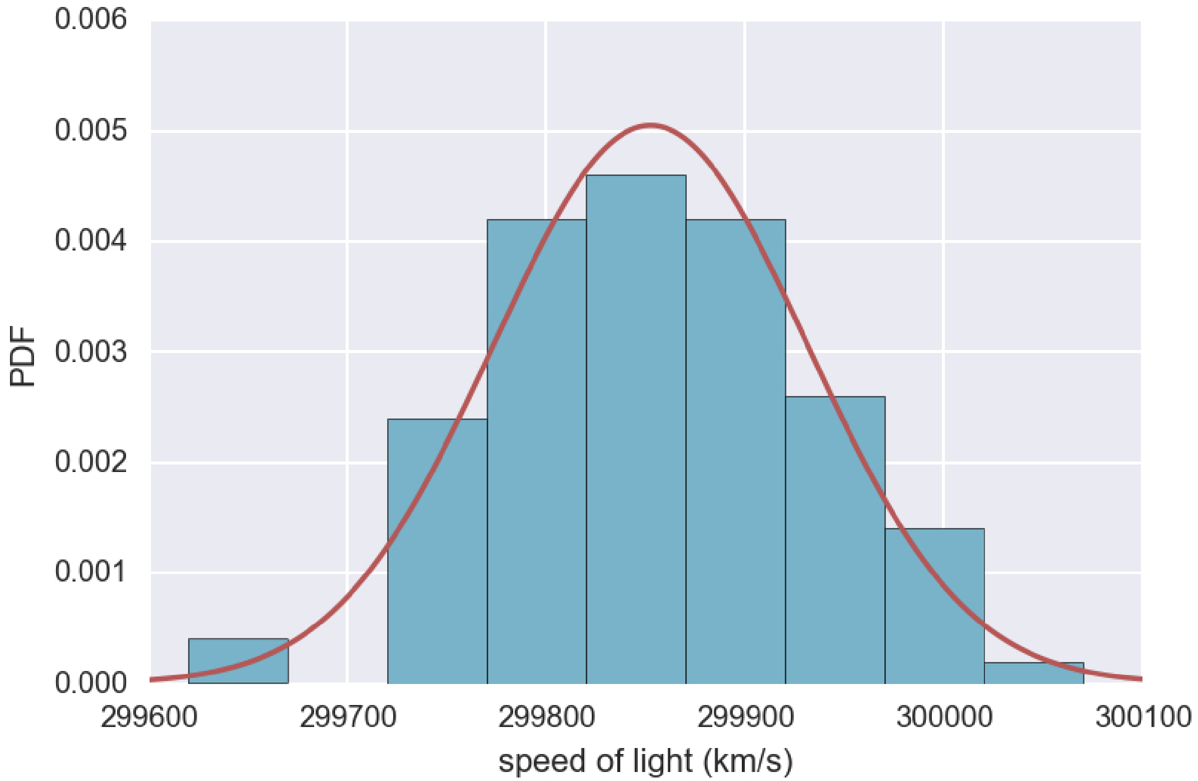
Generating bootstrap replicates

STATISTICAL THINKING IN PYTHON (PART 2)

Justin Bois

Lecturer at the California Institute of Technology

Michelson's speed of light measurements



[1] Data: Michelson, 1880

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[    ,     ,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[    ,     ,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3,      , 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[27.1,      ,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[27.1,     ,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[27.1, 26.0,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.3, 26.0]
```

Mean = 25.2

Resampled data:

```
[27.1, 26.0,     ,     ,     ]
```

Resampling an array

Data:

