



STATISTICAL THINKING IN PYTHON I

Introduction to Exploratory Data Analysis



Exploratory data analysis

- The process of organizing, plotting, and summarizing a data set



“Exploratory data analysis can never be the whole story, but nothing else can serve as the foundation stone.”

—John Tukey



2008 US swing state election results

2008_swing_states.csv

```
1 state,county,total_votes,dem_votes,rep_votes,dem_share
2 PA,Erie County,127691,75775,50351,60.08
3 PA,Bradford County,25787,10306,15057,40.64
4 PA,Tioga County,17984,6390,11326,36.07
5 PA,McKean County,15947,6465,9224,41.21
6 PA,Potter County,7507,2300,5109,31.04
7 PA,Wayne County,22835,9892,12702,43.78
8 PA,Susquehanna County,19286,8381,10633,44.08
9 PA,Warren County,18517,8537,9685,46.85
10 OH,Ashtabula County,44874,25027,18949,56.94
11 OH,Lake County,121335,60155,59142,50.46
12 PA,Crawford County,38134,16780,20750,44.71
13 OH,Lucas County,219830,142852,73706,65.99
14 OH,Fulton County,21973,9900,11689,45.88
15 OH,Geauga County,51102,21250,29096,42.23
16 OH,Williams County,18397,8174,9880,45.26
17 PA,Wyoming County,13138,5985,6983,46.15
18 PA,Lackawanna County,107876,67520,39488,63.10
19 PA,Elk County,14271,7290,6676,52.20
20 PA,Forest County,2444,1038,1366,43.18
21 PA,Venango County,23307,9238,13718,40.24
22 OH,Erie County,41229,23148,17432,57.01
```



2008 US swing state election results

```
In [1]: import pandas as pd
```

```
In [2]: df_swing = pd.read_csv('2008_swing_states.csv')
```

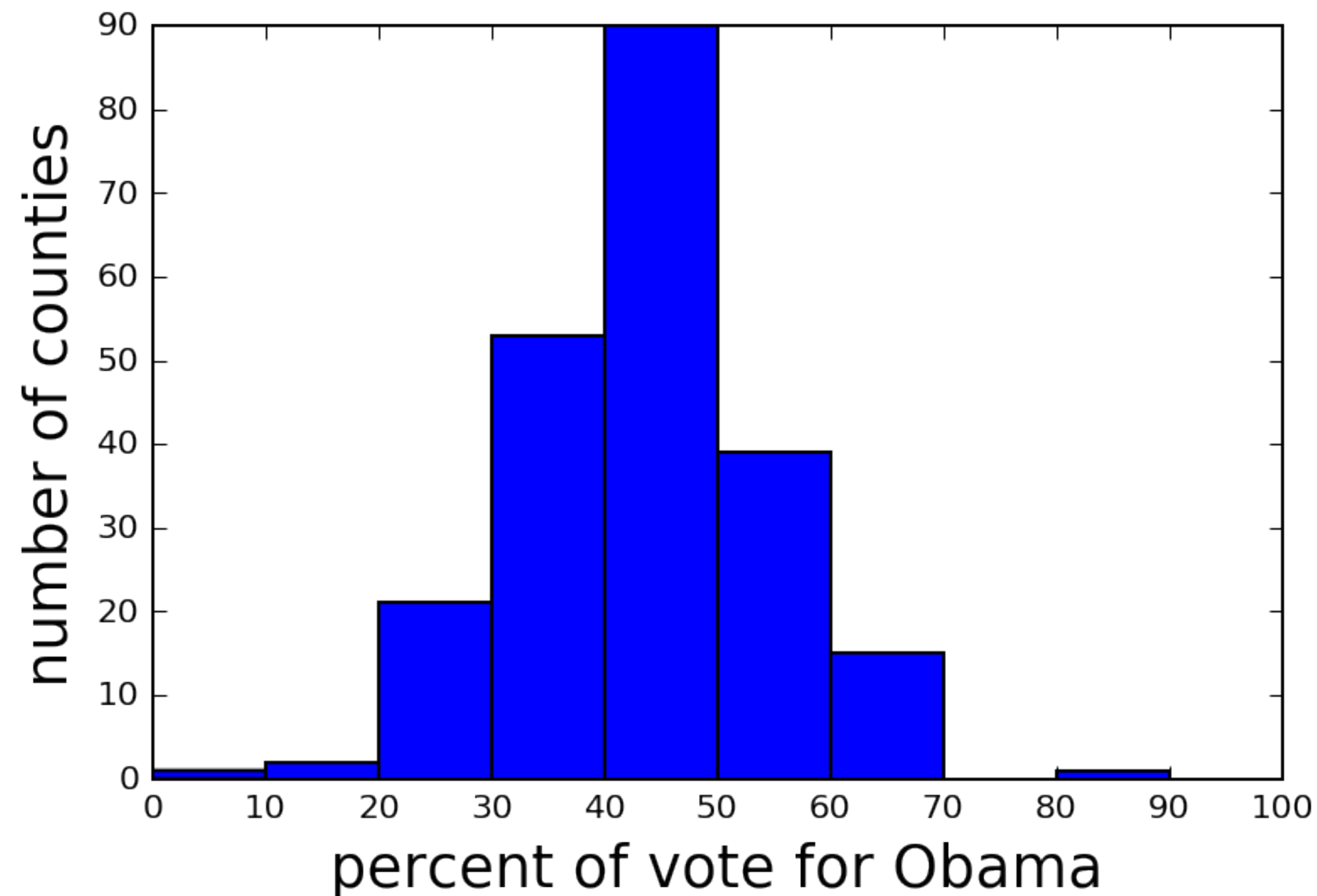
```
In [3]: df_swing[['state', 'county', 'dem_share']]
```

```
Out[3]:
```

	state	county	dem_share
0	PA	Erie County	60.08
1	PA	Bradford County	40.64
2	PA	Tioga County	36.07
3	PA	McKean County	41.21
4	PA	Potter County	31.04
5	PA	Wayne County	43.78
6	PA	Susquehanna County	44.08
7	PA	Warren County	46.85
8	OH	Ashtabula County	56.94



2008 US swing state election results





STATISTICAL THINKING IN PYTHON I

Let's practice!

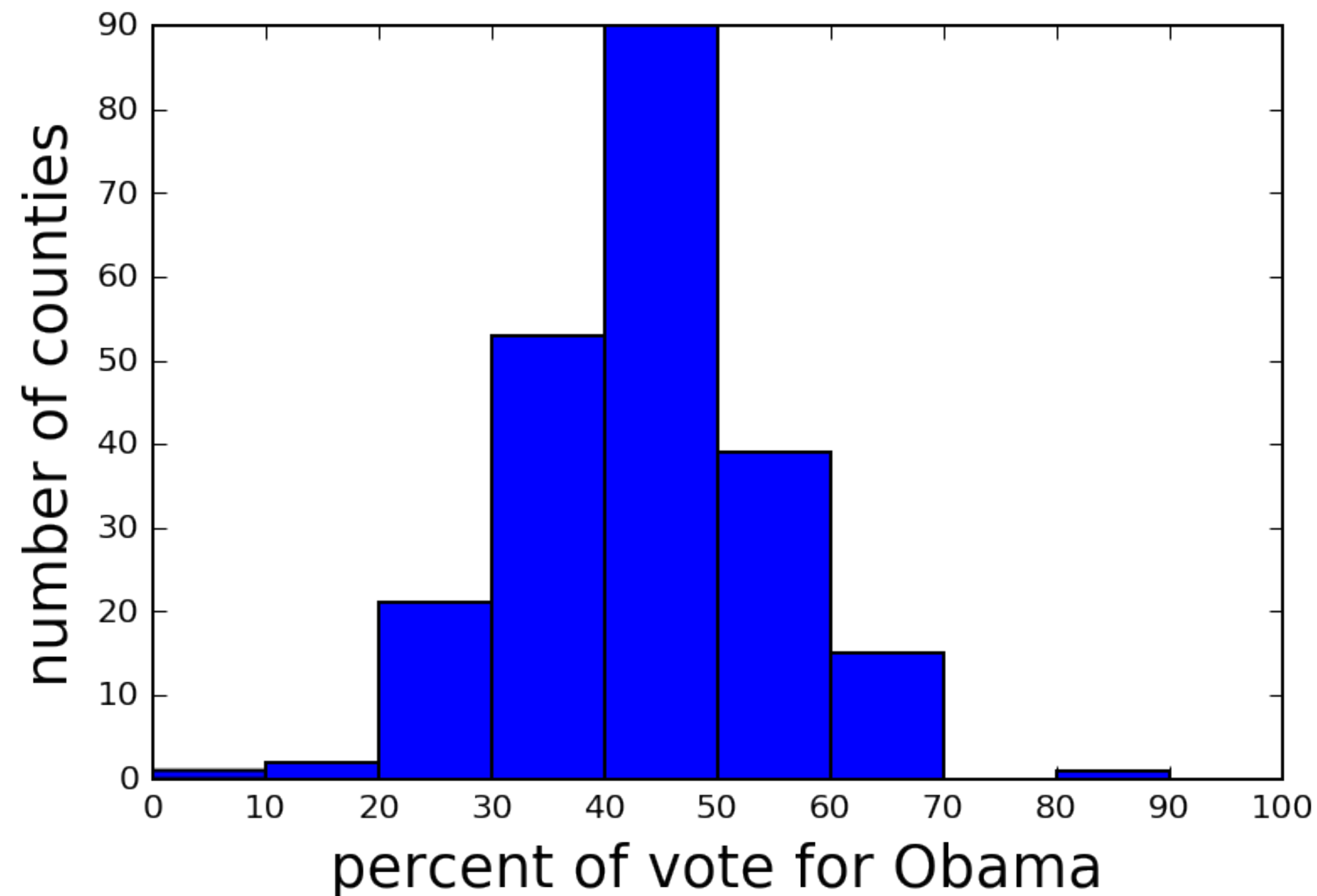


STATISTICAL THINKING IN PYTHON I

Plotting a histogram



2008 US swing state election results





Generating a histogram

```
In [1]: import matplotlib.pyplot as plt
```

```
In [2]: _ = plt.hist(df_swing['dem_share'])
```

```
In [3]: _ = plt.xlabel('percent of vote for Obama')
```

```
In [4]: _ = plt.ylabel('number of counties')
```

