mapping

May 22, 2024

1 Mapping

In response to the 2008 U.S. Farm Bill, the U.S. Department of Agriculture's Economic Research Service compiled a June 2009 report to Congress:

> According to data from the latest census (2000), about 23.5 million people, or 8.4 percent of the U.S. population, live in low-income neighborhoods that are more than a mile from a supermarket. Low-income neighborhoods are areas where more than 40 percent of the population has income less than or equal to 200 percent of the Federal poverty threshold ($44,000 per year for a family of four in 2008).

In this assessment, we'll simulate their analysis by creating geospatial maps to help us understand food access in Washington. There are three geographic region types that we'll use in this assessment:

- **Census tract** is a geographic region used in the U.S. census. It is the smallest of the three region types.
- **County** is a geographic region used for administrative purposes that can include one or more census tracts.
- **State** is a geographic region such as the State of Washington. It is the largest of the three region types.

A census tract is defined as **low access** if enough people in the tract do not have nearby access to grocery stores offering affordable and nutritious food. In urban areas, "low access" is defined as 0.5 miles; in rural areas, "low access" is defined as 10 miles.

`tl_2010_53_tract00.shp` contains the 2010 US census dataset in geospatial shapefile format only for Washington state (53). The only columns you need to use are `CTIDFP00`, the census tract identifier, and `geometry`, the geometric shape of the tract.

`food_access.csv` contains the food access dataset in tabular CSV format. Each row in the dataset corresponds to a census tract for every state in the country (not just Washington). This dataset has many columns but you only need to understand the following:

- `CensusTract` is the census tract identifier.
- `State` is the state name for the census tract.
- `County` is the county name for the census tract.
- `Urban` is a flag (0 or 1) that indicates if this census tract is an urban environment.
- `Rural` is a flag that indicates if this census tract is a rural environment.
- `LATracts_half` is a flag that indicates if this census tract is "low access" in a half mile radius.
- `LATracts10` is a flag that indicates if this census tract is "low access" in a 10 mile radius.
- `LowIncomeTracts` is a flag that indicates if this census tract is "low income".

- `POP2010` is the number of people in this census tract according to the 2010 census.
- `lapophalf` is the number of people in this census tract considered having "low access" in a half mile radius.
- `lapop10` is the number of people in this census tract considered having "low access" in a 10 mile radius.
- `lalowihalf` is similar to `lapophalf` but only counts people considered low access and low income.
- `lalowi10` is similar to `lapop10` but only counts people considered low access and low income.

[11]:
```
!pip install -q folium mapclassify

import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd
import folium

# For testing purposes
import json
import numpy as np
from geopandas.plotting import _plot_polygon_collection
from matplotlib.collections import PatchCollection

with open("expected_idx.json") as f:
    rural_idx, rural_la_idx, urban_ha_idx, lalowi_idx = json.load(f)

class MockAxes:
    def add_collection(self, *args, **kwargs):
        pass

    def autoscale_view(self, *args, **kwargs):
        pass

def assert_patches_allclose(actual_patches, ax=MockAxes(), num_colors=None,␣
 ↪**kwargs):
    if isinstance(kwargs.get("geoms"), str):
        kwargs["geoms"] = gpd.read_file(kwargs["geoms"]).geometry
    expected_patches = _plot_polygon_collection(ax=ax, **kwargs)
    for expected, actual in zip(expected_patches.get_paths(), actual_patches.
 ↪get_paths()):
        try:
            np.testing.assert_allclose(expected.vertices, actual.vertices)
        except AssertionError as e:
            e.args = "wrong selection",
            raise e
    if "color" in kwargs:
        try:
```

```
            np.testing.assert_allclose(expected_patches.get_fc(),␣
↪actual_patches.get_fc())
        except AssertionError as e:
            e.args = "wrong color",
            raise e
    elif isinstance(num_colors, int):
        num_unique = len(np.unique(actual_patches.get_fc(), axis=0))
        assert num_unique == num_colors, f"expected {num_colors} colors, got␣
↪{num_unique} colors"
```

1.1 Collaboration and Conduct

Students are expected to follow Washington state law on the Student Conduct Code for the University of Washington. In this course, students must:

- Indicate on your submission any assistance received, including materials distributed in this course.
- Not receive, generate, or otherwise acquire any substantial portion or walkthrough to an assessment.
- Not aid, assist, attempt, or tolerate prohibited academic conduct in others.

Update the following code cell to include your name and list your sources. If you used any kind of computer technology to help prepare your assessment submission, include the queries and/or prompts. Submitted work that is not consistent with sources may be subject to the student conduct process.

