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

1.

```
gdf_whole['centroid'] = gdf_whole.centroid
zips_all_centroids = gdf_whole[['ZCTA5', 'centroid']]
print(zips_all_centroids.shape)
print('We have 33120 rows and 2 columns. ZCTA5 is the postal zip codes of
 ↪  districts in the U.S.')
print('ZCTA5 is the approximate postal zip codes of districts in the U.S..')
print('''Centroid represents the position of center point of the area covered
 ↪  by a specific zip code in the Geographic Coordinate System:
 ↪  GCS_North_American_1983, with Datum being D_North_American_1983 and
 ↪  GRS_1980 spheroid and Greenwich prime meridian.''')
```

```
(33120, 2)
We have 33120 rows and 2 columns. ZCTA5 is the postal zip codes of districts
in the U.S.
ZCTA5 is the approximate postal zip codes of districts in the U.S..
Centroid represents the position of center point of the area covered by a
specific zip code in the Geographic Coordinate System:
GCS_North_American_1983, with Datum being D_North_American_1983 and GRS_1980
spheroid and Greenwich prime meridian.
```

2.

```
zips_texas_centroids = (zips_all_centroids[zips_all_centroids['ZCTA5']
                        .str.startswith(("75", "76", "77", "78", "79"))])
zip_texas = zips_texas_centroids['ZCTA5'].nunique()
print(f'{zip_texas} unique zip codes in zips_texas_centroids.')
# States boadering Texas:
# New Mexico: 870-884
# Oklahoma: 73-74
# Arkansas: 716-729
# Louisiana: 700-715
# Texas: 75 - 79
# Combined: 70-79, 870-884
prefixes_70_to_79 = [f"{i}" for i in range(70, 80)]
prefixes_870_to_884 = [f"{i}" for i in range(870, 885)]
```

```
prefixes_tuple = tuple(prefixes_70_to_79 + prefixes_870_to_884)
zips_texas_borderstates_centroids =
 ↪  (zips_all_centroids[zips_all_centroids['ZCTA5']
                        .str.startswith(prefixes_tuple)])
zip_boarders = zips_texas_borderstates_centroids['ZCTA5'].nunique()
print(f'{zip_boarders} unique zip codes in
 ↪  zips_texas_borderstates_centroids.')
```

```
1935 unique zip codes in zips_texas_centroids.
4057 unique zip codes in zips_texas_borderstates_centroids.
```

3.

```
zips_texas_borderstates_centroids["ZCTA5"] =
 ↪  pd.to_numeric(zips_texas_borderstates_centroids["ZCTA5"])
zips_withhospital_centroids = pd.merge(zips_texas_borderstates_centroids,
 ↪  active_2016,
                                       left_on='ZCTA5', right_on='ZIP_CD',
 ↪  how='inner', indicator=True
                                       ).drop_duplicates('ZCTA5')
zips_withhospital_centroids = zips_withhospital_centroids[['ZCTA5',
 ↪  'centroid']]
print('''I did inner merge, on ZCTA5 in zips_texas_borderstates_centroidsand
 ↪  ZIP_CD in active_2016, zip codes in the two dataframes.''')
 ↪
```

```
I did inner merge, on ZCTA5 in zips_texas_borderstates_centroidsand ZIP_CD in
active_2016, zip codes in the two dataframes.
```

4. (a)

```
# Sample
start_time = time.time()
sample_zip = zips_texas_centroids.head(10)
sample_zip = sample_zip.set_geometry('centroid')
zips_withhospital_centroids =
 ↪  zips_withhospital_centroids.set_geometry('centroid')
join_to_hospital_sample = gpd.sjoin_nearest(
```

```
    sample_zip,
    zips_withhospital_centroids,
    how='inner',
    distance_col="distance"
)
end_time = time.time()
print(f'It took {end_time - start_time} seconds.')
print(f'Full merge is expected to take {(end_time - start_time) * (1935/10)}
 ↪  seconds.')
```

```
It took 0.013988256454467773 seconds.
Full merge is expected to take 2.706727623939514 seconds.
```