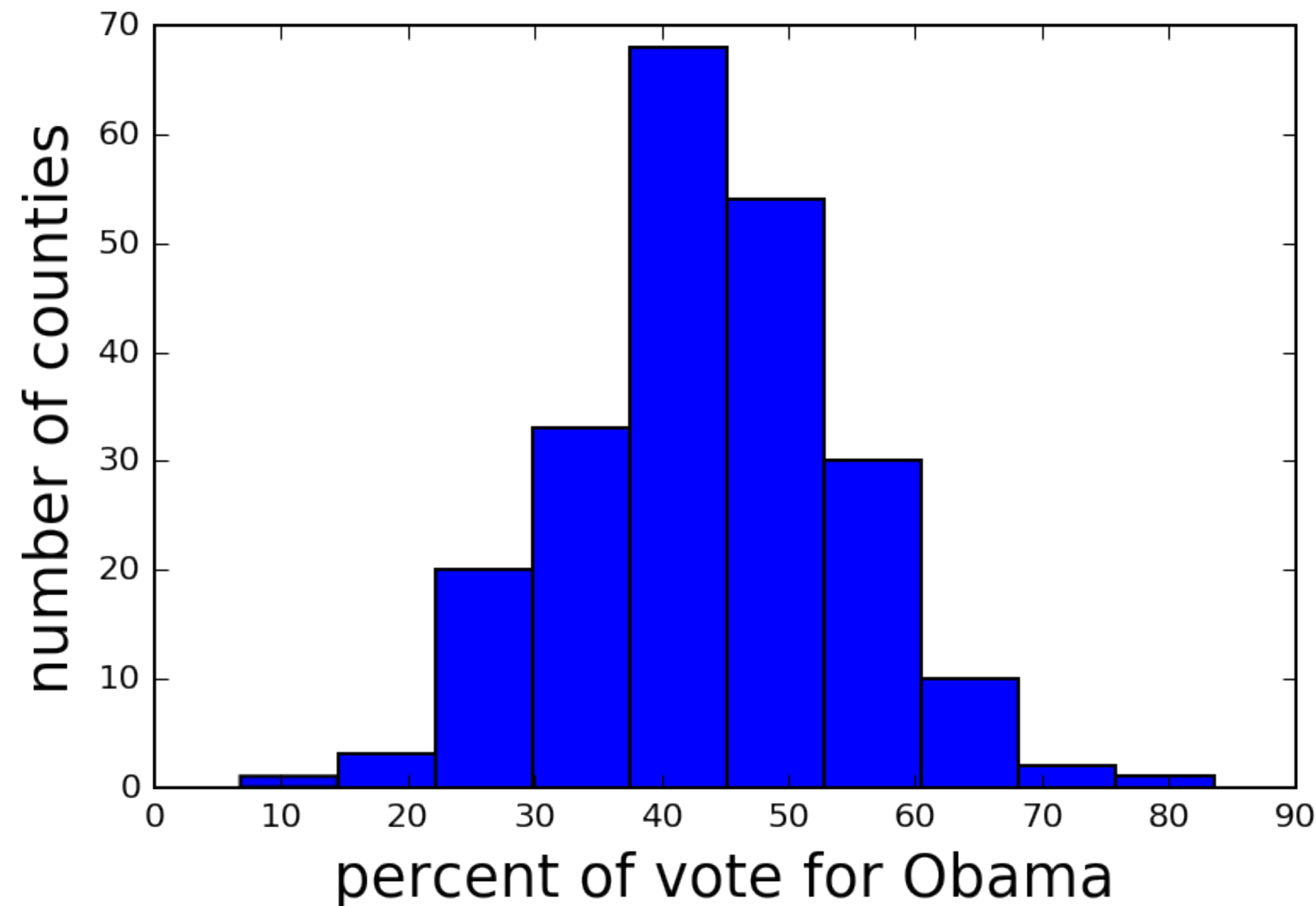
```
In [5]: plt.show()
```



- Always label your axes

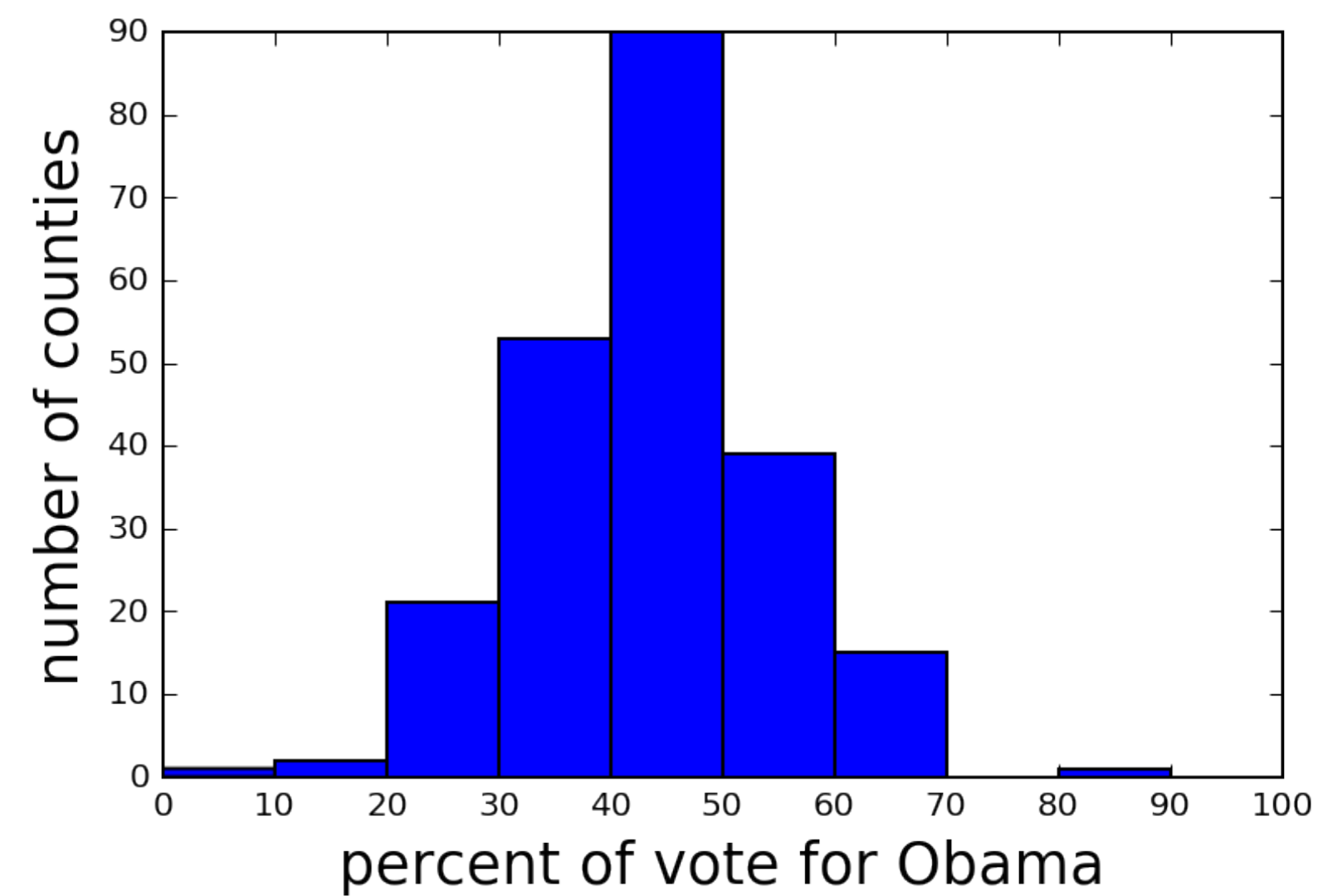
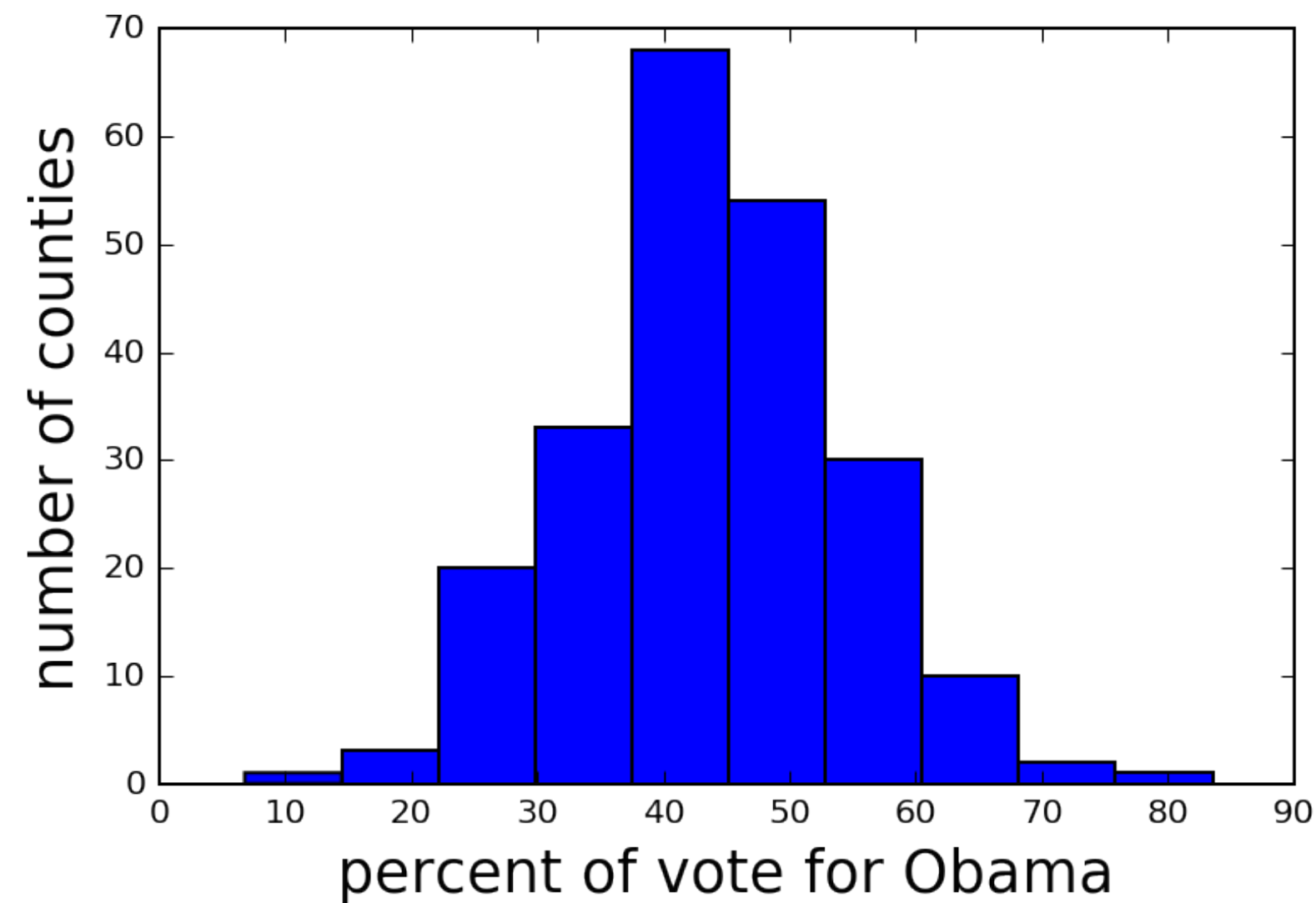


