IS-777 Data Analytics

By Dr. Gunes Koru

Complete Project Report

Group C

Leroy Kim
Tristan Oetker
Kunal Mali
Ruchita Parab
Shailendra Singh
Rashmi Gangadharaiah

Table of Contents

D2: Project Characteristics and Descriptive Analysis

Requirement

This deliverable will report the main characteristics of the data set after the cleaning steps found to be necessary by the group, and provide descriptive analysis results

* n:
* p (number of parameters):
* Response Variable
* Predictor Variables

* Descriptive Analysis
** Summary statistics obtained from R for each variable. These include mean, median, and quartiles along with some other statistics
** Histograms for quantitative variables and barcharts for the qualitative variables all produced in R
** R source code which can reproduce all analysis steps including importing data and plotting.

* Analysis Plan: Discuss your current plan about how the rest of your analysis will proceed.

Your deliverable should include a zip file and uploaded by only the project manager before the due date and time. Your zip file MUST include only two files: One file will be for your report; the other will be for your R script. All figures and tables should be embedded in your report.

Cleaning the data

Descriptive analysis

This chapter includes following informations for each data file:

- n
- p (number of parameters)
- Summary statistics obtained from R for each variable. These include mean, median, and quartiles along with some other statistics
- Histograms for quantitative variables and barcharts for the qualitative variables all produced in R
- R source code which can reproduce all analysis steps including importing data and plotting.

1. Cutilization

Summary statistics

Code:

```
>#import cleaned cutilization csv file
>setwd("D:\\Fall 2018\\IS 777 Data Analytics\\D2\\Plotting-kkm")
>df<-read.csv('Cutilization_cleaned.csv')
>summary(df)
```

Output:

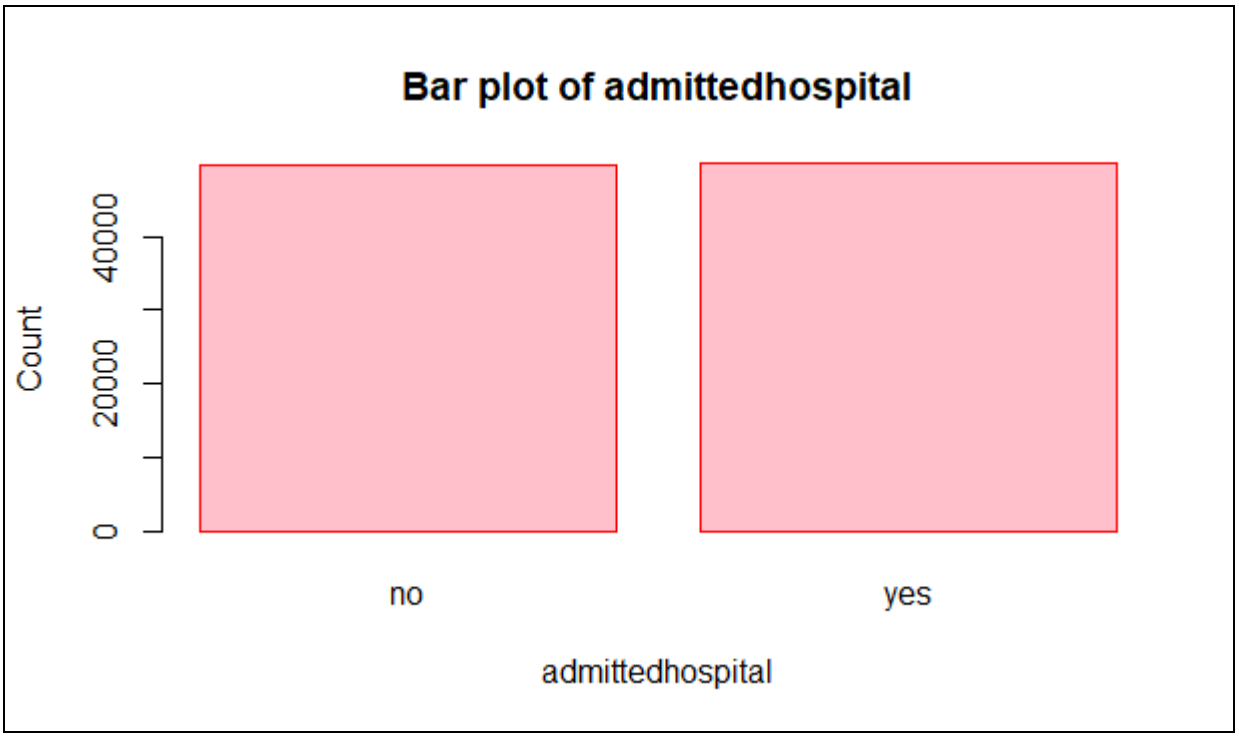
```
> summary(df)
      id        admittedhospital urgent      readmittedhospital emergencyhospital      LOS
 Min.   :    1   no :49689        no :51483   no :59002          no :79626         Min.   : 0.01
 1st Qu.:24887   yes:49857        yes:48063   yes:40544          yes:19920         1st Qu.:18.95
 Median :49774                                                                     Median :25.60
 Mean   :49774                                                                     Mean   :25.60
 3rd Qu.:74660                                                                     3rd Qu.:32.12
 Max.   :99546                                                                     Max.   :71.06
>
```

Bar plot for admittedhospital

Code:

```
>#barplotting of qualitative values in cutilization
>barplot(table(charts$admittedhospital),main="Bar    plot    of    admittedhospital",
xlab="admittedhospital",
   ylab="Count", border="red", col="pink")
```

Output:

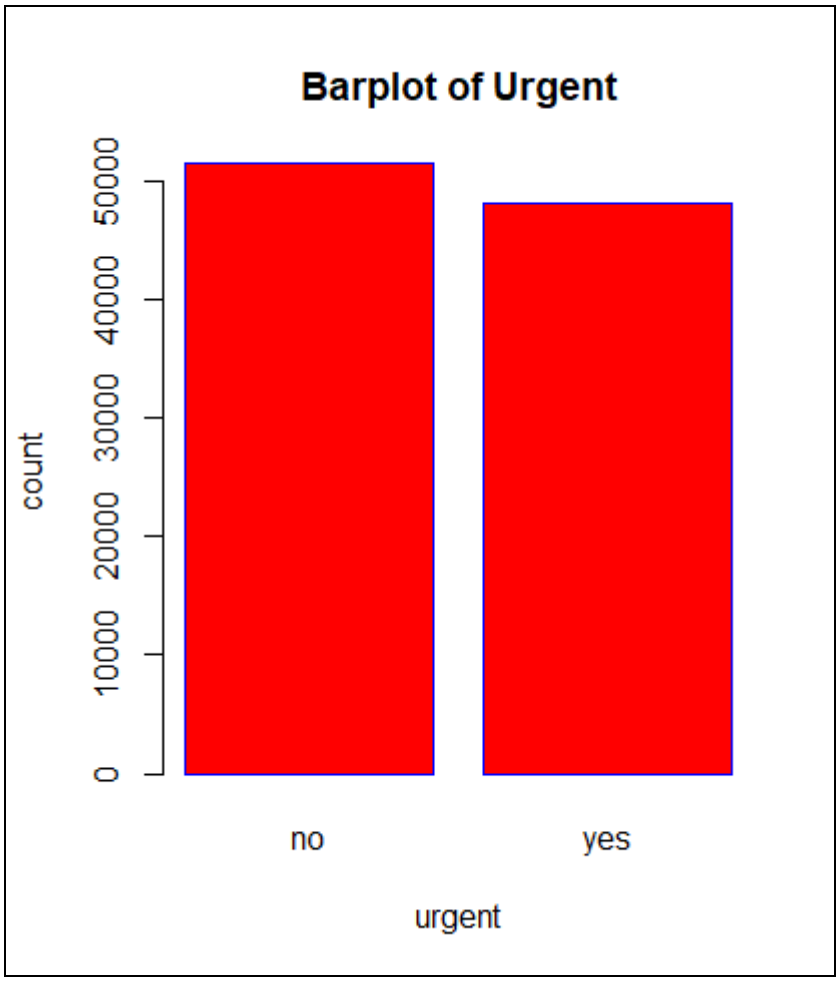


Bar plot for Urgent

Code:

```
>#barplotting of qualitative values in cutilization
>barplot(table(df$urgent), main = "Barplot of Urgent",
 xlab = "urgent", ylab = "count", border = "blue", col =
 "red")
```