```
[23.3, 27.1, 24.3, 25.7, 26.0]
```

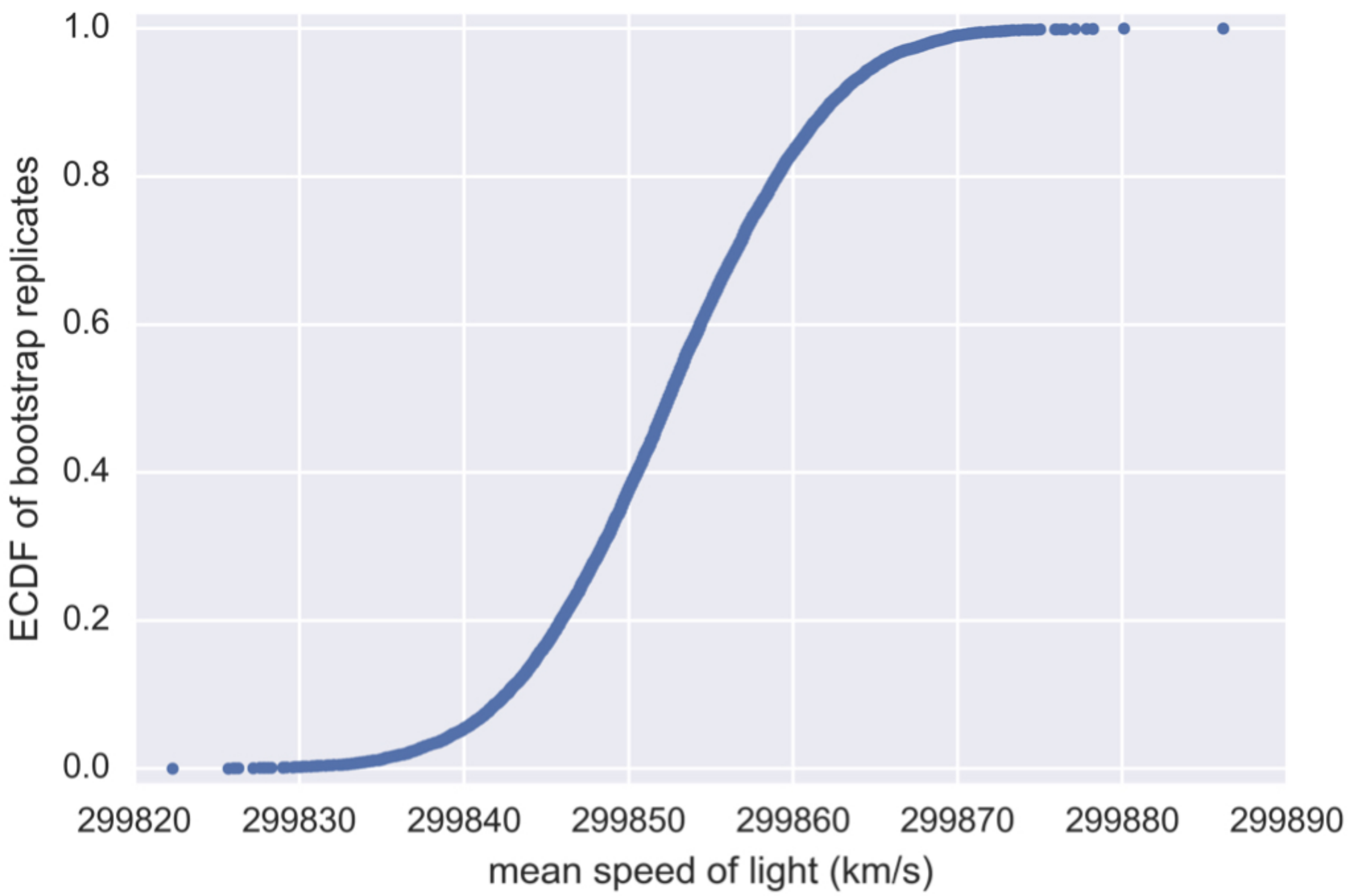
Mean = 25.2

Resampled data:

```
[27.1, 26.0, 23.3, 25.7, 23.3]
```

Mean = 25.08

Mean of resampled Michelson measurements



Bootstrapping

- The use of resampled data to perform statistical inference

Bootstrap sample

- A resampled array of the data

Bootstrap replicate

- A statistic computed from a resampled array

Resampling engine: np.random.choice()

