MLR and Bootstrap

Cheng Peng

West Chester University

1. SAT Scores

The working data set is at:

https://raw.githubusercontent.com/pengATwcupa/STA321SP2020/master/sat-scores.txt

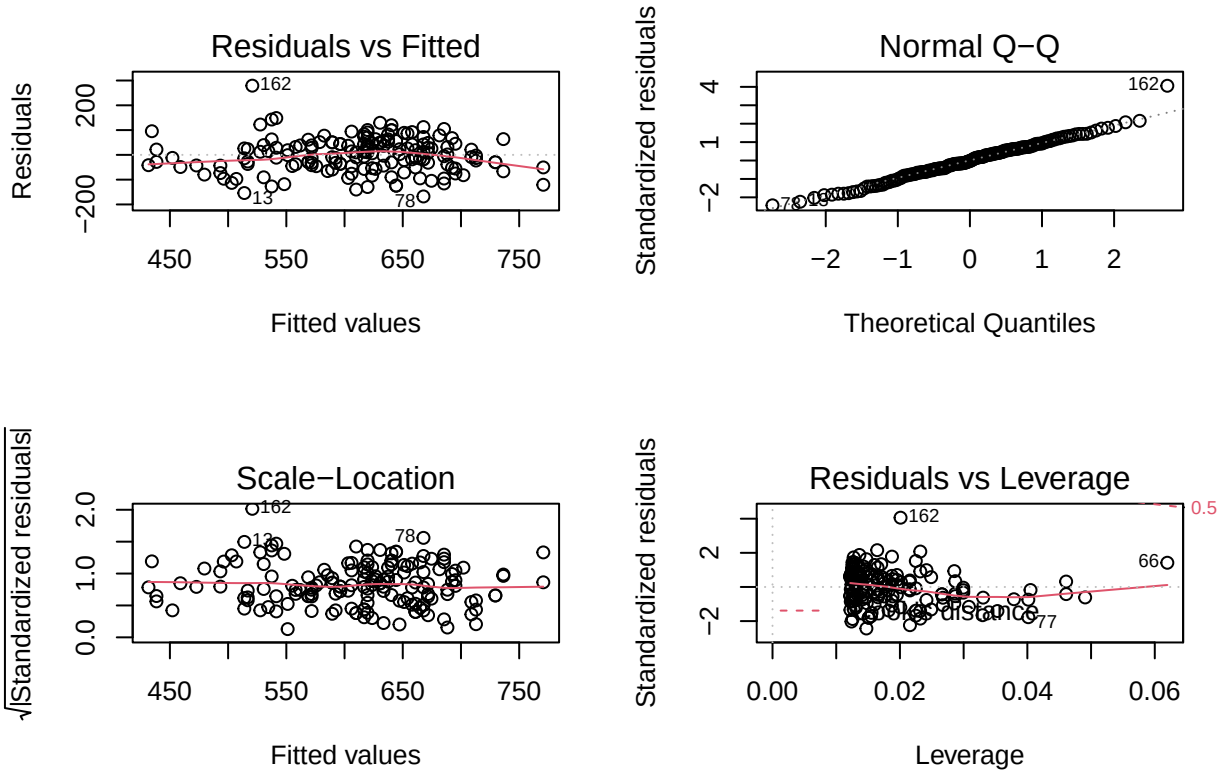
The data set has three variables: Verbal score, Math score and gender. 162 observations are included in the data set. This is a simple and almost perfect data.

2. Fit A Linear Regression Model to the Data

```
sat = read.table("https://raw.githubusercontent.com/pengATwcupa/STA321SP2020/master/sat-scores.txt", hea
# head(sat)
m=lm(Math~Verbal+Sex, data=sat)
summary(m)
```

```
##
## Call:
## lm(formula = Math ~ Verbal + Sex, data = sat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -167.786  -43.444   -2.023   44.512  279.214
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 184.58164   34.06782   5.418 2.19e-07 ***
## Verbal        0.68613    0.05513  12.446  < 2e-16 ***
## SexM         37.21856   10.93993   3.402 0.000846 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 69.49 on 159 degrees of freedom
## Multiple R-squared:  0.5047, Adjusted R-squared:  0.4985
## F-statistic: 81.02 on 2 and 159 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(m)
```



Residual plots do not indicate any serious violations to the model assumption. We can tell some story about the model based on the summarized statistics.

3. Bootstrapping Linear Regression - Non-parametric Approaches

Two Types of Bootstrap methods are commonly used in regression modeling.

3.1. Bootstrap Sample and Bootstrap Confidence Intervals

A bootstrap sample is a smaller sample that is "bootstrapped" from a larger sample.

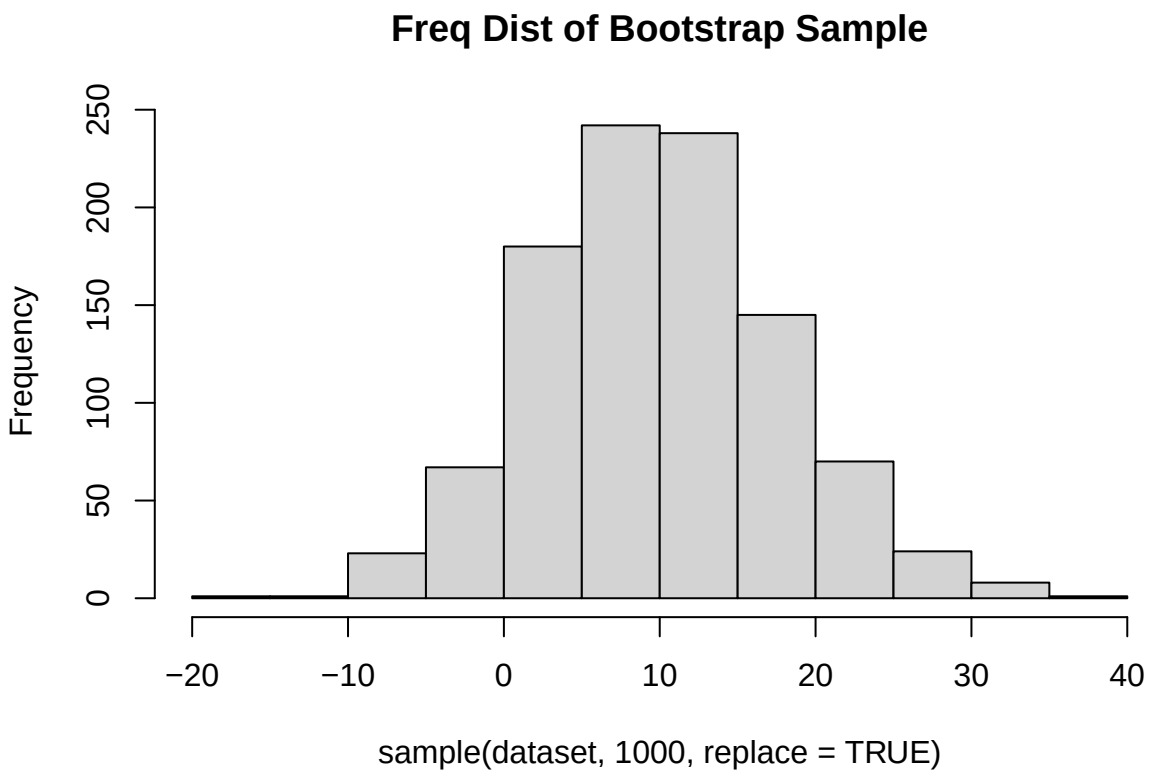
Bootstrapping is a type of re-sampling where large numbers of smaller samples of the same size are repeatedly drawn, with replacement, from a single original sample.

Example 1. Let's simulate a set of 1000 values from N(10, 8). We want to a bootstrap sample with size 1000 from this data set. I use R do this types of samples in the following

```
dataset = rnorm(1000, mean =10, sd = 8)
sort(sample(dataset,1000, replace = TRUE))[1:50]
```

```
##  [1] -17.583111 -14.870702 -14.460782 -10.961616 -10.961616 -10.644675
##  [7]  -9.186749  -9.186749  -7.977908  -7.903253  -7.903253  -7.845748
## [13]  -7.730297  -7.201350  -6.534340  -6.366029  -6.366029  -6.350118
## [19]  -6.349127  -5.843693  -5.352601  -5.281811  -5.221280  -5.148030
## [25]  -4.876696  -4.536919  -4.536919  -4.485525  -4.380402  -4.185517
## [31]  -3.709417  -3.658660  -3.565087  -3.406079  -3.406008  -3.343281
## [37]  -3.343281  -3.343281  -3.022915  -2.906025  -2.759291  -2.710130
## [43]  -2.402805  -2.147102  -2.062678  -2.049195  -2.041426  -2.041426
## [49]  -1.988128  -1.988128
```

```
hist(sample(dataset,1000, replace = TRUE),breaks = 15, main="Freq Dist of Bootstrap Sample")
```

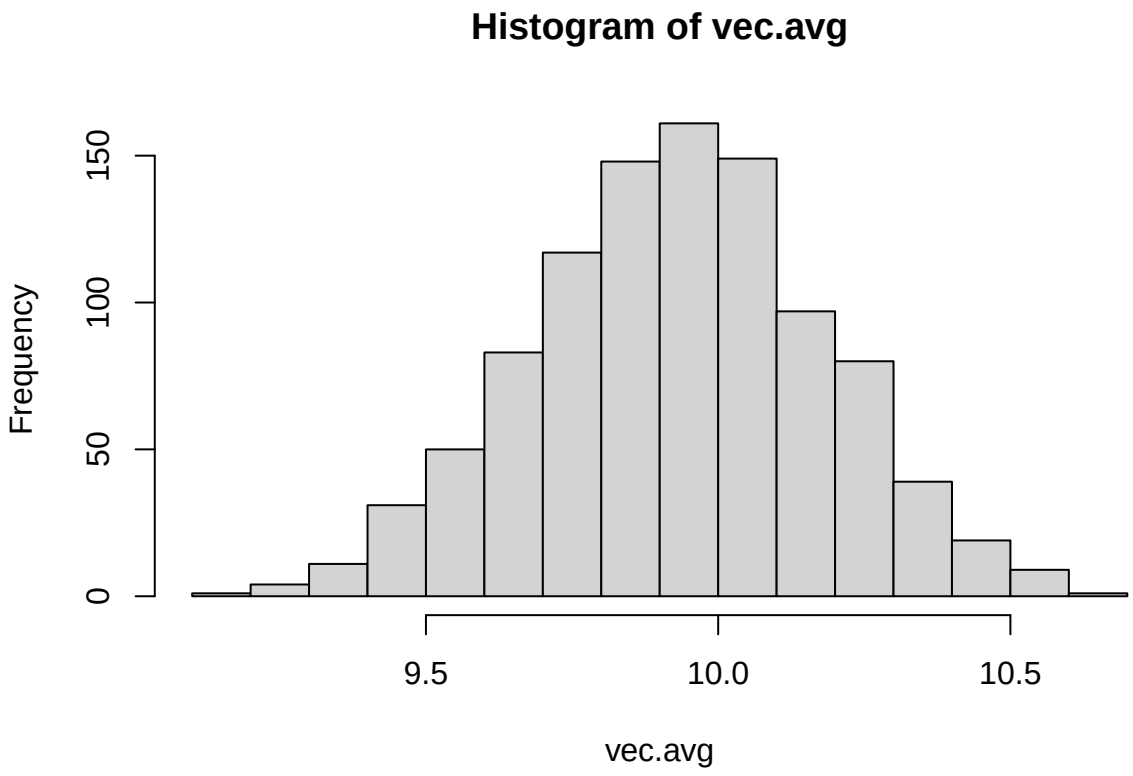


The histograms indicates that the sampling distribution of the Bootstrap samples is close to a normal distribution.

The bootstrap percentile method is a way to calculate confidence intervals for bootstrapped samples.

With the simple method, a certain percentage (e.g. 5% or 10%) is trimmed from the lower and upper end of the sample statistic (e.g. the mean or standard deviation). Which number you trim depends on the confidence interval you're looking for. For example, a 90% confidence interval would generate a 100% − 90% = 10% trim (i.e. 5% from both ends). Or, put another (slightly more technical) way, you can get a 90% confidence interval by taking the lower bound 5% and upper bound 95% quantiles of the B replication $T_1, T_2, \cdots, T_B$. B = bootstrap replicates.

```
vec.avg = NULL
B=1000
for(i in 1:B){
   vec.avg[i]  = mean(sample(dataset,1000, replace = TRUE))
}
hist(vec.avg, breaks =15)
```



3.2. Sampling Cases (Observations)

This is a naive Bootstrap sampling, we take B (usually a large number > 1000) random sets of records (observations) with replacement from the original data set and each subset set has the same size as the original data set. We fit the same regression model to each of the B sets and obtain B regression models. For linear regression model

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon$$

For b-th bootstrap sample ($b = 1, 2, \cdots, B$), we have fitted regression line

$$y = \hat{\beta}_0^{*(b)} + \hat{\beta}_1^{*(b)} x_1 + \cdots + \hat{\beta}_k^{*(b)} x_k$$

Then we have a vector of bootstrap estimate for each regression coefficient. That is,

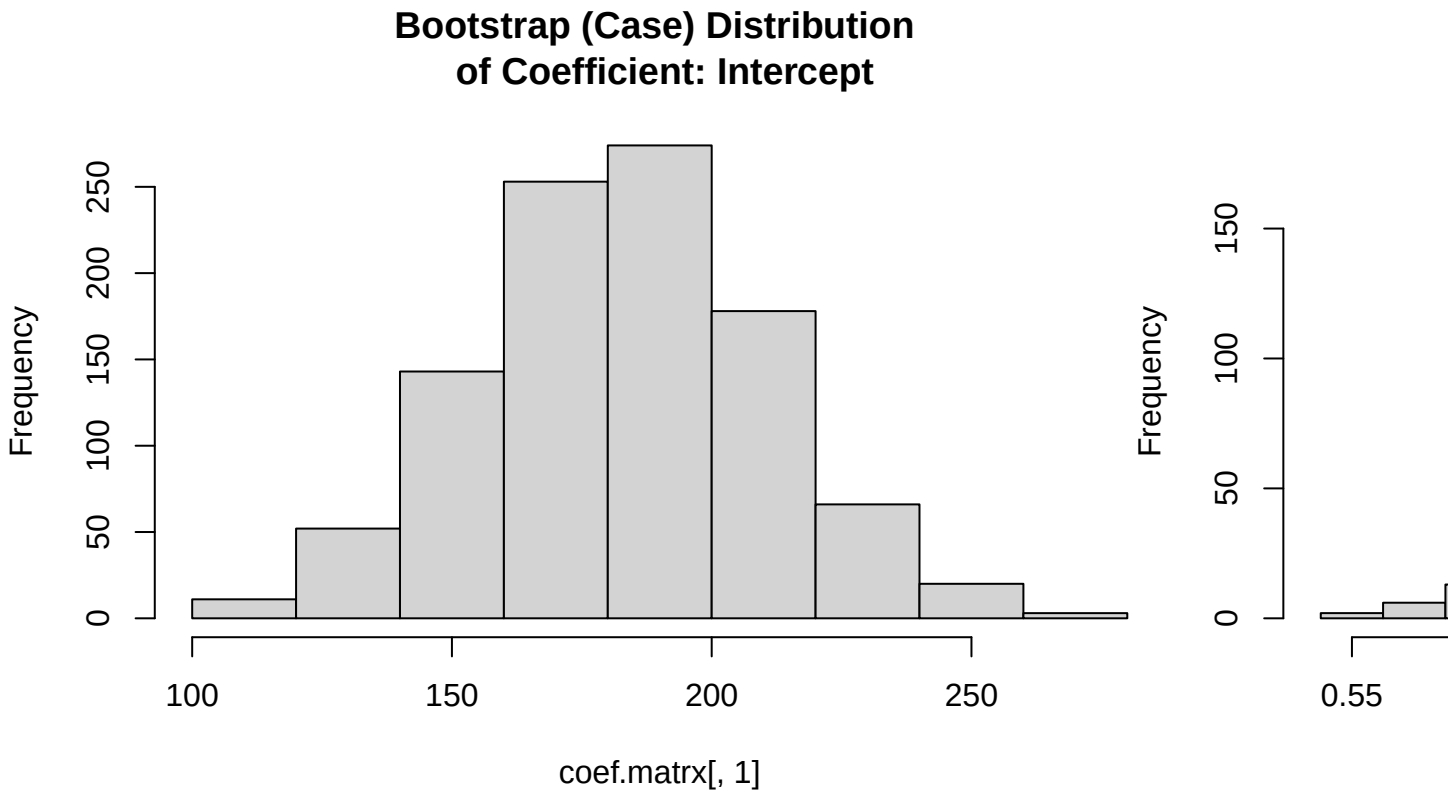
$$\hat{\beta}_i^* = (\hat{\beta}_i^{*(1)}, \hat{\beta}_i^{*(2)}, \cdots, \hat{\beta}_k^{*(B)})$$

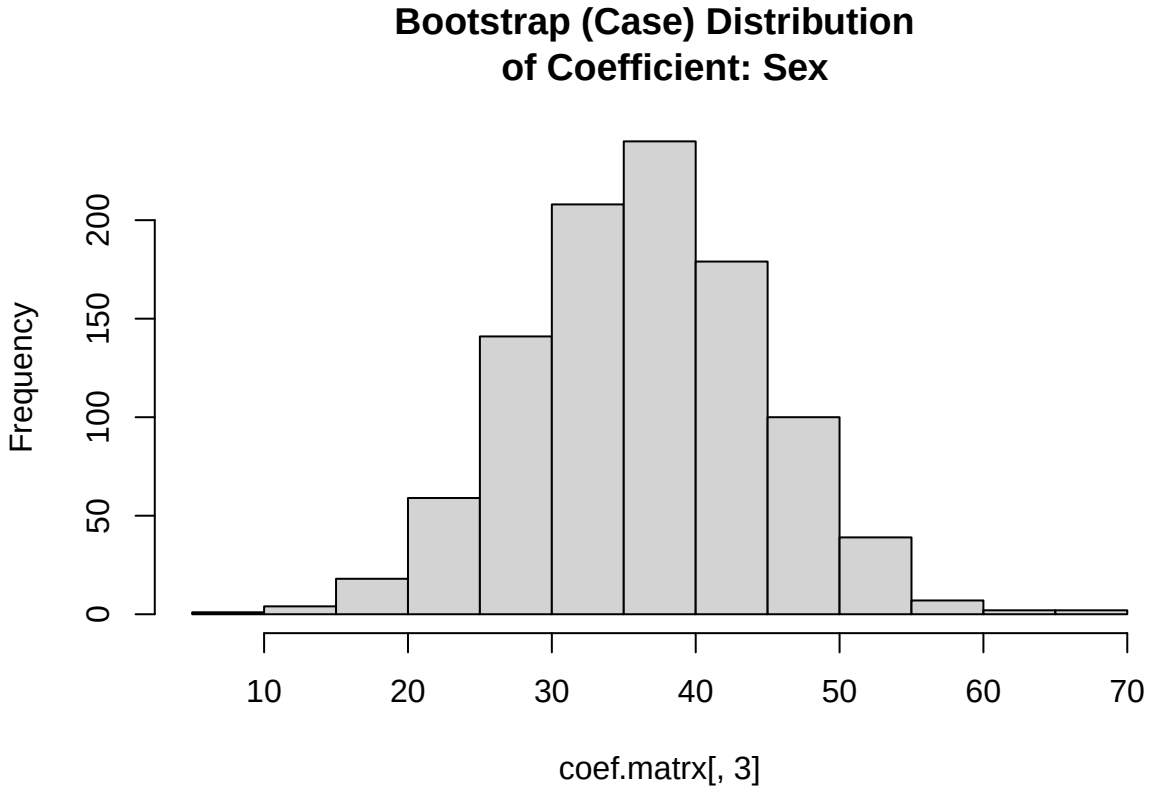
for $i = 0, 1, \cdots, k$. The bootstrap sampling distribution of $\hat{\beta}_i$ can be approximated by the empirical distribution of $\hat{\beta}_i^*$.

Example We continue to use the above SAT score example. The following R function will the Bootstrap regression by sampling the cases.

```
sat = read.table("https://raw.githubusercontent.com/pengATwcupa/STA321SP2020/master/sat-scores.txt", hea
###
My.lm01=function(dataset, B, histogram=TRUE){
    # dataset = input data set
    # B = number of bootstrap samples
    n0=dim(dataset)[1]
    var.name = c("Intercept",names(dataset)[-1])
    col.num=length(names(dataset))      ## number of variables = number of parameters
    row.num = B                         ## each bootstrap coefficients will be saved in a row
```

```
    coef.matrx = matrix(0,ncol=col.num, nrow=B)    # bootstrap coef matrix
    colnames(coef.matrx) = var.name
    for (i in 1:B){
         boot.data = unique(dataset[sample(1:n0, n0, replace=TRUE),])  ## Bootstrap data set
         boot.m = lm(Math~Verbal+Sex, data = boot.data)      ## Bootstrap model
         coef.matrx[i,] = coef(boot.m)             ## save the bootstrap coef to the matrix
    }
    ## original linear regression model
    m0 = lm(Math~Verbal+Sex, data = dataset)
    cof.m0=summary(m0)$coef
    q.025=function(x) quantile(x, 0.025)
    q.975=function(x) quantile(x, 0.975)
    Boot.LCI.025=apply(coef.matrx,2,q.025)
    Boot.UCI.975=apply(coef.matrx,2,q.975)
    new.coef=round(cbind(cof.m0, cbind(BT01.LCI.025=Boot.LCI.025, BT01.UCI.975=Boot.UCI.975)),4)
    if(histogram==TRUE){
       hist(coef.matrx[,1], main=paste("Bootstrap (Case) Distribution \n of Coefficient:",var.name[1]))
       hist(coef.matrx[,2], main=paste("Bootstrap (Case) Distribution \n of Coefficient:",var.name[2]))
       hist(coef.matrx[,3], main=paste("Bootstrap (Case) Distribution \n of Coefficient:",var.name[3]))
    }
    list(boot.coef=coef.matrx, coefficient=new.coef) # two types of outputs vailable from the function
}
My.lm01(dataset=sat, B=1000, histogram=TRUE)$coefficient
```






```
##             Estimate Std. Error t value Pr(>|t|) BT01.LCI.025 BT01.UCI.975
## (Intercept) 184.5816    34.0678  5.4181    0e+00     127.9711     239.5602
## Verbal        0.6861     0.0551 12.4457    0e+00       0.6045       0.7750
## SexM         37.2186    10.9399  3.4021    8e-04      20.3594      52.5302
```

3.3. Re-sampling the Errors (with Fixed Covariates)

The whole idea is to fix all covariates and find the fitted value of y with corresponding covariates (explanatory variables) and then random assign a bootstrap error take from the residuals.

We This method assumes that the iid residuals but does not assume the normal distribution!

Following are steps of Bootstrap algorithm (assuming taking B bootstrap samples):

- Estimate the regression coefficients for the original sample, and calculate the fitted value $\widehat{y}_i$ and residual e_i for each observation.
- Select b^{th} bootstrap samples of the residuals: we will denote these bootstrapped residuals as $\{e_1^{*(b)}, e_2^{*(b)}, \cdots, e_n^{*(b)}\}$. Then, the b^{th} bootstrapped $\widehat{y}_i^{*(b)}$ is defined to be $\widehat{y}_i + e_i^{*(b)}$, for $i = 1, \cdots, n$ and $b = 1, 2, \cdots, B$. The fitted values $\widehat{y}_i = \widehat{\beta}_0 + \sum_{k=1}^{p} \widehat{\beta}_k x_{ki}$ are obtained from the original regression.
- The new bootstrap data sets: For illustration purpose, we only write b^{th} Bootstrap data set in the following matrix

$$\begin{pmatrix} \widehat{y}_1^{*(b)} & x_{11} & x_{21} & \cdots & x_{p1} \\ \widehat{y}_2^{*(b)} & x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{y}_n^{*(b)} & x_{1n} & x_{2n} & \cdots & x_{pn} \end{pmatrix}$$

where

$$\hat{y}_i^{*(b)} = \hat{\beta}_0 + \sum_{k=1}^{p} \hat{\beta}_k x_{ki} + e_i^{*(b)}$$

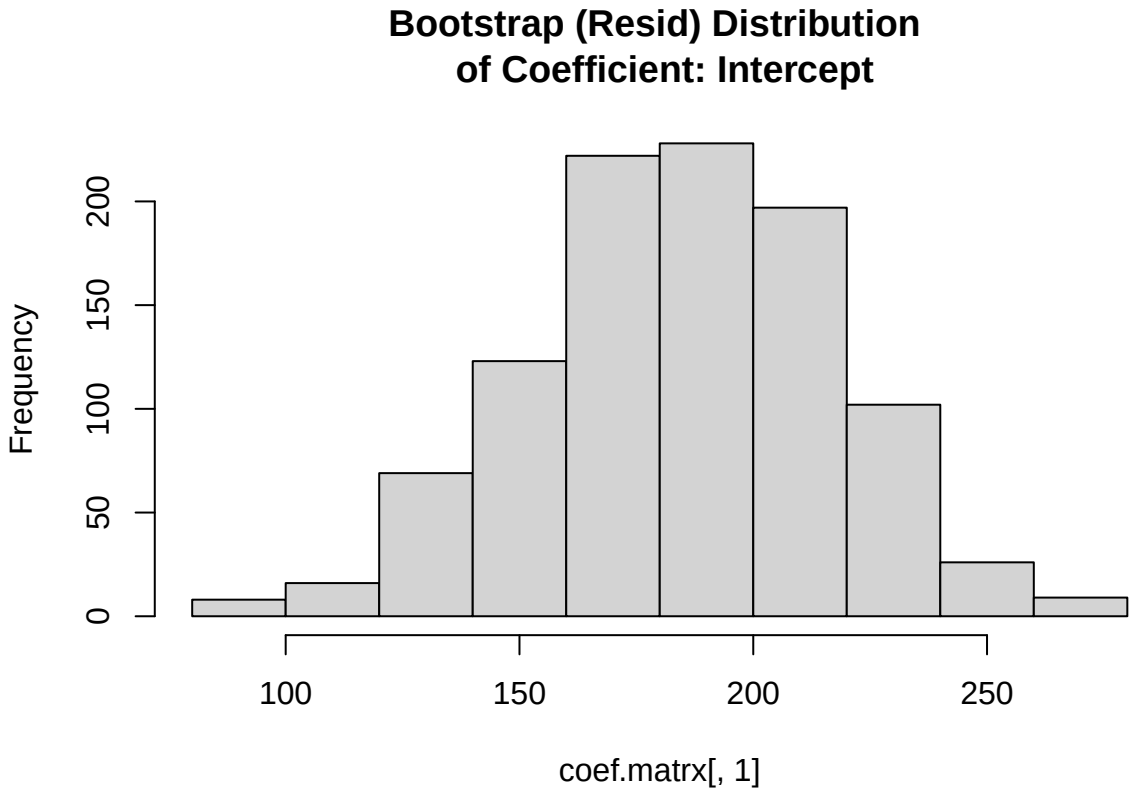
- Fit the the final model to each of the B bootstrap samples and obtain B linear regression models. The fitted Bootstrap linear regression models have the following form

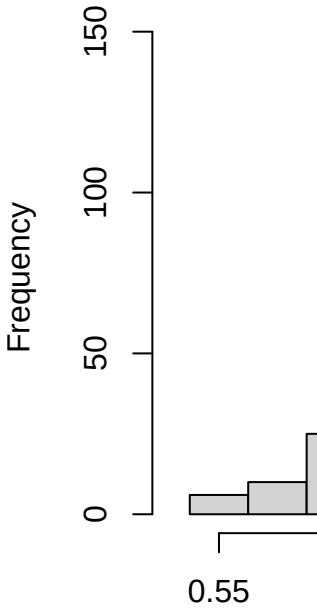
$$\hat{y}_i^{*(b)} = \hat{\beta}_0^{*(b)} + \sum_{k=1}^{p} \hat{\beta}_k^{*(b)} x_{ki}$$

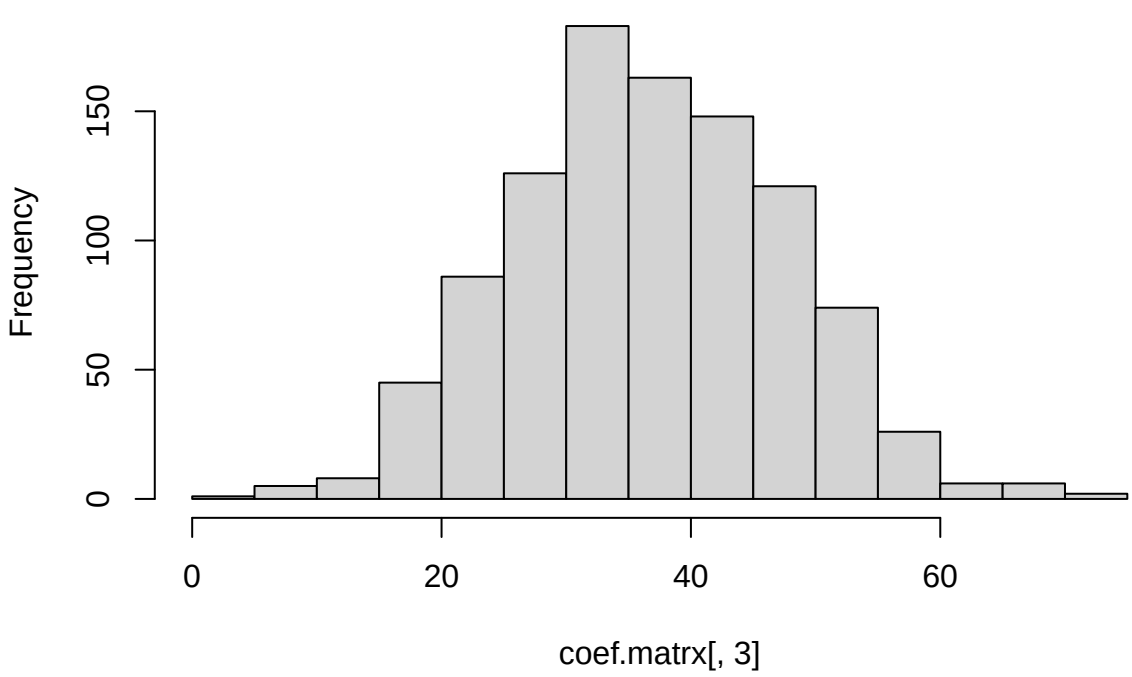
* For each regression coefficient β_i, for $i = 0, 1, \cdots, p$, we obtain B bootstrap estimates, denoted by $\hat{\beta}_i^{*(b)}$ for $b = 1, 2, \cdots, B$. The inference about β_i will be based on the corresponding Bootstrap estimates $$

Example (Continued)

```
My.lm02=function(dataset, B, histogram=TRUE){
    # dataset = input data set
    # B = number of bootstrap samples
    n0=dim(dataset)[1]
    var.name = c("Intercept",names(dataset)[-1])
    col.num=length(names(dataset))     ## number of variables = number of parameters
    row.num = B                        ## each bootstrap coefficients will be saved in a row
    coef.matrx = matrix(0,ncol=col.num, nrow=B)   # bootstrap coef matrix
    colnames(coef.matrx) = var.name
    ##
    m0 = lm(Math~Verbal+Sex, data = dataset)
    original.resid=resid(m0)
    original.fit = fitted(m0)
    Verbal=dataset$Verbal
    Sex=dataset$Sex
    ##
    for (i in 1:B){
         boot.resid = sample(original.resid, n0, replace=TRUE)  ## Bootstrap residuals
         boot.Math = original.fit + boot.resid  ## bootstrap YYY
         boot.m = lm(boot.Math~Verbal+Sex)      ## Bootstrap model
         coef.matrx[i,] = coef(boot.m)            ## save the bootstrap coef to the matrix
    }
    ## original linear regression model
    # m0 = lm(Math~Verbal+Sex, data = dataset)
    cof.m0=summary(m0)$coef
    q.025=function(x) quantile(x, 0.025)
    q.975=function(x) quantile(x, 0.975)
    Boot.LCI.025=apply(coef.matrx,2,q.025)
    Boot.UCI.975=apply(coef.matrx,2,q.975)
    new.coef=round(cbind(cof.m0, cbind(BT02.LCI.025=Boot.LCI.025, BT02.UCI.975=Boot.UCI.975)),4)
    if(histogram==TRUE){
       hist(coef.matrx[,1], main=paste("Bootstrap (Resid) Distribution \n of Coefficient:",var.name[1]))
       hist(coef.matrx[,2], main=paste("Bootstrap (Resid) Distribution \n of Coefficient:",var.name[2]))
       hist(coef.matrx[,3], main=paste("Bootstrap (Resid) Distribution \n of Coefficient:",var.name[3]))
    }
    list(boot.coef=coef.matrx, coefficient=new.coef) # two types of outputs vailable from the function
}
My.lm02(dataset=sat, B=1000, histogram=TRUE)$coefficient
```








```
##             Estimate Std. Error t value Pr(>|t|) BT02.LCI.025 BT02.UCI.975
## (Intercept) 184.5816    34.0678  5.4181    0e+00     121.6808     244.1202
## Verbal        0.6861     0.0551 12.4457    0e+00       0.5896       0.7901
## SexM         37.2186    10.9399  3.4021    8e-04      16.7542      57.6850
```

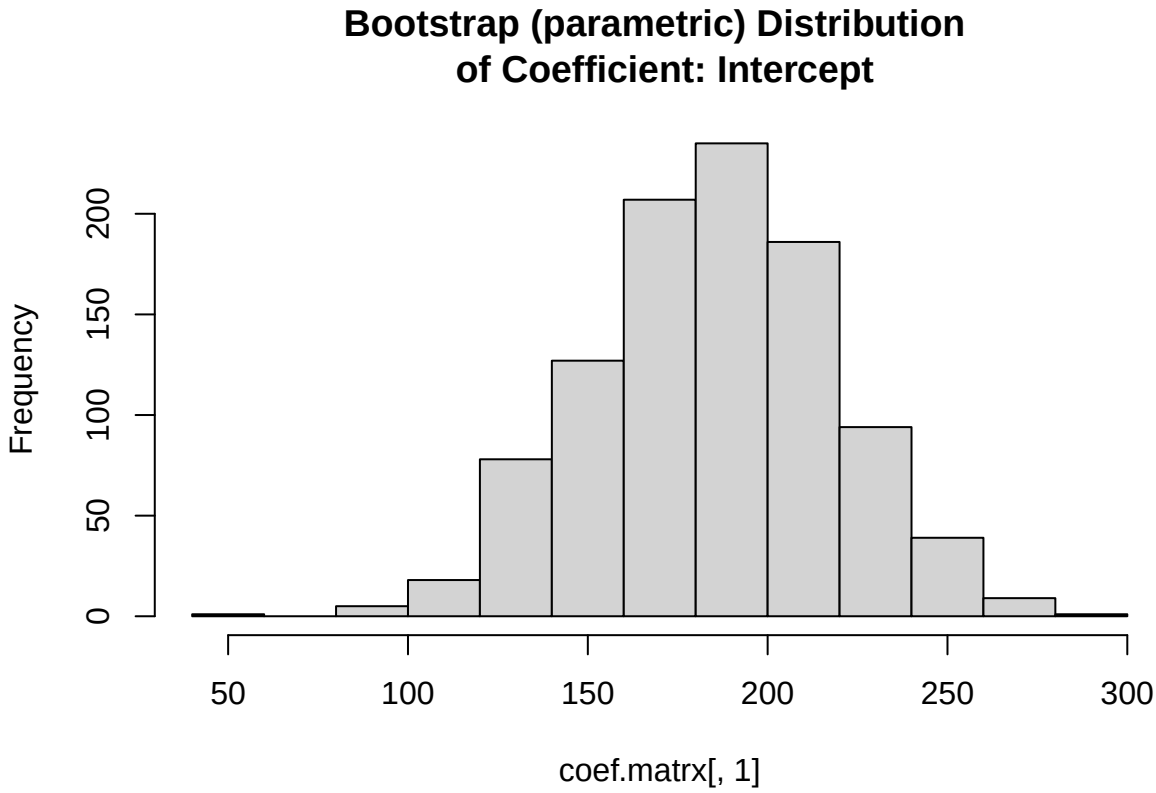
3.4. Parametric Bootstrap sampling

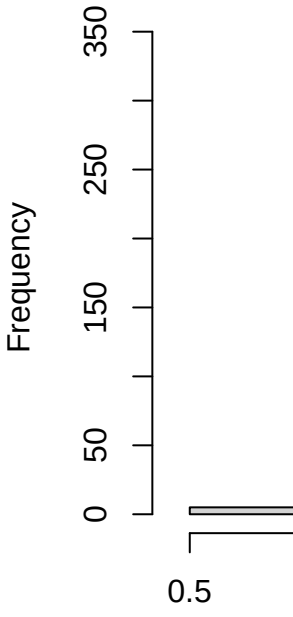
This bootstrap is similar to *Monte Carlo* simulation. We assume the residuals are normally distributed with a constant variance. Based on this assumption, we estimated the common variance from the residuals and then **generate residuals from the normal distribution** and assign the generated values to the fitted values based on the original data set.

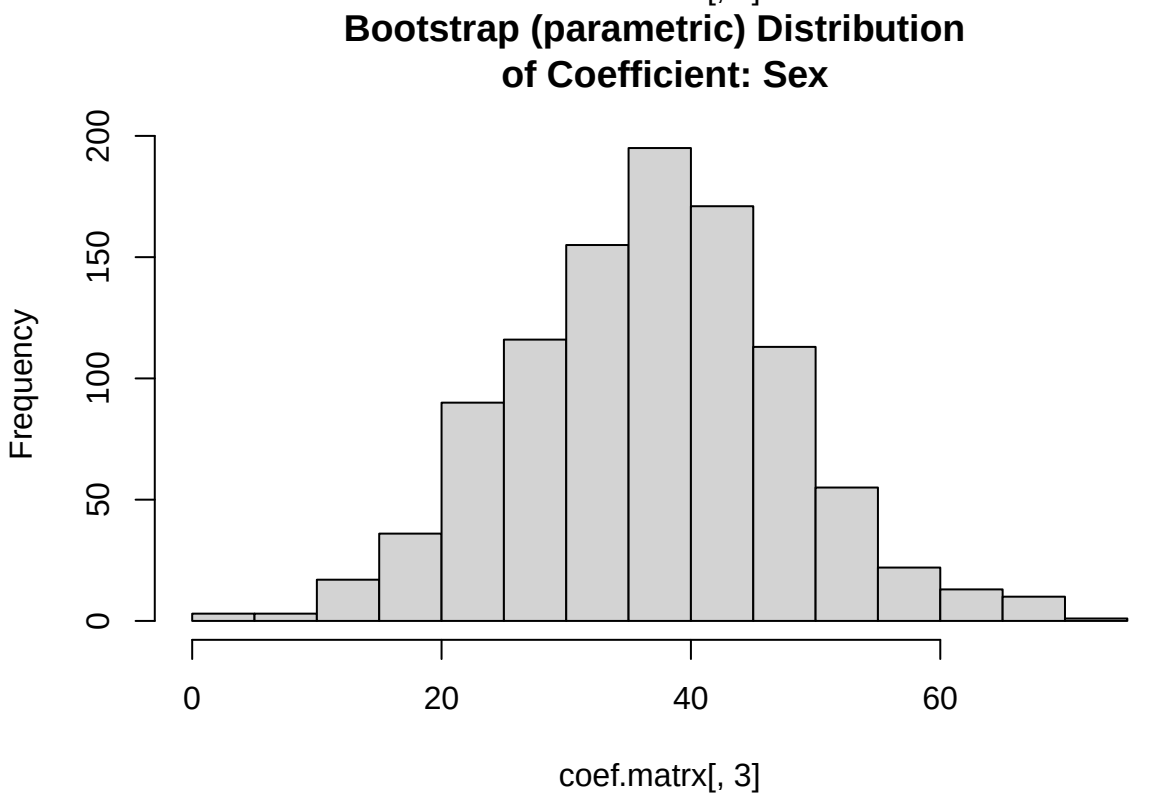
The steps of this Bootstrap regression is almost identical to the above method.

We only modify the above code to get a parametric bootstrap regression model.

```
My.lm03=function(dataset, B, histogram=TRUE){
    # dataset = input data set
    # B = number of bootstrap samples
    n0=dim(dataset)[1]
    var.name = c("Intercept",names(dataset)[-1])
    col.num=length(names(dataset))     ## number of variables = number of parameters
    row.num = B                        ## each bootstrap coefficients will be saved in a row
    coef.matrx = matrix(0,ncol=col.num, nrow=B)   # bootstrap coef matrix
    colnames(coef.matrx) = var.name
    ##
    m0 = lm(Math~Verbal+Sex, data = dataset)
    original.resid=resid(m0)
    sig = sd(original.resid)
    original.fit = fitted(m0)
    Verbal=dataset$Verbal
    Sex=dataset$Sex
    ##
    for (i in 1:B){
         boot.resid = rnorm(n0, mean=0, sd = sig)    ## Parametric Bootstrap residuals
         boot.Math = original.fit + boot.resid       ## bootstrap YYY
         boot.m = lm(boot.Math~Verbal+Sex)           ## Bootstrap model
         coef.matrx[i,] = coef(boot.m)               ## save the bootstrap coef to the matrix
    }
    ## original linear regression model
    # m0 = lm(Math~Verbal+Sex, data = dataset)
    cof.m0=summary(m0)$coef
    q.025=function(x) quantile(x, 0.025)
    q.975=function(x) quantile(x, 0.975)
    Boot.LCI.025=apply(coef.matrx,2,q.025)
    Boot.UCI.975=apply(coef.matrx,2,q.975)
    new.coef=round(cbind(cof.m0, cbind(BT03.LCI.025=Boot.LCI.025, BT03.UCI.975=Boot.UCI.975)),4)
    if(histogram==TRUE){
       hist(coef.matrx[,1], main=paste("Bootstrap (parametric) Distribution \n of Coefficient:",var.name
       hist(coef.matrx[,2], main=paste("Bootstrap (parametric) Distribution \n of Coefficient:",var.name
       hist(coef.matrx[,3], main=paste("Bootstrap (parametric) Distribution \n of Coefficient:",var.name
    }
    list(boot.coef=coef.matrx, coefficient=new.coef) # two types of outputs vailable from the function
}
My.lm03(dataset=sat, B=1000, histogram=TRUE)$coefficient
```








```
##             Estimate Std. Error t value Pr(>|t|) BT03.LCI.025 BT03.UCI.975
## (Intercept) 184.5816    34.0678  5.4181    0e+00     120.4735     249.5753
## Verbal        0.6861     0.0551 12.4457    0e+00       0.5785       0.7961
## SexM         37.2186    10.9399  3.4021    8e-04      16.3074      58.4119
```

4. Potential Issue in Knitting HTML: Update Current Version of Pandoc

Following code updates the old version of Pandoc.

```
# Download pandoc 2.7.1 built with ghc-8.6.4, and instruct
# RStudio + rmarkdown to use it.

local({

    # The directory where Pandoc will be extracted. Feel free
    # to adjust this path as appropriate.
    dir <- "~/rstudio-pandoc"

    # The version of Pandoc to be installed.
    version <- "2.7.1"

    # Create and move to the requested directory.
    dir.create(dir, showWarnings = FALSE, recursive = TRUE)
    owd <- setwd(dir)
    on.exit(setwd(owd), add = TRUE)

    # Construct path to pandoc.
    root <- "https://s3.amazonaws.com/rstudio-buildtools"
    suffix <- sprintf("pandoc-%s-windows-x86_64.zip", version)
    url <- file.path(root, "pandoc-rstudio", version, suffix)

    # Download and extract pandoc.
    file <- basename(url)
    utils::download.file(url, destfile = file)
    utils::unzip(file)
    unlink(file)

    # Write .Renviron to update the version of Pandoc used.
    entry <- paste("RSTUDIO_PANDOC", shQuote(path.expand(dir)), sep = " = ")
    contents <- if (file.exists("~/.Renviron")) readLines("~/.Renviron")
    filtered <- grep("^RSTUDIO_PANDOC", contents, value = TRUE, invert = TRUE)
    amended <- union(filtered, entry)
    writeLines(amended, "~/.Renviron")

    # Report change to the user.
    writeLines("Updated .Renviron:\n")
    writeLines(amended)
    writeLines("\nPlease restart RStudio for these changes to take effect.")

})
```