Sanders Data Targeting Strategy

April 21, 2019

1 Bernie Sanders Campaign Plan

1.0.1 Field Targeting: The Path to 270

1.0.2 Author: Josh Kuppersmith and the Sanders Campaign Team

1.0.3 Date: April 25, 2019

Description: This notebook will walk through a brief clustering analysis of 2020 polling data to provide projections and clusters of states that we expect to behave similarly in the general election. State clusters are used to determine targets for advertisement and field operations. In-house mapping software is used to provide visuals of our expectations for the election. Target states are then analyzed further by breaking down data for individual counties to determine which to target.

Import software needed for this analysis

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        from sklearn.cluster import DBSCAN
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import PolynomialFeatures
        import statsmodels.api as sm
        import matplotlib.style
        import matplotlib as mpl
        import geopandas as gpd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.colors
        pd.set_option('display.max_rows', 500)
        pd.set_option('display.max_columns', 500)
        pd.set_option('display.width', 1000)
        mpl.style.use('classic')
```

---

2 State Analysis: Path to 270

Load Polling Data

In [2]:
```
data = pd.read_csv('polling_data.csv')
data.head()
```

Out[2]:

	Unnamed: 0	2012_D	2012_R	2012_I	2016_D	2016_R	2016_I	2020_D	2020_R	2020_I	E
0	AL	38.40%	60.50%	1.10%	34.40%	62.10%	3.60%	35.20%	61.80%	3.10%	6
1	AK	41.20%	55.30%	3.50%	36.60%	51.30%	12.20%	37.50%	52.10%	10.40%	4
2	AZ	44.60%	53.70%	1.80%	45.10%	48.70%	6.20%	49.70%	45.00%	5.30%	7
3	AR	36.90%	60.60%	2.60%	33.70%	60.60%	5.80%	34.30%	60.60%	5.10%	6
4	CA	60.20%	37.10%	2.60%	61.70%	31.60%	6.70%	46.40%	32.70%	20.90%	5

Data Processing

In [3]:
```
for column in ['2012_D', '2012_R', '2012_I', '2016_D', '2016_R', '2016_I', '2020_D', '
    data[column] = data[column].apply(lambda x: float(x.split('%')[0]))
data.head()
```

Out[3]:

	Unnamed: 0	2012_D	2012_R	2012_I	2016_D	2016_R	2016_I	2020_D	2020_R	2020_I	E
0	AL	38.4	60.5	1.1	34.4	62.1	3.6	35.2	61.8	3.1	
1	AK	41.2	55.3	3.5	36.6	51.3	12.2	37.5	52.1	10.4	
2	AZ	44.6	53.7	1.8	45.1	48.7	6.2	49.7	45.0	5.3	
3	AR	36.9	60.6	2.6	33.7	60.6	5.8	34.3	60.6	5.1	
4	CA	60.2	37.1	2.6	61.7	31.6	6.7	46.4	32.7	20.9	

Generate 2024 Projections using quadratic regression to account for the Blue Wave

In [4]:
```
# add 2024 columns for better predictions
data['2024_D'] = [0] * len(data)
data['2024_R'] = [0] * len(data)
data['2024_I'] = [0] * len(data)

for index, row in data.iterrows():

    indices = np.array([2012.0,2016.0,2020.0])
    test_index = np.array(2024.0)
    dems_train = np.array(row[['2012_D', '2016_D', '2020_D']]).reshape(1, -1)
    reps_train = np.array(row[['2012_R', '2016_R', '2020_R']]).reshape(1, -1)
    inds_train = np.array(row[['2012_I', '2016_I', '2020_I']]).reshape(1, -1)

    model_1 = sm.OLS(dems_train[0], indices).fit()
    data.at[index, '2024_D'] = model_1.predict(test_index)

    model_2 = sm.OLS(reps_train[0], indices).fit()
    data.at[index, '2024_R'] = model_2.predict(test_index)
```

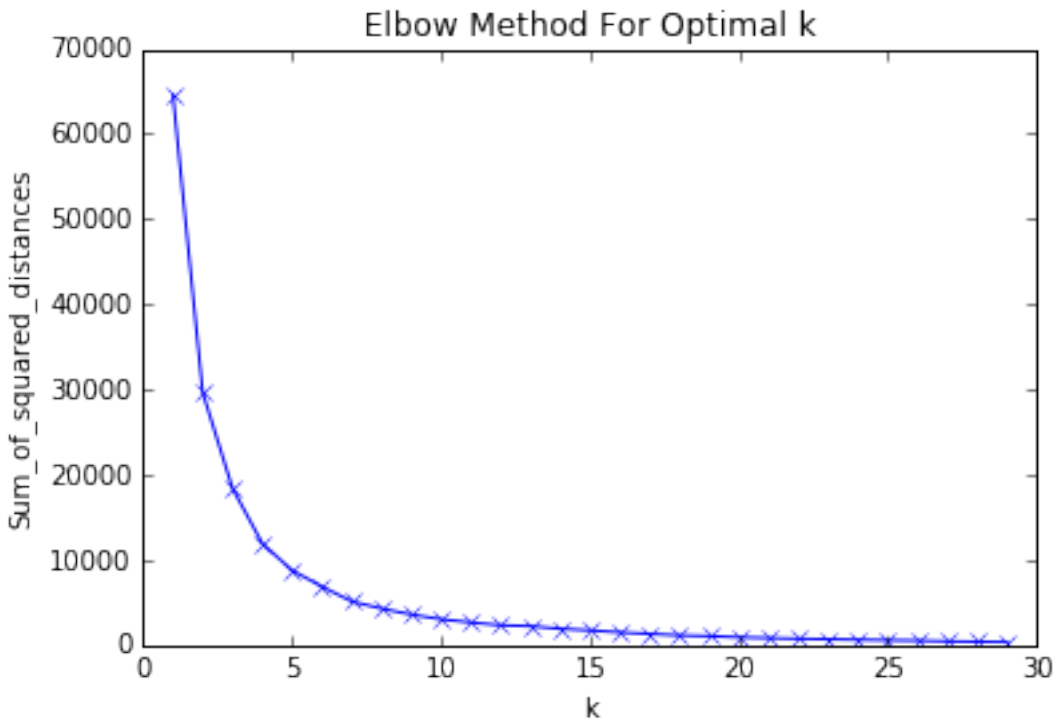
```
        model_3 = sm.OLS(inds_train[0], indices).fit()
        data.at[index, '2024_I'] = model_3.predict(test_index)

    data.head()
```

Out[4]:

	Unnamed: 0	2012_D	2012_R	2012_I	2016_D	2016_R	2016_I	2020_D	2020_R	2020_I
0	AL	38.4	60.5	1.1	34.4	62.1	3.6	35.2	61.8	3.1
1	AK	41.2	55.3	3.5	36.6	51.3	12.2	37.5	52.1	10.4
2	AZ	44.6	53.7	1.8	45.1	48.7	6.2	49.7	45.0	5.3
3	AR	36.9	60.6	2.6	33.7	60.6	5.8	34.3	60.6	5.1
4	CA	60.2	37.1	2.6	61.7	31.6	6.7	46.4	32.7	20.9

Elbow Plot for Rigorous Optimal Number of Clusters

In [5]:
```
# elbow plot for best number of clusters

cols = ['2012_D', '2012_R', '2012_I', '2016_D', '2016_R', '2016_I', '2020_D', '2020_R'
Sum_of_squared_distances = []
K = range(1,30)
for k in K:
    km = KMeans(n_clusters=k, random_state=0).fit(data[cols])
    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Perform and Save Results for Optimal K=7

```
In [6]: kmeans = KMeans(n_clusters=7, random_state=0).fit(data[cols])

        labels = kmeans.labels_
        data['clust_labels'] = labels
```

