Name: Krishna GSVV
Roll no. AV.EN.U4CSE22016

In [131…
```
path_data = './Datasets/'

import numpy as np
import pandas as pd

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

import warnings
warnings.filterwarnings('ignore')
```

Percentiles

A Numerical Example

In [132…
```
sizes = np.array([12, 17, 6, 9, 7])
```

In [133…
```
np.sort(sizes)
```

Out[133…
```
array([ 6,  7,  9, 12, 17])
```

In [134…
```
np.percentile(sizes, 70, interpolation='nearest')
```

Out[134…
```
np.int64(12)
```

Example

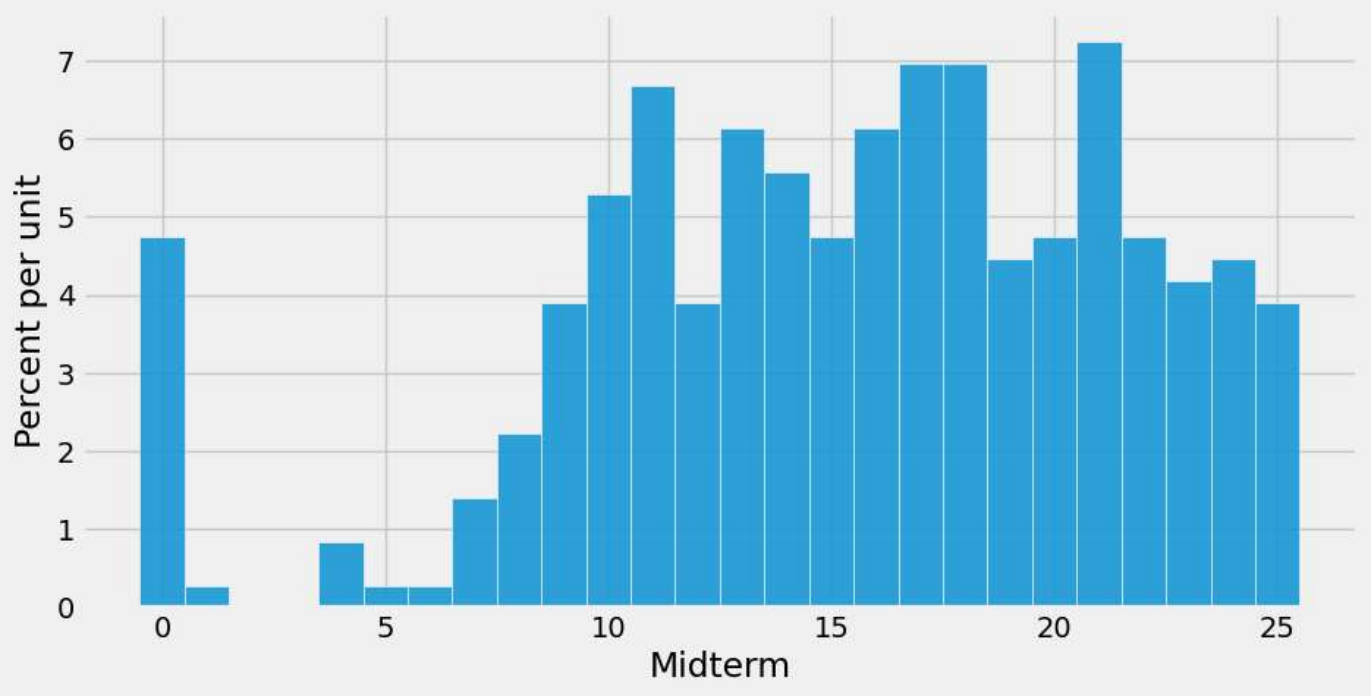
The table `scores_and_sections` contains one row for each student in a class of 359 students. The columns are the student's discussion section and midterm score.

In [135…
```
scores_and_sections = pd.read_csv(path_data + 'scores_by_section.csv')
scores_and_sections.head(10)
```

Out[135…

	Unnamed: 0	Section	Midterm
0	0	1	22
1	1	2	12
2	2	2	23
3	3	2	14
4	4	1	20
5	5	3	25
6	6	4	19
7	7	1	24
8	8	5	8
9	9	6	14

In [136…
```
unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(scores_and_sections['Midterm'], bins = np.arange(-0.5, 25.6, 1), density=True, alpha=

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Midterm'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.title('');

plt.show()
```



In [137…
```
scores = scores_and_sections.iloc[:,1]

#or

#scores = scores_and_sections['Midterm']
```

In [138…
```
np.percentile(scores , 85, interpolation='nearest')
```

Out[138… np.int64(11)

In [139…
```
sorted_scores = np.sort(scores_and_sections.iloc[:,1])
```

In [140…
```
0.85 * 359
```

Out[140… 305.15

In [141…
```
# The 306th element of the sorted array

sorted_scores.item(305)
```

Out[141… 11

In [142…
```
np.percentile(scores, 25, interpolation='nearest')
```

Out[142… np.int64(3)

In [143…
```
np.percentile(scores, 50, interpolation='nearest')
```

Out[143… np.int64(6)

In [144…
```
np.percentile(scores, 75, interpolation='nearest')
```

Out[144… np.int64(9)

The Bootstrap

Employee Compensation in the City of San Francisco

In [145…
```
sf2015 = pd.read_csv(r".\Datasets\san_francisco_2015.csv")
```

In [146…
```
sf2015.head(10)
```

Out[146…

	Unnamed: 0	Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	
0	0	Calendar	2015	2	Public Works, Transportation & Commerce	WTR	PUC Water Department	21.0	Prof Engi Miscella L
1	1	Calendar	2015	2	Public Works, Transportation & Commerce	DPW	General Services Agency - Public Works	12.0	Linole S Worker
2	2	Calendar	2015	4	Community Health	DPH	Public Health	790.0	Miscella Loc
3	3	Calendar	2015	4	Community Health	DPH	Public Health	351.0	Mu Ex Assoc Miscell
4	4	Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	790.0	Miscella Loc
5	5	Calendar	2015	1	Public Protection	POL	Police	911.0	C Assc
6	6	Calendar	2015	4	Community Health	DPH	Public Health	791.0	SEIU and Pe Nurse
7	7	Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	253.0	Tra Wo Op L
8	8	Calendar	2015	6	General Administration & Finance	CAT	City Attorney	311.0	Mu Att Assc
9	9	Calendar	2015	3	Human Welfare & Neighborhood Development	DSS	Human Services	535.0	SEIU - Service

10 rows × 23 columns

In [147…

```
sf2015[sf2015['Job'] == 'Mayor']
```

Out[147…

	Unnamed: 0	Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	Uni
3335	3335	Calendar	2015	6	General Administration & Finance	MYR	Mayor	556.0	Elect Offici

1 rows × 23 columns

In [148…

```
sf2015.sort_values(by=['Total Compensation'])
```

Out[148…

	Unnamed: 0	Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	
27308	27308	Calendar	2015	1	Public Protection	FIR	Fire Department	798.0	Fi Mis
15746	15746	Calendar	2015	4	Community Health	DPH	Public Health	790.0	Mis
24576	24576	Calendar	2015	1	Public Protection	JUV	Juvenile Probation	790.0	Mis
42982	42982	Calendar	2015	6	General Administration & Finance	CPC	City Planning	21.0	Mis
23310	23310	Calendar	2015	6	General Administration & Finance	CPC	City Planning	21.0	Mis
…	…	…	…	…	…	…	…	…	
5171	5171	Calendar	2015	4	Community Health	DPH	Public Health	351.0	As Mis
17805	17805	Calendar	2015	2	Public Works, Transportation & Commerce	AIR	Airport Commission	351.0	As Mis
499	499	Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	164.0	Phy Mis
13194	13194	Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	164.0	Phy Mis
19177	19177	Calendar	2015	6	General Administration & Finance	RET	Retirement System	351.0	As Mis

