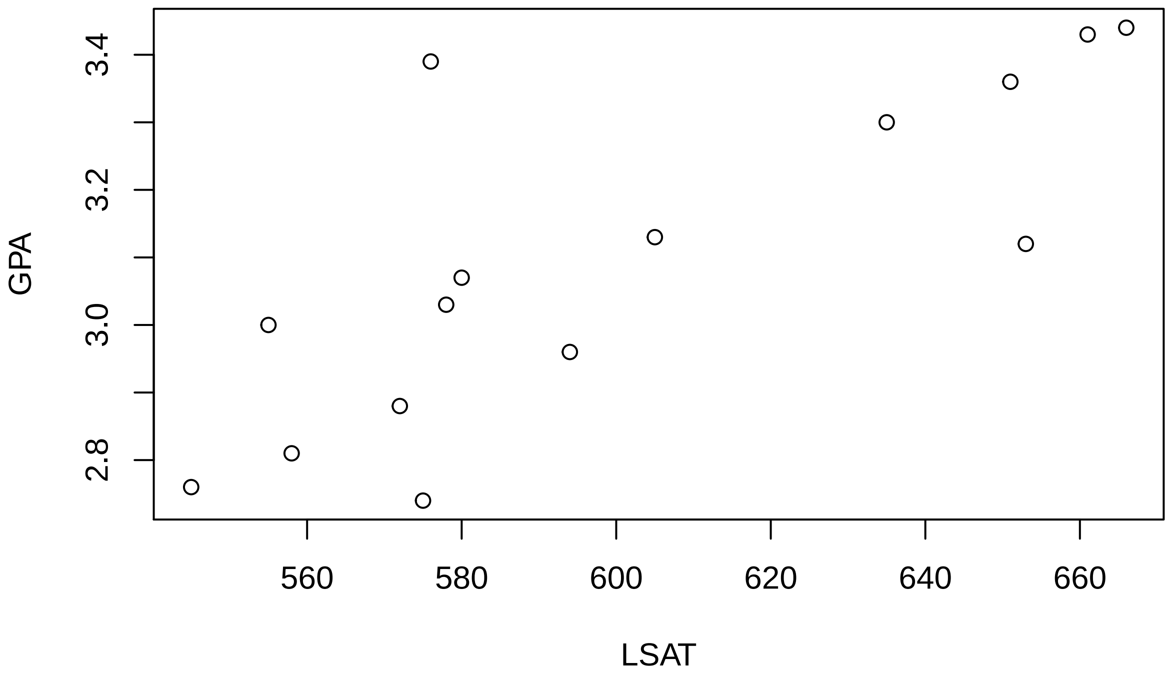
Statistical Learning - Homework 4 (Applied)

James Hahn

Question 5

a)

```
library(bootstrap)
data(law)
plot(law)
```



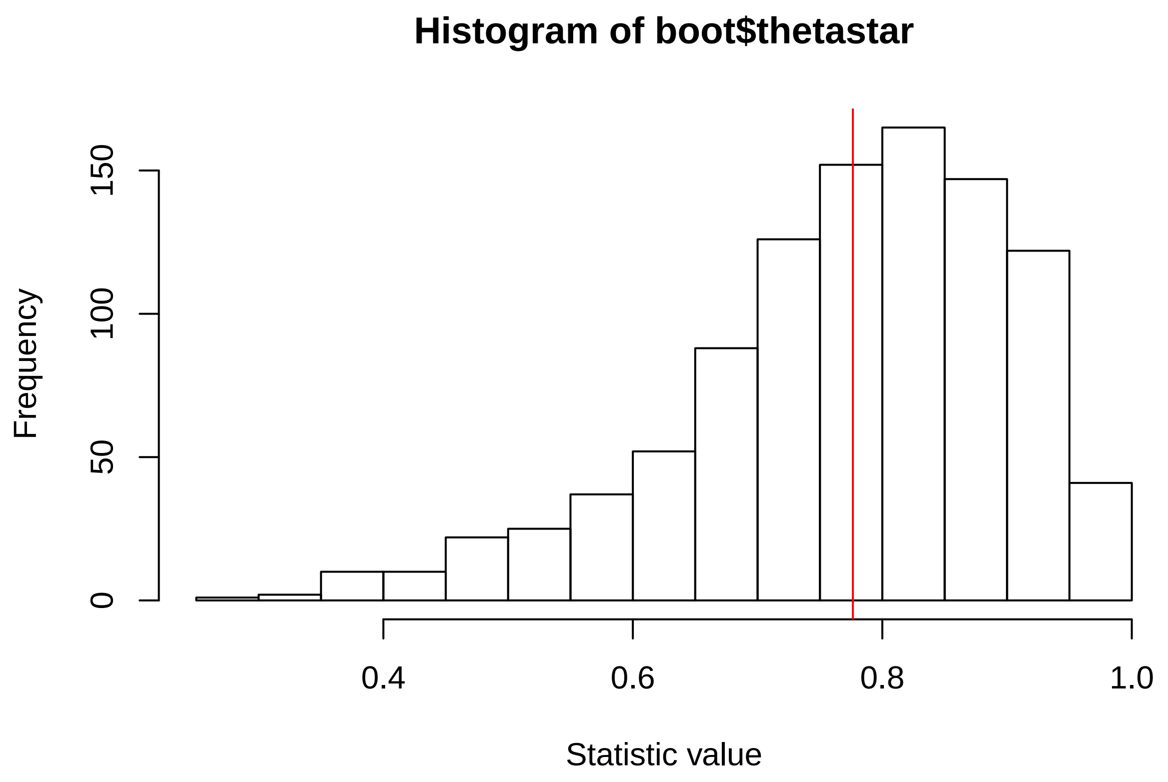
```
lawCor <- cor(law)[2] # grab correlation in 1st row, 2nd column representing LSAT and GPA
lawCor
```

```
## [1] 0.7763745
```

Yes, there appears to be a strong relationship. The correlation between GPA and the LSAT is 0.7764, which is close to 1.

