Problem Set 4

Ben Schiffman and Hallie Lovin

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points.

Style Points (10 pts)

Submission Steps (10 pts)

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

```
#import necessary packages
import pandas as pd
import altair as alt
alt.renderers.enable("png")
```

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

1. The variables that I pulled were: PRVDR_CTGRY_SBTYP_CD - shows the subtype of the provider PRVDR_CTGRY_CD - shows the type of provider CITY_NAME - shows the city that the provider is physically located in FAC_NAME - shows the name of the provider that is providing services PRVDR_NUM - shows the CMS certification number STATE_CD - shows the state abbreviation ST_ADR - shows the street address where the provider is located PGM_TRMNTN_CD - shows the termination status of the provider TRMNTN_EXPRTN_DT - shows the date that the provider was terminated ZIP_CD - shows the zip code of the providers physical address FIPS_STATE_CD - shows the FIPS state code CBSA_URBN_RRL_IND - shows if the town is urban or rural
2.

```
#import the data
import os

path = r"/Users/hallielovin/Documents/GitHub/problem-set-4-ben_hallie"
ben_path = r"/Users/benschiffman/Desktop/Python 2/problem-set-4-ben_hallie"
pos2016 = r"pos2016.csv"

pos2016_df = pd.read_csv(os.path.join(path, pos2016))

#convert zip code to a string
pos2016_df["ZIP_CD"] = pos2016_df["ZIP_CD"].astype(str).str.replace(".0", "
 ↪  ", regex = False)

pos2016_df.head()
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CITY_NAME	FAC_NAME
0	1.0	1	DOTHAN	SOUTHEAST ALABAMA
1	1.0	1	BRIDGEPORT	NORTH JACKSON HOSPI
2	1.0	1	BOAZ	MARSHALL MEDICAL CE
3	1.0	1	FLORENCE	ELIZA COFFEE MEMORI
4	1.0	1	OPP	MIZELL MEMORIAL HOS

```
#subset data to only include those with provider type code 1 and subtype code
 ↪  1
pos2016_df = pos2016_df[(pos2016_df["PRVDR_CTGRY_CD"] == 1) &
 ↪  (pos2016_df["PRVDR_CTGRY_SBTYP_CD"] == 1)]
```

```
#determine how many hospitals are reported in the data
len(pos2016_df)
```

7245

a. There are 7,245 hospitals reported in the data that are considered "short term" hospitals. According to the brief by the Kaiser Family Foundation, there are around 5,000 short term, acute care hospitals in the US. That said, the numbers in this data seem much higher.

b. After looking at more sources, a CMS report showed that there were 3,436 medicaid participating hospitals in 2016. The reasons that all of these numbers may differ is because some hospitals may have closed which we have not yet accounted for in our data. Additionally, the Kaiser Foundation is looking at all hospitals that are short term and not filtering out those that bill medicare and medicaid.

3.

```
#load in the data from 2017, 2018, and 2019

##2017
pos2017 = r"pos2017.csv"

pos2017_df = pd.read_csv(os.path.join(path, pos2017))

#convert zips to string
pos2017_df["ZIP_CD"] = pos2017_df["ZIP_CD"].astype(str).str.replace(".0", "
↪  ", regex = False)

pos2017_df.head()
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CITY_NAME	FAC_NAME
0	1.0	1	DOTHAN	SOUTHEAST ALABAMA
1	1.0	1	BRIDGEPORT	NORTH JACKSON HOSPI
2	1.0	1	BOAZ	MARSHALL MEDICAL CE
3	1.0	1	FLORENCE	ELIZA COFFEE MEMORI
4	1.0	1	OPP	MIZELL MEMORIAL HOS

```
##2018-- I was getting an error (UnicodeDecodeError) so I had to get chatgpt
↪  to help
pos2018 = r"pos2018.csv"

pos2018_df = pd.read_csv(os.path.join(path, pos2018), encoding="ISO-8859-1")

#convert zip to string
pos2018_df["ZIP_CD"] = pos2018_df["ZIP_CD"].astype(str).str.replace(".0", "
↪  ", regex = False)

pos2018_df.head()
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CITY_NAME	FAC_NAME
0	1.0	1	DOTHAN	SOUTHEAST ALABAMA
1	1.0	1	BRIDGEPORT	NORTH JACKSON HOSPI
2	1.0	1	BOAZ	MARSHALL MEDICAL CE
3	1.0	1	FLORENCE	ELIZA COFFEE MEMORI
4	1.0	1	OPP	MIZELL MEMORIAL HOS