2008 US swing state election results





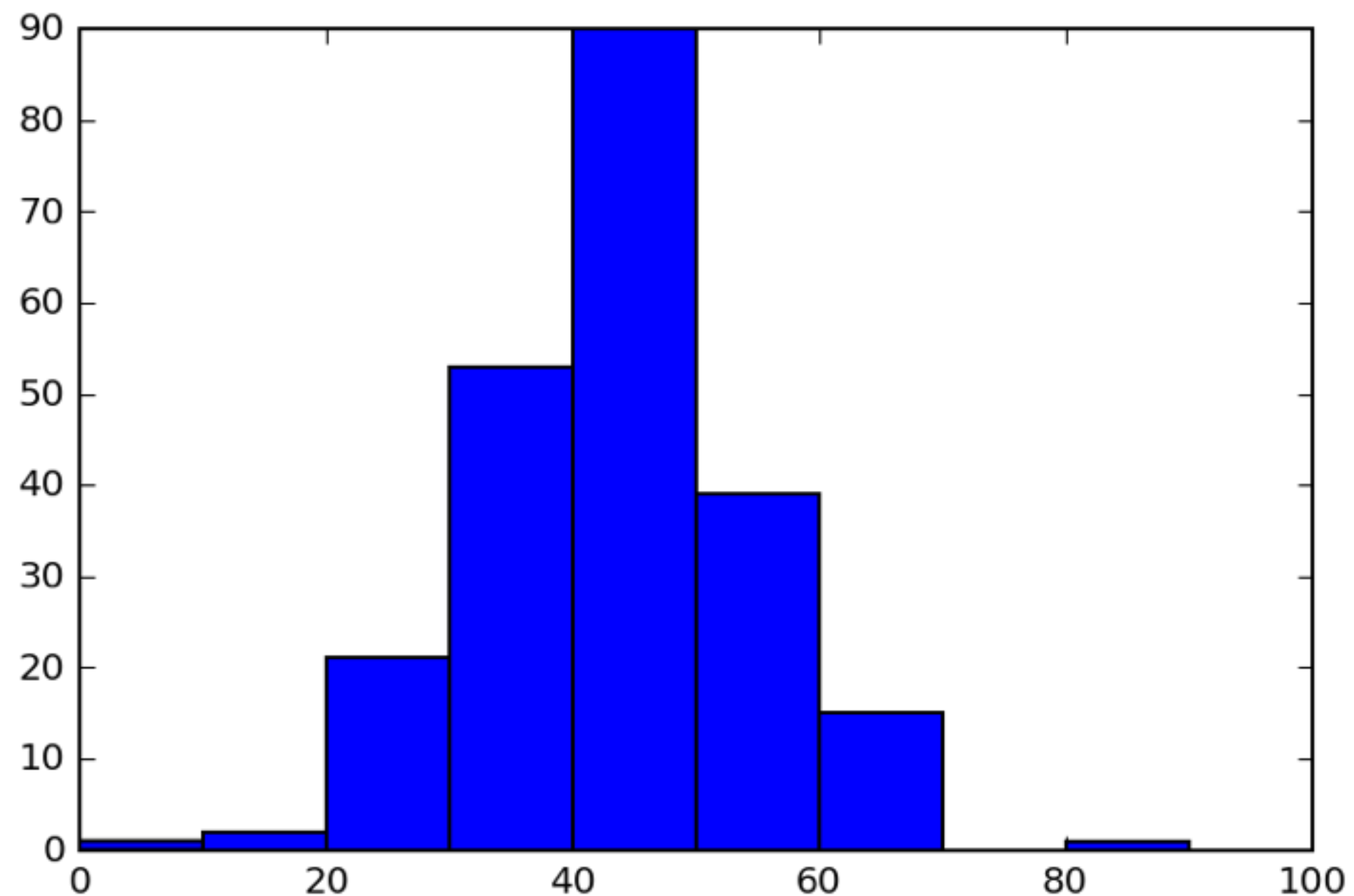
Histograms with different binning





Setting the bins of a histogram

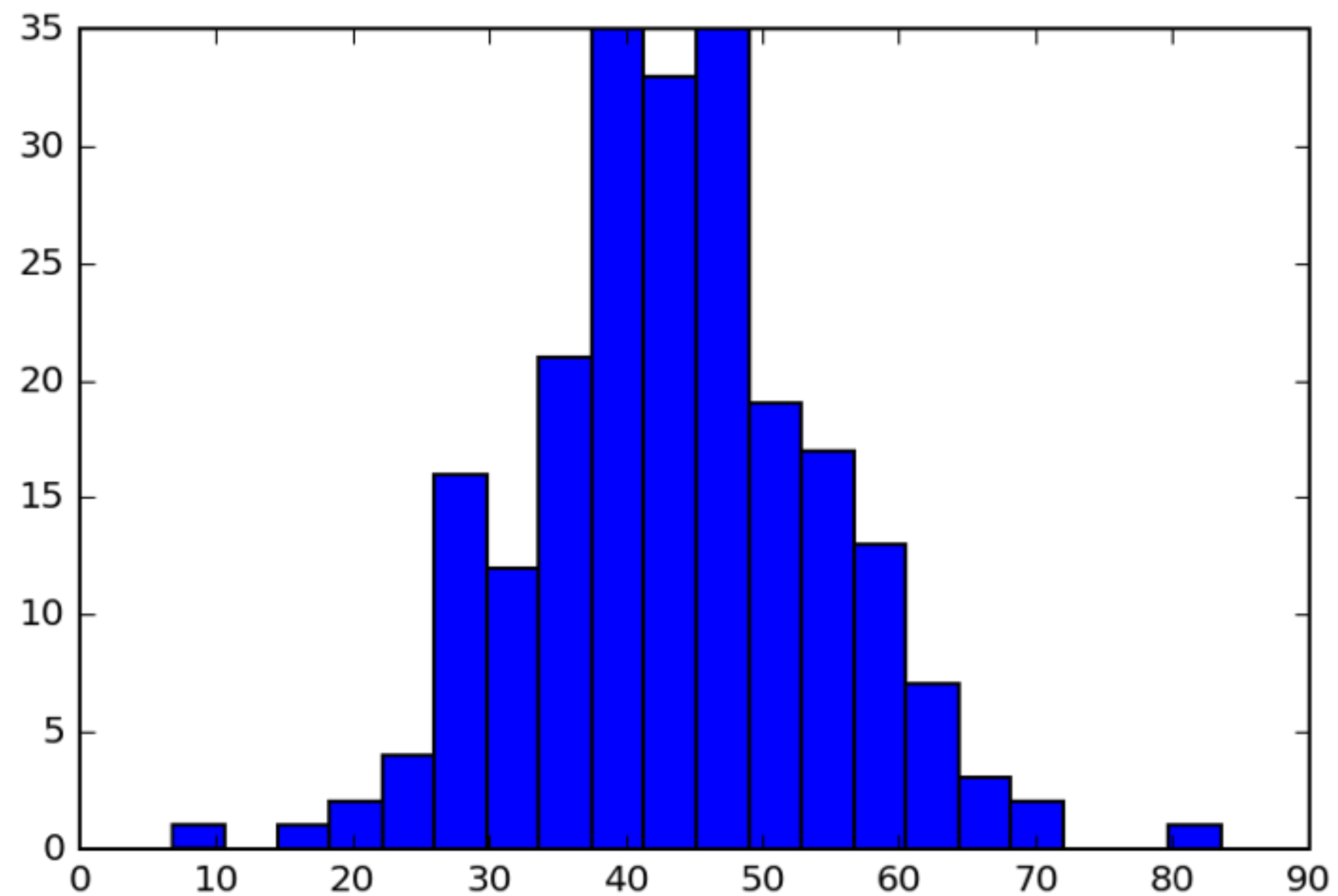
```
In [1]: bin_edges = [0, 10, 20, 30, 40, 50,  
....:                60, 70, 80, 90, 100]  
  
In [2]: _ = plt.hist(df_swing['dem_share'], bins=bin_edges)  
  
In [3]: plt.show()
```





Setting the bins of a histogram

```
In [1]: _ = plt.hist(df_swing['dem_share'], bins=20)  
  
In [2]: plt.show()
```



Seaborn

- An excellent Matplotlib-based statistical data visualization package written by Michael Waskom



Setting Seaborn styling

```
In [1]: import seaborn as sns
```

```
In [2]: sns.set()
```

```
In [3]: _ = plt.hist(df_swing['dem_share'])
```

