# Post01: Don’t Think Small!!

## How Bootstrapping Can Help Overcome Challenges with Small Data Sets

Welcome to Jobe Cowen’s Post01 on Bootstrapping!! We’ll start by introducing the concept of bootstrapping, and then we’ll play around with some calculations in R to show some interesting properties of this concept. Then we’ll see how we can use it to solve certain problems and create some useful applications of it. References and further reading is included at the bottom of the post.

First of all, what the heck is bootstrapping?!

Basically speaking, bootstrapping is a method of resampling a small data set over and over again, in order to overcome problems that arise because the set is so small. In a funny way, bootstrapping sometimes has a rather American connotation. Many of us have heard of the idea of someone “pulling themselves up by their bootstraps.” This is what a lot of people call the American Dream - starting from the bottom, working hard, and achieving success. This concept of using the existing resources available to you to get yourself into, or out of, a certain situation is actually a central component to all the various definitions of bootstrapping throughout different fields of study. As we will come to find out, bootstrapping is a concept which has many practical applications, including in the world of business and finance. In fact, the author first became aware of bootstrapping, and the related concept of Monte Carlo simulations (mentioned at the end), while studying finance in London.

Let’s first take a look at a problem that could arise, where bootstrapping could help, and then we will get into more of the conceptual meat of the subject. We’ll need a few packages for our code:

```
#Let's load these packages which will be used later
library(graphics)
library(boot)
library(stats)
```

# Buying a New Car: Applying Bootstrapping

Let’s imagine a problem, where we are in the market to get a new, or slightly-used, car, and we are trying to find the average price in our area so we know what we should expect to pay. However, let’s say we only have a handful of car prices available to us. Now, of course, we could use simple {base} and {stats} functions to calculate some basic statistics of our small data set (e.g. the mean and the standard deviation):

```
#create a vector of 10 local car prices (in thousands of USD)
car_price <- c(10, 15, 16, 18, 20, 25, 18, 15, 16, 21)

#mean
mean(car_price)
```

```
## [1] 17.4
```

```
#standard deviation
sd(car_price)
```

```
## [1] 4.060651
```

# The Meat of the Matter: Some Important Concepts in Bootstrapping

Ok! Looks good, but we can do better. It’s important to remember that when doing random sample with a small number of samples problems can arise. In fact, the smaller the sample set, the more likely you are to have inaccuracy. If this is difficult to conceptualize, recall that in probability theory, the Central Limit Theorem tells us that if you have a large enough set of samples, the distribution of all of the independent trials (or samples) can be thought of as a normal distribution. This is true no matter whether the distribution of the individual trials is normal or not. And, as we know, having a nice normal distribution with its beautiful bell-curve to work with, means that we can enjoy the well-known and convenient properties of the normal distribution, its mean, and its standard deviation. We’ll move on, but the reader is encouraged to further explore these questions within probability theory.

But… wait! In our car price problem, we only had 10 values in our set to sample. Surely this is not enough… but, this is where the power of bootstrapping shows itself! Basically we’re going to *resample* those 10 elements within our sample set repeatedly, so that we have a *new* sample set of means (or medians, or whatever statistic we want to calculate) as large as we like! Let’s see what this will look like.

# Calculating and Visualizing Bootstrapped Data

First we’ll play around with some of the functions we’ll be using, to see how they work. Remember that the base function **seq()** creates a sequence of numbers starting *from* the number in the first argument, *to* the number in the second argument, progessing *by* steps as large as the third argument. For example:

```
#create a sequence from 2 to 10, progressing by steps of 2
seq(from = 2, to = 10, by = 2)
```

```
## [1]  2  4  6  8 10
```

Great! Now let’s use another {base} function **replicate(n, …)** to take our vector and *replicate* it n number of times. The output here will be a matrix with n columns.

```
#replicate our previous sequence 4 times
replicate(n = 4, seq(2, 10, 2))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    2    2    2
## [2,]    4    4    4    4
## [3,]    6    6    6    6
## [4,]    8    8    8    8
## [5,]   10   10   10   10
```

Terrific! Lastly we want to play around with {base} function **sample()** to see how to take a certain number of random samples of a data set. Importantly, in order to be a bootstrap sample, the sample has to be the same size as the set (here that is 10). Unless we alter the *prob* = parameter, each element of the sample set is, by default, equally likely to be chosen. Also, the default of the *replace* = parameter is FALSE, meaning sampling *without* replacement.

```
#take a random sample, with replacement, of our car price data of size 10
sample1 <- sample(x = car_price, size = 10, replace = TRUE)
sample1
```

```
##  [1] 20 10 18 16 18 15 16 25 20 15
```