```
##2019-- I was getting an error (UnicodeDecodeError) so I had to get chatgpt
 ↪  to help
pos2019 = r"pos2019.csv"

pos2019_df = pd.read_csv(os.path.join(path, pos2019), encoding="ISO-8859-1")

#convert zip to string
pos2019_df["ZIP_CD"] = pos2019_df["ZIP_CD"].astype(str).str.replace(".0", "
 ↪  ", regex = False)

pos2019_df.head()
```

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CITY_NAME	FAC_NAME
0	1.0	1	DOTHAN	SOUTHEAST ALABAMA
1	1.0	1	BRIDGEPORT	NORTH JACKSON HOSPI
2	1.0	1	BOAZ	MARSHALL MEDICAL CE
3	1.0	1	FLORENCE	NORTH ALABAMA MEDI
4	1.0	1	OPP	MIZELL MEMORIAL HOS

```
#subset out the dataframes so only the code 1's are shown
##2017
pos2017_df = pos2017_df[(pos2017_df["PRVDR_CTGRY_CD"] == 1) &
 ↪  (pos2017_df["PRVDR_CTGRY_SBTYP_CD"] == 1)]
```

```
##2018
pos2018_df = pos2018_df[(pos2018_df["PRVDR_CTGRY_CD"] == 1) &
 ↪  (pos2018_df["PRVDR_CTGRY_SBTYP_CD"] == 1)]
```

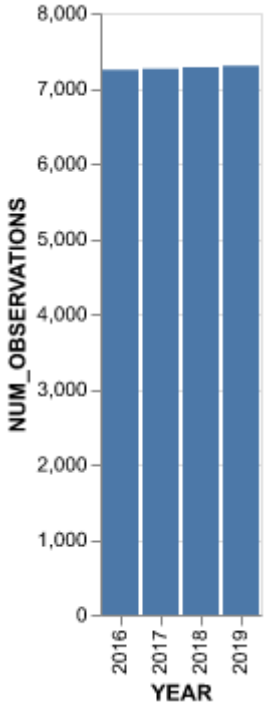
```
##2019
pos2019_df = pos2019_df[(pos2019_df["PRVDR_CTGRY_CD"] == 1) &
 ↪  (pos2019_df["PRVDR_CTGRY_SBTYP_CD"] == 1)]
```

```
#append all of the data sets together into one pos data set using the CMS
 ↪  certification number (prvdr_num)

##add a year column to each dataset
pos2016_df["YEAR"] = 2016
pos2017_df["YEAR"] = 2017
pos2018_df["YEAR"] = 2018
pos2019_df["YEAR"] = 2019
```

```
##make the large dataframe
pos_df = pd.concat([pos2016_df, pos2017_df, pos2018_df, pos2019_df],
 ↪  ignore_index = True)
```

```
#group the dataset by year and take the count of how many observations are in
 ↪  each year
pos_years_count = pos_df.groupby("YEAR").agg(
  NUM_OBSERVATIONS = ("YEAR", "size")
).reset_index()
```

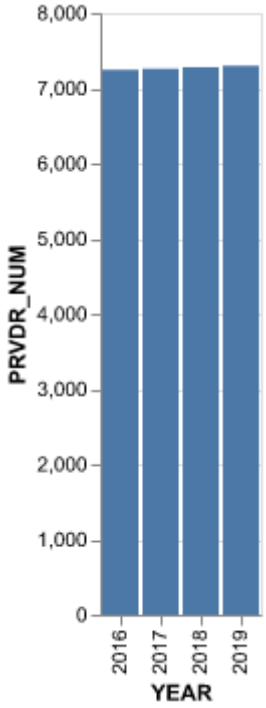
```
#make a plot of the number of observations by year
alt.Chart(pos_years_count).mark_bar().encode(
  alt.X("YEAR:O"),
  alt.Y("NUM_OBSERVATIONS:Q")
)
```



4. a.

```
#make a new dataframe that groups the data by the year counting the unique
 ↪  provider numbers
pos_unique_count =
 ↪  pos_df.groupby("YEAR")["PRVDR_NUM"].nunique().reset_index()
```

```
#make a chart
alt.Chart(pos_unique_count).mark_bar().encode(
  alt.X("YEAR:O"),
  alt.Y("PRVDR_NUM:Q")
)
```



b. These plots look the exact same and that is because the numbers are the same that it is taking in. This tells us that the data is only contains one observation for each hospital in each year. There are no duplicates for the hospitals.

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

1.

