PS4: Spatial

Hiroaki Kurachi, Luis Senires

2024-11-03

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)

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 (Hiroaki Kurachi and hiroakik):
- Partner 2 (Luis Senires and ldsenires):

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: **HK LS**
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: **01** Late coins left after submission: **02**
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

```
import os
import pandas as pd
import altair as alt
import geopandas as gpd
import matplotlib.pyplot as plt
import numpy as np
import time
import datetime as datetime

import warnings
warnings.filterwarnings('ignore')
```

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

1.

The variables we selected are;

- PRVDR_NUM: CMS certification number
- PRVDR_CTGRY_CD: the type of provider participating in the Medicare/Medicaid program
- PRVDR_CTGRY_SBTYP_CD: the subtype of the provider, within the primary category
- FAC_NAME: name of the provider certified to participate in the Medicare and/or Medicaid programs
- CITY_NAME: city in which the provider is physically located
- ST_ADR: street address where the provider is located
- ZIP_CD; five-digit ZIP code for a provider's physical address
- TRMNTN_EXPRTN_DT: date the provider was terminated
- PGM_TRMNTN_CD: indicates the current termination status for the provider

2.

```
# Set the absolute path of the data folder
# base = (r"C:\Users\hkura\Documents\Uchicago\04 2024
 ↪  Autumn\Python2\problem-set-4-hiro-luis\data")
base = (r"C:\Users\LUIS\Documents\GitHub\problem-set-4-hiro-luis\data")
path_data_2016 = os.path.join(base, "pos2016.csv")

# Load 2016 dataset
df_pos2016 = pd.read_csv(path_data_2016, encoding="utf-8")

# Define function to subset a dataset for short-term hospitals


def subset_short_term(df):
    df_result = df[(df["PRVDR_CTGRY_CD"] == 1) &
                   (df["PRVDR_CTGRY_SBTYP_CD"] == 1)]
    return df_result


# Create short-term hospitals subsets for 2016 dataset
df_pos2016_short = subset_short_term(df_pos2016)
```

a.

```
# Count the number of observations
print(len(df_pos2016_short))
```

```
7245
```

This subset for short-term hospitals in 2016 includes 7245 hospitals.

This number seems not making sense, as it occupies a half of number of whole hospitals though it is just one subtype.

```
# Count the number of observations
print(len(df_pos2016))
```

```
141557
```

b. We couldn't find references which includes the exact number of short-term hospitals at the end of 2016, but the 2016 CMS statistics specifies the number as 3,436 for the Dec 31, 2015, which is the most recent data available in the search result(https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/CMS-Statistics-Reference-Booklet/Downloads/2016_CMS_Stats.pdf).

Considering the number 3,436 at the end of 2015, 7245 is drastically larger, even considering the change in one year.

As far as we could find, there seems no substantial policy change in 2016, which could potentially change the number of healthcare providers drastically.

As both the dataset we use and the cross-reference are both published by Centers for Medicare & Medicaid Services, it might be plausible to assume their credibilities. And as the dataset dictionary says nothing about change in definition of the provider categories/sub-categories. Then one reason we can consider is, for instance, the change in the data gathering methods.

3.

```
# Load 2017-2019 datasets
path_data_2017 = os.path.join(base, "pos2017.csv")
path_data_2018 = os.path.join(base, "pos2018.csv")
path_data_2019 = os.path.join(base, "pos2019.csv")
df_pos2017 = pd.read_csv(path_data_2017, encoding="utf-8")
df_pos2018 = pd.read_csv(path_data_2018, encoding="latin1")
df_pos2019 = pd.read_csv(path_data_2019, encoding="latin1")

# Clean 2018, 2019 datasets: fix colname


def fix_col(df):
    col = df.columns.tolist()
    col[0] = "PRVDR_NUM"
    df.columns = col
    return df


df_pos2018 = fix_col(df_pos2018)
df_pos2019 = fix_col(df_pos2019)
```

```
# Create short-term hospitals subsets for 2017-2019 datasets
df_pos2017_short = subset_short_term(df_pos2017)
df_pos2018_short = subset_short_term(df_pos2018)
df_pos2019_short = subset_short_term(df_pos2019)


# Summarize the number of observations for each subset
df_summary_obs = pd.DataFrame(
    {"year": list(range(2016, 2020, 1)),
     "observations": [len(df_pos2016_short), len(df_pos2017_short),
      ↪  len(df_pos2018_short), len(df_pos2019_short)]
     }
)

df_summary_obs
```

	year	observations
0	2016	7245
1	2017	7260
2	2018	7277
3	2019	7303

The number for q4 in 2017/2018/2019 is on similar level with 2016, which are unmatched with the cross-reference.

```
# Add "DATA_YEAR" column to each subsets
df_pos2016_short["DATA_YEAR"] = 2016
df_pos2017_short["DATA_YEAR"] = 2017
df_pos2018_short["DATA_YEAR"] = 2018
df_pos2019_short["DATA_YEAR"] = 2019

# Concaterate the subsets
df_pos_append = pd.concat(
    [df_pos2016_short, df_pos2017_short, df_pos2018_short, df_pos2019_short])

# Turn off the row number limit
alt.data_transformers.disable_max_rows()

# Plot the number of observation for each subset's year
```

```
chart_obs = alt.Chart(df_pos_append).mark_line().encode(
    alt.X("DATA_YEAR:N", title="Year (Q4)", axis=alt.Axis(labelAngle=0)),
    alt.Y("count()", title="Number of Observation",
          scale=alt.Scale(domain=[7000, 8000]))
).properties(width=250, height=250)
chart_obs
```

```
alt.Chart(...)
```

4. a.

```
# Summarize the number of unique providers for each subset
df_summary_unique = pd.DataFrame(
    {"year": list(range(2016, 2020, 1)),
     "unique_hospitals":  [len(df_pos2016_short["PRVDR_NUM"].value_counts()),
      ↪  len(df_pos2017_short["PRVDR_NUM"].value_counts()),
      ↪  len(df_pos2018_short["PRVDR_NUM"].value_counts()),
      ↪  len(df_pos2019_short["PRVDR_NUM"].value_counts())]
     }
)

df_summary_unique

# Plot the number of unique hospitals for each subset's year
chart_unique = alt.Chart(df_summary_unique).mark_line().encode(
    alt.X("year:N", title="Year (Q4)", axis=alt.Axis(labelAngle=0)),
    alt.Y("unique_hospitals:Q", title="Number of unique hospitals",
          scale=alt.Scale(domain=[7000, 8000]))
).properties(width=250, height=250)
chart_unique
```

```
alt.Chart(...)
```

b.
The numbers of unique hospitals are the same as the numbers of observations for all years we examined. This suggests that this dataset is structured so that each row contains parameters/attributions of each hospital.

