

Bootstrap Tutorial

Hal Varian

This article is an elementary tutorial describing how to implement the bootstrap method in *Mathematica*.

■ Introduction

According to Efron and Tibshirani [1], “The bootstrap is a computer-based method for assigning measures of accuracy to sample estimates.” This remarkable computational technique allows us to generate an estimate of the sampling distribution of almost any statistic using very simple methods. There are several books and articles that explain the statistical theory behind this method, including [1], which provides an excellent introduction. There are also two notebooks available on *MathSource* that implement the bootstrap in *Mathematica* [2, 3].

■ Load Required Packages

```
In[1]:= << Statistics`DescriptiveStatistics`  
        << Statistics`DataManipulation`  
        << Graphics`Graphics`  
        << Statistics`LinearRegression`
```

To make debugging easier, we will seed the random number generator so we get the same random numbers each time the notebook is executed.

```
In[5]:= SeedRandom[123456]
```

■ Distribution of Sample Mean

We begin by reviewing the most elementary problem in statistics, the distribution of the sample mean.

Let (x_i) for $i = 1, \dots, n$ be a sample of IID random variables with $E(x_i) = \mu$ and $\text{var}(x_i) = \sigma^2$. The sample mean is defined by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Since the observations are random variables, the sample mean is a random variable and we can, in principle, compute its distribution. Using the properties of expected value, it is a standard exercise to show that $E(\bar{x}) = \mu$ and

$\text{var}(\bar{x}) = \sigma^2 / n$, which gives us the first two moments of the sampling distribution. If the x_i terms are distributed $N(\mu, \sigma^2)$, we know that a sum of these variables will be normally distributed, so \bar{x} will be distributed $N(\mu, \sigma^2 / n)$.

But what do we do if the x_i s are not normally distributed or if we are computing some function of the sample that is more complex than the mean?

Classical statistics was driven by analytic tractability, and the methods used in classical statistics only apply to certain well-behaved distributions and certain, mostly linear, computations. With modern computers, analytic complexity is no barrier to computing estimates of the sampling distribution of almost any statistic, as we demonstrate next using Monte Carlo simulation.

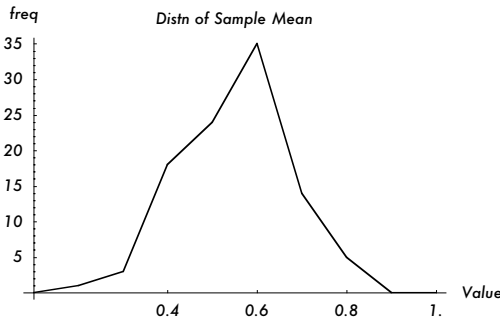
Here we draw a list of 25 uniformly distributed random numbers, compute the mean, and repeat this 100 times. This will give us 100 different estimates of the mean of the underlying distribution.

```
In[6]:= sampleList = Table[Mean[Table[Random[], {25}]], {100}];
```

Let us look at the distribution of these 100 calculated means; this frequency distribution can be viewed as an estimate of the true sampling distribution.

```
In[7]:= Print["Sample list mean: ", Mean[sampleList],
  " and variance: ", Variance[sampleList]]
  ticks1 = {{1, .1}, {4, .4}, {6, .6}, {8, .8}, {10, 1.0}}, Automatic];
  ListPlot[BinCounts[sampleList, {.25, .75, .05}],
  PlotJoined -> True, AxesLabel -> {"Value", "freq"},
  PlotLabel -> "Distn of Sample Mean", Ticks -> ticks1]
```

Sample list mean: 0.501391 and variance: 0.00361008



Since the underlying random variable is uniformly distributed on $[0,1]$, the estimated mean should be close to 0.5. The variance of the uniform distribution is

$$\text{theoryVar} = N \left[\int_0^1 \left(x - \frac{1}{2} \right)^2 dx \right]$$

```
Out[10]= 0.0833333
```

So the variance of the sample mean of 25 observations should be

```
In[11]:= meanVar =  $\frac{\text{theoryVar}}{25}$ 
```

```
Out[11]= 0.00333333
```

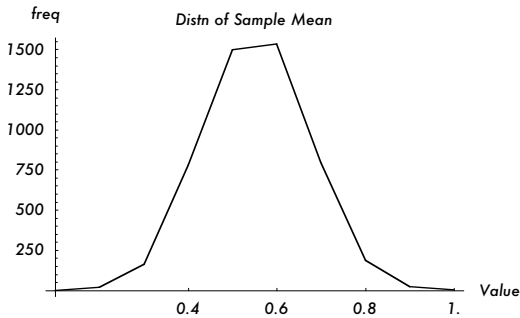
The estimates we have computed should not be too far from these numbers.