```
#making 2016 list of actives
active_2016 = pos_df[pos_df["PGM_TRMNTN_CD"] == 0]
active_2016 = active_2016[active_2016["YEAR"] == 2016]
len(active_2016)

#sort by 2016 actives
term_p2016 = pos_df[(pos_df["PRVDR_NUM"].isin(active_2016["PRVDR_NUM"])) &
 ↪  (pos_df["YEAR"]> 2016)]

#finding rows where termination code isnt 0
term_p2016 = term_p2016[term_p2016["PGM_TRMNTN_CD"] != 0]
```

```
#group by facility name then only retain the lowest year
term_p2016 = term_p2016.groupby("PRVDR_NUM")[["FAC_NAME", "ZIP_CD",
↪  "YEAR"]].min().reset_index()
term_p2016.head()

print(len(term_p2016), "hospitals fit this definition")
#consult with hallie on repeats by different zips
```

```
174 hospitals fit this definition
```

2.

```
#sort values by facility name
term_p2016 = term_p2016.sort_values(by = "FAC_NAME")

#printing top 10 results
print(term_p2016.head(10))
```

	PRVDR_NUM	FAC_NAME	ZIP_CD	YEAR
127	440162	(CLOSED) HEALTHSOUTH CHATTANOOGA REHAB HOSPITAL	37404	2018
4	030001	ABRAZO MARYVALE CAMPUS	85031	2017
10	050196	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	93230	2017
97	360151	AFFINITY MEDICAL CENTER	44646	2018
80	330189	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	12208	2017
140	450488	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	75662	2017
62	250159	ALLIANCE LAIRD HOSPITAL	39365	2019
101	370032	ALLIANCEHEALTH DEACONESS	73112	2019
21	060036	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	81050	2017
156	520048	ASCENSION NE WISCONSIN MERCY CAMPUS	54904	2018

3. a.

```
#first create a df that will count active hospitals by zip by year
active_zips = pos_df.copy()
active_zips["terminated"] = active_zips["PGM_TRMNTN_CD"].apply(lambda x: "no"
↪  if x == 0 else "yes")
active_zips["ZIP_CD"]

zips_term_counts = active_zips.groupby(["ZIP_CD", "terminated",
↪  "YEAR"]).size().reset_index(name = "count")

#test that there are actually unique results in here
```

```
zips_term_counts["count"].unique()
zips_term_counts = zips_term_counts[["ZIP_CD", "YEAR", "count"]]
```

```
#now I'll test whether the zips listed in when_term_results actually change
 ↪  in the year after shown
term_p2016["year+"] = term_p2016["YEAR"] + 1

#eliminate any duplicates
zips_term_counts = zips_term_counts.groupby(["ZIP_CD", "YEAR"],
 ↪  as_index=False).agg({"count": "sum"})

#merge on year terminated
results1 = zips_term_counts.merge(term_p2016[["ZIP_CD", "YEAR"]],
  how = "right")
results1 = results1[["ZIP_CD", "YEAR", "count"]]

#merge on next year
results2 = zips_term_counts.merge(term_p2016[["ZIP_CD", "year+"]],
  left_on = ["ZIP_CD", "YEAR"],
  right_on = ["ZIP_CD", "year+"],
  how = "right")
results2 = results2[["ZIP_CD", "YEAR", "count"]]

#merge together to find where counts overlap
results = results1.merge(results2,
  on = ["ZIP_CD"])
results.head(10)
len(results)
```

```
176
```

```
#pulling out the non-terminated bunch
nterminated = results[results["count_x"] == results["count_y"]]
nterminated.head()

print("Based on the above,",len(nterminated),"of", len(term_p2016), "zip
 ↪  codes are not true terminations")
```

```
Based on the above, 92 of 174 zip codes are not true terminations
```

b.

```
#do reverse of n terminated and keep only those zips

terminated = results[results["count_x"] != results["count_y"]]
terminated_df = pos_df[pos_df["ZIP_CD"].isin(terminated["ZIP_CD"])]
terminated_df = terminated_df.groupby("FAC_NAME").size().reset_index()
terminated_df = terminated_df["FAC_NAME"]

print("After first pass,", len(terminated_df), "hospitals remain")
```

```
After first pass, 185 hospitals remain
```

```
#finding the non-mergers
term_non_mergers = pos_df[pos_df["ZIP_CD"].isin(terminated["ZIP_CD"])]
term_non_mergers= term_non_mergers[term_non_mergers["PGM_TRMNTN_CD"] != 1]
term_non_mergers = term_non_mergers.groupby("FAC_NAME").size().reset_index()
term_non_mergers = term_non_mergers["FAC_NAME"]

print("After removing voluntary mergers,", len(term_non_mergers), "hospitals
 ↪  remain")
```

```
After removing voluntary mergers, 154 hospitals remain
```

c.

