) & (df['PRVDR_CTGRY_SBTYP_CD'] == 1)]
    print(
        f'There are {df.shape[0]} short-term hospitals reported in {year}
        ↪  data')
    return df

# import pos_2016.csv
df_pos2016 = read_short_term_hospitals(2016, pset_path)
```

```
There are 7245 short-term hospitals reported in 2016 data
```

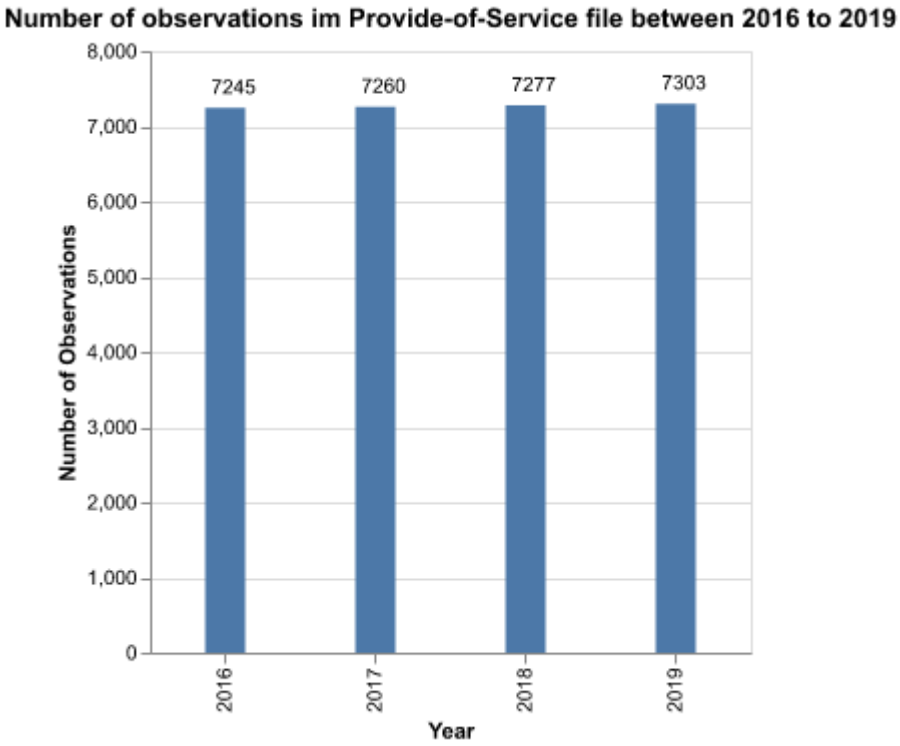
b.
According to the American Hospital Association (https://www.aha.org/statistics/2018-01-09-fast-facts-us-hospitals-2018-pie-charts), there were 5,534 hospitals in the U.S. in total which is much lower than 7,245 considering that our data only consists of short-term hospitals. One reason could be due to the difference in definition of a hospital from each source.

3.

```
# repeat the steps for each year
df_pos2017 = read_short_term_hospitals(2017, pset_path)
df_pos2018 = read_short_term_hospitals(2018, pset_path)
df_pos2019 = read_short_term_hospitals(2019, pset_path)

# append all the pos data
df_pos2016_to_2019 = pd.concat([df_pos2016.assign(year=2016),
↪  df_pos2017.assign(
    year=2017), df_pos2018.assign(year=2018), df_pos2019.assign(year=2019)],
↪  ignore_index=True)
```

```
There are 7260 short-term hospitals reported in 2017 data
There are 7277 short-term hospitals reported in 2018 data
There are 7303 short-term hospitals reported in 2019 data
```

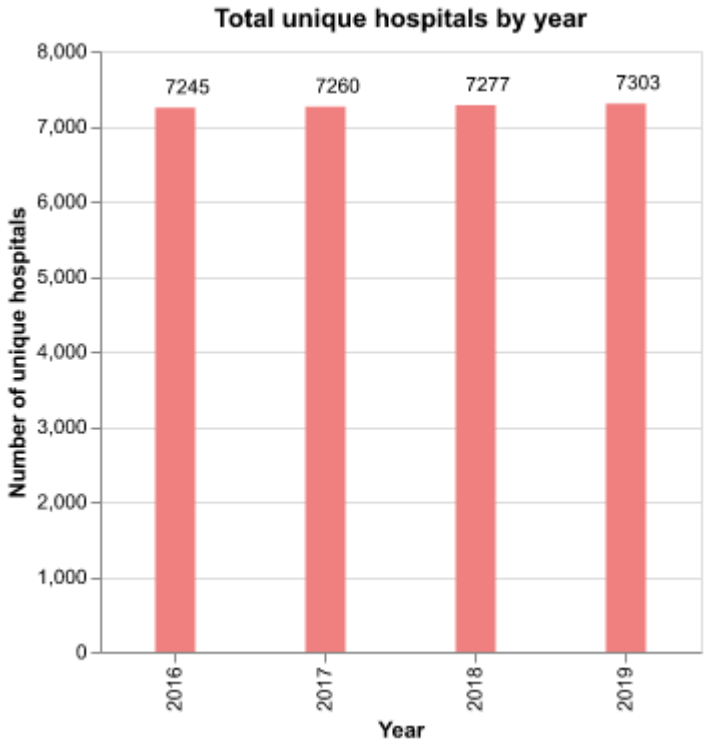
```
# plot number of observations by year
observation_by_year = df_pos2016_to_2019.groupby(
    'year').size().reset_index(name='observation_count')