Print Results

```
In [7]: cluster_data = []

        for label in list(set(list(data['clust_labels']))):
            data_clust = data[data['clust_labels'] == label]
            individual = {}
            print("Cluster Number: " + str(label))
            print("States: " + str(list(data_clust['Unnamed: 0'])))
            print("Total electoral votes: " + str(sum(data_clust['Votes'])))
            print("2020 Average Dem Support: " + str(np.mean(data_clust['2020_D'])))
            print("2020 Average Rep Support: " + str(np.mean(data_clust['2020_R'])))
            individual['num'] = label
            individual['votes'] = sum(data_clust['Votes'])
            individual['dems'] = np.mean(data_clust['2020_D'])
            individual['reps'] = np.mean(data_clust['2020_R'])
```

```
            individual['states'] = str(list(data_clust['Unnamed: 0']))
            cluster_data.append(individual)
            print()
            print('------------------')
            print()
```

```
Cluster Number: 0
States: ['AL', 'AK', 'AR', 'IN', 'KS', 'KY', 'LA', 'MS', 'MO', 'MT', 'NE 1', 'SC', 'SD', 'TN',
Total electoral votes: 132
2020 Average Dem Support: 37.459999999999994
2020 Average Rep Support: 57.58

------------------

Cluster Number: 1
States: ['CT', 'DE', 'HI', 'IL', 'Maine 1', 'MD', 'MA', 'NJ ', 'OR', 'RI']
Total electoral votes: 81
2020 Average Dem Support: 56.07000000000001
2020 Average Rep Support: 37.74

------------------

Cluster Number: 2
States: ['AZ', 'CO', 'FL', 'GA', 'IA', 'Maine 2', 'MI', 'MN', 'NE 2', 'NV', 'NH', 'NC', 'OH',
Total electoral votes: 185
2020 Average Dem Support: 46.756249999999994
2020 Average Rep Support: 47.293749999999996

------------------

Cluster Number: 3
States: ['DC']
Total electoral votes: 3
2020 Average Dem Support: 90.9
2020 Average Rep Support: 4.7

------------------

Cluster Number: 4
States: ['CA', 'NM', 'NY', 'VT', 'WA']
Total electoral votes: 104
2020 Average Dem Support: 45.58
2020 Average Rep Support: 32.42

------------------

Cluster Number: 5
States: ['UT']
```

```
Total electoral votes: 6
2020 Average Dem Support: 24.9
2020 Average Rep Support: 45.0

------------------

Cluster Number: 6
States: ['ID', 'NE 3', 'ND', 'OK', 'WV', 'WY']
Total electoral votes: 23
2020 Average Dem Support: 26.73333333333333
2020 Average Rep Support: 66.11666666666667

------------------
```

Analyze individually states in the 'Battleground' Category

In [8]:
```
import copy

battleground_states = copy.deepcopy(data[data.clust_labels==2])
kmeans = KMeans(n_clusters=4, random_state=0).fit(battleground_states[cols])

border_labels = kmeans.labels_
battleground_states['border_clusters'] = border_labels
```

Perform KMeans Clustering for k=16 and Print Results

In [9]:
```
kmeans = KMeans(n_clusters=16, random_state=0).fit(data[cols])

labels = kmeans.labels_
data['clust_labels_take2'] = labels

cluster_data_2 = []

for label in list(set(list(data['clust_labels_take2']))):
    data_clust = data[data['clust_labels_take2'] == label]
    individual = {}
    print("Cluster Number: " + str(label))
    print("States: " + str(list(data_clust['Unnamed: 0'])))
    print("Total electoral votes: " + str(sum(data_clust['Votes'])))
    print("2020 Average Dem Support: " + str(np.mean(data_clust['2020_D'])))
    print("2020 Average Rep Support: " + str(np.mean(data_clust['2020_R'])))
    individual['num'] = label
    individual['votes'] = sum(data_clust['Votes'])
    individual['dems'] = np.mean(data_clust['2020_D'])
    individual['reps'] = np.mean(data_clust['2020_R'])
    individual['states'] = str(list(data_clust['Unnamed: 0']))
```

```
            cluster_data_2.append(individual)
            print()
            print('------------------')
            print()
```

```
Cluster Number: 0
States: ['AZ', 'FL', 'GA', 'NE 2', 'NC']
Total electoral votes: 72
2020 Average Dem Support: 47.059999999999995
2020 Average Rep Support: 48.720000000000006

------------------

Cluster Number: 1
States: ['DC']
Total electoral votes: 3
2020 Average Dem Support: 90.9
2020 Average Rep Support: 4.7

------------------

Cluster Number: 2
States: ['MD', 'MA']
Total electoral votes: 21
2020 Average Dem Support: 60.45
2020 Average Rep Support: 34.05

------------------

Cluster Number: 3
States: ['NE 3', 'WY']
Total electoral votes: 4
2020 Average Dem Support: 22.200000000000003
2020 Average Rep Support: 70.75

------------------

Cluster Number: 4
States: ['AL', 'AR', 'KY', 'SD', 'TN']
Total electoral votes: 37
2020 Average Dem Support: 34.44
2020 Average Rep Support: 61.160000000000004

------------------

Cluster Number: 5
States: ['CO', 'MI', 'MN', 'NV', 'PA', 'VA', 'WI']
Total electoral votes: 84
```

```
2020 Average Dem Support: 48.528571428571425
2020 Average Rep Support: 45.942857142857136

------------------

Cluster Number: 6
States: ['UT']
Total electoral votes: 6
2020 Average Dem Support: 24.9
2020 Average Rep Support: 45.0

------------------

Cluster Number: 7
States: ['CA', 'NY', 'VT']
Total electoral votes: 87
2020 Average Dem Support: 46.79999999999999
2020 Average Rep Support: 31.26666666666667

------------------

Cluster Number: 8
States: ['AK', 'KS', 'MT', 'NE 1']
Total electoral votes: 13
2020 Average Dem Support: 36.825
2020 Average Rep Support: 55.45

------------------

Cluster Number: 9
States: ['NH', 'NM', 'WA']
Total electoral votes: 21
2020 Average Dem Support: 42.79999999999999
2020 Average Rep Support: 36.266666666666666

------------------

Cluster Number: 10
States: ['CT', 'DE', 'IL', 'Maine 1', 'NJ ', 'OR', 'RI']
Total electoral votes: 56
2020 Average Dem Support: 53.699999999999996
2020 Average Rep Support: 39.957142857142856

------------------

Cluster Number: 11
States: ['OK', 'WV']
Total electoral votes: 12
```

```
2020 Average Dem Support: 29.0
2020 Average Rep Support: 66.44999999999999

------------------

Cluster Number: 12
States: ['IN', 'LA', 'MS', 'MO', 'SC', 'TX']
Total electoral votes: 82
2020 Average Dem Support: 40.4
2020 Average Rep Support: 56.01666666666666

------------------

Cluster Number: 13
States: ['HI']
Total electoral votes: 4
2020 Average Dem Support: 63.9
2020 Average Rep Support: 29.6

------------------

Cluster Number: 14
States: ['ID', 'ND']
Total electoral votes: 7
2020 Average Dem Support: 29.0
2020 Average Rep Support: 61.15

------------------

Cluster Number: 15
States: ['IA', 'Maine 2', 'OH']
Total electoral votes: 25
2020 Average Dem Support: 44.06666666666666
2020 Average Rep Support: 50.333333333333336

------------------
```

Rank Clustering by Democratic Support (D-R)

```
In [10]: # make a column for ranked clusters by support

         dem_supports = []
         for label in list(set(list(data['clust_labels_take2']))):
             data_clust = data[data['clust_labels_take2'] == label]
             dem_support = np.mean(data_clust['2020_D']) - np.mean(data_clust['2020_R'])
             dem_supports.append({'index': label, 'support': dem_support})
```

```
        new_support = sorted(dem_supports, reverse=True, key=lambda k: k['support'])

        ranked_cluster_list = []
        for i in range(len(data)):
            cluster = list(data.clust_labels_take2)[i]
            for j in range(len(new_support)):
                if new_support[j]['index'] == cluster:
                    ranked_cluster_list.append(j)

        data['ranked_cluster'] = ranked_cluster_list
```