```
#first 10 remaining hospitals
print(term_non_mergers.sort_values().head(10))
```

```
0                              ADVENTIST HEALTH TULARE
1                              ALLIANCE LAIRD HOSPITAL
2                             ALLIANCEHEALTH DEACONESS
3                       AMERICAN TRANSITIONAL HOSPITAL
4                        ANNE BATES LEACH EYE HOSPITAL
5                        BARIX CLINICS OF PENNSYLVANIA
6                      BAYLOR EMERGENCY MEDICAL CENTER
7    BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...
8                            BELMONT COMMUNITY HOSPITAL
9                                BIG SKY MEDICAL CENTER
Name: FAC_NAME, dtype: object
```

Download Census zip code shapefile (10 pt)

1. a. The five file types are: .dbf: contains information about the file, the data records, and the end of the file .prj: contains information about the coordinate system so the data can be projected onto a map .shp: contains information about geographic information and attributes of the data .shx: contains information about the shape files that are used for fonts and line types .xml: contains data that is in plain text format

   b. The size of the files are as follows. These are listed in order from biggest to smallest: .shp: 837,544,580 bytes (852.9 MB on disk) .dbf: 6,425,474 bytes (6.4 MB on disk) .shx: 265,060 bytes (266 KB on disk) .xml: 15,639 bytes (16 KB on disk) .prj: 165 bytes (4 KB on disk)

2.

```
#load in necessary package
import shapely
import geopandas as gpd
```

```
#load in the data
shp_path =
 ↪  "/Users/hallielovin/Documents/GitHub/problem-set-4-ben_hallie/gz_2010_us_860_00_500k"

ben_shp_path = f"/Users/benschiffman/Desktop/Python
 ↪  2/problem-set-4-ben_hallie/gz_2010_us_860_00_500k"


shp = gpd.read_file(shp_path)
```

```
#Pull out only the Texas zip codes
##indicate what the TX zip codes are
texas_code = ("75", "76", "77", "78", "79")

#make a new dataframe that only pulls out those rows that are from texas
texas = shp[shp["ZCTA5"].astype(str).str.startswith(texas_code)]
```

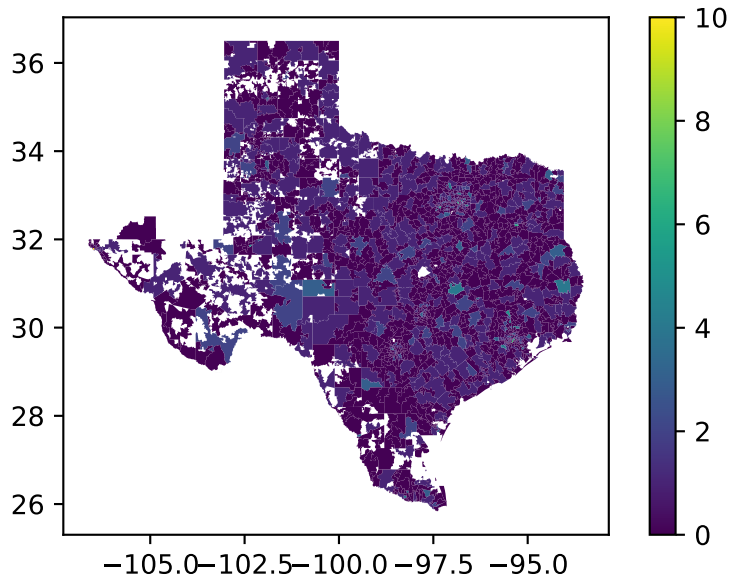
```
#Calculate the number of hospitals per zipcode based on the pos2016_df from
 ↪  part 1

##restrict this to tx
```

```
texas_2016 =
 ↪  pos2016_df[pos2016_df["ZIP_CD"].astype(str).str.startswith(texas_code)]

#group by zipcode and count the number of hospitals
zip_count = texas_2016.groupby("ZIP_CD").agg(
  NUM_HOSPITALS = ("ZIP_CD", "size")
).reset_index()
```