observation_by_year_plot =
 ↪  alt.Chart(observation_by_year).mark_bar(size=20).encode(
    alt.X('year:O', title='Year'),
    alt.Y('observation_count:Q', title='Number of Observations'),
).properties(
    title='Number of observations im Provide-of-Service file between 2016 to
 ↪  2019',
    width=300,
    height=300
)

observation_label = observation_by_year_plot.mark_text(
    align='left',
    baseline='bottom',
    dx=-7,
    dy=-5,
    fontSize=10
).encode(
    text='observation_count:Q'
)

observation_by_year_plot + observation_label
```



4. a. Plot number of unique hospitals by year

```
unique_hospital_by_year = df_pos2016_to_2019.groupby(
    'year')['PRVDR_NUM'].nunique().reset_index(name='unique_hospital_count')

unique_hospital_by_year_plot =
 ↪  alt.Chart(unique_hospital_by_year).mark_bar(size=20,
 ↪  color='lightcoral').encode(
    alt.X('year:O', title='Year'),
    alt.Y('unique_hospital_count:Q', title='Number of unique hospitals')
).properties(
    title='Total unique hospitals by year',
    width=300,
    height=300
)

unique_label = unique_hospital_by_year_plot.mark_text(
    align='left',
    baseline='bottom',
    dx=-5,
    dy=-5,
```

```
    fontSize=10
).encode(
    text='unique_hospital_count:Q'
)

unique_hospital_by_year_plot + unique_label
```



b.

From the two plots, we can see that the number of unique hospitals is exactly the same as the number of observations in each year. It tells us that a hospital only appears in the data once in a specific year, that is each row in each year represents a snapshot of a unique hospital.

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

1.

```
df = df_pos2016_to_2019.copy()

# initialize a dictionary to store termination years
termination_years = {}

# extract the active hospitals in 2016
active_2016 = df[(df['year'] == 2016) & (df['PGM_TRMNTN_CD'] == 0)]

active_df = df[df['PRVDR_NUM'].isin(active_2016['PRVDR_NUM'])]

# group data by CMS code
for hospital, group in active_df.groupby('PRVDR_NUM'):
    # sort records by year for this hospital
    group = group.sort_values('year')

    # set a default termination year as None
    termination_year = None

    # check each year from 2017 to 2019 for termination
    for year in [2017, 2018, 2019]:
        # filter the data for the current year
        yearly_data = group[group['year'] == year]

        # if no record for the hospital in this year, mark as terminated
        if yearly_data.empty:
            termination_year = year
            break
        # if the hospital is present but not active, mark as terminated
        elif yearly_data['PGM_TRMNTN_CD'].values[0] != 0:
            termination_year = year
            break

    # if a termination year was identified, store it in the dictionary
    if termination_year:
        termination_years[hospital] = termination_year

# convert the termination years to a DataFrame
terminate_year_df = pd.DataFrame(list(termination_years.items()), columns=[
                                'PRVDR_NUM', 'Termination_Year'])

# adding ZIP code information for each hospital from original dataset
closed_hospitals_zipcode = active_df[active_df['year'] == 2016][[
```

```
    'PRVDR_NUM', 'FAC_NAME', 'ZIP_CD']].drop_duplicates('PRVDR_NUM')

# combine termination years with ZIP code information
closed_hospitals = terminate_year_df.merge(
    closed_hospitals_zipcode, on='PRVDR_NUM', how='left')
```

```
print(
    f"{closed_hospitals.shape[0]} hospitals were active in 2016 that were
     ↪  suspected to have closed by 2019")
```

```
174 hospitals were active in 2016 that were suspected to have closed by 2019
```

2.

```
closed_hospitals.sort_values('FAC_NAME')[['FAC_NAME','Termination_Year']].head(10)
```

	FAC_NAME	Termination_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.