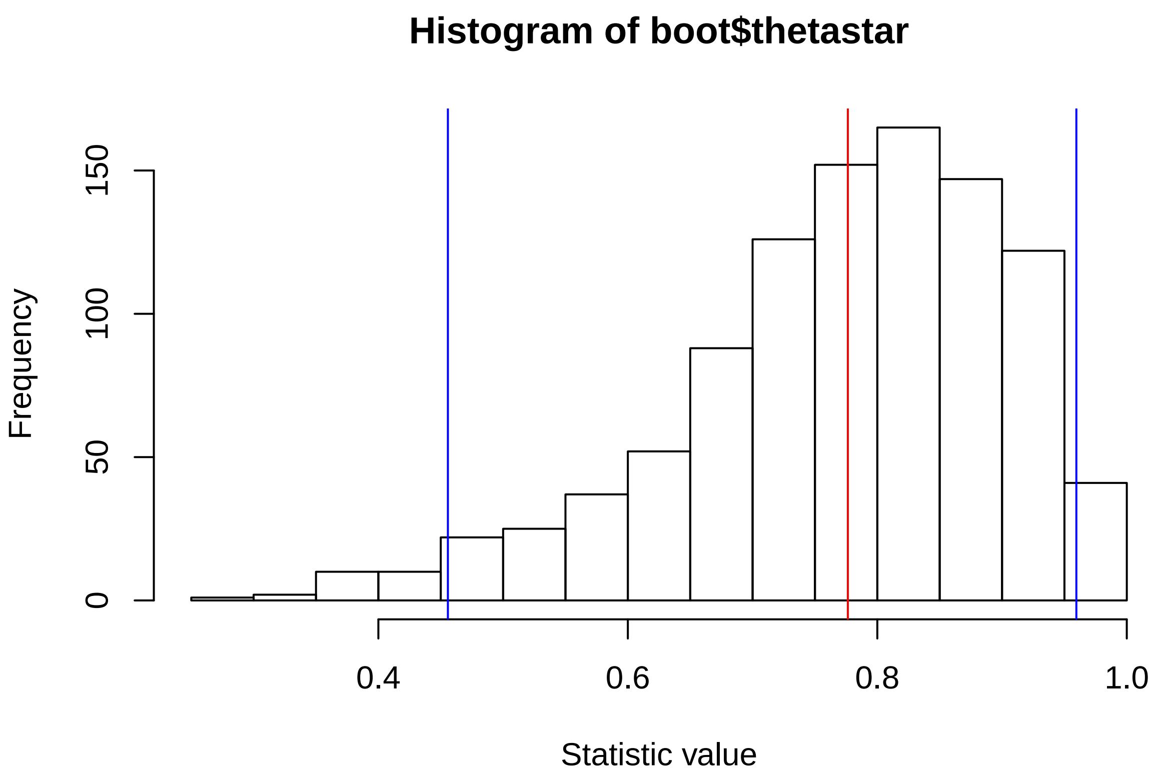
b)

```
set.seed(38) # this seed gave the best looking distribution (i.e. the bins were granular and it looked
corTheta <- function(x, xData){cor(xData[x,"LSAT"], xData[x,"GPA"])} # x = row number, xData = the actu
boot <- bootstrap(x=1:nrow(law), nboot=1000, theta=corTheta, law)
hist(boot$thetastar, xlab="Statistic value")
abline(v=lawCor, col="red")
```



c)

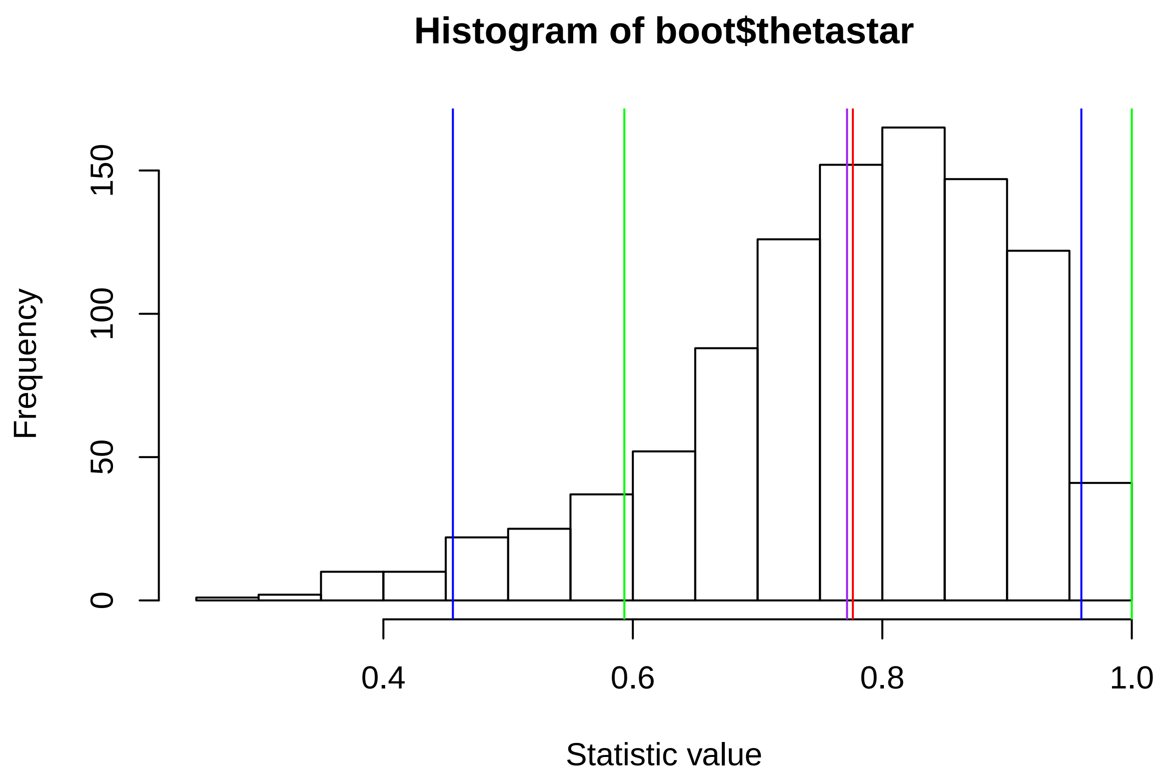
```
lowerCI <- quantile(boot$thetastar, 0.025) # 2.5th percentile of correlations
upperCI <- quantile(boot$thetastar, 0.975) # 97.5th percentile of correlations

hist(boot$thetastar, xlab="Statistic value")
abline(v=lawCor, col="red") # our original estimate
abline(v=lowerCI, col="blue") # lower bound of 95% CI
abline(v=upperCI, col="blue") # upper bound of 95% CI
```



No, we fail to reject the null because a correlation of 0.5 is in our 95% confidence interval [0.456, 0.960]. As such, we believe 0.5 belongs to this distribution and it is a feasible value for the true correlation.

d)

```
baggedEstimate <- mean(boot$thetastar) # the bagged estimate of our bootstrap samples
bc <- 2*lawCor - baggedEstimate # bias-corrected value of the estimate = 2*original_estimate - bagged_e
biasEstimate <- baggedEstimate - lawCor

# take the min and max because correlation is bounded at -1 and 1
correctedLowerCI <- max(2*lawCor - upperCI, -1) # bias-corrected lower bound of the 95% confidence inte
correctedUpperCI <- min(2*lawCor - lowerCI, 1) # bias-corrected upper bound of the 95% confidence inter

hist(boot$thetastar, xlab="Statistic value")
abline(v=lawCor, col="red") # the original estimate
abline(v=lowerCI, col="blue") # lower bound of 95% CI
abline(v=upperCI, col="blue") # upper bound of 95% CI
abline(v=correctedLowerCI, col="green") # corrected lower bound of 95% CI
abline(v=correctedUpperCI, col="green") # corrected upper bound of 95% CI
abline(v=baggedEstimate, col="purple") # bagged estimate of the correlation
```



I went ahead and included an additional vertical line in purple displaying the bootstrap estimate of the correlation ($\bar{\rho}^* = 0.772$). This lies very close to the initial sample estimate of the correlation already mentioned above and displayed as a red vertical line in the histogram ($\hat{\rho}_o = 0.776$). The bootstrap estimate of bias is calculated as $\hat{bias}(\hat{\theta}) = \bar{\rho}^* - \hat{\rho}_o = 0.772 - 0.776 = -0.004$, which is simply the bagged estimate minus the initial sample estimate of the parameter. In the above histogram, the blue lines represent the previously calculated 95% confidence interval for the parameter. The green lines represent the bias corrected 95% confidence interval. The new bounds are calculated as $[2\hat{\theta}_o - UB, 2\hat{\theta}_o - LB] = [2\hat{\theta}_o - \hat{\theta}^*_{0.975}, 2\hat{\theta}_o - \hat{\theta}^*_{0.025}] = [2(0.772) - 0.960, 2(0.772) - 0.456] = [0.593, 1]$.