```
#Clean up the zip codes a bit -- needed chatgpt help on this since it was not
 ↪  working at first
zip_count["ZIP_CD"] = zip_count["ZIP_CD"].astype(str).str.replace('.0', '',
 ↪  regex=False).str.strip()

shp["ZCTA5"] = shp["ZCTA5"].astype(str).str.strip()

#merge the datasets by zip code
texas = texas.merge(zip_count, left_on="ZCTA5", right_on="ZIP_CD",
 ↪  how="left")

#Make anything that is NA = to 0
texas["NUM_HOSPITALS"] = texas["NUM_HOSPITALS"].fillna(0)
```

```
import matplotlib.pyplot as plt
```

```
texas.plot(column = "NUM_HOSPITALS", legend = True)
```



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

1.

```
zips_all_centroids = shp.copy()
zips_all_centroids = zips_all_centroids.to_crs(epsg=3857) #this is from
 ↪  chatGPT after I recieved a message about this issue
zips_all_centroids["centroid"] = zips_all_centroids.centroid
zips_all_centroids.head()
zips_all_centroids.area
```

```
0         1.078828e+08
1         1.862316e+08
2         2.481179e+07
3         1.336166e+08
4         2.142780e+08
             ...
33115     4.011196e+08
33116     4.747644e+08
33117     5.231207e+07
33118     1.336986e+08
33119     7.008696e+07
Length: 33120, dtype: float64
```

Source for the below answers: https://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2024/TGRSHP2

Column1: GEO_ID refers to a unique alpha numeric identifier used by the census bureau for geographic areas for which is tabulates data. Column2: ZCTA5 is the 5 digit zip code of the tabulation area. Column 3: Name in this case just repeats zip code, but in other cases may refer to the actual name of an area. Column 4: LSAD = Legal statistical area description Col5: CENSUSAREA Geometry: Eitehr municipal or otherwise designated areas for statistical purposes https://www.census.gov/programs-surveys/geography/about/glossary.html Column6: GEOMETRY contains the WKT pologyon data for each census area. Column7: Centroid contains the points that represent the centroids of each census area.

2.

```
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids["ZCTA5"].astype(str).str.startswith(texas_code)]
len(zips_texas_centroids)
texas_border_codes = ("87", "880", "881", "882", "883", "884", "73", "74",
 ↪  "70", "71", "72")
all_codes = texas_code + texas_border_codes
zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids["ZCTA5"].astype(str).str.startswith(all_codes)]

print("zips_texas_centroids zips:", len(zips_texas_centroids),
 ↪  "\nzzips_texas_borderstates_centroids:",
 ↪  len(zips_texas_borderstates_centroids))
```

```
zips_texas_centroids zips: 1935
zzips_texas_borderstates_centroids: 4057
```

3.

```
#Create zip code list from 2016 of >= 1active hospitals
zips_2016 = active_2016.copy()
zips_2016 = zips_2016.groupby("ZIP_CD").size().reset_index()

#manipulate so merge will work
zips_2016["ZIP_CD"] = zips_2016["ZIP_CD"].str.strip()

#merge
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
  zips_2016,
  left_on = "ZCTA5",
  right_on = "ZIP_CD",
```

```
  how = "inner"
)
len(zips_withhospital_centroids)
zips_withhospital_centroids.tail()
```

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry
443	8600000US77375	77375	77375	ZCTA5	34.049	MULTIPOLYGON (((-10636192.291 3
444	8600000US74361	74361	74361	ZCTA5	171.051	POLYGON ((-10589690.355 4350710.8
445	8600000US75035	75035	75035	ZCTA5	23.643	POLYGON ((-10772124.633 3924040.0
446	8600000US78028	78028	78028	ZCTA5	250.675	POLYGON ((-11043092.415 3526837.7
447	8600000US78412	78412	78412	ZCTA5	8.798	POLYGON ((-10832297.827 3212445.6

I used an inner merge to preserve only rows where ZIP_CD and ZCTA5 matched. These are the variables I merged on, both represent zip code. In the output there are some redundant columns as a result, can be dealt with later. 4. a.

```
import time
start = time.time()
ten_zips = zips_texas_centroids.head(10)
ten_zip_test = gpd.sjoin_nearest(
  ten_zips,
  zips_withhospital_centroids,
  how = "inner",
  distance_col = "distance"
)
end = time.time()

time_test = end - start
print(time_test)
```

```
0.4406769275665283
```

Given that this took .35 seconds and there are 1935 zips in zips_texas_centoids, I expect the entire operation to take about 68 seconds. b.