```python
import numpy as np
np.random.choice([1,2,3,4,5], size=5)
```

```
array([5, 3, 5, 5, 2])
```

Computing a bootstrap replicate

```
bs_sample = np.random.choice(michelson_speed_of_light,
                             size=100)
np.mean(bs_sample)
```

```
299847.79999999999
```

```
np.median(bs_sample)
```

```
299845.0
```

```
np.std(bs_sample)
```

```
83.564286025729331
```

Let's practice!

STATISTICAL THINKING IN PYTHON (PART 2)

Bootstrap confidence intervals

STATISTICAL THINKING IN PYTHON (PART 2)

Justin Bois

Lecturer at the California Institute of Technology

Bootstrap replicate function

```python
def bootstrap_replicate_1d(data, func):
    """Generate bootstrap replicate of 1D data."""
    bs_sample = np.random.choice(data, len(data))
    return func(bs_sample)
bootstrap_replicate_1d(michelson_speed_of_light, np.mean)
```

```
299859.20000000001
```

```python
bootstrap_replicate_1d(michelson_speed_of_light, np.mean)
```

```
299855.70000000001
```

```python
bootstrap_replicate_1d(michelson_speed_of_light, np.mean)
```

```
299850.29999999999
```

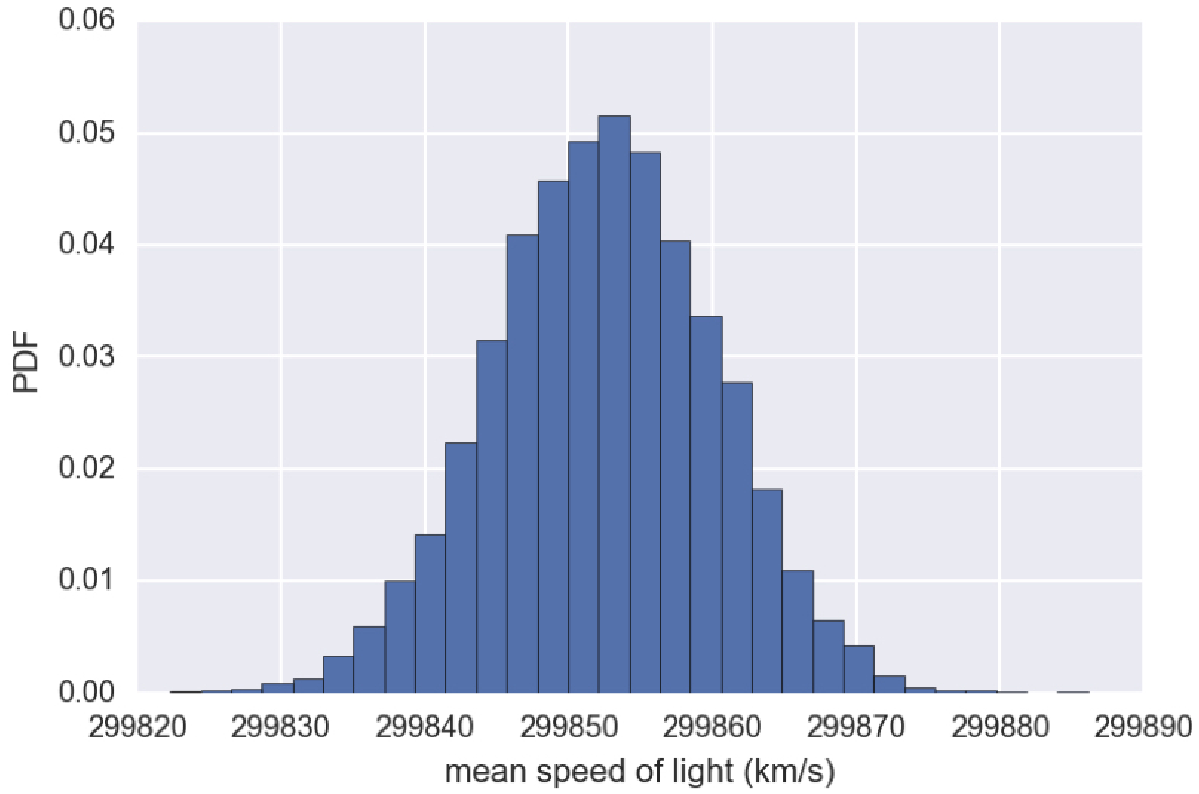
Many bootstrap replicates

