PS4: Spatial Analysis of Rural Hospital Closures by Attaullah Abbasi (attaullahabbasi12)

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 (Attaullah Abbasi and attaullahabbasi):
    - Partner 2 (N/A - assignment completed solo):
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: **___** **___**
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. For the 2016 data, I pulled the following variables:

    i. `PRVDR_NUM`: Unique CMS certification number for each hospital
    ii. `FAC_NAME`: Facility name
    iii. `PRVDR_CTGRY_CD`: Provider category, to identify hospitals
    iv. `PRVDR_CTGRY_SBTYP_CD`: Provider subtype, to identify short-term hospitals
    v. `CRTFCTN_DT`: Certification date, to verify active facilities
    vi. `ORGNL_PRTCPTN_DT`: Original participation date, to confirm participation history
    vii. `PGM_TRMNTN_CD`: Termination code, to track closures
    viii. `TRMNTN_EXPRTN_DT`: Termination date, to identify closure timelines
    ix. `ZIP_CD`: Zip code for geographic analysis
    x. `STATE_CD`: State abbreviation, useful for filtering Texas and nearby states
    xi. `CITY_NAME`: City name for additional geographic context
    xii. `ST_ADR`: Street address for potential geolocation
    xiii. `CBSA_URBN_RRL_IND`: Urban-rural indicator, helpful for understanding location context
    xiv. `GNRL_CNTL_TYPE_CD`: Control type, to analyze hospital ownership

These variables allow for filtering, spatial analysis, tracking closures, and understanding hospital characteristics as required by the problem set.

2.

```
import pandas as pd

# Load the pos2016.csv file
data_2016 = pd.read_csv("pos2016.csv")

# Filter for short-term hospitals with provider type code 01 and subtype code
 ↪  01
short_term_hospitals_2016 = data_2016[(data_2016['PRVDR_CTGRY_CD'] == 1) &
 ↪  (data_2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]
```

```
# a. Count the number of hospitals in this subset
hospital_count_2016 = short_term_hospitals_2016.shape[0]
hospital_count_2016
```

7245

a.

The dataset reports 7,245 short-term hospitals for 2016.

This count seems high compared to typical figures from industry sources, such as the American Hospital Association (AHA), which might suggest differences in data scope or classification criteria used by CMS.

b.

According to the American Hospital Association (AHA), there were approximately 5,534 registered hospitals in the U.S. in 2016, with around 4,840 classified as community hospitals, which includes most short-term general hospitals. This figure is notably lower than the 7,245 short-term hospitals reported in the CMS dataset.

Reasons for the Discrepancy:

Data Scope: The CMS dataset may include facilities that are Medicare/Medicaid-certified but not registered with the AHA, potentially increasing the count.

Different Definitions: CMS may classify certain specialty hospitals or facilities providing limited services as short-term if they meet Medicare eligibility criteria, even if other sources don't typically include them.

Timing and Updates: The dataset represents a Q4 2016 snapshot, while the AHA's data may reflect closures, mergers, or reclassifications over the entire year.

These factors could explain the higher count of short-term hospitals in the CMS dataset compared to other industry sources.

3.

```
import pandas as pd
import altair as alt

# Load each year's data with appropriate encoding where needed
data_2016 = pd.read_csv("pos2016.csv")
data_2017 = pd.read_csv("pos2017.csv")
data_2018 = pd.read_csv("pos2018.csv", encoding="ISO-8859-1")
```

```
data_2019 = pd.read_csv("pos2019.csv", encoding="ISO-8859-1")

# Define a function to filter for short-term hospitals
def filter_short_term(data):
    return data[(data['PRVDR_CTGRY_CD'] == 1) & (data['PRVDR_CTGRY_SBTYP_CD']
    ↪  == 1)]

# Apply filtering to each dataset and add a 'Year' column for each
data_2016 = filter_short_term(data_2016)
data_2016['Year'] = 2016

data_2017 = filter_short_term(data_2017)
data_2017['Year'] = 2017

data_2018 = filter_short_term(data_2018)
data_2018['Year'] = 2018

data_2019 = filter_short_term(data_2019)
data_2019['Year'] = 2019

# Combine all datasets into a single DataFrame
all_years_data = pd.concat([data_2016, data_2017, data_2018, data_2019])

# Count the number of observations (hospitals) by year
observations_by_year =
↪  all_years_data.groupby('Year').size().reset_index(name='Count')

# Plot using Altair
bars = alt.Chart(observations_by_year).mark_bar(color='steelblue').encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Count:Q', title='Number of Short-Term Hospitals')
).properties(
    title='Number of Observations in Dataset by Year',
    width=400,
    height=300
)

# Add text labels on top of each bar
text = bars.mark_text(
    align='center',
    baseline='bottom',
    dy=-5  # Adjust the position of the text above the bars
```

```
).encode(
    text='Count:Q'
)

# Combine the bars and the text
(bars + text).display()
```

```
/var/folders/sd/jd56rfvs19gbncmb5yqlmfym0000gn/T/ipykernel_50159/949909169.py:7:
DtypeWarning: Columns (13) have mixed types. Specify dtype option on import
or set low_memory=False.
  data_2018 = pd.read_csv("pos2018.csv", encoding="ISO-8859-1")
```