```
# count the active hospitals of each zip code every year

active_counts = active_df.groupby(['year', 'ZIP_CD'])['PGM_TRMNTN_CD'].apply(
    lambda x: (x == 0).sum()).reset_index(name='active_count')

zip_and_close_year = pd.DataFrame(closed_hospitals.groupby(
    ['ZIP_CD', 'Termination_Year']).size()).reset_index()

zip_with_merger = []
```

```
for index, row in zip_and_close_year.iterrows():
    zip = row['ZIP_CD']
    year = row['Termination_Year']
    if year == 2019:
        continue

    zip_data = active_counts[active_counts['ZIP_CD'] == zip]
    count_this_year = zip_data[zip_data['year'] == year]['active_count']
    count_next_year = zip_data[zip_data['year'] == (year +
 ↪  1)]['active_count']

    if not count_this_year.empty and not count_next_year.empty:
        # extract scalar values (there should be only one value per year now)
        count_this_year = count_this_year.iloc[0]
        count_next_year = count_next_year.iloc[0]

        if count_this_year <= count_next_year:
            zip_with_merger.append(zip)
```

a. How many hospitals fit this definition of potentially being a merger/acquisition?

```
print(f"{len(zip_with_merger)} hospitals fit this definition of potentially
 ↪  being a merger/acquisition.")
```

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

b. After correcting for this, how many hospitals do you have left?

```
# filter merger hospitials
closed_hospitals = closed_hospitals[~closed_hospitals['ZIP_CD'].isin(
    zip_with_merger)]

print(
    f"{closed_hospitals.shape[0]} hospitals were active in 2016 that were
     ↪  suspected to have closed by 2019")
```

78 hospitals were active in 2016 that were suspected to have closed by 2019

c.

```
closed_hospitals.sort_values(
    'FAC_NAME')[['FAC_NAME', 'Termination_Year']].head(10)
```

	FAC_NAME	Termination_Year
62	ALLIANCE LAIRD HOSPITAL	2019
101	ALLIANCEHEALTH DEACONESS	2019
26	ANNE BATES LEACH EYE HOSPITAL	2019
115	BARIX CLINICS OF PENNSYLVANIA	2019
171	BAYLOR EMERGENCY MEDICAL CENTER	2019
166	BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...	2019
98	BELMONT COMMUNITY HOSPITAL	2019
67	BIG SKY MEDICAL CENTER	2019
65	BLACK RIVER COMMUNITY MEDICAL CENTER	2019
142	CARE REGIONAL MEDICAL CENTER	2019

Download Census zip code shapefile (10 pt)

1. a. File Type | Information Type |
   .dbf| Attribute information |
   .prj | Coordinate Reference System (CRS) description |
   .shp | Geometric or Spatial data (i.e., points, polygon) |
   .shx | Positional index for looking up geometries |
   .xml | Metadata about the dataset including descriptions, purpose, agreement of use etc. |

   b.

```
zip_path = '/Users/katikaklinkaew/Documents/data/gz_2010_us_860_00_500k'

# Print the size of each file in the folder
for dataset in os.listdir(zip_path):
    file_path = os.path.join(zip_path, dataset)
    size = os.path.getsize(file_path)
    if size < 1024:
        size_str = f"{size:.2f} Bytes"
    elif size < 1024 ** 2:
        size_str = f"{size/1024:.2f} KB"
    else:
        size_str = f"{size/(1024 ** 2):.2f} MB"
```

```
    print(f"{dataset}: {size_str}")
```

```
gz_2010_us_860_00_500k.prj: 165.00 Bytes
gz_2010_us_860_00_500k.shx: 258.85 KB
gz_2010_us_860_00_500k.shp: 798.74 MB
gz_2010_us_860_00_500k.dbf: 6.13 MB
gz_2010_us_860_00_500k.xml: 15.27 KB
```

2.

```
import geopandas as gpd
shapefile_path = os.path.join(zip_path, 'gz_2010_us_860_00_500k.shp')
zip_shp = gpd.read_file(shapefile_path)
```

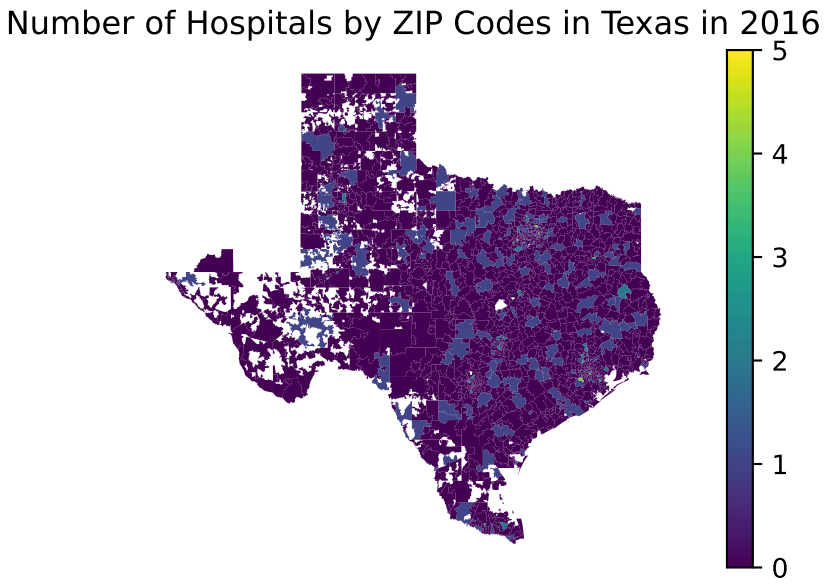
```
# Texas zip codes start with 733, and 750 - 799
zip_tx = zip_shp[zip_shp['NAME'].str.startswith(
    ('733', '75', '76', '77', '78', '79'))]