Tipping Point Analysis

In [11]:
```
# GO THROUGH AND SHOW TIPPING POINT

print("TIPPING POINT ANALYSIS")
print("RANKING CLUSTERS BY DEMOCRATIC FAVORABILITY")
print()
print()
print()

max_clust = np.max(np.array(list(data.ranked_cluster)))
total_states = []
total_votes = 0

for i in range(max_clust):
    data_clust = data[data['ranked_cluster'] == i]

    print("Step " + str(i) + ":")
    print("States Won: " + str(list(data_clust['Unnamed: 0'])))
    total_states.extend(list(data_clust['Unnamed: 0']))
    print("New States Dem Pref: " + str(np.mean(data_clust['2020_D'])) + ", New State
    print("Total States: " + str(total_states))
    print("Electoral Votes Won: " + str(sum(data_clust['Votes'])))
    total_votes = total_votes + sum(data_clust['Votes'])
    print("Total Electoral Votes: " + str(total_votes))
    print()
    print('----------------------------------------')
    print()
```

```
TIPPING POINT ANALYSIS
RANKING CLUSTERS BY DEMOCRATIC FAVORABILITY



Step 0:
```

```
States Won: ['DC']
New States Dem Pref: 90.9, New States Rep Pref: 4.7
Total States: ['DC']
Electoral Votes Won: 3
Total Electoral Votes: 3

----------------------------------------

Step 1:
States Won: ['HI']
New States Dem Pref: 63.9, New States Rep Pref: 29.6
Total States: ['DC', 'HI']
Electoral Votes Won: 4
Total Electoral Votes: 7

----------------------------------------

Step 2:
States Won: ['MD', 'MA']
New States Dem Pref: 60.45, New States Rep Pref: 34.05
Total States: ['DC', 'HI', 'MD', 'MA']
Electoral Votes Won: 21
Total Electoral Votes: 28

----------------------------------------

Step 3:
States Won: ['CA', 'NY', 'VT']
New States Dem Pref: 46.79999999999999, New States Rep Pref: 31.26666666666667
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT']
Electoral Votes Won: 87
Total Electoral Votes: 115

----------------------------------------

Step 4:
States Won: ['CT', 'DE', 'IL', 'Maine 1', 'NJ ', 'OR', 'RI']
New States Dem Pref: 53.699999999999996, New States Rep Pref: 39.957142857142856
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 56
Total Electoral Votes: 171

----------------------------------------

Step 5:
States Won: ['NH', 'NM', 'WA']
New States Dem Pref: 42.79999999999999, New States Rep Pref: 36.266666666666666
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
```

```
Electoral Votes Won: 21
Total Electoral Votes: 192

----------------------------------------

Step 6:
States Won: ['CO', 'MI', 'MN', 'NV', 'PA', 'VA', 'WI']
New States Dem Pref: 48.528571428571425, New States Rep Pref: 45.942857142857136
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 84
Total Electoral Votes: 276

----------------------------------------

Step 7:
States Won: ['AZ', 'FL', 'GA', 'NE 2', 'NC']
New States Dem Pref: 47.059999999999995, New States Rep Pref: 48.720000000000006
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 72
Total Electoral Votes: 348

----------------------------------------

Step 8:
States Won: ['IA', 'Maine 2', 'OH']
New States Dem Pref: 44.06666666666666, New States Rep Pref: 50.333333333333336
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 25
Total Electoral Votes: 373

----------------------------------------

Step 9:
States Won: ['IN', 'LA', 'MS', 'MO', 'SC', 'TX']
New States Dem Pref: 40.4, New States Rep Pref: 56.01666666666666
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 82
Total Electoral Votes: 455

----------------------------------------

Step 10:
States Won: ['AK', 'KS', 'MT', 'NE 1']
New States Dem Pref: 36.825, New States Rep Pref: 55.45
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 13
Total Electoral Votes: 468
```

```
----------------------------------------

Step 11:
States Won: ['UT']
New States Dem Pref: 24.9, New States Rep Pref: 45.0
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 6
Total Electoral Votes: 474

----------------------------------------

Step 12:
States Won: ['AL', 'AR', 'KY', 'SD', 'TN']
New States Dem Pref: 34.44, New States Rep Pref: 61.160000000000004
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 37
Total Electoral Votes: 511

----------------------------------------

Step 13:
States Won: ['ID', 'ND']
New States Dem Pref: 29.0, New States Rep Pref: 61.15
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 7
Total Electoral Votes: 518

----------------------------------------

Step 14:
States Won: ['OK', 'WV']
New States Dem Pref: 29.0, New States Rep Pref: 66.44999999999999
Total States: ['DC', 'HI', 'MD', 'MA', 'CA', 'NY', 'VT', 'CT', 'DE', 'IL', 'Maine 1', 'NJ ', '(
Electoral Votes Won: 12
Total Electoral Votes: 530

----------------------------------------
```

2.0.1 Visualize Results

Load State Boundary Data for Mapping

```
In [12]: us_map = gpd.read_file('states_21basic/states.shp')
         us_map
```

Out[12]:

	STATE_NAME	DRAWSEQ	STATE_FIPS	SUB_REGION	STATE_ABBR	
0	Hawaii	1	15	Pacific	HI	(POLYGON

1	Washington	2	53	Pacific	WA	(POLYGON
2	Montana	3	30	Mountain	MT	POLYGON
3	Maine	4	23	New England	ME	(POLYGON
4	North Dakota	5	38	West North Central	ND	POLYGON
5	South Dakota	6	46	West North Central	SD	POLYGON
6	Wyoming	7	56	Mountain	WY	POLYGON
7	Wisconsin	8	55	East North Central	WI	(POLYGON
8	Idaho	9	16	Mountain	ID	POLYGON
9	Vermont	10	50	New England	VT	POLYGON
10	Minnesota	11	27	West North Central	MN	POLYGON
11	Oregon	12	41	Pacific	OR	POLYGON
12	New Hampshire	13	33	New England	NH	POLYGON
13	Iowa	14	19	West North Central	IA	POLYGON
14	Massachusetts	15	25	New England	MA	(POLYGON
15	Nebraska	16	31	West North Central	NE	POLYGON
16	New York	17	36	Middle Atlantic	NY	(POLYGON
17	Pennsylvania	18	42	Middle Atlantic	PA	POLYGON
18	Connecticut	19	09	New England	CT	POLYGON
19	Rhode Island	20	44	New England	RI	(POLYGON
20	New Jersey	21	34	Middle Atlantic	NJ	POLYGON
21	Indiana	22	18	East North Central	IN	POLYGON
22	Nevada	23	32	Mountain	NV	POLYGON
23	Utah	24	49	Mountain	UT	POLYGON
24	California	25	06	Pacific	CA	(POLYGON
25	Ohio	26	39	East North Central	OH	POLYGON
26	Illinois	27	17	East North Central	IL	POLYGON
27	District of Columbia	28	11	South Atlantic	DC	POLYGON
28	Delaware	29	10	South Atlantic	DE	POLYGON
29	West Virginia	30	54	South Atlantic	WV	POLYGON
30	Maryland	31	24	South Atlantic	MD	(POLYGON
31	Colorado	32	08	Mountain	CO	POLYGON
32	Kentucky	33	21	East South Central	KY	(POLYGON
33	Kansas	34	20	West North Central	KS	POLYGON
34	Virginia	35	51	South Atlantic	VA	(POLYGON
35	Missouri	36	29	West North Central	MO	POLYGON
36	Arizona	37	04	Mountain	AZ	POLYGON
37	Oklahoma	38	40	West South Central	OK	POLYGON
38	North Carolina	39	37	South Atlantic	NC	(POLYGON
39	Tennessee	40	47	East South Central	TN	POLYGON
40	Texas	41	48	West South Central	TX	(POLYGON
41	New Mexico	42	35	Mountain	NM	POLYGON
42	Alabama	43	01	East South Central	AL	POLYGON
43	Mississippi	44	28	East South Central	MS	POLYGON
44	Georgia	45	13	South Atlantic	GA	(POLYGON
45	South Carolina	46	45	South Atlantic	SC	(POLYGON
46	Arkansas	47	05	West South Central	AR	POLYGON
47	Louisiana	48	22	West South Central	LA	(POLYGON
48	Florida	49	12	South Atlantic	FL	(POLYGON

49	Michigan	50	26	East North Central	MI	(POLYGON
50	Alaska	51	02	Pacific	AK	(POLYGON

Merge State Boundary Data with Cluster Data

In [13]:
```
clusters = []

for index, row in us_map.iterrows():
    state = row['STATE_ABBR']
    cluster = list(data[data.State == state].ranked_cluster)
    if len(cluster) > 1:
        cluster = [-1]

    clusters.append(int(cluster[0]))

us_map['cluster'] = clusters
us_map
```

Out[13]:

	STATE_NAME	DRAWSEQ	STATE_FIPS	SUB_REGION	STATE_ABBR	
0	Hawaii	1	15	Pacific	HI	(POLYGON
1	Washington	2	53	Pacific	WA	(POLYGON
2	Montana	3	30	Mountain	MT	POLYGON
3	Maine	4	23	New England	ME	(POLYGON
4	North Dakota	5	38	West North Central	ND	POLYGON
5	South Dakota	6	46	West North Central	SD	POLYGON
6	Wyoming	7	56	Mountain	WY	POLYGON
7	Wisconsin	8	55	East North Central	WI	(POLYGON
8	Idaho	9	16	Mountain	ID	POLYGON
9	Vermont	10	50	New England	VT	POLYGON
10	Minnesota	11	27	West North Central	MN	POLYGON
11	Oregon	12	41	Pacific	OR	POLYGON
12	New Hampshire	13	33	New England	NH	POLYGON
13	Iowa	14	19	West North Central	IA	POLYGON
14	Massachusetts	15	25	New England	MA	(POLYGON
15	Nebraska	16	31	West North Central	NE	POLYGON
16	New York	17	36	Middle Atlantic	NY	(POLYGON
17	Pennsylvania	18	42	Middle Atlantic	PA	POLYGON
18	Connecticut	19	09	New England	CT	POLYGON
19	Rhode Island	20	44	New England	RI	(POLYGON
20	New Jersey	21	34	Middle Atlantic	NJ	POLYGON
21	Indiana	22	18	East North Central	IN	POLYGON
22	Nevada	23	32	Mountain	NV	POLYGON
23	Utah	24	49	Mountain	UT	POLYGON
24	California	25	06	Pacific	CA	(POLYGON
25	Ohio	26	39	East North Central	OH	POLYGON
26	Illinois	27	17	East North Central	IL	POLYGON
27	District of Columbia	28	11	South Atlantic	DC	POLYGON
28	Delaware	29	10	South Atlantic	DE	POLYGON