We can do the same thing for 5000 repetitions, in which case the estimated results should be much closer to the theoretical predictions.

```
In[12]:= sampleList = Table[Mean[Table[Random[], {25}]], {5000}];
```

```
In[13]:= Print["Sample list mean: ", Mean[sampleList],
  " and variance: ", Variance[sampleList]]
ticks1 = {{1, .1}, {4, .4}, {6, .6}, {8, .8}, {10, 1.0}}, Automatic};
ListPlot[BinCounts[sampleList, {.25, .75, .05}],
  PlotJoined → True, AxesLabel → {"Value", "freq"},
  PlotLabel → "Distn of Sample Mean", Ticks → ticks1]
```

```
Sample list mean: 0.501418 and variance: 0.00330898
```



The Monte Carlo method can be used to compute an estimate of the sampling distribution for virtually any statistic, as long as we know the distribution from which the samples are drawn.

■ The Bootstrap

Unfortunately, in most cases we do not know the underlying distribution from which the sample is drawn. At best we may suspect that the true distribution is in some family of distributions, but we generally do not know the parameters of the distribution. If we did, we could just apply the theoretical formulas and be done with it.

So suppose that we have just one sample. Is there any way to use that one sample to compute an estimate of the sampling distribution of a statistic? This is where the bootstrap comes in.

The idea is to repeatedly sample (with replacement) from the single sample you have, and use these “samples” to compute the sampling distribution of the

statistic in which you are interested. In the previous Monte Carlo exercise, we drew a “fresh” sample each time; in the bootstrap case, we *resample* from the single sample that we have. Other than that difference, the procedures are essentially the same. If our original sample is reasonably representative of the population, then resampling from that sample should look pretty much like drawing a new sample.

The remarkable thing about the bootstrap is that even though we only have a single sample, it can often be used to give a quite good estimate of what would happen if we really were able to draw new, fresh samples.

Here is a function that will resample (with replacement) from a list.

```
In[16]:= Resample[list_] :=  
        list[[Table[Random[Integer, {1, Length[list]}], {Length[list}]]]
```

To make sure it works, we will try it on a test list.

```
In[17]:= test = {a, b, c, d, e, f, g}
```

```
Out[17]= {a, b, c, d, e, f, g}
```

```
In[18]:= Resample[test]
```

```
Out[18]= {f, c, d, b, e, f, c}
```

Here is a fixed random sample of 25 numbers.

```
In[19]:= SeedRandom[123456]
```

```
In[20]:= theSample = Table[Random[], {25}];
```

Now we will resample from this fixed sample 5000 times and look at this distribution of estimated means.

```
In[21]:= bootList = Table[Mean[Resample[theSample]], {5000}];
```

■ Comparison of the Distributions

Here are some summary statistics comparing the sampling distribution computed earlier and the bootstrap distribution.

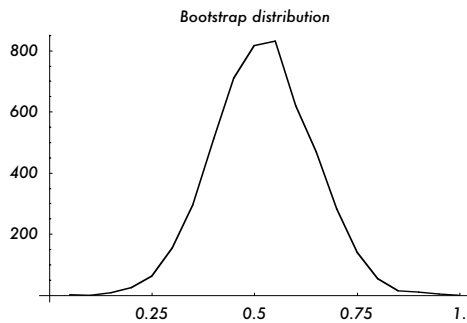
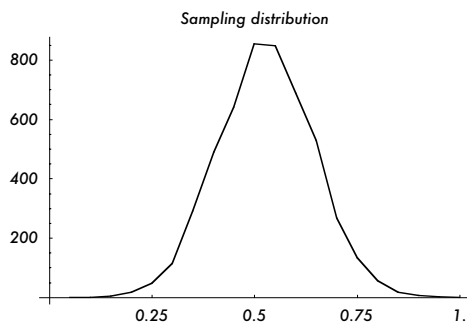
```

In[22]:= Print["Sample list mean: ", Mean[sampleList],
  " and variance: ", Variance[sampleList]]
Print["Boot list mean: ", Mean[bootList],
  " and variance: ", Variance[bootList]]
ticks2 = {{5, .25}, {10, .5}, {15, .75}, {20, 1.0}}, Automatic};
ListPlot[BinCounts[sampleList, {.25, .75, .025}], PlotJoined → True,
  PlotLabel → "Sampling distribution", Ticks → ticks2]
ListPlot[BinCounts[bootList, {.25, .75, .025}], PlotJoined → True,
  PlotLabel → "Bootstrap distribution", Ticks → ticks2]

```

Sample list mean: 0.501418 and variance: 0.00330898

Boot list mean: 0.497918 and variance: 0.00356321



Note that the distributions are reasonably similar. This is the genius of the bootstrap: resampling from the single sample provides a reasonable way to estimate what would happen if we actually drew many separate samples.

