Chapter 14. Generating more samples; resampling methods

Steve Elston

2/28/2021

Introduction to Resampling

"There were others who had forced their way to the top from the lowest rung by the aid of their bootstraps." James Joyce, 'Ulysses' 1922

Resampling methods are powerful and widely used in computational statistics. By repeatedly re-sampling data some of the assumptions of classical statistical methods can be relaxed. These computationally intensive methods are largely products of the computer age. Resampling methods provide a natural way to find uncertainty when performing statistical inferences.

Resampling methods draw heavily on the central limit theorem (CLT) (Chapter XX) and the weak law of large numbers (Chapter XX). The weak law of large numbers tells us that a resampled estimate of a static converges to the correct value, when certain conditions are met. The CLT tells us that the sampling distribution of mean estimates are converge to a Normal, as the number of resamples increases.

There are a great many use cases for resampling methods. Specifically re-sampling methods:

- Estimate a probability distribution of a statistic.

- Make minimal distributional assumptions, when compared to classical frequentist statistics.
- Are computationally intensive, but often highly parallelizable.

Commonly used re-sampling methods include:

- **Randomization or Permutation methods:** for hypothesis tests.
- **Non-parametric bootstrap resampling:** to compute statistics.
- **Jackknife:** or leave one out re-sampling to compute statistics.
- **Cross validation:** resample into multiple folds without replacement to assess performance of statistical and machine learning models.

Randomization and permutation methods

Randomization and permutation methods were pioneered by Fisher as early as 1911. Fisher fully developed the theory in his 1935 book. Scalability of fully rank permutation methods remain limited, even with modern computers. But, modern methods using limited numbers of resamples have proved robust and scalable.

A null distribution is estimated by randomly permuting the response variable. The statistic is computed many times using a different permutation each time. This result represents a sampling distribution of a null model. A null distribution holds if the statistic is not statistically significant and the model is not predictive. A test statistic is then compared to the quantile of the null distribution.

Jack knife methods

Jack knife methods are often effective when there are only limited data samples. Maurice Quenouille originally suggested this method in 1949. The jack knife was fully developed by John W. Tukey, who gave the method its name, in 1958. Tukey saw that method as a simple tool useful for many purposes like a pocket knife.

The jack knife computes multiple values of a statistic by **leaving out one observation** each time. Therefore, for n observations there are n estimates of the statistic. The expected value of the statistic is then the mean of the resampled estimates.

Jack knife estimates often work surprisingly well for small samples. As a result of this and some other useful properties, jack knife methods are still in use today.

Cross-validation

Today, cross-validation is widely used in the testing of machine learning models. Cross-validation was originally proposed by Kurtz in 1948. Mosier extended the method to double cross validation in 1951. The modern method of nested or multicross-validation were introduced by Krus and Fuller in 1982.

At each resample, the cross validation algorithm evaluates a model by dividing the cases into **k folds**. For number of observations n, each fold contains n/k samples. The model parameters are estimated (model trained in machine learning terminology) using k-1 folds and then evaluated using the kth fold. This process is repeated k times. The average model performance and the variance of the performance metrics is then computed from the k cross validation estimates.

When $k = n$, cross validation is a leave one out algortihm. In this case, cross validation is similar to the jack knife algorithm.

Bootstrap

The bootstrap is an extremely general and powerful re-sampling method. In principle, the bootstrap algorithm and can provide estimates of the distributions of most any statistic. The bootstrap method was first suggested by Efron and Hinkley in 1978 and further developed by Efron in 1979. A full treatment was provided in Efron's 1980 book.

By repeatedly re-sampling the data, bootstrap methods relax some of the assumptions of classical statistical methods. For example, nonparametric bootstrap methods do not require any assumptions about a sampling distribution. In effect, bootstrap methods trade intensive computations for the mind power of the statistican.

As with other re-sampling methods, the bootstrap algorithm is computationally intensive. However, with increased computing power, use of bootstrap methods continues to expand. Further, the algorithm can be readily parallelized.

The bootstrap algorithm is the focus of this chapter.

Pitfalls of Resampling Methods

Re-sampling methods are general and powerful but, there is no magic involved! There are pitfalls one should be aware of!