4. (b)

```
# Full data
start_time = time.time()
zips_texas_centroids = zips_texas_centroids.set_geometry('centroid')
join_to_hospital_full = gpd.sjoin_nearest(
    zips_texas_centroids,
    zips_withhospital_centroids,
    how='inner',
    distance_col="distance"
)
end_time = time.time()
print(f'It took {end_time - start_time} seconds.')
print('The actual time taken is far more shorter than estimated.')
```

```
It took 0.02020883560180664 seconds.
The actual time taken is far more shorter than estimated.
```

4. (c)

From the .prj file, we can see that the unit is specified in degrees. The UNIT parameter is defined as "Degree",0.017453292519943295, so the unit of this coordinate system is a degree, of latitude and longitude. One degree equals approximately 69 miles.

```
# Convert degrees into miles
join_to_hospital_full['distance_miles'] = join_to_hospital_full['distance'] *
 ↪  69
```

5.

```
avg_distance = join_to_hospital_full['distance'].mean()
print(avg_distance)
```

```
0.21101748566398393
```

5. (a)

It is in degrees.

5. (b)

```
avg_distance = join_to_hospital_full['distance'].mean()
avg_distance_mile = avg_distance * 69
print(f'The average distance in miles is {avg_distance_mile:.2f} miles.')
```

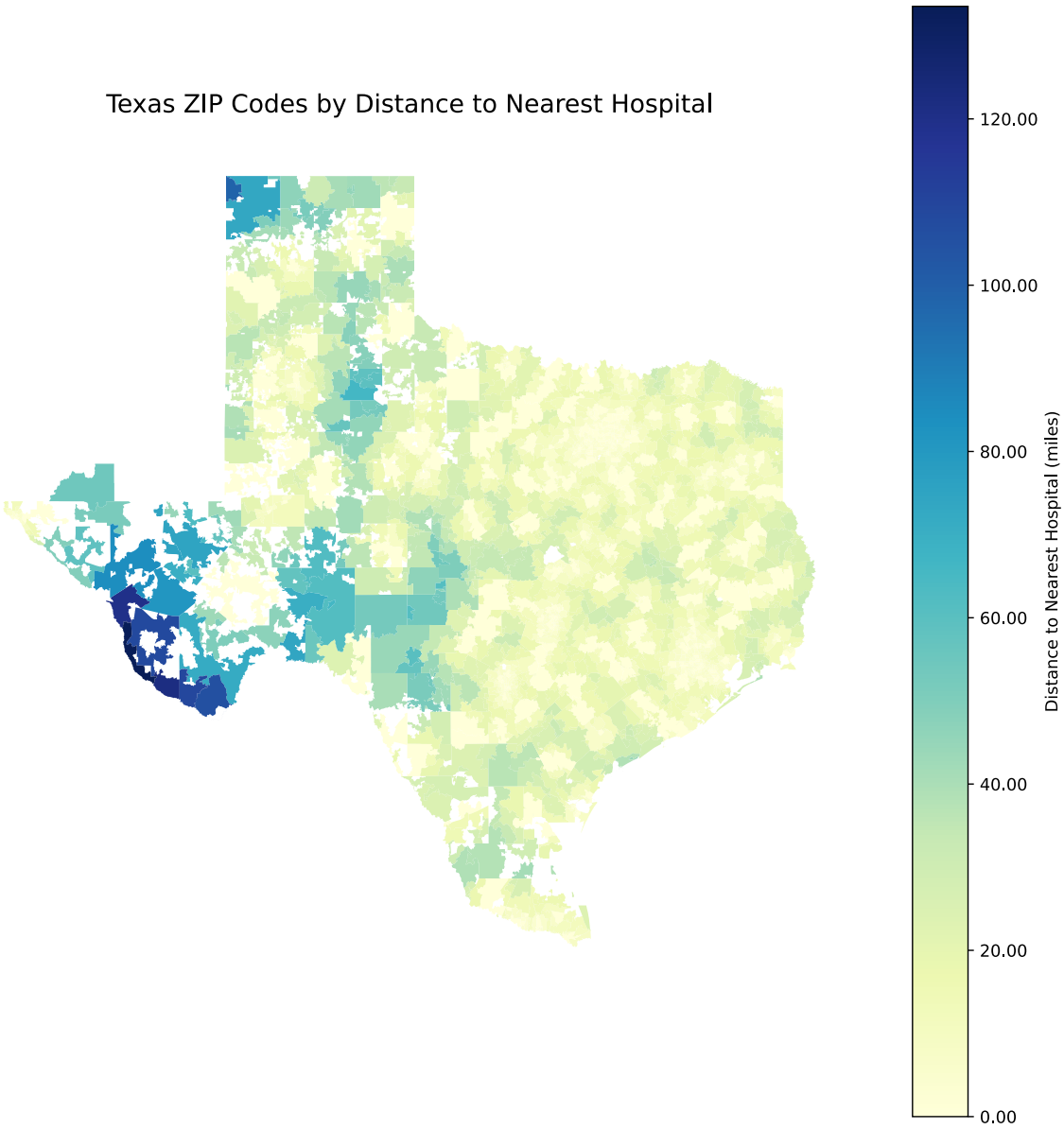
```
The average distance in miles is 14.56 miles.
```

