Assignment_3

February 11, 2024

[1]:
```
#load libraries
import pandas as pd
import numpy as np
import seaborn as sns
import geopandas as gpd
import matplotlib.pyplot as plt
%matplotlib inline
import pulp
from pulp import LpProblem, LpVariable, LpMinimize, lpSum, value
from plotnine import (ggplot, aes, geom_map, geom_text, geom_label,
                      ggtitle, element_blank, element_rect,
                      scale_fill_manual, theme_minimal, theme, coord_fixed,␣
 ↪xlim, ylim)
import adjustText as aT
import itertools
import folium
from folium import plugins

# View all columns and rows
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 90)

# Set up notebook to display multiple outputs in one cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

1 Load Datasets

1.1 County Adjacency Data

[2]:
```
adj_raw = pd.read_csv('county_adjacency2010.csv')
adj_raw.head()
adj_raw.info()
```

[2]:

	countyname	fipscounty	neighborname	fipsneighbor
0	Autauga County, AL	1001	Autauga County, AL	1001
1	Autauga County, AL	1001	Chilton County, AL	1021

```
 2  Autauga County, AL        1001   Dallas County, AL          1047
 3  Autauga County, AL        1001   Elmore County, AL          1051
 4  Autauga County, AL        1001  Lowndes County, AL          1085

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22200 entries, 0 to 22199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   countyname    22200 non-null  object
 1   fipscounty    22200 non-null  int64
 2   neighborname  22200 non-null  object
 3   fipsneighbor  22200 non-null  int64
dtypes: int64(2), object(2)
memory usage: 693.9+ KB
```

[3]:
```
#isolate Washington counties in adjacency matrix

adj = adj_raw.loc[adj_raw['countyname'].str.contains(', WA')]
adj['countyname'] = adj['countyname'].str[:-4]
adj['neighborname'] = adj['neighborname'].str[:-4]
adj.reset_index(drop = True, inplace = True)
adj.info()
adj.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 254 entries, 0 to 253
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   countyname    254 non-null    object
 1   fipscounty    254 non-null    int64
 2   neighborname  254 non-null    object
 3   fipsneighbor  254 non-null    int64
dtypes: int64(2), object(2)
memory usage: 8.1+ KB

/var/folders/z0/v3y1p30945d16_whz3lt8v_h0000gn/T/ipykernel_34501/4144378083.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/var/folders/z0/v3y1p30945d16_whz3lt8v_h0000gn/T/ipykernel_34501/4144378083.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

[3]:
	countyname	fipscounty	neighborname	fipsneighbor
0	Adams County	53001	Adams County	53001
1	Adams County	53001	Franklin County	53021
2	Adams County	53001	Grant County	53025
3	Adams County	53001	Lincoln County	53043
4	Adams County	53001	Whitman County	53075

1.2 Washington State Shapefile

[4]:
```
#read Washington shapefile
shapefile_WA = gpd.read_file('WA_County_Boundaries.shp')
shapefile_WA.head()
```

[4]:
	OBJECTID	JURISDICT_	JURISDIC_1	JURISDIC_2	JURISDIC_3	\
0	35374	25	4	Grant	Grant County	
1	39535	33	4	Garfield	Garfield County	
2	39897	8	4	Island	Island County	
3	40525	4699350	4	Kittitas	Kittitas County	
4	40569	35	4	Walla Walla	Walla Walla County	

	JURISDIC_4	JURISDIC_5	JURISDIC_6	EDIT_DATE	EDIT_STATU	EDIT_WHO	\
0	13	53025	None	2018-03-15	1	TSTE490	
1	12	53023	None	2022-06-23	1	TSTE490	
2	15	53029	None	2018-03-15	1	TSTE490	
3	19	53037	None	2023-07-27	0	TSTE490	
4	36	53071	None	2015-10-14	1	JDUG490	

	GLOBALID	\
0	{E82D6621-C75E-43A9-ACC2-71D374E5721C}	
1	{2D436843-A80E-4802-B5F5-0C2C62BD5D27}	
2	{8E32964C-BB29-460B-8D7C-1CCC33181DD4}	
3	{F24278BB-7AAD-458D-BB15-5F6321C597B4}	
4	{4D0C8CF7-D96F-4C1C-BE41-E7ED558E5544}	

	geometry
0	POLYGON ((-13245041.204 6100462.041, -13245049…
1	POLYGON ((-13077215.155 5893282.479, -13076922…
2	POLYGON ((-13645903.473 6175425.382, -13645833…
3	POLYGON ((-13482428.890 6040101.397, -13482417…
4	POLYGON ((-13170470.944 5878093.595, -13170220…

1.3 County Population Data

[5]:
```
#import csv of Washington county data (excluding King, Pierce, and Snohomish␣
 ↪Counties)
df = pd.read_csv('WA_Counties.csv')
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Python_County_Code  36 non-null     int64
 1   County              36 non-null     object
 2   County_Code         36 non-null     int64
 3   Population          36 non-null     int64
 4   White_Pop           36 non-null     int64
 5   Percent_White_Pop   36 non-null     float64
 6   Latitude            36 non-null     float64
 7   Longitude           36 non-null     float64
dtypes: float64(3), int64(4), object(1)
memory usage: 2.4+ KB
```