```
start2 = time.time()
zips_texas_centroids_distances = gpd.sjoin_nearest(
  zips_texas_centroids,
  zips_withhospital_centroids,
  how = "inner",
```

```
  distance_col = "distance"
)
end2 = time.time()

time_test2 = end2 - start2
print(time_test2)
```

```
109.92364692687988
```

I was wrong! It took 94.71 seconds. This means that the time growth is not linear. c.

```
so interestingly enough, the units are in degrees. That said, I already
converted to meters because an earlier warning message, and some help from
the chatGPT, reccomended I did so. The following answer will convert from
meters to miles.
```

```
zips_texas_centroids_distances["distance"] =
 ↪  zips_texas_centroids_distances["distance"] * 0.000621371
```

5. a.

```
zips_texas_centroids_distances
mean_dist_to_hospital_tx =
 ↪  zips_texas_centroids_distances.groupby("ZIP_CD").agg(
  mean_distance = ("distance", "mean"),
  geometry = ("geometry", "first")
  ).reset_index()
```

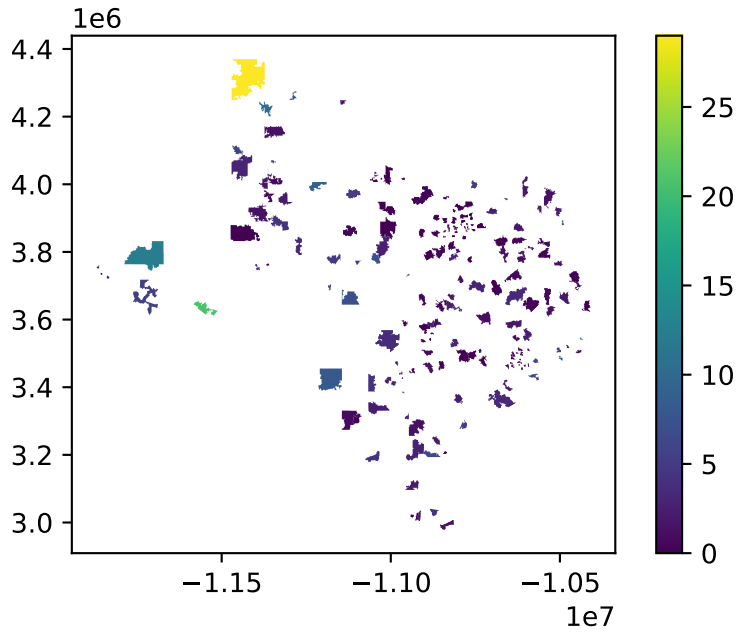
The distance is in miles b.

```
mean_dist = mean_dist_to_hospital_tx["mean_distance"].mean()
mean_dist
```

```
np.float64(1.7274344237449712)
```

The mean as indicated above is 1.73 miles. This distance doesn't really make sense because the question iself is somewhat confusing, what does it mean to be the closest hospital to a zip code? What the merge we did only finds the closest zip code with a hospital… so this may not be all that accurate. c.

```
mean_dist_to_hospital_tx = gpd.GeoDataFrame(mean_dist_to_hospital_tx,
 ↪  geometry='geometry')
mean_dist_to_hospital_tx.plot(column = "mean_distance", legend = True)
```



Effects of closures on access in Texas (15 pts)

1.

```
#create a df of the facilities that had a closure in Texas based on the
 ↪  term_p2016 chart made in previous steps
texas_closure =
 ↪  term_p2016[term_p2016["ZIP_CD"].astype(str).str.startswith(texas_code)]
```

```
#group my zip code and count the number of closures by each zipcode
tx_zip_closures = texas_closure.groupby("ZIP_CD").size().reset_index(name =
 ↪  "NUM_CLOSURES")