42989 rows × 23 columns

In [149…
```
sf2015 = sf2015[sf2015['Salaries'] > 10000]
```

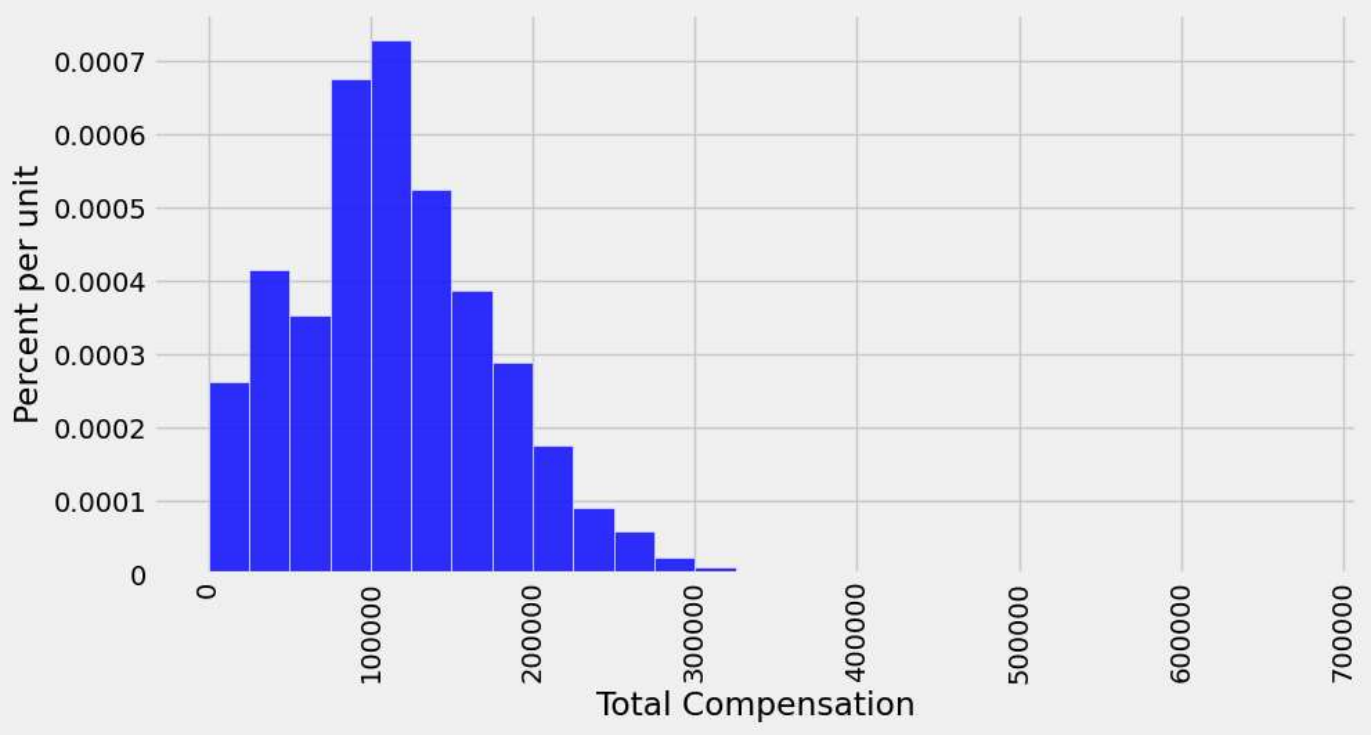
In [150…
```
len(sf2015)
```

Out[150… 36569

Population and Parameter

In [151…

```
sf_bins = np.arange(0, 700000, 25000)

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(sf2015['Total Compensation'], bins = sf_bins, density=True, color='blue', alpha=0.8,

#ax.scatter(observed_distance, 0, color='red', s=40, zorder=10).set_clip_on(False)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Total Compensation'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [152…

```
sf2015.sort_values(by=['Total Compensation'], ascending=False).head(2)
```

Out[152…

	Unnamed: 0	Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	
19177	19177	Calendar	2015	6	General Administration & Finance	RET	Retirement System	351.0	As Mis
13194	13194	Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	164.0	and Mis

2 rows × 23 columns

In [153…

```
pop_median = np.percentile(sf2015['Total Compensation'], 50)
pop_median
```

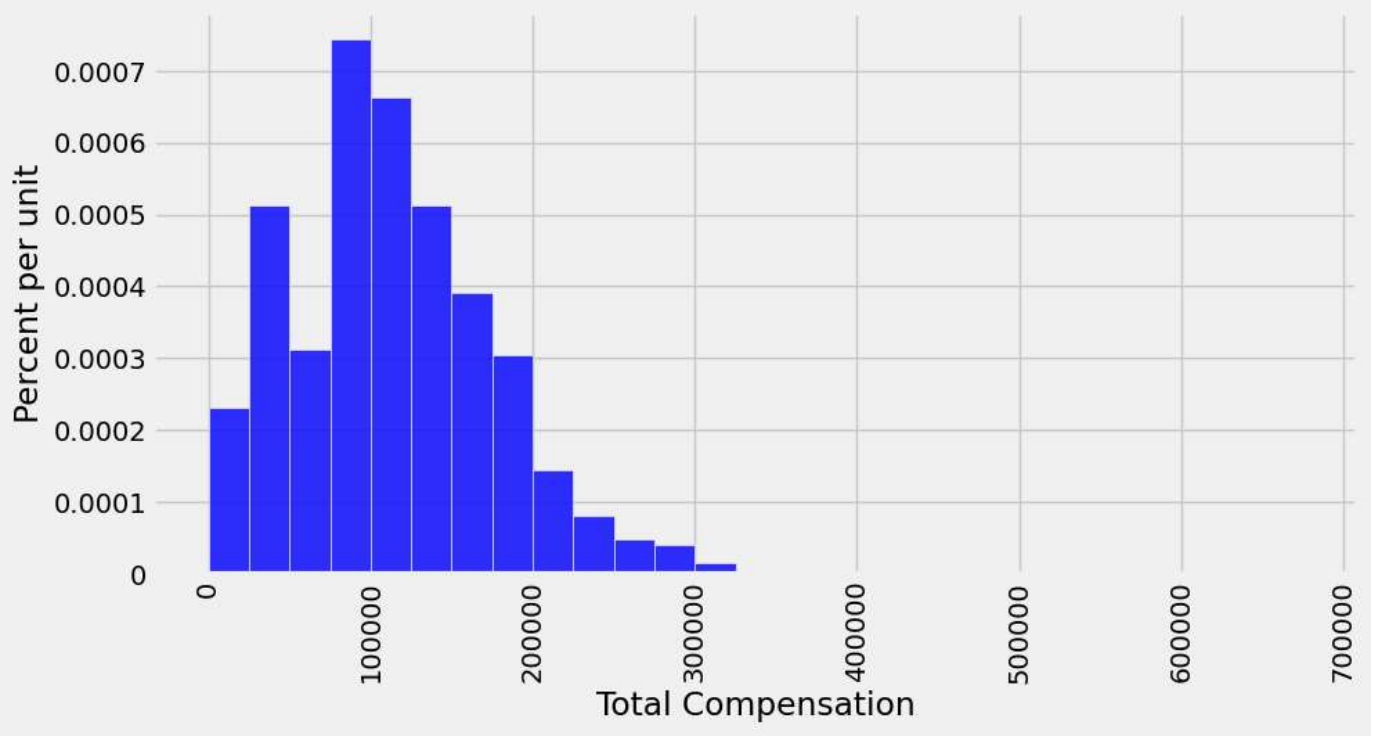
Out[153…

```
np.float64(110305.79)
```

A Random Sample and an Estimate

In [154…

```
our_sample = sf2015.sample(500, replace=False)

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(our_sample['Total Compensation'], bins = sf_bins, density=True, color='blue', alpha=0

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Total Compensation'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [155…
```
est_median = np.percentile(our_sample['Total Compensation'], 50)
est_median
```

Out[155…
```
np.float64(105973.76)
```

The Bootstrap: Resampling from the Sample

- **Treat the original sample as if it were the population.**
- **Draw from the sample**, at random **with** replacement, **the same number of times as the original sample size.**