Yes, we can reject the null since the new 95% confidence interval is [0.593, 1.0]. Since 0.5 is not in the confidence interval, a correlation of 0.5 is not a feasible estimate for the true correlation.

e)

```r
nullPerc <- ecdf(boot$thetastar)
nullPerc(0.0) # percentile of a correlation estimate of 0
```

```
## [1] 0
```

Let's set up a permutation test, similar to a typical hypothesis test:

$$H_o : \rho = 0 H_a : \rho \neq 0$$

We let our test statistic be the hypothesized correlation, 0:

$$t^* = \rho_0 = \mathbf{0}$$

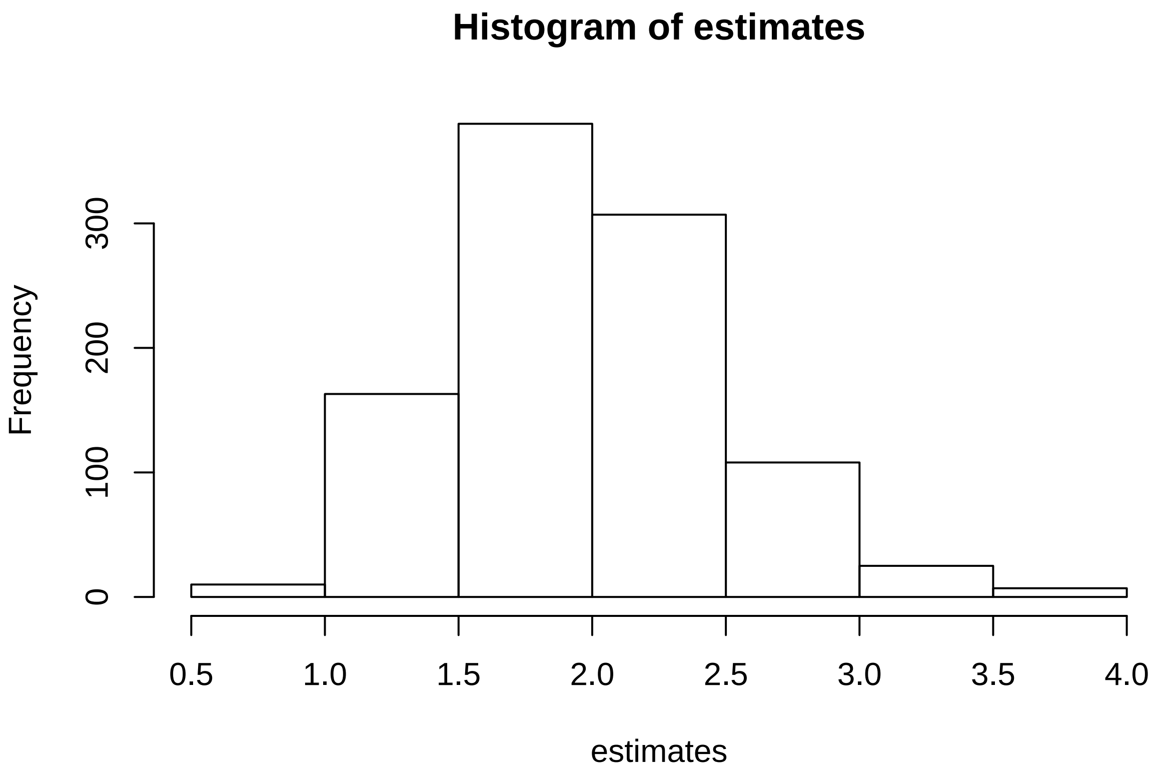
Now, to find the p-value of this estimate, we calculate the percentile of this null hypothesis estimate, which is t^*. In the code above, one can clearly observe the percentile of a correlation being 0, indicating the p-value $<$ 0.001. Since this p-value is statistically significant and deviates from our distribution by a significant amount, we reject H_o and conclude 0 is not a feasible estimate of the true correlation.

Question 6

a) The max of a distribution is an example of a statistic where bootstrapping might not provide the most accurate sampling distribution. More generally, extreme-value statistics cause bootstrapping to be less effective. For example, if we take the maximum value of a normal distribution, $\mathcal{N}(0,1)$, we expect 99.7% of the values to lie within 3 standard deviations of the mean. So, if we reasonably assume the max is 3σ above μ, then the max of the distribution should lie at a value of 3. However, in the below code, we clearly observe the bagged estimate for the max value of a distribution is 2, representing 2σ above μ. This does not align with our claim.

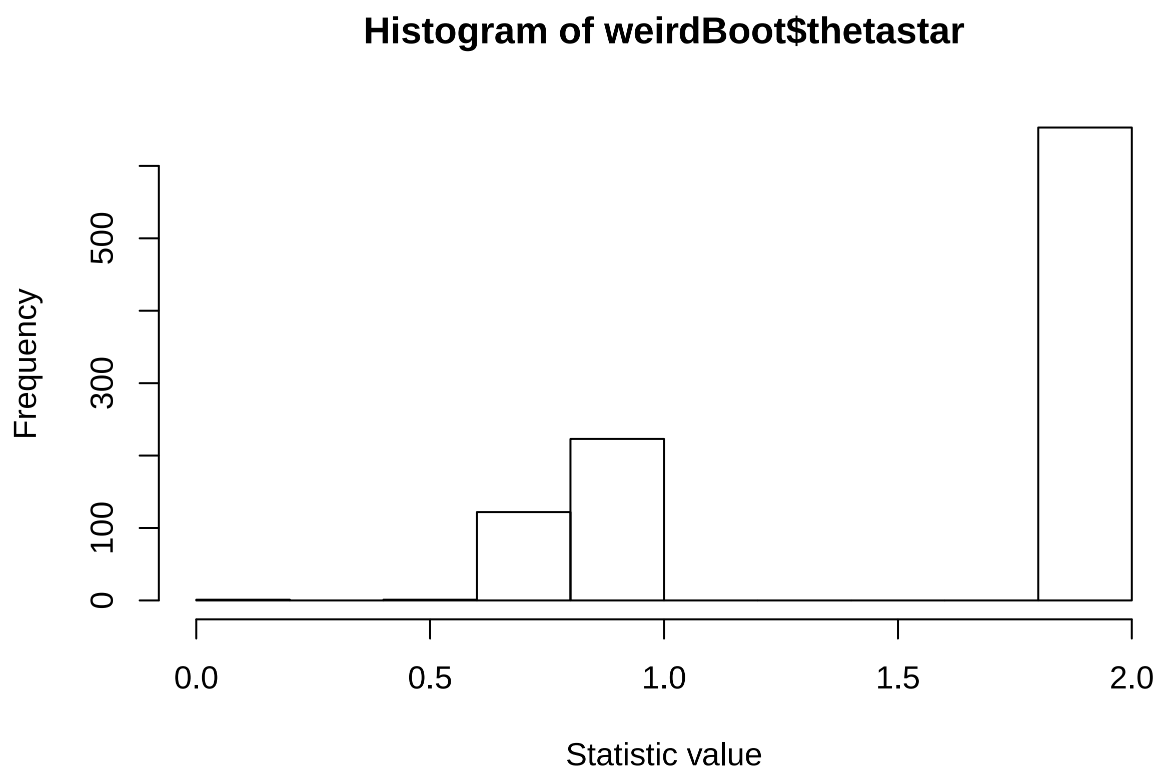
b)

```r
set.seed(1)
library(pracma)
normalData <- as.data.frame(matrix(0, nrow=1000, ncol=25))
for(i in 1:nrow(normalData)){
  normalData[i, 1:ncol(normalData)] = rnorm(ncol(normalData))
}
statistic = function(x){max(x)}
estimates <- apply(normalData, 1, statistic)
hist(estimates)
```