This result makes sense because according to info page provided, the number of rural hospital closures has increased significantly in recent years. Since Texas is a state where a substantial share of population living in relatively remote rural areas, it is possible that the average distance to hospital is relatively long.

5. (c)

```
# Merge polygons back to join data
join_to_hospital_full['ZCTA5_left'] =
 ↪  join_to_hospital_full['ZCTA5_left'].astype(str)
gdf['ZCTA5'] = gdf['ZCTA5'].astype(str)
merged_gdf = pd.merge(join_to_hospital_full, gdf[['ZCTA5', 'geometry']],
 ↪  left_on='ZCTA5_left', right_on='ZCTA5')
merged_gdf['distance'] = merged_gdf['distance'] * 69
merged_gdf = merged_gdf.set_geometry('geometry')
```

```
# Form the plot
fig, ax = plt.subplots(1, 1, figsize=(12, 12))
plot = merged_gdf.plot(column='distance', cmap='YlGnBu', legend=True,
                       legend_kwds={'label': 'Distance to Nearest Hospital
 ↪  (miles)', 'orientation': "vertical",
                       'format': '%.2f'}, ax=ax)
ax.set_title('Texas ZIP Codes by Distance to Nearest Hospital', fontsize=15,
 ↪  pad=15)
ax.set_axis_off()

plt.show()
```



Effects of closures on access in Texas (15 pts)

1.

Here we will use the further cleaned dataset of corrected closure which takes ZIP codes with no hospitals reappearing into consideration. (Another version is by simple non-decrease approach, resulting in a corrected closure of 77 hospitals.)