29	West Virginia	30	54	South Atlantic	WV	POLYGON
30	Maryland	31	24	South Atlantic	MD	(POLYGON
31	Colorado	32	08	Mountain	CO	POLYGON
32	Kentucky	33	21	East South Central	KY	(POLYGON
33	Kansas	34	20	West North Central	KS	POLYGON
34	Virginia	35	51	South Atlantic	VA	(POLYGON
35	Missouri	36	29	West North Central	MO	POLYGON
36	Arizona	37	04	Mountain	AZ	POLYGON
37	Oklahoma	38	40	West South Central	OK	POLYGON
38	North Carolina	39	37	South Atlantic	NC	(POLYGON
39	Tennessee	40	47	East South Central	TN	POLYGON
40	Texas	41	48	West South Central	TX	(POLYGON
41	New Mexico	42	35	Mountain	NM	POLYGON
42	Alabama	43	01	East South Central	AL	POLYGON
43	Mississippi	44	28	East South Central	MS	POLYGON
44	Georgia	45	13	South Atlantic	GA	(POLYGON
45	South Carolina	46	45	South Atlantic	SC	(POLYGON
46	Arkansas	47	05	West South Central	AR	POLYGON
47	Louisiana	48	22	West South Central	LA	(POLYGON
48	Florida	49	12	South Atlantic	FL	(POLYGON
49	Michigan	50	26	East North Central	MI	(POLYGON
50	Alaska	51	02	Pacific	AK	(POLYGON

Important Mapping Settings

In [14]:
```
fig_size = plt.rcParams["figure.figsize"]
print("Current size:" + str(fig_size))
fig_size[0] = 14
fig_size[1] = 12
plt.rcParams["figure.figsize"] = fig_size
print("Current size:" + str(fig_size))

cmap_politics = matplotlib.colors.LinearSegmentedColormap.from_list("", ["navy","dark
```

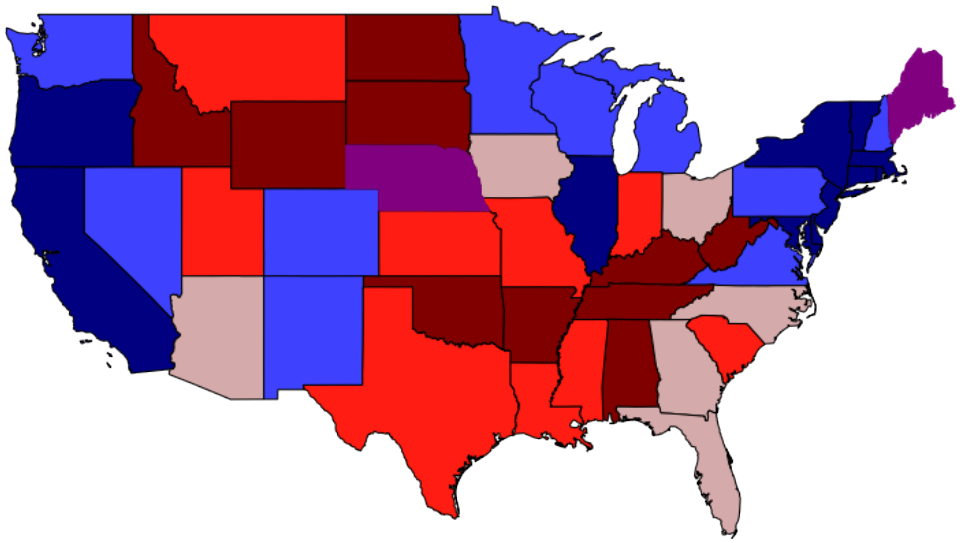
```
Current size:[6.0, 4.0]
Current size:[14, 12]
```

Plot Lower 48 States + DC

In [15]:
```
import geoplot

states = list(range(1,50))
states.remove(3)
states.remove(15)

fig = plt.figure(1)
ax1 = plt.subplot(111)
```

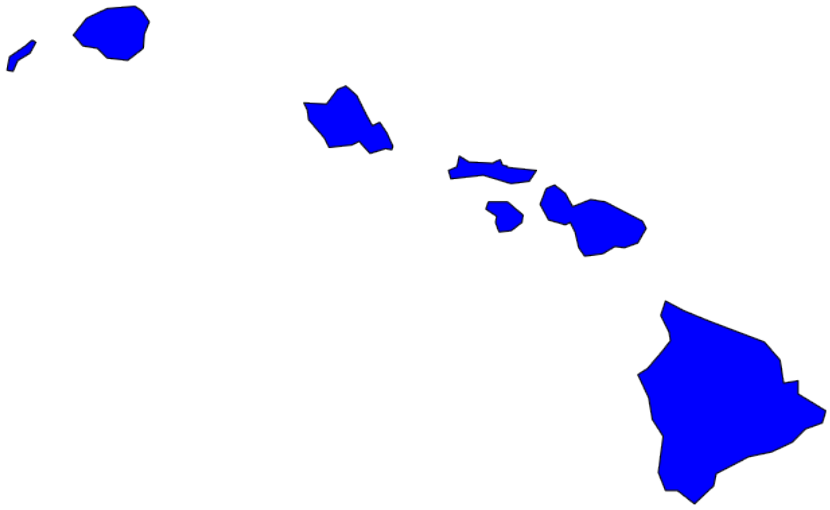
```
        fig.set_figheight(8)
        fig.set_figwidth(14)

        p1 = geoplot.choropleth(us_map.loc[states, :], hue='cluster', cmap=cmap_politics, ax=a
        p2 = geoplot.choropleth(us_map[3:4], hue='cluster', ax=ax1, color= "purple", edgecolo
        p3 = geoplot.choropleth(us_map[15:16], hue='cluster', ax=ax1, color= "purple", edgeco
        plt.xlim(-125,-60)
        plt.ylim(22,50)
        plt.axis('off')
        plt.show()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/pysal/__init__.py:65: VisibleDepre
  ), VisibleDeprecationWarning)
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureW
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/pysal/esda/mapclassify.py:702: Run
  gadf = 1 - self.adcm / adam
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/matplotlib/patches.py:83: UserWarn
  warnings.warn("Setting the 'color' property will override"
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/matplotlib/patches.py:83: UserWarn
  warnings.warn("Setting the 'color' property will override"
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/matplotlib/patches.py:83: UserWarn
  warnings.warn("Setting the 'color' property will override"
```



Hawaii

```
In [16]: us_map.loc[[0], :].plot()
         plt.axis('off')
         plt.show()
```



Alaska

```
In [17]: us_map.loc[[50], :].plot(color = 'red')
         plt.axis('off')
         plt.show()
```

3 County Analysis: Winning Key Votes

We want to target young, diverse, urban communities in these key states to drive high turnout in these communities and capitalize on 2018 gains in these challenging state to flip the state for 2020.

Load Census Data on Age and Race to build County Database

In [18]:
```
age = pd.read_csv('demographics/age.csv')
age['County'] = age.Geography.apply(lambda x: x.split(' County')[0])
age['State'] = age.Geography.apply(lambda x: x.split('County, ')[1])
print(len(age))
age.head()
```

```
341
```

Out[18]:

	Id	Id2	Geography	Estimate; Total:	Margin of Error; T
0	0500000US04001	4001	Apache County, Arizona	71602	
1	0500000US04003	4003	Cochise County, Arizona	126516	
2	0500000US04005	4005	Coconino County, Arizona	138639	
3	0500000US04007	4007	Gila County, Arizona	53145	
4	0500000US04009	4009	Graham County, Arizona	37700	

	Margin of Error; Male: - 40 to 44 years	Estimate; Male: - 45 to 49 years	Margin o
0	189	2000	
1	320	3549	
2	343	3547	
3	168	1395	
4	173	1276	

	Estimate; Female: - 5 to 9 years	Margin of Error; Female: - 5 to 9 years	Estimat
0	2722	190	
1	3818	318	
2	4232	325	
3	1457	153	
4	1433	198	

	Estimate; Female: - 55 to 59 years	Margin of Error; Female: - 55 to 59 years	Esti
0	2351	157	
1	4176	374	
2	4559	250	
3	2203	292	
4	812	112	

In [19]:
```
race = pd.read_csv('demographics/race.csv')
age['County'] = age.Geography.apply(lambda x: x.split(' County')[0])
```

```
age['State'] = age.Geography.apply(lambda x: x.split('County, ')[1])
race.head()
```

Out[19]:

	Id	Id2	Geography	Estimate; Total:	Margin of Error; T
0	0500000US04001	4001	Apache County, Arizona	71602	
1	0500000US04003	4003	Cochise County, Arizona	126516	
2	0500000US04005	4005	Coconino County, Arizona	138639	
3	0500000US04007	4007	Gila County, Arizona	53145	
4	0500000US04009	4009	Graham County, Arizona	37700	

	Estimate; Total: - Two or more races: - Two races excluding Some other race, and th
0	1437
1	4753
2	4361
3	1382
4	743

```
In [20]: counties = pd.read_csv('county_data/county_fips.csv')
         counties.head()
```