Resampled estimate of a statistic can be no better that the original sample of observations allows. If a sample is biased, the re-sampled statistic estimate based on that sample will be biased. As an example consider that the bootstrap estimate of mean is an **unbiased sample estimate**, $\bar{x}$. But, there is no guarantee this estimate is unbiased with respect to the population parameter, μ.

The resampleed variance and confidence intervals can be no better than the sample distribution allows. In fact, bootstrap CIs are known to be optimistically biased. Be suspicious if the confidence intervals you compute seem too good to be true!

All resampling methods are computationally intensive. However, all of the commonly used methods are highly parallelizable. Thus, in 21st Century computing environments, resampling methods are quite scalable. But there are limits. Computing resamples statistics from very large data sets directly can be prohibitive.

Point Estimates vs. Distribution Estimates

The goal of **frequentist statistics** is to compute a **point estimate** of a statistic or parameter and **confidence interval** for the point estimate. By a point estimate, we mean a single most likely value. The confidence interval is based on the properties of some assumed sampling distribution.For example, for the difference in means, we estimate the confidence intervals by assuming a t-distribution for the sampling distribution.

Bootstrap methods are firmly in this frequentist camp. The goal is to estimate the sampling distribution using bootstrap resamples, $\hat{\mathcal{F}}^*$, of the original sample, $\hat{\mathcal{F}}$. The statistic computed from each resample, $s(\hat{\mathcal{F}}^*)$. is assumed to arise from the sampling distribution.

Rather than computing a point estimate directly, bootstrap methods compute a **bootstrap distribution** of the statistic. The bootstrap distribution is an approximation of the sampling distribution of the statistic. The concept of sampling the bootstrap distribution is shown in the figure.

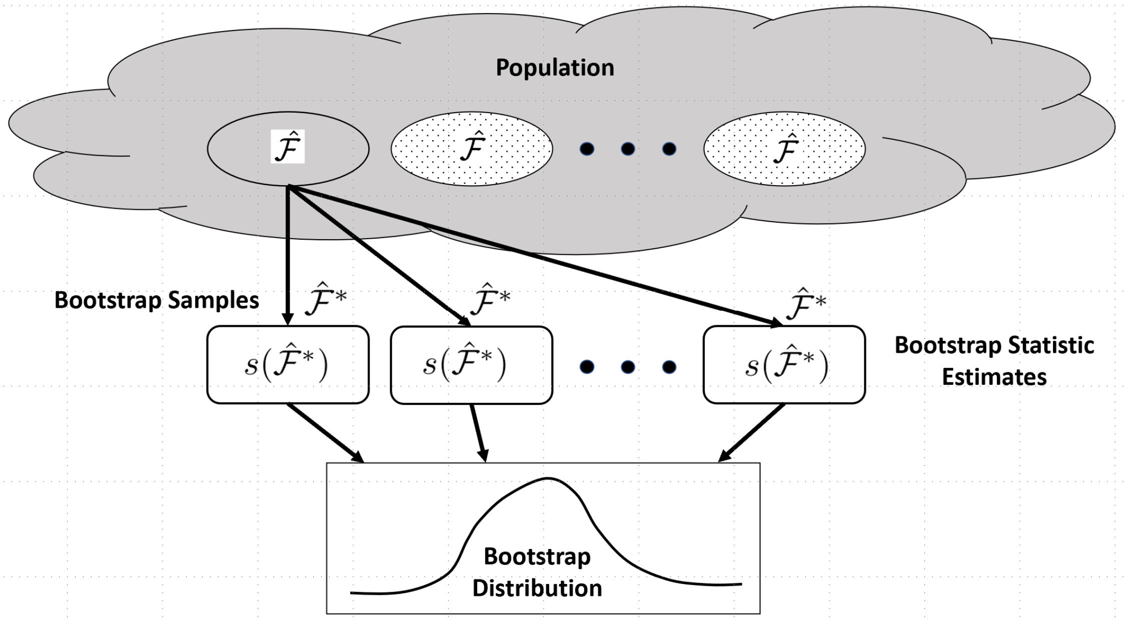


Figure 1: Resampling to estiamte the bootstrap distribution of a statistic

The bootstrap distribution is comprised of values of the statistic computed from **bootstrap samples** of the original data sample. Based on this distribution a mostly likely point estimate of the statistic, or **bootstrap estimate**, is computed as the mean of the bootstrap distribution. The **bootstrap confidence interval** is also computed from the bootstrap distribution. This approach is in contrast to the purely frequentist approach of computing point estimates and confidence intervals using an assumed sampling distribution.

