Bootstrap Methods

Jonathan Navarrete

June 11, 2017

Introduction

Bootstrap methods in simple terms are methods of *resampling* observed data to estimate the empirical CDF from which the observed data is supposed to have originate from. Suppose we observe *independent* samples $x_1, ..., x_n$ from pdf/pmf f, and whose CDF F is unobservable (directly). Well, given that $X = (x_1, ..., x_n)^T$ originates from F, we can use X to generate F_n which is itself an estimate of F. If we sample (with replacement) another set of n observations from F_n, we will have $X^* = (x_1^*, ..., x_n^*)^T$. This new sample X^* can then generate a CDF, F_n^* which is another estimate of F_n. That is, F_n^* is a bootstrap estimator of F. We can continue this process of resampling with replacement to obtain samples $X_1^*, X_2^*, ..., X_B^*$ and $F_{n,1}^*, F_{n,2}^*, ..., F_{n,B}^*$.

In addition to estimating F, there may be a statistic of interest θ (e.g. mean). We can use bootstrap methods to calculate an empirical distribution of θ. From our original sample X we can calculate estimate $\hat{\theta}$. Our bootstrap sample can also be used to calcualte an estimate, $\hat{\theta}_1^*, ..., \hat{\theta}_B^*$.

A simple bootstrap algorithm for *independent* samples X is:

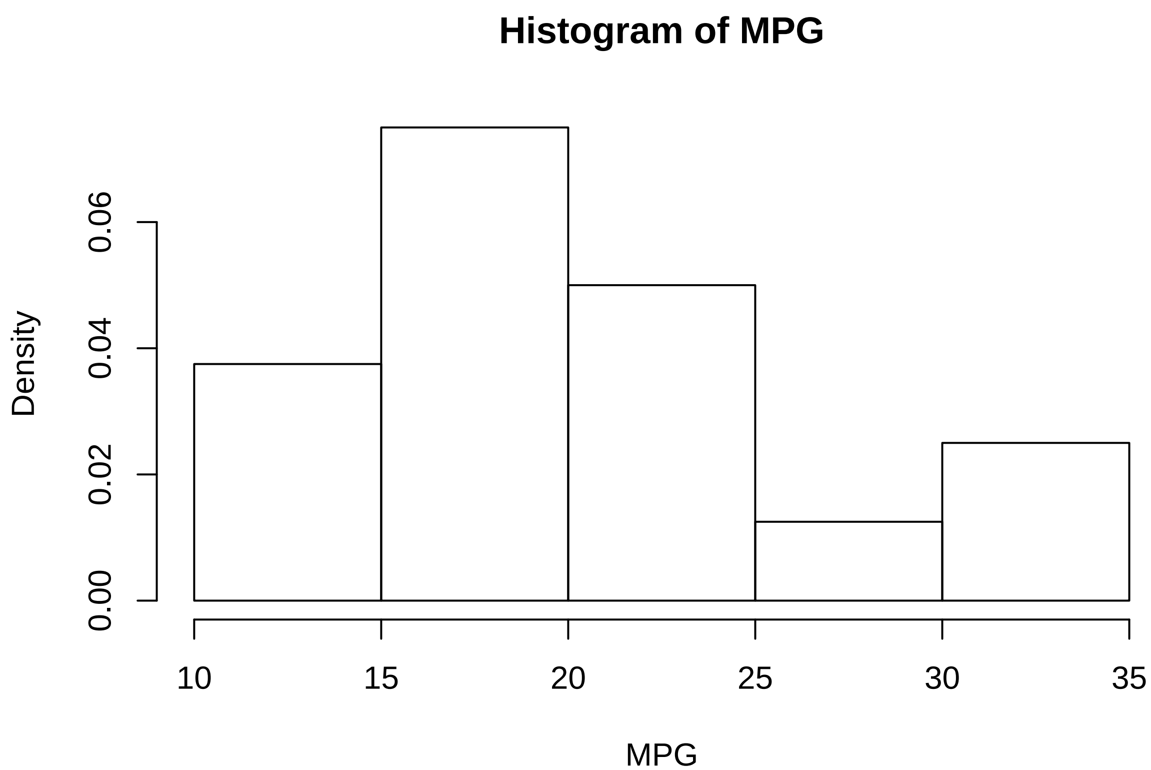
To generate B bootstrap samples, for b in 1, .., B do

1. Sample $x_1, ..., x_n$ with replacement; each sample has a probability of *1/n* of being in the new sample.
2. Calculate $\hat{\theta}_b^*$

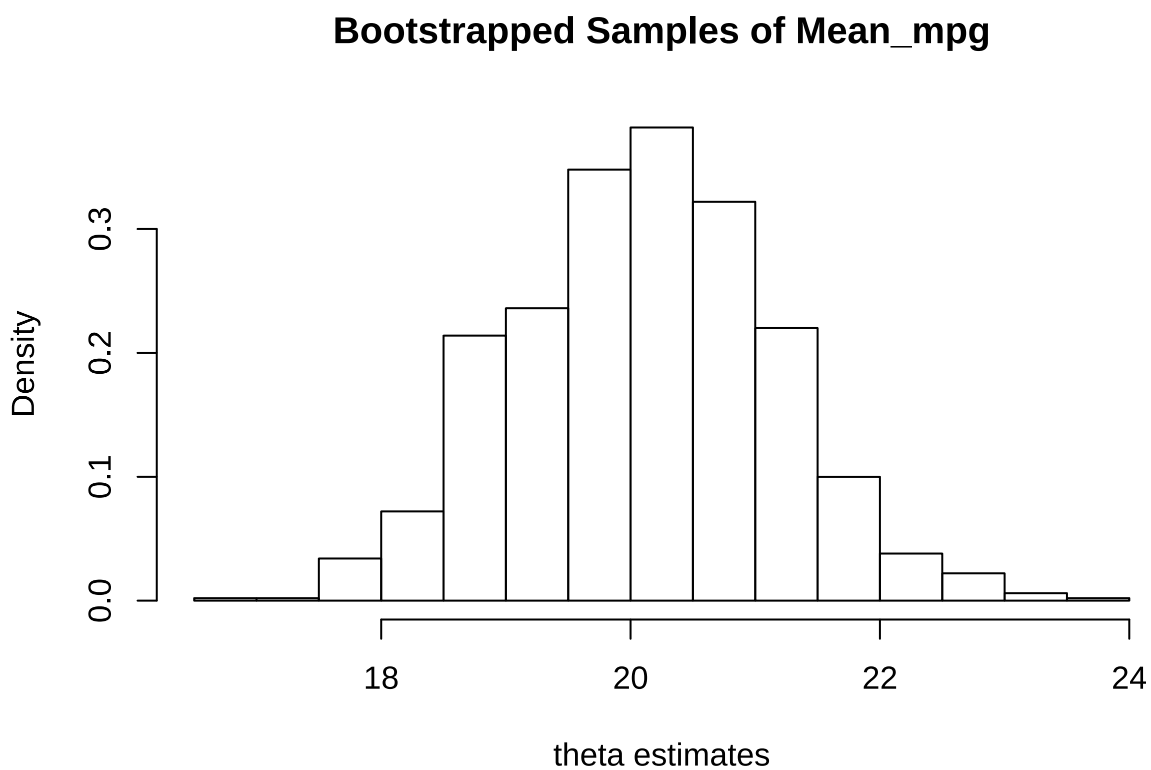
We will then observe the empirical distribution of $\hat{\theta}$, $F_{\hat{\theta}}$.

We will use the `mtcars` data set to illustrate a simple implementation.

```
data("mtcars")
mpg = mtcars$mpg
n = length(mpg)
hist(x = mpg, probability = TRUE, xlab = "MPG", main = "Histogram of MPG")
```




```
B = 1000
results = numeric(B)
for(b in 1:B){
  i = sample(x = 1:n, size = n, replace = TRUE)
  bootSample = mpg[i]
  thetaHat = mean(bootSample)
  results[b] = thetaHat
}

hist(x = results, probability = TRUE,
     main = "Bootstrapped Samples of Mean_mpg",
     xlab = "theta estimates")
```




```r
print(table(i)/n)
```

```
## i
##       2       3       6       7       8      10      11      12      13
## 0.06250 0.03125 0.03125 0.03125 0.06250 0.06250 0.03125 0.06250 0.03125
##      14      15      16      18      19      22      23      24      25
## 0.03125 0.03125 0.03125 0.03125 0.03125 0.09375 0.03125 0.03125 0.06250
##      26      27      28      31      32
## 0.03125 0.03125 0.06250 0.06250 0.03125
```

As a precaution and note on proper use of bootstrap methods, before enbarking on resampling we must ask what variables are *iid* in order to determine a correct bootstrapping approach. Bootstrap methods are *not* a method of generating new data for, say, a regression setting when observed samples are low. In the above example, it is assumed that each observation in the `mpg` data set is indpendent and identically distributed from an unknown distribution f. However, if there were to have existed some autocorrelation structure (as exist in time-series data) then we would need to adjust our resampling methodology. When dealing with time-series data, we will use a method called *block bootsrap*.

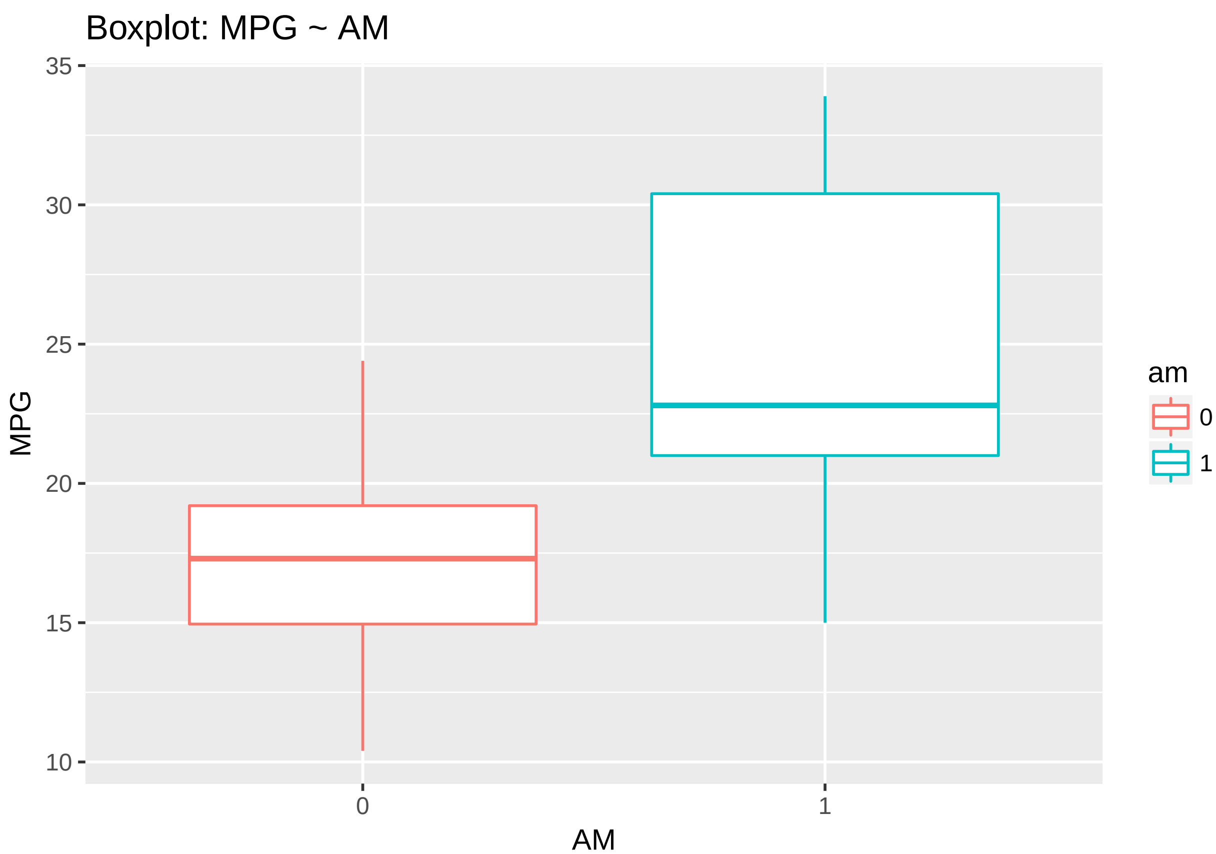
Paired Bootstrapping

Let's continue to work with the `mtcars` data set. Say we wanted to make inferences about the linear regression parameters.

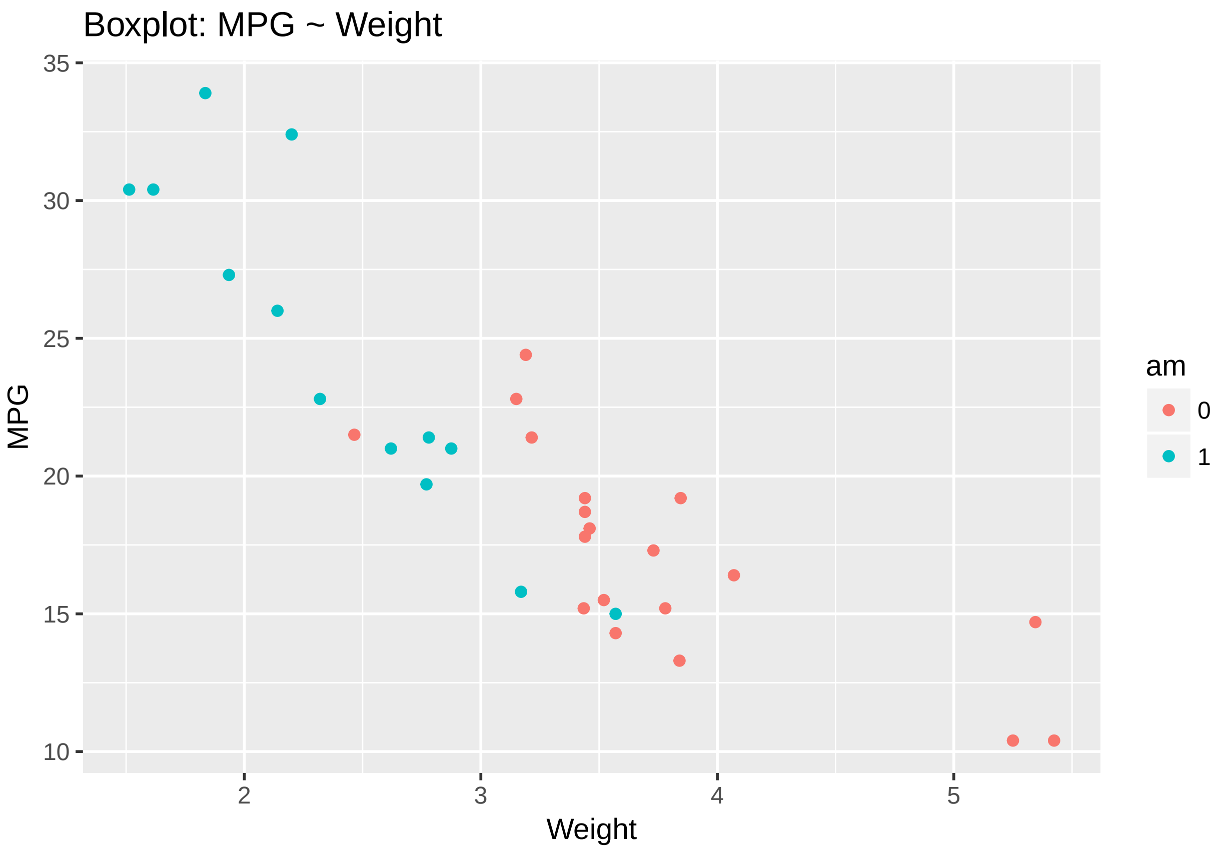
```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked _by_ '.GlobalEnv':
##
##     mpg
```

```r
mtcars$am <- as.factor(mtcars$am)

fit = lm(formula = mpg ~ wt + am, data = mtcars)

qplot(x = as.factor(am), y = mpg, data = mtcars, geom = "boxplot",
      main = "Boxplot: MPG ~ AM", ylab = "MPG", xlab = "AM",
      colour = am)
```




```r
qplot(x = wt, y = mpg, data = mtcars,
      main = "Boxplot: MPG ~ Weight", ylab = "MPG", xlab = "Weight",
      colour = am)
```




```
## see summary of model
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ wt + am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5295 -2.3619 -0.1317  1.4025  6.8782
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.32155    3.05464  12.218 5.84e-13 ***
## wt          -5.35281    0.78824  -6.791 1.87e-07 ***
## am1         -0.02362    1.54565  -0.015    0.988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.098 on 29 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7358
## F-statistic: 44.17 on 2 and 29 DF,  p-value: 1.579e-09
```

```
## see coefficients
beta_int = coefficients(fit)[1]
beta_wt = coefficients(fit)[2]
beta_am = coefficients(fit)[3]
```

```r
n = dim(mtcars)[1]
B = 1000

results = matrix(data = NA, nrow = B, ncol = 3,
                 dimnames = list(NULL, c("Intercept", "wt", "am")))
for(b in 1:B){
  i = sample(x = 1:n, size = n, replace = TRUE)
  temp = mtcars[i,]
  temp_model =  lm(formula = mpg ~ wt + am, data = temp)
  coeff = matrix(data = coefficients(temp_model), ncol = 3)
  if(sum(is.na(coeff)) > 0){
    break
  }
  results[b,] = coeff
}

results <- data.frame(results, check.names = FALSE)

head(results)
```

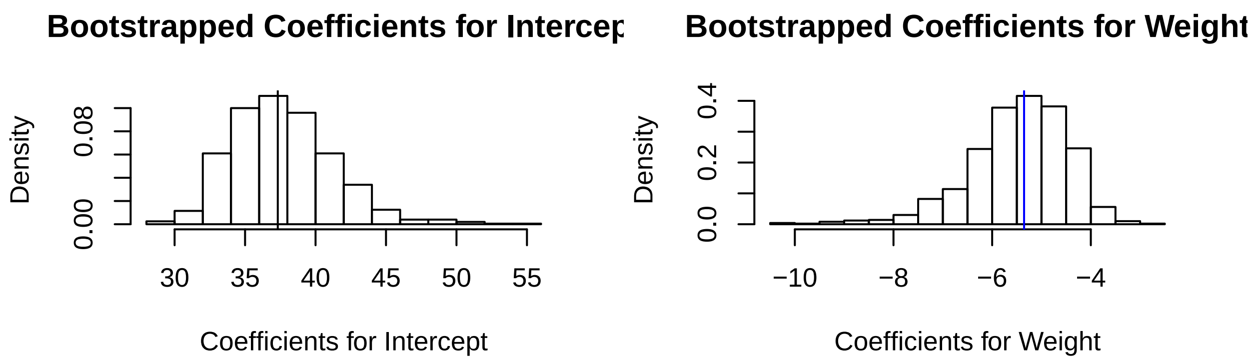
```
##   Intercept        wt         am
## 1  33.05097 -4.227641  0.5568170
## 2  37.85325 -5.435074 -1.6254900
## 3  38.53333 -5.852539 -0.5896856
## 4  39.16575 -5.916732 -0.1280177
## 5  36.96294 -4.942564 -0.3379845
## 6  36.52314 -5.039614 -0.8691914
```

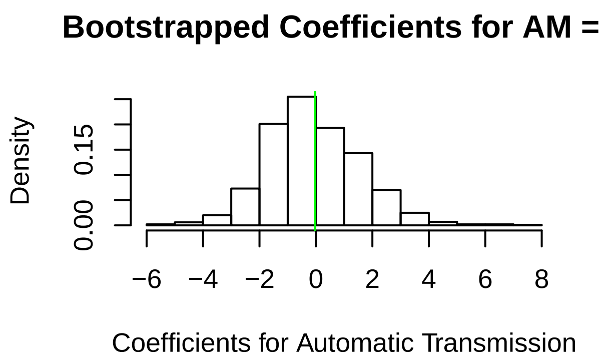
```r
tail(results)
```

```
##      Intercept        wt         am
## 995   40.45304 -6.071098 -1.7976744
## 996   32.77474 -4.208075 -0.3090841
## 997   37.38477 -5.645128  0.6896641
## 998   38.09603 -6.008380  0.5627437
## 999   39.24106 -5.847341 -0.1181249
## 1000  43.90428 -7.038502 -3.6627247
```

```r
boot_int = results[,"Intercept"]
boot_wt = results[,"wt"]
boot_am = results[,"am"]


par(mfrow = c(2,2))
hist(boot_int, main = "Bootstrapped Coefficients for Intercept",
     xlab = "Coefficients for Intercept", probability = TRUE)
abline(v = coefficients(fit)[1], col = "black")


hist(boot_wt, main = "Bootstrapped Coefficients for Weight",
     xlab = "Coefficients for Weight", probability = TRUE)
abline(v = coefficients(fit)[2], col = "blue")


hist(boot_am, main = "Bootstrapped Coefficients for AM = 1",
```

```r
     xlab = "Coefficients for Automatic Transmission", probability = TRUE)
abline(v = coefficients(fit)[3], col = "green")
```





Now we can estimate bias

```r
bias_int = mean(boot_int - beta_int)
print(bias_int)
```

```
## [1] 0.3790739
```

```r
bias_wt = mean(boot_wt - beta_wt)
print(bias_wt)
```

```
## [1] -0.1161098
```

```r
bias_am = mean(boot_am - beta_am)
print(bias_am)
```

```
## [1] -0.05653533
```

```r
## incorportate our bias into the coefficients
## we now have bias corrected coefficients

intercept = beta_int - bias_int
print(intercept)
```

```
## (Intercept)
##    36.94248
```

```r
wt = beta_wt - bias_wt
print(wt)
```

```
##        wt
## -5.236702
```

```r
am = beta_am - bias_am
print(am)
```

```
##        am1
## 0.03292011
```

Define Bias as $Bias(\theta) = E[\theta^*] - \theta$, where in our scenario we have $Bias(\hat{\theta}) = E[\hat{\theta}^*] - \hat{\theta}$. Our bootstrap bias corrected estimates are then $\hat{\theta}_{BC} = \hat{\theta} - Bias(\hat{\theta})$.

Another method for applying the bootstrap approach to building an empirical distribution of $\hat{\beta}$ is to bootstrap the residuals. However, bootstrapping the cases is often more robust when there are doubts about a constant variance for the residuals, such as heteroskedasticity. Additionally, paired bootstrap more resembles the original data generation mechanisms.