Out[20]:

	msa	pmsa	county	state	county_fips	state_fips
0	NaN	NaN	Aleutians East	AK	13	2
1	NaN	NaN	Aleutians West	AK	16	2
2	380.0	NaN	Anchorage	AK	20	2
3	NaN	NaN	Bethel	AK	50	2
4	NaN	NaN	Bristol Bay	AK	60	2

Load County Voting Dataset

```
In [21]: voting = pd.read_csv('county_pres.csv')
         voting.head()
```

Out[21]:

	year	state	state_po	county	FIPS	office	candidate	party	can
0	2000	Alabama	AL	Autauga	1001.0	President	Al Gore	democrat	
1	2000	Alabama	AL	Autauga	1001.0	President	George W. Bush	republican	
2	2000	Alabama	AL	Autauga	1001.0	President	Ralph Nader	green	
3	2000	Alabama	o	Autauga	1001.0	President	Other	NaN	
4	2000	Alabama	AL	Baldwin	1003.0	President	Al Gore	democrat	

```
In [22]: states = ['Arizona', 'Georgia', 'Florida', 'North Carolina']
         voting = voting[voting['state'].isin(states)]
         counties = set(list(voting.county))

         county_list = []
         d_2016 = []
         r_2016 = []
         d_2012 = []
         r_2012 = []
         geography = []
```

```
        for state in states:
            state_data = voting[voting['state'] == state]
            counties = set(list(state_data.county))
            for county in counties:
                county_list.append(county)
                county_data = voting[voting.county == county]
                data_2016 = county_data[county_data.year == 2016]
                data_2012 = county_data[county_data.year == 2012]

                dem_2016 = data_2016[data_2016.party == 'democrat']
                dem_2012 = data_2012[data_2012.party == 'democrat']
                rep_2016 = data_2016[data_2016.party == 'republican']
                rep_2012 = data_2012[data_2012.party == 'republican']

                d_2016.append(list(dem_2016.candidatevotes)[0]*1.0/list(dem_2016.totalvotes)[
                d_2012.append(list(dem_2012.candidatevotes)[0]*1.0/list(dem_2012.totalvotes)[
                r_2016.append(list(rep_2016.candidatevotes)[0]*1.0/list(rep_2016.totalvotes)[
                r_2012.append(list(rep_2012.candidatevotes)[0]*1.0/list(rep_2012.totalvotes)[
                geography.append(county + " County, " + str(state))

        county_voting = pd.DataFrame({'County': list(county_list), 'Geography': geography, '2
        county_voting['2016_r'].apply(lambda x: x*100.0)
        county_voting['2012_r'].apply(lambda x: x*100.0)
        county_voting['2016_d'].apply(lambda x: x*100.0)
        county_voting['2012_d'].apply(lambda x: x*100.0)
        #county_voting['Geography'] = county_voting['County'].apply(lambda x: x + " County")
        county_voting.to_csv("County_voting.csv")
        county_voting.head()
```

Out[22]:

	County	Geography	2016_r	2012_r	2016_d	2012_d
0	Greenlee	Greenlee County, Arizona	0.364407	0.535306	0.210324	0.440484
1	Cochise	Cochise County, Arizona	0.561671	0.601943	0.348895	0.378467
2	Gila	Gila County, Arizona	0.386462	0.625000	0.190833	0.357534
3	Graham	Graham County, Arizona	0.395203	0.681232	0.162563	0.304429
4	Apache	Apache County, Arizona	0.297892	0.319174	0.617584	0.663378

Merge Datasets

In [23]:
```
demographic_data = pd.merge(age, race, on='Geography')
print(len(demographic_data))
demographic_data.head()
```

```
341
```

Out[23]:

	Id_x	Id2_x	Geography	Estimate; Total:_x	Margin of Erro
0	0500000US04001	4001	Apache County, Arizona	71602	
1	0500000US04003	4003	Cochise County, Arizona	126516	

```
2  0500000US04005   4005  Coconino County, Arizona              138639
3  0500000US04007   4007      Gila County, Arizona               53145
4  0500000US04009   4009    Graham County, Arizona               37700

   Margin of Error; Male: - 40 to 44 years  Estimate; Male: - 45 to 49 years Margin o
0                                      189                              2000
1                                      320                              3549
2                                      343                              3547
3                                      168                              1395
4                                      173                              1276

   Estimate; Female: - 5 to 9 years  Margin of Error; Female: - 5 to 9 years  Estimate
0                              2722                                     190
1                              3818                                     318
2                              4232                                     325
3                              1457                                     153
4                              1433                                     198

   Estimate; Female: - 55 to 59 years  Margin of Error; Female: - 55 to 59 years  Esti
0                                2351                                       157
1                                4176                                       374
2                                4559                                       250
3                                2203                                       292
4                                 812                                       112

   Margin of Error; Total: - Black or African American alone  Estimate; Total: - Ameri
0                                                134
1                                                413
2                                                241
3                                                112
4                                                 95
```

```
In [24]: demographic_data = pd.merge(demographic_data, county_voting, on='Geography')
```

Analyze list of available features for these counties and Make new features that are representative of our target audience

Selected features: 'Geography', 'County', 'State', 'percent_male_0_17', 'percent_male_18_24' ,'percent_male_25_34', 'percent_male_35_44', 'percent_male_45_59', 'percent_male_60_79', 'percent_male_80_up', 'percent_female_0_17', 'percent_female_18_24', 'percent_female_25_34', 'percent_female_35_44', 'percent_female_45_59', 'percent_female_60_79', 'percent_female_80_up', 'percent_white', 'percent_black', 'percent_native', 'percent_asian', 'percent_hawaiian', 'percent_other', 'percent_two', 'percent_two_other', 'percent_three'

```
In [25]: #for col in demographic_data.columns:
         #    print(col)

         clustering_cols = ['Geography', 'County_x', 'State', 'percent_male_0_17','percent_male
```

```
demographic_data['percent_male_0_17'] = demographic_data[['Estimate; Male: - Under 5 y
demographic_data['percent_male_18_24'] = demographic_data[['Estimate; Male: - 18 and
demographic_data['percent_male_25_34'] = demographic_data[['Estimate; Male: - 25 to 29
demographic_data['percent_male_35_44'] = demographic_data[['Estimate; Male: - 35 to 39
demographic_data['percent_male_45_59'] = demographic_data[['Estimate; Male: - 45 to 49
demographic_data['percent_male_60_79'] = demographic_data[['Estimate; Male: - 60 and
demographic_data['percent_male_80_up'] = demographic_data[['Estimate; Male: - 80 to 84
demographic_data['percent_female_0_17'] = demographic_data[['Estimate; Female: - Under
demographic_data['percent_female_18_24'] = demographic_data[['Estimate; Female: - 18 a
demographic_data['percent_female_25_34'] = demographic_data[['Estimate; Female: - 25 t
demographic_data['percent_female_35_44'] = demographic_data[['Estimate; Female: - 35 t
demographic_data['percent_female_45_59'] = demographic_data[['Estimate; Female: - 45 t
demographic_data['percent_female_60_79'] = demographic_data[['Estimate; Female: - 60 a
demographic_data['percent_female_80_up'] = demographic_data[['Estimate; Female: - 80 t
demographic_data['percent_white'] = demographic_data['Estimate; Total: - White alone']
demographic_data['percent_black'] = demographic_data['Estimate; Total: - Black or Afri
demographic_data['percent_native'] = demographic_data['Estimate; Total: - American Ind
demographic_data['percent_asian'] = demographic_data['Estimate; Total: - Asian alone']
demographic_data['percent_hawaiian'] = demographic_data['Estimate; Total: - Native Hau
demographic_data['percent_other'] = demographic_data['Estimate; Total: - Some other ra
demographic_data['percent_two'] = demographic_data['Estimate; Total: - Two or more rac
demographic_data['percent_two_other'] = demographic_data['Estimate; Total: - Two or mo
demographic_data['percent_three'] = demographic_data['Estimate; Total: - Two or more r
demographic_data['percent_male'] = demographic_data['Estimate; Male:'].div(demographic
demographic_data['percent_female'] = demographic_data['Estimate; Female:'].div(demogra

demographic_data[clustering_cols].head()
```

Out[25]:

	Geography	County_x	State	percent_male_0_17	percent_male_18_24
0	Apache County, Arizona	Apache	Arizona	0.139298	0.052526
1	Cochise County, Arizona	Cochise	Arizona	0.110381	0.047630
2	Coconino County, Arizona	Coconino	Arizona	0.109082	0.093949
3	Gila County, Arizona	Gila	Arizona	0.104318	0.035714
4	Graham County, Arizona	Graham	Arizona	0.139178	0.059761

Split by State

In [26]:
```
az_data = demographic_data[demographic_data.State == 'Arizona']
nc_data = demographic_data[demographic_data.State == 'North Carolina']
ga_data = demographic_data[demographic_data.State == 'Georgia']
fl_data = demographic_data[demographic_data.State == 'Florida']
print("Number of counties in GA: " + str(len(ga_data)))
print("Number of counties in FL: " + str(len(fl_data)))
print("Number of counties in AZ: " + str(len(az_data)))
print("Number of counties in NC: " + str(len(nc_data)))
```

```
Number of counties in GA: 159
Number of counties in FL: 66
```

```
Number of counties in AZ: 15
Number of counties in NC: 100
```