[5]:

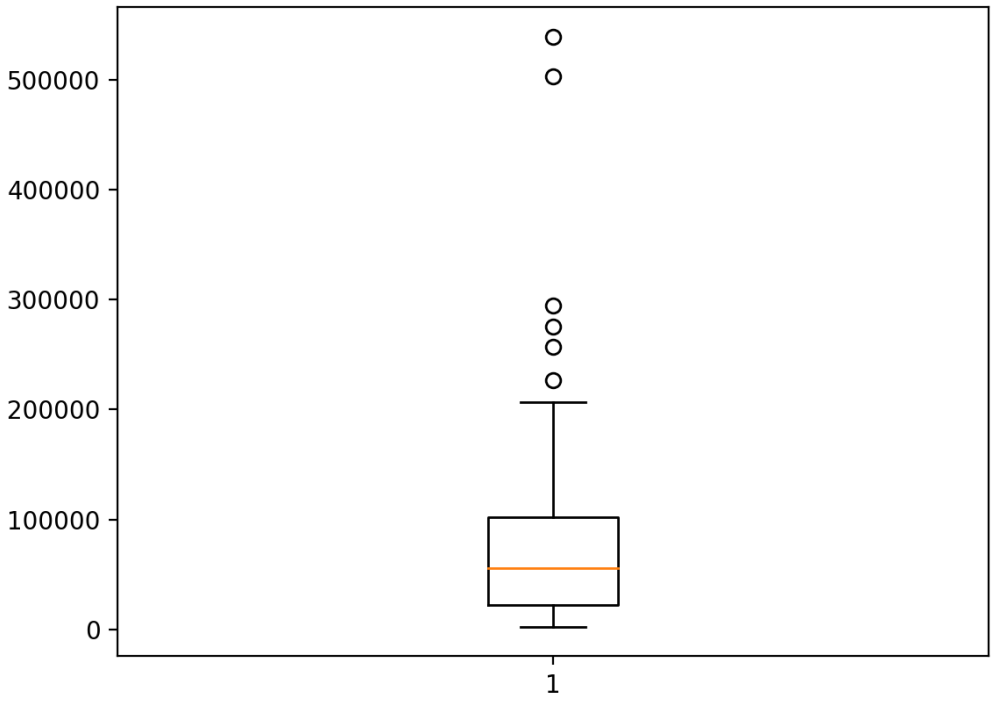
	Python_County_Code	County	County_Code	Population	White_Pop	Percent_White_Pop	Latitude	Longitude
0	1	Adams County	1	20613	17078	0.83	46.9272	-118.5110
1	2	Asotin County	3	22285	20813	0.93	46.1460	-117.2085
2	3	Benton County	5	206873	182727	0.88	46.3166	-119.5022
3	4	Chelan County	7	79074	70583	0.89	47.9445	-120.6749
4	5	Clallam County	9	77155	70469	0.91	48.1134	-123.7986

2 Investigate Population Distribution

[61]:
```
#looking for population breakpoint in case we need to add a constraint to break␣
 ↪up large counties
plt.boxplot(df['Population'])
```

[61]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x29acf6fb0>,
  <matplotlib.lines.Line2D at 0x29acf6d10>],
 'caps': [<matplotlib.lines.Line2D at 0x29acf6a70>,
```

```
 <matplotlib.lines.Line2D at 0x29acf6890>],
'boxes': [<matplotlib.lines.Line2D at 0x29acf7250>],
'medians': [<matplotlib.lines.Line2D at 0x29acf66b0>],
'fliers': [<matplotlib.lines.Line2D at 0x29acf6410>],
'means': []}
```



3 Integer Progamming Model

[62]:
```
#define population bounds
max_pop = 811034
min_pop = 663573

#define district limits
max_districts = 5

#create list of county names
counties = list(df['County'])
total_counties = len(counties)
total_counties
```

```
#create list of county populations
county_pop = df['Population']
len(county_pop)

#create variable names for model
var_names = [str(i) + str(j) for j in range(1, max_districts + 1) for i in␣
 ↪range(1, total_counties + 1)]
len(var_names)
```

[62]: 36

[62]: 36

[62]: 180

[63]:
```
#define minimization problem
WA_Districts = LpProblem("Washington_Districting_Model", LpMinimize)

#define variables

#decision variable - groups of counties
dv_group = LpVariable.matrix("Y", var_names, cat = "Binary")
group = np.array(dv_group).reshape(36, 5)

#decision variable - populations
dv_pop = LpVariable.matrix("X", var_names, cat = "Integer", lowBound = 0)
pop = np.array(dv_pop).reshape(36, 5)

#decision variable - neighboring counties
neighbors = LpVariable.dicts("Adj", [(i, j, k) for i in range(total_counties)
                                     for j in range(total_counties) for k in␣
 ↪range(max_districts)], cat = "Binary")

#objective function to minimize groupings of counties
WA_obj_function = lpSum(group)
WA_Districts += WA_obj_function

#define constraints

#entire county population must be used
for i in range(total_counties):
    for j in range(max_districts):
        WA_Districts += lpSum(pop[i][j] for j in range(max_districts)) ==␣
 ↪county_pop[i], "Total Population" + str(i) + str(j)