Overview of the Nonparametric Bootstrap Algorithm

The nonparametric bootstrap algorithm is used to compute an estimate of the sampling distribution of most any statistic. The term **nonparametric** is applied in this case, since no assumptions about parametric sampling distributions are required. Instead, the resample estimates of the statistic are an approximation of the sampling distribution.

Bootstrap samples and the bootstrap distributon

Rather than computing a point estimate directly, bootstrap methods compute a **bootstrap distribution** of a statistic. The bootstrap distribution is comprised of values of the statistic computed from bootstrap resamples of the original observations (data sample). Computing the nonparametric bootstrap distribution requires **no assumptions about population distribution!**.

The nonparametric **bootstrap estimate** of a statistic is mostly likely point estimate of the statistic given the bootstrap distribution. As a consequence of the central limit theorem, the bootstrap estimate is the mean of the bootstrap distribution. We will address computing bootstrap confidence intervals shortly.

The nonparametric bootstrap algorithm

The nonparametric bootstrap method follows a simple algorithm. Estimates of the statistic are accumulated by these steps:

1. **Randomly sample with replacement** (e.g. Bernoulli sample) N values from an original data sample of N values. That is, the re-sample is the same size as the original data sample. This resample is called a **bootstrap sample**.

2. Re-compute the statistic with the current bootstrap sample. This is a **bootstrap estimate** of the statistic.

3. Repeat steps 1 and 2 to accumulate the required number of bootstrap estiamtes of the statistic.
4. The accumulated statistic values form the **bootstrap distribution**.

5. The mean of the computed statistic values is the **bootstrap point estimate** of the statistic.
   For example, you can compute the bootstrap mean as:

$$Meanboot = \frac{\Sigma_i mean(sample_i)}{nsample}$$

where, for example, given 10 data values, the ith bootstrap sample might be:

$$sample_i = X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8 + X_1 + X_5$$

Notice two key points about the bootstrap sample which are results of randomly sampling with replacement:
1. Some values from the orriginal sample will not be included in a bootstrap sample.
2. Some values from the orriginal sample will occur multiple times in the bootstap sample.

This nonparametric bootstrap algorithm is illustrated in the figure.

