12.1) Bootstrap

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): Ch 13.2 Bootstrap

https://www.inferentialthinking.com/

Employee Compensation in San Francisco

```
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
sf2015 = Table.read_table(path_data + 'san_francisco_2015.csv')
```

Total Salary	Retirement	Health/Dental	Other Benefits	Total Benefits	Total Compensation
82146	16942.2	12340.9	6337.73	35620.8	117767
33987.9	0	4587.51	2634.42	7221.93	41209.8

```
sf2015.where('Job', are.equal_to('Mayor'))
```

Total Salary	Retirement	Health/Dental	Other Benefits	Total Benefits	Total Compensation
288964	58117	12424.5	20293	90834.5	379798

Ignore the Lower Values in the Total Compensation

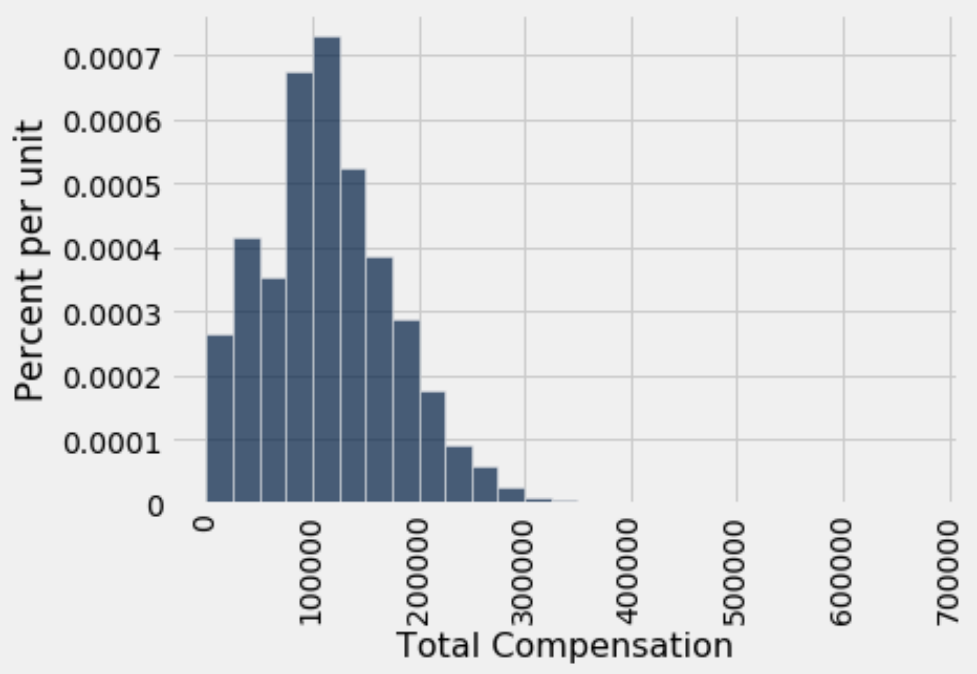
```
sf2015.sort('Total Compensation')
```

Total Salary	Retirement	Health/Dental	Other Benefits	Total Benefits	Total Compensation
0	0	0	-423.76	-423.76	-423.76
-292.4	0	-95.58	-22.63	-118.21	-410.61

```
sf2015 = sf2015.where('Salaries', are.above(10000))
```