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

1.

```
# Make termination status per year a separate column
df_pos2016_short.rename(
    columns={"PGM_TRMNTN_CD": "PGM_TRMNTN_CD_2016"}, inplace=True)
df_pos2017_short.rename(
    columns={"PGM_TRMNTN_CD": "PGM_TRMNTN_CD_2017"}, inplace=True)
df_pos2018_short.rename(
    columns={"PGM_TRMNTN_CD": "PGM_TRMNTN_CD_2018"}, inplace=True)
df_pos2019_short.rename(
    columns={"PGM_TRMNTN_CD": "PGM_TRMNTN_CD_2019"}, inplace=True)
```

```
# Create new merged df
df_list = [df_pos2017_short, df_pos2018_short, df_pos2019_short]
df_pos_termination = df_pos2016_short

for df in df_list:
    col_name = df.columns[-2]
    df_pos_termination = df_pos_termination.merge(
        df[["PRVDR_NUM", col_name]], on="PRVDR_NUM", how="outer")

df_pos_termination = df_pos_termination.drop(columns=["DATA_YEAR"])
```

```
# Add suspected termination year
def add_termination_year(row):
    if row["PGM_TRMNTN_CD_2016"] != 0:
        return None
    elif row["PGM_TRMNTN_CD_2017"] != 0:
        return 2017
    elif row["PGM_TRMNTN_CD_2018"] != 0:
        return 2018
    elif row["PGM_TRMNTN_CD_2019"] != 0:
        return 2019
    return None


df_pos_termination["TRMNTN_YEAR"] = df_pos_termination.apply(
    add_termination_year, axis=1)
```

```
# Create list of closed hospitals that were active in 2016
df_pos_active2016 =
 ↪  df_pos_termination[df_pos_termination["PGM_TRMNTN_CD_2016"] == 0]
```

```
closed_hospitals = []
closed_hospitals =
 ↪  df_pos_active2016["FAC_NAME"][df_pos_termination["TRMNTN_YEAR"] >
 ↪  2016].tolist(
)

len(closed_hospitals)
```

```
174
```

174 hospitals

```
# Create list of closed hospital
df_pos_closed =
 ↪  df_pos_active2016.loc[df_pos_active2016["TRMNTN_YEAR"].notna()]
df_pos_closed = df_pos_closed[["PRVDR_NUM",
                               "FAC_NAME", "ZIP_CD", "TRMNTN_YEAR"]]
```

2.

```
# Sort by hospital name
df_pos_closed.sort_values(by="FAC_NAME", inplace=True)
df_pos_closed.head(10)
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	TRMN
168	030001	ABRAZO MARYVALE CAMPUS	85031.0	2017.0
568	050196	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230.0	2017.0
4852	360151	AFFINITY MEDICAL CENTER	44646.0	2018.0
4356	330189	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	12208.0	2017.0
6273	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2017.0
3596	250159	ALLIANCE LAIRD HOSPITAL	39365.0	2019.0
4990	370032	ALLIANCEHEALTH DEACONESS	73112.0	2019.0
1384	100240	ANNE BATES LEACH EYE HOSPITAL	33136.0	2019.0
1044	060036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050.0	2017.0
3975	290006	BANNER CHURCHILL COMMUNITY HOSPITAL	89406.0	2017.0

3. a.

```
# Revert column names
df_pos2016_short.rename(
    columns={"PGM_TRMNTN_CD_2016": "PGM_TRMNTN_CD"}, inplace=True)
df_pos2017_short.rename(
    columns={"PGM_TRMNTN_CD_2017": "PGM_TRMNTN_CD"}, inplace=True)
df_pos2018_short.rename(
    columns={"PGM_TRMNTN_CD_2018": "PGM_TRMNTN_CD"}, inplace=True)
df_pos2019_short.rename(
    columns={"PGM_TRMNTN_CD_2019": "PGM_TRMNTN_CD"}, inplace=True)
```

```
# Create dfs active and closed hospitals
df_active2016 = df_pos2016_short[df_pos2016_short["PGM_TRMNTN_CD"] == 0][[
    "PRVDR_NUM", "FAC_NAME", "ZIP_CD"]]

df_suspect_closures = pd.DataFrame()
df_active_total = df_active2016

for year, df_year in zip([2017, 2018, 2019], [df_pos2017_short,
 ↪  df_pos2018_short, df_pos2019_short]):
    active = df_year[df_year["PGM_TRMNTN_CD"] == 0]["PRVDR_NUM"]
    closures = df_active_total[~df_active_total["PRVDR_NUM"].isin(active)]
    closures["TRMNTN_YEAR"] = year

    df_suspect_closures = pd.concat([df_suspect_closures, closures])

    df_active_total = df_active_total[df_active_total["PRVDR_NUM"].isin(
        active)]

df_closed = df_suspect_closures.drop_duplicates(subset=["PRVDR_NUM"])
```

```
# Create new df of potential mergers
df_by_year = {2016: df_pos2016_short, 2017: df_pos2017_short,
              2018: df_pos2018_short, 2019: df_pos2019_short}

potential_mergers = []

for _, row in df_closed.iterrows():
    zip_code = row["ZIP_CD"]
    closure_year = row["TRMNTN_YEAR"]

    if closure_year > 2016 and closure_year <= 2019:
```

```
        df_prev_year = df_by_year[closure_year - 1]
        df_current_year = df_by_year[closure_year]

        active_prev_year = len(df_prev_year[(df_prev_year["ZIP_CD"] ==
↪  zip_code) & (
            df_prev_year["PGM_TRMNTN_CD"] == 0)])

        active_current_year = len(df_current_year[(
            df_current_year["ZIP_CD"] == zip_code) &
↪  (df_current_year["PGM_TRMNTN_CD"] == 0)])

        if active_current_year >= active_prev_year:
            potential_mergers.append(row)

df_potential_mergers = pd.DataFrame(potential_mergers)
len(df_potential_mergers)
```

```
8
```

There are 8 hospital closures that fall under the definition of potential mergers.