Output:



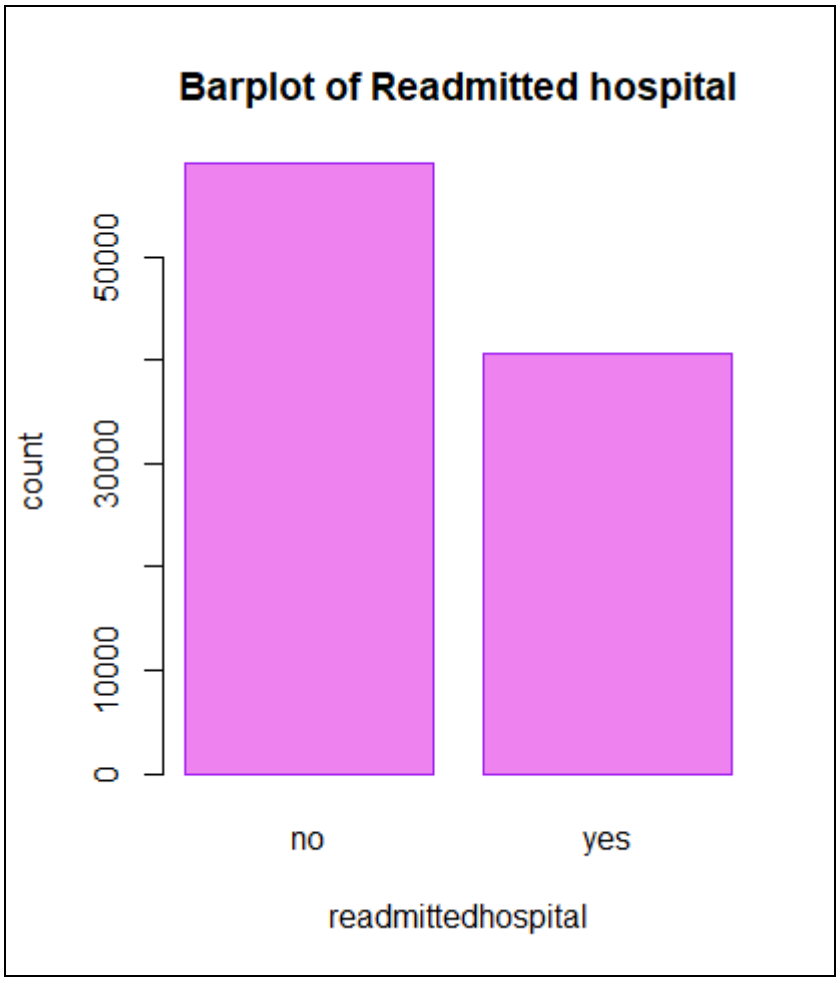
Bar plot for readmittedhospital

Code:

```
>#barplotting of qualitative values in cutilization
>barplot(table(df$readmittedhospital), main = "Barplot of
 Readmitted hospital", xlab = "readmittedhospital", ylab =
 "count", border = "purple", col = "violet")
```

Output:

:

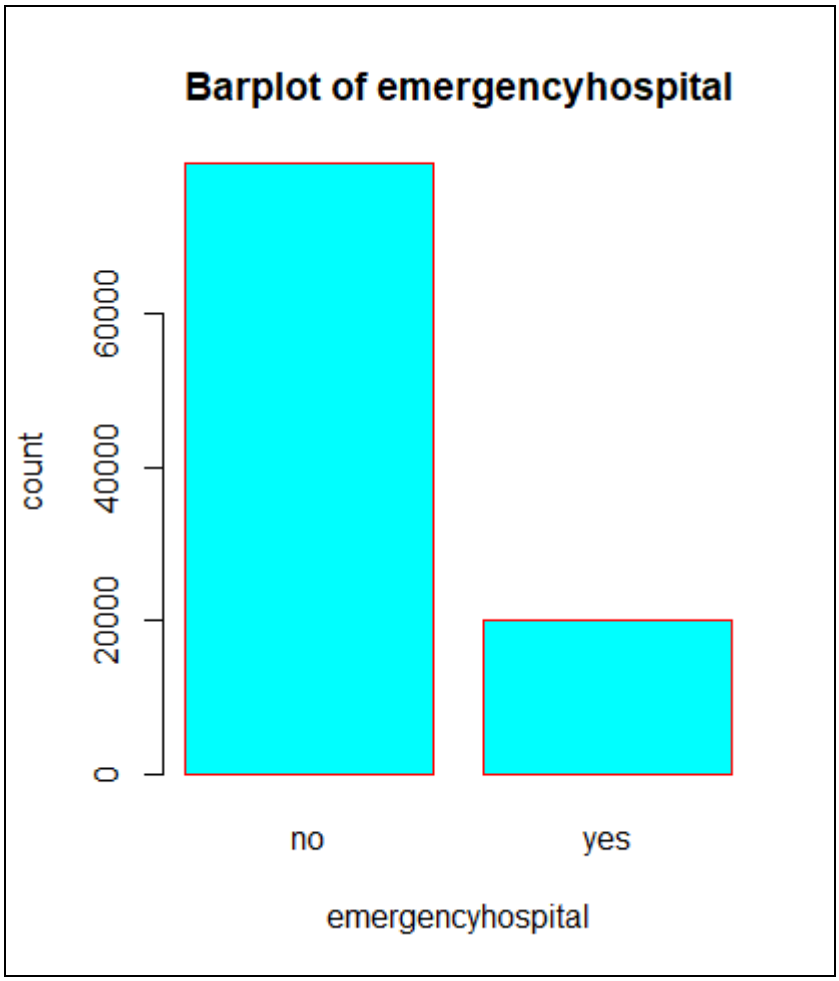


Bar plot for emergencyhospital

Code:

```
>#barplotting of qualitative values in cutilization
>barplot(table(df$emergencyhospital), main = "Barplot of
 emergencyhospital", xlab = "emergencyhospital", ylab =
 "count", border = "red", col = "cyan")
```

Output:

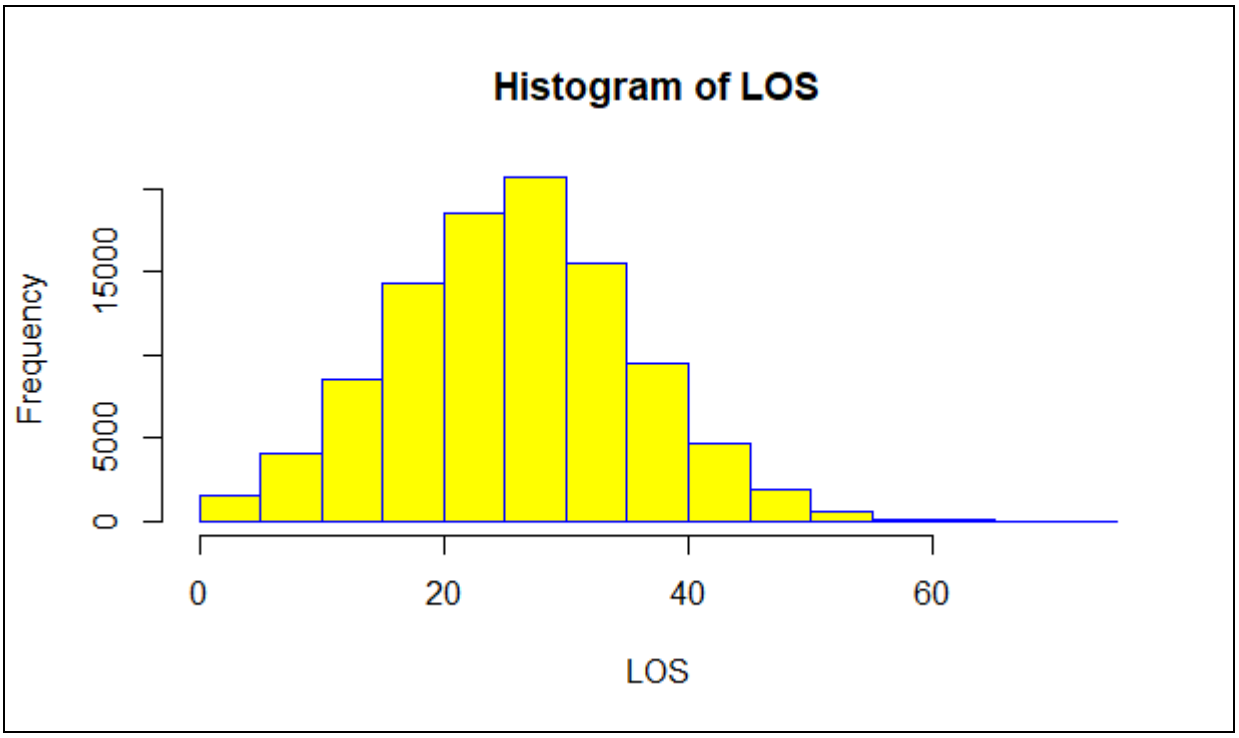


Histogram for LOS

Code:

```
>charts<-read.csv("E:\\Fall 2018\\IS 777\\D2\\cutilization_v4.csv")
>hist(charts$LOS, main="Histogram of LOS", xlab="LOS", ylab = "Number of Days",
border="blue",
   col="yellow")
```

Output:



Cquality

Summary Statistics

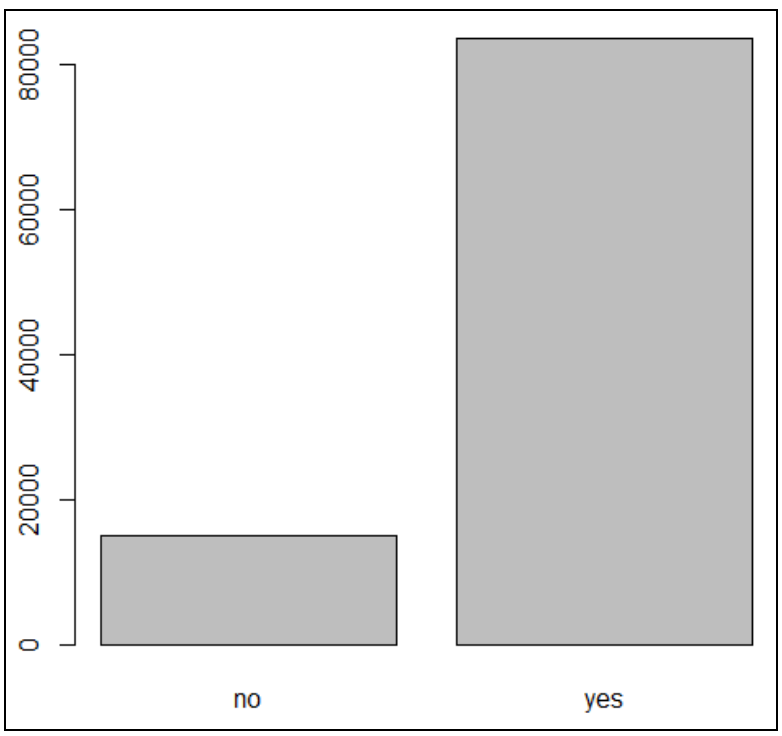
```
       id          istimely     taughtdrug   checkfall    checkdepression
 Min.   :     1   no :14944    no : 3086    no :49007    no :51345
 1st Qu.:24880    yes:83470    yes:95328    yes:49407    yes:47069
 Median :49785
 Mean   :49774
 3rd Qu.:74665
 Max.   :99546
 checkflushot checkvaccine checkfootcare HHTcare      HHTcomm      HHTdiscuss
 no : 9959    no :49100    no :49303     no : 3803    no : 1965    no : 1231
 yes:88455    yes:49314    yes:49111     yes:94611    yes:96449    yes:97183
```

Bar plot for istimely

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$istimely))
```

Output:

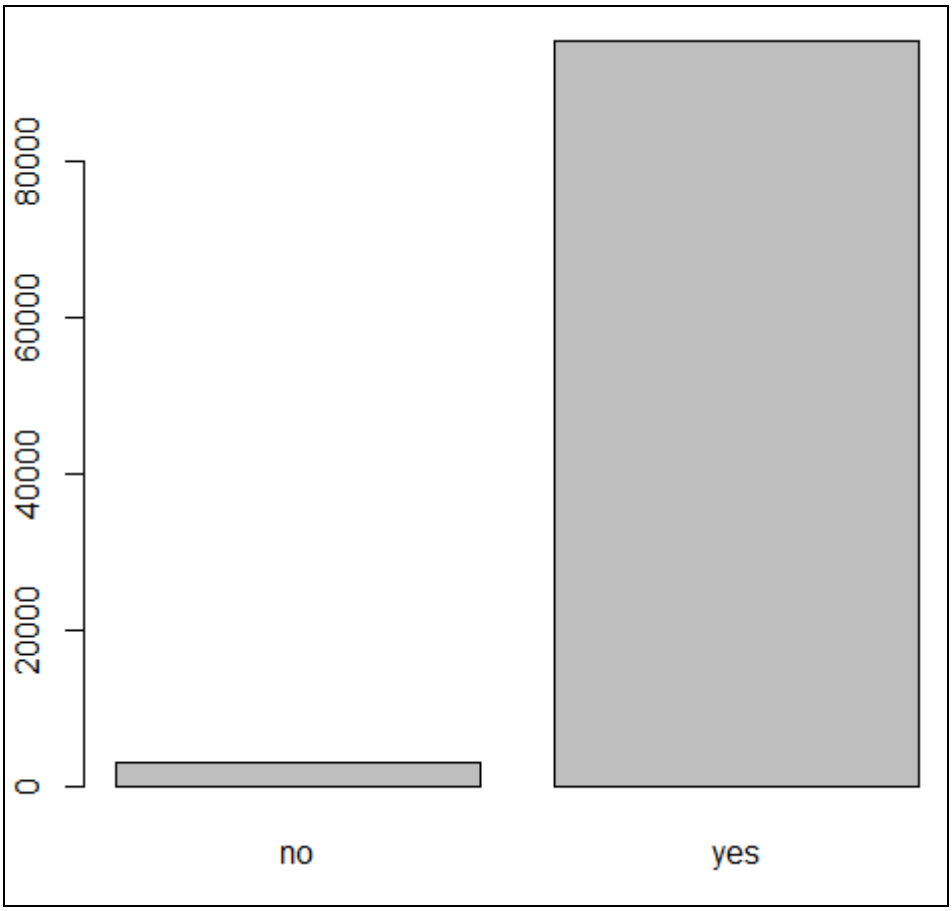


Bar plot for taughtdrug

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$taughtdrug))
```

Output:

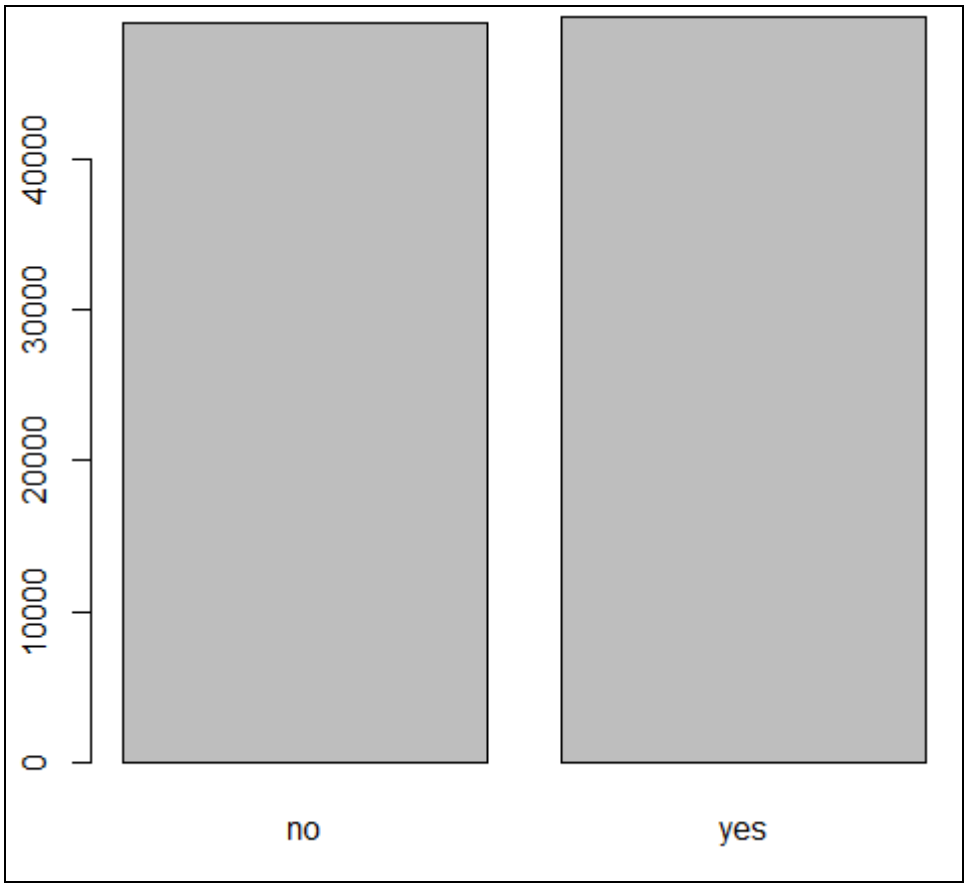


Bar plot for checkfall

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$checkfall))
```

Output:

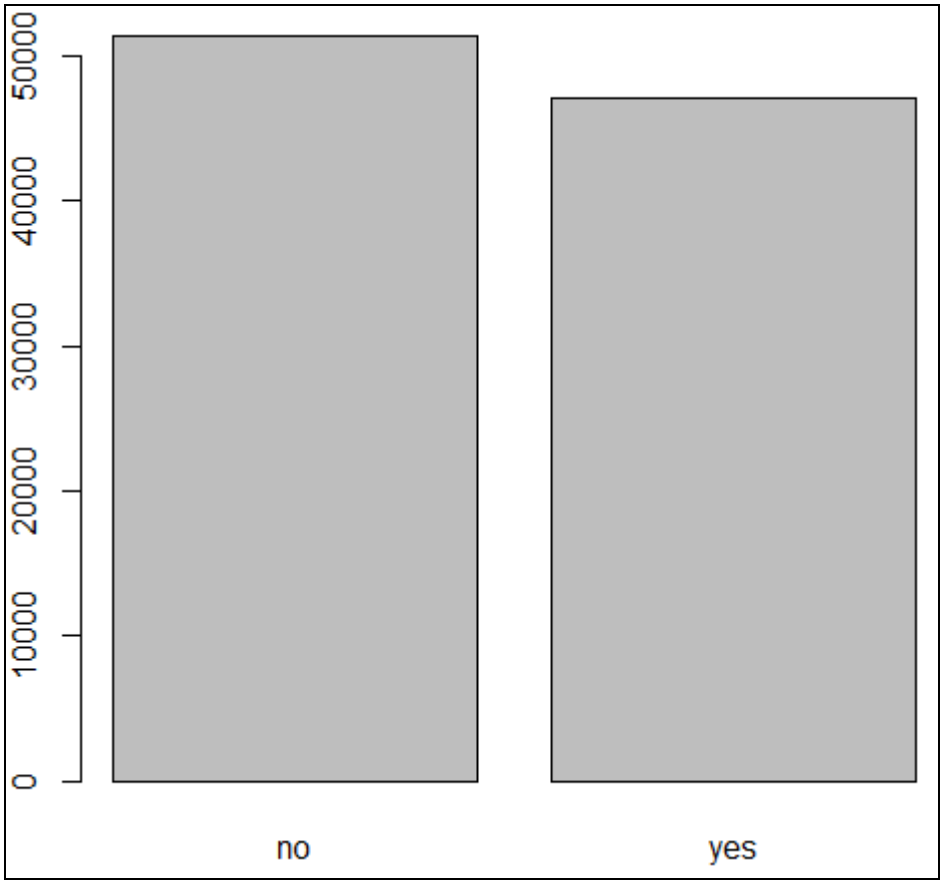


Bar plot for checkdepression

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$checkdepression))
```

Output:

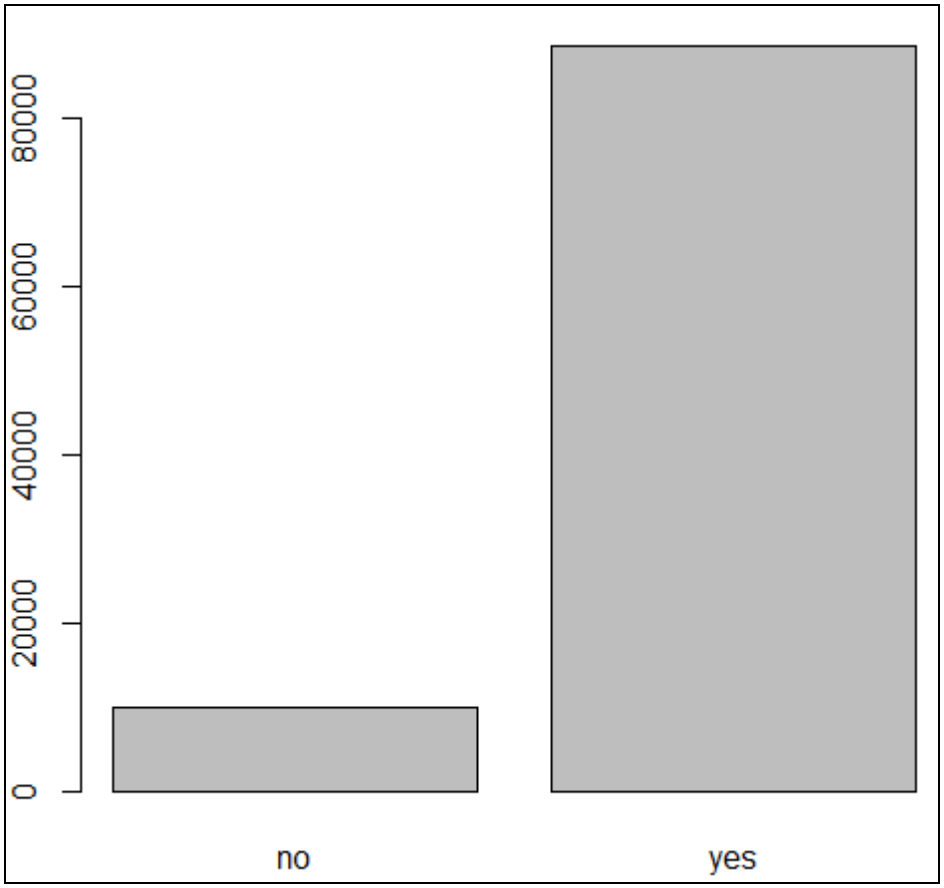


Bar plot for checkflushot

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$checkflushot))
```

Output:

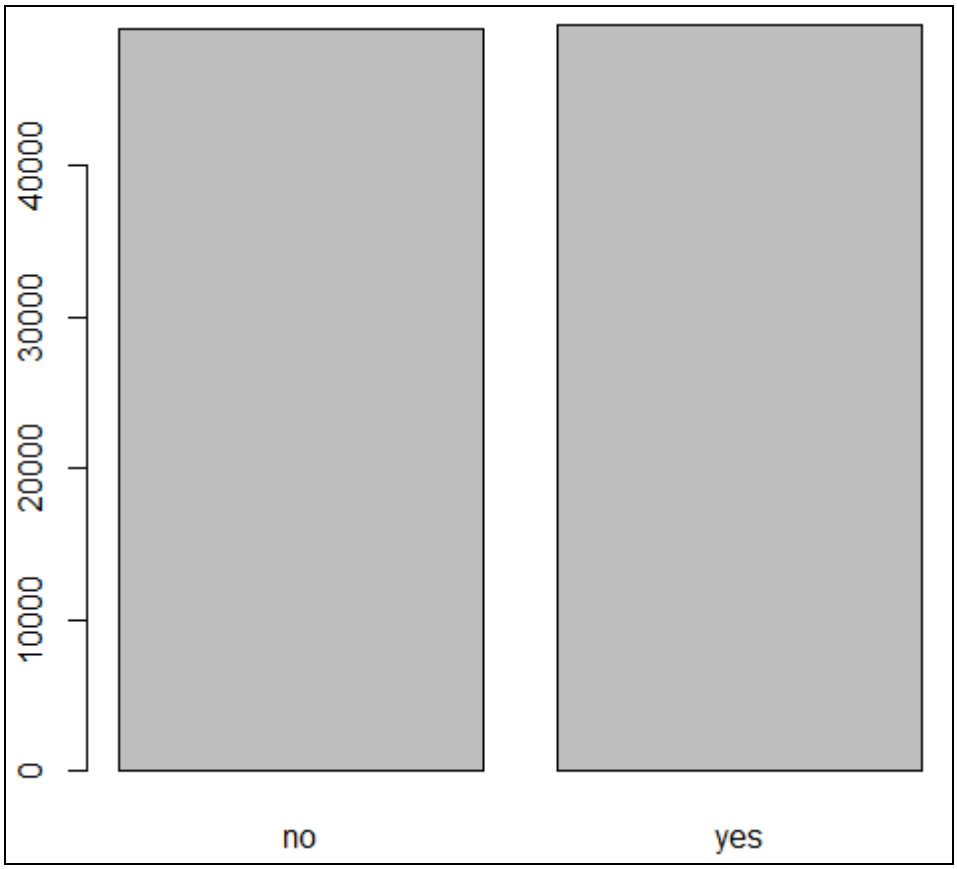


Bar plot for checkvaccine

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$checkvaccine))
```

Output:

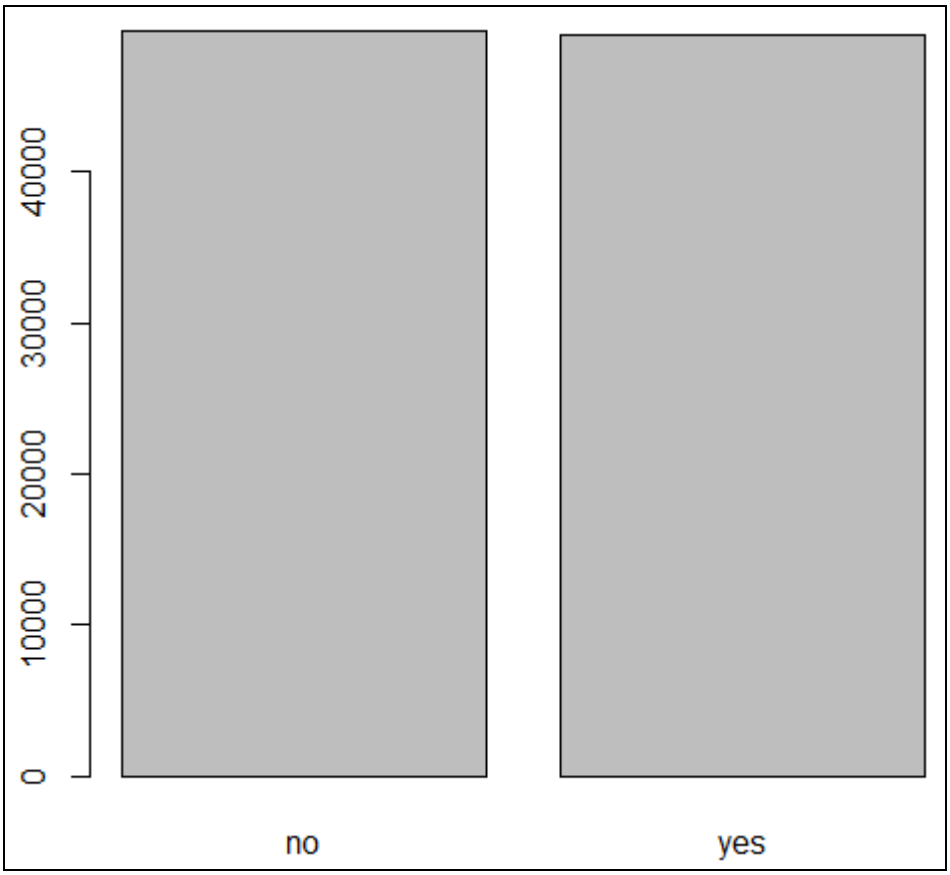


Bar plot for checkfootcare

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$checkfootcare))
```

Output:

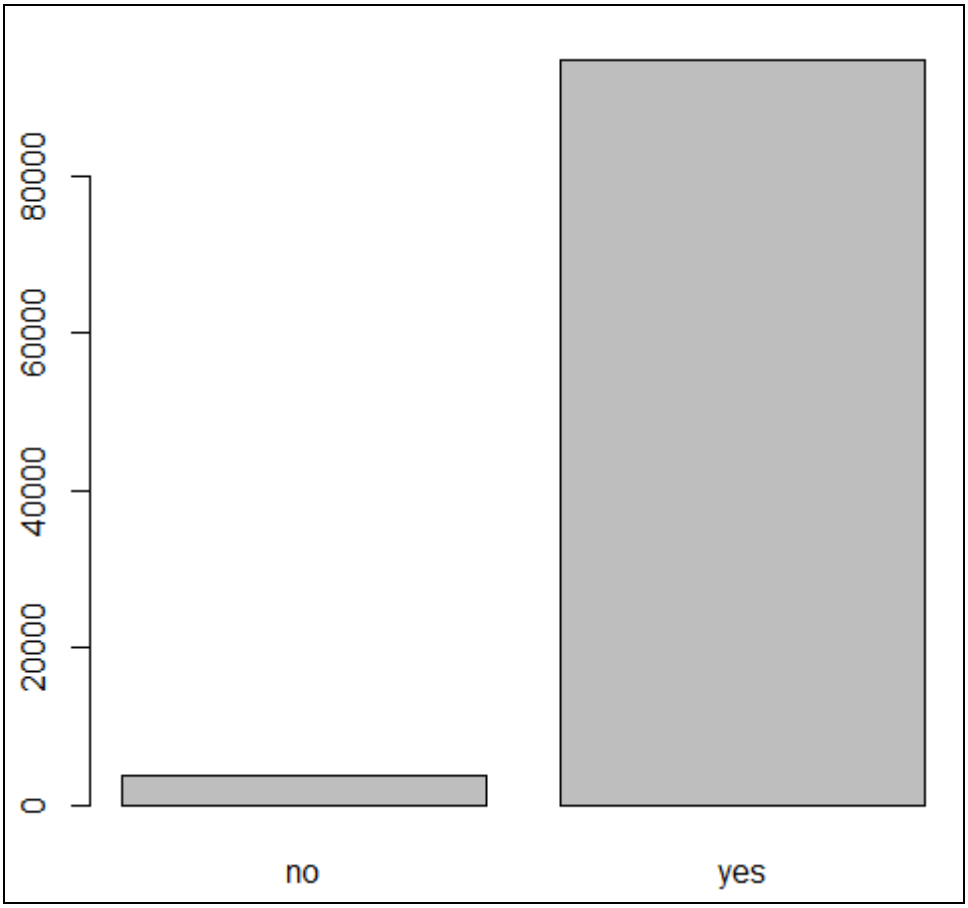


Bar plot for HHTcare

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$HHTcare))
```