36569

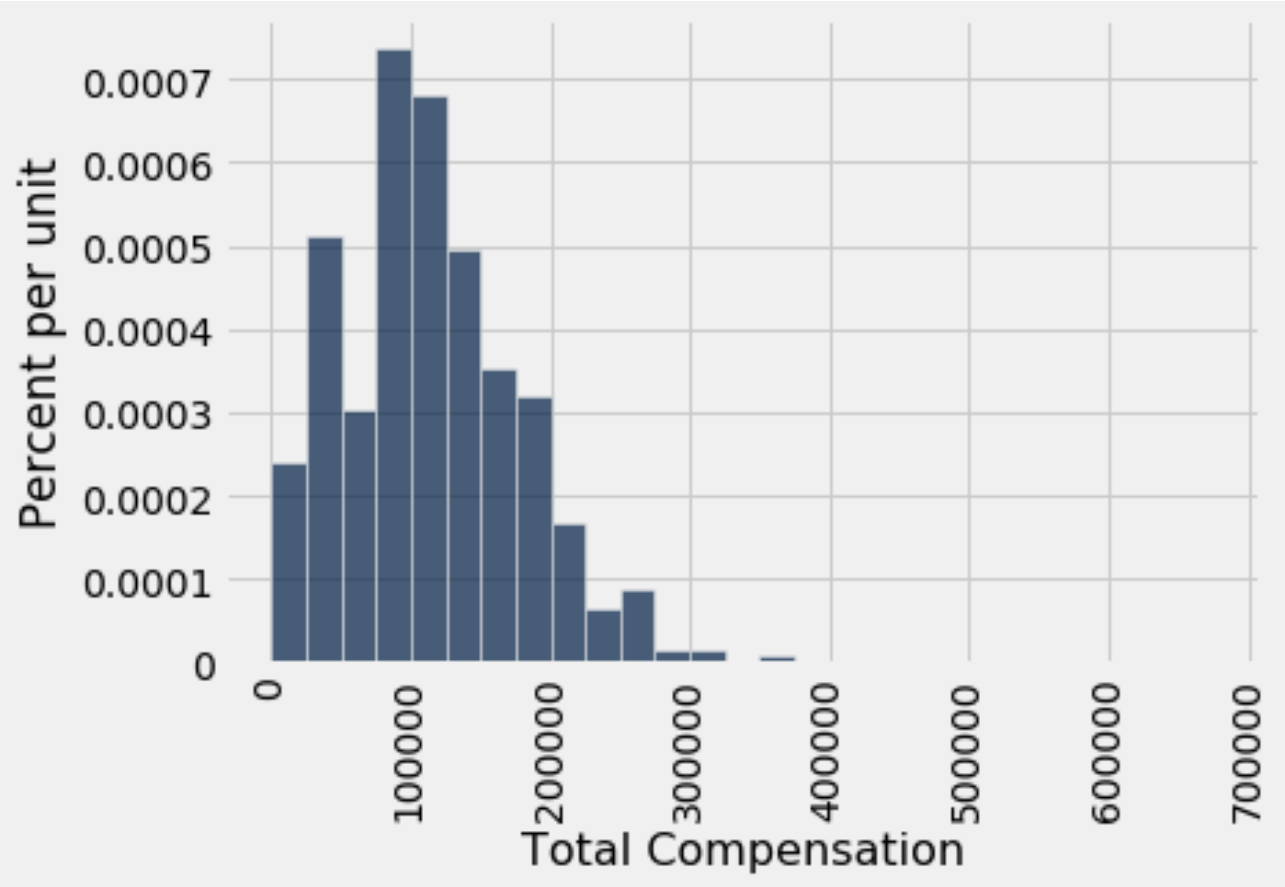
```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
sf_bins = np.arange(0, 700000, 25000)
sf2015.select('Total Compensation').hist(bins=sf_bins)
```




```
pop_median = percentile(50,
      sf2015.column('Total Compensation'))
```

110305.79

```
our_sample = sf2015.sample(500, with_replacement=False)
our_sample.select('Total Compensation').hist(bins=sf_bins)
```