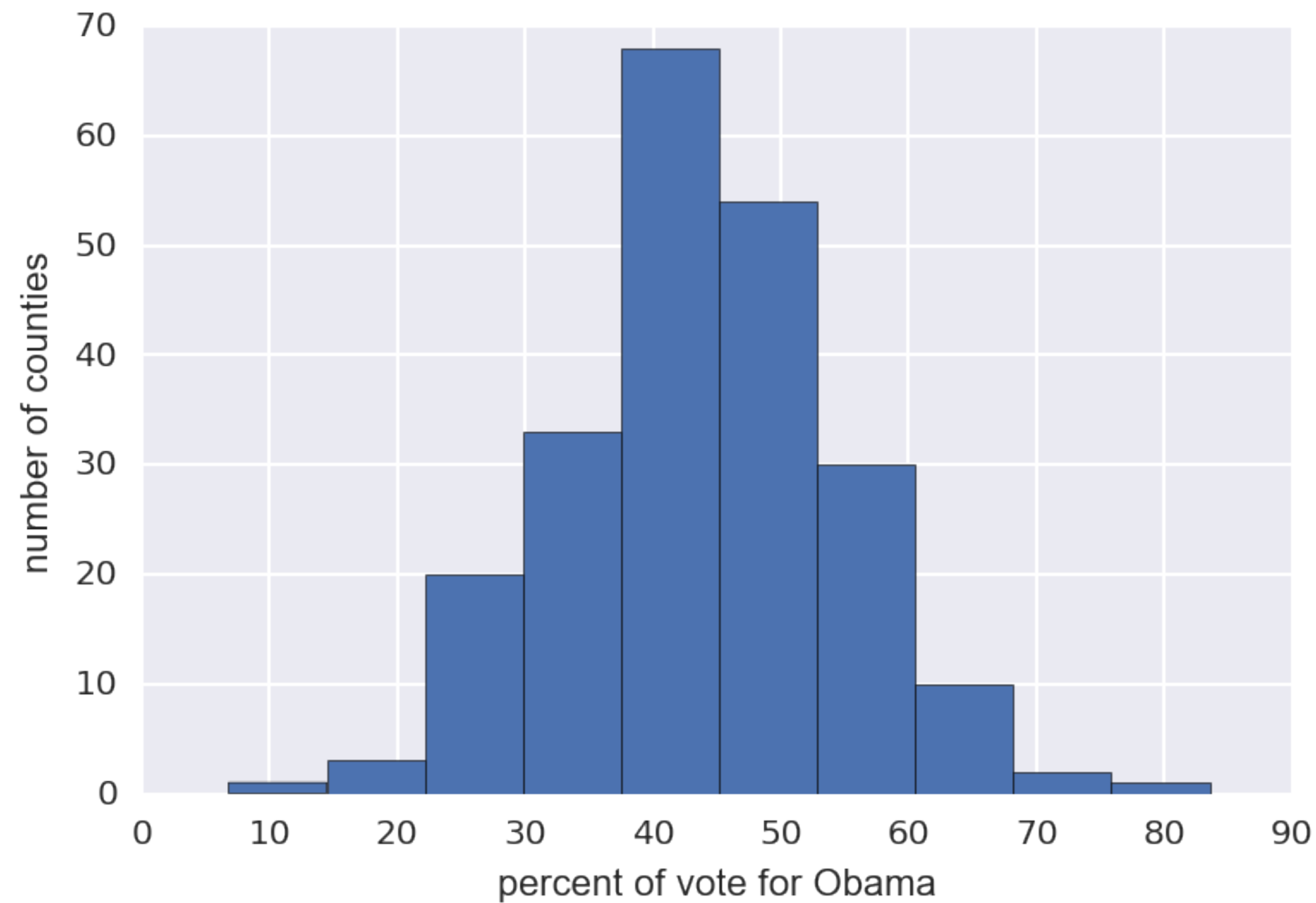
```
In [4]: _ = plt.xlabel('percent of vote for Obama')
```

```
In [5]: _ = plt.ylabel('number of counties')
```

```
In [6]: plt.show()
```



A Seaborn-styled histogram





STATISTICAL THINKING IN PYTHON I

Let's practice!