A Resampled Median

In [156…
```
resample_1 = our_sample.sample(len(our_sample), replace=True)
resample_1
```

Out[156…

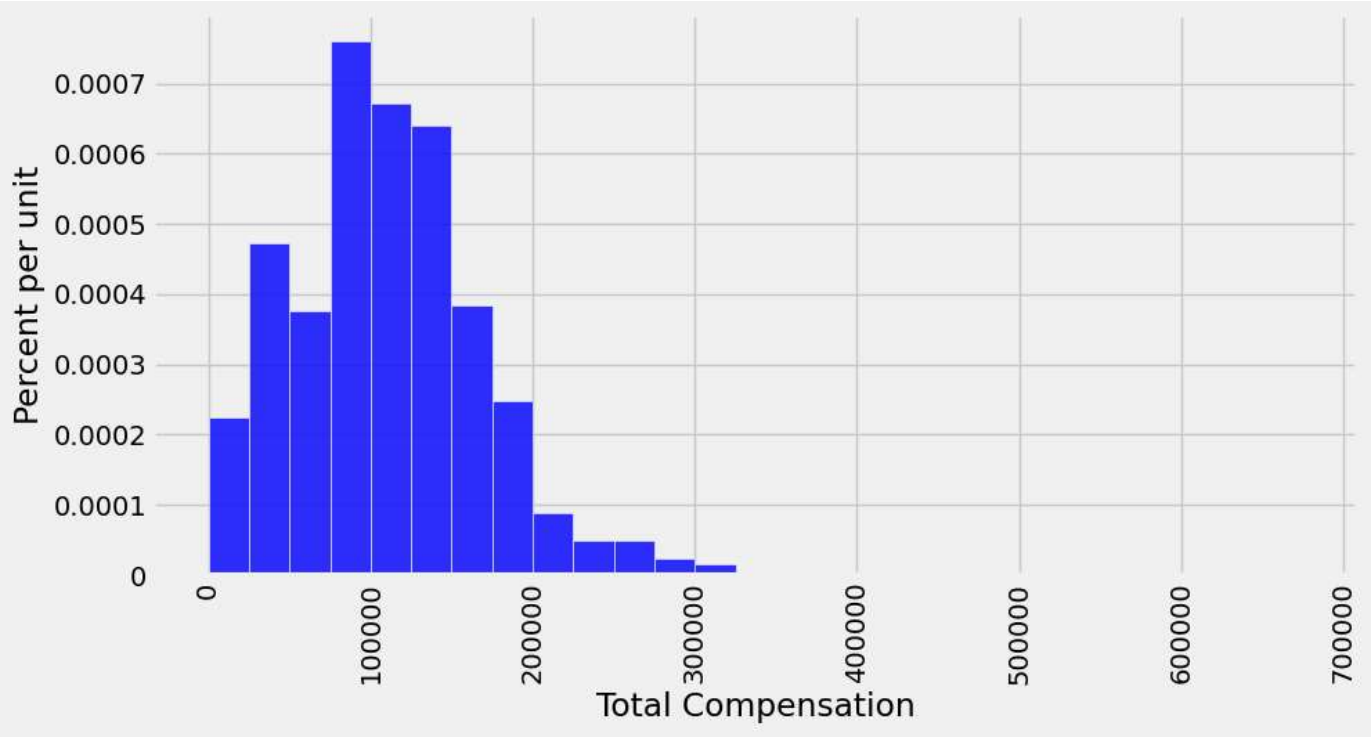
	Unnamed: 0	Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	
17781	17781	Calendar	2015	2	Public Works, Transportation & Commerce	CWP	PUC Wastewater Enterprise	39.0	
22064	22064	Calendar	2015	4	Community Health	DPH	Public Health	250.0	S W
23740	23740	Calendar	2015	2	Public Works, Transportation & Commerce	DBI	Department of Building Inspection	790.0	M
34984	34984	Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	21.0	M
30906	30906	Calendar	2015	1	Public Protection	FIR	Fire Department	798.0	M
...	...	...	...	...	...	...	...	...	
13372	13372	Calendar	2015	1	Public Protection	CRT	Superior Court	356.0	
12446	12446	Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	21.0	M
32860	32860	Calendar	2015	2	Public Works, Transportation & Commerce	PUC	PUC Public Utilities Commission	21.0	M
1734	1734	Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	261.0	
7602	7602	Calendar	2015	4	Community Health	DPH	Public Health	791.0	a

500 rows × 23 columns

In [157…

```
unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resample_1['Total Compensation'], bins = sf_bins, density=True, color='blue', alpha=0

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Total Compensation'
```

```
ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [158…
```
resampled_median_1 = np.percentile(resample_1['Total Compensation'], 50, interpolation='neare
resampled_median_1
```

Out[158…
```
np.float64(105430.57)
```

In [159…
```
resample_2 = our_sample.sample(len(our_sample), replace=True)
resampled_median_2 = np.percentile(resample_2['Total Compensation'], 50, interpolation='neare
resampled_median_2
```

Out[159…
```
np.float64(108426.22)
```

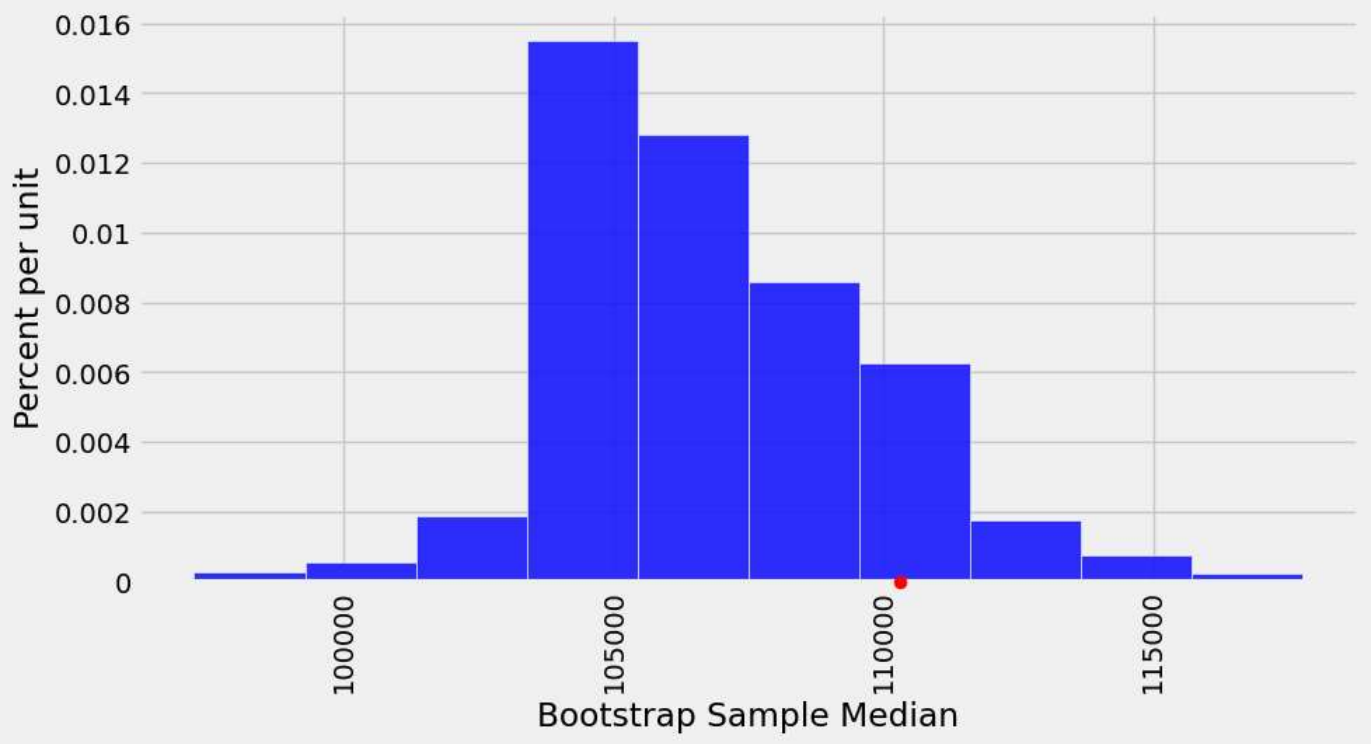
Bootstrap Empirical Distribution of the Sample Median

In [160…
```
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[label]
    medians = np.array([])
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample(len(just_one_column), replace=True)
        resampled_median = np.percentile(bootstrap_sample, 50)
        medians = np.append(medians, resampled_median)

    return medians
```

In [161…
```
bstrap_medians = bootstrap_median(our_sample, 'Total Compensation', 5000)
```

In [162…

```
resampled_medians = pd.DataFrame({'Bootstrap Sample Median':bstrap_medians})

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resampled_medians, density=True, color='blue', alpha=0.8, ec='white')

ax.scatter(pop_median, 0, color='red', s=40, zorder=10).set_clip_on(False)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Median'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



Do the Estimates Capture the Parameter?