```
# Calculate number of hospitals remaining
len(closed_hospitals) - len(df_potential_mergers)
```

```
166
```

There are 166 terminated hospitals left.

c.

```
# Sort data
df_closed_corrected = df_pos_closed[~df_pos_closed["PRVDR_NUM"].isin(
    df_potential_mergers["PRVDR_NUM"])]

df_closed_corrected.sort_values(by="FAC_NAME", inplace=True)
df_closed_corrected.head(10)
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	TRMN
168	030001	ABRAZO MARYVALE CAMPUS	85031.0	2017.0
568	050196	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230.0	2017.0
4852	360151	AFFINITY MEDICAL CENTER	44646.0	2018.0

	PRVDR_NUM	FAC_NAME	ZIP_CD	TRMN
4356	330189	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	12208.0	2017.0
6273	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662.0	2017.0
3596	250159	ALLIANCE LAIRD HOSPITAL	39365.0	2019.0
4990	370032	ALLIANCEHEALTH DEACONESS	73112.0	2019.0
1384	100240	ANNE BATES LEACH EYE HOSPITAL	33136.0	2019.0
1044	060036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050.0	2017.0
3975	290006	BANNER CHURCHILL COMMUNITY HOSPITAL	89406.0	2017.0

Download Census zip code shapefile (10 pt)

1. a. The file types and the types of information for each of those included in the data is below;

   - .shp : feature geometrics
   - .shx : positional index
   - .dbf : attribute information
   - .prj : description about the CRS
   - .xml : metadata for the data

   b.

```
# Define a function to get the file size
def get_filesize(path):
    size = os.path.getsize(path)
    if size > (1024 ** 2):
        size = size / (1024 ** 2)
        result = f"{round(size,1)}MB"
    elif size > 1024:
        size = size / 1024
        result = f"{round(size,1)}KB"
    else:
        result = f"{round(size,1)}bytes"
    return result

# Get the file sizes for each file
shape_filetypes = ["shp", "shx", "dbf", "prj", "xml"]

paths_shape = [os.path.join(base, "gz_2010_us_860_00_500k",
 ↪  f"gz_2010_us_860_00_500k.{x}") for x in shape_filetypes]
```

```
shape_filesizes = pd.DataFrame(
  {"file": [f"gz_2010_us_860_00_500k.{x}" for x in shape_filetypes],
  "size":[get_filesize(x) for x in paths_shape]
  }
)

print(shape_filesizes)
```

```
                         file      size
0  gz_2010_us_860_00_500k.shp   798.7MB
1  gz_2010_us_860_00_500k.shx   258.8KB
2  gz_2010_us_860_00_500k.dbf     6.1MB
3  gz_2010_us_860_00_500k.prj  165bytes
4  gz_2010_us_860_00_500k.xml    15.3KB
```

2.

```
# Load the shp file
path_shp = os.path.join(
    base, r"gz_2010_us_860_00_500k\gz_2010_us_860_00_500k.shp")
df_shp = gpd.read_file(path_shp)
```

```
# Specify the variable for ZIP code in the dataset
df_shp.head
df_shp["ZCTA5"] = df_shp["ZCTA5"].astype(int)
```

From the metadata in the XML file, we can restrict the zip code by filtering the column "ZCTA5". Now given that Texas zip codes start from 75-79, we can operate the restriction as below;