active_2016['ZIP_CD'] = active_2016['ZIP_CD'].astype(
    int).astype(str).str.zfill(5)
zip_tx['NAME'] = zip_tx['NAME'].astype(str).str.zfill(5)

tx_hospitals = active_2016[active_2016['ZIP_CD'].isin(zip_tx['NAME'])]

tx_hospitals_zip = tx_hospitals.groupby(
    'ZIP_CD').size().reset_index(name='Number of Hospitals')

zip_tx = zip_tx.rename(columns={'NAME': 'ZIP_CD'})

# Merge on ZIP_CD to get Texas ZIP codes with hospital counts
tx_hospitals_merged = zip_tx.merge(tx_hospitals_zip, on='ZIP_CD', how='left')

tx_hospitals_merged['Number of Hospitals'] = tx_hospitals_merged['Number of
↪  Hospitals'].fillna(
    0)
```

```
tx_hospitals_merged[tx_hospitals_merged['Number of Hospitals']
                    == tx_hospitals_merged['Number of Hospitals'].max()]
```

	GEO_ID	ZCTA5	ZIP_CD	LSAD	CENSUSAREA	geometry
472	8600000US77030	77030	77030	ZCTA5	2.521	POLYGON ((-95.41436 29.69445, -9

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
tx_hospitals_zip_plot = tx_hospitals_merged.plot(
    column='Number of Hospitals',
    legend=True,
    linewidth=0.5).set_axis_off()
plt.title('Number of Hospitals by ZIP Codes in Texas in 2016')
plt.show()
```

```
<Figure size 3000x3000 with 0 Axes>
```



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 = gpd.GeoDataFrame({
    'ZIP_CD': zip_shp['ZCTA5'],
```

```
    'centroid': zip_shp.geometry.centroid
})

zips_all_centroids.shape
```

```
(33120, 2)
```

The dimensions of the GeoDataFrame include 33120 rows and 2 columns. The first column 'ZIP_CD' is the zip codes nationally and the second column 'centroid' is the position of all the points in the zip code polygons.

2.

```
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZIP_CD'].str.startswith(
    ('733', '75', '76', '77', '78', '79'))]

# the border states include NM, OK, AR, LA
# zip codes start with 700 - 749, 870 - 884

zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZIP_CD'].str.startswith(
    ('7', '87', '88'))]

print(
    f"Unique zip codes in zip_texas_centroids:
     ↪  {zips_texas_centroids['ZIP_CD'].nunique()}")
print(
    f"Unique zip codes in zips_texas_borderstates_centroids:
     ↪  {zips_texas_borderstates_centroids['ZIP_CD'].nunique()}")
```

```
Unique zip codes in zip_texas_centroids: 1935
Unique zip codes in zips_texas_borderstates_centroids: 4057
```

3. I will do a inner join merge on variables merging on ZIP_CD.

```
zip_2016 = pd.DataFrame(active_2016['ZIP_CD'].fillna(
    0).astype(int).astype(str).str.zfill(5).drop_duplicates())
zip_2016.columns = ['ZIP_CD']
zips_texas_borderstates_centroids['ZIP_CD'] =
 ↪  zips_texas_borderstates_centroids['ZIP_CD'].astype(
    str)
```

```
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    zip_2016, on='ZIP_CD')
```

```
zips_withhospital_centroids.head(1)
```

	ZIP_CD	centroid
0	70043	POINT (-89.96276 29.94804)

4. a. Try the join with 10 zip codes

```
import time
from shapely.geometry import Point

zips_texas_centroids = zips_texas_centroids.set_geometry('centroid')
zips_withhospital_centroids = zips_withhospital_centroids.set_geometry(
    'centroid')

# subset the first ten row
zips_texas_centroids_subset = zips_texas_centroids[:10]

start_time = time.time()

nearest_distances = []

# loop over each row in the first 10 rows of zips_texas_centroids
for _, row_tx in zips_texas_centroids_subset.iterrows():
    point1 = row_tx['centroid']
    nearest_distance = float('inf')  # Initialize to a large number

    # loop over each row in zips_withhospital_centroids
    for _, row_all in zips_withhospital_centroids.iterrows():
        point2 = row_all['centroid']

        # calculate distance
        distance_new = point1.distance(point2)

        # update nearest distance if a closer point is found
        if distance_new < nearest_distance:
            nearest_distance = distance_new
```

```
    # append the nearest distance for the current row in zips_texas_centroids
    nearest_distances.append(nearest_distance)