[ ]:
```
your_name = "Cassie Hoang"
sources = [
    "geospatial-data.ipynb",
    "dissolve-intersect-and-join.ipynb"
]

assert your_name != "", "your_name cannot be empty"
assert ... not in sources, "sources should not include the placeholder ellipsis"
assert len(sources) >= 2, "must include at least 2 sources, inclusive of␣
↪lectures and sections"
```

1.2 Task: Load in data

Write a function `load_data` that takes path for census dataset and the path for the food access dataset and returns the `GeoDataFrame` resulting from merging the two datasets on the census tract identifiers `CTIDFP00` / `CensusTract`. Assume the census tract identifier columns exist: use only these two column names. Not all census tracts have food access data.

[2]:
```
def load_data(shp_path, csv_path):
    """This function takes the path for the census dataset and the food access␣
↪dataset as parameters.
```

```
    It returns a GeoDataFram of the two datasets merged together, including all␣
 ↪census tracts,
    even if there is no food access data."""
    shp = gpd.read_file(shp_path)
    csv = pd.read_csv(csv_path)
    #how='left' includes all census data tracts, even if they do not have food␣
 ↪access data
    merge = shp.merge(csv, left_on='CTIDFP00', right_on='CensusTract',␣
 ↪how='left')
    merge_gdf = gpd.GeoDataFrame(merge, geometry='geometry')
    return merge_gdf

state_data = load_data("tl_2010_53_tract00.shp", "food_access.csv")
display(state_data)
assert type(state_data) == gpd.GeoDataFrame
assert list(state_data.columns) == [
    "STATEFP00", "COUNTYFP00", "TRACTCE00", "CTIDFP00", "NAME00", "NAMELSAD00",␣
 ↪"MTFCC00",
    "FUNCSTAT00", "ALAND00", "AWATER00", "INTPTLAT00", "INTPTLON00",␣
 ↪"geometry", "CensusTract",
    "State", "County", "Urban", "Rural", "LATracts_half", "LATracts10",␣
 ↪"GroupQuartersFlag",
    "OHU2010", "NUMGQTRS", "PCTGQTRS", "LowIncomeTracts", "POP2010",␣
 ↪"lapophalf", "lalowihalf",
    "lapop10", "lalowi10",
]
assert len(state_data) == 1318
```

```
     STATEFP00 COUNTYFP00 TRACTCE00     CTIDFP00  NAME00           NAMELSAD00  \
0           53        077    001400  53077001400      14      Census Tract 14
1           53        077    001600  53077001600      16      Census Tract 16
2           53        077    000700  53077000700       7       Census Tract 7
3           53        077    002400  53077002400      24      Census Tract 24
4           53        077    002200  53077002200      22      Census Tract 22
…          …         …         …          …     …                 …
1313        53        063    010202  53063010202  102.02  Census Tract 102.02
1314        53        063    010301  53063010301  103.01  Census Tract 103.01
1315        53        063    010504  53063010504  105.04  Census Tract 105.04
1316        53        063    010303  53063010303  103.03  Census Tract 103.03
1317        53        063    001600  53063001600      16      Census Tract 16

     MTFCC00 FUNCSTAT00    ALAND00  AWATER00  … GroupQuartersFlag OHU2010  \
0      G5020          S    5539748         0  …               0.0  1203.0
1      G5020          S   97657363   1509774  …               NaN     NaN
2      G5020          S    2930010         0  …               0.0  2602.0
3      G5020          S  232557960     69748  …               NaN     NaN
4      G5020          S  207645882         0  …               0.0  2501.0
```