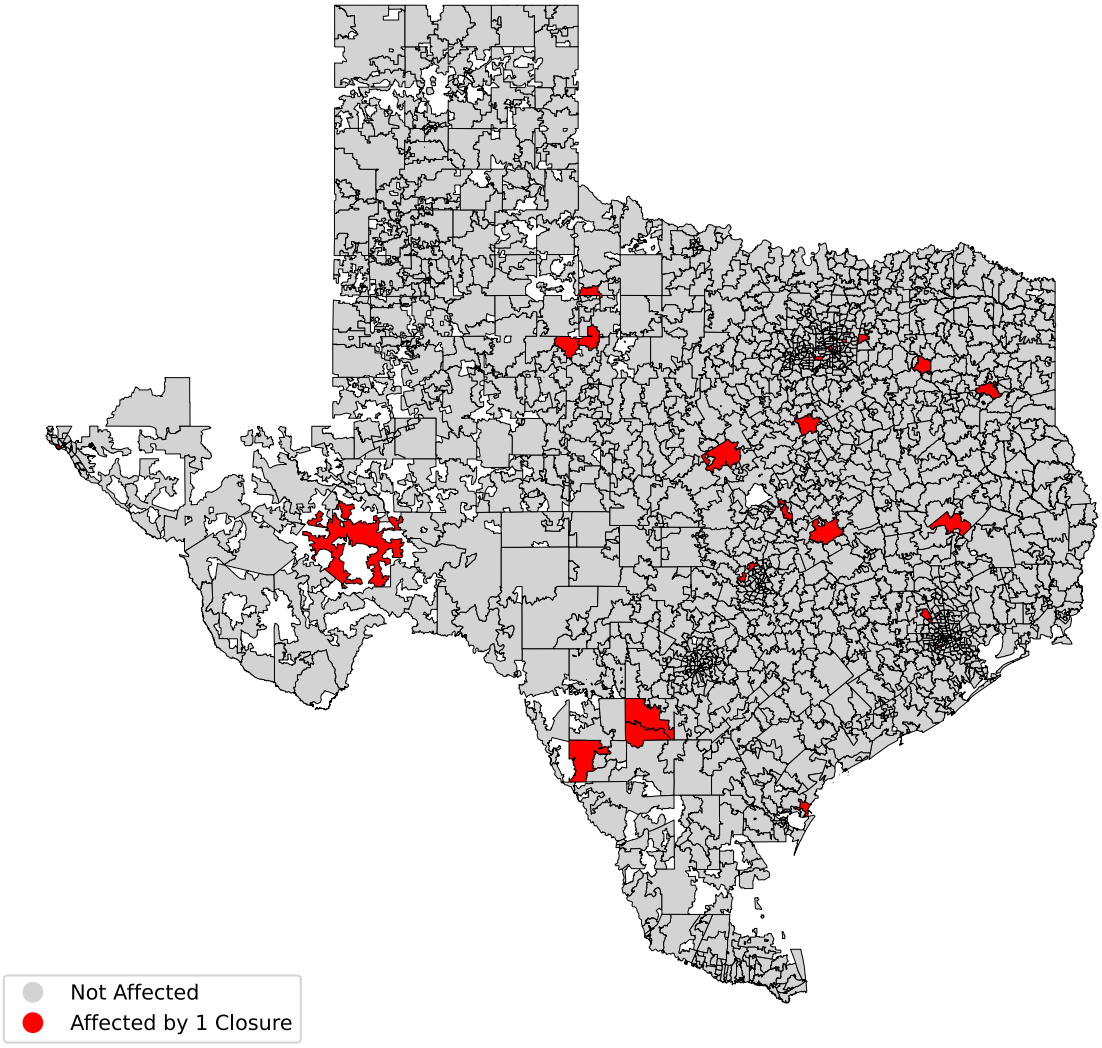
```
zip_vs_counts = corrected_closed.groupby(
    "ZIP_CD").size().reset_index(name="Closure_Count")
# Restrict to Texas
zip_vs_counts["ZIP_CD"] = zip_vs_counts["ZIP_CD"].astype(int).astype(str)
zip_vs_counts = zip_vs_counts[
    zip_vs_counts["ZIP_CD"].str.startswith(("75", "76", "77", "78", "79"))
].reset_index(drop=True)
zip_vs_counts
```

	ZIP_CD	Closure_Count
0	75042	1
1	75051	1
2	75087	1
3	75140	1
4	75235	1
5	75390	1
6	75662	1
7	75862	1
8	76502	1
9	76520	1
10	76531	1
11	76645	1
12	77035	1
13	77065	1
14	77429	1
15	78017	1
16	78061	1
17	78336	1
18	78613	1
19	78734	1
20	78834	1
21	79520	1
22	79529	1
23	79553	1
24	79735	1
25	79902	1

2.

For the choropleth:

```
# Create a GeoDataFrame
gdf["ZCTA5"] = gdf["ZCTA5"].astype(str)
gdf_zip_counts = gdf.merge(zip_vs_counts, left_on="ZCTA5", right_on="ZIP_CD",
 ↪  how="left").fillna(0)
gdf_zip_counts["Closure_Count"] =
 ↪  gdf_zip_counts["Closure_Count"].astype(float)

# Make the Plot
cmap = ListedColormap(["lightgrey", "yellow", "red"])

fig, ax = plt.subplots(figsize=(10, 10))

ax = gdf_zip_counts.plot(
    column="Closure_Count",
    cmap=cmap,
    legend=True,
    categorical=True,
    legend_kwds={"loc": "lower left"},
    linewidth=0.1,
    edgecolor="black",
    ax=ax
)
ax.set_title("Texas ZIP Codes with Hospital Closures")

legend_labels = {"0.0": "Not Affected", "1.0": "Affected by 1 Closure",
 ↪  "2.0": "Affected by 2 Closures"}
for txt in ax.get_legend().get_texts():
    txt.set_text(legend_labels.get(txt.get_text(), txt.get_text()))

plt.axis("off")
plt.show()
```