```python
bs_replicates = np.empty(10000)
for i in range(10000):
    bs_replicates[i] = bootstrap_replicate_1d(
                           michelson_speed_of_light, np.mean)
```

Plotting a histogram of bootstrap replicates

```
_ = plt.hist(bs_replicates, bins=30, normed=True)
_ = plt.xlabel('mean speed of light (km/s)')
_ = plt.ylabel('PDF')
plt.show()
```

Bootstrap estimate of the mean



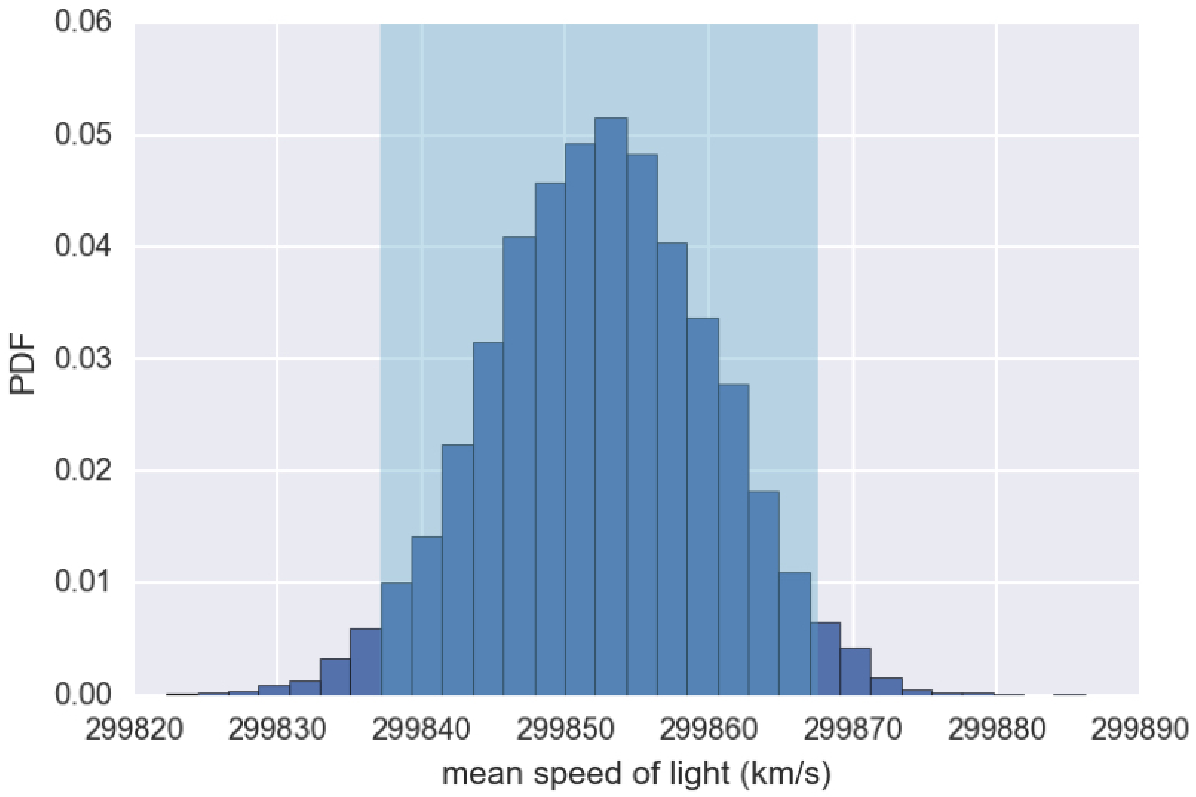
Confidence interval of a statistic

- If we repeated measurements over and over again, `p%` of the observed values would lie within the `p%` confidence interval.

Bootstrap confidence interval