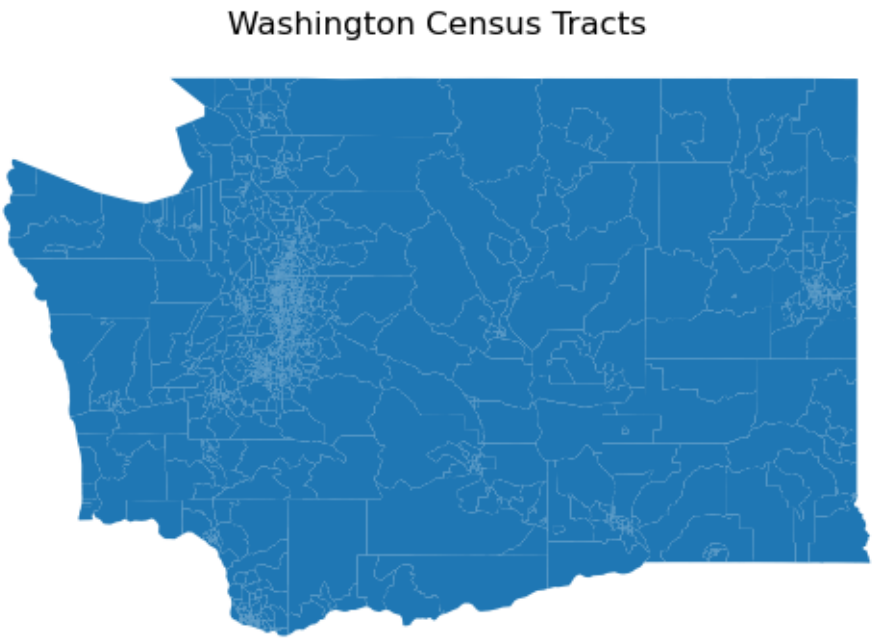
```
…       …         …          …         …  …                 …      …
1313   G5020          S  184070644         0  …               0.0  2346.0
1314   G5020          S   21667422         0  …               0.0  1612.0
1315   G5020          S    9371197         0  …               0.0  1325.0
1316   G5020          S  107033392         0  …               0.0  1085.0
1317   G5020          S    2104249         0  …               0.0  1430.0

     NUMGQTRS  PCTGQTRS LowIncomeTracts POP2010    lapophalf   lalowihalf  \
0        62.0  0.018002             1.0  3444.0  2883.838461  1594.727661
1         NaN       NaN             NaN     NaN          NaN          NaN
2       141.0  0.019938             1.0  7072.0  1881.362199   948.972610
3         NaN       NaN             NaN     NaN          NaN          NaN
4        23.0  0.002970             1.0  7745.0  6137.595205  2770.949604
…        …         …               …       …          …          …
1313      8.0  0.001146             0.0  6983.0  6842.568092   860.247108
1314     13.0  0.003047             1.0  4266.0  3597.752448  1770.888294
1315      0.0  0.000000             0.0  3546.0  3237.127237   517.875884
1316     12.0  0.003971             0.0  3022.0  3022.000003   507.539103
1317      0.0  0.000000             1.0  3738.0   145.702168    99.504575

          lapop10   lalowi10
0        0.000000   0.000000
1             NaN        NaN
2        0.000000   0.000000
3             NaN        NaN
4        0.000000   0.000000
…           …         …
1313   112.663119  15.763564
1314     0.000000   0.000000
1315     0.000000   0.000000
1316     0.000000   0.000000
1317     0.000000   0.000000

[1318 rows x 30 columns]
```

1.3 Task: Plot census tracts

Write a function `plot_census_map` that takes the merged data and returns the `Axes` that contains shapes of all the census tracts in Washington. Title the plot "Washington Census Tracts" and turn off axis labels.

[3]:
```
def plot_census_map(state_data):
    """This function takes the merged data as a parameter and returns a plot␣
 ↪with the shapes of all
    census tracts in Washington."""
    ax = state_data.plot()
    ax.set(title="Washington Census Tracts")
    ax.set_axis_off()
```

```
    return ax

ax = plot_census_map(state_data)
layers = ax.findobj(PatchCollection)
assert_patches_allclose(layers[0], geoms=state_data.geometry)
assert len(layers) == 1, "unexpected extra plot layers"
assert ax.get_title() == "Washington Census Tracts", "title does not match␣
 ↪expected"
assert not ax.axison, "borders and labels must be hidden"
```



When given no arguments, the `dissolve` method considers the entire `GeoDataFrame` as a single group. This will be useful for plotting backgrounds later.

[4]:
```
entire_state = state_data[["geometry"]].dissolve()
display(entire_state)
ax = entire_state.plot(color="#EEE")
ax.set_axis_off()
```