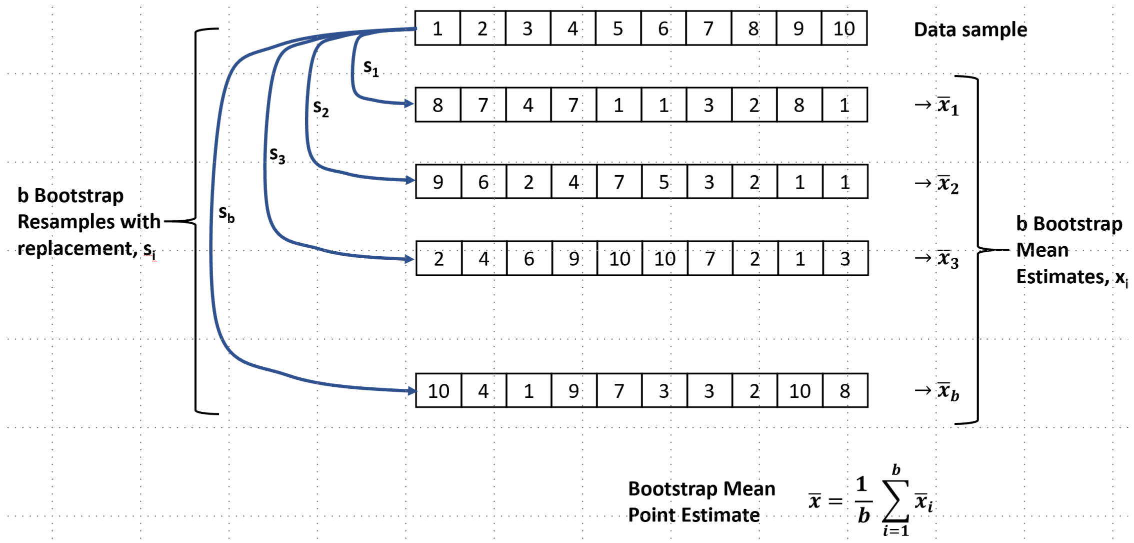


Figure 2: Outline of bootstrap resampling algorithm to compute mean

Example; single sample bootstrap

Computing a bootstrap distribution of a mean estimate is one of the simplest examples of applying the nonparametric bootstrap algorithm. A bootstrap estimates of the mean are computed a number of times from a single original sample. The result is a bootstrap distribution of the mean. Since a single original sample is resampled, this methods is know as the **one sample bootstrap algorithm**. While we focus on the mean statistic in this example, it is important to realize that this algorithm is applicable to **most any single sample statistic**.

In this section we will compute the bootstrap distribution and bootstrap mean estimates from student standardized math test scores. This data set is known as HSB2 and is a subset of a larger sample. The data were collected by the National Center for Education Statistics.

We start by examining the head of a data frame containing these data.

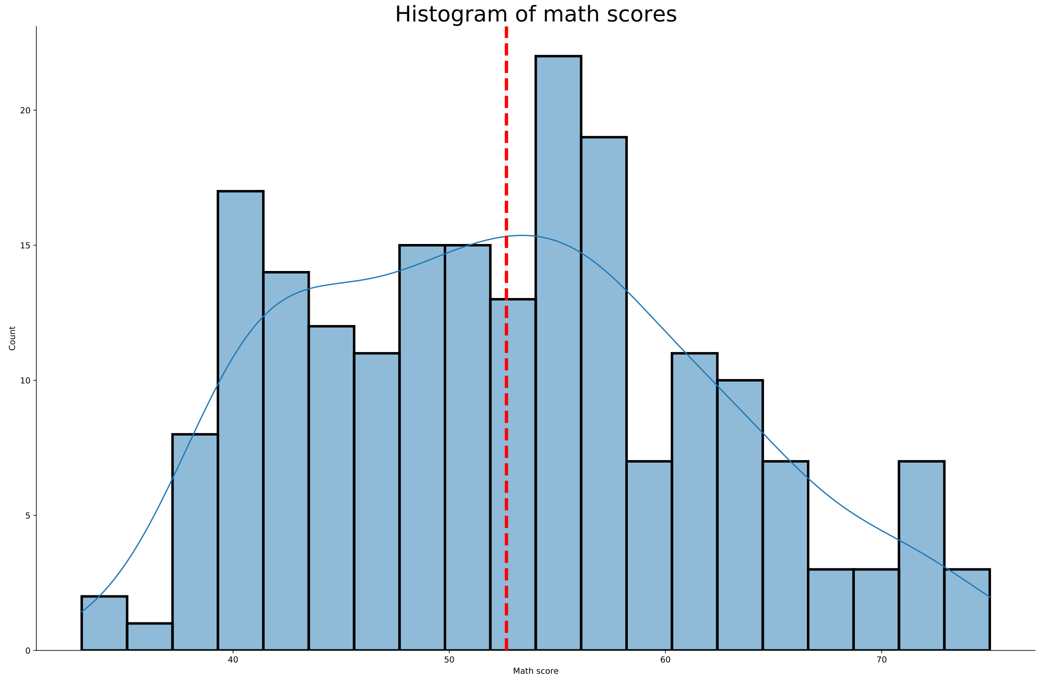
```
import pandas as pd
import numpy as np
import numpy.random as nr
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

test_scores = pd.read_csv('../data/hsb2.csv', index_col=0)
test_scores.head()
```

```
##      female  race  ses  schtyp  prog  read  write  math  science  socst
## id
## 70        0     4    1       1     1    57     52    41       47     57
## 121       1     4    2       1     3    68     59    53       63     61
## 86        0     4    3       1     1    44     33    54       58     31
## 141       0     4    3       1     3    63     44    47       53     56
## 172       0     4    2       1     2    47     52    57       53     61
```

The first three columns show the student's sex, race and socioeconomic status (SES). The next columns indicate the type of school (public or private) and the type of program the student is in (general, academic, vocational). The final columns contain the students' scores on standardized tests for five subjects.

The code in the cell below displays a histogram with kernal density estiamte of the math scores of all students in the sample.

```
## Plot the historgram of the math scores
def plot_hist(x, xlab, title, bins=20, height=15):
       sns.displot(x, bins=bins, kde=True, height=height, aspect=1.4, linewidth=3)
       plt.rcParams.update({'font.size': 22})
       plt.axvline(x=np.mean(x), color='red', linestyle='dashed', linewidth=4)
       plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.8)
       plt.xlabel(xlab)
       plt.title(title)

math = test_scores.loc[:,'math']
plot_hist(math, 'Math score', 'Histogram of math scores')
plt.show()
```



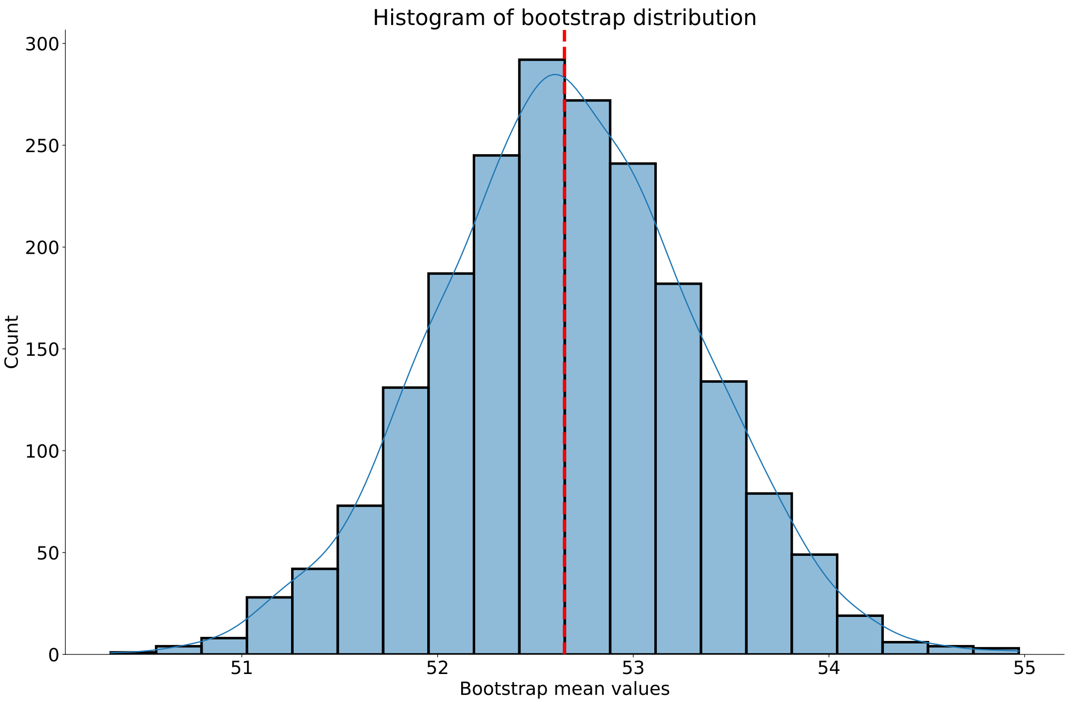
It is questionable if the distribution of math scores is Normally distributed. Fortunately, using bootstrap methods we do not need to concern ourselves with either the population or sampling distribution assumptions.

The code below generates bootstrap and bootstrap mean estimates from those samples samples. These resamples are drawn with replacement from the math scores of all students using the numpy.random.choice function. The point estimate of the mean is printed and a histogram of the bootstrap distribution is then displayed.

```
## Compute and plot the one sample bootstrap distribution of the mean
def bootstrap_statistic(x, b, statistic):
    n_samps = len(x)
    boot_vals = []
    for _ in range(b):
        boot_vals.append(statistic(nr.choice(x, size=n_samps, replace=True)))
    boot_estimate = np.mean(boot_vals)
    print('Bootstap point estimate = {:6.2f}'.format(boot_estimate))
    return(boot_estimate, boot_vals)

bootstrap_mean_estimate, boot_means = bootstrap_statistic(math, 2000, np.mean)
```

```
## Bootstap point estimate =  52.65
```

```
plot_hist(boot_means, 'Bootstrap mean values', 'Histogram of bootstrap distribution')
plt.show()
```



You can see that the bootstrap distribution of the mean estiamte is close to Normally distributed. This is a result of the CLT.

You may well wonder how many bootstrap samples should you use to estimate the bootstrap distribution? Efrom and Tibshirani (1993) and Efron and Hasti (2016) recommend using at least 200 bootstrap samples for point estimates. More recent work by several authors, including Chihara and Hesterberg (2018) indicate that more samples are desirable. With modern computers using 2,000 or more samples is considered good practice.

Exercise 14-1: In order to verify that the bootstrap distribution of the mean estimate for the math scores is nearly Normally distributed compute and display the quantile-quantial (Q-Q) plot of the bootstrap mean estimates. You can use the statsmodels.graphics.gofplots.qqplot function, with the `line=45` argument. With the excepition of a few outliers, does the bootstrap distribution appear Normally distributed?

Exercise 14-2: As has been discussed, the one sample bootstrap algorithm can be applied to most any suitable statistic. Compute the bootstrap point estiamte of the meadian of the agregate math scores from the SBS2 dataset. Display the histogram and Q-Q plots of the bootstrap distribution. Notice the deviations from the Normal distribution of this statistic. Why do you think this should be the case?

Bootstrap Confidence Intervals

Now that we can compute a bootstrap distribution the net step is to find the confidence intervals. The direct approach is to use the **percentile method**. In simple terms, the . . . .

- Use percentile method:
    1. Define confidence level; 95% or $\alpha = 0.05$
    2. Order b bootstrap samples, s_i\$, by value
    3. Lower CI index; $i = b * \alpha/2$
    4. Upper CI index; $i = b * (1 - \alpha/2)$
- Percentile method is know to be biased
    - Bias correction methods available
- Efrom and Tibshirani (1993) and Efron and Hasti (2016) recommend using at least 2,000 bootstrap samples to estimate confidence intervals

Bootstrap confidence intervals are known to be biased!

- Often bootstrap CIs are overly optimistic
- Bias can be significant for asymmetric distributions

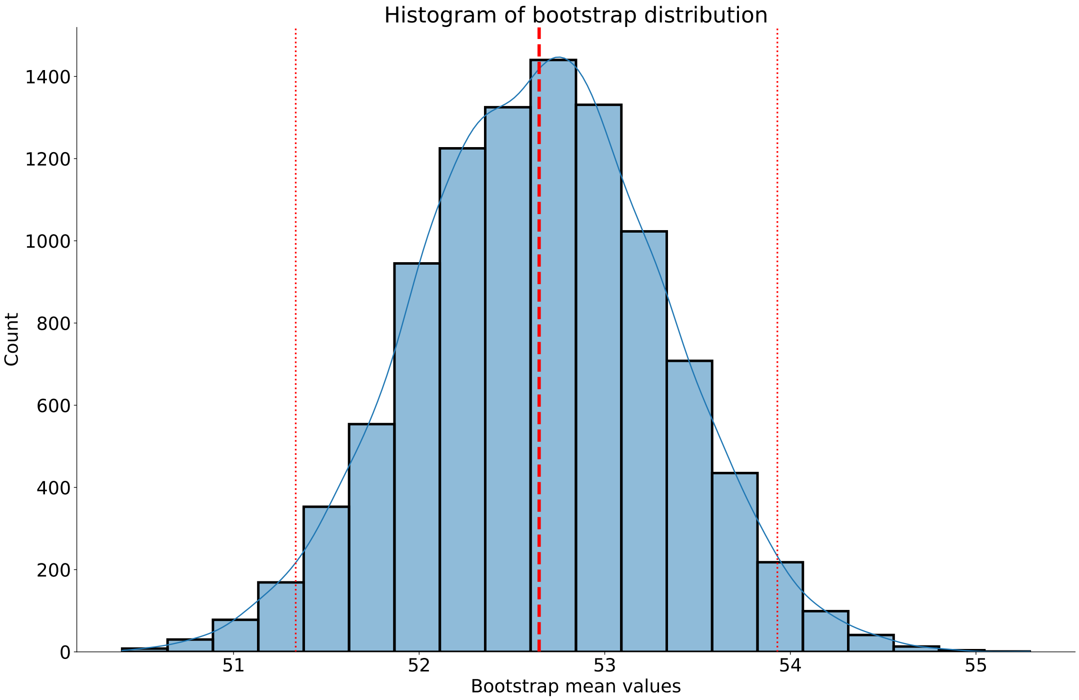
```python
## Compute and plot the one sample bootstrap of means with confidence intervals
def bootstrap_cis(boot_samples, alpha=0.05):
      n = len(boot_samples)
      sorted = np.sort(boot_samples)
      index_lci = int(n * alpha / 2)
      index_uci = int(n * (1 - alpha / 2))
      print('At alpha = {0:3.2f}, lower and upper bootstap confidence intervals = {1:6.2f}   {2:6.2f}'.
      return(sorted[index_lci], sorted[index_uci])

bootstrap_mean_estimate, boot_means = bootstrap_statistic(math, 10000, np.mean)
```

```
## Bootstap point estimate =  52.65
```

```
LCI, UCI = bootstrap_cis(boot_means)
```

```
## At alpha = 0.05, lower and upper bootstap confidence intervals =  51.34    53.93
```

```
plot_hist(boot_means, 'Bootstrap mean values', 'Histogram of bootstrap distribution')
plt.axvline(x=LCI, color='red', linestyle='dotted', linewidth=2)
plt.axvline(x=UCI, color='red', linestyle='dotted', linewidth=2)
plt.show()
```



As with the bootstrap point estimates, there is a question of how many bootstrap samples should you use to estimate the bootstrap confidence intervals? Since confidence intervals are estimated in the tails of the bootstrap distribution, more samples than the case of point estimates should be used. Efrom and Tibshirani (1993) and Efron and Hasti (2016) recommend using at least 2,000 bootstrap samples for confidence interval estimates. More recent work by several authors, including Chihara and Hesterberg (2018) indicate that more samples are desirable. With modern computers using 10,000 or more samples is considered good practice.

Two Sample Bootstrap

In the previous example you computed the means and confidence intervals of the sample distributions of the male and female adult child height data. Now you will bootstrap the difference in means to determine if it is significant. The result is the bootstrap distribution of the difference of means. The point estimate of the difference in means is then the mean of this distribution. And, the confidence intervals for this point estimate are also computed from this distribution.

How can we apply the bootstrap algorithm for two-sample statistics?

- Example, difference of means of two independently sampled populations
- How to generate boot strap samples?
- Can we just sample the concatenation of the two samples?
- **No!**
  - There is no guarantee of a correct number of resamples for each group
  - Imbalanced sampling leads to bias
- Must independently sample the two groups or populations
  - Use two independent bootstrap samples to compute statistic
  - Step one; compute statistic from independent resamples
  - Step two; compute (another) statistic from the two bootrap estimates

Two Sample Bootstrap

Example: algorithm to compute difference of means:

1. Independently randomly sample (e.g. Bernoulli sample) n data with replacement from each original data sample; The number of resamples for each populations is the number of samples for that population
2. Compute the statistic (e.g. mean) for the two resamples
3. Compute the two-sample statistic; e.g. difference of means
4. Repeat steps 1, 2, and 3 to accumulate the required number of bootstrap samples Accumulated bootstrap values form the bootstrap distribution; an estimate of the sample distribution of the statistic
5. The mean of the computed statistic values is the bootstrap point estimate of the statistic; e.g. difference of means
6. Compute CIs from bootstrap distribution