#every county must be in one district
for i in range(total_counties):
```

```
    for j in range(max_districts):
        WA_Districts += pop[i][j] <= sum(county_pop)*group[i][j], "Group␣
 ↪Population" + str(i) + str(j)

#districts cannot be over/under min/max population bounds
for j in range(max_districts):
    WA_Districts += lpSum(pop[i][j] for i in range(total_counties)) <= max_pop,␣
 ↪"District Population Maximum" + str(j)
    WA_Districts += lpSum(pop[i][j] for i in range(total_counties)) >= min_pop,␣
 ↪"District Population Minimum" + str(j)

#counties in districts must be adjacent
for k in range(max_districts):
    for i in range(total_counties):
        for j in range(total_counties):
            if i != j and (adj.loc[i, 'neighborname'] == adj.loc[j,␣
 ↪'neighborname']):
                WA_Districts += neighbors[i, j, k] + neighbors[j, i, k] >= 2 *␣
 ↪group[i][k]

#solve objective function
WA_Districts.solve()

print("Optimal solution status:", pulp.LpStatus[WA_Districts.status])
print("Objective value:", value(WA_obj_function))

for i in range(total_counties):
    for j in range(max_districts):
        if pop[i][j].value() > 0:
            print('County %d assigned to district %d: ' % (i, j), pop[i][j].
 ↪value())
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /Users/baileyscoville/anaconda3/lib/python3.10/site-
packages/pulp/solverdir/cbc/osx/64/cbc /var/folders/z0/v3y1p30945d16_whz3lt8v_h0
000gn/T/de24836d71af4e4285911c2e79508949-pulp.mps timeMode elapsed branch
printingOptions all solution /var/folders/z0/v3y1p30945d16_whz3lt8v_h0000gn/T/de
24836d71af4e4285911c2e79508949-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 435 COLUMNS
At line 3256 RHS
At line 3687 BOUNDS
At line 4108 ENDATA
```

```
Problem MODEL has 430 rows, 420 columns and 1800 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 1 - 0.00 seconds
Cgl0004I processed model has 221 rows, 360 columns (360 integer (180 of which
binary)) and 720 elements
Cutoff increment increased from 1e-05 to 0.9999
Cbc0038I Initial state - 8 integers unsatisfied sum - 1.97397
Cbc0038I Pass   1: suminf.    0.37652 (3) obj. 36.3765 iterations 43
Cbc0038I Solution found of 39
Cbc0038I Cleaned solution of 39
Cbc0038I Before mini branch and bound, 340 integers at bound fixed and 0
continuous
Cbc0038I Full problem 221 rows 360 columns, reduced to 13 rows 14 columns
Cbc0038I Mini branch and bound improved solution from 39 to 38 (0.01 seconds)
Cbc0038I Round again with cutoff of 36.9001
Cbc0038I Pass   2: suminf.    0.37652 (3) obj. 36.3765 iterations 0
Cbc0038I Pass   3: suminf.    0.48033 (4) obj. 36.9001 iterations 18
Cbc0038I Pass   4: suminf.    0.48033 (4) obj. 36.9001 iterations 3
Cbc0038I Pass   5: suminf.    0.82734 (4) obj. 36.9001 iterations 23
Cbc0038I Pass   6: suminf.    0.78872 (3) obj. 36.9001 iterations 13
Cbc0038I Pass   7: suminf.    0.48033 (4) obj. 36.9001 iterations 25
Cbc0038I Pass   8: suminf.    1.09991 (3) obj. 36.9001 iterations 48
Cbc0038I Pass   9: suminf.    1.09991 (3) obj. 36.9001 iterations 1
Cbc0038I Pass  10: suminf.    0.86356 (4) obj. 36.3084 iterations 46
Cbc0038I Pass  11: suminf.    0.57723 (4) obj. 36.5772 iterations 6
Cbc0038I Pass  12: suminf.    1.39656 (4) obj. 36.9001 iterations 43
Cbc0038I Pass  13: suminf.    0.71665 (3) obj. 36.9001 iterations 52
Cbc0038I Pass  14: suminf.    1.56545 (6) obj. 36.9001 iterations 57
Cbc0038I Pass  15: suminf.    1.56545 (6) obj. 36.9001 iterations 0
Cbc0038I Pass  16: suminf.    1.10089 (5) obj. 36.9001 iterations 29
Cbc0038I Pass  17: suminf.    0.52588 (3) obj. 36.9001 iterations 18
Cbc0038I Pass  18: suminf.    1.98654 (9) obj. 36.9001 iterations 55
Cbc0038I Pass  19: suminf.    0.42580 (3) obj. 36.9001 iterations 27
Cbc0038I Pass  20: suminf.    0.77214 (3) obj. 36.7876 iterations 52
Cbc0038I Pass  21: suminf.    0.11282 (2) obj. 36.9001 iterations 26
Cbc0038I Pass  22: suminf.    0.88464 (3) obj. 36.9001 iterations 35
Cbc0038I Pass  23: suminf.    2.10421 (9) obj. 36.9001 iterations 73
Cbc0038I Pass  24: suminf.    0.37293 (3) obj. 36.9001 iterations 41
Cbc0038I Pass  25: suminf.    0.50910 (2) obj. 36.5705 iterations 45
Cbc0038I Pass  26: suminf.    0.09991 (1) obj. 36.9001 iterations 32
Cbc0038I Pass  27: suminf.    1.63674 (7) obj. 36.9001 iterations 48
Cbc0038I Pass  28: suminf.    0.81735 (5) obj. 36.9001 iterations 46
Cbc0038I Pass  29: suminf.    1.62707 (5) obj. 36.9001 iterations 53
Cbc0038I Pass  30: suminf.    0.90009 (3) obj. 36.9001 iterations 20
Cbc0038I Pass  31: suminf.    0.55069 (2) obj. 36.5507 iterations 8
Cbc0038I Rounding solution of 37 is better than previous of 38
```