tx_zip_closures["ZIP_CD"].astype(str)
print(tx_zip_closures)
```

	ZIP_CD	NUM_CLOSURES
0	75042	1
1	75051	1
2	75087	1
3	75140	1
4	75231	1
5	75235	1
6	75390	1
7	75601	1
8	75662	1
9	75835	1
10	75862	1
11	76502	1
12	76520	1
13	76531	1
14	76645	1
15	77035	1
16	77054	1
17	77065	1
18	77429	1
19	77479	1
20	77598	1
21	78017	1
22	78061	1
23	78336	1
24	785	1
25	78613	1
26	78734	1
27	78834	1
28	79520	1
29	79529	1
30	79553	1
31	79735	1
32	79761	1
33	79902	1

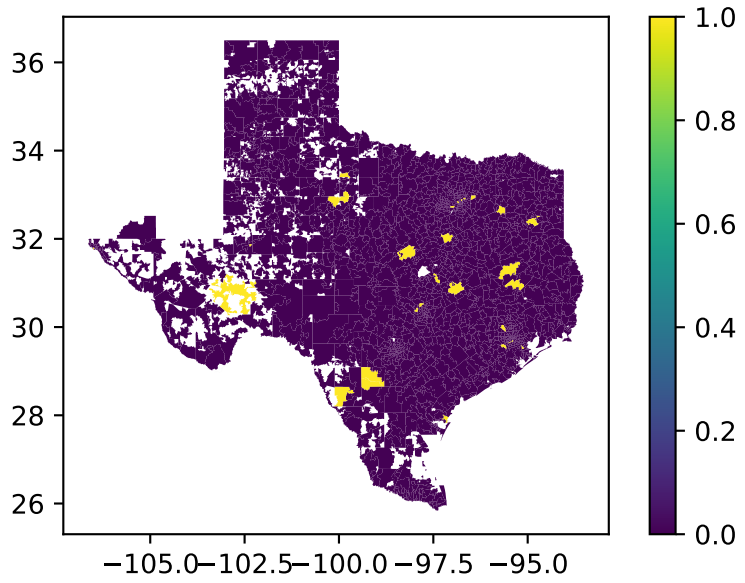
2.

```
#convert zip codes so string and strip them
tx_zip_closures["ZIP_CD"] = tx_zip_closures["ZIP_CD"].astype(str).str.strip()

texas["ZCTA5"] = texas["ZCTA5"].astype(str).str.strip()
```

```
#combine the zip codes with closures to the original texas shape file and
 ↪  only keep those that are in the shape file
shp_texas_closure = texas.merge(tx_zip_closures, left_on="ZCTA5",
 ↪  right_on="ZIP_CD", how="left")

shp_texas_closure["NUM_CLOSURES"] =
 ↪  shp_texas_closure["NUM_CLOSURES"].fillna(0)

shp_texas_closure.head()
```

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry
0	8600000US78624	78624	78624	ZCTA5	708.041	POLYGON ((-98.96423 30.49848, -98.96
1	8600000US78626	78626	78626	ZCTA5	93.046	POLYGON ((-97.60944 30.57185, -97.61
2	8600000US78628	78628	78628	ZCTA5	73.382	POLYGON ((-97.69285 30.57122, -97.69
3	8600000US78631	78631	78631	ZCTA5	325.074	POLYGON ((-99.13053 30.36555, -99.13
4	8600000US78632	78632	78632	ZCTA5	96.278	POLYGON ((-97.40946 29.75929, -97.40

```
#make a chloropleth to indicate where the zip codes are that had at least one
 ↪  closure
shp_texas_closure.plot(column = "NUM_CLOSURES", legend = True)
```



```
len(tx_zip_closures)
```

```
34
```

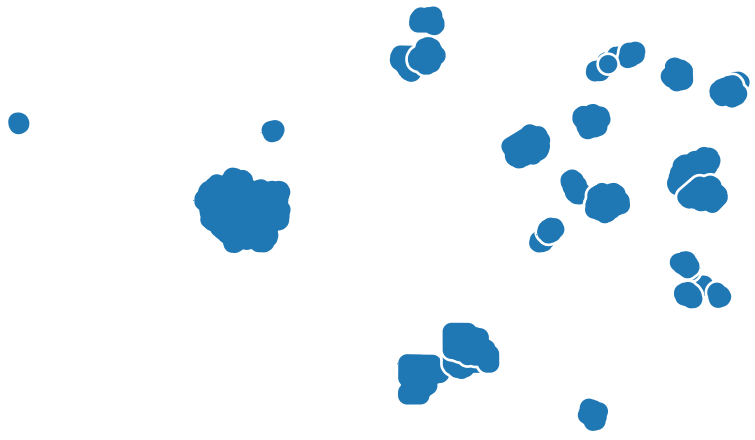
There were 34 affected zip codes that had at least one closure and are directly affected.

3.