Looks good! We'll try it once more, this time using the {base} **set.seed()** function to get the same result each time we run the code (since nothing is truly random!!).

```
#take a random sample, with replacement, of our car price data of size 10
set.seed(123)
sample2 <- sample(car_price, size = 10, replace = TRUE)
sample2
```

```
##  [1] 16 15 20 16 21 10 25 16 25 20
```

For good measure we'll take the mean and standard deviation of sample2:

```
#mean
mean(sample2)
```

```
## [1] 18.4
```

```
#standard deviation
sd(sample2)
```

```
## [1] 4.695151
```

Now it's time to do some bootstrapping!! Let's put it all together! We're going to take the mean of our sample, and replicate this process over and over again. Let's have R repeat it 100 times. Again we'll set the seed for consistent results, and we'll examine the mean and standard deviation:

```
set.seed(123)
#create a bootstrapped set with 100 resamplings
bootstrap1 <- replicate(100, mean(sample(car_price, 10, replace = TRUE)))

#mean
mean(bootstrap1)
```

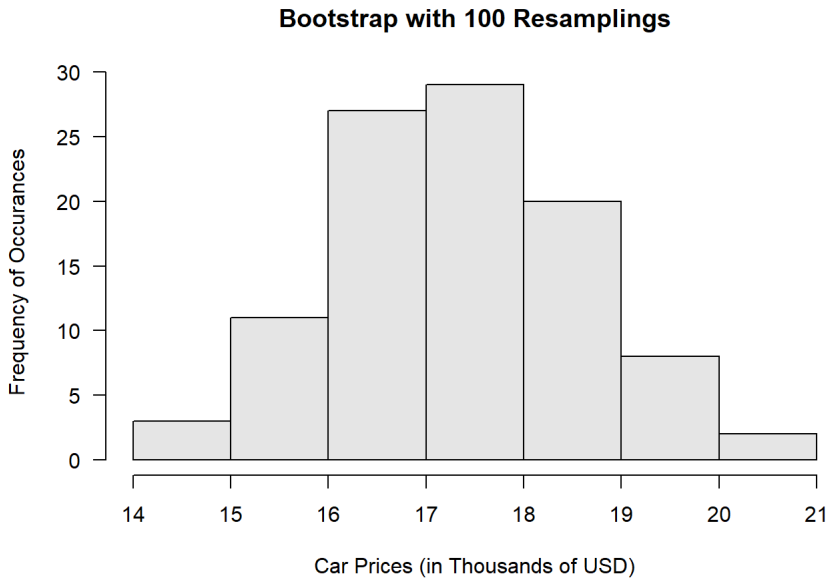
```
## [1] 17.394
```

```
#standard deviation
sd(bootstrap1)
```

```
## [1] 1.234587
```

Interesting! Notice how even with the same seed, we get different results for the mean and standard deviation when we bootstrap!! In fact, the standard deviation is much lower in the bootstrapped variable! What does this look like though? We'll use the **hist()** function in the {graphics} package to get a histogram of replicated results:

```
#histogram of our bootstrapped data set
hist(bootstrap1,main = "Bootstrap with 100 Resamplings",
     xlab = 'Car Prices (in Thousands of USD)', ylab = 'Frequency of Occurances',
     col = 'grey90', las = 1)
```




Looks pretty good! But now let's try upping the number of times we resample the set. Let's try 10000 times:

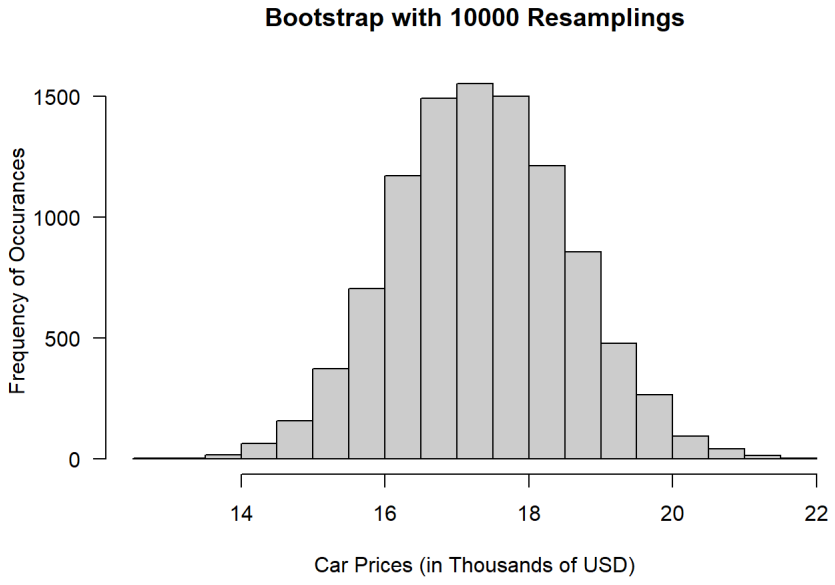
```
set.seed(123)
#create a bootstrapped set with 10000 resamplings
bootstrap2 <- replicate(10000, mean(sample(car_price, 10, replace = TRUE)))

#mean
mean(bootstrap2)
```

```
## [1] 17.39733
```

```
#standard deviation
sd(bootstrap2)
```

```
## [1] 1.218983
```

```
hist(bootstrap2, main = "Bootstrap with 10000 Resamplings",
     xlab = 'Car Prices (in Thousands of USD)', ylab = 'Frequency of Occurances',
     col = 'grey80', las = 1)
```




```
#, abline(v = mean(g) + 1)
```

Amazing! Just as we mentioned above, and as enumerated by the Central Limit Theorem, as the size of the sample set goes to large numbers, we approach a normal distribution! Here 10000 resamplings is good enough to begin to see the normal curve and its mean.

Finally, let's quickly finish off by showing one example of a package in R that was designed specifically for bootstrapping. We'll take a look at the **boot()** with the {boot} package. There are other bootstrapping functions in R, including **bootstrap()** in {broom}, but we will leave these aside for now.

We need to define a function which will be one of our parameters in the **boot()** function.

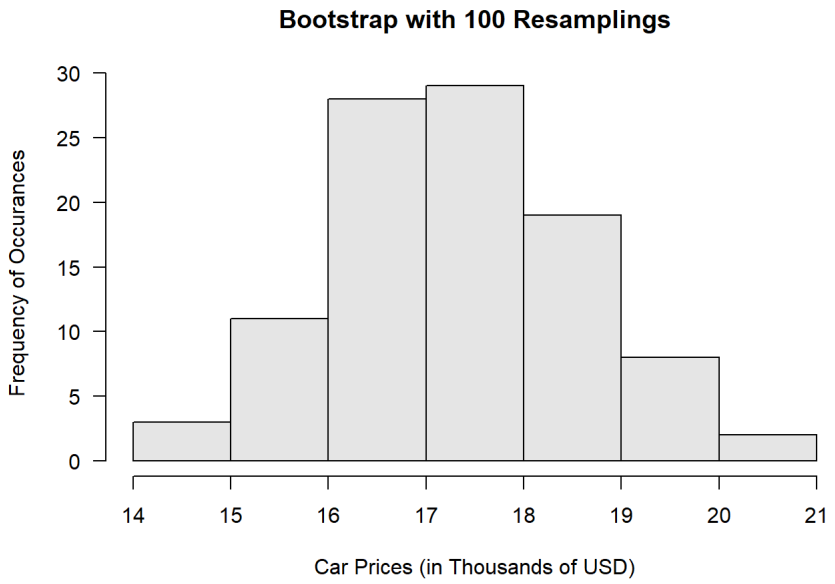
```
#create a function that returns the mean of one of our random samples
bootfunc <- function(x){
  return(mean(sample(x, 10, replace = TRUE)))
}
```

Now we simply call the boot function with the appropriate arguments to create our bootstrapped data set. Here we use *R* = to set the number of resamplings, and in our particular case, we'll need to set the *sim* = argument to 'parametric' (due to the dimension of our sample set).

```
#use boot function to create set with 100 resamplings
set.seed(123)
newboot1 <- boot(car_price, bootfunc, R = 100, sim = 'parametric')
```