# assign the calculated distances
zips_texas_centroids_subset['nearest_distance'] = nearest_distances

end_time = time.time()

print(f"Runtime: {end_time - start_time}")
```

```
Runtime: 0.0890359878540039
```

It takes around 0.13 seconds to join the 10 zip codes. As there are 1935 unique zip codes in zip_texas_centroids, the total would be 0.13*1935/10 = 25 seconds. In other words, we estimate the entire procedure will take around 25 seconds.

b. Doing the full join

```
start_time = time.time()

nearest_distances = []

# loop over each row in zips_texas_centroids
for _, row_tx in zips_texas_centroids.iterrows():
    point1 = row_tx['centroid']
    nearest_distance = float('inf')  # initialize to a large number

    # loop over each row in zips_withhospital_centroids
    for _, row_all in zips_withhospital_centroids.iterrows():
        point2 = row_all['centroid']

        # calculate distance
        distance_new = point1.distance(point2)

        # update nearest distance if a closer point is found
        if distance_new < nearest_distance:
            nearest_distance = distance_new

    # append the nearest distance for the current row in zips_texas_centroids
    nearest_distances.append(nearest_distance)
```

```
# assign the calculated distances
zips_texas_centroids['nearest_distance'] = nearest_distances

end_time = time.time()

print(f"Runtime: {end_time - start_time}")
```

```
Runtime: 15.550760984420776
```

c.
The unit is 'Degree'. One degree of latitude is approximately 69 miles, while one degree of longitude is approximately 54.6 miles. In this case, we will neglect longitude and multiply the degree by 69 to convert it to miles.

```
# convert the degree unit to miles
zips_texas_centroids['nearest_distance'] =
 ↪  zips_texas_centroids['nearest_distance'] * 69
```

5. Calculate the average distance to the nearest hospital for each zip code in Texas

    a. The unit is 'Miles'.

    b. Report the average distance in miles

```
zips_texas_centroids['nearest_distance'].mean()
```

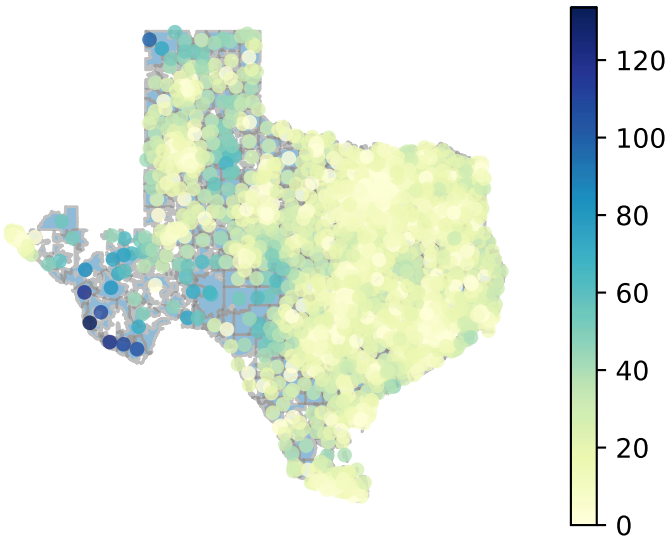
```
np.float64(14.560206510814892)
```

The average distance to the nearest hospital for each zip code in Texas is 14.56 miles, which makes sense.

c. Map the value for each zip code

```
fig, ax = plt.subplots()

# Plotting the polygons from zip_tx
zip_tx.plot(ax=ax, color=None, edgecolor="grey", alpha=0.5, label="Dense")

# Plotting the centroids with a color gradient based on 'nearest_distance'
# `cmap` sets the color map (e.g., 'viridis', 'coolwarm', 'plasma', etc.)
zips_texas_centroids.plot(
    ax=ax,
    column='nearest_distance',
```

```
    cmap='YlGnBu',  # Choose a color map that suits your visualization
    alpha=0.8,
    markersize=20,
    legend=True,  # Show legend for color scale
    label="Centroids"
)

# Remove axis for a cleaner look
ax.set_axis_off()

# Show plot
plt.show()
```



Most areas on the east side of Texas have nearby hospitals, while some areas on the west side near the border have a much greater distance to the nearest hospital.

Effects of closures on access in Texas (15 pts)

1.

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

# filter to hospital closures in Texas
```

```
closed_hospitals_tx = closed_hospitals[closed_hospitals['ZIP_CD'].isin(
    zip_tx['ZIP_CD'])]

# count number of hospital closures by each zip code
closed_hospitals_tx_zip = closed_hospitals_tx.groupby(
    'ZIP_CD').size().reset_index(name='Number of Hospital Closures')

closed_hospitals_tx_zip
```

	ZIP_CD	Number of Hospital Closures
0	75051	1
1	75087	1
2	75140	1
3	75235	1
4	75390	1
5	76520	1
6	76531	1
7	76645	1
8	77065	1
9	78336	1
10	78613	1
11	79520	1
12	79529	1
13	79902	1

