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 statitics. By repeatedly re-sampling data some of the assumptions of classical statistical methods can be relaxed. These computationally intensive methods are products of the computer age. Resampling methods provide a natural way to find uncertainty when performing statistical inferences.

Re-sampling 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 the conditions re met. The CLT tells us that the distribution of mean estiamtes is Normally distributed, as the number of resamples increases.

There are several use cases for resampling methods. Specifically re-sampling methods:

- Allow computation of statistics from limited data samples.
- Esimate 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:** hypothesis tests.
- **Non-parametric bootstrap resampling:** compute statistics.
- **Jackknife:** or leave one out re-sampling.
- **Cross validation:** re-sample into multiple folds without replacement.

Re-sampling methods are general and powerful but, there is no magic involved! When using re-sampling methods always keep in mind these pitfalls: - If a sample is biased, the re-sampled statistic estimated from that sample will be biased. - The sample variance and Cis can be no better than the data sample allows.

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 these permutation methods remain limited, even with modern computers.

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.

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.

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.

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.

General Characteristics of Resampling Methods

General characteristics of resampling methods include

- Allow computation of statistics from limited data samples for statistics with continuous derivatives
- Repeatedly compute statistics from multiple resamples of dataset
- The result converges to the sample distribution of the statistic being computed
- Make minimal distributional assumptions, when compared to classical frequentist statistics

Pitfalls of Resampling Methods

Re-sampling methods are general and powerful but, there is no magic involved! There are pitfalls!

- If a sample is biased, the re-sampled statistic estimate based on that sample will be biased
  - Results can be no better than the sample you start with
  - Example; the bootstrap estimate of mean is the unbiased sample estimate, $\bar{x}$, not the population parameter, μ
- The sample variance and Cis can be no better than the sample distribution allows
  - Be suspicious of overly optimistic confidence intervals
  - CIs can be optimistically biased
- Are computationally intensive, but often highly parallelizable

Point Estimates vs. Distribution Estimates

The goal of **frequentist statistics** is to compute a **point estimate** and **confidence interval** for the point estimate. By a point estimate, we mean a single most likely value for a statistic. The confidence interval is based on the properties of some assumed probability distribution of the statistic. Up to now we have worked with hypothesis tests and summary statistics computed with frequentist methods. For example, with the t-test we found the most likely value of the t-statistic and the confidence interval assuming the difference of means follows a t-distribution.

Are there alternative to this frequentist approach? Yes, indeed there are. The most widely used alternative are found in **Bayesian statistics**. In this lesson, we will focus on re-sampling methods, particularly **bootstrap** methods. The bootstrap lies somewhere between classical frequentist statistics and Bayesian methods.

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

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:

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

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

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

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!**

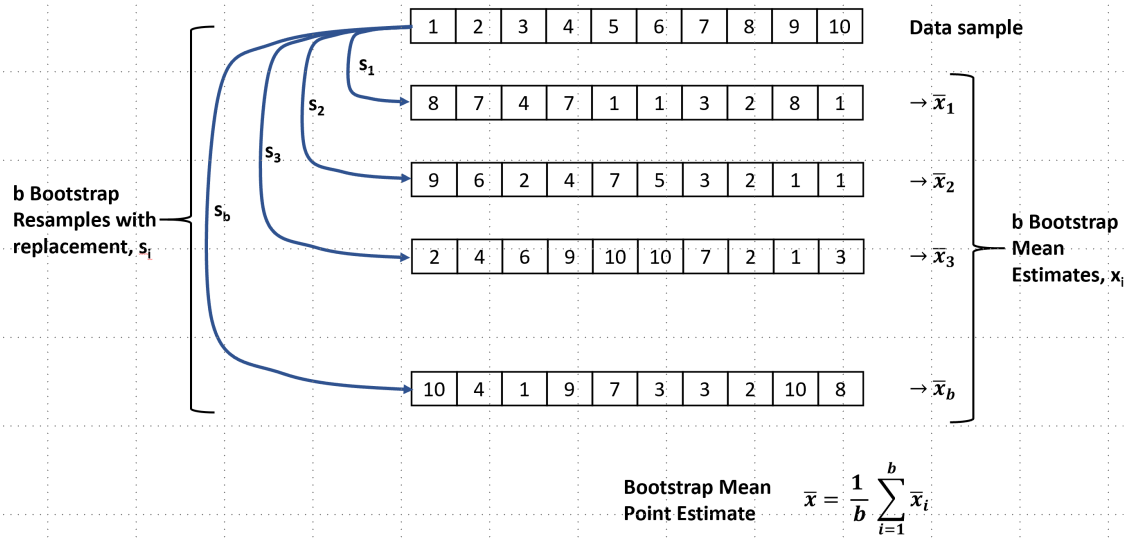


Figure 1: Outline of bootstrap resampling algorithm to compute mean

  - Bootstrap resampling substitutes computer power for paper and pencil statistician power

- Bootstrap resampling estiamtes 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 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)
       plt.rcParams.update({'font.size': 22})
       plt.axvline(x=np.mean(x), color='red', linestyle='dashed', linewidth=2)
       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.