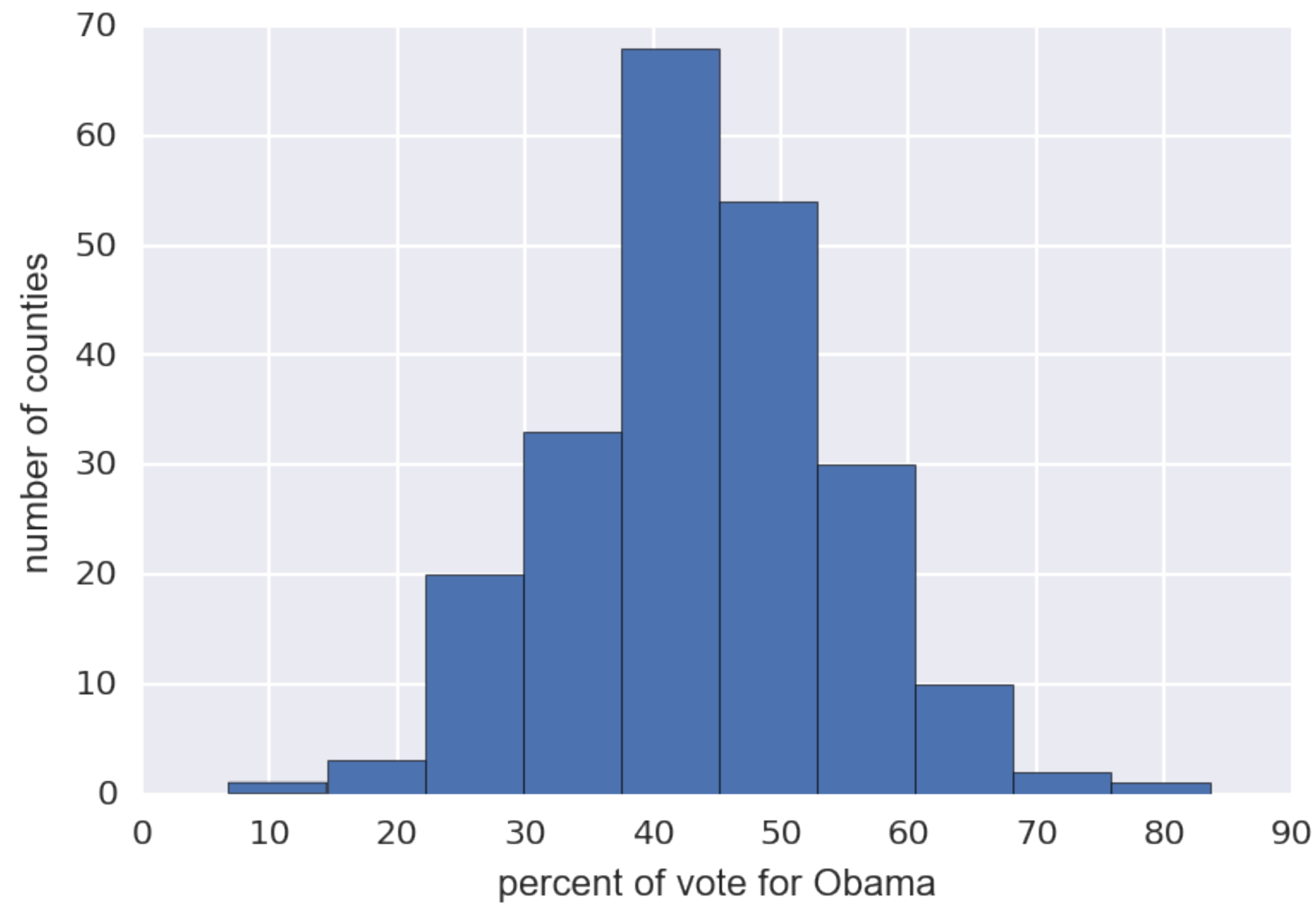


STATISTICAL THINKING IN PYTHON I

Plot all of your data: Bee swarm plots

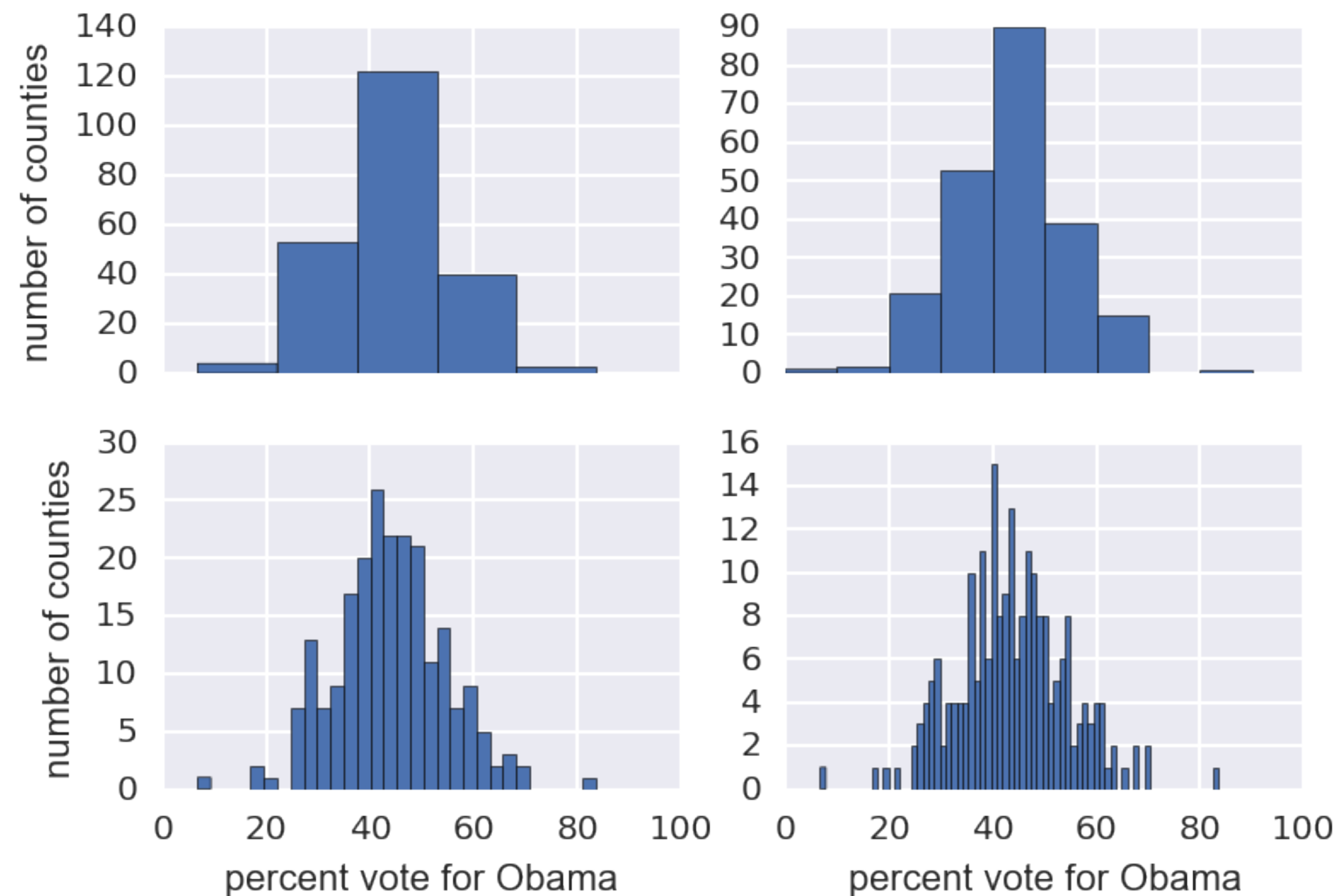


2008 US swing state election results





2008 US swing state election results



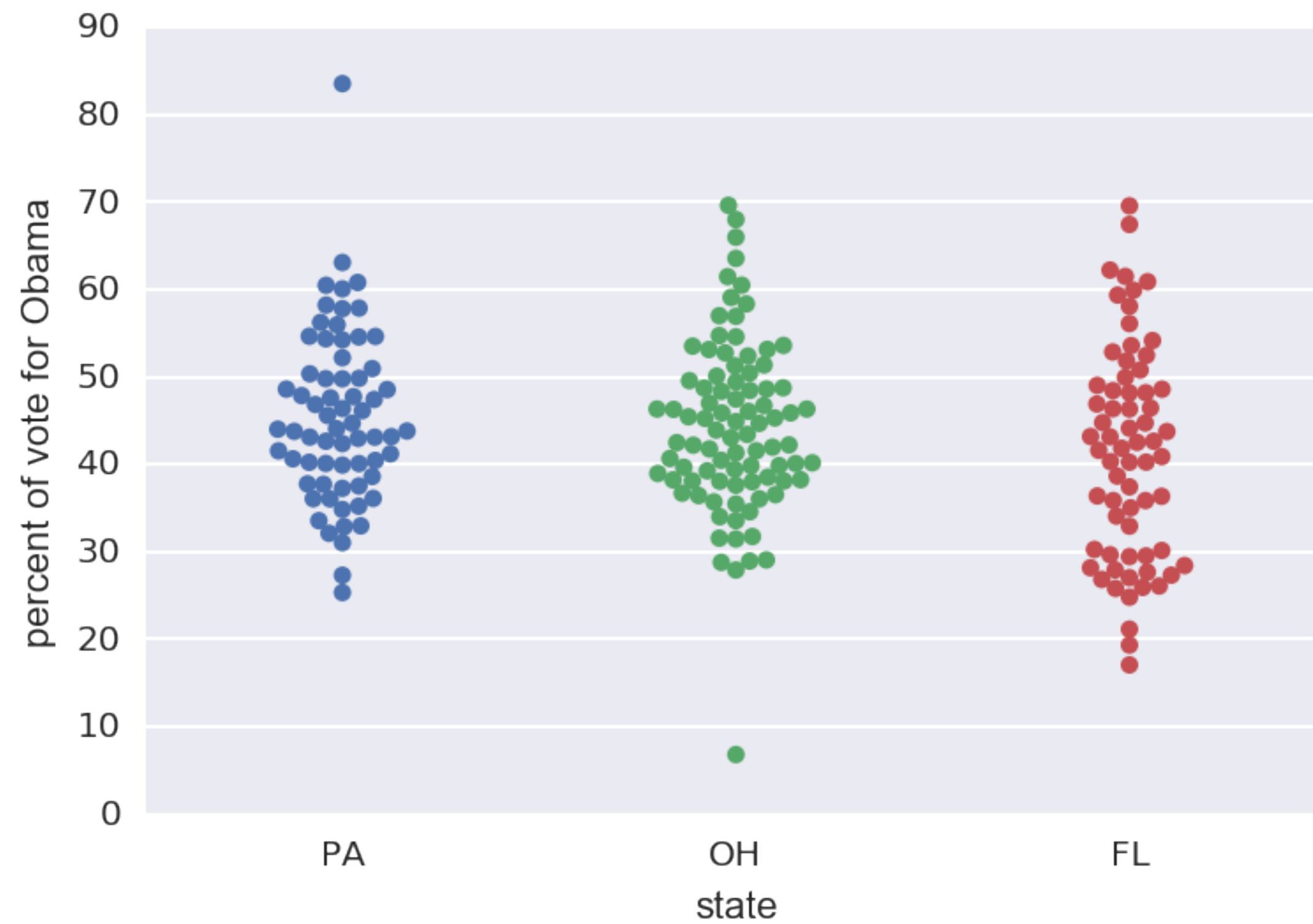


Binning bias

- The same data may be interpreted differently depending on choice of bins



Bee swarm plot





Organization of the data frame

features of interest

observation

	state	county	total_votes	dem_votes	rep_votes	dem_share
0	PA	Erie County	127691	75775	50351	60.08
1	PA	Bradford County	25787	10306	15057	40.64
2	PA	Tioga County	17984	6390	11326	36.07
3	PA	McKean County	15947	6465	9224	41.21
4	PA	Potter County	7507	2300	5109	31.04
5	PA	Wayne County	22835	9892	12702	43.78
6	PA	Susquehanna County	19286	8381	10633	44.08
7	PA	Warren County	18517	8537	9685	46.85
8	OH	Ashtabula County	44874	25027	18949	56.94
⋮	⋮	⋮	⋮	⋮	⋮	⋮



Generating a bee swarm plot

```
In [1]: _ = sns.swarmplot(x='state', y='dem_share', data=df_swing)
```

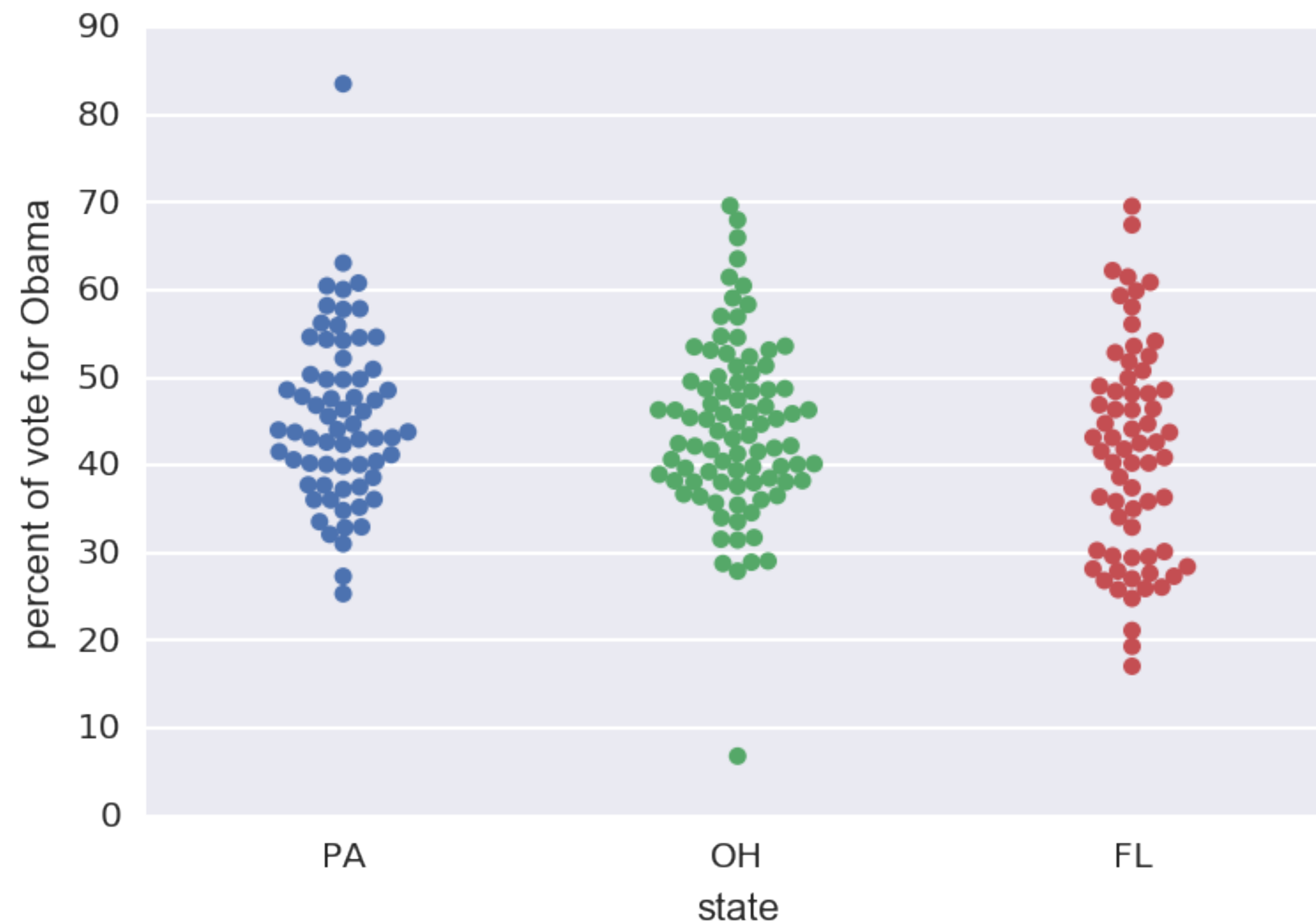
```
In [2]: _ = plt.xlabel('state')
```

```
In [3]: _ = plt.ylabel('percent of vote for Obama')
```

```
In [4]: plt.show()
```



2008 US swing state election results





STATISTICAL THINKING IN PYTHON I

Let's practice!