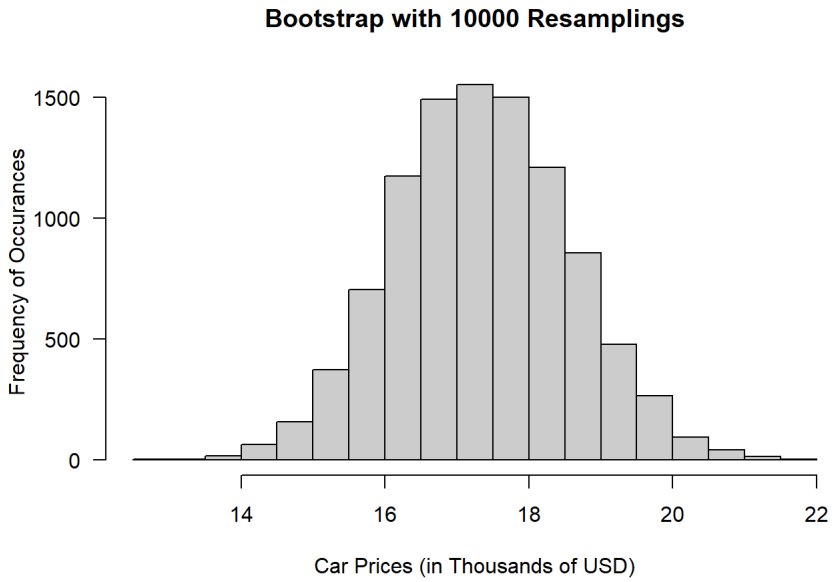
Great! Now we'll just plot this using our **hist()** function and the dollar operator ($) (since we only want to select one list out of several that are actually created by the **boot()** function).

```
#create a new histogram
hist(newboot1$t, main = "Bootstrap with 100 Resamplings",
     xlab = 'Car Prices (in Thousands of USD)', ylab = 'Frequency of Occurances',
     col = 'grey90', las = 1)
```




Wonderful! Now we'll create one more bootstrapped set with the **boot()** function and plot its histogram:

```
#use boot function to create set with 100 resamplings
set.seed(123)
newboot2 <- boot(car_price, bootfunc, R = 10000, sim = 'parametric')

#create one final histogram
hist(newboot2$t, main = "Bootstrap with 10000 Resamplings",
     xlab = 'Car Prices (in Thousands of USD)', ylab = 'Frequency of Occurances',
     col = 'grey80', las = 1)
```




Awesome! It looks just like the histogram of the bootstrapped data we calculated on our own!!

Lastly, we'll just display **newboot2** along with it's summary and mean to give the reader an idea of how the **boot()** function stores the data.

```
newboot2
```

```
##
## PARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = car_price, statistic = bootfunc, R = 10000, sim = "parametric")
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     18.4 -1.00286    1.218974
```

```
summary(newboot2)
```

```
##           Length Class  Mode
## t0             1  -none- numeric
## t          10000  -none- numeric
## R              1  -none- numeric
## data          10  -none- numeric
## seed         626  -none- numeric
## statistic      1  -none- function
## sim            1  -none- character
## call           5  -none- call
## ran.gen        1  -none- function
## mle            0  -none- NULL
```

```
mean(newboot2$t)
```

```
## [1] 17.39714
```

Now obviously these concepts and functions can be applied to much more complex problems, but this gives us a good taste of what we are dealing with.

# Conclusion

So whether it's the American ideals of buying a new car and of pulling yourself up by your own bootstraps, or more sophisticated problems of mapping out stock volatility and measuring confidence intervals, the applications of bootstrapping are wide and varied. They certainly span more than this post can cover, so the reader is encouraged to explore the topic in greater detail. Reading through the provided references will not only provide information relevant to the creation of this post, but the reader will also begin to explore other related topics, including jackknifing and exciting Monte Carlo simulations. Happy exploring!!

---

## References


- https://www.statmethods.net/advstats/bootstrapping.html
- https://cran.r-project.org/web/packages/boot/boot.pdf

- https://stats.idre.ucla.edu/r/faq/how-can-i-generate-bootstrap-statistics-in-r/
- https://www.youtube.com/watch?v=cv2TSOHPp1o
- https://www.rdocumentation.org/packages/base/versions/3.4.1/topics/sample
- https://www.rdocumentation.org/packages/base/versions/3.4.1/topics/lapply

(for replicate function)

- https://www.rdocumentation.org/packages/graphics/versions/3.4.0/topics/hist
- https://cran.r-project.org/web/packages/broom/vignettes/bootstrapping.html

(further reading - bootstrapping with broom package)

- https://stats.stackexchange.com/questions/104040/resampling-simulation-methods-monte-carlo-bootstrapping-jackknifing-cross

(further reading - related topics)

---