```
conf_int = np.percentile(bs_replicates, [2.5, 97.5])
```

```
array([ 299837.,  299868.])
```



Let's practice!

STATISTICAL THINKING IN PYTHON (PART 2)

Pairs bootstrap

STATISTICAL THINKING IN PYTHON (PART 2)

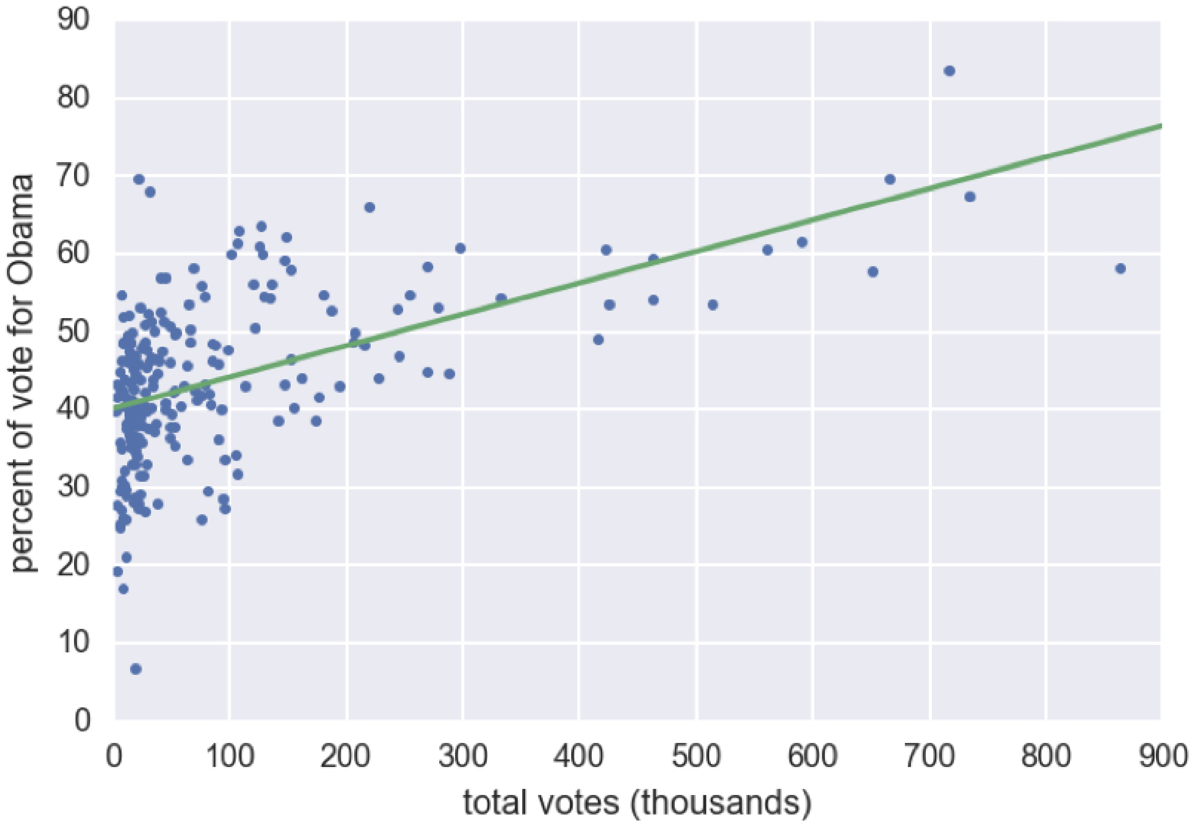
Justin Bois

Lecturer at the California Institute of Technology

Nonparametric inference

- Make no assumptions about the model or probability distribution underlying the data

2008 US swing state election results



[1] Data retrieved from Data.gov (https://www.data.gov/)

Pairs bootstrap for linear regression

- Resample data in pairs
- Compute slope and intercept from resampled data
- Each slope and intercept is a bootstrap replicate
- Compute confidence intervals from percentiles of bootstrap replicates

Generating a pairs bootstrap sample