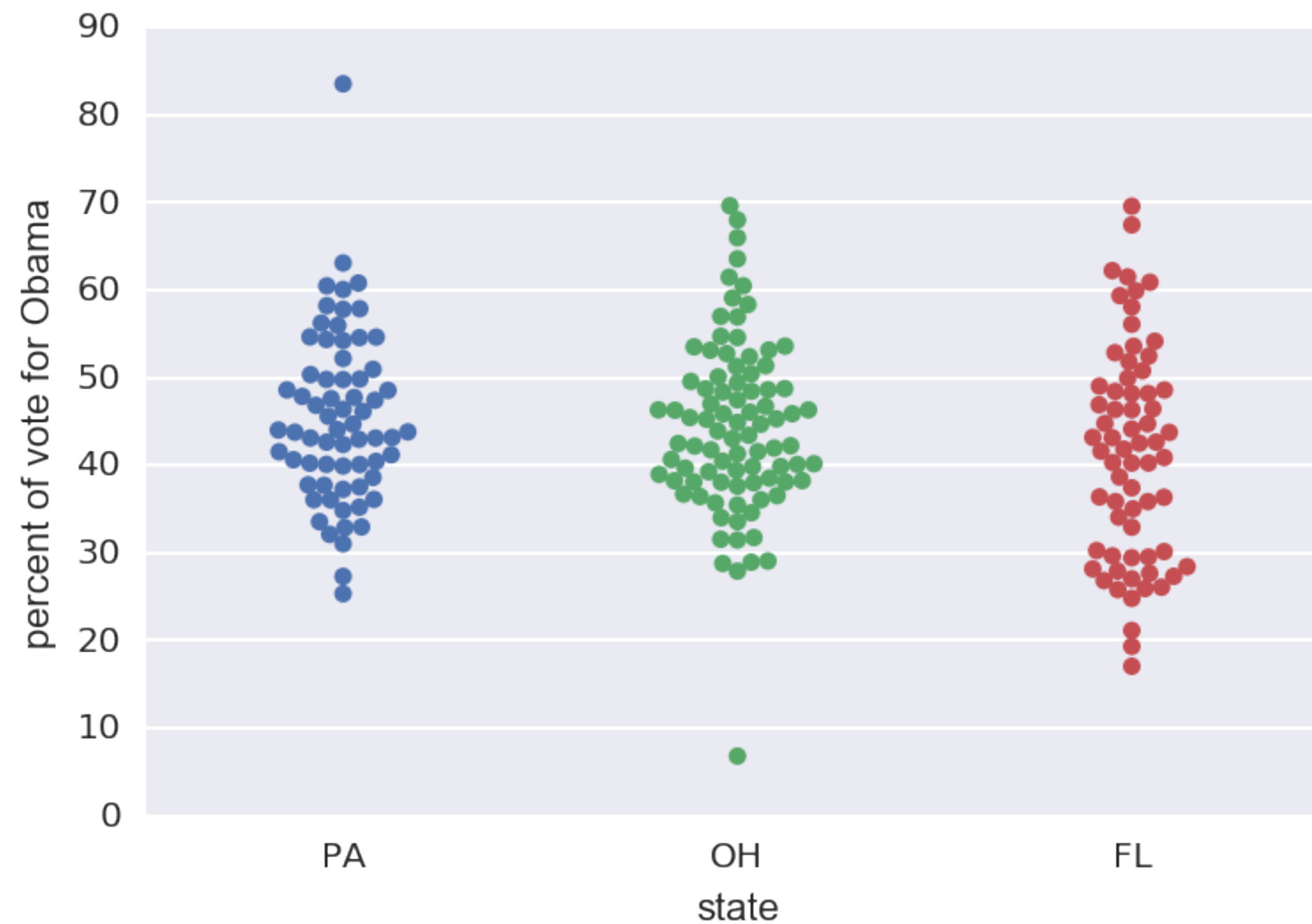


STATISTICAL THINKING IN PYTHON I

Plot all of your data: ECDFs

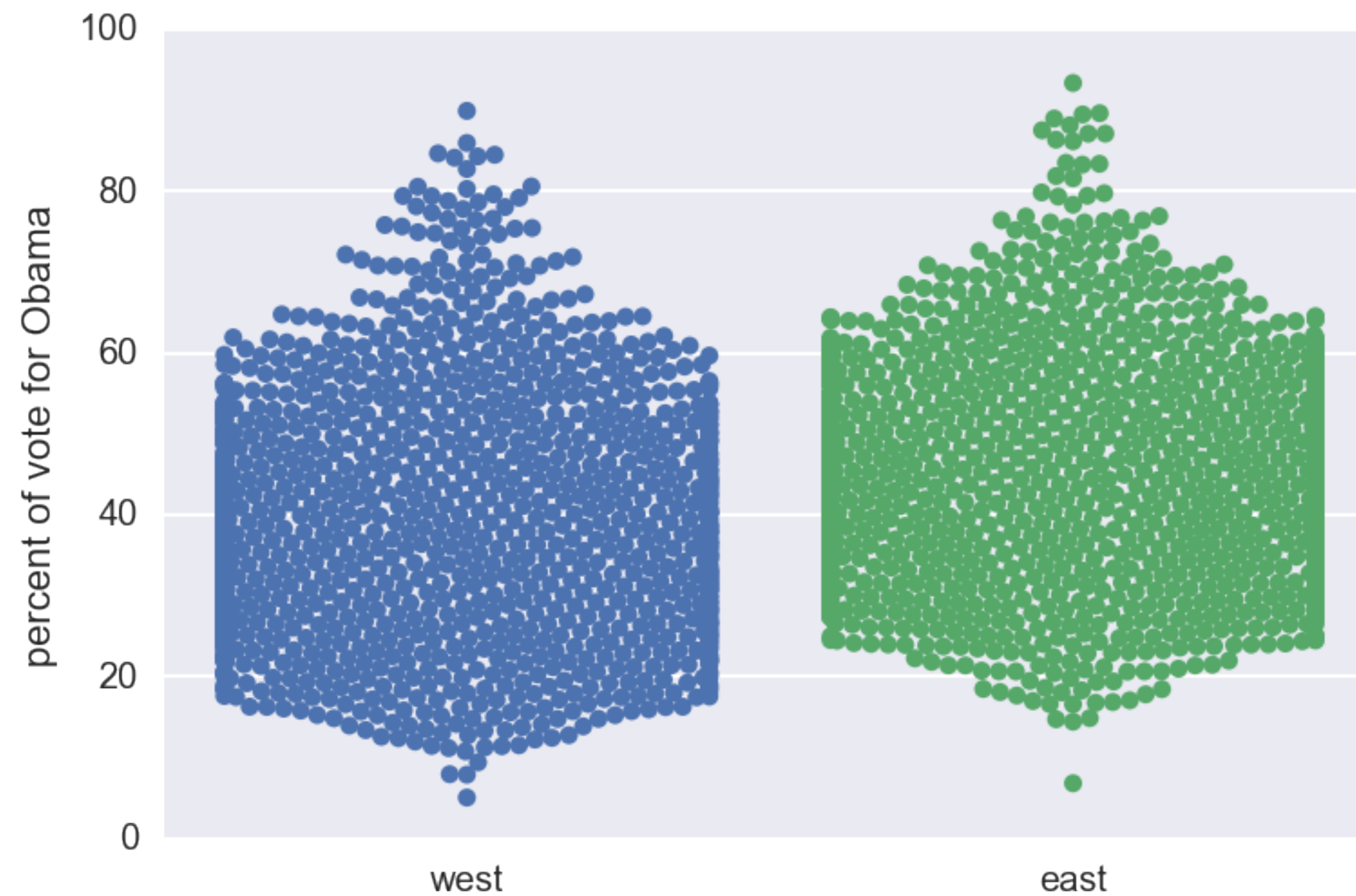


2008 US swing state election results



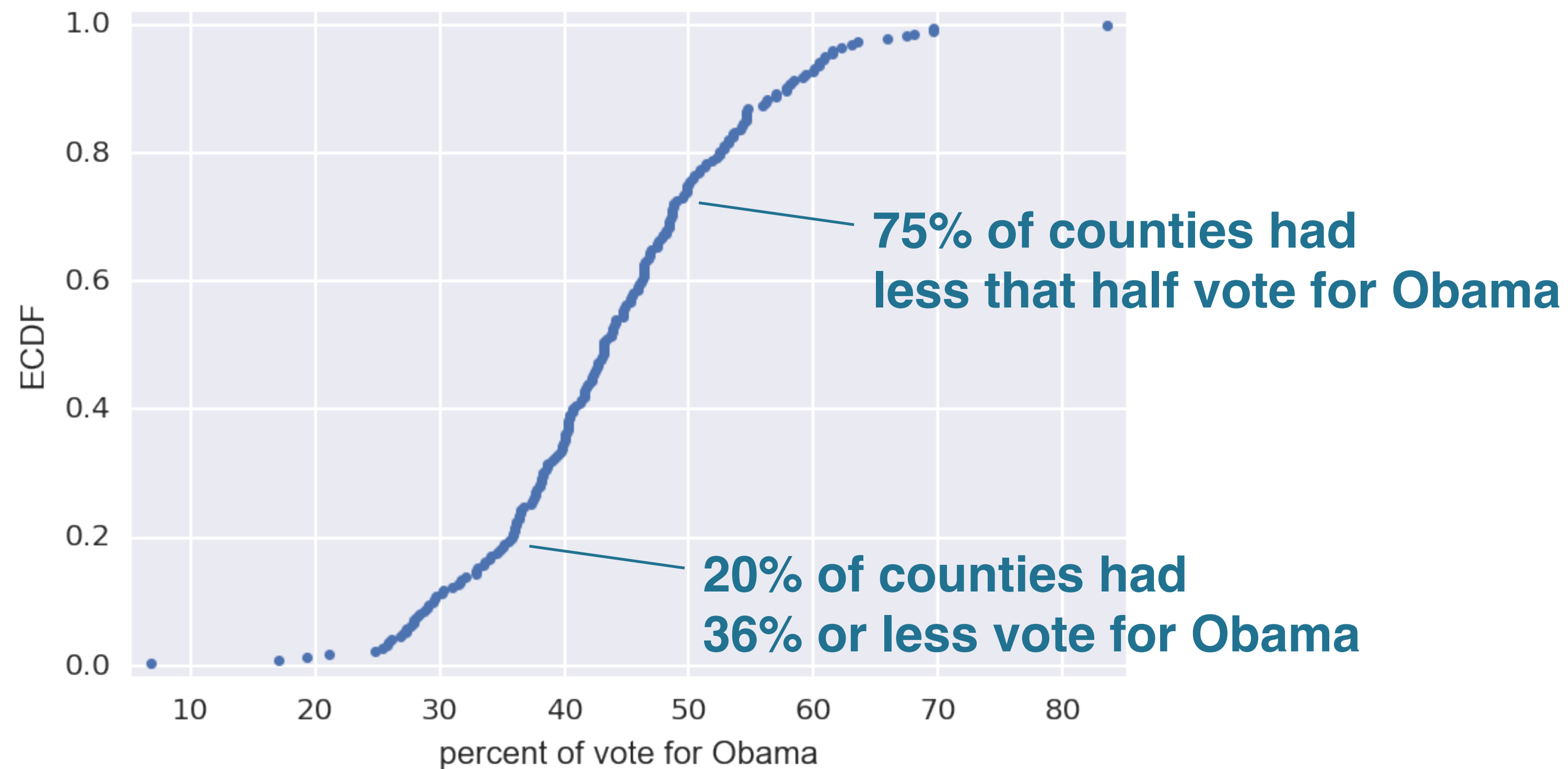


2008 US election results: East and West





Empirical cumulative distribution function (ECDF)