c)

```r
sampleRow <- sample(1:nrow(normalData), 1) # take one sample from the data
oneSample <- normalData[sampleRow, ]
weirdBoot <- bootstrap(x=as.matrix(oneSample), nboot=1000, theta=statistic)
hist(weirdBoot$thetastar, xlab="Statistic value")
```



d) The “true" underline sampling distribution should have a mode at or near 3.0, which represents 3 std. above the mean. The distributions represented in b) and c) showcase a mode/median around 2.0. Additionally, c) is especially awkward, with a heavily left-skewed, sparse distribution. The histogram in part c) can be explained away since we bootstrap 1 sample from the original 1000-row matrix we created. As such, the maximum value of the distribution in c) will always be capped at max(sample), regardless of the bootstrap we draw. Furthermore, any bootstrap sample of this sample will have a max below or equal to max(sample), causing the left-skew. The distribution in b) is a bit more difficult to explain. It’s slightly right-skewed, but shows a distribution more akin to what we should see, especially compared to the sparse histogram in part c). Regardless of whether it’s ‘better’ or not, the b) histogram is still off since the max should be centered at 3.0 or higher, but is actually around 2.0. So, both of these distributions differ from the “true” sampling distribution, but I hypothesize these distributions will look ‘correct’ if we samples with size more than 25, which was a requirement of this exercise, since we will have a more continuous, ‘complete’ sample from the normal distribution.

Question 7

a)

```r
library(bootstrap)
set.seed(1)
# training data
x1 <- runif(50)
x2 <- runif(50)
eps <- rnorm(50, mean=0, sd=0.25)
y <- x1+x2+eps
df <- data.frame(y, x1, x2)

# testing data
x3 <- runif(30)
```

```
x4 <- runif(30)
eps2 <- rnorm(30, mean=0, sd=0.25)
y2 <- x3+x4+eps2
df2 <- data.frame(y2, x3, x4)
names(df2) <- c("y", "x1", "x2")

# train model on training data
modelF <- lm(formula = y ~ x1 + x2, data = df)
summary(modelF)
```

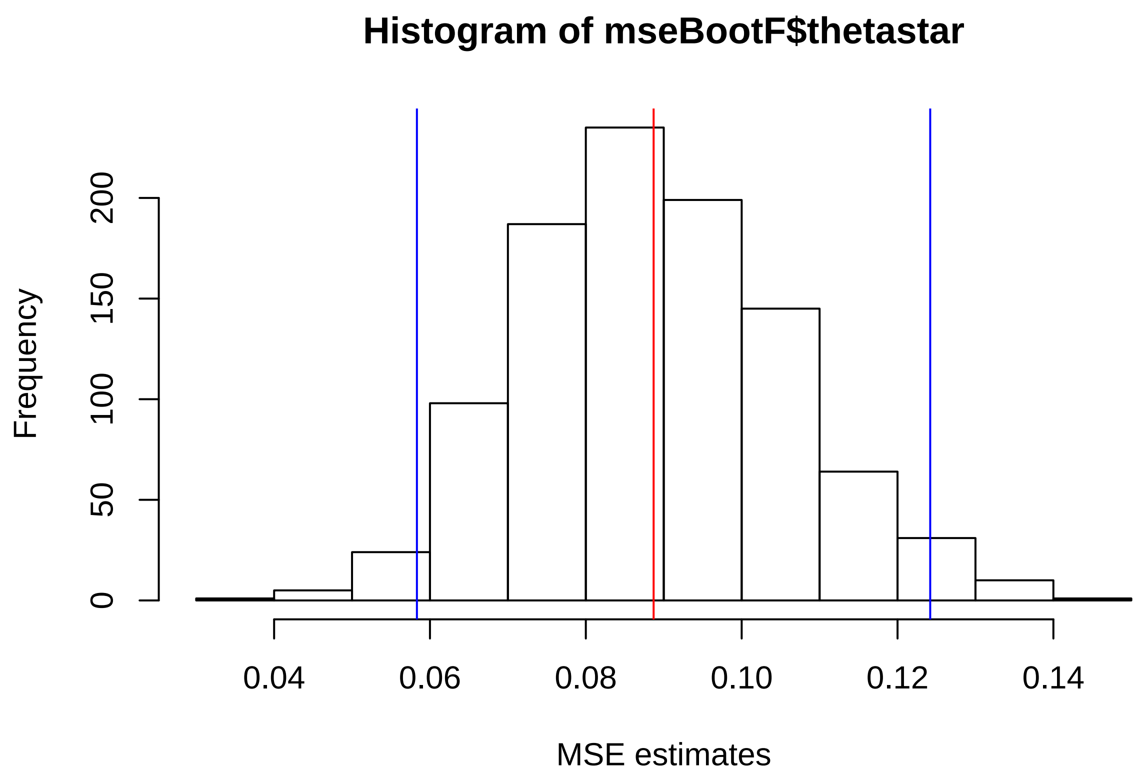
```
##
## Call:
## lm(formula = y ~ x1 + x2, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51877 -0.16512  0.03754  0.11099  0.61852
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.11193    0.09695  -1.154    0.254
## x1           1.21371    0.12582   9.647 1.01e-12 ***
## x2           1.05454    0.12936   8.152 1.51e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2394 on 47 degrees of freedom
## Multiple R-squared:  0.7819, Adjusted R-squared:  0.7726
## F-statistic: 84.26 on 2 and 47 DF,  p-value: 2.866e-16
```

```
mse <- function(x, yTrue, yPred){mean((yTrue[x]-yPred[x])^2)} # MSE_o statistic
yPredF <- predict(modelF, df2)

mseOF <- mse(1:length(y2), y2, yPredF)
```

From the above code, we have calculated an MSE estimate $M\hat{S}E_0 = 0.089$.

b)

```
set.seed(1)

# MSE bootstrap
mseBootF <- bootstrap(x=1:length(y2), nboot=1000, theta=mse, y2, yPredF)
hist(mseBootF$thetastar, xlab="MSE estimates")

# 95% confidence intervals of MSE estimates
lowerCIF <- quantile(mseBootF$thetastar, 0.025)
upperCIF <- quantile(mseBootF$thetastar, 0.975)
abline(v=lowerCIF, col="blue")
abline(v=upperCIF, col="blue")
abline(v=mseOF, col="red")
```