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

4. a.

```
import pandas as pd
import altair as alt

# Assuming data for each year is already filtered and loaded
# If not, make sure data_2016, data_2017, data_2018, data_2019 are loaded and
 ↪  filtered as before

# Add a 'Year' column to each dataset
data_2016['Year'] = 2016
data_2017['Year'] = 2017
data_2018['Year'] = 2018
data_2019['Year'] = 2019

# Combine all datasets into a single DataFrame
all_years_data = pd.concat([data_2016, data_2017, data_2018, data_2019])

# Count unique hospitals (by PRVDR_NUM) per year
unique_hospitals_by_year =
 ↪  all_years_data.groupby('Year')['PRVDR_NUM'].nunique().reset_index(name='UniqueCount')

# Plot using Altair
bars = alt.Chart(unique_hospitals_by_year).mark_bar(color='seagreen').encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('UniqueCount:Q', title='Number of Unique Short-Term Hospitals')
).properties(
    title='Number of Unique Short-Term Hospitals in Dataset by Year',
```

```
    width=400,
    height=300
)

# Add text labels on top of each bar
text = bars.mark_text(
    align='center',
    baseline='bottom',
    dy=-5  # Adjust the position of the text above the bars
).encode(
    text='UniqueCount:Q'
)

# Display the combined chart with bars and labels
(bars + text).display()


# Double checking for duplicates
# Check for duplicates in PRVDR_NUM within each year
duplicates_2016 = data_2016.duplicated(subset='PRVDR_NUM').sum()
duplicates_2017 = data_2017.duplicated(subset='PRVDR_NUM').sum()
duplicates_2018 = data_2018.duplicated(subset='PRVDR_NUM').sum()
duplicates_2019 = data_2019.duplicated(subset='PRVDR_NUM').sum()

print(f"2016 Duplicates: {duplicates_2016}")
print(f"2017 Duplicates: {duplicates_2017}")
print(f"2018 Duplicates: {duplicates_2018}")
print(f"2019 Duplicates: {duplicates_2019}")
```

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

```
2016 Duplicates: 0
2017 Duplicates: 0
2018 Duplicates: 0
2019 Duplicates: 0
```

b.

The identical values in both plots indicate that each short-term hospital appears only once per year, with no duplicates. This structure confirms that each hospital's CMS certification number is unique annually, allowing for clear year-over-year tracking.

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

1.

```
# Step 1: Filter active hospitals in each year
active_2016 = data_2016[data_2016['PGM_TRMNTN_CD'] == 0]
active_2017 = data_2017[data_2017['PGM_TRMNTN_CD'] == 0]
active_2018 = data_2018[data_2018['PGM_TRMNTN_CD'] == 0]
active_2019 = data_2019[data_2019['PGM_TRMNTN_CD'] == 0]

# Step 2: Define function to check if provider is active in a given year
def provider_in_year(df, provider_num):
    row = df[df['PRVDR_NUM'] == provider_num]
    return not row.empty and row['PGM_TRMNTN_CD'].values[0] == 0

# Step 3: Determine closure year for each provider number
def determine_closure_year(provider_num):
    if not provider_in_year(active_2017, provider_num):
        return 2017
    elif not provider_in_year(active_2018, provider_num):
        return 2018
    elif not provider_in_year(active_2019, provider_num):
        return 2019
    return None

# Step 4: Apply closure detection on active hospitals in 2016
active_2016_only = active_2016[active_2016['PGM_TRMNTN_CD'] == 0]
active_2016_only['Year_Closed'] =
 ↪  active_2016_only['PRVDR_NUM'].apply(determine_closure_year)

# Step 5: Filter out hospitals that have a closure year assigned
closed_hospitals =
 ↪  active_2016_only.dropna(subset=['Year_Closed']).reset_index(drop=True)

# Output the result
print(f"Number of suspected hospital closures: {len(closed_hospitals)}")
closed_hospitals[['FAC_NAME', 'ZIP_CD', 'Year_Closed']]
```

```
Number of suspected hospital closures: 174
```