3.0.1 Arizona County Analysis

```
In [27]: az_data.head()
```

Out[27]:

	Id_x	Id2_x	Geography	Estimate; Total:_x	Margin of Erro
0	0500000US04001	4001	Apache County, Arizona	71602	
1	0500000US04003	4003	Cochise County, Arizona	126516	
2	0500000US04005	4005	Coconino County, Arizona	138639	
3	0500000US04007	4007	Gila County, Arizona	53145	
4	0500000US04009	4009	Graham County, Arizona	37700	

	Margin of Error; Male: - 40 to 44 years	Estimate; Male: - 45 to 49 years	Margin o
0	189	2000	
1	320	3549	
2	343	3547	
3	168	1395	
4	173	1276	

	Estimate; Female: - 5 to 9 years	Margin of Error; Female: - 5 to 9 years	Estimat
0	2722	190	
1	3818	318	
2	4232	325	
3	1457	153	
4	1433	198	

	Estimate; Female: - 55 to 59 years	Margin of Error; Female: - 55 to 59 years	Esti
0	2351	157	
1	4176	374	
2	4559	250	
3	2203	292	
4	812	112	

	Margin of Error; Total: - Black or African American alone	Estimate; Total: - Amer
0	134	
1	413	
2	241	
3	112	
4	95	

	2012_d	percent_male_0_17	percent_male_18_24	percent_male_25_34	percent_male_3
0	0.663378	0.139298	0.052526	0.063727	0.05
1	0.378467	0.110381	0.047630	0.071564	0.05
2	0.565560	0.109082	0.093949	0.070211	0.05
3	0.357534	0.104318	0.035714	0.048979	0.04

```
4  0.304429          0.139178          0.059761          0.089284          0.0
```

In [28]:
```
#clustering_cols_analysis = ['2012_d', '2012_r', '2016_d', '2016_r','percent_male_0_1
clustering_cols_analysis = ['2012_d', '2012_r', '2016_d', '2016_r','percent_male_18_24
kmeans = KMeans(n_clusters=3, random_state=0).fit(az_data[clustering_cols_analysis])

labels = kmeans.labels_
az_data['az_clusters'] = labels
#az_data.head()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWi
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

In [29]:
```
cluster_data = []

print("Expecting Cococino, Navajo, Apache, Maricopa, Pima, Santa Cruz")
print()
print()

for label in list(set(list(az_data['az_clusters']))):
    data_clust = az_data[az_data['az_clusters'] == label]
    individual = {}
    print("Cluster Number: " + str(label))
    print("Counties: " + str(list(data_clust['County_x'])))
    print("Percent 18-24: " + str(np.average(list(data_clust['percent_male_18_24'])+ n
    print("Percent White: " + str(np.average(list(data_clust['percent_white']))*100.0)
    print("Percent Black: " + str(np.average(list(data_clust['percent_black']))*100.0)
    print("Percent Native American: " + str(np.average(list(data_clust['percent_native
    print("Percent Other: " + str(np.average(list(data_clust['percent_other']))*100.0)
    print("Percent Male: " + str(np.average(list(data_clust['percent_male']))*100.0) +
    print("Percent Female: " + str(np.average(list(data_clust['percent_female']))*100
    print("Percent Dem: " + str(np.average(list(data_clust['2016_d']))*100.0) + "%")
    print()
    print('------------------')
    print()
```

```
Expecting Cococino, Navajo, Apache, Maricopa, Pima, Santa Cruz


Cluster Number: 0
Counties: ['Cochise', 'Gila', 'Graham', 'Greenlee', 'La Paz', 'Mohave', 'Pinal', 'Yavapai']
Percent 18-24: 8.076219342642531%
Percent White: 83.77142934028477%
Percent Black: 1.9192058677049266%
```

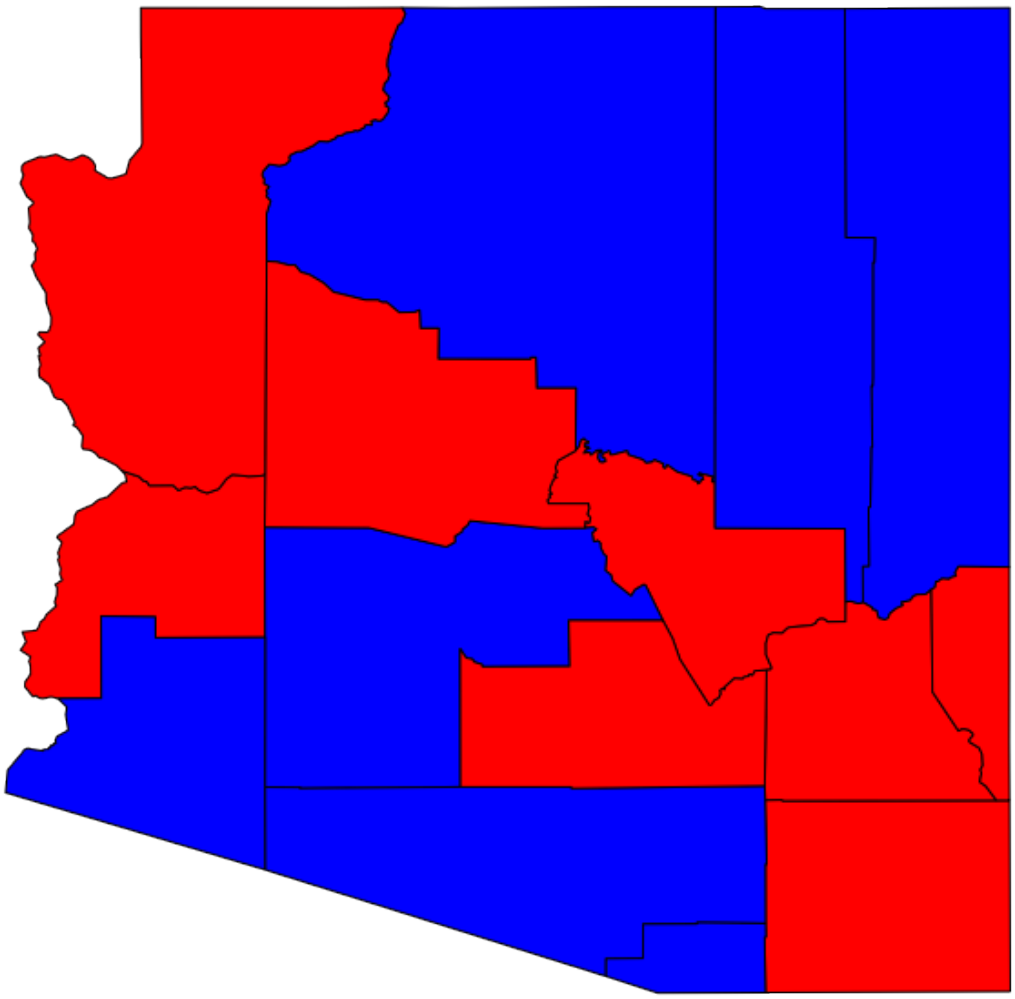
```
Percent Native American: 7.016827393991859%
Percent Other: 3.3682159500233304%
Percent Male: 51.2587880487463%
Percent Female: 48.7412119512537%
Percent Dem: 25.953447081957837%

------------------

Cluster Number: 1
Counties: ['Apache', 'Coconino', 'Navajo']
Percent 18-24: 13.27111416362613%
Percent White: 44.74338973261921%
Percent Black: 0.9215862648670751%
Percent Native American: 48.0126883673408%
Percent Other: 2.262869136228706%
Percent Male: 49.56516151279517%
Percent Female: 50.434838487204836%
Percent Dem: 52.26916603956886%

------------------

Cluster Number: 2
Counties: ['Maricopa', 'Pima', 'Santa Cruz', 'Yuma']
Percent 18-24: 10.738596695770477%
Percent White: 78.54421001048408%
Percent Black: 2.8527495520351063%
Percent Native American: 1.7751143151573008%
Percent Other: 11.614904330566288%
Percent Male: 49.49658973016402%
Percent Female: 50.50341026983598%
Percent Dem: 53.917193321978495%

------------------
```

```
In [30]: binary = []
         for index, row in az_data.iterrows():
             if row.az_clusters == 0:
                 binary.append(2)
             else:
                 binary.append(1)

         az_data['binary_clusters'] = binary
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingW
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [31]: county = gpd.read_file('azcounty/azcounties.shp')
         county.rename(columns={'NAME':'County_x'},inplace=True)
         county = pd.merge(az_data[['County_x', 'binary_clusters']], county, on='County_x')
         cmap_politics_2 = matplotlib.colors.LinearSegmentedColormap.from_list("", ["blue","red
         county = gpd.GeoDataFrame(county)
         county.plot(column="binary_clusters", cmap=cmap_politics_2)
         plt.axis('off')
         plt.show()
```



3.0.2 Florida County Analysis