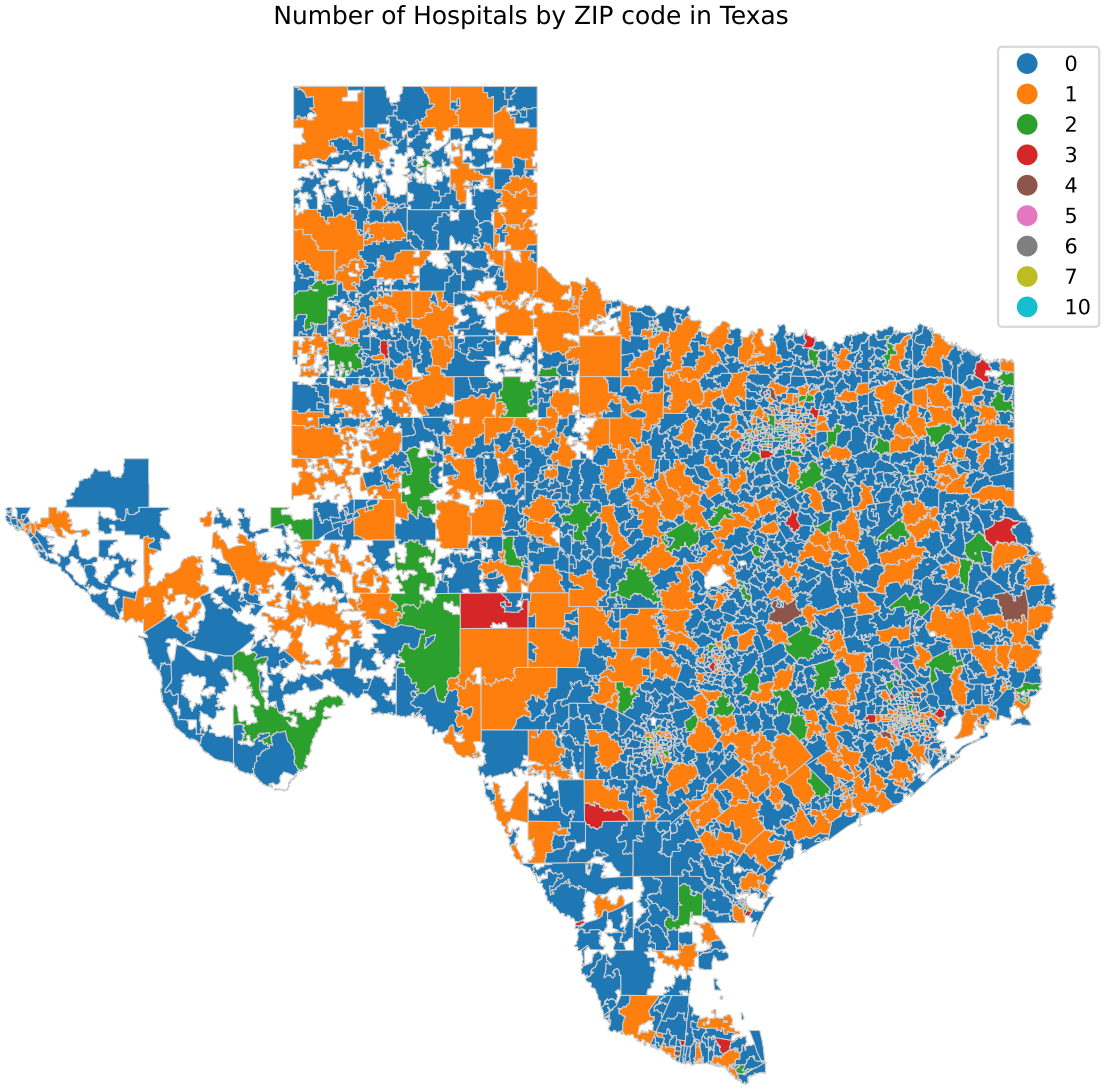
```
# Restrict dataset to Texas ZIP codes
df_shp_tx = df_shp[(df_shp["ZCTA5"] >= 75000) & (df_shp["ZCTA5"] < 80000)]

# Extract a list of Texas ZIP codes
zip_tx = df_shp_tx["ZCTA5"].value_counts().index.astype(int).to_list()

# Summarize the number of hospitals for each zip code on 2016 subset of POS
 ↪  data
df_pos2016_short_tx =
 ↪  df_pos2016_short[df_pos2016_short["ZIP_CD"].isin(zip_tx)]
```

```
summary_pos2016_short_tx = df_pos2016_short_tx.groupby(
    "ZIP_CD").size().reset_index()
summary_pos2016_short_tx.columns = ["ZIP_CD", "HOSPITAL_NUM"]
summary_pos2016_short_tx["ZIP_CD"] =
 ↪  summary_pos2016_short_tx["ZIP_CD"].astype(
    int)

print(summary_pos2016_short_tx)
```

```
     ZIP_CD  HOSPITAL_NUM
0     75001             1
1     75010             2
2     75013             1
3     75020             3
4     75028             1
..      ...           ...
485   79905             1
486   79915             1
487   79925             1
488   79936             1
489   79938             1

[490 rows x 2 columns]
```

```
# Merge the dataframes
df_shp_tx_merge = df_shp_tx.merge(
    summary_pos2016_short_tx,
    left_on="ZCTA5",
    right_on="ZIP_CD",
    how="outer"
)

# Replace NAs in the number of hospitals with the value 0
df_shp_tx_merge["HOSPITAL_NUM"] = df_shp_tx_merge["HOSPITAL_NUM"].fillna(
    0).astype(int).astype("category")

# Plot the number of hospital in TX on the map
df_shp_tx_merge.plot(column="HOSPITAL_NUM", linewidth=0.5,
                     edgecolor="0.8", legend=True, figsize=(10, 10))

plt.axis("off")
```

```
plt.title("Number of Hospitals by ZIP code in Texas")
plt.show
```



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

1.

```
# Create a centroid column for all ZIP codes
zips_all_centroids = df_shp.copy()

zips_all_centroids["geometry"] = zips_all_centroids.centroid
```

Dimentions:

```
print(zips_all_centroids.shape)
```

```
(33120, 6)
```

33120 rows and 6 columns.

Columns: the columuns in the dataset is below;

```
print(zips_all_centroids.columns)
```

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

And from the metadata, the meaning for each column is;

- "GEO_ID": A unique identifier for each geographic entity
- "ZCTA5": The 5-digit ZIP Code Tabulation Area code
- "NAME": The name of the geographic entity
- "LSAD": Legal/Statistical Area Description code
- "CENSUSAREA": The area of the entity in square miles
- "geometry": the geometries (polygons) for each area

2.

```
# Create subset for all zip codes in Texas
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids["ZCTA5"].isin(zip_tx)]

# Count the unique zip codes in the subset
print(len(zips_texas_centroids["ZCTA5"].unique()))
```

```
1935
```

The Texas subset includes 1935 unique zip codes.

From the map in the wikipedia page cited in the PDF, the border states has the prefixes for Zip code of “70”, “71”, “72”, “73”, “74”, “87”, “88”. Thus;

```
# Specify zipcodes for border states
zips_borderstates_centroids =
 ↪  zips_all_centroids[((zips_all_centroids["ZCTA5"] >= 70000) & (
    zips_all_centroids["ZCTA5"] < 75000)) | ((zips_all_centroids["ZCTA5"] >=
 ↪  87000) & (zips_all_centroids["ZCTA5"] < 89000))]

# Extract a list of Texas ZIP codes
zip_borderstates = zips_borderstates_centroids["ZCTA5"].value_counts(
).index.astype(int).to_list()

zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids["ZCTA5"].isin(
    tuple(zip_tx + zip_borderstates))]

# Count the unique zip codes in the subset
print(len(zips_texas_borderstates_centroids["ZCTA5"].unique()))
```

```
4057
```

The Texas and border-states subset includes 4057 unique zip codes.