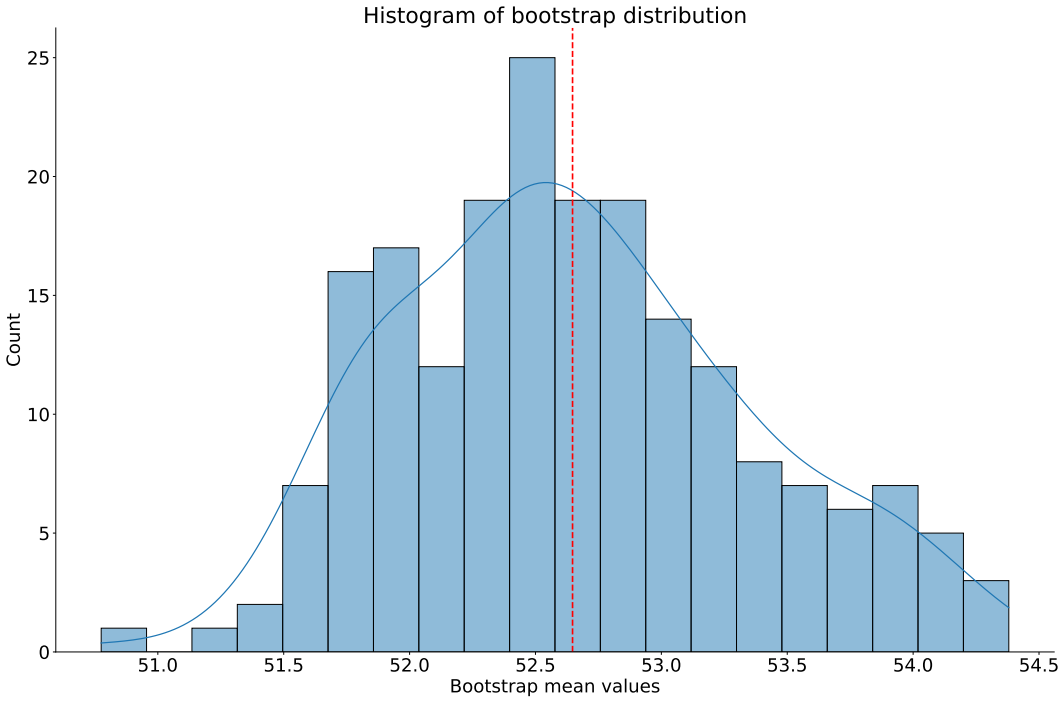
Run the code and examine the plotted results.

```
## 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 = {:6.2f}'.format(boot_estimate))
    return(boot_estimate, boot_vals)

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

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

```
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 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

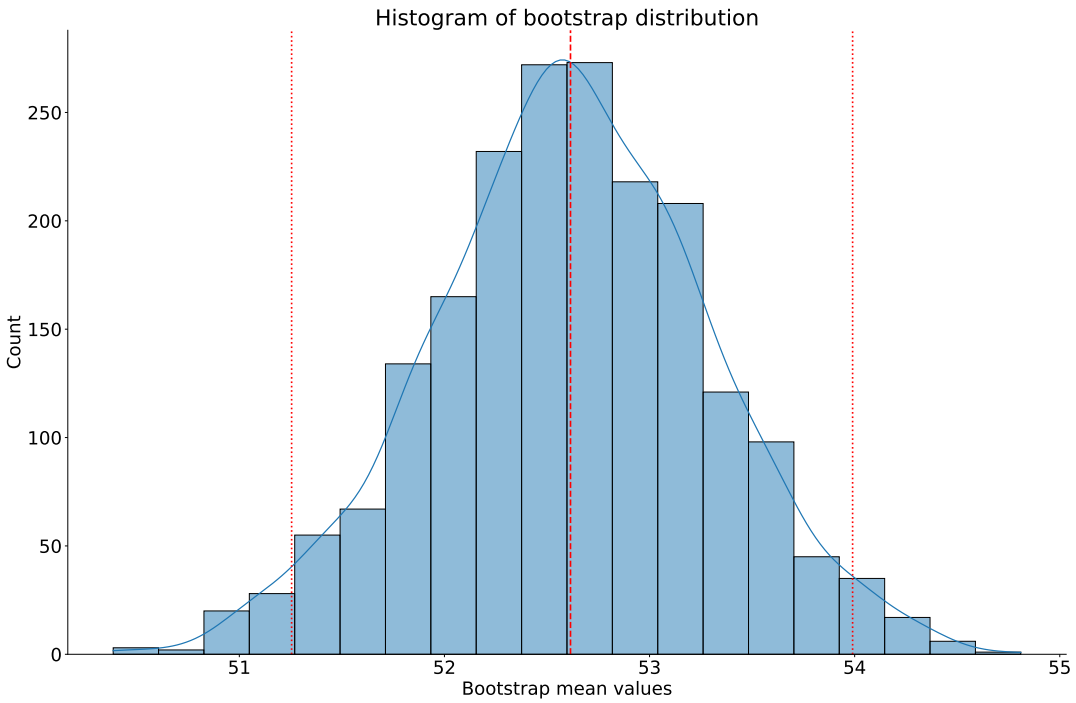
```
## Compute and plot the one sample bootstrap of means with confidence intervals
def bootstrap_cis(boot_samples, alpha=0.05):
      n = len(boot_samples)
      sorted = np.sort(boot_samples)
      index_lci = int(n * alpha / 2)
      index_uci = int(n * (1 - alpha / 2))
      print('At alpha = {0:3.2f}, 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, 2000, np.mean)
```

```
## Bootstap point estimate =  52.61
```

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

```
## At alpha = 0.05, lower and upper bootstap confidence intervals =  51.26    53.99
```

```
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

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, 2000, np.mean, lambda
```

```
## Bootstap point estimate =  -3.09
```

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

```
## At alpha = 0.05, lower and upper bootstap confidence intervals =  -6.21     0.04
```

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