Let's set up the permutation with the following hypotheses:

$$H_o : \beta_1 = \beta_2 = 0$$

$$H_a : \exists i \quad \beta_i \neq 0$$

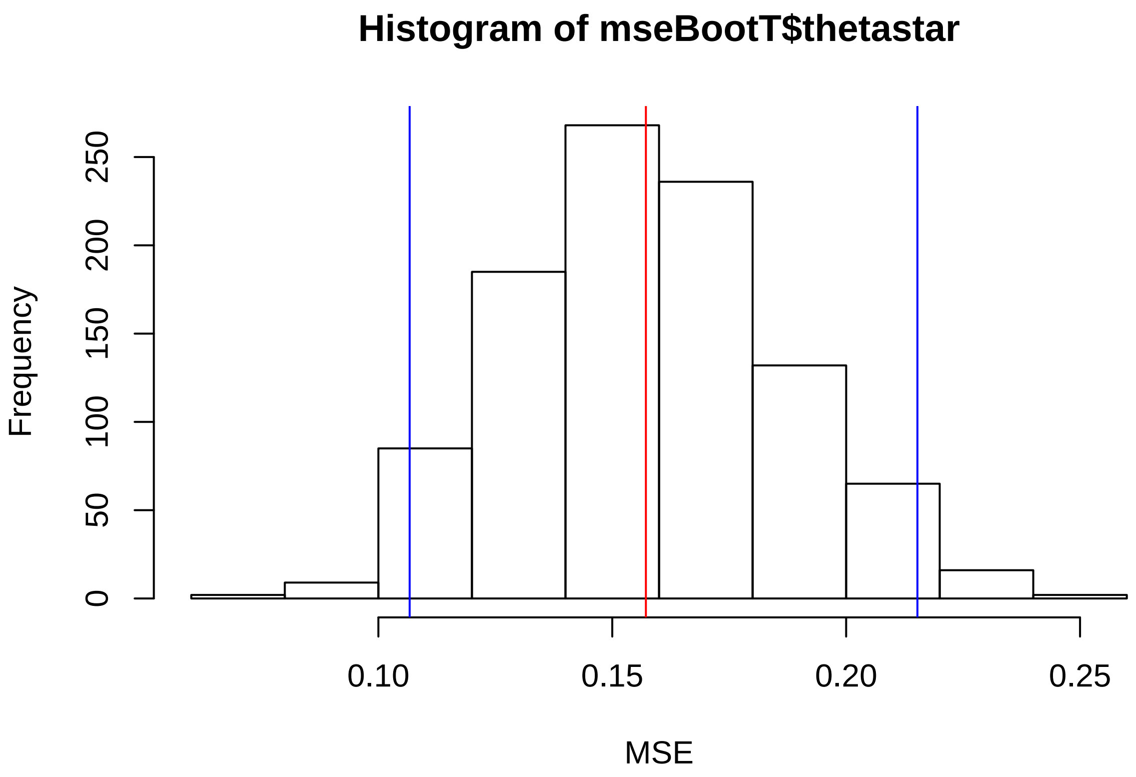
After performing 1000 permutations, the above histogram displays a confidence interval of [0.058, 0.124]. Our MSE estimate (0.089) lies in the 95% confidence interval. As such, we fail to reject the null hypothesis and conclude this model of $y \sim x1 + x2$ is not a significantly great predictor of y.

c)

```r
set.seed(1)

# train model on training data
modelT <- lm(formula = y ~ x2, data = df)
summary(modelT)
```

```
##
## Call:
## lm(formula = y ~ x2, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85190 -0.21590  0.00848  0.29108  0.88086
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5005     0.1252   3.998 0.000219 ***
## x2            1.1221     0.2206   5.086 6.02e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.409 on 48 degrees of freedom
## Multiple R-squared:  0.3502, Adjusted R-squared:  0.3366
## F-statistic: 25.86 on 1 and 48 DF,  p-value: 6.019e-06
```

```
yPredT <- predict(modelT, df2)

mseOT <- mse(1:length(y2), y2, yPredT) # MSE_o t-test estimate

# MSE bootstrap
mseBootT <- bootstrap(x=1:length(y2), nboot=1000, theta=mse, y2, yPredT)
hist(mseBootT$thetastar, xlab="MSE")

# 95% confidence intervals of MSE estimates
lowerCIT <- quantile(mseBootT$thetastar, 0.025)
upperCIT <- quantile(mseBootT$thetastar, 0.975)
abline(v=lowerCIT, col="blue")
abline(v=upperCIT, col="blue")
abline(v=mseOT, col="red")
```



Let's set up the permutation with the following hypotheses:

$$H_o : \beta_2 = 0$$

$$H_a : \beta_2 \neq 0$$

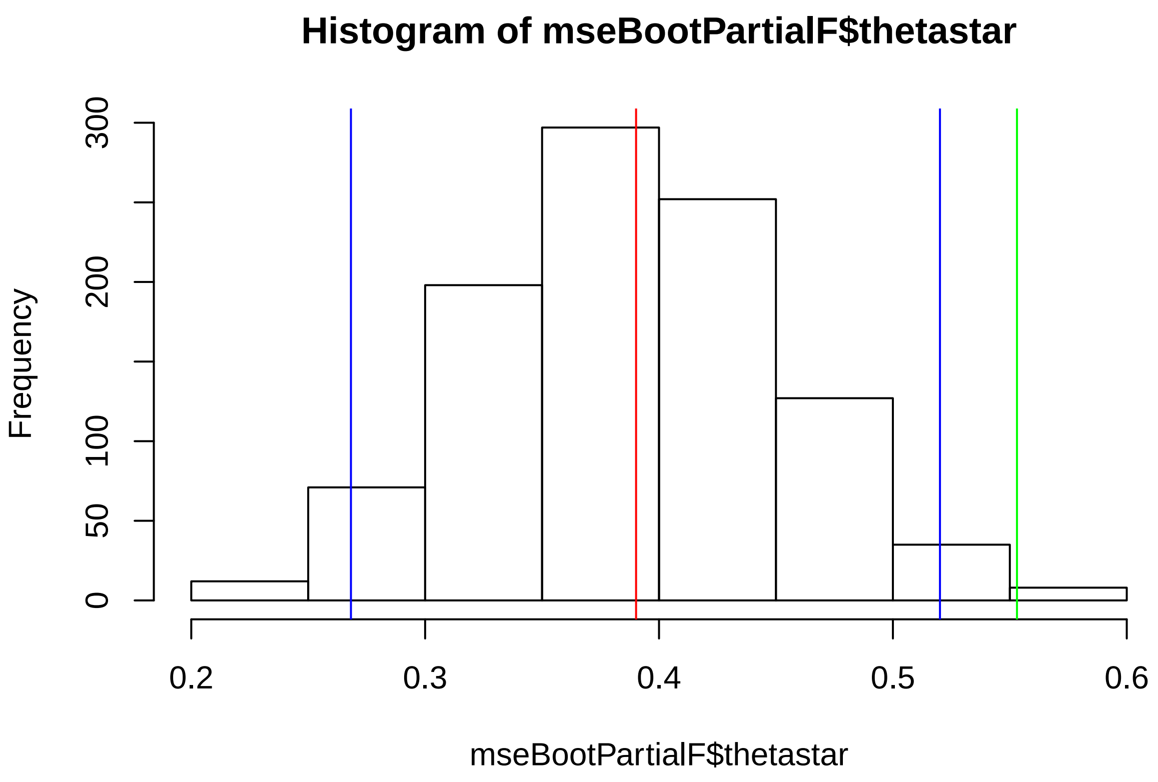
No. The MSE for the model $y \sim x2$ is 0.157. As displayed in the above histogram, the 95% confidence interval for this model's MSE estimate is [0.107, 0.215]. As such, since the MSE is in the confidence interval, we fail to reject the null and conclude β_2 is not a significant predictor.

d)

```r
set.seed(1)

# training data
x5 <- matrix(0, nrow=500, ncol=10)
eps3 <- matrix(0, nrow=500, ncol=10)
for(i in 1:500){
  x5[i, 1:10] = runif(10)
  eps3[i, 1:10] = rnorm(10, mean=0, sd=0.25)
}
yTrain <- x5+eps3
yTrain <- apply(yTrain, 1, sum)
df3 <- data.frame(yTrain, x5)
names(df3) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10")

# testing data
x6 <- matrix(0, nrow=50, ncol=10)
eps4 <- matrix(0, nrow=50, ncol=10)
for(i in 1:50){
  x6[i, 1:10] = runif(10)
  eps4[i, 1:10] = rnorm(10, mean=0, sd=0.25)
}
yTest <- x6+eps4
yTest <- apply(yTest, 1, sum)
df4 <- data.frame(yTest, x6)
names(df4) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10")
```

e)

```r
set.seed(1)

# reduced and full models for partial-F tests
modelFull <- lm(formula = y ~ ., data = df3) # full model
modelReduced <- lm(formula = y ~ . - x8 - x9 - x10, data = df3)

# model predictions
modelFullPred <- predict(modelFull, df4)
modelReducedPred <- predict(modelReduced, df4)

# model MSEs
modelFullMSE <- mse(1:length(modelFullPred), modelFullPred, yTest)
modelReducedMSE <- mse(1:length(modelFullPred), modelReducedPred, yTest)

# MSE bootstrap
mseBootPartialF <- bootstrap(x=1:length(yTest), nboot=1000, theta=mse, yTest, modelFullPred)
hist(mseBootPartialF$thetastar)

# 95% confidence intervals of MSE estimates
lowerCIPartialF <- quantile(mseBootPartialF$thetastar, 0.025)
upperCIPartialF <- quantile(mseBootPartialF$thetastar, 0.975)
abline(v=lowerCIPartialF, col="blue")
abline(v=upperCIPartialF, col="blue")
abline(v=modelFullMSE, col="red")
abline(v=modelReducedMSE, col="green")
```



In this partial-F test, we want to examine if any of the features X_8, X_9, X_{10} are significant predictors of F. We have 1 reponse, y, and 10 predictors, $x_1 \dots x_{10}$. The full model is represented as $y = \beta_0 + \beta_1 X_1 + \dots \beta_{10} X_{10}$. The reduced model is represented as $y = \beta_0 + \beta_1 X_1 + \dots \beta_7 X_7$. As such, we perform a permutation test to examine whether the MSE of the reduced model is significantly lower than the MSE of the full model. If the reduced MSE is outside the 95% confidence interval of the permutation test estimate, then we can conclude the reduced model is a significant predictor of y. We proceed as follows:

$$H_o : \beta_8 = \beta_9 = \beta_{10} = 0$$
$$H_a : \exists\, i \in [8, 9, 10]\ \beta_i \neq 0$$

After forming the reduced model, we reach $MSE_{reduced} = 0.553$. The full model's MSE (red line in above histogram) produces $MSE_{full} = 0.390$. The above histogram showcases the permutation test distribution of the MSE estimate. The blue lines represent the 95% confidence interval of the estimate, which is [.268, 520]. Since $MSE_{reduced} \notin [.268, .520]$, we reject H_o and conclude the reduced model, indicated by the green line, is a significant predictor of y, so X_8, X_9, X_{10} are significant predictors. However, note the $MSE_{reduced}$ is above the confidence interval. As such, the reduced model's MSE is significantly worse than the full model's MSE.

Question 8

```r
library(class)
library(MASS)
library(stats)
library(cvTools)
```

```
## Loading required package: lattice

## Loading required package: robustbase
```

```r
library(rfUtilities)
library(bootstrap)

set.seed(1)

# Reading data and preprocessing
newsDataOriginal <- read.table("OnlineNewsPopularity.csv", header=TRUE, sep=",")
newsDataOriginal$shares = as.numeric(newsDataOriginal$shares)
sharesIqr <- IQR(newsDataOriginal$shares)
shares75Quant <- quantile(newsDataOriginal$shares, 0.75)
shares25Quant <- quantile(newsDataOriginal$shares, 0.25)
newsData <- newsDataOriginal[newsDataOriginal$shares < (1.5*sharesIqr + shares75Quant) & newsDataOrigina
newsDataQuant <- newsData[, sapply(newsData, class) == "numeric"]
newsDataQuant <- newsDataQuant[sample(nrow(newsDataQuant)),] # randomly shuffle the data

# Linear regression model discussed in Homework 3
newsLm <- lm(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_is_world + l

# Process binarized and trinarized datasets
newsDataBinary <- data.frame(newsDataQuant) # make a copy
newsDataBinary$shares <- ifelse(newsDataBinary$shares > quantile(newsDataBinary$shares, 0.5), 1, 0)
newsDataTrinary <- data.frame(newsDataQuant) # make a copy
newsDataTrinary$shares <- ifelse(newsDataTrinary$shares > quantile(newsDataTrinary$shares, 0.333), ifel

# Set the dataset used for all classification problems
newsClassif <- newsDataBinary # current classification set we're using

# Create training and testing sets
trainIndex <- sample(1:nrow(newsClassif), 0.8*nrow(newsClassif)) # train indices
testIndex <- setdiff(1:nrow(newsClassif), trainIndex) # test indices
train <- newsClassif[trainIndex,]
test <- newsClassif[testIndex,]
trainQuant <- newsDataQuant[trainIndex,]
testQuant <- newsDataQuant[testIndex,]
trainX <- newsClassif[trainIndex, -61]
trainY <- newsClassif[trainIndex, "shares"]
testX <- as.data.frame(newsClassif[testIndex, -61])
testY <- as.data.frame(newsClassif[testIndex, "shares"])
testYQuant <- as.data.frame(newsDataQuant[testIndex, "shares"])

# set up folds for cross-validation
k = 10
folds <- cut(seq(1, nrow(newsClassif)), breaks = k, labels = FALSE)

# Logistic regression model discussed in Homework 3
newsLogit <- glm(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_is_worl
summary(newsLogit)
```

```
##
## Call:
## glm(formula = shares ~ data_channel_is_entertainment + data_channel_is_socmed +
##     data_channel_is_world + kw_avg_avg + LDA_02 + weekday_is_saturday:is_weekend,
##     family = binomial, data = newsDataBinary)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3302  -1.1270  -0.8155   1.1339   1.8008
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -2.956e-01  3.817e-02  -7.744 9.66e-15 ***
## data_channel_is_entertainment  -8.040e-01  3.052e-02 -26.343  < 2e-16 ***
## data_channel_is_socmed          8.407e-01  5.330e-02  15.774  < 2e-16 ***
## data_channel_is_world          -3.374e-01  5.095e-02  -6.623 3.51e-11 ***
## kw_avg_avg                      1.479e-04  1.053e-05  14.055  < 2e-16 ***
## LDA_02                         -6.817e-01  7.354e-02  -9.270  < 2e-16 ***
## weekday_is_saturday:is_weekend  1.044e+00  5.047e-02  20.688  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48563  on 35102  degrees of freedom
## Residual deviance: 45884  on 35096  degrees of freedom
## AIC: 45898
##
## Number of Fisher Scoring iterations: 4
```

```
# LDA model discussed in Homework 3
newsLda <- lda(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_is_world +
summary(newsLda)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   12     -none- numeric
## scaling  6     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
# QDA model discussed in Homework 3
newsQda <- qda(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_is_world +
summary(newsQda)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   12     -none- numeric
## scaling 72     -none- numeric
## ldet     2     -none- numeric
## lev      2     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
mse_sum = 0
# K-fold cross-validation
for(i in 1:k){
  # grab the i-th fold
  testIndices <- which(folds == i, arr.ind=TRUE)
  testData <- newsClassif[testIndices,]
  trainData <- newsClassif[-testIndices,]

  # train t he model
  newsLm <- lm(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_is_world
  lmPred <- predict(newsLm, testQuant)
  summary(newsLm)
  mse <- mean((lmPred - t(testYQuant))^2) # get the MSE

  mse_sum <- mse_sum + mse
}
mse_sum <- mse_sum / k
cat(k, "-fold cross-validation MSE for LM: ", mse_sum)
```

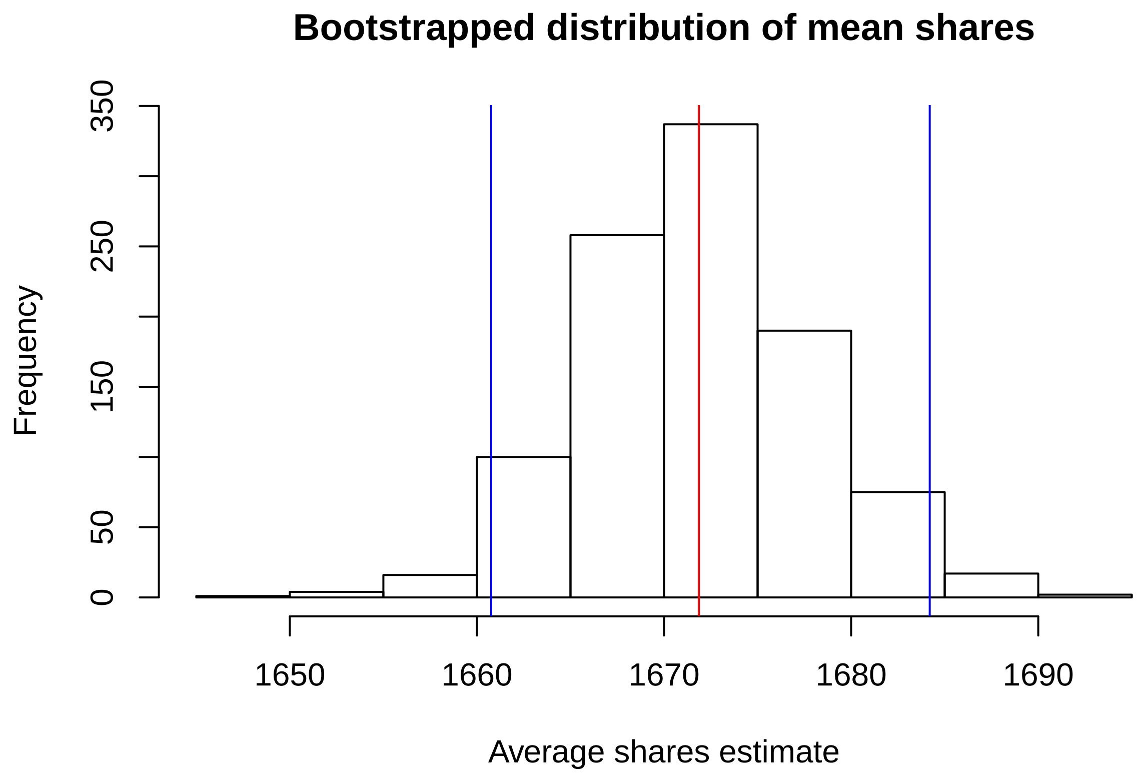
```
## 10 -fold cross-validation MSE for LM:  1128384
```

```
acc_sum = 0
# K-fold cross-validation
for(i in 1:k){
  # grab the i-th fold
  testIndices <- which(folds == i, arr.ind=TRUE)
  testData <- newsClassif[testIndices,]
  trainData <- newsClassif[-testIndices,]

  # train t he model
  newsKnn <- DMwR::kNN(shares ~ data_channel_is_entertainment + data_channel_is_socmed + data_channel_i
  newsKnn <- as.numeric(newsKnn)-1 # transform outputs from 1 & 2 to 0 & 1

  # compute accuracy
  correct = 0
  for(j in 1:length(newsKnn)){
    if(newsKnn[j] == testData$shares[j] && j < length(newsKnn) && j < length(testData$shares)){
      correct <- correct + 1
    }
  }
  acc_sum <- acc_sum + (correct/length(newsKnn))
}
acc_sum <- acc_sum / k
cat(k, "-fold cross-validation kNN accuracy: ", acc_sum)
```

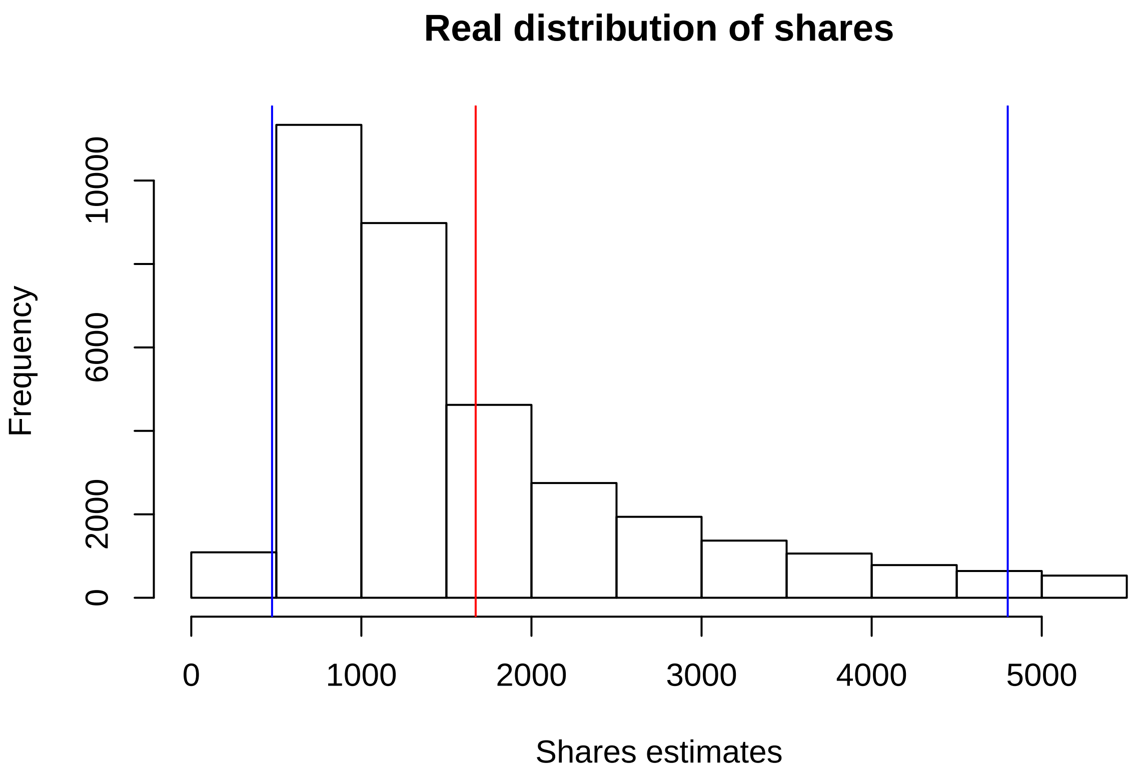
```
## 10 -fold cross-validation kNN accuracy:  0.6369828
```

```
# statistics
avgShares <- function(x){mean(x)}
sharesBoot <- bootstrap(newsDataQuant$shares, 1000, avgShares)
avgSharesEstimate <- mean(newsDataQuant$shares)
avgSharesBaggedEstimate <- mean(sharesBoot$thetastar)
sharesLowerCI <- quantile(newsDataQuant$shares, 0.025)
sharesUpperCI <- quantile(newsDataQuant$shares, 0.975)
sharesBootLowerCI <- quantile(sharesBoot$thetastar, 0.025)
```

```
sharesBootUpperCI <- quantile(sharesBoot$thetastar, 0.975)

# bootstrapping and bagging
hist(sharesBoot$thetastar, main="Bootstrapped distribution of mean shares", xlab="Average shares estima
abline(v=avgSharesBaggedEstimate, col="red")
abline(v=sharesBootLowerCI, col="blue")
abline(v=sharesBootUpperCI, col="blue")
```



```
hist(newsDataQuant$shares, main="Real distribution of shares", xlab="Shares estimates")
abline(v=avgSharesEstimate, col="red")
abline(v=sharesLowerCI, col="blue")
abline(v=sharesUpperCI, col="blue")
```



It’s easy to see why cross-validation would be effective and easy to implement for this project. We can use cross-validation for the trained classifiers (kNN, logistic regression, LDA, QDA, and linear regression) to reduce variance in accuracies (this is a pretty obvious result of cross-validation and taking the mean of the k folds’ accuracies), which is useful since a classifier can be unlucky for a particular seed and produce sub-par accuracy or get lucky lucky for another seed and produce a spectacularly high accuracy. Cross-validation with k = 5 or k = 10 would make good use of the 40k samples of the training set to average the accuracy or RSS and produce an accurate ‘final’ summarization of how a given classifier fits to the dataset and can even dictate the best version of a model (i.e. value of k for kNN). However, even with that being said, cross-validation isn’t immensely useful in our dataset, considering the small-ish number of features compared to the number of samples. In fact, ISLR agrees with this point by stating, “In the absence of a very large designated test set that can be used to directly estimate the test error rate, … [cross-validation] estimates the test error rate … “. In our case, we have a large designated test set, so cross-validation isn’t expected to produce substantial results. Finally, out of all cross-validation variations, LOOCV is the most useless since we have a large training and test set and LOOCV is designed for datasets with an extremely small test set or not enough samples for a test set.

Permutation tests don’t have as much applicability in this project. We can make use of statistical inference to reason about the number of shares for a given article, but we’re not hypothesizing about a specific value for a given feature. Moreso, we’re not even interested in hypothesizing about each feature of the dataset. If anything, we would only be interested in the response, which is the number of shares for an article. Perhaps, if anything, we could use permutation tests to hypothesize about the coefficients for features in the linear regression model. Or, we could train each classifier and investigate whether the accuracy/error is statistically significant, checking if the model is useful compared to others. However, this is more for an ‘experiment’, rather than for practical purposes, since we’re already using 4 different classifiers, meaning it’s not really vital to the process.

Bootstrapping has limited application, similar to permutation tests. One practical application is it could help find confidence intervals for the predicted shares of an article. So, instead of giving one precise prediction, which could have a value in the area of hundreds or thousands, we output a 95% confidence interval to a customer, telling them where we estimate the article to lie in terms of number of shares.

Resampling (bootstrapping, cross-validation) in general isn't necessary for our dataset, but has a few practical uses worth testing. This is easily understood since the dataset has 40k samples over mearly 59 predictors. Moreso, the predictors do not have wide ranges. In fact, 24 of the 59 predictors are binary features. Resampling to estimate the value of a binary feature is almost completely useless. The beauty of bootstrapping, and specifically bagging, is to utilize a small sample size to estimate the value of a statistic. I strongly believe 40k samples is enough to estimate any given feature without bagging.

In general, I trust the models built with tools from Chapters 3 and 4 a lot. I tested kNN with several different values of k, so I know how well the model performs for small, medium, and large values of k. Also, the dataset, as previously mentioned several times, is large. Therefore, this reduces risk of underfitting significantly and I believe the models aren't as prone to overfitting due to only marginally better testing and training error (due to dimensionality of the dataset).

After some analysis, it was found with 10-fold cross-validation that the accuracy of knn was 63.74% for 101-nearest-neighbors, which essentially exactly matches the 63% binary accuracy achieved without cross-validation in homework 3. Additionally, we reach a 61.95% accuracy with 11-nearest-neighbors and 63.24% on 401-nearest-neighbors. As such, cross-validation is going to be of no use for us and we can reasonably conclude a value of k=101 for kNN is the best parameter for the model, which is the predicted best parameter value from homework 3. I believe, even if there is just a 2% increase in accuracy, despite the higher model complexity, k=101 is better than k-11. To summarize, in a roundabout way, one can even simulate cross-validation by using a different seed in different simulations and taking the average of the accuracies, which is what I did in homework 3 anyway. As such, naturally, formally implementing k-fold cross-validation is not going to help our model. That's one sampling method that's not going to help us too much, mostly due to our already large dataset. I will not even attempt to replicate k-fold cross-validation with the other models, since it doesn't make sense practically or theoretically and it would be a waste of resources and time. As an additional test, I ran 10-fold cross-validation on our regression problem with multiple linear regression and the same model as used in Homework 3. The estimated MSE is 1,128,384 which means each residual is off by about $\sqrt{1128384} = 1062.254$ shares. As such, the model doesn't seem to do the best, but performs similarly to our model in Homework 3. As such, cross-validation isn't entirely useful in both the regression and classification problems.

To test out bootstrapping with our predictor, shares, the two above histograms were created. The first histogram displays the bootstrapped distribution of the number of shares for an article. This distribution is roughly Normal with a bagged estimate (mean) of 1672 shares. The true distribution of shares across the entire dataset is heavily right-skewed, but still has a mean of 1672 shares. As such, the distributions may look different, but their means are exactly the same. This was to be expected due to the large number of samples in the dataset. These mentioned estimates are both shown as red vertical lines in their distributions. Alongside these estimates, the 95% confidence intervals of the distributions were computed and shown in blue. For the bootstrapped distribution, the CI is [1660, 1684]. For the true distribution, the CI is [475, 4800]. As such, although there is not a significant difference in the means of the distributions, the confidence intervals for the estimates have a vast difference. If we were to set up a permutation test for this situation, it would have the following hypotheses: $H_o : \theta = 0$ vs. $H_a : \theta \neq 0$, where θ is the mean number of shares. Since 0 is clearly in neither of these CIs, we can reject the null and conclude the average number of shares is significantly higher than 0. We can also set up another permutation test to check whether our sample estimated number of shares is significantly different from 1600. The hypotheses are as follows: $H_o : \hat{\theta} = 1600$ vs. $H_a : \hat{\theta} \neq 1600$. However, since 1600 is in both of the CIs, we fail to reject H_o and conclude the average number of shares is not significantly different from 1600.