STATS 202 | HW:4| Sagar Ganapaneni | SUID# 06167633

Problem: 1

Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction.

Answer:

Standard deviation of prediction (statistic) can be estimated using the bootstrap method.

- Using bootstrap, we can obtain multiple random samples from original set of data by resampling with replacement.
- Estimate the prediction with each resampled data and repeat this step for B times
- Now standard deviation of prediction can have measured from the pool of predictions estimates we got from the above exercise

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}$$

- And the above formula is used to measure Standard error of these boot strapped estimates

Problem 2

(a)Fit a logistic regression model that uses "income" and "balance" to predict "default".

```
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.2.5

attach(Default)
## set random seed
set.seed(12)
glm_model <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(glm_model)

##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b)Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

```
## split data
split = sample(1:nrow(Default), size=nrow(Default)/2)
```

```
Default_train <- Default[split, ]
Default_test<- Default[-split,]
```

ii. Fit a multiple logistic regression model using only the training observations.

```
glm_model <- glm(default ~ income + balance, data = Default_train, family = "binomial")
summary(glm_model)

##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2757  -0.1335  -0.0531  -0.0184   3.7695
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.195e+01  6.431e-01 -18.575   <2e-16 ***
## income       2.186e-05  7.231e-06   3.023   0.0025 **
## balance      5.875e-03  3.362e-04  17.476   <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: 1450.21  on 4999  degrees of freedom
## Residual deviance:  756.78  on 4997  degrees of freedom
## AIC: 762.78
##
## Number of Fisher Scoring iterations: 8
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the "default" category if the posterior probability is greater than 0.5.

```
ods <- predict(glm_model, Default_test, type = "response")
glm_pred <- rep("No", length(ods))
glm_pred[ods > 0.5] <- "Yes"
table(glm_pred, Default_train$default)

##
## glm_pred   No  Yes
```

```
##      No  4757  165
##      Yes   78    0
```

```
mean(glm_pred != Default_train$default)
```

```
## [1] 0.0486
```

Estimated mean Test Error is 4.86% in this case

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
# run 1
split = sample(1:nrow(Default), size=nrow(Default)/2)
Default_train <- Default[split, ]
Default_test<- Default[-split,]
glm_model <- glm(default ~ income + balance, data = Default_train, family = "binomial")
ods <- predict(glm_model, Default_test, type = "response")
glm_pred <- rep("No", length(ods))
glm_pred[ods > 0.5] <- "Yes"
table(glm_pred, Default_train$default)
```

```
##
## glm_pred   No  Yes
##      No  4743  171
##      Yes   86    0
```

```
mean(glm_pred != Default_train$default)
```

```
## [1] 0.0514
```

```
# run 2
split = sample(1:nrow(Default), size=nrow(Default)/2)
Default_train <- Default[split, ]
Default_test<- Default[-split,]
glm_model <- glm(default ~ income + balance, data = Default_train, family = "binomial")
ods <- predict(glm_model, Default_test, type = "response")
glm_pred <- rep("No", length(ods))
glm_pred[ods > 0.5] <- "Yes"
table(glm_pred, Default_train$default)
```

```
##
## glm_pred   No  Yes
##      No  4752  168
##      Yes   77    3
```

```
mean(glm_pred != Default_train$default)
```

```
## [1] 0.049
```

```
# run 3
split = sample(1:nrow(Default), size=nrow(Default)/2)
Default_train <- Default[split, ]
Default_test<- Default[-split,]
glm_model <- glm(default ~ income + balance, data = Default_train, family = "binomial")
ods <- predict(glm_model, Default_test, type = "response")
glm_pred <- rep("No", length(ods))
glm_pred[ods > 0.5] <- "Yes"
table(glm_pred, Default_train$default)

##
## glm_pred   No  Yes
##      No  4756  171
##      Yes   70    3

mean(glm_pred != Default_train$default)

## [1] 0.0482
```

Test error estimates are varying with each validation sample but the values are close.

(d)Now consider a logistic regression model that predicts the probability of "default" using "income", "balance", and a dummy variable for "student". Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for "student" leads to a reduction in the test error rate.