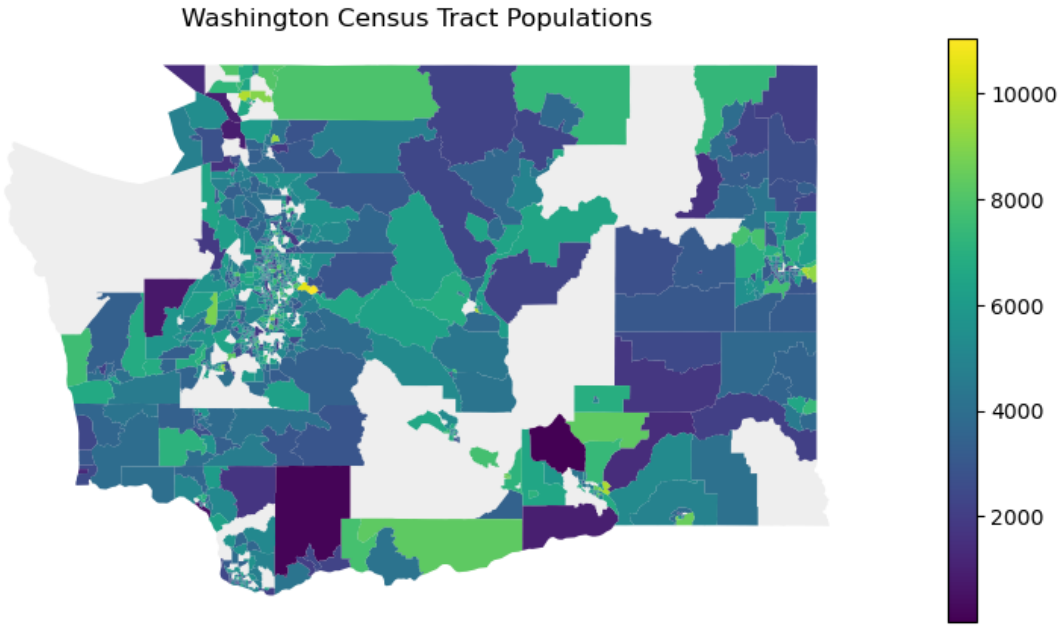
```
                                                 geometry
0  POLYGON ((-122.88260 46.05175, -122.88261 46.0…
```

1.4 Task: Plot census tract populations

Write a function `plot_census_population_map` that takes the merged data and return the `Axes` that plots all the census tracts in Washington where each census tract is colored according to the `POP2010` column. There will be some missing census tracts. Underneath, plot the entire state of Washington in the background color `#EEE`. Title the plot "Washington Census Tract Populations", turn off axis labels, include a legend, and increase the figure size so that the map is the same height as the legend.

[5]:
```
def plot_census_population_map(state_data):
    """This function takes the merged data as a parameter in returns a plot␣
 ↪that displays all
    census tracts in Washington by color based on their population size. Some␣
 ↪census tracts will
    not display a representative color due to missing population values."""
    fig, ax = plt.subplots(figsize=(13,5))
    entire_state = state_data[["geometry"]].dissolve()
    entire_state.plot(ax=ax,color="#EEE")
    pop_plot = state_data.plot(ax=ax, column="POP2010", legend=True)
    ax.set_title("Washington Census Tract Populations")
    ax.set_axis_off()
    return ax

ax = plot_census_population_map(state_data)
layers = ax.findobj(PatchCollection)
assert_patches_allclose(layers[0], geoms=entire_state.geometry, color="#EEE")
```

```
assert_patches_allclose(layers[1], geoms=state_data.dropna().geometry,␣
↪num_colors=183)
assert len(layers) == 2, "unexpected extra plot layers"
assert ax.get_title() == "Washington Census Tract Populations", "title does not␣
↪match expected"
assert not ax.axison, "borders and labels must be hidden"
cbar = ax.get_figure().get_axes()[-1]
assert cbar.get_label() == "<colorbar>", "missing legend"
assert ax.bbox.height == cbar.bbox.height, "map can be enlarged"
```