```
In [32]: clustering_cols_analysis = ['2012_d', '2012_r', '2016_d', '2016_r','percent_male_18_24
         kmeans = KMeans(n_clusters=3, random_state=0).fit(fl_data[clustering_cols_analysis])

         labels = kmeans.labels_
         fl_data['fl_clusters'] = labels
         #az_data.head()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWi
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """
```

```
In [33]: cluster_data = []
         for label in list(set(list(fl_data['fl_clusters']))):
             data_clust = fl_data[fl_data['fl_clusters'] == label]
             individual = {}
             print("Cluster Number: " + str(label))
             print("Counties: " + str(list(data_clust['County_x'])))
             print("Percent 18-24: " + str(np.average(list(data_clust['percent_male_18_24'])+ n
             print("Percent White: " + str(np.average(list(data_clust['percent_white']))*100.0)
             print("Percent Black: " + str(np.average(list(data_clust['percent_black']))*100.0)
             print("Percent Native American: " + str(np.average(list(data_clust['percent_native
             print("Percent Other: " + str(np.average(list(data_clust['percent_other']))*100.0)
             print("Percent Male: " + str(np.average(list(data_clust['percent_male']))*100.0) +
             print("Percent Female: " + str(np.average(list(data_clust['percent_female']))*100
             print("Percent Dem: " + str(np.average(list(data_clust['2016_d']))*100.0) + "%")
             print()
             print('------------------')
             print()
```

```
Cluster Number: 0
Counties: ['Baker', 'Bay', 'Bradford', 'Calhoun', 'Clay', 'Columbia', 'Dixie', 'Franklin', 'Gil
Percent 18-24: 8.254400265935418%
Percent White: 82.19533669628824%
Percent Black: 12.674482342540038%
Percent Native American: 0.48237761753699887%
Percent Other: 1.3346839003247681%
Percent Male: 53.829176761967034%
Percent Female: 46.170823238032966%
Percent Dem: 22.841300984641023%

------------------

Cluster Number: 1
Counties: ['Brevard', 'Charlotte', 'Citrus', 'Collier', 'Escambia', 'Flagler', 'Glades', 'Hamil
Percent 18-24: 7.644140415005304%
Percent White: 83.01945966908968%
Percent Black: 10.667609197060838%
Percent Native American: 0.5535211411889396%
Percent Other: 1.8657831497436514%
Percent Male: 49.778534002622585%
Percent Female: 50.22146599737742%
```

```
Percent Dem: 37.27347267867142%

------------------

Cluster Number: 2
Counties: ['Alachua', 'Broward', 'Duval', 'Gadsden', 'Hillsborough', 'Jefferson', 'Leon', 'Madi
Percent 18-24: 10.779946009433996%
Percent White: 65.1442296519164%
Percent Black: 26.490241984654073%
Percent Native American: 0.26724388438947577%
Percent Other: 2.7944530976374997%
Percent Male: 49.238901513275145%
Percent Female: 50.761098486724855%
Percent Dem: 56.14042315179603%

------------------
```

```
In [34]: binary = []
         for index, row in fl_data.iterrows():
             if row.fl_clusters == 2:
                 binary.append(1)
             else:
                 binary.append(2)

         fl_data['binary_clusters'] = binary
```

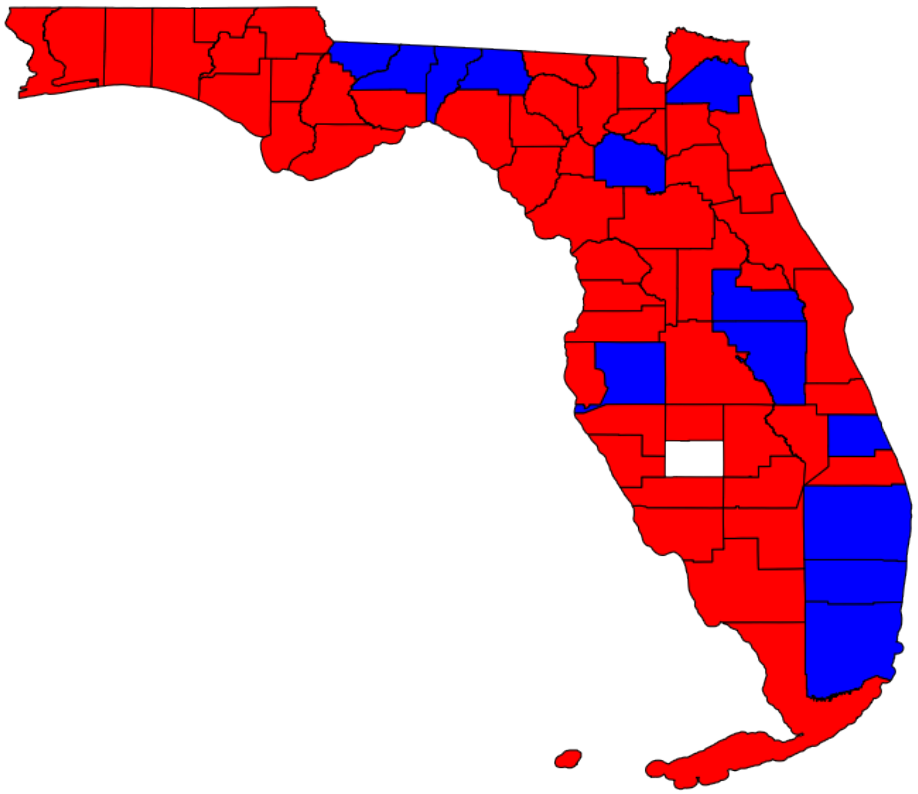
```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [35]: def strip(string):
             punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
             no_punct = ""
             for char in string:
                 if char not in punctuations:
                     no_punct = no_punct + char
             return no_punct
```

```
In [36]: county = gpd.read_file('scuov/scuo.shp')
         county = county[county.STATE == 'FL']
         #print(len(county[county.STATE == 'FL']))
         #print(len(fl_data))
         county.head()
```

```
county.rename(columns={'NAME':'County_x'},inplace=True)
county['county_lower'] = county['County_x'].str.lower().apply(lambda x: strip(x))
fl_data['county_lower'] = fl_data['County_x'].str.lower().apply(lambda x: strip(x))
county = pd.merge(fl_data[['county_lower', 'binary_clusters']], county, on='county_lo
#cmap_politics_2 = matplotlib.colors.LinearSegmentedColormap.from_list("", ["blue","r
county = gpd.GeoDataFrame(county)
county.plot(column="binary_clusters", cmap=cmap_politics_2)
plt.axis('off')
plt.show()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingW
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```



3.0.3 Georgia County Analysis

```
In [37]: clustering_cols_analysis = ['2012_d', '2012_r', '2016_d', '2016_r','percent_male_18_24
         kmeans = KMeans(n_clusters=3, random_state=0).fit(ga_data[clustering_cols_analysis])

         labels = kmeans.labels_
         ga_data['ga_clusters'] = labels
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWi
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """
```

```
In [38]: for label in list(set(list(ga_data['ga_clusters']))):
             data_clust = ga_data[ga_data['ga_clusters'] == label]
             individual = {}
             print("Cluster Number: " + str(label))
             print("Counties: " + str(list(data_clust['County_x'])))
             print("Percent 18-24: " + str(np.average(list(data_clust['percent_male_18_24'])+ n
             print("Percent White: " + str(np.average(list(data_clust['percent_white']))*100.0)
             print("Percent Black: " + str(np.average(list(data_clust['percent_black']))*100.0)
             print("Percent Native American: " + str(np.average(list(data_clust['percent_native
             print("Percent Other: " + str(np.average(list(data_clust['percent_other']))*100.0)
             print("Percent Male: " + str(np.average(list(data_clust['percent_male']))*100.0) +
             print("Percent Female: " + str(np.average(list(data_clust['percent_female']))*100
             print("Percent Dem: " + str(np.average(list(data_clust['2016_d']))*100.0) + "%")
             print()
             print('------------------')
             print()
```

```
Cluster Number: 0
Counties: ['Baldwin', 'Bibb', 'Burke', 'Chatham', 'Clarke', 'Clayton', 'DeKalb', 'Dooly', 'Doug
Percent 18-24: 10.31077530305337%
Percent White: 42.48359794672628%
Percent Black: 51.7109685099099%
Percent Native American: 0.15900394987587643%
Percent Other: 2.0658484417585576%
Percent Male: 49.4800963534584%
Percent Female: 50.5199036465416%
Percent Dem: 57.17616733282734%

------------------

Cluster Number: 1
Counties: ['Appling', 'Bacon', 'Banks', 'Barrow', 'Bartow', 'Berrien', 'Brantley', 'Bryan', 'Ca
Percent 18-24: 8.502606707818076%
```

```
Percent White: 86.21194137230563%
Percent Black: 8.414656938105042%
Percent Native American: 0.313710159801181%
Percent Other: 1.9246788406649131%
Percent Male: 49.22737838783696%
Percent Female: 50.77262161216305%
Percent Dem: 19.321639811931874%

------------------

Cluster Number: 2
Counties: ['Atkinson', 'Baker', 'Ben Hill', 'Bleckley', 'Brooks', 'Bulloch', 'Butts', 'Calhoun
Percent 18-24: 9.724561202507553%
Percent White: 63.10235513398429%
Percent Black: 31.611743776961287%
Percent Native American: 0.24526766549575946%
Percent Other: 2.2658655179404383%
Percent Male: 49.91925428188878%
Percent Female: 50.08074571811122%
Percent Dem: 32.28358612847163%

------------------
```