```
## with dummy variable student

split = sample(1:nrow(Default), size=nrow(Default)/2)
Default_train <- Default[split, ]
Default_test<- Default[-split,]
glm_model <- glm(default ~ income + balance+student, data = Default_train, family = "binomial")
ods <- predict(glm_model, Default_test, type = "response")
glm_pred <- rep("No", length(ods))
glm_pred[ods > 0.5] <- "Yes"
table(glm_pred, Default_train$default)

##
## glm_pred   No  Yes
##      No  4785  162
##      Yes   51    2

mean(glm_pred != Default_train$default)

## [1] 0.0426
```

The estimated test error didn't change much even after adding a dummy variable: student

Problem 3:

(a)Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with "income" and "balance" in a multiple logistic regression model that uses both predictors.

```
set.seed(12)
attach(Default)

## The following objects are masked from Default (pos = 3):
##
##     balance, default, income, student

glm_model <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(glm_model)

##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

β0, β1 & β2 are marked in yellow and standards errors are marked in green respectively

(b)Write a function, boot.fn(), that takes as input the "Default" data set as well as an index of the observations, and that outputs the coefficient estimates for "income" and "balance" in the multiple logistic regression model.

```
boot.fn <- function(data, split) {
  glm_model <- glm(default ~ income + balance, data = data, family = "binomial", subset = split)
  return (coef(glm_model))
}
```

(c)Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for "income" and "balance".

```
library(boot)
boot(Default, boot.fn, 1000)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##        original              bias             std. error
## t1* -1.154047e+01 -3.132754e-02 4.224566e-01
## t2*  2.080898e-05  1.195612e-07 4.791184e-06
## t3*  5.647103e-03  1.330443e-05 2.221212e-04
```

Estimates and standard errors obtained from bootstrap are very close to the respective outcomes of the above logistic regression model

Problem 4:

(a)Based on this data set, provide an estimate for the population mean of "medv".

```
library(MASS)
attach(Boston)
EMu_medv<- mean(medv)
EMu_medv

## [1] 22.53281
```

(b)Provide an estimate of the standard error of μ^. Interpret this result.

```
SE_EMu_mdev <- sd(medv) / sqrt(dim(Boston)[1])
SE_EMu_mdev

## [1] 0.4088611
```

(c)Now estimate the standard error of μ^ using the bootstrap. How does this compare to your answer from (b) ?

```
library(boot)
set.seed(12)
boot.fn <- function(data, split) {
   EMu <- mean(data[split])
   return (EMu)
}
boot(medv, boot.fn, 1000)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.009365217   0.4123366
```

The bootstrap estimated standard error is close to the estimate found in (b) of 0.4089.

(d)Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of "medv". Compare it to the results obtained using t.test(Boston$medv).

```
t.test(medv)

##
##  One Sample t-test
##
## data:  medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281

CI_EMu <- c(22.53 - 2 * 0.4123, 22.53 + 2 * 0. 4123)
CI_EMu

## [1] 21.7054 23.3546
```

The bootstrap confidence interval is very close to the one provided by the t.test

(e)Based on this data set, provide an estimate, μ^ med, for the median value of "medv" in the population.

```
EMed_medv <- median(medv)
EMed_medv

## [1] 21.2
```

(f) We now would like to estimate the standard error of μ^med. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot.fn <- function(data, split) {
  Emedian <- median(data[split])
  return (Emedian)
}
boot(medv, boot.fn, 1000)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
```

```
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     21.2 -0.01685   0.3775859
```

The estimated median value we got here is equal to the estimated median value obtained in (e)

(g)Based on this data set, provide an estimate for the tenth percentile of "medv" in Boston suburbs. Call this quantity $\mu\hat{}_{0.1}$