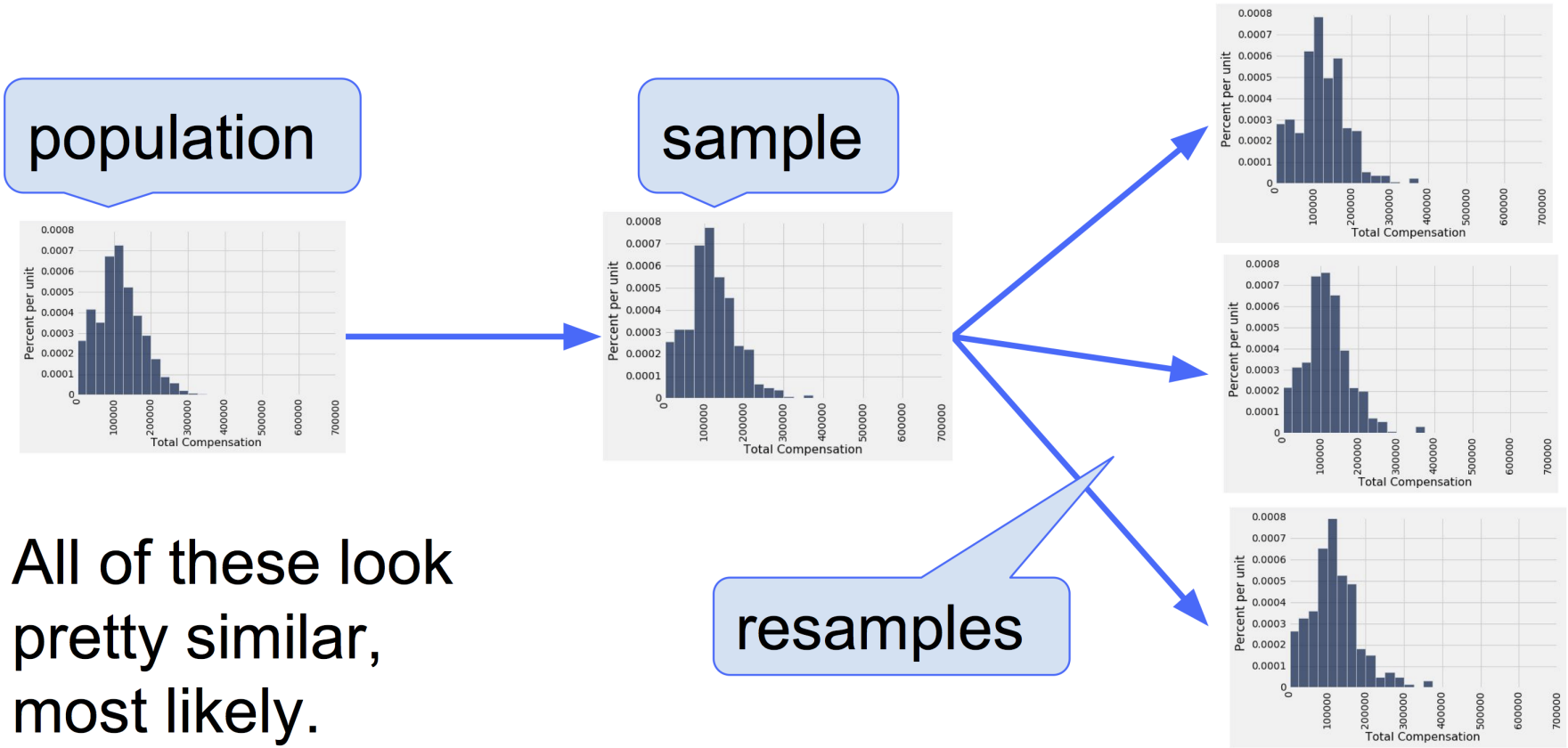



```
est_median = percentile(50,
      our_sample.column('Total Compensation'))
```