In [163…

```
left = np.percentile(bstrap_medians, 2.5, interpolation='nearest')
left
```

Out[163… `np.float64(102238.67)`

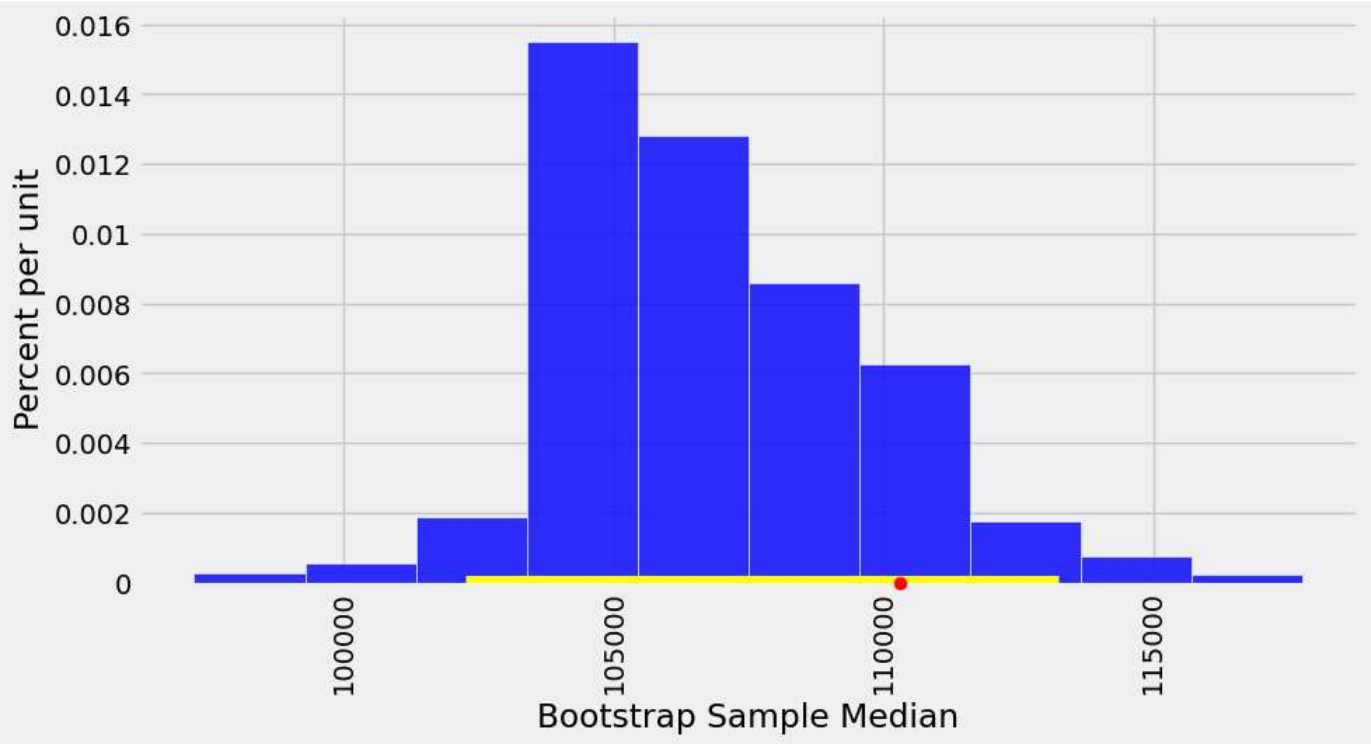
In [164…

```
right = np.percentile(bstrap_medians, 97.5, interpolation='nearest')
right
```

Out[164… `np.float64(113252.49)`

In [165…

```
resampled_medians = pd.DataFrame({'Bootstrap Sample Median':bstrap_medians})

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resampled_medians, density=True, color='blue', alpha=0.8, ec='white', zorder=5)

ax.plot(np.array([left, right]), np.array([0,0]), color='yellow', lw=8, zorder=10)

ax.scatter(pop_median, 0, color='red', s=40, zorder=15).set_clip_on(False)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Median'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [166…

```
total_comps = sf2015[['Total Compensation']]
total_comps
```

Out[166…

	Total Compensation
0	117766.86
1	41209.83
2	110561.13
3	38624.97
6	260280.95
…	…
42983	61349.71
42984	132788.81
42986	73295.59
42987	19973.37
42988	55812.90

36569 rows × 1 columns

In [167…

```
# THE BIG SIMULATION: This one takes several minutes.

# Generate 100 intervals, in the table intervals

left_ends = np.array([])
right_ends = np.array([])

total_comps = sf2015[['Total Compensation']]

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

intervals = pd.DataFrame(
    {'Left':left_ends,
    'Right':right_ends}
)
```

In [168…

```
intervals
```

Out[168…

	Left	Right
0	101777.145125	114593.075000
1	105487.095000	115284.620000
2	104100.370000	115540.285000
3	101942.560000	111409.906750
4	104084.910000	117027.420000
...	...	...
95	105587.890000	115014.020000
96	101299.100000	115383.290000
97	108037.990000	117553.785000
98	106682.810000	118278.653625
99	108822.310000	119297.730000

100 rows × 2 columns

In [169…