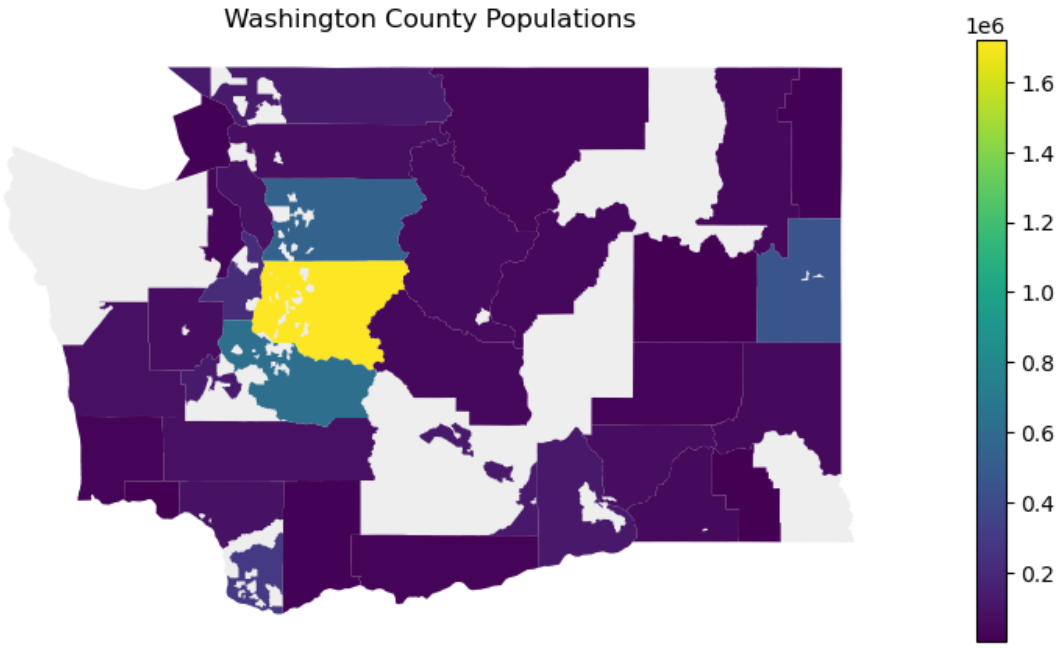


1.5 Task: Plot county populations

Write a function `plot_county_populations_map` that takes the merged data and returns the `Axes` that plots all the counties in Washington where each county is colored according to the `POP2010` column. This will require combining all the census tract data and geometries for each county, though there will be missing data for some counties. Underneath, plot the entire state of Washington in the background color `#EEE`. Title the plot "Washington County Populations", turn off axis labels, include a legend, and increase the figure size so that the map is the same height as the legend.

[6]:
```
def plot_county_population_map(state_data):
    """This function takes the merged data as a parameter and returns a plot␣
↪that displays all
    counties in Washington by color based on their population size. Some␣
↪counties will not display
    a representative color due to missing population values."""
```

```
    fig, ax = plt.subplots(figsize=(13,5))
    entire_state = state_data[["geometry"]].dissolve()
    entire_state.plot(ax=ax,color="#EEE")
    counties = state_data.dissolve(by="County", aggfunc="sum")[['geometry',␣
 ↪'POP2010']]
    counties.plot(ax=ax, column="POP2010", legend=True)
    ax.set_title("Washington County Populations")
    ax.set_axis_off()
    return ax

ax = plot_county_population_map(state_data)
layers = ax.findobj(PatchCollection)
assert_patches_allclose(layers[0], geoms=entire_state.geometry, color="#EEE")
assert_patches_allclose(layers[1], geoms="counties.geojson", num_colors=20)
assert len(layers) == 2, "unexpected extra plot layers"
assert ax.get_title() == "Washington County Populations", "title does not match␣
 ↪expected"
assert not ax.axison, "borders and labels must be hidden"
cbar = ax.get_figure().get_axes()[-1]
assert cbar.get_label() == "<colorbar>", "missing legend"
assert ax.bbox.height == cbar.bbox.height, "map can be enlarged"
```



1.6 Task: Plot food access by county

Write a function `plot_food_access_by_county_map` that takes the merged data and returns a 4-tuple of `Axes` that represent the subplots in a 2-by-2 figure consisting of 4 choropleth maps:

- Top left plot titled "Low Access: Half Mile" showing the proportion of people per county who have low access to food within a half mile `lapophalf` out of the total population `POP2010`.
- Top right plot titled "Low Access + Low Income: Half Mile" showing the proportion of people per county considered low income who also have low access to food within a half mile `lalowihalf` out of the total population `POP2010`.
- Bottom left plot titled "Low Access: 10 Miles" showing the proportion of people per county who have low access to food within 10 miles `lapop10` out of the total population `POP2010`.
- Bottom right plot titled "Low Access + Low Income: 10 Miles" showing the proportion of people per county considered low income who also have low access to food within 10 miles `lalowi10` out of the total population `POP2010`.

When calling `plot`, specify the keyword arguments `vmin=0` and `vmax=1` so that the subplots all share the same scale. Underneath, plot the entire state of Washington in the background color `#EEE`. We recommend preparing subplots with `figsize=(15, 10)`. Turn off axis labels on each subplot.