```
Mu_10p <- quantile(medv, c(0.1))
Mu_10p

##   10%
## 12.75
```

(h)Use the bootstrap to estimate the standard error of $\mu\hat{}_{0.1}$. Comment on your findings.

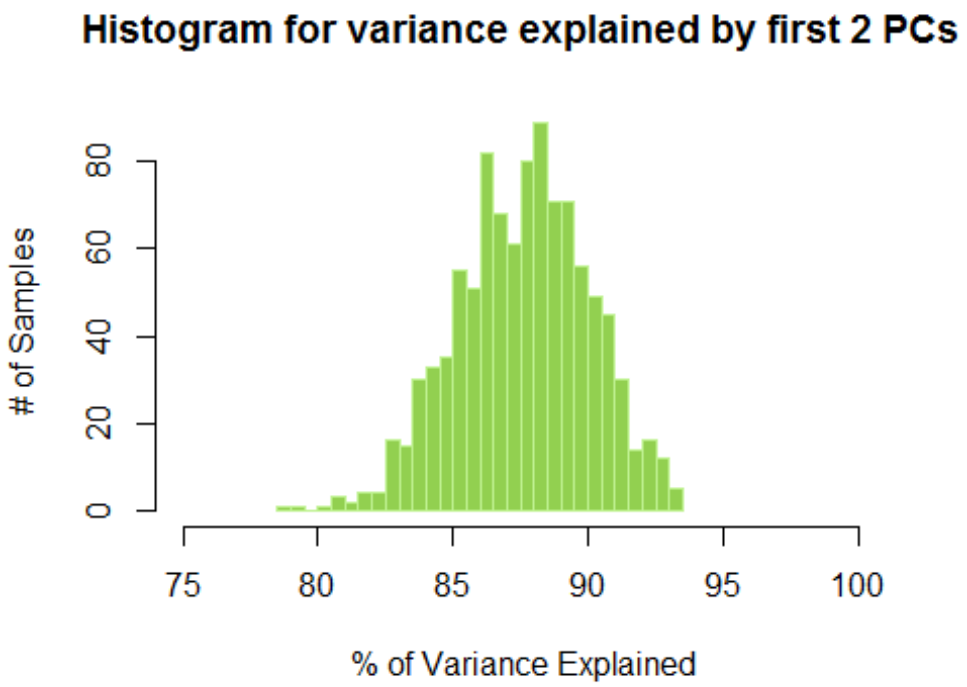
```
boot.fn <- function(data, split) {
  mu <- quantile(data[split], c(0.1))
  return (mu)
}
boot(medv, boot.fn, 1000)

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 -0.0095   0.5151797
```

The estimated tenth percentile value: 12.75 which is again equal to the value obtained in (g), with a standard error of 0.5151

Problem 5:

(1) Using the USArrests dataset, plot a histogram of the proportion of variance explained by the first 2 principal components in 1000 Bootstrap resamplings of the data.

```
## Load data
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.2.5

attach(USArrests)
set.seed(12)
PC12_prop <- numeric(1000)
for(i in 1:1000){
  ## sample USArrest
  split = sample(1:nrow(USArrests), size=nrow(USArrests)/2)
  USArrest_boot <- USArrests[split, ]
  ## do PCA
  PCA_USArrests <- prcomp(USArrest_boot,center = TRUE,scale. = TRUE)

  ## Explnation by first two PCS
  PC_variance<- PCA_USArrests$sdev^2
  PC_var_prop <- PC_variance/sum(PC_variance)
  PC12_prop[i] <- (PC_var_prop[1]+PC_var_prop[2])*100
}

hist(PC12_prop,
     main="Histogram for variance explained by first 2 PCs",
     ylab="# of Samples ",
     xlab=" % of Variance Explained",
     border="#c0f292",
     col="#92d050",
     xlim=c(75,100),
     breaks=25)
```



(2) Estimate a standard error and 95% confidence intervals for the proportion of variance explained by the first 2 principal components.

```
# mean
pc_mean<-mean(PC12_prop)
# sd
pc_sd<-sd(PC12_prop)
# confidance interval
pc_CI <- c(pc_mean-2*pc_sd,pc_mean+2*pc_sd)
pc_CI