```
pop_median
```

Out[169…

```
np.float64(110305.79)
```

In [170…

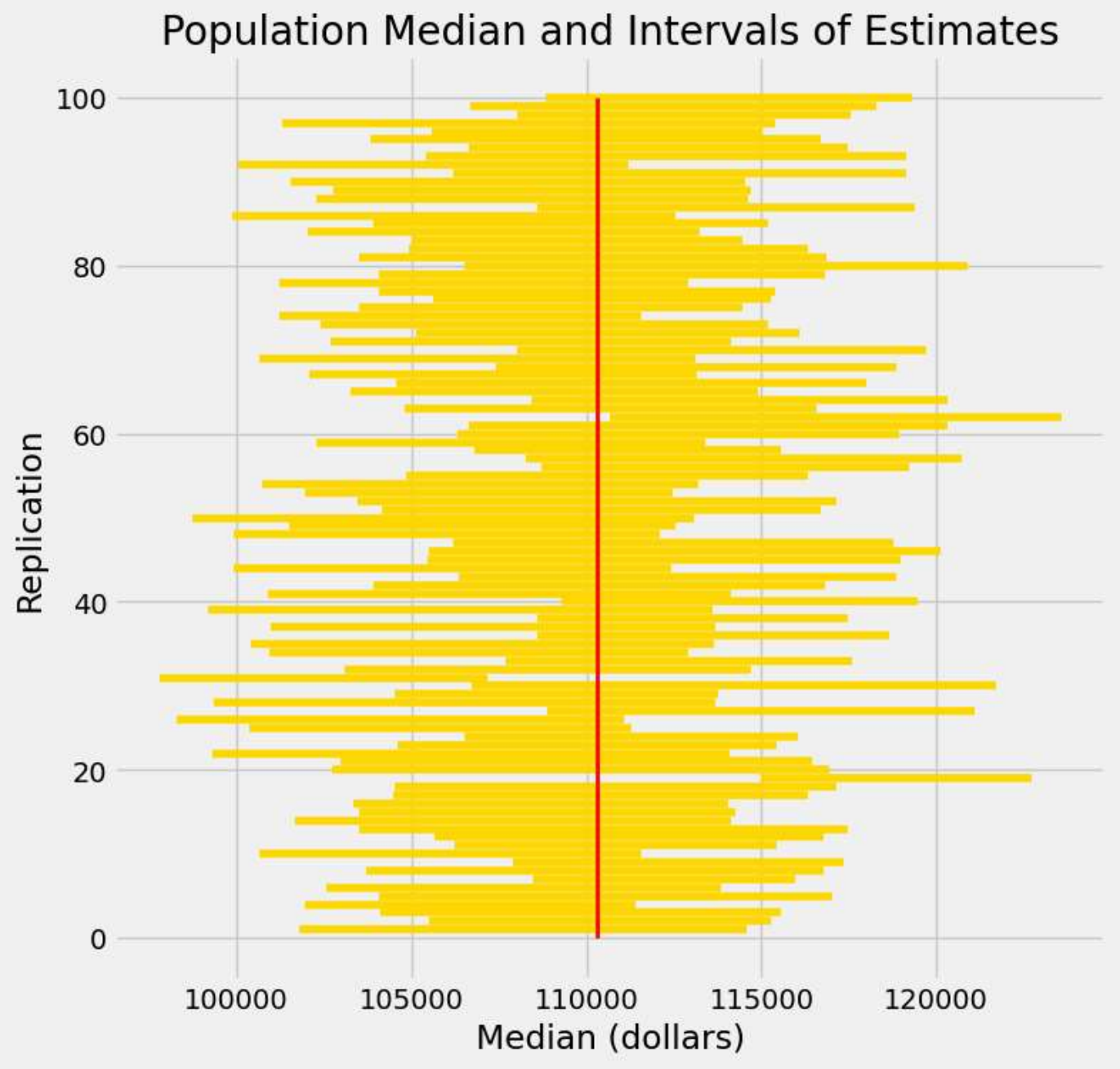
```
len(
    intervals[
        (intervals['Left'] < pop_median) &
              (intervals['Right'] > pop_median)
    ]
)
```

Out[170…

```
97
```

In [171…

```
replication_number = np.arange(1, 101)

replication_number = replication_number.astype(str)

intervals2 = pd.DataFrame(np.array([left_ends, right_ends]), columns=[replication_number])

intervals2

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



Confidence Intervals

Confidence Interval for a Population Median: Bootstrap Percentile Method

In [172…
```
baby = pd.read_csv(r".\Datasets\baby.csv")
```

In [173…
```
baby
```

Out[173…

	Unnamed: 0	Birth Weight	Gestational Days	Maternal Age	Maternal Height	Maternal Pregnancy Weight	Maternal Smoker
0	0	120	284	27	62	100	False
1	1	113	282	33	64	135	False
2	2	128	279	28	64	115	True
3	3	108	282	23	67	125	True
4	4	136	286	25	62	93	False
…	…	…	…	…	…	…	…
1169	1169	113	275	27	60	100	False
1170	1170	128	265	24	67	120	False
1171	1171	130	291	30	65	150	True
1172	1172	125	281	21	65	110	False
1173	1173	117	297	38	65	129	False

1174 rows × 7 columns

In [174…

```
ratios = baby[['Birth Weight', 'Gestational Days']]

ratios['Ratio BW/GD'] = baby['Birth Weight']/baby['Gestational Days']
```

In [175…

```
ratios
```

Out[175…

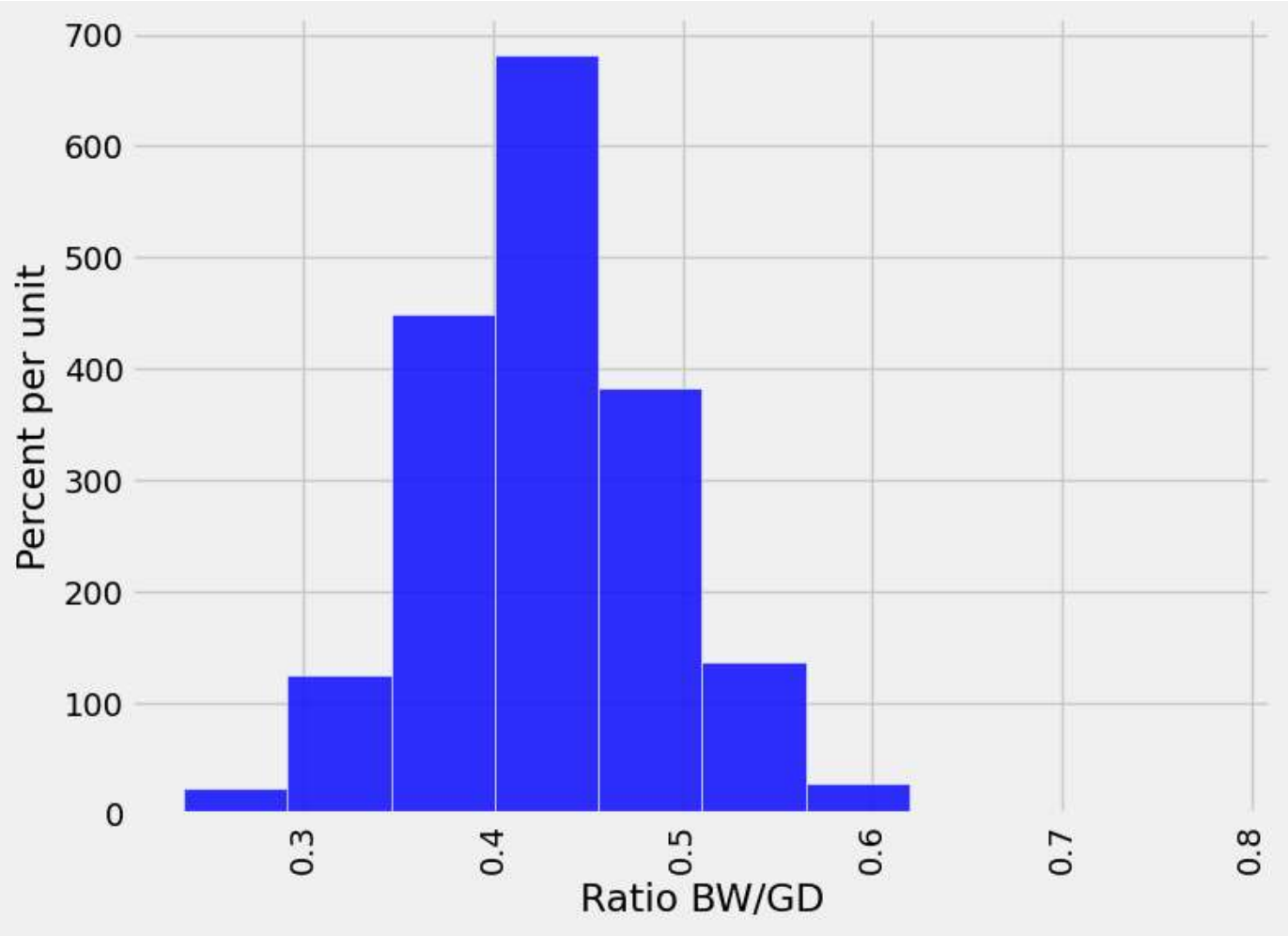
	Birth Weight	Gestational Days	Ratio BW/GD
0	120	284	0.422535
1	113	282	0.400709
2	128	279	0.458781
3	108	282	0.382979
4	136	286	0.475524
…	…	…	…
1169	113	275	0.410909
1170	128	265	0.483019
1171	130	291	0.446735
1172	125	281	0.444840
1173	117	297	0.393939