■ Bootstrapping Other Statistics

We can generalize the previous operations to bootstrap other statistics. Here is a function that will bootstrap any statistic `stat` of any sample.

```

In[27]:= bootIt[sample_, n_, stat_] := Table[stat[Resample[sample]], {n}]

```

```

In[28]:= bootIt[theSample, 10, Mean]
Out[28]= {0.420568, 0.628813, 0.450471, 0.478979, 0.543052,
          0.522497, 0.490412, 0.493006, 0.425277, 0.60749}

In[29]:= bootIt[theSample, 10, StandardDeviation]
Out[29]= {0.285901, 0.24937, 0.271834, 0.220513, 0.315868,
          0.308899, 0.33619, 0.302769, 0.319604, 0.294518}

In[30]:= bootIt[theSample, 10, Median]
Out[30]= {0.395873, 0.395873, 0.395873, 0.642715, 0.591733,
          0.60591, 0.591733, 0.322472, 0.322472, 0.306926}
    
```

The sampling distribution of the mean and standard deviation have been well studied. The sampling distribution of the median, on the other hand, is less well known, but it is trivial to compute an estimate of it with the bootstrap.

■ Regression

Now we will bootstrap a regression model. There are two ways to do this: we can assume that the matrix of explanatory variables is fixed or random. The fixed case is easier on both theoretical and computational grounds, but the random case is more realistic in most applications of interest. To keep the exposition simple, we will only examine the fixed regressors case. This case is applicable if, for example, the explanatory variables were created as part of a controlled experiment (see [1] for a more detailed discussion).

We start with some data and estimate the regression model.

```

In[31]:= SeedRandom[123456]
In[32]:= xData = Table[Random[], {50}];
          yData = Table[Random[], {50}];
In[34]:= Regress[Transpose[{xData, yData}], {1, x}, x]
Out[34]= {ParameterTable →
          Estimate SE TStat PValue
          1 0.463401 0.0803805 5.7651 5.71645 × 10-7,
          x 0.0606434 0.13231 0.458344 0.648773
          RSquared → 0.00435759, AdjustedRSquared → -0.016385,
          EstimatedVariance → 0.0805626,
          ANOVATable →
          DF SumOfSq MeanSq FRatio PValue
          Model 1 0.0169246 0.0169246 0.21008 0.648773 }
          Error 48 3.867 0.0805626
          Total 49 3.88393
    
```

Since the x and the y variables are IID, the intercept term should be the mean of the y variable (.5) and the coefficient on the x variable should be about 0. In this

case the coefficient on the x variable is not significantly different from 0, as illustrated by the low value of the t -statistic.

To do the bootstrap, we form a set of the data pairs and then draw a random sample, with replacement, from that set. The same resampling function we used before works for this.

```
In[35]:= testData = Transpose[{{a1, a2, a3}, {b1, b2, b3}}]
Out[35]= {{a1, b1}, {a2, b2}, {a3, b3}}
In[36]:= Resample[testData]
Out[36]= {{a3, b3}, {a3, b3}, {a3, b3}}
```

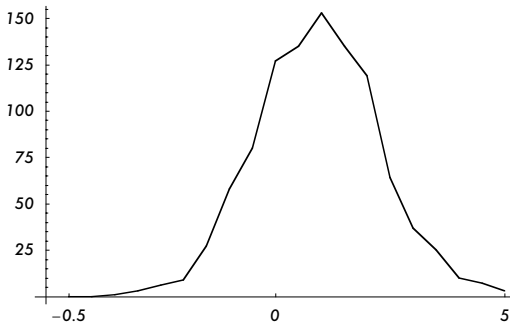
Now we simply run our regression on the resampled data and keep track of the regression coefficient. Here is a function that runs a regression and returns the slope coefficient.

```
In[37]:= coeff[data_] := Rest[Fit[data, {1, xx}, xx]][[1]]
In[38]:= allData = Transpose[{xData, yData}];
In[39]:= coeff[allData]
Out[39]= 0.0606434
```

Now we will do the bootstrap on the coefficient 1000 times.

```
In[40]:= slopeBoot = Table[coeff[Resample[allData]], {1000}];
In[41]:= Print["Boot list mean: ", Mean[slopeBoot],
  " and standard deviation: ", StandardDeviation[slopeBoot]]
ListPlot[BinCounts[slopeBoot, {- .5, .5, .05}], PlotJoined → True,
  Ticks → {{{1, - .5}, {10, 0}, {20, +5}}, Automatic]]
```

Boot list mean: 0.0674443 and standard deviation: 0.134648



The mean of the sampling distribution should be close to zero, and its standard deviation should be about what the regression computed for the standard error of the coefficient.

The bootstrap is a handy tool and particularly easy to implement in *Mathematica*. With the resampling functions described earlier, we can estimate the sampling distribution of virtually any statistic.

■ References

- [1] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Boca Raton, FL: CRC Press, 1994.
- [2] S. Aksenov, "Confidence Intervals by Bootstrap," *MathSource*. (Jan 27, 2003) library.wolfram.com/infocenter/MathSource/4272.
- [3] E. Sinikaran, "BootStrapPackage: A Package of Bootstrap Algorithms for Mean, Simple Linear Regression Models and Correlation Coefficient," *MathSource*. (May 30, 2002) library.wolfram.com/infocenter/MathSource/815.

About the Author

Hal Varian is the Class of 1944 Professor in the School of Information Management and Systems, the Haas School of Business, and the Department of Economics at UC Berkeley. He has edited two books on economic and financial applications of *Mathematica*.

Hal Varian

www.sims.berkeley.edu/~hal