## [1] 82.73915 92.49213
```

Hence the Confidence interval is 82.73915 - 92.49213

(3) Suppose we compute the first principal component from each of 1000 Bootstrap resamplings of the data. Using the resulting 1000 vectors, we estimate the standard error of each entry or loading using Eq. 5.8 in the textbook. Explain why this would be problematic.

Answer:

If the interested statistic is First Principal component, a vector, it would be difficult to measure standard deviation of vector component in multi-dimensional space.

(4) There is a way around the problem alluded to in part 3. Write a function in R which, given a data.frame:

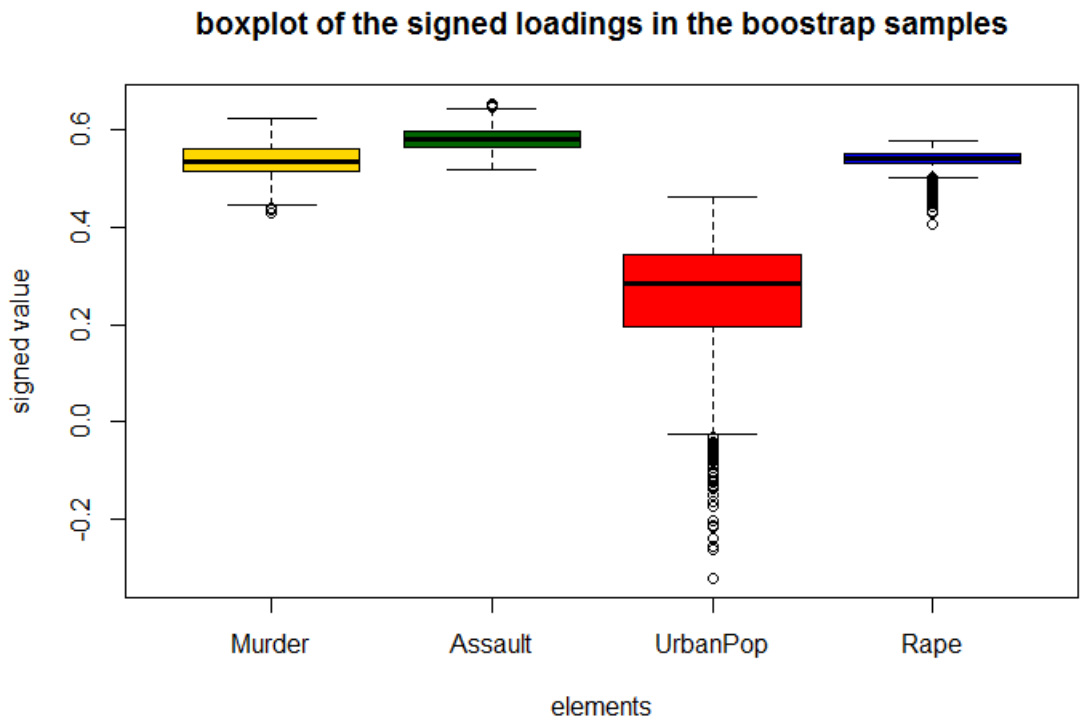
Computes the vector of loadings for the first principal component and defines i to be the index of the element with highest absolute value. For each of 1000 bootstrap resamplings of the data.frame, computes the vector of loadings for the first principal component and multiplies it by the sign of its ith element to generate signed loadings. Plots a boxplot of the signed loadings in the boostrap samples.

```
library(ISLR)
attach(USArrests)

## The following objects are masked from USArrests (pos = 3):
##
##     Assault, Murder, Rape, UrbanPop

set.seed(12)


signed_loadings_generator<- function(data){

   ## initialize output dataframe

   ## loop 1000 times
   for (n in 1:1000){

   ## sample data
   split = sample(1:nrow(data), size=nrow(data)/2)
   data_boot <- USArrests[split, ]
   ## do PCA
   PCA_data <- prcomp(data_boot,center = TRUE,scale. = TRUE)

   ## PC1
   pc_1<-PCA_data$rotation[,1]

   ##  index
   i <- which.max(abs(pc_1))

   ##  sign loading
   if (pc_1[i]<0) pc_1<- (-pc_1)

   ## rbind
   pc_1<-t(as.data.frame(pc_1))
   if (exists("pc1_loadings")) pc1_loadings<-rbind(pc1_loadings,pc_1)
   else pc1_loadings<-pc_1
   }
```

```
 ## plot
boxplot(pc1_loadings, main= "boxplot of the signed loadings in the boostrap samples" , col=(c("gold","darkgree
n","red","blue")),ylab="signed value" , xlab="elements")
}
```

(5) Apply the function to USArrests

```
signed_loadings_generator(USArrests)
```



(6) The function described in part 4 yields estimates of the standard error for each loading of the first principal component. On what assumption does this method rely? Would this give good standard error estimates for principal components beyond the first few? Explain.

- The loading values of elements derived from PCA could possibly go through a sign change, this can distort the distribution and standard deviation. The underlying assumptions is all elements should have same sign for a given bootstrap sample.
- As we do down the list of Principal components, generally variance increase, also we have the sign change problem, so estimated standard error terms may not be correct.