108483.1

Bootstrap: Resampling from the Sample

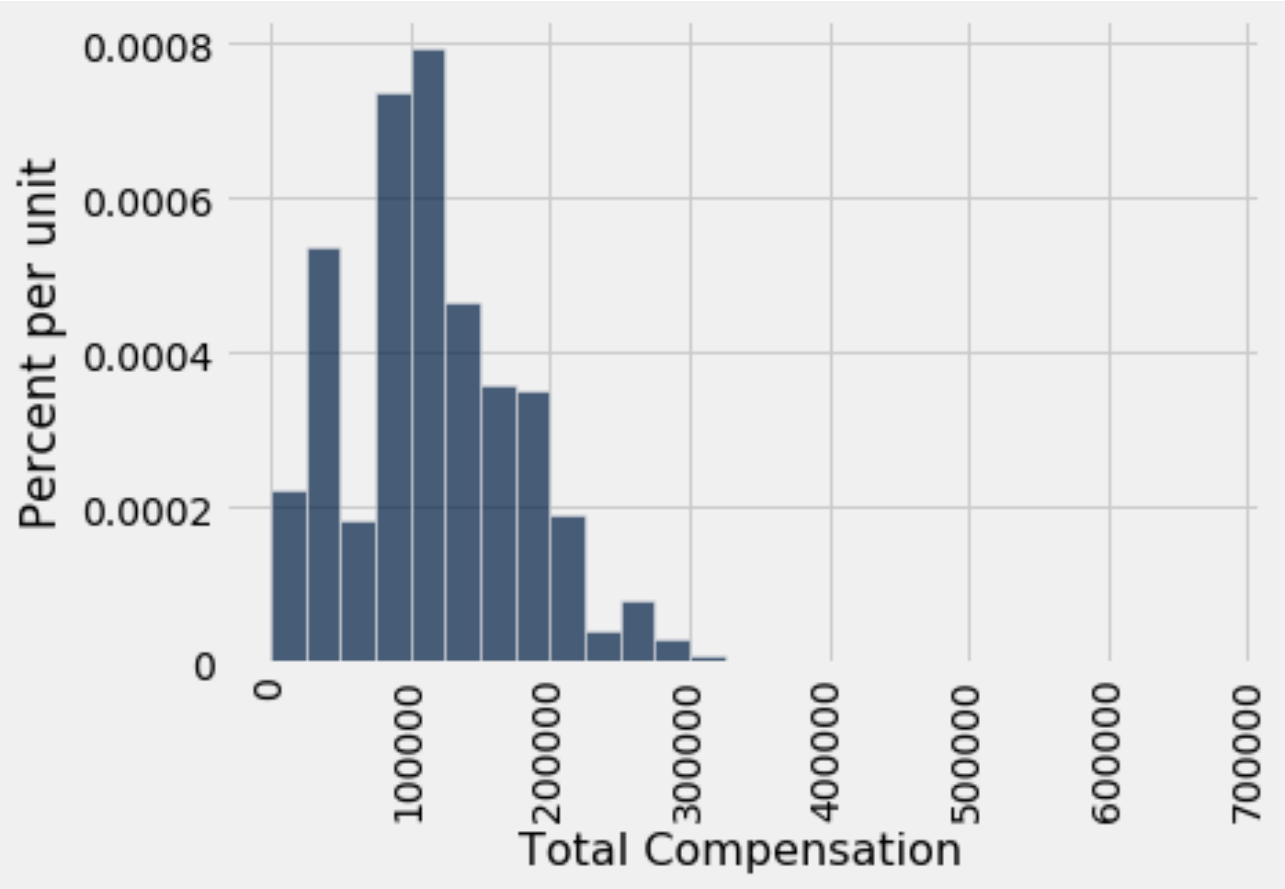
Treat the original sample as if it were the population



All of these look pretty similar, most likely.

A Resampled Median

```
resample_1 = our_sample.sample()
resample_1.select('Total Compensation').hist(bins=sf_bins)
```



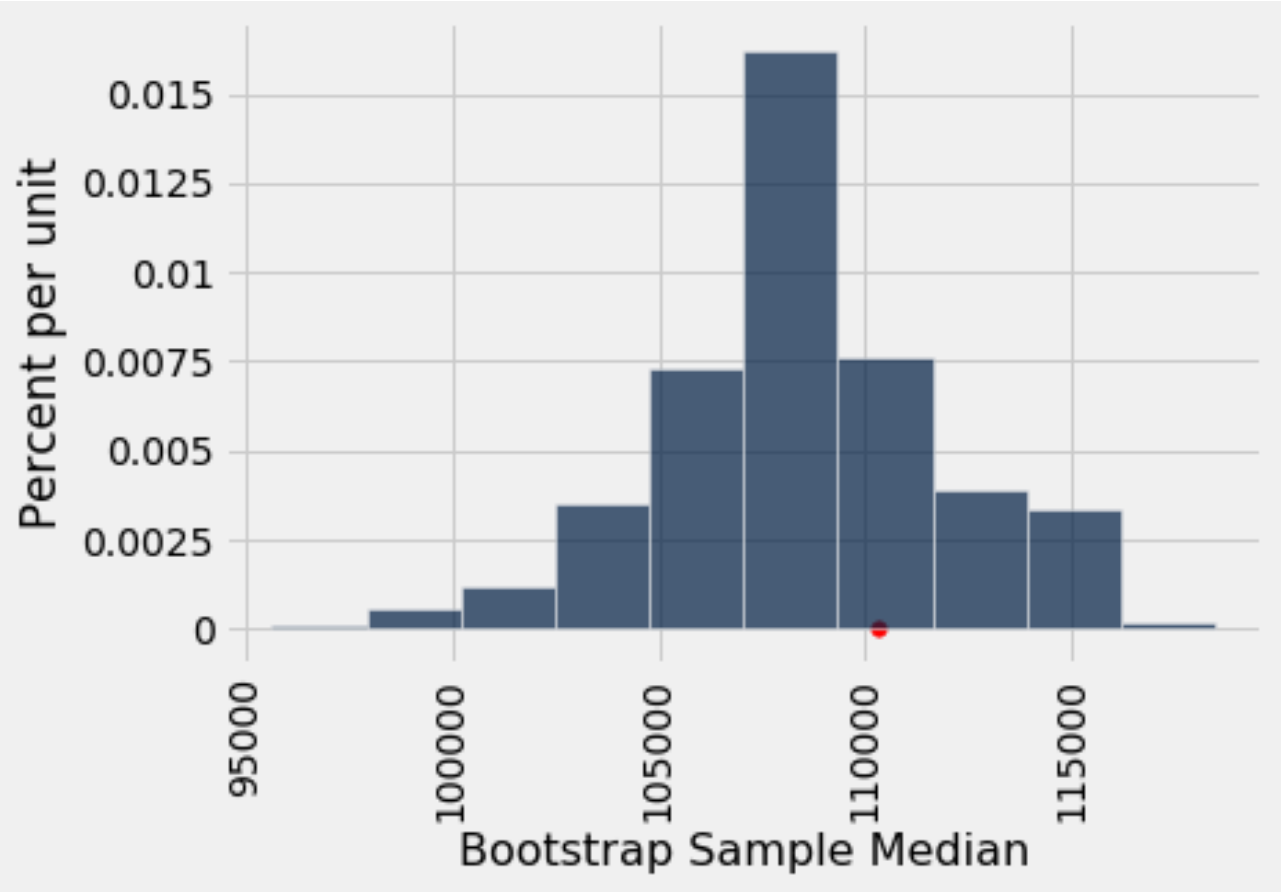

```
resampled_median_1 = percentile(50,
      resample_1.column('Total Compensation'))
```

112291.11

Bootstrap Function

```
def bootstrap_median(original_sample, label, replications):
    """Returns an array of bootstrapped sample medians:
    original_sample: table containing the original sample
    label: label of column containing the variable
    replications: number of bootstrap samples
    """
    just_one_column = original_sample.select(label)
    medians = make_array()
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample()
        resampled_median = percentile(50, bootstrap_sample.column(0))
        medians = np.append(medians, resampled_median)
    return medians
```

```
bstrap_medians = bootstrap_median(our_sample,
                  'Total Compensation', 5000)
resampled_medians = Table().with_column('Bootstrap Sample Median',
                                        bstrap_medians)
resampled_medians.hist()
plots.scatter(pop_median, 0, color='red', s=30);
```