1174 rows × 3 columns

In [176…

```
unit = ''

fig, ax = plt.subplots(figsize=(8,6))

ax.hist(ratios['Ratio BW/GD'], density=True, color='blue', alpha=0.8, ec='white')

y_vals = ax.get_yticks()
```

```
y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Ratio BW/GD'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [177…
```
ratios_1 = ratios.sort_values(by=['Ratio BW/GD'], ascending=False)

ratios_1.iloc[[0]]
```

Out[177…

	Birth Weight	Gestational Days	Ratio BW/GD
238	116	148	0.783784

In [178…
```
np.median(ratios.iloc[:,2])
```

Out[178… np.float64(0.42907801418439717)

In [179…
```
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
```

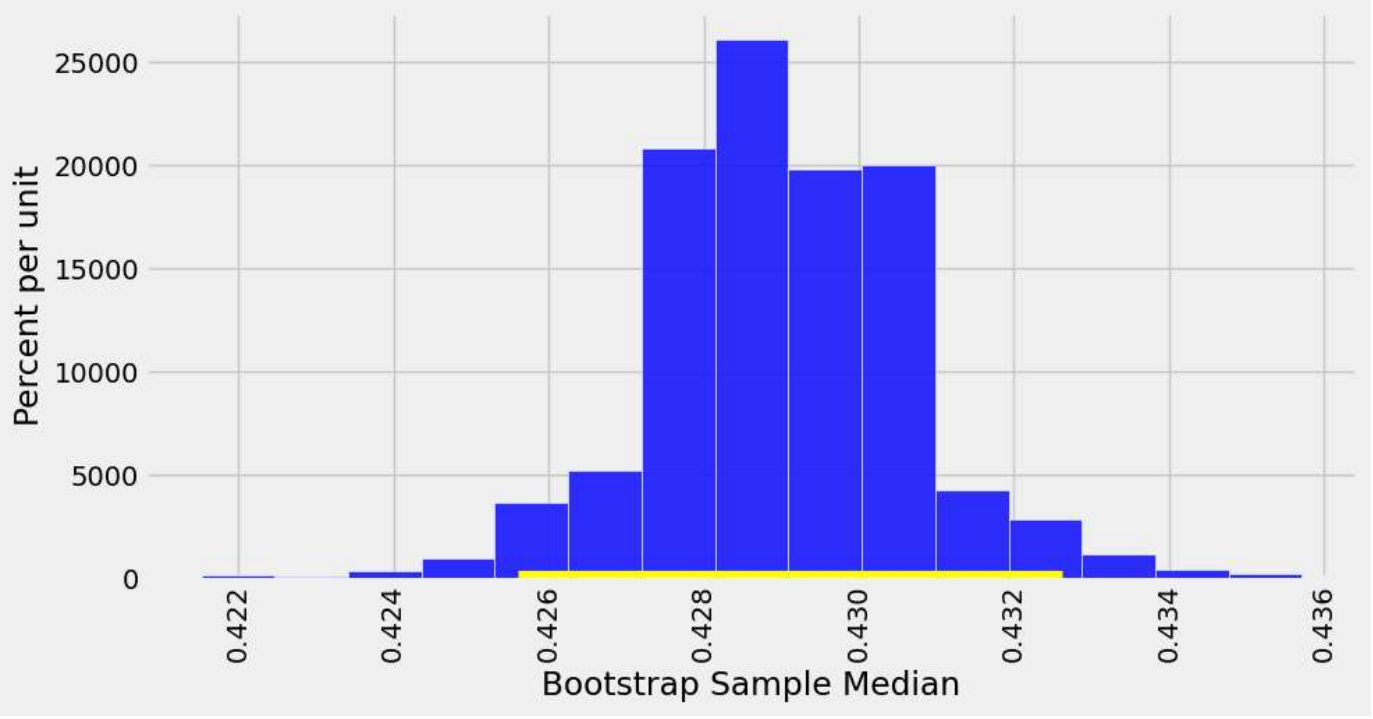
```
    replications: number of bootstrap samples
    """
    just_one_column = original_sample[[label]]
    medians = np.array([])
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample(len(just_one_column), replace=True)
        resampled_median = np.percentile(bootstrap_sample, 50)
        medians = np.append(medians, resampled_median)

    return medians
```

In [180…
```
# Generate the medians from 5000 bootstrap samples
bstrap_medians = bootstrap_median(ratios, 'Ratio BW/GD', 5000)
```

In [181…
```
# Get the endpoints of the 95% confidence interval
left = np.percentile(bstrap_medians, 2.5, interpolation='nearest')
right = np.percentile(bstrap_medians, 97.5, interpolation='nearest')

np.array([left, right])
```

Out[181…
```
array([0.42559856, 0.43262411])
```

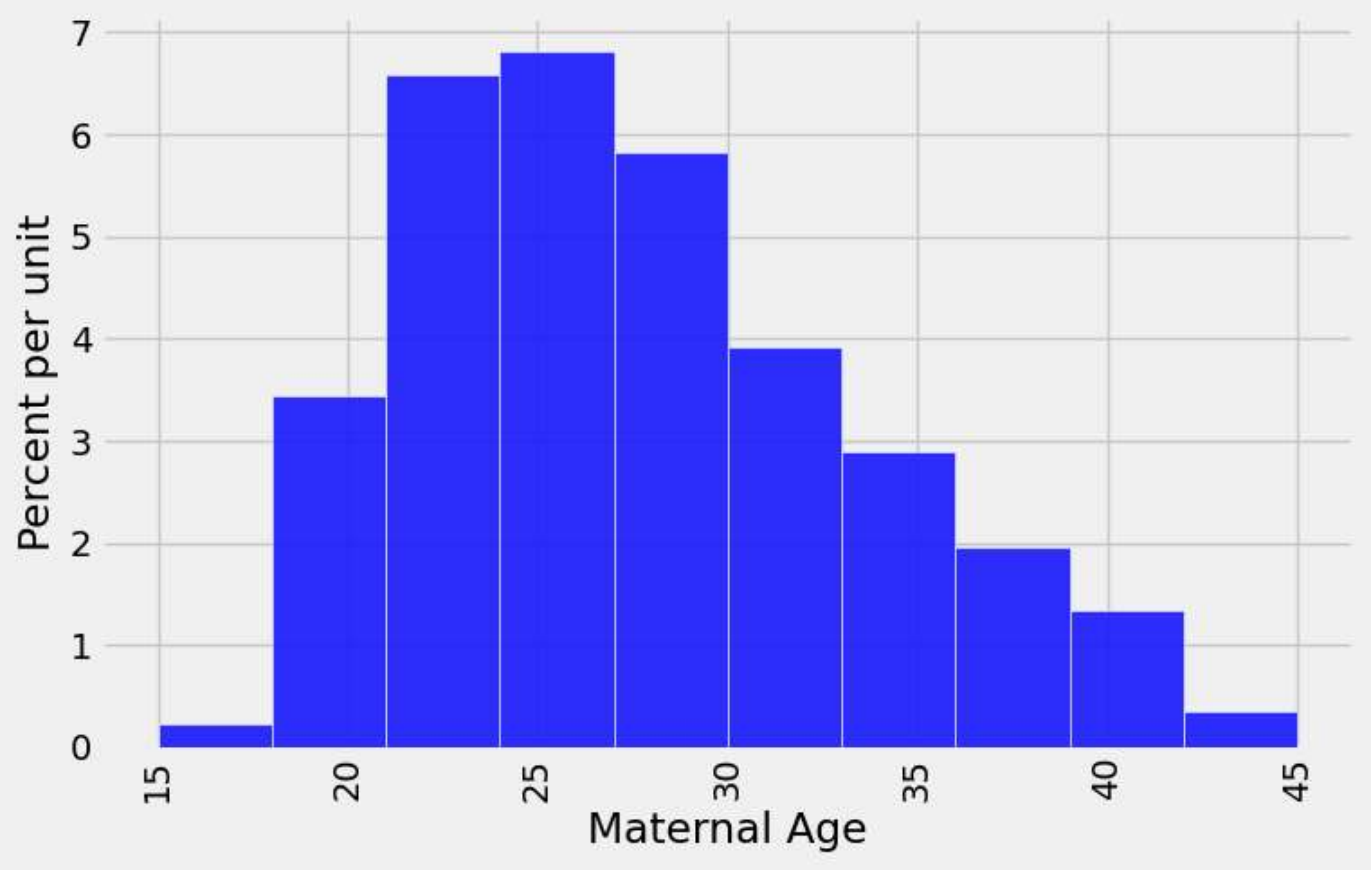
In [182…
```
resampled_medians = pd.DataFrame({'Bootstrap Sample Median':bstrap_medians})