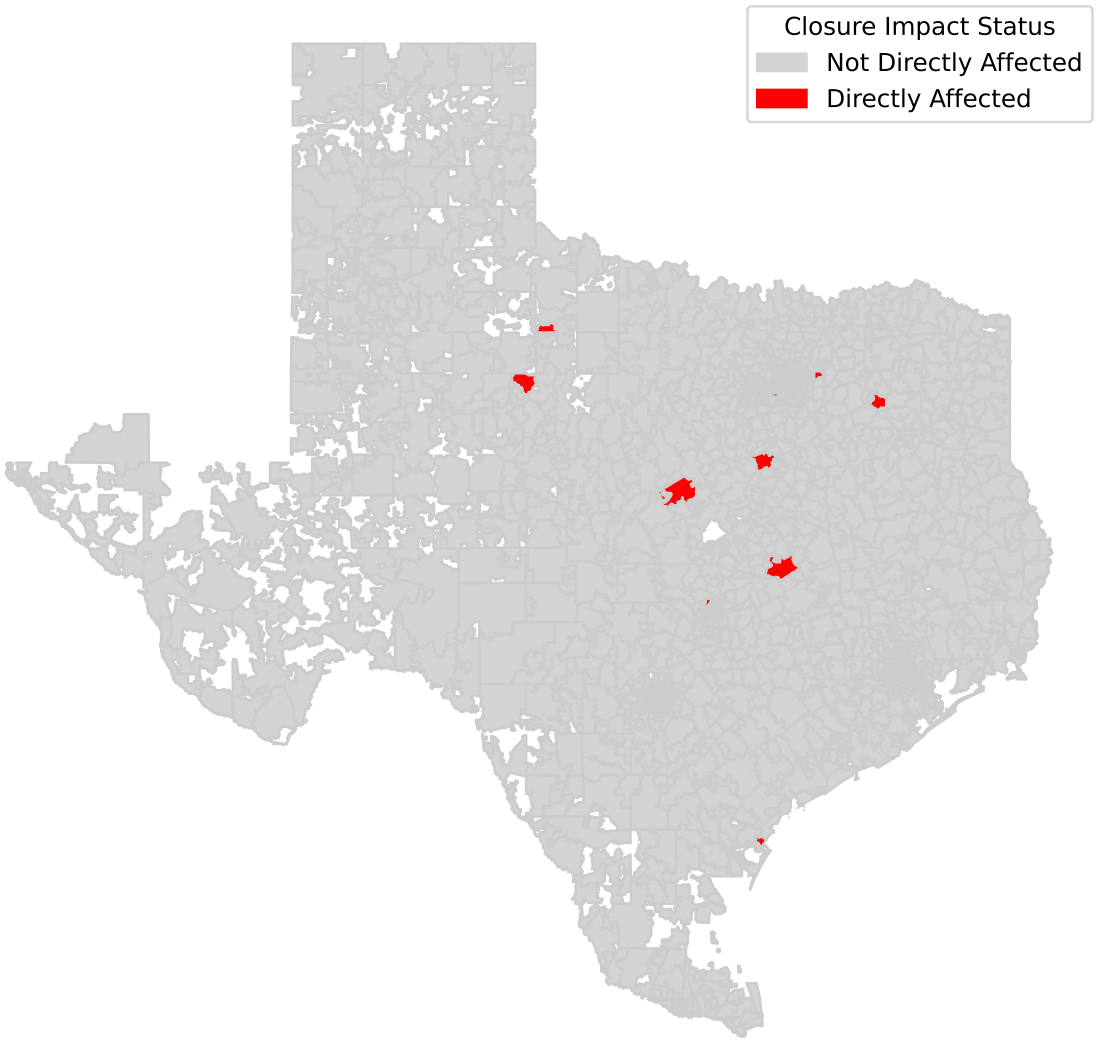
2.

```
# merge hospital closures in Texas to Texas shapefile
closed_hospitals_tx_zip_merged = zip_tx.merge(
    closed_hospitals_tx_zip, on='ZIP_CD', how='left')

# replace na with 0
closed_hospitals_tx_zip_merged['Number of Hospital Closures'] =
 ↪  closed_hospitals_tx_zip_merged['Number of Hospital Closures'].fillna(
    0)

closed_hospitals_tx_zip_merged['directly affected'] = (
    closed_hospitals_tx_zip_merged['Number of Hospital Closures'] >
 ↪  0).astype(int)
directly_affected_zip_count = closed_hospitals_tx_zip_merged['directly
 ↪  affected'].sum(
```

```
)

print('There are', directly_affected_zip_count,
      'directly affected zip codes in Texas')
```

```
There are 14 directly affected zip codes in Texas
```

```
import matplotlib.patches as mpatches

# plot a choropleth with directly affected zip codes in red and others in
 ↪ lightgray
fig, ax = plt.subplots(figsize=(10, 8))
closed_hospitals_tx_zip_merged.plot(
    column='directly affected',
    color=closed_hospitals_tx_zip_merged['directly affected'].map(
        {0: 'lightgray', 1: 'red'}),  # Single color for affected areas
    edgecolor='0.8',
    legend=True,
    ax=ax
)
# create custom legends
affected_patch = mpatches.Patch(color='red', label='Directly Affected')
not_affected_patch = mpatches.Patch(
    color='lightgray', label='Not Directly Affected')
plt.legend(handles=[not_affected_patch, affected_patch],
           loc='upper right', title="Closure Impact Status")

ax.set_axis_off()
plt.title('Texas ZIP Codes directly affected by hospital closure in 2016 -
 ↪ 2019')
plt.show()
```