[7]:
```
def plot_food_access_by_county_map(state_data):
    """This function takes the merged data as a parameter and returns 4 plots␣
 ↪that displays the
    proportion of individuals per county in Washington that have low food␣
 ↪access and/or low income.
    Some counties may not display a representative color due to missing␣
 ↪population values."""
    fig, [[ax1, ax2], [ax3, ax4]] = plt.subplots(2, 2, figsize=(15, 10))
    axs = [ax1, ax2, ax3, ax4]

    accesses = ["lapophalf", "lalowihalf", "lapop10", "lalowi10"]
    titles = ["Low Access: Half Mile", "Low Access + Low Income: Half Mile",␣
 ↪"Low Access: 10 Miles",
              "Low Access + Low Income: 10 Miles"]

    for access, ax, title in zip(accesses, axs, titles):
        entire_state = state_data[["geometry"]].dissolve()
        entire_state.plot(ax=ax, color="#EEE")
        low_access = state_data.dissolve(by="County",␣
 ↪aggfunc="sum")[['geometry', 'POP2010',
                                                    'lapophalf',␣
 ↪'lalowihalf', 'lapop10', 'lalowi10']]
        low_access[access] = low_access[access]/ low_access["POP2010"]
        low_access.plot(ax=ax, column=access, vmin=0, vmax=1)
        ax.set(title=title)
        ax.set_axis_off()

    return axs

axs = plot_food_access_by_county_map(state_data)
```

```
expected_titles = ["Low Access: Half Mile", "Low Access + Low Income: Half␣
 ↪Mile",
                   "Low Access: 10 Miles", "Low Access + Low Income: 10 Miles"]
for ax, expected_num_colors, expected_title in zip(axs, [31, 23, 19, 16],␣
 ↪expected_titles):
    layers = ax.findobj(PatchCollection)
    assert_patches_allclose(layers[0], geoms=entire_state.geometry,␣
 ↪color="#EEE")
    assert_patches_allclose(layers[1], geoms="counties.geojson",␣
 ↪num_colors=expected_num_colors)
    assert len(layers) == 2, "unexpected extra plot layers"
    assert ax.get_title() == expected_title, f"title {ax.get_title()} does not␣
 ↪match expected"
    assert not ax.axison, "borders and labels must be hidden"
```

1.7 Writeup: Food access by county

Setting aside the lack of a legend in `plot_food_access_by_county_map`, is it an effective visualization? Why or why not?

Aside from the lack of a legend that makes it difficult to visualize and specify population values, plot_food_access_by_county_map is not a very effective visualization because it is difficult to clarify the characteristics of a single county. Having to analyze all four chlorepleth maps to define

the varying food access of characteristics of each county is tedious and ineffective. It would be easier for all access variables to be visible for every county on one map, for example averaging the access of each county instead.

1.8 Task: Plot low access census tracts

Write a function `plot_census_low_access_map` that takes the merged data and returns the `Axes` that plots all census tracts (not counties) considered low access using 5 `plot` layers for the following definitions for "low access" in urban and rural tracts. For this task, do not use the `LATracts_half` or `LATracts10` columns in the merged data; the procedure described below intentionally results in different values.

1. Plot the map of Washington in the background with color `#EEE`.
2. Plot all the `Urban` and `Rural` census tracts for which we have food access data in the color `#AAA`.
3. Plot all the `Urban` and `Rural` census tracts considered low access in the default (blue) color.

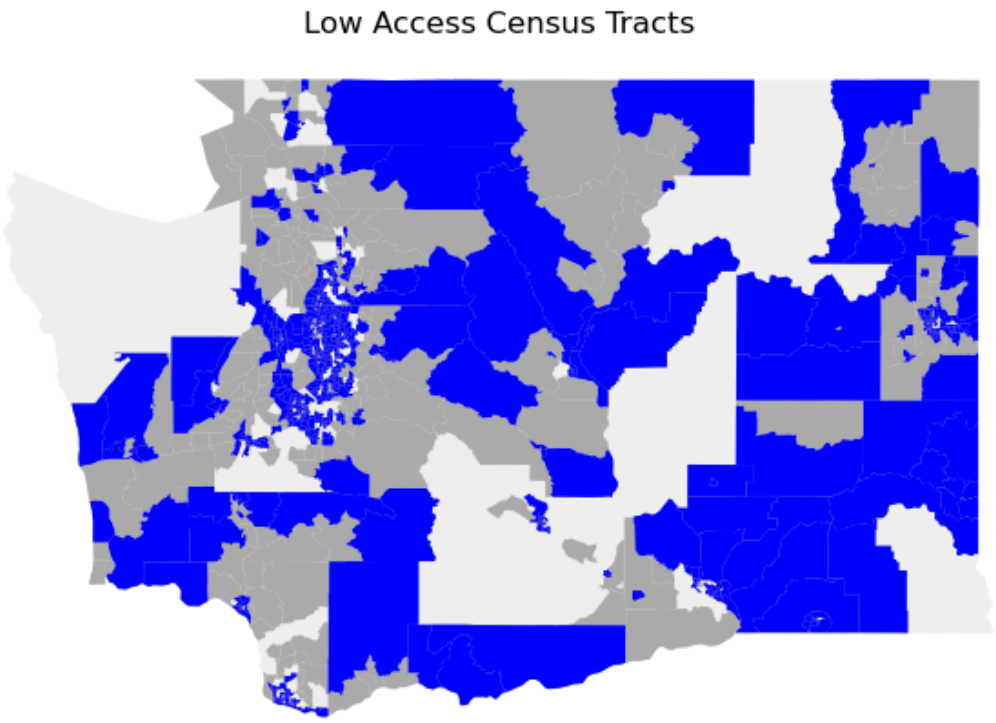
   Low access in an urban census tract is defined by a `lapophalf` value that is at least 500 people or at least 33% of the census tract population.

   Low access in a rural census tract is defined by a `lapop10` value that is at least 500 people or at least 33% of the census tract population.

Finally, title the plot "Low Access Census Tracts" and turn off axis labels.

[8]:
```
def plot_census_low_access_map(state_data):
    """This function takes the merged data as a parameter and returns a plot␣
 ↪that displays census
    tracts in Washington based on their urban and rural status, and if the␣
 ↪census tract is considered
    low in access to food. Dark gray colored census tracts are representative␣
 ↪of urban or rural areas,
    while blue colored tracts represent low access. Some census tracts will not␣
 ↪have a representative
    color due to missing population data."""
    fig, ax = plt.subplots(figsize=(13,5))
    entire_state = state_data[["geometry"]].dissolve()
    entire_state.plot(ax=ax, color="#EEE")

    urban = state_data[state_data['Urban'] == 1.0]
    urban.plot(ax=ax, legend = True, color='#AAA')
    rural = state_data[state_data['Rural'] == 1.0]
    rural.plot(ax=ax, legend = True, color='#AAA')

    low_urban = urban[(urban['lapophalf'] > 500) | ((urban['lapophalf'] /␣
 ↪urban['POP2010']) > 0.33)]
    low_urban.plot(ax=ax, color='blue')
```

```
    low_rural = rural[(rural['lapop10'] > 500) | ((rural['lapop10'] /␣
 ↪rural['POP2010']) > 0.33)]
    low_rural.plot(ax=ax, color='blue')

    ax.set_title("Low Access Census Tracts")
    ax.set_axis_off()

    return ax

ax = plot_census_low_access_map(state_data)
layers = ax.findobj(PatchCollection)
assert_patches_allclose(layers[0], geoms=entire_state.geometry, color="#EEE")
urban_idx = state_data["Urban"].notna() & ~state_data.index.isin(rural_idx)
urban_la_idx = urban_idx & ~state_data.index.isin(urban_ha_idx)
error = None
for i, j, k, l in [
    [1, 2, 3, 4], # urban, rural, urban low access, rural low access
    [2, 1, 3, 4], # rural, urban, urban low access, rural low access
    [1, 2, 4, 3], # urban, rural, rural low access, urban low access
    [2, 1, 4, 3], # rural, urban, rural low access, urban low access
    [1, 3, 2, 4], # urban, urban low access, rural, rural low access
    [3, 1, 4, 2], # rural, rural low access, urban, urban low access
]:
    try:
        assert_patches_allclose(layers[i], geoms=state_data.loc[urban_idx,␣
 ↪"geometry"], color="#AAA")
        assert_patches_allclose(layers[j], geoms=state_data.loc[rural_idx,␣
 ↪"geometry"], color="#AAA")
        assert_patches_allclose(layers[k], geoms=state_data.loc[urban_la_idx,␣
 ↪"geometry"])
        assert_patches_allclose(layers[l], geoms=state_data.loc[rural_la_idx,␣
 ↪"geometry"])
        break
    except AssertionError as e:
        if error is None: # Store only the first error encountered during␣
 ↪testing
            error = e
else: # Only raise an error if none of the possible ways to layer the plot␣
 ↪worked
    raise error
assert len(layers) == 5, "unexpected extra plot layers"
assert ax.get_title() == "Low Access Census Tracts", "title does not match␣
 ↪expected"
assert not ax.axison, "borders and labels must be hidden"
```



1.9 Writeup: Data-driven decision-making

What is one way that government or food access organizations could use `plot_food_access_by_county` or `plot_low_access_tracts` to shape their decisions or policies? Then, explain one limitation or concern with using the plots in the way you suggested.

plot_food_access_by_county can be used to analyze which census tracts are in low access to food, and use this data to provide more resources or implement plans that will improve this low access. The plot can also be used to analyze any trends in low access depending on locations, and whether certain variables or characteristics of locations show similar low access to food. However, a limitation could be that this plot only records and displays low access, and not varying levels of access. Low access does not necessarily correlate to high access, and examining all varying levels of food access could provide a better understanding of these census tracts to help these populations.

1.10 Task: Interactive map

Although the initial report to Congress was completed in June 2009, the Economic Research Service has since then maintained an interactive map for their **Food Access Research Atlas**. Open this interactive map, turn off the default layer "LI and LA and 1 and 10 miles", and turn on the layer "LI and LA at 1/2 and 10 miles". This layer displays:

> Low-income census tracts where a significant number or share of residents is more than 1/2 mile (urban) or 10 miles (rural) from the nearest supermarket.

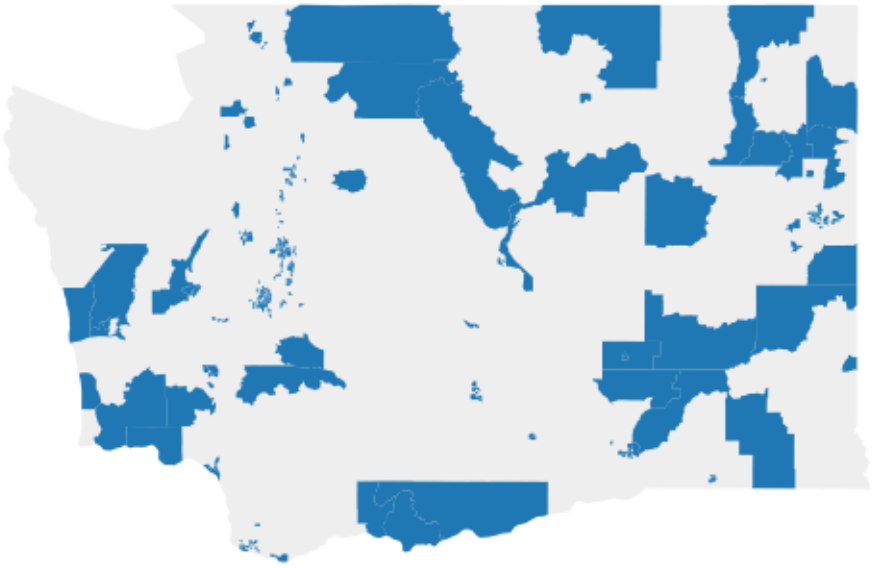
Write a function `interactive_map` that returns a Folium `Map` of low income and low access census tracts in Washington. Include only `LowIncomeTracts` that are either low access at half a mile `LATracts_half` or low access at 10 miles `LATracts10` depending on whether the census tract is `Urban` or `Rural`, respectively. This dataset does not match the Food Access Research Atlas, so some differences are to be expected. It is also OK if your interactive map does not appear in the PDF printout as PDF files cannot embed interactive maps.

[18]:
```
def interactive_map(state_data):
    """This function takes the merged data as a parameter and returns an␣
 ↪interactive map of low
    income and low access census tracts in Washington. This map only includes␣
 ↪low income access
    tracts that are either low access at half a mile or 10 miles, depending on␣
 ↪if the census tract
    is considered urban or rural."""
    low_income = state_data[state_data['LowIncomeTracts'] == 1.0]
    urban = (low_income['LATracts_half'] == 1.0) & (low_income['Urban'] == 1.0)
    rural = (low_income['LATracts10'] == 1.0) & (low_income['Rural'] == 1.0)
    lila_tracts = low_income[urban | rural]

    return lila_tracts.explore()

map = interactive_map(state_data)
display(map)
last_child = next(reversed(map._children.values()))
from folium.features import GeoJson
assert type(last_child) == GeoJson, "last child should be GeoJson; do not␣
 ↪specify column"
geojson = last_child.data
assert set(int(d["id"]) for d in geojson["features"]) == set(lalowi_idx),␣
 ↪"wrong selection"
```

```
<folium.folium.Map at 0x7c01e989a200>
```

The following cell plots a preview of your interactive map for the PDF printout.

[19]:
```
ax = entire_state.plot(color="#EEE")
gpd.GeoDataFrame.from_features(geojson, crs="EPSG:4326").plot(ax=ax)
ax.set_axis_off()
```



1.11 Writeup: Build a new supermarket

Using the interactive map above, locate the low-income low-access census tract closest to your favorite place in Washington. Then, identify a location (a specific street intersection, such as "University Way NE & NE 45th St") to add a new supermarket that would serve the people living in that census tract. Finally, explain the considerations that factored into your choice of location.

This dataset is outdated, so assume there are no supermarkets in any low-income low-access census tract even if supermarkets are present today.

I examined the areas around the University and identified a low-income low-access census tract. I chose census tract 52, and chose to add a new supermarket near NE 43rd and 8th St. I chose this specific location because it is at the center of many apartment and housing buildings. By adding a supermarket at a central location that is accessible to areas of housing, this supermarket can appropriately and adequately serve this community.