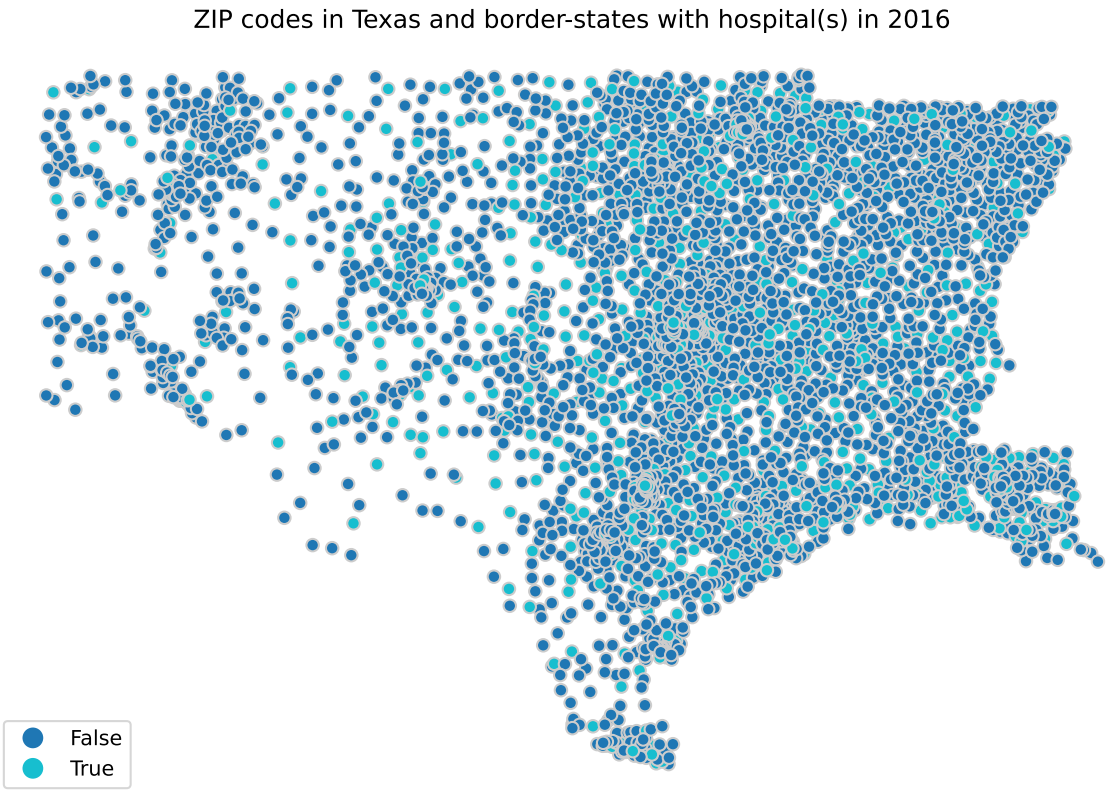
3.

```
# Summarize the number of hospitals for each zip code on 2016 subset of POS
 ↪  data
df_pos2016_short_tx_borderstates =
 ↪  df_pos2016_short[df_pos2016_short["ZIP_CD"].isin(
    zip_tx + zip_borderstates)]
summary_pos2016_short_tx_borderstates =
 ↪  df_pos2016_short_tx_borderstates.groupby(
    "ZIP_CD").size().reset_index()
summary_pos2016_short_tx_borderstates.columns = ["ZIP_CD", "HOSPITAL_NUM"]
summary_pos2016_short_tx_borderstates["ZIP_CD"] =
 ↪  summary_pos2016_short_tx_borderstates["ZIP_CD"].astype(
    int)

# Merge the dataframes
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
```

```
    summary_pos2016_short_tx_borderstates,
    left_on="ZCTA5",
    right_on="ZIP_CD",
    how="outer"
)

print(zips_withhospital_centroids.head(5))

# Replace NAs in the number of hospitals with the value 0
zips_withhospital_centroids["HOSPITAL_NUM"] =
 ↪  zips_withhospital_centroids["HOSPITAL_NUM"].fillna(
    0).astype(int)

zips_withhospital_centroids["HOSPITAL"] = [
    x != 0 for x in zips_withhospital_centroids["HOSPITAL_NUM"]]

print(zips_withhospital_centroids)

# Plot a choropleth of directly affected ZIP codes
zips_withhospital_centroids.plot(
    column="HOSPITAL", linewidth=1, edgecolor="0.8", legend=True,
 ↪  figsize=(10, 10))
plt.axis("off")
plt.title("ZIP codes in Texas and border-states with hospital(s) in 2016")
plt.show
```

```
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US70001  70001  70001  ZCTA5       6.003
1  8600000US70002  70002  70002  ZCTA5       3.219
2  8600000US70003  70003  70003  ZCTA5       7.019
3  8600000US70005  70005  70005  ZCTA5       4.180
4  8600000US70006  70006  70006  ZCTA5       2.589

                     geometry   ZIP_CD  HOSPITAL_NUM
0  POINT (-90.16718 29.98372)  70001.0           2.0
1   POINT (-90.1622 30.01053)  70002.0           1.0
2  POINT (-90.21397 29.99864)      NaN           NaN
3  POINT (-90.13386 29.99946)      NaN           NaN
4  POINT (-90.19168 30.01399)  70006.0           3.0
              GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0     8600000US70001  70001  70001  ZCTA5       6.003
1     8600000US70002  70002  70002  ZCTA5       3.219
```

```
2     8600000US70003  70003  70003  ZCTA5       7.019
3     8600000US70005  70005  70005  ZCTA5       4.180
4     8600000US70006  70006  70006  ZCTA5       2.589
...              ...    ...    ...    ...         ...
4052  8600000US88430  88430  88430  ZCTA5     367.397
4053  8600000US88431  88431  88431  ZCTA5     209.703
4054  8600000US88434  88434  88434  ZCTA5     374.429
4055  8600000US88435  88435  88435  ZCTA5    1826.306
4056  8600000US88436  88436  88436  ZCTA5     131.986

                         geometry   ZIP_CD  HOSPITAL_NUM  HOSPITAL
0      POINT (-90.16718 29.98372)  70001.0             2      True
1       POINT (-90.1622 30.01053)  70002.0             1      True
2      POINT (-90.21397 29.99864)      NaN             0     False
3      POINT (-90.13386 29.99946)      NaN             0     False
4      POINT (-90.19168 30.01399)  70006.0             3      True
...                           ...      ...           ...       ...
4052  POINT (-103.18124 35.59036)      NaN             0     False
4053  POINT (-104.27796 35.17597)      NaN             0     False
4054    POINT (-103.30241 35.186)      NaN             0     False
4055  POINT (-104.69762 34.91293)  88435.0             1      True
4056   POINT (-103.12167 36.1315)      NaN             0     False

[4057 rows x 9 columns]
```



We merged the GeoDataFrame "zips_texas_borderstates_centroids" with the pos data for short-term hospitals in 2016 merging on Zip codes of the area , filtering to the ZIP codes whose "HOSPITAL_NUM" is not 0 (those which had at least one hospital in 2016).

4. a.

```
# Create a subset of 10 zip codes in TX for testing the calculation of
 ↪  distance
zips_texas_test = zips_texas_centroids.head(10)

# Start timer
start = time.time()

# Join the dataframes
join_test = gpd.sjoin_nearest(
    zips_texas_test,
    zips_withhospital_centroids,
    how = "inner",
    distance_col = "distance"
```

```
)

# Find out nearest distance for each TX ZIP code
join_test_result =
 ↪  join_test.loc[join_test.groupby("ZCTA5_left")["distance"].idxmin()]

# Count the lap time
time_count = time.time() - start

# Show the lap time once the row number is correct
print("This time, to join the datasets, it took approximately",
 round(time_count, 2), "seconds.")

time_estimate = (time_count * 1935 / 10)

print(f"As there are 1935 ZIP codes in Texas, we can estimate that it would
 ↪  take approximately {round(time_count, 2)} * 1935 / 10 =
 ↪  {round(time_estimate, 2)} seconds.")
```

```
This time, to join the datasets, it took approximately 0.04 seconds.
As there are 1935 ZIP codes in Texas, we can estimate that it would take
approximately 0.04 * 1935 / 10 = 7.71 seconds.
```

b.

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

# Join the dataframes
join_tx = gpd.sjoin_nearest(
    zips_texas_centroids,
    zips_withhospital_centroids,
    how="right",
    distance_col="distance"
)

# Find out nearest distance for each TX ZIP code
join_tx_min = join_tx.loc[join_tx.groupby("ZCTA5_left")["distance"].idxmin()]

# Count the lap time
time_count = time.time() - start
```

```
# Show the lap time once the row number is correct
print("To join the datasets, it took approximately",
      round(time_count, 2), "seconds, which is the quite same level as the
       ↪  test, against our estimation above.")
```

```
To join the datasets, it took approximately 0.06 seconds, which is the quite
same level as the test, against our estimation above.
```

```
c. As the .prj file notes, its unit is a degree (of latitude). A degree of
latitude is constantly approximately convertible to 69 miles, while longitude
varies depend on the location, but still maximum about 69
miles.(https://www.thoughtco.com/degree-of-latitude-and-longitude-distance-4070616)

Thus, when interpret the dataset according to the .prj file, it would be
suitable to estimate 1 degree = 69 miles.
```

5. a. The unit is a degree.

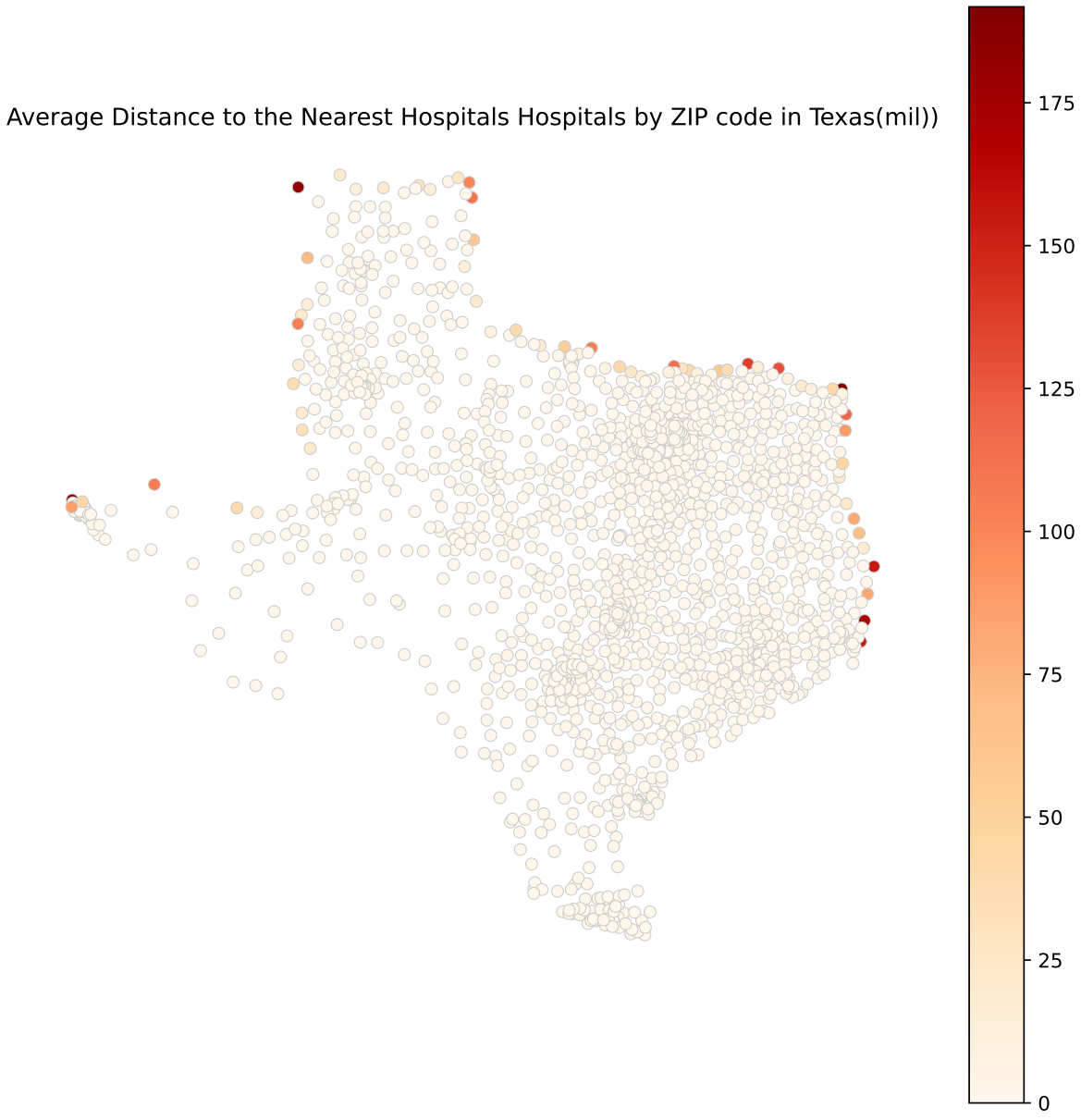
   b.

```
# Summarize average distance from the nearest hospital (by ZIP code where the
 ↪  hospital locates) for each ZIP code
join_tx_avg = join_tx.groupby("ZCTA5_left")["distance"].agg(
    "mean").reset_index(name="distance")

# Convert to mile
join_tx_avg["distance"] = join_tx_avg["distance"] * 69

avg_join_tx_avg = join_tx_avg["distance"].mean()

print(avg_join_tx_avg)
```

```
1.8753916211405324

    This intuitively makes sense.

c.
```

```
# Merge the dataframes
zips_texas_centroids_merge = zips_texas_centroids.merge(
    join_tx_avg,
    left_on="ZCTA5",
    right_on="ZCTA5_left",
```

```
    how="right"
)

# Plot the number of hospital in TX on the map
zips_texas_centroids_merge.plot(column="distance", linewidth=0.5,
                                edgecolor="0.8", cmap="OrRd", legend=True,
 ↪  figsize=(10, 10))

plt.axis("off")
plt.title(
    "Average Distance to the Nearest Hospitals Hospitals by ZIP code in
     ↪  Texas(mil))")
plt.show
```



Effects of closures on access in Texas (15 pts)

1.

```
# Subset the appended dataset
df_pos_append["ZIP_CD"] = df_pos_append["ZIP_CD"].astype(int)
```

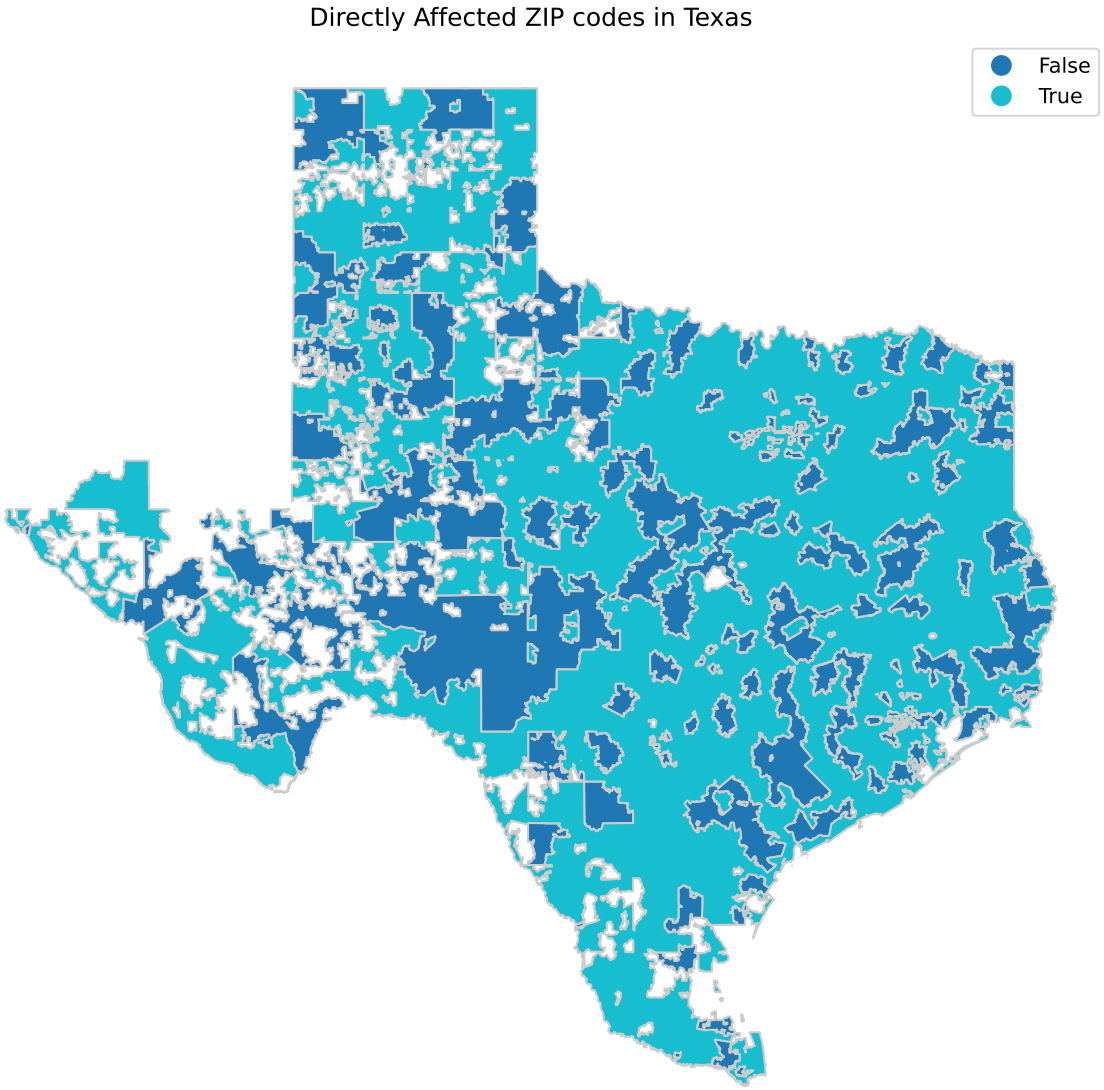
```
df_pos_append_tx = df_pos_append[(df_pos_append["ZIP_CD"] >= 75000) & (
    df_pos_append["ZIP_CD"] < 80000)]

# Summarize the number of directly affected zip codes
summary_direct = df_pos_append_tx[df_pos_append_tx["PGM_TRMNTN_CD"] !=
 ↪  0].groupby(
    ["ZIP_CD", "FAC_NAME"]).size().reset_index()

summary_direct = summary_direct.groupby("ZIP_CD").size().reset_index()
summary_direct.columns = ["ZIP code", "Number of closures"]

print(summary_direct)
```

```
     ZIP code  Number of closures
0       75010                   1
1       75020                   2
2       75041                   1
3       75042                   3
4       75051                   2
..        ...                 ...
339     79901                   1
340     79902                   7
341     79904                   1
342     79915                   1
343     79925                   1

[344 rows x 2 columns]
```

2.

```
# Merge the summary to the GeoDataFrame
df_shp_tx_merge = df_shp_tx_merge.merge(
    summary_direct,
    left_on="ZCTA5",
    right_on="ZIP code",
    how="outer"
)

df_shp_tx_merge["DIRECT"] = df_shp_tx_merge["Number of closures"].apply(
    pd.isna)

# Dissolve the GeoDataFrame by the parameter for directly affected ZIP codes
```

```
dissolved = df_shp_tx_merge[["ZCTA5", "geometry", "DIRECT"]].dissolve(
    by="DIRECT", aggfunc="sum").reset_index()
dissolved

# Plot a choropleth of directly affected ZIP codes
dissolved.plot(column="DIRECT", linewidth=1, edgecolor="0.8",
               legend=True, figsize=(10, 10))

plt.axis("off")
plt.title("Directly Affected ZIP codes in Texas")
plt.show
```




```
# Count the number of directly affected ZIP codes
print(len(summary_direct))
```

```
344
```

There are 344 directly affected zip codes in Texas.

3.

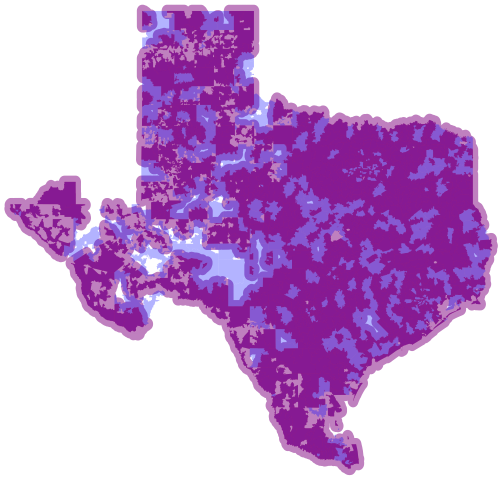
```
# Create a GeoDataFrame of the directly affected zip codes
direct_zips = dissolved[dissolved["DIRECT"] == 1]

# Create a 10-mile buffer around direct_zips
buf_degree = 10 / 69

buffer_10 = direct_zips.copy()
buffer_10["geometry"] = buffer_10.geometry.buffer(buf_degree)

# Spatial join with the overall Texas zip code shapefile
indirect_zips = gpd.sjoin(df_shp_tx, buffer_10,
                          how="inner", predicate="intersects")

print(len(indirect_zips))
```

```
1935
```

There are 1935 indirectly affected zipcodes in Texas.

4.

```
fig, ax = plt.subplots()
direct_zips.plot(ax=ax, color="red", alpha=0.7,
 ↪  label="Direct").set_axis_off()
buffer_10.plot(ax=ax, color="purple", alpha=0.5,
               label="10-mile").set_axis_off()
indirect_zips.plot(ax=ax, color="blue", alpha=0.3, label="Indirect")

plt.show()
```



Reflecting on the exercise (10 pts)

(Partner 1: Hiro) One possible reason for the number of active hospitals in a zip code to not decrease is the opening of a totally new hospital. The "first-pass" method does not distinguish between a potential merger (where the hospital name/address is the same) and the opening of a totally new hospital. In such a case, a valid closure would not be counted. To get a more accurate confirmation of hospital closures in using the "first-pass" method, we could also look at a secondary datapoint such as address which is harder to change compared to hospital name. Another possible way is to add more details in the termination status column. For example, a code specific for mergers could be added that would make it easier to filter these entries.

(Partner 2: Luis)

As our research design simply focus on geographical distance (impact of closures to those who arewithin the same ZIP codes, or within 10 miles distance), it might not reflect the real usage of hospitals by residents. For instance, as we ignore their preferences or genres of hospitals correlated to diseases categories, thus some people might have been using specific hospitals which is not the nearest ones from their place. Also, we ignore the transportation and the concentration of hospitals (hospitals might be originally concentrated on large city and frequently used by residents living broadly around the city). To improve this research from those reflection, we can merge additional attributes to the GeoDataFrame, such as the population for each ZIP code, the concentration of hospitals (= number of hospitals per ZIP code), and if possible, parameter of whether major transportation (railway, bus, highway) exists in the ZIP code or not.