For the number of directly affected zip codes:

```
number_affected_zip = len(zip_vs_counts)
print(f"There are {number_affected_zip} ZIP codes directly affected by a
 ↪  closure in 2016-2019.")
```

```
There are 26 ZIP codes directly affected by a closure in 2016-2019.
```

3.

```
# Create a GeoDataFrame of directly affected ZIP codes
gdf_directly_affected = gdf_zip_counts[gdf_zip_counts["Closure_Count"] > 0]

# Apply 10-mile buffer
buffer_degree = 10/69  # convert 10 miles into degrees
buffered_zones = gdf_directly_affected.copy()
buffered_zones["geometry"] = buffered_zones.geometry.buffer(buffer_degree)

# Spatial join to indentify all ZIP codes within the 10-mile buffer
gdf_indirectly_affected = gpd.sjoin(
    gdf, buffered_zones, how="inner", predicate="intersects")
gdf_indirectly_affected =
 ↪  gdf_indirectly_affected[~gdf_indirectly_affected["ZCTA5_left"].isin(
    gdf_directly_affected["ZCTA5"])]

# Find how many
gdf_indirectly_affected["ZCTA5_left"].nunique()
```

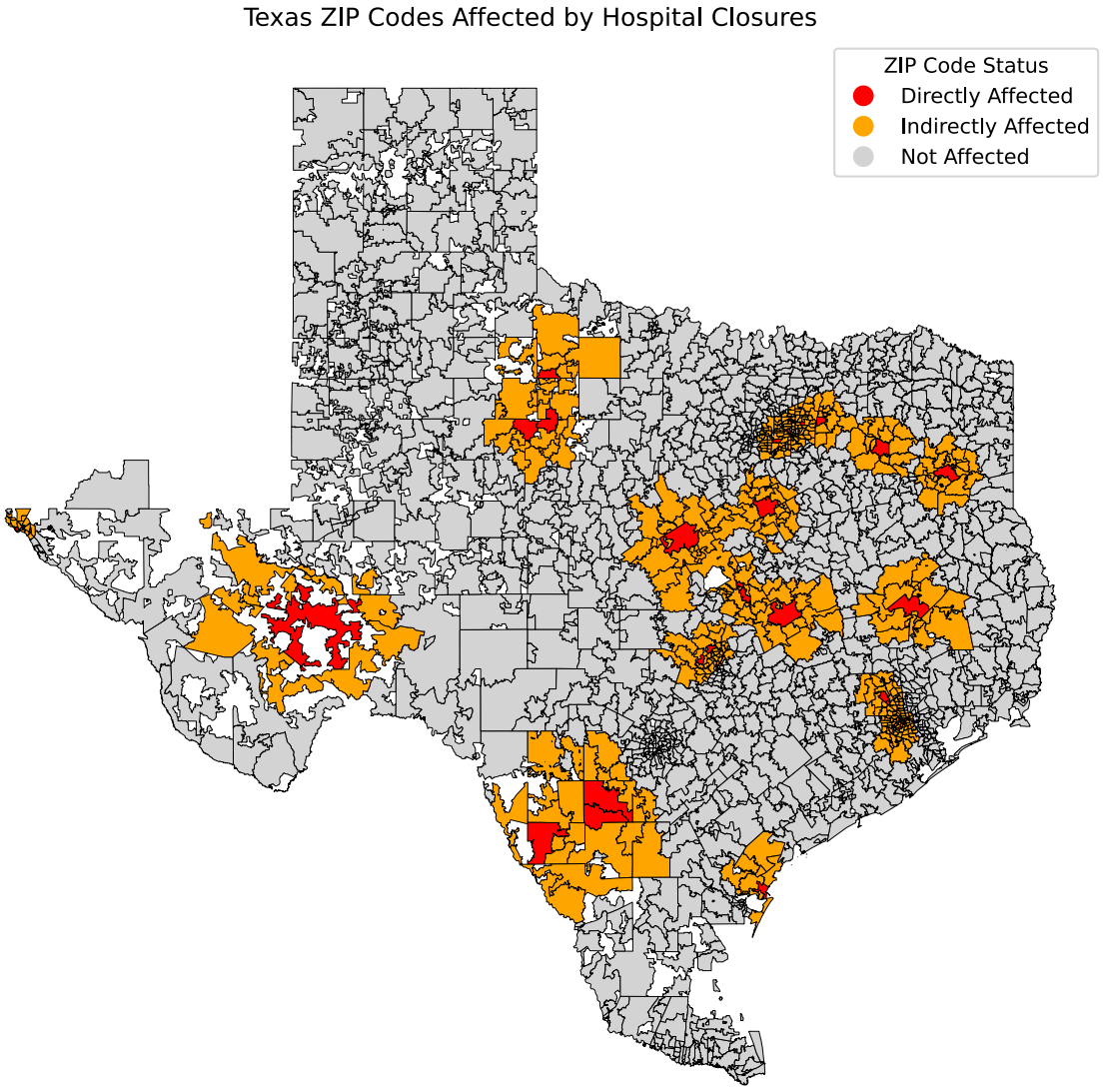
```
491
```

Therefore, there are 491 ZIP codes that are indirectly affected.

4.

```
# Divide ZIP codes into 3 status
gdf["Status"] = "Not Affected"

gdf.loc[gdf["ZCTA5"].isin(gdf_directly_affected["ZCTA5"]), "Status"] =
 ↪  "Directly Affected"

gdf.loc[gdf["ZCTA5"].isin(gdf_indirectly_affected["ZCTA5_left"]), "Status"] =
 ↪  "Indirectly Affected"

# Step 3: Plot the choropleth
cmap = ListedColormap(["red", "orange", "lightgrey"])

fig, ax = plt.subplots(figsize=(10, 10))
gdf.plot(
    column="Status",
    cmap=cmap,
```

```
    legend=True,
    legend_kwds={
        "title": "ZIP Code Status",
        "loc": "upper right",
        "labels": [ "Directly Affected", "Indirectly Affected","Not
        ↪  Affected"]
    },
    edgecolor="black",
    linewidth=0.1,
    ax=ax
)
ax.set_title("Texas ZIP Codes Affected by Hospital Closures")

plt.axis("off")
plt.show()
```



Reflecting on the exercise (10 pts)

1.

The "first-pass" method can lead to several issues:
- Overly Simplistic Comparison: In a true closure scenario, the number of active hospitals in the closure year already decreases. Therefore, even if the active hospital count in the year after is larger or equal to the closure year, it is natural and doesn't necessarily indicate a

merger/acquisition.
- Inability to Confirm 2019 Closures: This method doesn't allow us to confirm for 2019 closures because we lack data for the following year. As a result, this will misidentify mergers and overestimate closures.