Making an ECDF

```
In [1]: import numpy as np
```

```
In [2]: x = np.sort(df_swing['dem_share'])
```

```
In [3]: y = np.arange(1, len(x)+1) / len(x)
```

```
In [4]: _ = plt.plot(x, y, marker='.', linestyle='none')
```

```
In [5]: _ = plt.xlabel('percent of vote for Obama')
```

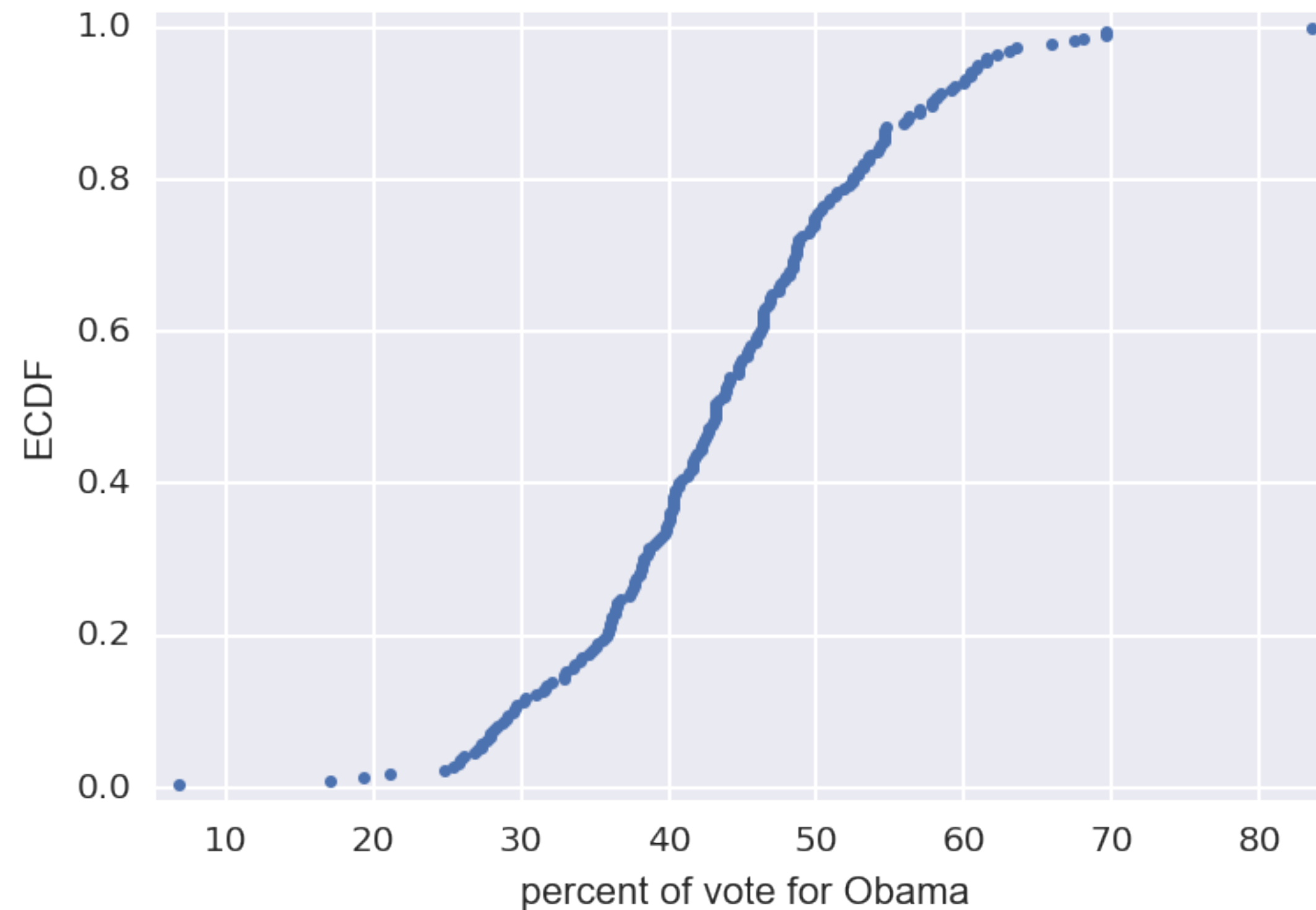
```
In [6]: _ = plt.ylabel('ECDF')
```

```
In [7]: plt.margins(0.02) # Keeps data off plot edges
```

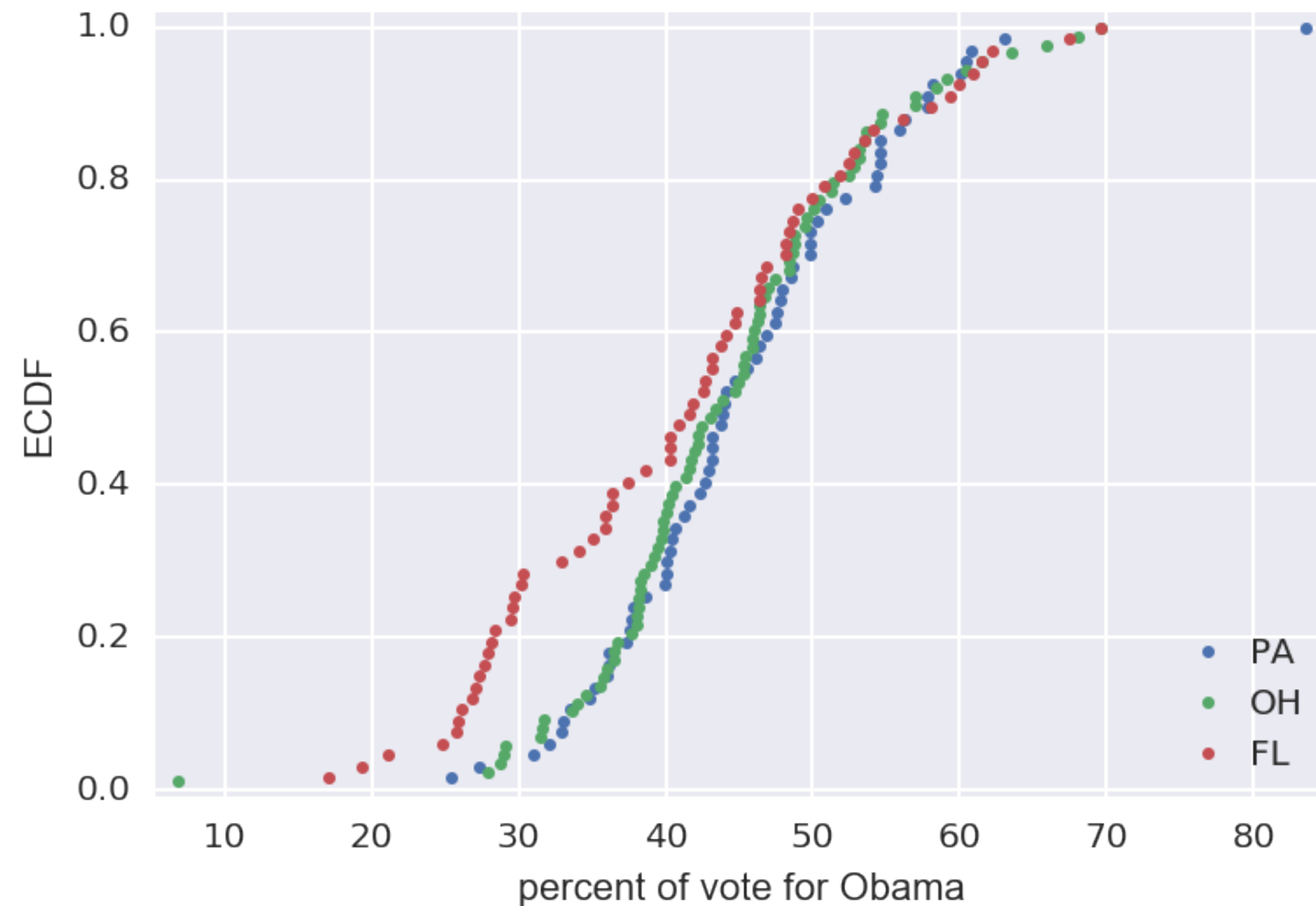
```
In [8]: plt.show()
```



2008 US swing state election ECDF



2008 US swing state election ECDFs





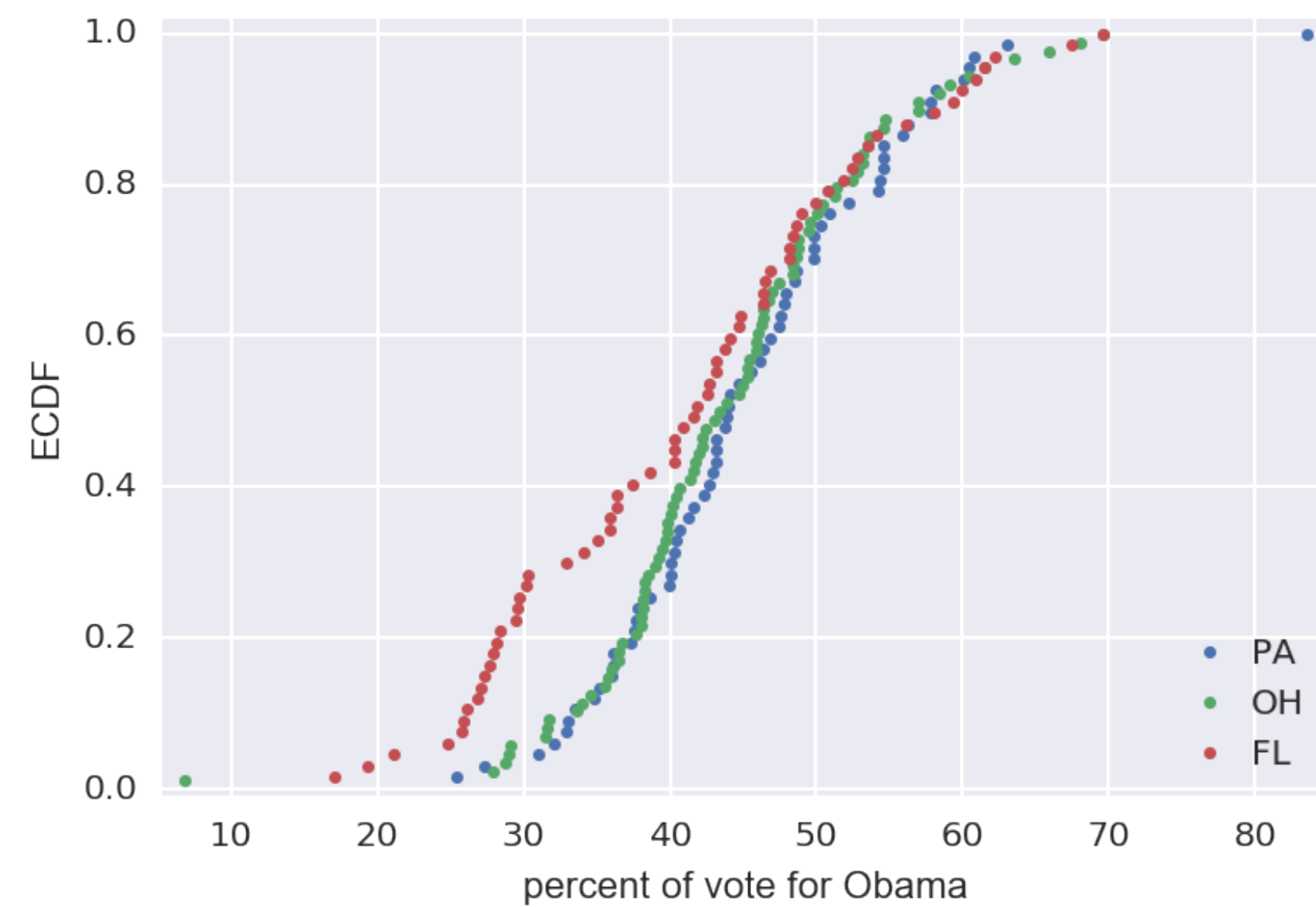
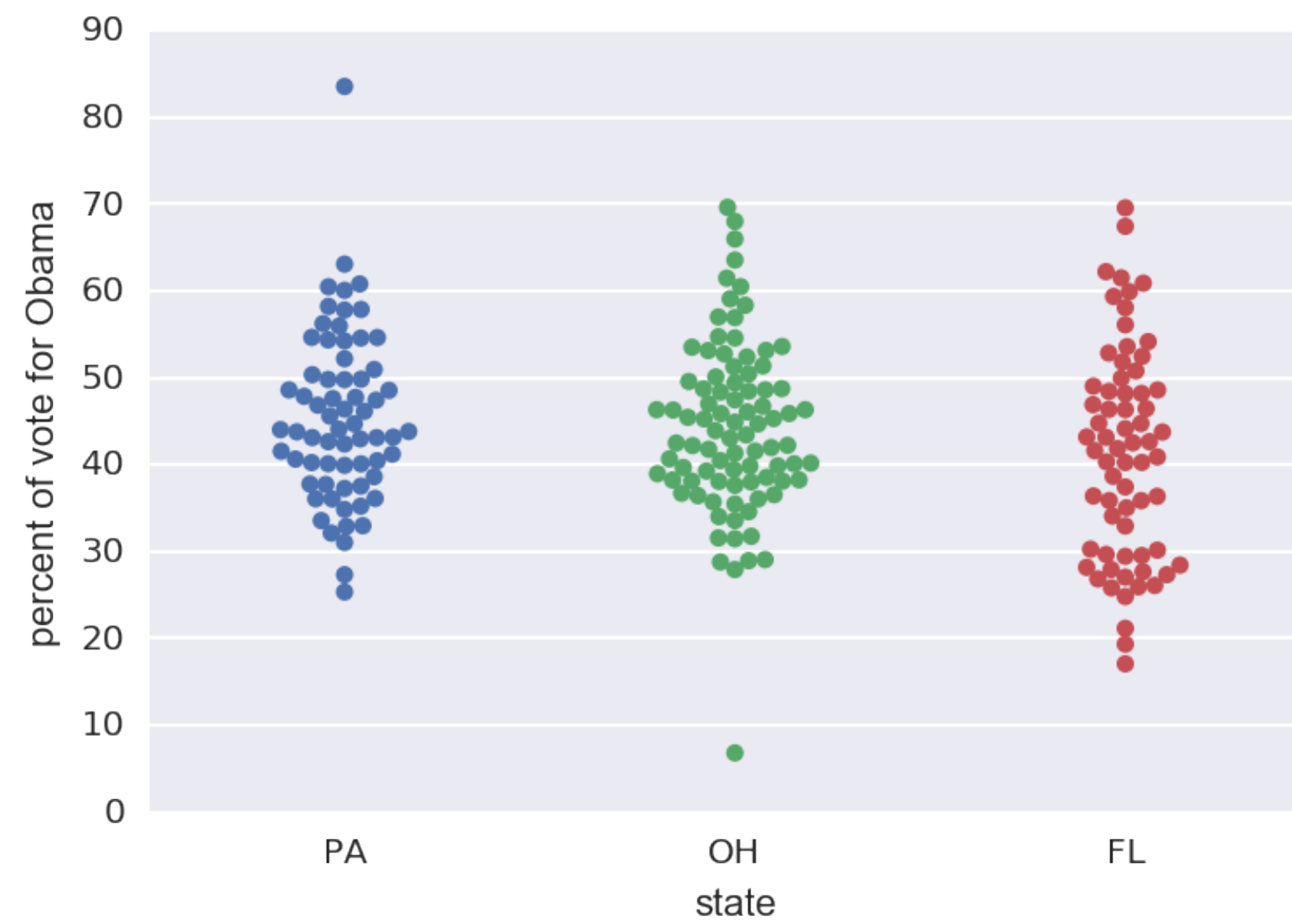
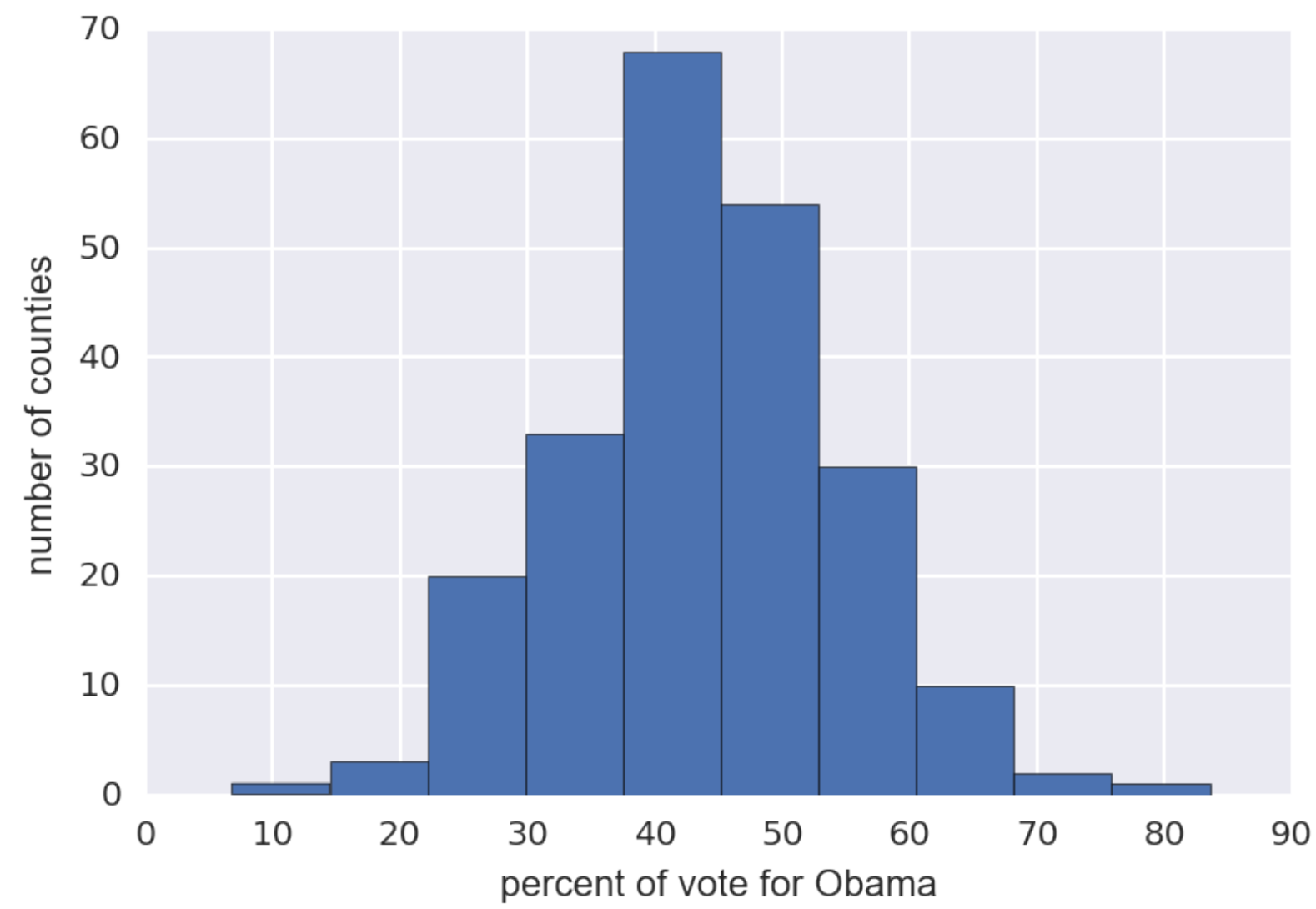
STATISTICAL THINKING IN PYTHON I

Let's practice!



STATISTICAL THINKING IN PYTHON I

**Onward toward
the whole story!**





“Exploratory data analysis can never be the whole story, but nothing else can serve as the foundation stone.”

—John Tukey

Coming up...

- Thinking probabilistically
- Discrete and continuous distributions
- The power of hacker statistics using `np.random()`



STATISTICAL THINKING IN PYTHON I

Let's get to work!