```
Cbc0038I Before mini branch and bound, 176 integers at bound fixed and 0
continuous
Cbc0038I Full problem 221 rows 360 columns, reduced to 115 rows 168 columns
Cbc0038I Mini branch and bound improved solution from 37 to 36 (0.03 seconds)
Cbc0038I After 0.03 seconds - Feasibility pump exiting with objective of 36 -
took 0.02 seconds
Cbc0012I Integer solution of 36 found by feasibility pump after 0 iterations and
0 nodes (0.03 seconds)
Cbc0001I Search completed - best objective 36, took 0 iterations and 0 nodes
(0.03 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from 36 to 36
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                36.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.02
Time (Wallclock seconds):       0.03

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.03   (Wallclock seconds):       0.03
```

[63]: 1

```
Optimal solution status: Optimal
Objective value: 36.0
County 0 assigned to district 2:  20613.0
County 1 assigned to district 4:  22285.0
County 2 assigned to district 4:  206873.0
```

```
County 3 assigned to district 3:  79074.0
County 4 assigned to district 3:  77155.0
County 5 assigned to district 3:  503311.0
County 6 assigned to district 3:  3952.0
County 7 assigned to district 2:  110730.0
County 8 assigned to district 0:  42938.0
County 9 assigned to district 3:  7178.0
County 10 assigned to district 2:  96749.0
County 11 assigned to district 3:  2286.0
County 12 assigned to district 1:  99123.0
County 13 assigned to district 0:  75636.0
County 14 assigned to district 1:  86857.0
County 15 assigned to district 2:  275611.0
County 16 assigned to district 3:  44337.0
County 17 assigned to district 4:  82149.0
County 18 assigned to district 3:  10876.0
County 19 assigned to district 0:  42104.0
County 20 assigned to district 0:  23365.0
County 21 assigned to district 4:  17788.0
County 22 assigned to district 4:  12036.0
County 23 assigned to district 3:  46445.0
County 24 assigned to district 4:  294793.0
County 25 assigned to district 2:  62584.0
County 26 assigned to district 1:  226847.0
County 27 assigned to district 1:  256728.0
County 28 assigned to district 4:  32977.0
County 29 assigned to district 0:  22735.0
County 30 assigned to district 2:  65726.0
County 31 assigned to district 0:  13401.0
County 32 assigned to district 2:  129523.0
County 33 assigned to district 0:  539339.0
County 34 assigned to district 4:  4422.0
County 35 assigned to district 0:  47973.0
```

3.1 Create Model Results Dataframe

[9]:
```
districts_final = []
for i in range(total_counties):
    for j in range(max_districts):
        var_output = {
            'Python_County_Code': i + 1,
            'District': j+1,
            'Assignment': int(group[i][j].value()*(j+1)),
            'Allocation': pop[i][j].value()}
        districts_final.append(var_output)

results = pd.DataFrame(districts_final)
```

```
results = results[results['Assignment'] != 0]
results = results.sort_values(['Python_County_Code', 'District'])
results.head()
```

[9]:

	Python_County_Code	District	Assignment	Allocation
2	1	3	3	20613.0
9	2	5	5	22285.0
14	3	5	5	206873.0
18	4	4	4	79074.0
23	5	4	4	77155.0

[64]:
```
#merge results dataframe with population/county dataframe
df1 = pd.merge(results, df, left_on = 'Python_County_Code', right_on =␣
 ↪'Python_County_Code', how = 'left')
df1.head()
```

[64]:

	Python_County_Code	District	Assignment	Allocation	County	\
0	1	3	3	20613.0	Adams County	
1	2	5	5	22285.0	Asotin County	
2	3	5	5	206873.0	Benton County	
3	4	4	4	79074.0	Chelan County	
4	5	4	4	77155.0	Clallam County	

	County_Code	Population	White_Pop	Percent_White_Pop	Latitude	Longitude
0	1	20613	17078	0.83	46.9272	-118.5110
1	3	22285	20813	0.93	46.1460	-117.2085
2	5	206873	182727	0.88	46.3166	-119.5022
3	7	79074	70583	0.89	47.9445	-120.6749
4	9	77155	70469	0.91	48.1134	-123.7986

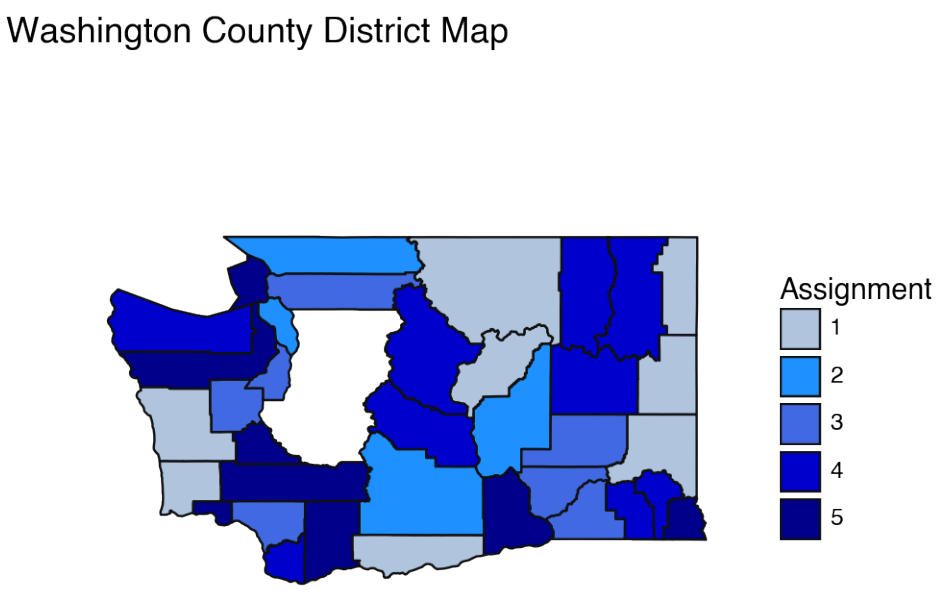
[65]:
```
#confirm district population bounds are met
df2 = df1.groupby(['District'])['Population'].sum()
df2
```

[65]:
```
District
1    807491
2    669555
3    761536
4    774614
5    673323
Name: Population, dtype: int64
```

4 Create District Map

[92]:
```
shade_dict = { 1 : 'lightsteelblue',
               2 : 'dodgerblue',
               3 : 'royalblue',
               4 : 'mediumblue',
               5 : 'darkblue',
               6 : 'grey',
               12: 'grey', 13: 'grey', 14: 'grey', 15: 'grey', 16: 'grey',
               21: 'grey', 23: 'grey', 24: 'grey', 25: 'grey', 26: 'grey',
               31: 'grey', 32: 'grey', 34: 'grey', 35: 'grey', 36: 'grey',
               41: 'grey', 42: 'grey', 43: 'grey', 45: 'grey', 46: 'grey',
               51: 'grey', 52: 'grey', 53: 'grey', 54: 'grey', 56: 'grey',
               61: 'grey', 62: 'grey', 63: 'grey', 64: 'grey', 65: 'grey'}

def WA_map(map_data):
    plot_district_map = (
        ggplot(map_data)
    + geom_map(aes(fill=str('Assignment')))
    + theme_minimal()
    + theme(axis_text_x=element_blank(),
            axis_text_y=element_blank(),
            axis_title_x=element_blank(),
            axis_title_y=element_blank(),
            axis_ticks=element_blank(),
            panel_grid_major=element_blank(),
            panel_grid_minor=element_blank(),
            figure_size=(5, 4))
    + xlim(-14000000, -13000000)
    + ylim(5500000, 6500000)
    + ggtitle('Washington County District Map')
    + scale_fill_manual(values=shade_dict))

    return plot_district_map
```

[93]:
```
map_model1 = shapefile_WA.merge(df1, left_on='JURISDIC_3', right_on='County',␣
 ↪suffixes=('_left', '_right'))
map_model1['District'] = map_model1['District']+1
map_model1_labels = map_model1
map_model1_labels['District'] = map_model1_labels['District'].astype('category')
map_model1_labels['Assignment'] = map_model1_labels['Assignment'].
 ↪astype("category")

WA_map(map_model1_labels)
```



[93]: <Figure Size: (500 x 400)>

5 Re-Run IP Model with Race Constraint

[68]:
```
#define population bounds
max_pop = 811034
min_pop = 663573

#define district limits
max_districts = 5

#create list of county names
counties = list(df['County'])
total_counties = len(counties)
total_counties

#create list of county populations
county_pop = df['Population']
len(county_pop)

#create variable names for model
```

```
var_names = [str(i) + str(j) for j in range(1, max_districts + 1) for i in␣
 ↪range(1, total_counties + 1)]
len(var_names)
```

[68]: 36

[68]: 36

[68]: 180

[69]:
```
#define minimization problem
WA_Districts = LpProblem("Washington_Districting_Model", LpMinimize)

#define variables

#decision variable - groups of counties
dv_group = LpVariable.matrix("Y", var_names, cat = "Binary")
group = np.array(dv_group).reshape(36, 5)

#decision variable - populations
dv_pop = LpVariable.matrix("X", var_names, cat = "Integer", lowBound = 0)
pop = np.array(dv_pop).reshape(36, 5)

#decision variable - neighboring counties
neighbors = LpVariable.dicts("Adj", [(i, j, k) for i in range(total_counties)
                                     for j in range(total_counties) for k in␣
 ↪range(max_districts)], cat = "Binary")