```
np.arange(7)
```

```
array([0, 1, 2, 3, 4, 5, 6])
```

```
inds = np.arange(len(total_votes))
bs_inds = np.random.choice(inds, len(inds))
bs_total_votes = total_votes[bs_inds]
bs_dem_share = dem_share[bs_inds]
```

Computing a pairs bootstrap replicate

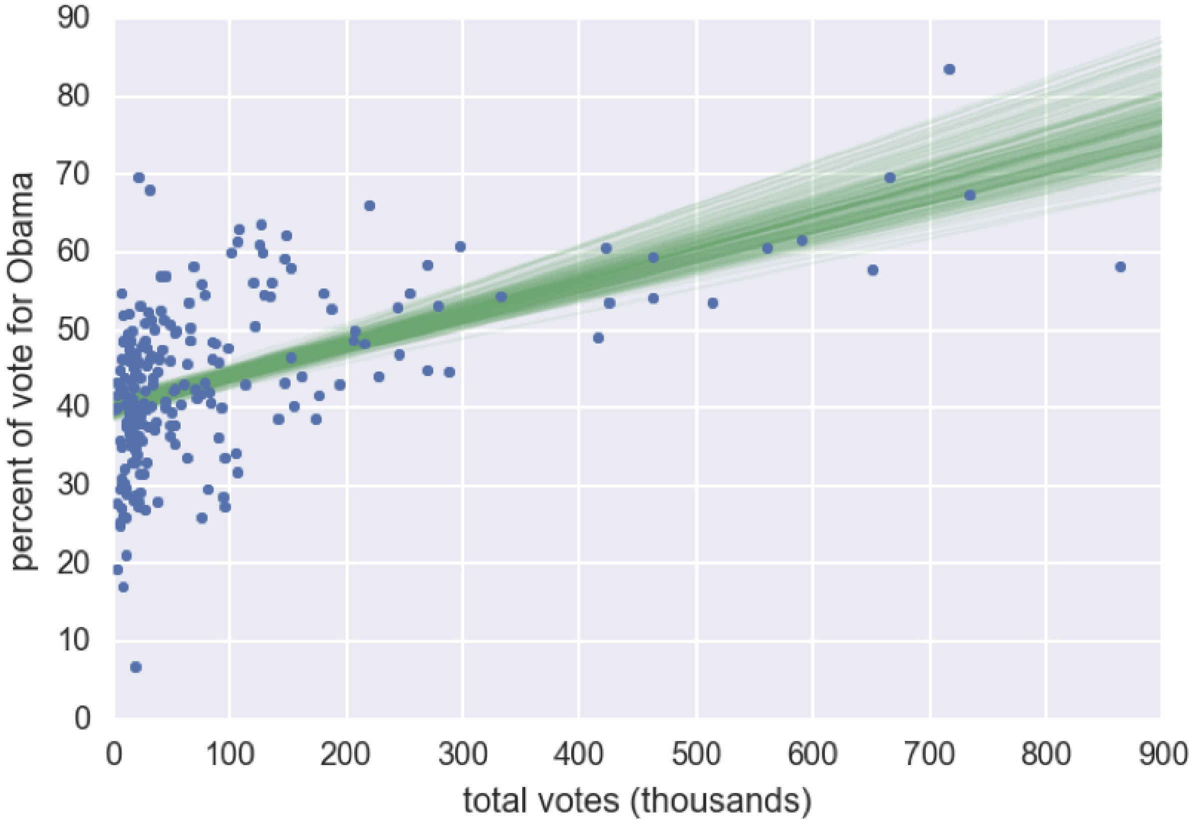
```
bs_slope, bs_intercept = np.polyfit(bs_total_votes,
                                    bs_dem_share, 1)
bs_slope, bs_intercept
```

```
(3.9053605692223672e-05, 40.387910131803025)
```

```
np.polyfit(total_votes, dem_share, 1)  # fit of original
```

```
array([  4.03707170e-05,   4.01139120e+01])
```

2008 US swing state election results



[1] Data retrieved from Data.gov (https://www.data.gov/)

Let's practice!

STATISTICAL THINKING IN PYTHON (PART 2)