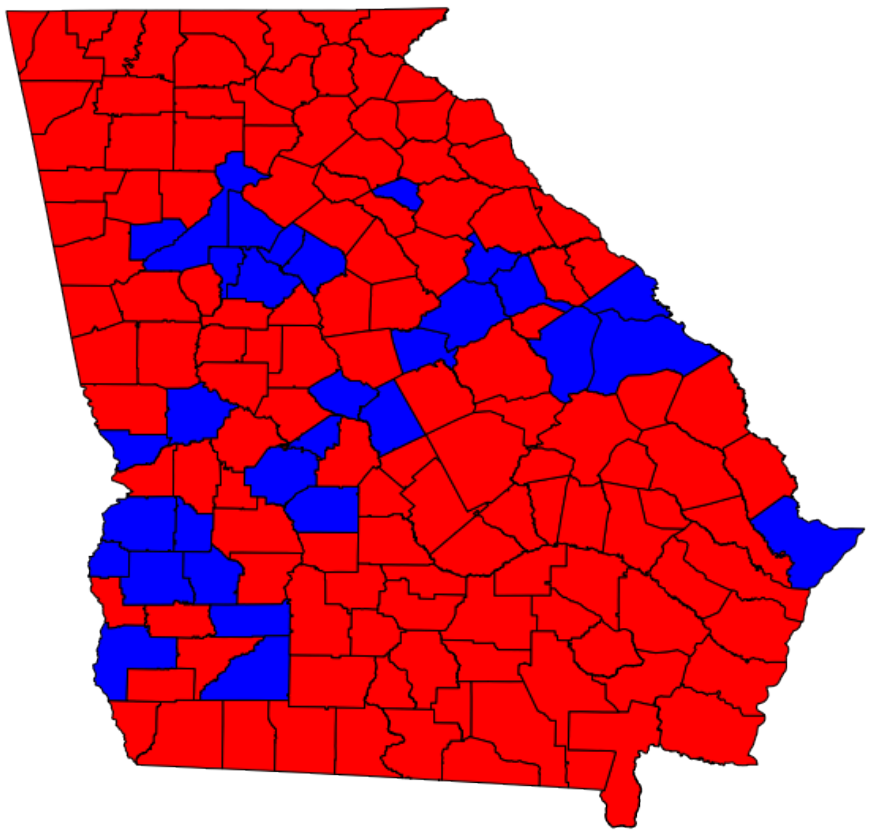
```
In [39]: binary = []
         for index, row in ga_data.iterrows():
             if row.ga_clusters == 0:
                 binary.append(1)
             else:
                 binary.append(2)

         ga_data['binary_clusters'] = binary
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingW
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [40]: county = gpd.read_file('scuov/scuo.shp')
         county = county[county.STATE == 'GA']
         #print(len(county[county.STATE == 'FL']))
         #print(len(fl_data))
         county.head()
         county.rename(columns={'NAME':'County_x'},inplace=True)
```

```
county['county_lower'] = county['County_x'].str.lower().apply(lambda x: strip(x))
ga_data['county_lower'] = ga_data['County_x'].str.lower().apply(lambda x: strip(x))
county = pd.merge(ga_data[['county_lower', 'binary_clusters']], county, on='county_lo
#cmap_politics_2 = matplotlib.colors.LinearSegmentedColormap.from_list("", ["blue","r
county = gpd.GeoDataFrame(county)
county.plot(column="binary_clusters", cmap=cmap_politics_2)
plt.axis('off')
plt.show()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingW
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```



3.0.4 North Carolina

```
In [41]: clustering_cols_analysis = ['2012_d', '2012_r', '2016_d', '2016_r','percent_male_18_24
         kmeans = KMeans(n_clusters=3, random_state=0).fit(nc_data[clustering_cols_analysis])

         labels = kmeans.labels_
         nc_data['nc_clusters'] = labels
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWi
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """
```

```
In [42]: for label in list(set(list(nc_data['nc_clusters']))):
             data_clust = nc_data[nc_data['nc_clusters'] == label]
             individual = {}
             print("Cluster Number: " + str(label))
             print("Counties: " + str(list(data_clust['County_x'])))
             print("Percent 18-24: " + str(np.average(list(data_clust['percent_male_18_24'])+ n
             print("Percent White: " + str(np.average(list(data_clust['percent_white']))*100.0)
             print("Percent Black: " + str(np.average(list(data_clust['percent_black']))*100.0)
             print("Percent Native American: " + str(np.average(list(data_clust['percent_native
             print("Percent Other: " + str(np.average(list(data_clust['percent_other']))*100.0)
             print("Percent Male: " + str(np.average(list(data_clust['percent_male']))*100.0) +
             print("Percent Female: " + str(np.average(list(data_clust['percent_female']))*100
             print("Percent Dem: " + str(np.average(list(data_clust['2016_d']))*100.0) + "%")
             print()
             print('------------------')
             print()
```

```
Cluster Number: 0
Counties: ['Alexander', 'Alleghany', 'Ashe', 'Avery', 'Brunswick', 'Caldwell', 'Camden', 'Carte
Percent 18-24: 8.304250966050086%
Percent White: 86.4926336957563%
Percent Black: 7.846066385145869%
Percent Native American: 0.8393503494104979%
Percent Other: 2.0434490689251716%
Percent Male: 49.30048360411178%
Percent Female: 50.699516395888224%
Percent Dem: 28.321563270359295%

------------------
```

```
Cluster Number: 1
Counties: ['Alamance', 'Beaufort', 'Bladen', 'Buncombe', 'Burke', 'Cabarrus', 'Caswell', 'Chatl
Percent 18-24: 9.117285768681912%
Percent White: 69.53801025892716%
Percent Black: 22.934815931335358%
Percent Native American: 1.4442829372636043%
Percent Other: 2.602371252792902%
Percent Male: 49.20365919312344%
Percent Female: 50.79634080687656%
Percent Dem: 41.964321191485595%

------------------

Cluster Number: 2
Counties: ['Anson', 'Bertie', 'Cumberland', 'Durham', 'Edgecombe', 'Guilford', 'Halifax', 'Hert
Percent 18-24: 10.086594294190613%
Percent White: 46.856345672424325%
Percent Black: 42.60668791504097%
Percent Native American: 3.8942458073961475%
Percent Other: 2.490314611557077%
Percent Male: 48.65237326893738%
Percent Female: 51.34762673106261%
Percent Dem: 58.292207444924756%

------------------
```

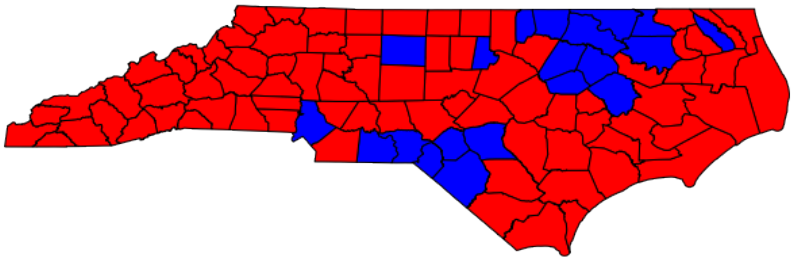
In [43]:
```
binary = []
for index, row in nc_data.iterrows():
    if row.nc_clusters == 2:
        binary.append(1)
    else:
        binary.append(2)

nc_data['binary_clusters'] = binary
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingWi
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

In [44]:
```
county = gpd.read_file('scuov/scuo.shp')
county = county[county.STATE == 'NC']
#print(len(county[county.STATE == 'FL']))
```

```
        #print(len(fl_data))
        county.head()
        county.rename(columns={'NAME':'County_x'},inplace=True)
        county['county_lower'] = county['County_x'].str.lower().apply(lambda x: strip(x))
        nc_data['county_lower'] = nc_data['County_x'].str.lower().apply(lambda x: strip(x))
        county = pd.merge(nc_data[['county_lower', 'binary_clusters']], county, on='county_lo
        #cmap_politics_2 = matplotlib.colors.LinearSegmentedColormap.from_list("", ["blue","r
        county = gpd.GeoDataFrame(county)
        county.plot(column="binary_clusters", cmap=cmap_politics_2)
        plt.axis('off')
        plt.show()
```

```
/Users/joshkuppersmith/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingW
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: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```



In [ ]: