

# 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  $k$ th 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 algorithm. 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 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.

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 than 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,  $\mu$ .

The resampled 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 of the

Rather than computing a point estimate directly, bootstrap methods compute a **bootstrap distribution** of the 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** (the mean of the distribution) is computed. 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 the original data sample.

## Overview of the Bootstrap Algorithm

Bootstrap and re-sampling methods are widely applicable statistical methods. Unlike the statistical methods we have encountered before, these methods are products of the computer age. By repeatedly re-sampling the data some of the assumptions of classical statistical methods can be relaxed. The trade-off is that re-sampling methods are computationally intensive.

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

1. Randomly sample (e.g. Bernoulli sample) N data with replacement from an original data sample of N values. That is, the re-sample is the same size as the original data sample.
2. Re-compute the statistic with the current sample sample. This is a **bootstrap sample** of the statistic.
3. Repeat steps 1 and 2 to accumulate the required number of bootstrap samples.
4. 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:

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

where, for example with 10 data values and example of the ith re-sample might be:

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

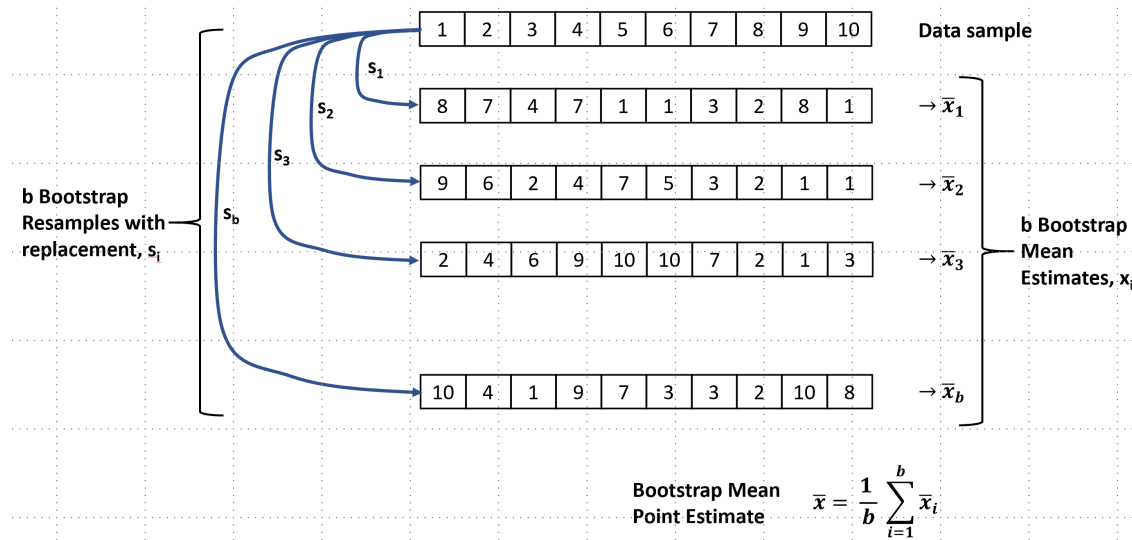


Figure 1: Outline of bootstrap resampling algorithm to compute mean

## Bootstrap Distribution

Rather than computing a point estimate directly, bootstrap methods compute a bootstrap distribution of a statistic

- Bootstrap distribution is comprised of values of the statistic computed from bootstrap resamples of the original observations (data sample)
- Computing bootstrap distribution requires **no assumptions about population distribution!**
  - Bootstrap resampling substitutes computer power for paper and pencil statistician power
- Bootstrap resampling estimates the **bootstrap distribution** of a statistic
  - Compute mostly likely point estimate of the statistic, or bootstrap estimate
  - The bootstrap confidence interval is computed from the bootstrap distribution

### Example; single 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.

```
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
```

*## Plot the histogram 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()
```



The code in the cell below generates bootstrap samples from the full male and female data sub-sets and then computes the difference in the means. 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.

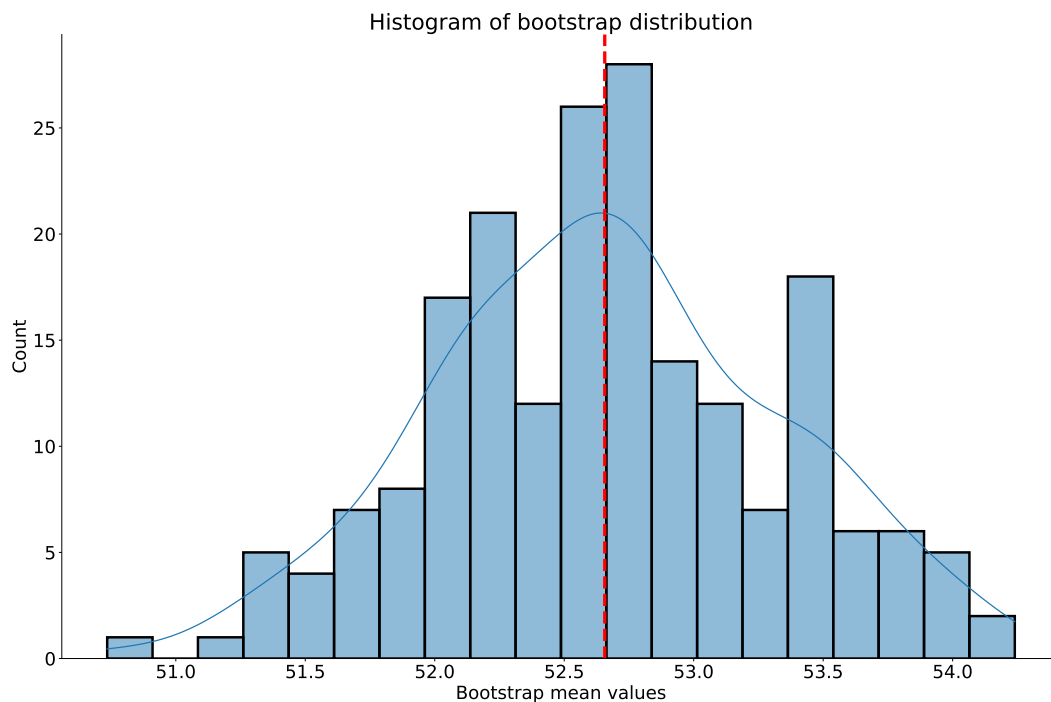
Run the code and examine the plotted results.

```
## 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 = {:.2f}'.format(boot_estimate))
    return(boot_estimate, boot_vals)

bootstrap_mean_estimate, boot_means = bootstrap_statistic(math, 200, np.mean)
```

```
## Bootstap point estimate = 52.66
```

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



## Bootstrap Confidence Intervals

Distribution of 2000 bootstrap bootstrap confidence intervals?

- 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 known to be biased
  - Bias correction methods available
- Efron 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

```
## 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 bootstrap confidence intervals = {1:6.2f} {2:6.2f}'.format(alpha, sorted[index_lci], sorted[index_uci]))
    return(sorted[index_lci], sorted[index_uci])

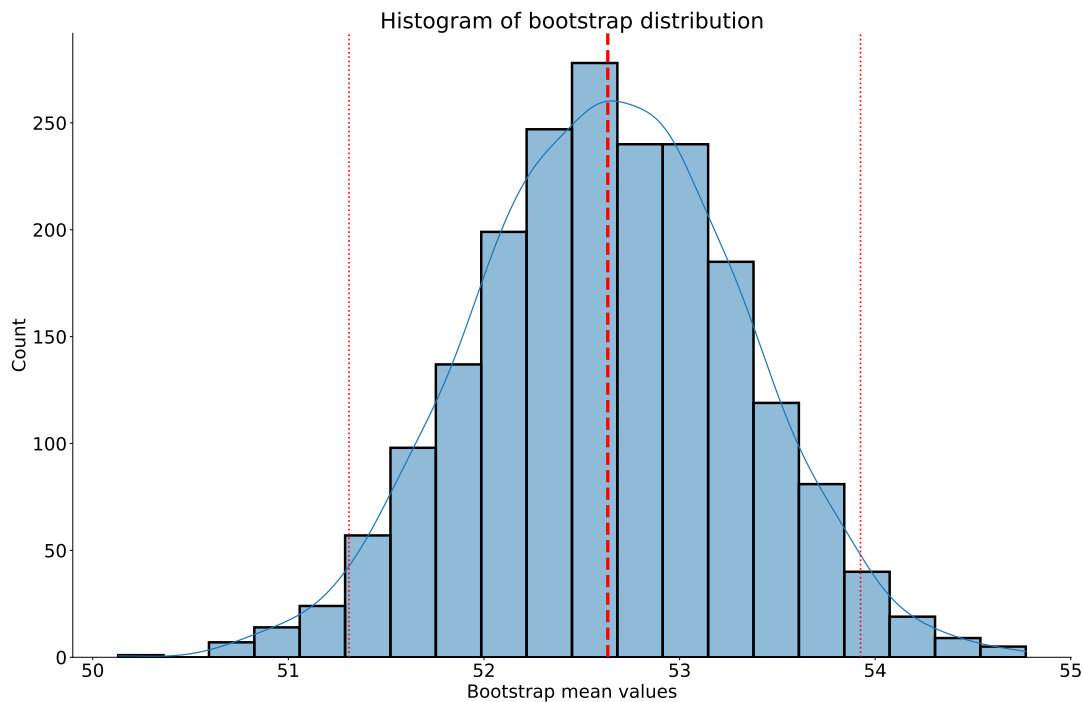
bootstrap_mean_estimate, boot_means = bootstrap_statistic(math, 2000, np.mean)
```

```
## Bootstrap point estimate = 52.63
```

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

```
## At alpha = 0.05, lower and upper bootstrap confidence intervals = 51.31 53.92
```

```
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()
```



## 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 bootstrap 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('Bootstrap point estimate = {:.6.2f}'.format(boot_estimate))
    return(boot_estimate, two_boot_values)

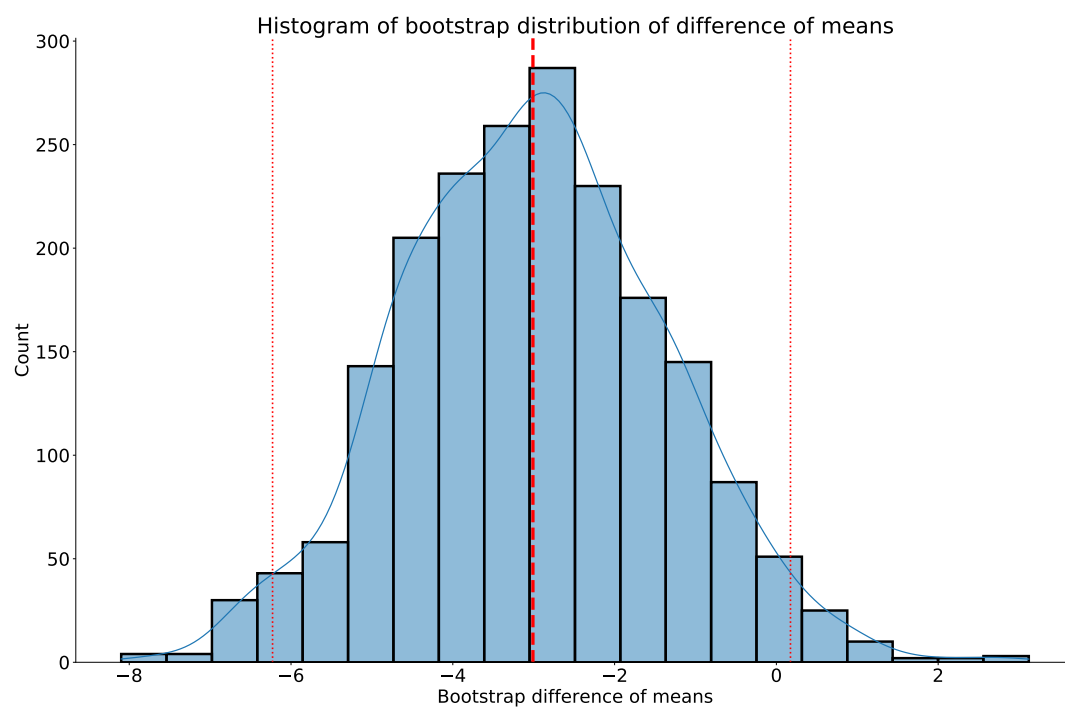
math_low_ses = test_scores.loc[test_scores.loc[:, 'ses']==1, 'math']
math_mid_ses = test_scores.loc[test_scores.loc[:, 'ses']==2, 'math']
bootstrap_diff_of_mean, boot_diffs = two_boot_two_stat(math_low_ses, math_mid_ses, 2000, np.mean, lambda x, y: x - y)

## Bootstrap point estimate = -3.01

LCI, UCI = bootstrap_cis(boot_diffs)

## At alpha = 0.05, lower and upper bootstrap confidence intervals = -6.23    0.17

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



Copyright 2020, 2021, Stephen F. Elston. All rights reserved.