	FAC_NAME	ZIP_CD	Year_Closed
0	WEDOWEE HOSPITAL	36278.0	2019.0
1	GEORGIANA MEDICAL CENTER	36033.0	2019.0
2	RMC JACKSONVILLE	36265.0	2018.0
3	NORTH ALABAMA SPECIALITY HOSPITAL	35611.0	2018.0
4	ABRAZO MARYVALE CAMPUS	85031.0	2017.0
...	...	...	...
169	LITTLE RIVER HEALTHCARE CAMERON HOSPITAL	76520.0	2019.0
170	BAY AREA REGIONAL MEDICAL CENTER, LLC	77598.0	2018.0
171	BAYLOR EMERGENCY MEDICAL CENTER	75087.0	2019.0
172	CONTINUECARE HOSPITAL AT MEDICAL CENTER ODESSA	79761.0	2017.0
173	TEXAS GENERAL HOSPITAL- VZRMC LP	75140.0	2019.0

2.

```
# Sort the closed hospitals by facility name
sorted_closed_hospitals =
 ↪  closed_hospitals.sort_values(by='FAC_NAME').reset_index(drop=True)

# Display the names and year of suspected closure for the first 10 rows
sorted_closed_hospitals[['FAC_NAME', 'Year_Closed']].head(10)
```

	FAC_NAME	Year_Closed
0	ABRAZO MARYVALE CAMPUS	2017.0
1	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2017.0
2	AFFINITY MEDICAL CENTER	2018.0
3	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2017.0
4	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2017.0
5	ALLIANCE LAIRD HOSPITAL	2019.0
6	ALLIANCEHEALTH DEACONESS	2019.0
7	ANNE BATES LEACH EYE HOSPITAL	2019.0
8	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	2017.0
9	BANNER CHURCHILL COMMUNITY HOSPITAL	2017.0

3. a.

```
# Count active hospitals by ZIP code for each year
active_by_zip_2016 = active_2016['ZIP_CD'].value_counts().to_dict()
active_by_zip_2017 = active_2017['ZIP_CD'].value_counts().to_dict()
active_by_zip_2018 = active_2018['ZIP_CD'].value_counts().to_dict()
```

```
active_by_zip_2019 = active_2019['ZIP_CD'].value_counts().to_dict()

# Define function to check if closure might be due to merger/acquisition
def is_possible_merger(row):
    zip_code = row['ZIP_CD']
    year_closed = row['Year_Closed']

    if year_closed == 2017:
        return active_by_zip_2017.get(zip_code, 0) >=
         ↪  active_by_zip_2016.get(zip_code, 0)
    elif year_closed == 2018:
        return active_by_zip_2018.get(zip_code, 0) >=
         ↪  active_by_zip_2017.get(zip_code, 0)
    elif year_closed == 2019:
        return active_by_zip_2019.get(zip_code, 0) >=
         ↪  active_by_zip_2018.get(zip_code, 0)
    return False

# Apply function to identify potential mergers/acquisitions
closed_hospitals['Possible_Merger'] =
 ↪  closed_hospitals.apply(is_possible_merger, axis=1)

# Filter suspected closures that might be due to a merger/acquisition
potential_mergers = closed_hospitals[closed_hospitals['Possible_Merger']]

# Output the result
num_potential_mergers = len(potential_mergers)
print(f"Number of suspected hospital closures that fit the merger/acquisition
 ↪  definition: {num_potential_mergers}")
```

```
Number of suspected hospital closures that fit the merger/acquisition
definition: 8
```

b.
c.

Download Census zip code shapefile (10 pt)

1. a.
   b.
2.

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

1.
2.
3.
4. a.
   b.
5. a.
   b.
   c.

Effects of closures on access in Texas (15 pts)

1.
2.
3.
4.

Reflecting on the exercise (10 pts)