3.

```
# create a GeoDataFrame of the directly affected zip codes
directly_affected_zips =
 ↪  closed_hospitals_tx_zip_merged[closed_hospitals_tx_zip_merged['directly
 ↪  affected'] == 1].copy()

directly_affected_zips = directly_affected_zips[['ZIP_CD', 'geometry']]
```

```
print('Check if the object is a GepDataFrame:', type(directly_affected_zips))
```

```
Check if the object is a GepDataFrame: <class
'geopandas.geodataframe.GeoDataFrame'>
```

```
# geopandas uses meters to creat buffer, check the unit for our GeoDataFrame
directly_affected_zips.crs
# create a list of directly affected zip codes
directly_affected_zips_list =
 ↪  directly_affected_zips['ZIP_CD'].unique().tolist()

# create a copy and convert into crs for Texas that is in meters
directly_affected_buffer = directly_affected_zips.copy()
directly_affected_buffer = directly_affected_buffer.to_crs(epsg=3083)

# create a 10-mile buffer by converting into meters
directly_affected_buffer['10-mile radius'] =
 ↪  directly_affected_buffer.geometry.buffer(
    10*1609.34)
directly_affected_buffer = directly_affected_buffer.set_geometry(
    '10-mile radius')
```

```
# before doing spatial join, ensure the overall Texas ZIP shapefile is in the
 ↪  same crs
zip_tx = zip_tx.to_crs(epsg=3083)

# do the spatial join which will return all directly and indirectly affected
 ↪  zip codes
indirectly_affected_zips = gpd.sjoin(zip_tx, directly_affected_buffer,
                                     how="inner", predicate='intersects')

# rename ZIP_CD_left to 'ZIP_CD' for indirectly affected zips GeoDataFrame
 ↪  and set geometry back to geometry
indirectly_affected_zips = indirectly_affected_zips.rename(
    columns={'ZIP_CD_left': 'ZIP_CD'})

# create a list of only indirectly affected zip codes
indirectly_affected_zips_list =
 ↪  indirectly_affected_zips['ZIP_CD'].unique().tolist()
```

```
indirectly_affected_zips_list = [
    zip_code for zip_code in indirectly_affected_zips_list
    if zip_code not in directly_affected_zips_list
]

print('There are', len(indirectly_affected_zips_list),
      'indirectly affected zip codes in Texas')

# set geometry back to the original geometry column
indirectly_affected_zips = indirectly_affected_zips.set_geometry('geometry')
```

```
There are 342 indirectly affected zip codes in Texas
```

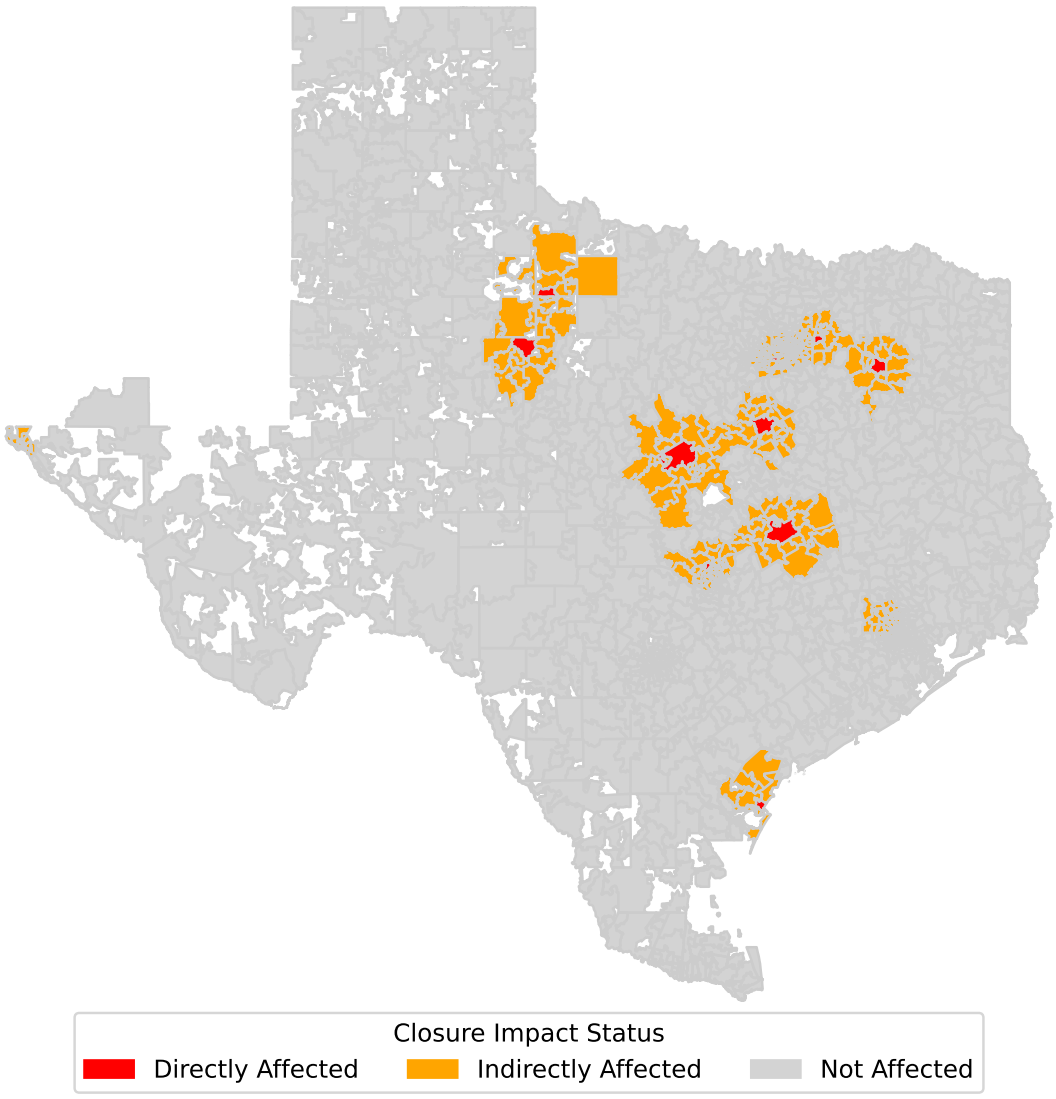
4.

```
# create a column representing Hospital Closures Impact Status
closed_hospitals_tx_zip_merged['Closure Impact Status'] = 'Not Affected'
closed_hospitals_tx_zip_merged.loc[closed_hospitals_tx_zip_merged['ZIP_CD'].isin(
    directly_affected_zips_list), 'Closure Impact Status'] = 'Directly
 ↪  Affected'
closed_hospitals_tx_zip_merged.loc[closed_hospitals_tx_zip_merged['ZIP_CD'].isin(
    indirectly_affected_zips_list), 'Closure Impact Status'] = 'Indirectly
 ↪  Affected'
print(closed_hospitals_tx_zip_merged['Closure Impact Status'].value_counts())
```

```
Closure Impact Status
Not Affected           1579
Indirectly Affected     342
Directly Affected        14
Name: count, dtype: int64
```

```
# create a color map for different impact status
color_map = {
    'Directly Affected': 'red',
    'Indirectly Affected': 'orange',
    'Not Affected': 'lightgray'
}

# plot a choropleth
fig, ax = plt.subplots(figsize=(10, 8))
closed_hospitals_tx_zip_merged.plot(
    column='Closure Impact Status',
```

```
    color=closed_hospitals_tx_zip_merged['Closure Impact Status'].map(
        color_map),
    edgecolor='0.8',
    legend=True,
    ax=ax
)

# create a custom legend
patches = [mpatches.Patch(color=color, label=status)
           for status, color in color_map.items()]
ax.legend(handles=patches, title="Closure Impact Status",
          loc='lower center', bbox_to_anchor=(0.5, -0.05), ncol=3)

ax.set_axis_off()
plt.title('Impact Status of Hospital Closures in Texas (2016-2019)')
plt.show()
```



Reflecting on the exercise (10 pts)

1. There are some deficiencies in the method. In Section 2, when attempting to identify 'false closures,' we removed zip codes where the number of active hospitals in the closure year did not decrease in subsequent years. However, this approach only partially addresses the possibility of mergers. Since the count of active hospitals in the closure year does not include the hospitals that closed, the number may remain stable in the follow-

ing year even if no merger or acquisition occurred. This means we may have mistakenly excluded some zip codes with actual closures. To improve, we could consider to compare the number of active hospitals in the year before the closure to the year after the closure. This would help us identify potential mergers more accurately, as it accounts for changes in hospital availability over a broader period.

2. This method can partially reflect changes in the hospital access due to hospital closures especially if that particular zip-code already has a lower access, that is low amount of hospitals in the zip code. Otherwise, they might not be affected much by a hospital closure. Same as indirectly affected zip codes, they might not be affected at all if there are other hospitals in their areas. So we should also consider existing hospitals density in that zip code and neighboring zip codes as well. Another thing to consider is the share of patients that the closed hospitals have in the zip-code level. Also, the change in travel distance to nearest hospital in the zip code could tell us more about the actual impact on that zip code.