#decision variable - % white of population
dv_white_pop = LpVariable.matrix("W", var_names, cat = "Continuous", lowBound =␣
 ↪0)
white_pop = np.array(dv_white_pop).reshape(36, 5)

#objective function to minimize groupings of counties
WA_obj_function = lpSum(group)
WA_Districts += WA_obj_function

#define constraints

#entire county population must be used
for i in range(total_counties):
    for j in range(max_districts):
        WA_Districts += lpSum(pop[i][j] for j in range(max_districts)) ==␣
 ↪county_pop[i], "Total Population" + str(i) + str(j)

#every county must be in one district
for i in range(total_counties):
```

```
    for j in range(max_districts):
        WA_Districts += pop[i][j] <= sum(county_pop)*group[i][j], "Group␣
 ↪Population" + str(i) + str(j)

#districts cannot be over/under min/max population bounds
for j in range(max_districts):
    WA_Districts += lpSum(pop[i][j] for i in range(total_counties)) <= max_pop,␣
 ↪"District Population Maximum" + str(j)
    WA_Districts += lpSum(pop[i][j] for i in range(total_counties)) >= min_pop,␣
 ↪"District Population Minimum" + str(j)

#add equal percentage white voters constraint
for j in range(max_districts):
    WA_Districts += lpSum(white_pop[i][j] for i in range(total_counties)) ==␣
 ↪lpSum(white_pop_percent[i] * pop[i][j] for i in range(total_counties)),␣
 ↪"Equal White Population %" + str(j)

#counties in districts must be adjacent
for k in range(max_districts):
    for i in range(total_counties):
        for j in range(total_counties):
            if i != j and (adj.loc[i, 'neighborname'] == adj.loc[j,␣
 ↪'neighborname']):
                WA_Districts += neighbors[i, j, k] + neighbors[j, i, k] >= 2 *␣
 ↪group[i][k]

#solve objective function
WA_Districts.solve()

print("Optimal solution status:", pulp.LpStatus[WA_Districts.status])
print("Objective value:", value(WA_obj_function))

for i in range(total_counties):
    for j in range(max_districts):
        if pop[i][j].value() > 0:
            print('County %d assigned to district %d: ' % (i, j), pop[i][j].
 ↪value())
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /Users/baileyscoville/anaconda3/lib/python3.10/site-
packages/pulp/solverdir/cbc/osx/64/cbc /var/folders/z0/v3y1p30945d16_whz3lt8v_h0
000gn/T/ee398bfa91614e2da38f0a42ee0e9270-pulp.mps timeMode elapsed branch
printingOptions all solution /var/folders/z0/v3y1p30945d16_whz3lt8v_h0000gn/T/ee
398bfa91614e2da38f0a42ee0e9270-pulp.sol (default strategy 1)
```

```
At line 2 NAME          MODEL
At line 3 ROWS
At line 440 COLUMNS
At line 3621 RHS
At line 4057 BOUNDS
At line 4478 ENDATA
Problem MODEL has 435 rows, 600 columns and 2160 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 1 - 0.00 seconds
Cgl0004I processed model has 221 rows, 360 columns (360 integer (180 of which
binary)) and 720 elements
Cutoff increment increased from 1e-05 to 0.9999
Cbc0038I Initial state - 6 integers unsatisfied sum - 1.51244
Cbc0038I Pass   1: suminf.    0.28424 (3) obj. 36.2842 iterations 43
Cbc0038I Solution found of 39
Cbc0038I Cleaned solution of 39
Cbc0038I Before mini branch and bound, 344 integers at bound fixed and 0
continuous
Cbc0038I Full problem 221 rows 360 columns, reduced to 10 rows 11 columns
Cbc0038I Mini branch and bound improved solution from 39 to 37 (0.01 seconds)
Cbc0038I Round again with cutoff of 36.0001
Cbc0038I Pass   2: suminf.    0.56839 (5) obj. 36.0001 iterations 32
Cbc0038I Pass   3: suminf.    1.68954 (5) obj. 36.0001 iterations 45
Cbc0038I Pass   4: suminf.    1.19589 (5) obj. 36.0001 iterations 3
Cbc0038I Pass   5: suminf.    1.07069 (5) obj. 36.0001 iterations 50
Cbc0038I Pass   6: suminf.    0.57704 (5) obj. 36.0001 iterations 3
Cbc0038I Pass   7: suminf.    1.54707 (7) obj. 36.0001 iterations 35
Cbc0038I Pass   8: suminf.    0.45054 (5) obj. 36.0001 iterations 56
Cbc0038I Pass   9: suminf.    0.45054 (5) obj. 36.0001 iterations 2
Cbc0038I Pass  10: suminf.    1.47222 (5) obj. 36.0001 iterations 65
Cbc0038I Pass  11: suminf.    1.06835 (5) obj. 36.0001 iterations 34
Cbc0038I Pass  12: suminf.    0.79182 (6) obj. 36.0001 iterations 13
Cbc0038I Pass  13: suminf.    2.14694 (7) obj. 36.0001 iterations 69
Cbc0038I Pass  14: suminf.    1.74873 (5) obj. 36.0001 iterations 7
Cbc0038I Pass  15: suminf.    1.00009 (4) obj. 36.0001 iterations 57
Cbc0038I Pass  16: suminf.    1.00000 (4) obj. 36 iterations 35
Cbc0038I Pass  17: suminf.    0.97638 (3) obj. 36.0001 iterations 37
Cbc0038I Pass  18: suminf.    0.97638 (3) obj. 36.0001 iterations 4
Cbc0038I Pass  19: suminf.    1.04815 (5) obj. 36.0001 iterations 53
Cbc0038I Pass  20: suminf.    2.21906 (8) obj. 36.0001 iterations 25
Cbc0038I Pass  21: suminf.    1.00000 (5) obj. 36 iterations 59
Cbc0038I Pass  22: suminf.    0.89361 (3) obj. 36.0001 iterations 15
Cbc0038I Pass  23: suminf.    0.89361 (3) obj. 36.0001 iterations 8
Cbc0038I Pass  24: suminf.    1.00009 (3) obj. 36.0001 iterations 26
Cbc0038I Pass  25: suminf.    1.27394 (8) obj. 36.0001 iterations 73
Cbc0038I Pass  26: suminf.    0.68967 (2) obj. 36.0001 iterations 82
Cbc0038I Pass  27: suminf.    0.73918 (2) obj. 36.0001 iterations 42
```