- False Flags Due to New Openings: In cases where a hospital truly closes but a new facility opens in the same zip code the following year, this method would incorrectly suggest that the closure was a merger/acquisition. This results in a misclassification of genuine closures, especially in areas with frequent new hospital openings.
To do a better job at confirming closures, firstly, for each facility that is no longer active, a closer inspection of the termination code would help clarify whether the hospital ceased operations due to closure or some other reason. Also, we can utilize additional databases or state health department records to validate closure status, especially for hospitals with ambiguous termination codes. Besides, using longitudinal data to track hospitals over multiple years would also be helpful.

2.

The definitions of directly and indirectly affected areas make sense, as local closures have an immediate impact that diminishes over distance, thus indirectly affecting nearby areas.

However, this measure fails to consider the actual distance to hospitals and the number of hospitals within each zip code. For instance, the impact of one hospital closing in a zip code with eight hospitals is far less significant than in a zip code with only one hospital. Moreover, the 10-mile buffer radius does not accurately reflect the varied distances across Texas zip codes to the nearest hospitals; some zip codes inherently have hospitals, whereas others are more than 100 miles away from the nearest facility. For zip codes that already have hospitals, the closure of a nearby hospital has a much smaller impact than those zip codes where the nearest hospital is extremely far away. Therefore, this measure fails to reflect the real changes in zip-code-level hospital access accurately.

To improve this measure, I suggest layering the map of zip codes with hospital closures with maps showing distances to the nearest hospital and the number of active hospitals. Specifically, I recommend displaying three maps side by side: the first map only marking zip codes with hospital closures, the second map layering these zip codes as points on a map showing distances to the nearest hospital, and the third map layering these zip codes as points on a map showing the number of active hospitals. This way, readers can understand not only which zip codes have experienced hospital closures but also gauge the actual impact of these closures.