```
#create the buffer distance to convert 10 miles into meters
buffer_distance = 10 * 1609.34
```

```
#had to use chatgpt to help me fix this because without this line of code it
 ↪  was generating one large circle
shp_texas_closure = shp_texas_closure.to_crs(epsg=32614)

#filter shp_texas_closure to only show those that had closures
yes_closed = shp_texas_closure[shp_texas_closure["NUM_CLOSURES"] > 0]

#make a chloropleth that includes the buffer
yes_closed.buffer(buffer_distance).plot(edgecolor="white")
plt.axis("off")
```

```
(np.float64(-286717.66442250466),
 np.float64(980964.7704973313),
 np.float64(3029566.491980887),
 np.float64(3757268.1700539053))
```



```
#make a new df for zip codes indirectly affected
shp_buffered_zip = yes_closed.copy()

#add a new column for geometry that is for the buffered distance
shp_buffered_zip["geometry"] =
 ↪  shp_buffered_zip.geometry.buffer(buffer_distance)
```

```
len(shp_buffered_zip)
```

```
33
```

```
#make sure they are the same crs
if texas.crs != shp_buffered_zip.crs:
    shp_buffered_zip = shp_buffered_zip.to_crs(texas.crs)

#do the spatial join
indirect_zips = gpd.sjoin(texas, shp_buffered_zip, how="inner",predicate =
 ↪  "intersects")
```

```
#find how many zip codes are indirectly affected
indirect_zips["ZCTA5_left"].nunique()
```

```
609
```

There are 609 zip codes that are indirectly affected by hospital closures.

4.

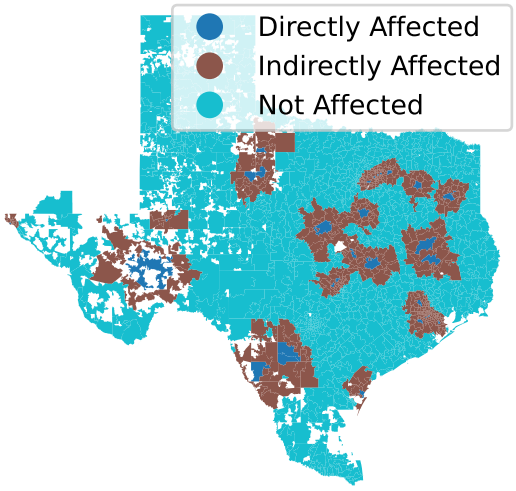
```
#make a new column in the texas shapefile which indicates if a zip code is
 ↪  affected by a closure
texas["AFFECT"] = "Not Affected"
```

```
# pull out the zip codes that were indirectly affected
indirectly_affected = indirect_zips["ZCTA5_left"].unique()

texas.loc[texas["ZCTA5"].isin(indirectly_affected), "AFFECT"] = "Indirectly
 ↪  Affected"
```

```
#pull out the zip codes that are directly affected
directly_affected = yes_closed["ZCTA5"].unique()

#find these in the texas file and label them as Directly Affected
texas.loc[texas["ZCTA5"].isin(directly_affected), "AFFECT"] = "Directly
 ↪  Affected"
```

```
#make a chloropleth and color code based on the AFFECT category
texas.plot(column = "AFFECT", legend = True)
plt.axis("off")
```

```
(np.float64(-107.30252635),
 np.float64(-92.85115865),
 np.float64(25.303987),
 np.float64(37.033881))
```



Reflecting on the exercise (10 pts)

1. Using the first pass method could cause some issues, one of which being, when we use this method we are not seeing the exact year that a hosptial closes, we are just assuming. That said, a hospital could have actually closed in 2016, but because of the way it is coded in the data we are forced to just assume 2017. Additionally, if hospitals do not appear in the data it is assumed that they are no longer active. This is a big assumption. A hospital may not appear in data for numerous other reasons, such as a data collection

misshap or it may have change categories of the type of hospital it is. It is unfair to assume that the hospital is no longer a funcitoning facility.

We can do a better job of confirming hopsital closures by having hospitals submit forms when they are officially closed to data collection agencies so they are not forced to assume.

2. I think the way we approached identifying zip codes by closures doesn't do a good job of relfecting zip code access to hospitals. For one thing, just counting the number of hospitals available in a zip code or the number of closures for that matter, doesn't necesarily reflect access to hospitals. In addition, mapping the distance to the nearest zip code with a hospital doesnt do much to demonstrate how easy to access these facilitites are.

I think one way we may want to instead measure access could be to find the location of hopitals and measure the distance to the centroid of eachzip code. That way we could map the average distance from the centroid to an actual hospital. This would better reflect access to a facility.