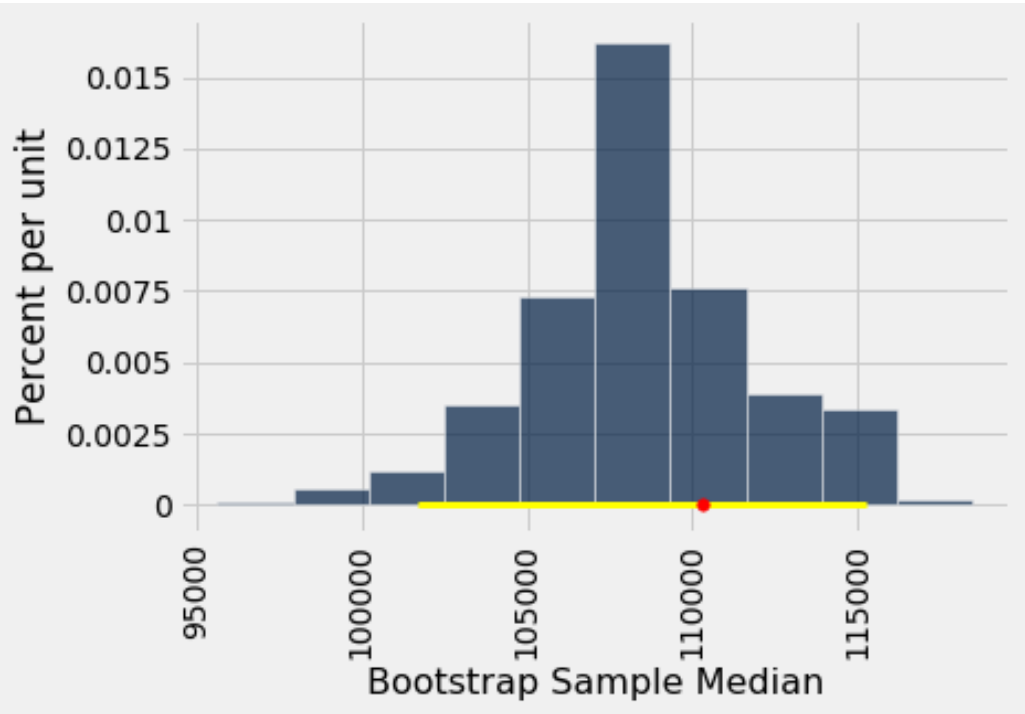
```
left = percentile(2.5, bstrap_medians)
```

101677.14

```
right = percentile(97.5, bstrap_medians)
```

115292.16

```
resampled_medians.hist()
plots.plot(make_array(left, right), make_array(0, 0),
           color='yellow', lw=3, zorder=1)
plots.scatter(pop_median, 0, color='red', s=30, zorder=2);
```



BIG SIMULATION: it takes several minutes

```
left_ends = make_array()
right_ends = make_array()
total_comps = sf2015.select('Total Compensation')
for i in np.arange(100):
    first_sample = total_comps.sample(500, with_replacement=False)
    medians = bootstrap_median(first_sample, 'Total Compensation', 5000)
    left_ends = np.append(left_ends, percentile(2.5, medians))
    right_ends = np.append(right_ends, percentile(97.5, medians))

intervals = Table().with_columns(
    'Left', left_ends,
    'Right', right_ends
)
```

intervals

Left	Right
104488	113958
99340.6	112916
104888	114684
102584	113013

```
pop_median                110305.79
```

```
intervals.where('Left', are.below(pop_median)).where('Right',
                        are.above(pop_median)).num_rows
```

94

Code for Plotting the Graphic

```
replication_number = np.ndarray.astype(np.arange(1, 101), str)
intervals2 = Table(replication_number).with_rows(make_array(left_ends,
                                                            right_ends))

plots.figure(figsize=(8,8))
for i in np.arange(100):
    ends = intervals2.column(i)
    plots.plot(ends, make_array(i+1, i+1), color='gold')
plots.plot(make_array(pop_median, pop_median),
           make_array(0, 100), color='red', lw=2)
plots.xlabel('Median (dollars)')
plots.ylabel('Replication')
plots.title('Population Median and Intervals of Estimates');
```

Confidence Intervals