Example; two sample bootstrap

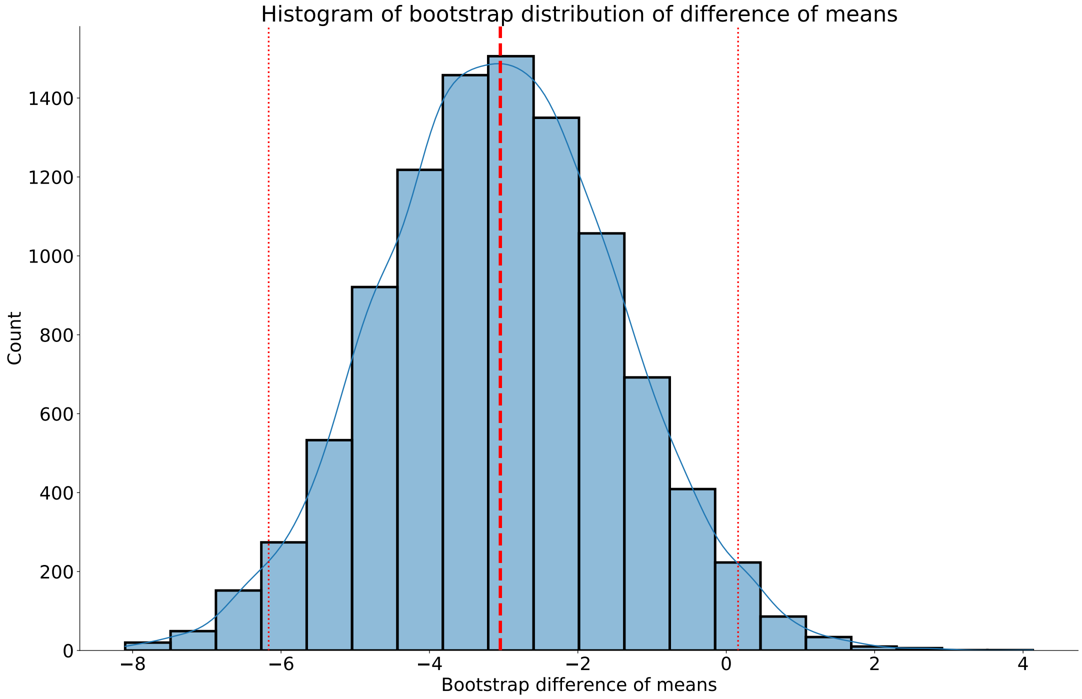
```
# Bootstrap the difference of means of low and mid SES students
def two_boot_two_stat(sample_1, sample_2, b, statistic_1, two_samp_statistic):
    two_boot_values = []
    n_samps_1 = len(sample_1)
    n_samps_2 = len(sample_2)
    for _ in range(b):
      boot_estimate_1 = statistic_1(nr.choice(sample_1, size=n_samps_1, replace=True))
      boot_estimate_2 = statistic_1(nr.choice(sample_2, size=n_samps_2, replace=True))
      two_boot_values.append(two_samp_statistic(boot_estimate_1, boot_estimate_2))
    boot_estimate = np.mean(two_boot_values)
    print('Bootstap point estimate = {:6.2f}'.format(boot_estimate))
    return(boot_estimate, two_boot_values)

math_low_ses = test_scores.loc[test_scores.loc[:,'ses']==1,'math']
math_mid_ses = test_scores.loc[test_scores.loc[:,'ses']==2,'math']
bootstrap_diff_of_mean, boot_diffs = two_boot_two_stat(math_low_ses, math_mid_ses, 10000, np.mean, lamb
```

```
## Bootstap point estimate =  -3.04
```

```
LCI, UCI = bootstrap_cis(boot_diffs)
```

```
## At alpha = 0.05, lower and upper bootstap confidence intervals =  -6.17     0.16
```

```
plot_hist(boot_diffs, 'Bootstrap difference of means', 'Histogram of bootstrap distribution of differen
plt.axvline(x=LCI, color='red', linestyle='dotted', linewidth=2)
plt.axvline(x=UCI, color='red', linestyle='dotted', linewidth=2)
plt.show()
```


Histogram of bootstrap distribution of difference of means
Count
Bootstrap difference of means