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resampled_medians, bins=15, density=True, color='blue', alpha=0.8, ec='white', zorder

ax.plot(np.array([left, right]), np.array([0,0]), color='yellow', lw=8, zorder=10)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Median'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



Confidence Interval for a Population Mean: Bootstrap Percentile Method

In [183…

```
unit = ''

fig, ax = plt.subplots(figsize=(8,5))

ax.hist(baby['Maternal Age'], density=True, color='blue', alpha=0.8, ec='white', zorder=5)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Maternal Age'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [184…
```
np.mean(baby['Maternal Age'])
```

Out[184…
```
np.float64(27.228279386712096)
```

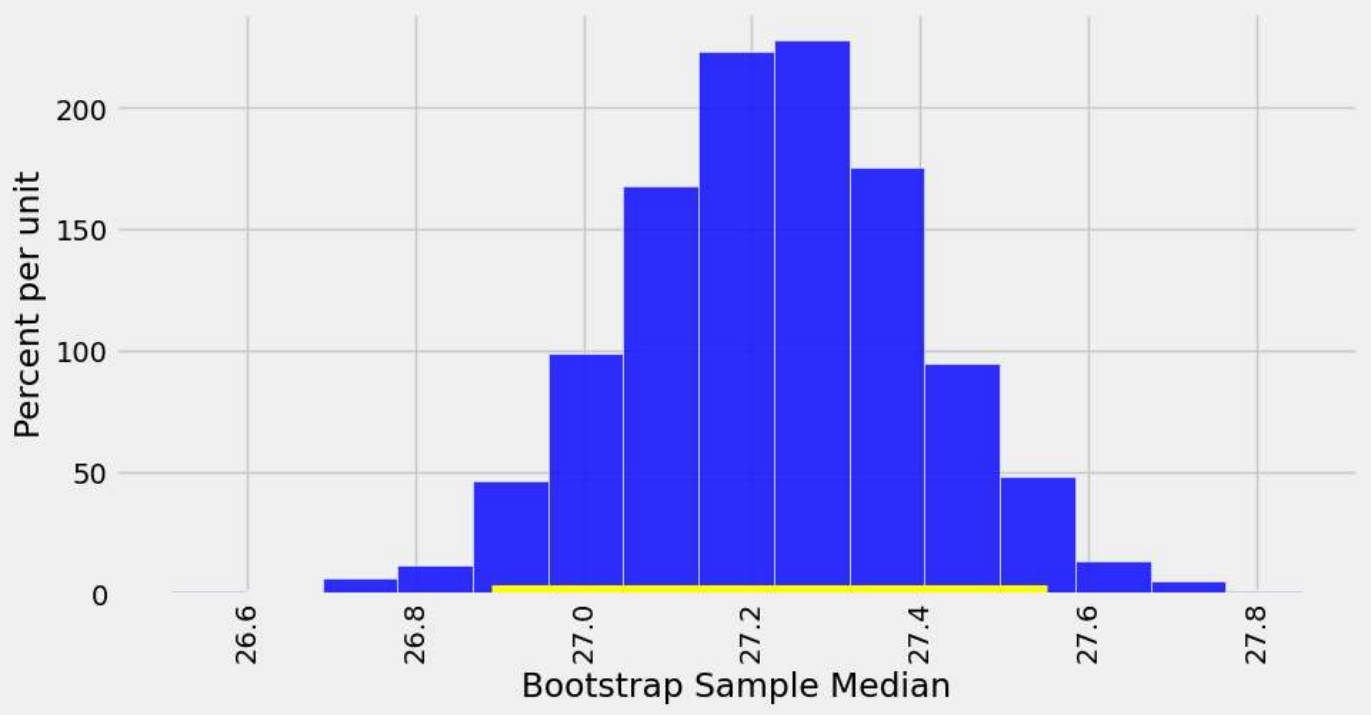
In [185…
```
def bootstrap_mean(original_sample, label, replications):

    """Returns an array of bootstrapped sample means:
    original_sample: table containing the original sample
    label: label of column containing the variable
    replications: number of bootstrap samples
    """

    just_one_column = original_sample[[label]]
    means = np.array([])
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample(len(just_one_column), replace=True)
        resampled_mean = np.mean(bootstrap_sample.iloc[:,0])
        means = np.append(means, resampled_mean)

    return means
```

In [186…
```
# Generate the means from 5000 bootstrap samples
bstrap_means = bootstrap_mean(baby, 'Maternal Age', 5000)

# Get the endpoints of the 95% confidence interval
left = np.percentile(bstrap_means, 2.5, interpolation='nearest')
right = np.percentile(bstrap_means, 97.5, interpolation='nearest')

np.array([left, right])
```

Out[186…
```
array([26.89097104, 27.55110733])
```

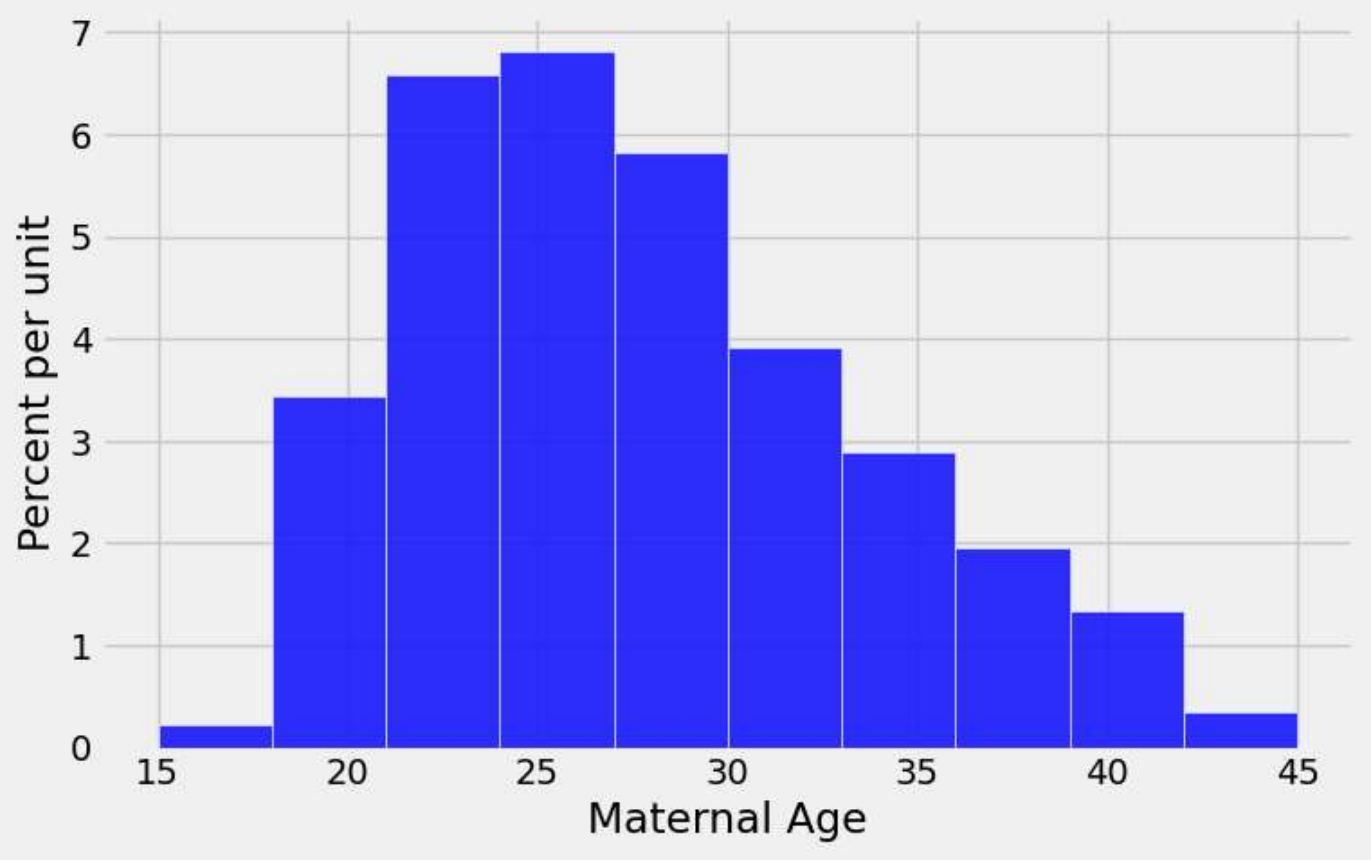
In [187…
```
resampled_means = pd.DataFrame({'Bootstrap Sample Mean':bstrap_means})