```
Cbc0038I Pass  28: suminf.    1.21646 (7) obj. 36.0001 iterations 44
Cbc0038I Pass  29: suminf.    0.24486 (3) obj. 36.0001 iterations 55
Cbc0038I Pass  30: suminf.    1.19887 (5) obj. 36.0001 iterations 64
Cbc0038I Pass  31: suminf.    0.45796 (3) obj. 36 iterations 53
Cbc0038I Before mini branch and bound, 185 integers at bound fixed and 0
continuous
Cbc0038I Full problem 221 rows 360 columns, reduced to 108 rows 157 columns
Cbc0038I Mini branch and bound improved solution from 37 to 36 (0.02 seconds)
Cbc0038I After 0.02 seconds - Feasibility pump exiting with objective of 36 -
took 0.02 seconds
Cbc0012I Integer solution of 36 found by feasibility pump after 0 iterations and
0 nodes (0.02 seconds)
Cbc0001I Search completed - best objective 36, took 0 iterations and 0 nodes
(0.02 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from 36 to 36
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                36.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.02
Time (Wallclock seconds):       0.03

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.03   (Wallclock seconds):       0.03
```

[69]: 1

```
Optimal solution status: Optimal
```

```
Objective value: 36.0
County 0 assigned to district 2:  20613.0
County 1 assigned to district 4:  22285.0
County 2 assigned to district 4:  206873.0
County 3 assigned to district 3:  79074.0
County 4 assigned to district 4:  77155.0
County 5 assigned to district 3:  503311.0
County 6 assigned to district 1:  3952.0
County 7 assigned to district 2:  110730.0
County 8 assigned to district 4:  42938.0
County 9 assigned to district 4:  7178.0
County 10 assigned to district 3:  96749.0
County 11 assigned to district 2:  2286.0
County 12 assigned to district 1:  99123.0
County 13 assigned to district 0:  75636.0
County 14 assigned to district 2:  86857.0
County 15 assigned to district 2:  275611.0
County 16 assigned to district 3:  44337.0
County 17 assigned to district 0:  82149.0
County 18 assigned to district 2:  10876.0
County 19 assigned to district 2:  42104.0
County 20 assigned to district 2:  23365.0
County 21 assigned to district 3:  17788.0
County 22 assigned to district 0:  12036.0
County 23 assigned to district 1:  46445.0
County 24 assigned to district 4:  294793.0
County 25 assigned to district 1:  62584.0
County 26 assigned to district 1:  226847.0
County 27 assigned to district 1:  256728.0
County 28 assigned to district 4:  32977.0
County 29 assigned to district 0:  22735.0
County 30 assigned to district 2:  65726.0
County 31 assigned to district 3:  13401.0
County 32 assigned to district 2:  129523.0
County 33 assigned to district 0:  539339.0
County 34 assigned to district 2:  4422.0
County 35 assigned to district 1:  47973.0
```

5.1 Create Model Results Dataframe

[70]:
```
districts_final = []
for i in range(total_counties):
    for j in range(max_districts):
        var_output = {
            'Python_County_Code': i + 1,
            'District': j+1,
            'Assignment': int(group[i][j].value()*(j+1)),
```

```
            'Allocation': pop[i][j].value(),
         'Percent White': white_pop[i][j].value()}
        districts_final.append(var_output)

results = pd.DataFrame(districts_final)
results = results[results['Assignment'] != 0]
results = results.sort_values(['Python_County_Code', 'District'])
results.head()
```

[70]:

	Python_County_Code	District	Assignment	Allocation	Percent White
2	1	3	3	20613.0	0.00
9	2	5	5	22285.0	606271.46
14	3	5	5	206873.0	0.00
18	4	4	4	79074.0	0.00
24	5	5	5	77155.0	0.00

[72]:
```
#merge results dataframe with population/county dataframe
df3 = pd.merge(results, df, left_on = 'Python_County_Code', right_on =␣
 ↪'Python_County_Code', how = 'left')
df3 = df3.sort_values(by = ['Assignment'], ascending = False)
df3.head()
```

[72]:

	Python_County_Code	District	Assignment	Allocation	Percent White	\
9	10	5	5	7178.0	0.00	
8	9	5	5	42938.0	0.00	
24	25	5	5	294793.0	0.00	
28	29	5	5	32977.0	0.00	
1	2	5	5	22285.0	606271.46	

	County	County_Code	Population	White_Pop	Percent_White_Pop	\
9	Ferry County	19	7178	6602	0.92	
8	Douglas County	17	42938	36824	0.86	
24	Thurston County	67	294793	260633	0.88	
28	Jefferson County	31	32977	30425	0.92	
1	Asotin County	3	22285	20813	0.93	

	Latitude	Longitude
9	48.4718	-118.4974
8	47.7791	-119.7475
24	46.8646	-122.7696
28	47.7425	-123.3040
1	46.1460	-117.2085

[75]:
```
#confirm district population bounds are met
df4 = df3.groupby(['District'])['Population'].sum()
df4
```

```
#compare with previous model without % white population constraint
df2
```

[75]:
```
District
1    731895
2    743652
3    772113
4    754660
5    684199
Name: Population, dtype: int64
```

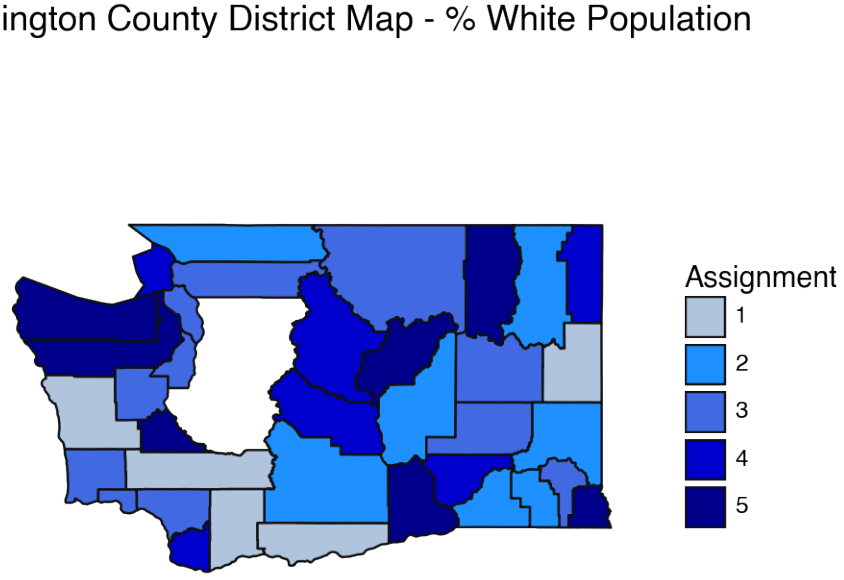
[75]:
```
District
1    807491
2    669555
3    761536
4    774614
5    673323
Name: Population, dtype: int64
```

6 Create District Map

[90]:
```
shade_dict = { 1 : 'lightsteelblue',
               2 : 'dodgerblue',
               3 : 'royalblue',
               4 : 'mediumblue',
               5 : 'darkblue',
               6 : 'grey',
               12: 'grey', 13: 'grey', 14: 'grey', 15: 'grey', 16: 'grey',
               21: 'grey', 23: 'grey', 24: 'grey', 25: 'grey', 26: 'grey',
               31: 'grey', 32: 'grey', 34: 'grey', 35: 'grey', 36: 'grey',
               41: 'grey', 42: 'grey', 43: 'grey', 45: 'grey', 46: 'grey',
               51: 'grey', 52: 'grey', 53: 'grey', 54: 'grey', 56: 'grey',
               61: 'grey', 62: 'grey', 63: 'grey', 64: 'grey', 65: 'grey'}

def WA_map(map_data):
    plot_district_map = (
        ggplot(map_data)
    + geom_map(aes(fill=str('Assignment')))
    + theme_minimal()
    + theme(axis_text_x=element_blank(),
            axis_text_y=element_blank(),
            axis_title_x=element_blank(),
            axis_title_y=element_blank(),
            axis_ticks=element_blank(),
            panel_grid_major=element_blank(),
            panel_grid_minor=element_blank(),
```

```
            figure_size=(5, 4))
    + xlim(-14000000, -13000000)
    + ylim(5500000, 6500000)
    + ggtitle('Washington County District Map - % White Population')
    + scale_fill_manual(values=shade_dict))

    return plot_district_map
```

[91]:
```
map_model2 = shapefile_WA.merge(df3, left_on='JURISDIC_3', right_on='County',␣
 ↪suffixes=('_left', '_right'))
map_model2['District'] = map_model2['District']+1
map_model2_labels = map_model2
map_model2_labels['District'] = map_model2_labels['District'].astype('category')
map_model2_labels['Assignment'] = map_model2_labels['Assignment'].
 ↪astype("category")

WA_map(map_model2_labels)
```



[91]: <Figure Size: (500 x 400)>

6.0.1 Export District Assignment Dataframes

[96]:
```
df1.to_csv('model1.csv', header=True, index=False)
df3.to_csv('model2.csv', header=True, index=False)
```