Untitled

Mario Ibanez

February 20, 2016

1. Double Bootstrap Method Double Bootstrap Method is another non-parametric method. The essential idea is that we need to do bootstrapping twice.
   samples for the first bootstrapping and samples obtained from each
   sample, total number of second samples is . We will construct three confidence level with different nominal levels (99%, 98%, 95%) for the each samples by Percentile Method, then compute the true probability of the intervals contain , use quadratic interpolation to find the confidence level (1-) that we need which the true coverage probability is 90%. Because when the nominal level is 90%, in reality the coverage probability may be 89% or 92%. Finally using Percentile Method again to construct the (1-)% confidence interval for , that is the average in the case. The algorithm:
2. Randomly select a bootstrap sample of size 12 from with replacement, repeat 500 times. These are samples.
3. For each of the samples, bootstrap 500 samples again. These are samples. Compute the average of each sample.
4. Construct 3 confidence interval with different nominal level 99%, 98%, 95% by Percentile Method.
5. Compute the proportion of the intervals that covers 19.2908 for the three nominal level that are the true coverage probability for each one.
6. Use quadratic interpolation to find [Equation]. Use average from B1 samples to construct the (1-)% confidence interval for

Step 1

```
# Haircut price data from class
data <- c(17.00, 16.49, 45.00,
          25.00, 12.00, 18.00,
          13.00, 20.00, 10.00,
          0.00, 15.00, 50.00)

x_bar <- mean(data)

# All data will be stored here
list_of_samples <- c()

# For loop to generate 500 samples with replacement
for (B in 1:500){
  # Selects a random sample with replacement from the data and calculate mean
  boot_strapped_sample <- sample(x = data, size = 12, replace = TRUE)
  sample_mean <- mean(boot_strapped_sample)
  # Stores that information into a list
  list_of_samples[[B]] <- list(boot_strapped_sample, sample_mean)
}

# Now we have all of our information
head(list_of_samples)
```

```
## [[1]]
## [[1]][[1]]
##  [1] 25.00  0.00 50.00 15.00 45.00 50.00 16.49 20.00 45.00 18.00 16.49
## [12] 50.00
##
## [[1]][[2]]
## [1] 29.24833
##
##
## [[2]]
## [[2]][[1]]
##  [1] 10.00 18.00  0.00 16.49 50.00 20.00 25.00 25.00 50.00 18.00 12.00
## [12] 15.00
##
## [[2]][[2]]
## [1] 21.62417
##
##
## [[3]]
## [[3]][[1]]
##  [1] 25.00 15.00 18.00 18.00  0.00 16.49 50.00  0.00 16.49 17.00 15.00
## [12] 20.00
##
## [[3]][[2]]
## [1] 17.58167
##
##
## [[4]]
## [[4]][[1]]
##  [1] 18.00 50.00 45.00 18.00 16.49 16.49 12.00  0.00 13.00 45.00 17.00
## [12] 25.00
##
## [[4]][[2]]
## [1] 22.99833
##
##
## [[5]]
## [[5]][[1]]
##  [1] 17 20 45 25 45 45 15 15 45 13 25 20
##
## [[5]][[2]]
## [1] 27.5
##
##
## [[6]]
## [[6]][[1]]
##  [1]  0 10 18 17 50 12 50 20 20  0 15 45
##
## [[6]][[2]]
## [1] 21.41667
```

Step 2

Now we have a list of 5 pieces of information. The first thing in the list is:

```
list_of_samples[[1]]
```

```
## [[1]]
##  [1] 25.00  0.00 50.00 15.00 45.00 50.00 16.49 20.00 45.00 18.00 16.49
## [12] 50.00
##
## [[2]]
## [1] 29.24833
```

That is the first boot strapped sample and its mean. If you want to get just the sample or just the mean, you do this:

```
list_of_samples[[1]][1]
```

```
## [[1]]
##  [1] 25.00  0.00 50.00 15.00 45.00 50.00 16.49 20.00 45.00 18.00 16.49
## [12] 50.00
```

```
list_of_samples[[1]][2]
```

```
## [[1]]
## [1] 29.24833
```

Now we need to do what we just did again on this bootstrapped sample. Right now each of the things in the list has two items, the sample, and its mean. We're going to add a 3rd thing to each one, ***another list of 500*** things!

```
# The first loop is for original bootstrapped samples
for(B in 1:500){
  # This is the one we are going to work on now:
  original_boot_strapped_sample <- list_of_samples[[B]][[1]]
  # Initialize the list that will be associated with this one
  second_list_of_samples <- list()
  # The second for loop is to repeat the process for each bootstrapped sample
  for (C in 1:500){
  # Selects a random sample with replacement from the data and calculate mean
  boot_strapped_sample <- sample(x = original_boot_strapped_sample, size = 12, replace = TRUE)
  sample_mean <- mean(boot_strapped_sample)
  # Stores that information into a list
  second_list_of_samples[[C]] <- list(boot_strapped_sample, sample_mean)
  }
  list_of_samples[[B]][[3]] <- second_list_of_samples
}
```

Kind of complicated..

For example, if you want to access the first bootstrapped sample that was made from the first bootsrapped sample, you would do this:

```
list_of_samples[[1]][[3]][[1]][[1]]
```

```
##  [1] 16.49 20.00 16.49 16.49 45.00 20.00 50.00 20.00 18.00 50.00 15.00
## [12] 16.49
```

The first number means which original bootstrap.

The second number means which do you want, the sample = 1, the mean = 2, or the list of samples from that sample = 3.

The third number means which which bootstrap sample do you want

The fourth means which do you want, 1 = sample, 2 = sample mean

In fact maybe we could make a function so that it's easier to understand:

```
access <- function(first_layer, item, second_layer = 1, sample_or_mean = 1){
  if(item == 3){
    return(list_of_samples[[first_layer]][[3]][[second_layer]][[sample_or_mean]])
  }
  else {
    return(list_of_samples[[first_layer]][[item]])
  }
}
```

```
access(1,2)
```

```
## [1] 29.24833
```

```
access(1,3,1,1)
```

```
##  [1] 16.49 20.00 16.49 16.49 45.00 20.00 50.00 20.00 18.00 50.00 15.00
## [12] 16.49
```

Step 3 Construct 3 confidence interval with different nominal level 99%, 98%, 95% by Percentile Method.

Now we need to make 3 confidence intervals for each of the first layers, using the 500 sample means found within the 1st bootstrapped sample. First we need to get the 500 sample means, then put them in order, then create the three confidence intervals, and then check if it contains the original samples sample mean.

```
# This data frame will store the results for each of the 500 cases
contains_x_bar <- data.frame(Conf95 = c(rep(0, 500)),
                             Conf98 = c(rep(0, 500)),
                             Conf99 = c(rep(0, 500)))

for(B in 1:500){
  # Now lets get the sample means, they'll be stored here
  vector_of_means <- c(rep(FALSE, 500))
  for(C in 1:500){
    vector_of_means[C] <- access(first_layer = B, item = 3, second_layer = C, sample_or_mean = 2)
```

```
  }
  # Sort the vector of means
  vector_of_means <- sort(vector_of_means)

  # Now we can check the percentile confidence intervals contain x_bar
  # note---- 99% means [5, 495]
  #          98% means [10, 490]
  #          95% means [25, 475]
  if((vector_of_means[25] <= x_bar) && (vector_of_means[475] >= x_bar)){
    contains_x_bar[B, 1] <- TRUE
  }
  if((vector_of_means[10] <= x_bar) && (vector_of_means[490] >= x_bar)){
    contains_x_bar[B, 2] <- TRUE
  }
  if((vector_of_means[5] <= x_bar) && (vector_of_means[495] >= x_bar)){
    contains_x_bar[B, 3] <- TRUE
  }
}
```

Let's see the results:

```
head(contains_x_bar, 50)
```

```
##    Conf95 Conf98 Conf99
## 1       0      1      1
## 2       1      1      1
## 3       1      1      1
## 4       1      1      1
## 5       0      1      1
## 6       1      1      1
## 7       1      1      1
## 8       1      1      1
## 9       1      1      1
## 10      1      1      1
## 11      1      1      1
## 12      1      1      1
## 13      1      1      1
## 14      1      1      1
## 15      1      1      1
## 16      1      1      1
## 17      1      1      1
## 18      1      1      1
## 19      1      1      1
## 20      0      0      0
## 21      0      0      0
## 22      1      1      1
## 23      0      0      0
## 24      1      1      1
## 25      0      0      0
## 26      1      1      1
## 27      1      1      1
## 28      1      1      1
## 29      0      1      1
```

```
## 30      1      1      1
## 31      1      1      1
## 32      1      1      1
## 33      1      1      1
## 34      1      1      1
## 35      1      1      1
## 36      1      1      1
## 37      1      1      1
## 38      1      1      1
## 39      1      1      1
## 40      1      1      1
## 41      1      1      1
## 42      0      0      0
## 43      1      1      1
## 44      0      0      0
## 45      1      1      1
## 46      1      1      1
## 47      1      1      1
## 48      1      1      1
## 49      1      1      1
## 50      1      1      1
```

Now to find the percentages:

```
sapply(contains_x_bar, sum) / 500
```

```
## Conf95 Conf98 Conf99
##  0.838  0.890  0.902
```