unit = ''

fig, ax = plt.subplots(figsize=(10,5))
```

```
ax.hist(resampled_means, bins=15, density=True, color='blue', alpha=0.8, ec='white', zorder=5

ax.plot(np.array([left, right]), np.array([0,0]), color='yellow', lw=8, zorder=10)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Median'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



In [188…

```
unit = ''

fig, ax = plt.subplots(figsize=(8,5))

ax.hist(baby['Maternal Age'], density=True, color='blue', alpha=0.8, ec='white', zorder=5)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Maternal Age'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.title('');

plt.show()
```



An 80% Confidence Interval

In [189…

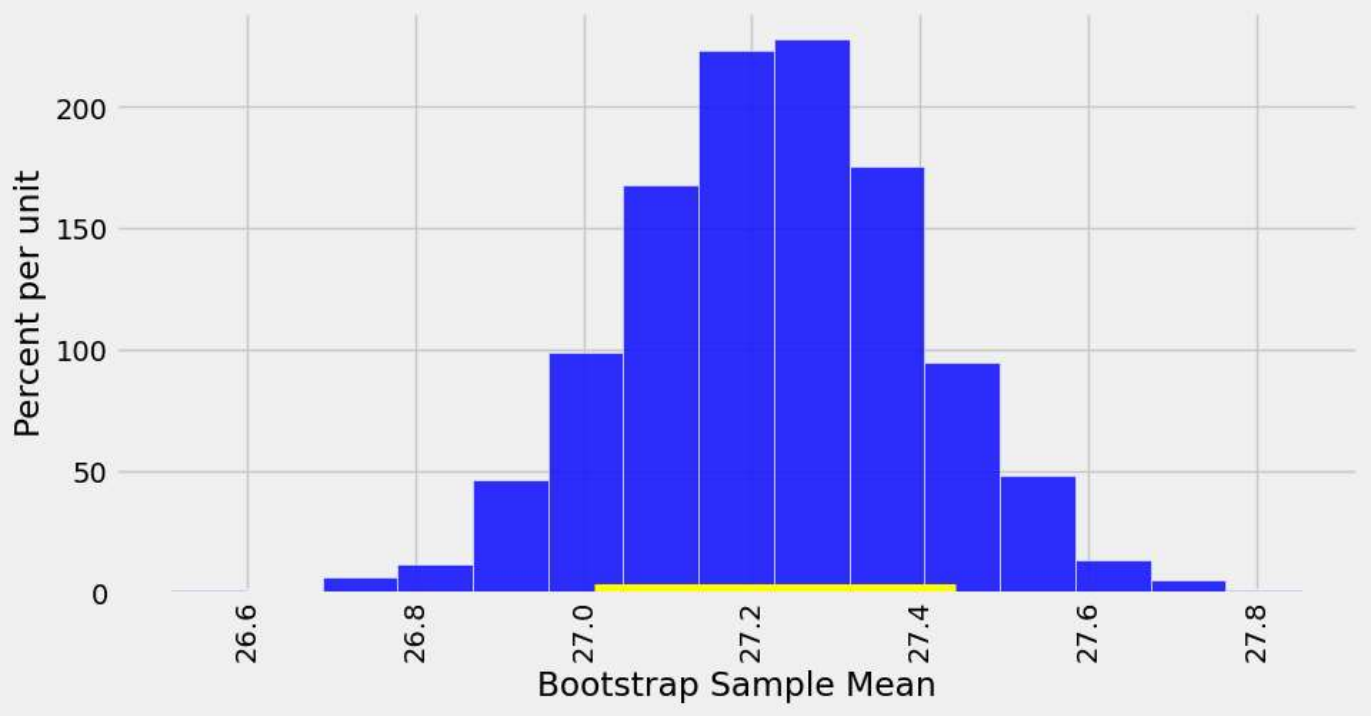
```
left_80 = np.percentile(bstrap_means, 10, interpolation='nearest')
right_80 = np.percentile(bstrap_means, 90, interpolation='nearest')
np.array([left_80, right_80])
```

Out[189…

```
array([27.01277683, 27.44122658])
```

In [190…

```
unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resampled_means, bins=15, density=True, color='blue', alpha=0.8, ec='white', zorder=5

ax.plot(np.array([left_80, right_80]), np.array([0,0]), color='yellow', lw=8, zorder=10)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Mean'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```



Confidence Interval for a Population Proportion: Bootstrap Percentile Method

In [191…
```
len(baby[baby['Maternal Smoker'] == True]) / len(baby)
```

Out[191…
```
0.3909710391822828
```

In [192…
```
smoking = baby['Maternal Smoker']
np.count_nonzero(smoking)/len(smoking)
```

Out[192…
```
0.3909710391822828
```

In [193…
```
def bootstrap_proportion(original_sample, label, replications):

    """Returns an array of bootstrapped sample proportions:
    original_sample: table containing the original sample
    label: label of column containing the Boolean variable
    replications: number of bootstrap samples
    """

    just_one_column = original_sample[[label]]
    proportions = np.array([])
    for i in np.arange(replications):
        bootstrap_sample = just_one_column.sample(len(just_one_column), replace=True)
        resample_array = bootstrap_sample.iloc[:,0]
        resampled_proportion = np.count_nonzero(resample_array)/len(resample_array)
        proportions = np.append(proportions, resampled_proportion)

    return proportions
```

In [194…
```
# Generate the proportions from 5000 bootstrap samples
bstrap_props = bootstrap_proportion(baby, 'Maternal Smoker', 5000)

# Get the endpoints of the 95% confidence interval
left = np.percentile(bstrap_props, 2.5, interpolation='nearest')
right = np.percentile(bstrap_props, 97.5, interpolation='nearest')

np.array([left, right])
```

Out[194…
```
array([0.36286201, 0.41822828])
```

In [195…

```
resampled_proportions = pd.DataFrame({'Bootstrap Sample Proportion':bstrap_props})

unit = ''

fig, ax = plt.subplots(figsize=(10,5))

ax.hist(resampled_proportions, bins=15, density=True, color='blue', alpha=0.8, ec='white', zo

ax.plot(np.array([left, right]), np.array([0,0]), color='yellow', lw=8, zorder=10)

y_vals = ax.get_yticks()

y_label = 'Percent per ' + (unit if unit else 'unit')

x_label = 'Bootstrap Sample Proportion'

ax.set_yticklabels(['{:g}'.format(x * 100) for x in y_vals])

plt.ylabel(y_label)

plt.xlabel(x_label)

plt.xticks(rotation=90)

plt.title('');

plt.show()
```