Output:

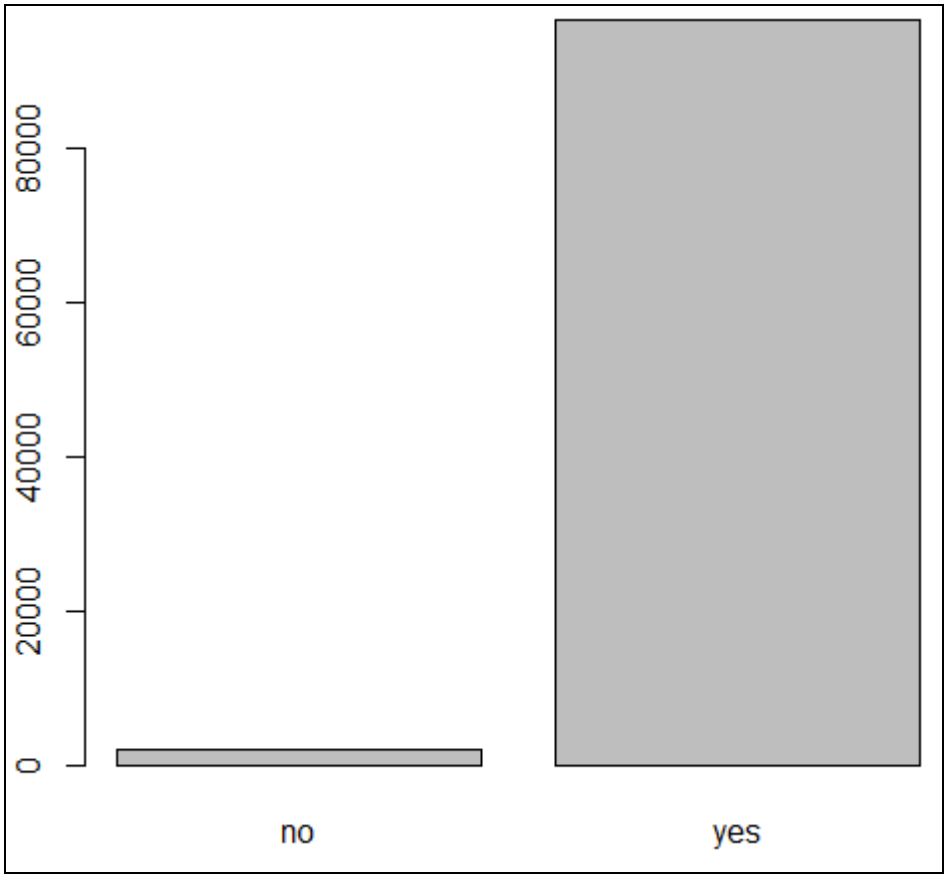


Bar plot for HHTcomm

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$HHTcomm))
```

Output:

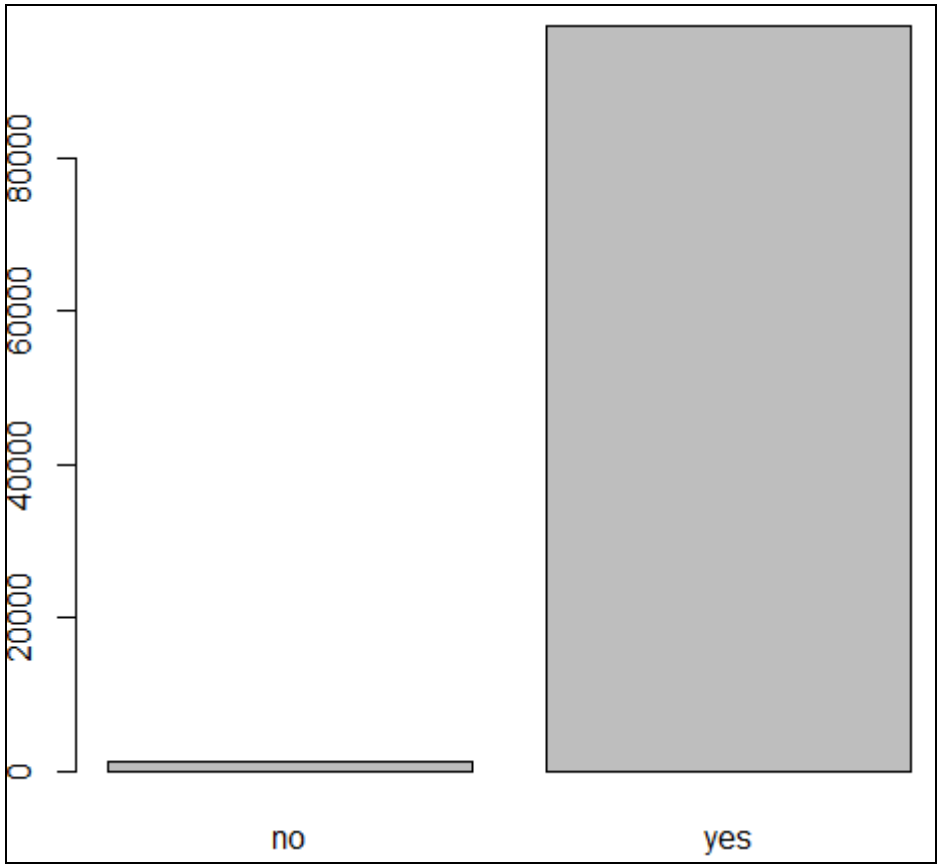


Bar plot for HHTdiscuss

Code:

```
> cquality <- read.csv(file.choose(), header=TRUE, sep=",")
> barplot(table(cquality$HHTdiscuss))
```

Output:



Cdemoses

Summary Statistics

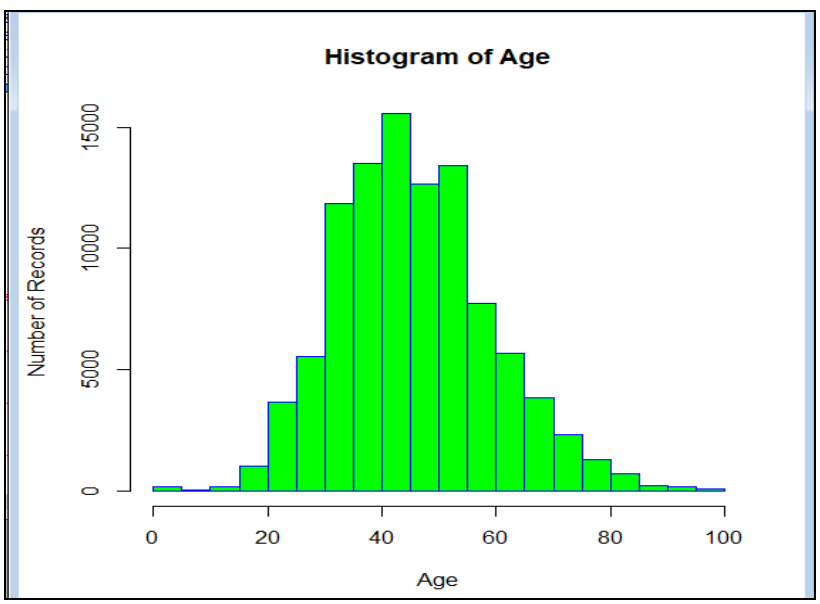
```
> summary(CleanCdemoses)
       id              age            gender        race        marital
 Min.   :     1   Min.   : 0.00   female:57449   AFA: 6310   DS : 3944
 1st Qu.:24887    1st Qu.:36.00   male  :42098   ASA: 7412   MP : 3880
 Median :49774    Median :44.89                  EUA: 9454   OTH:83834
 Mean   :49774    Mean   :45.73                  HLA: 7545   SG : 3942
 3rd Qu.:74660    3rd Qu.:54.76                  OTH:68826   WD : 3947
 Max.   :99546    Max.   :98.01
 NA's   :1
    area            income        education
 rural:46056   Min.   :  0.00   GD : 7573
 urban:53491   1st Qu.: 53.14   HS :12340
               Median : 86.87   OTH:72680
               Mean   : 94.81   UD : 6954
               3rd Qu.:126.68
               Max.   :511.09
```

Histogram for age

Code:

>hist(CleanCdemoses$age, main="Histogram of Age", xlab="Age", ylab = "Number of Records", border="blue", col="green")

Output:

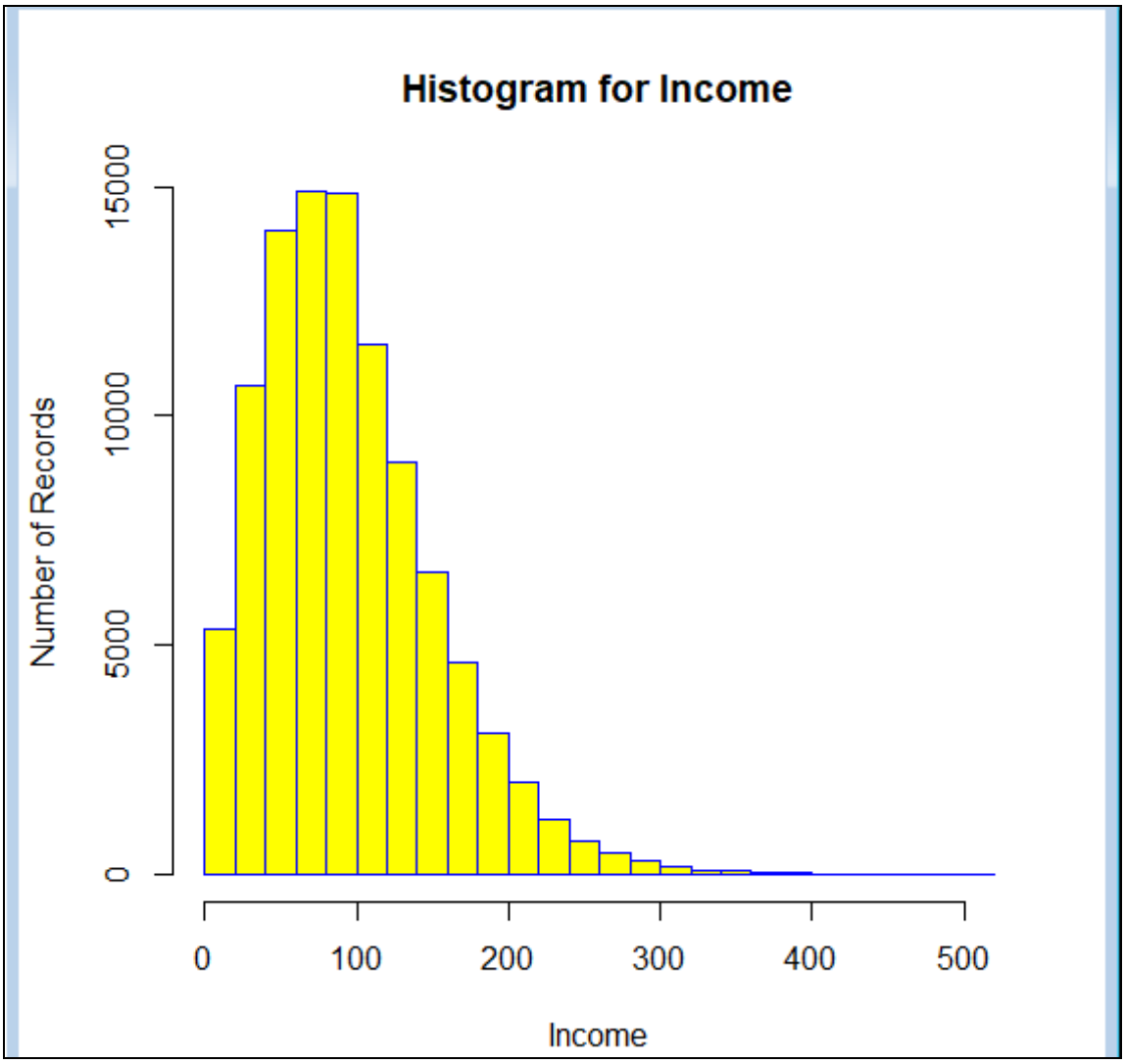


Histogram for Income

Code:

```
>hist(CleanCdemoses$income, main="Histogram for Income", xlab="Income", ylab = "Number of Records", border="blue", col="yellow", breaks=20)
```

Output:

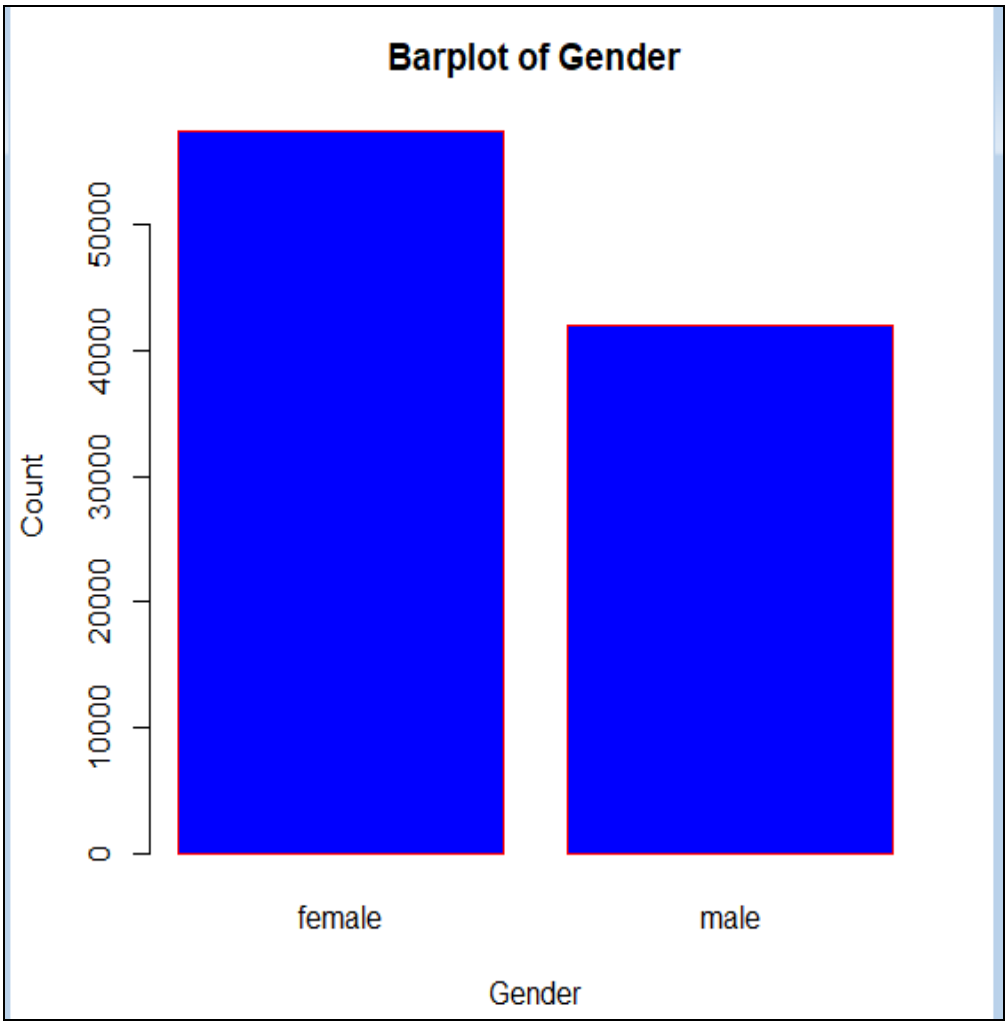


Bar plot for gender

Code:

```
>barplot(table(CleanCdemoses$gender),main="Barplot of Gender", xlab="Gender", ylab="Count",
border="red", col="blue")
```

Output:

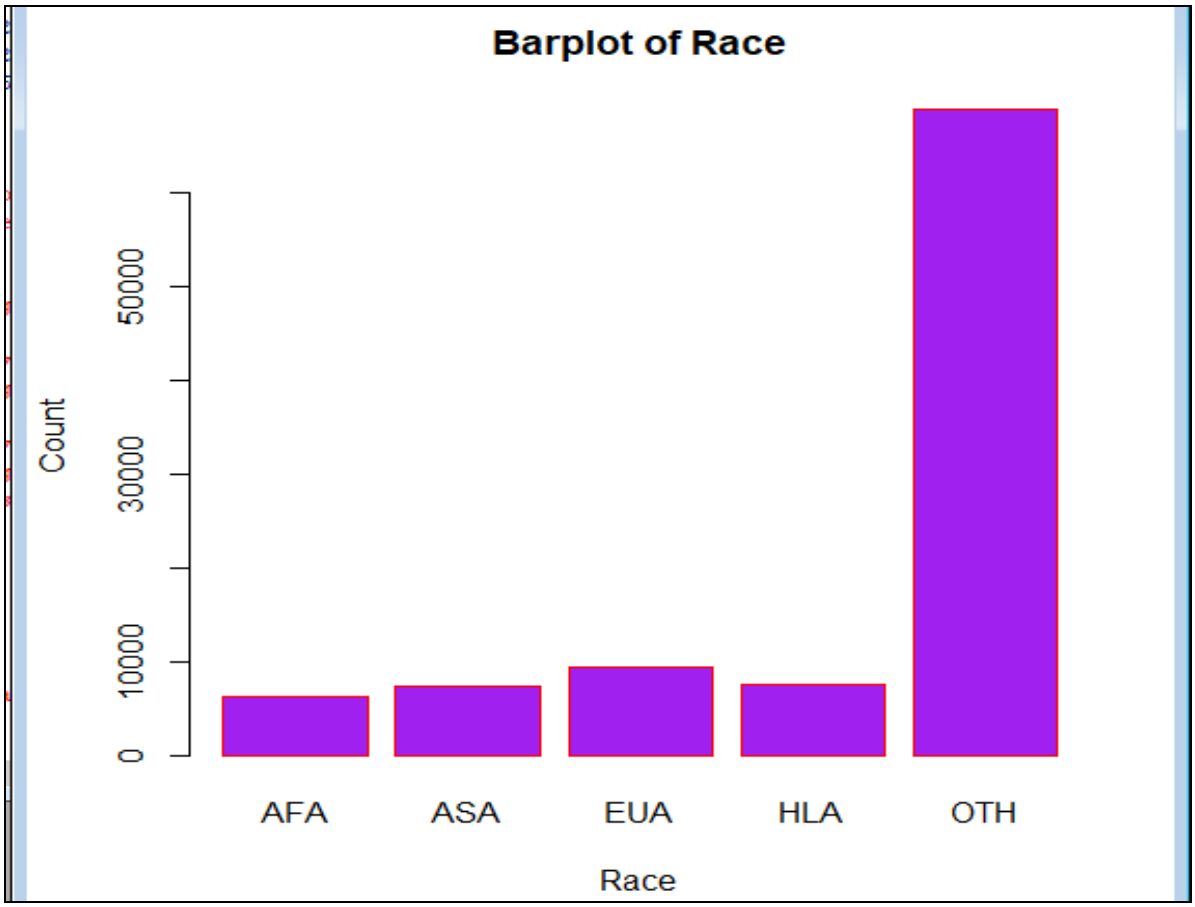


Bar plot for race

Code:

```
>barplot(table(CleanCdemoses$race),main="Barplot of Race", xlab="Race", ylab="Count", border="red", col="purple")
```

Output:

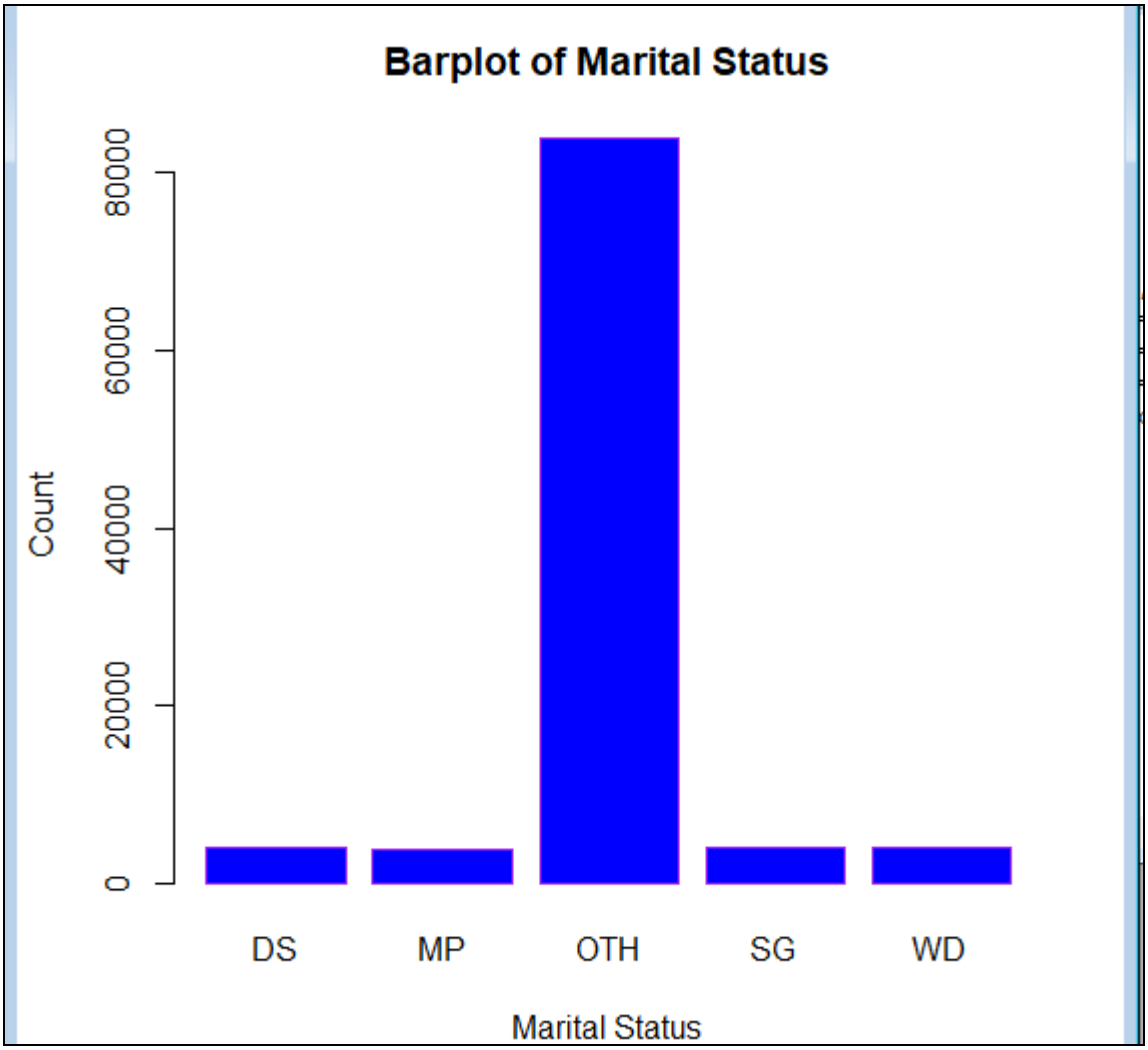


Bar plot for marital

Code:

```
>barplot(table(CleanCdemoses$marital),main="Barplot of Marital Status", xlab="Marital Status",
ylab="Count", border="purple", col="blue")
```

Output:

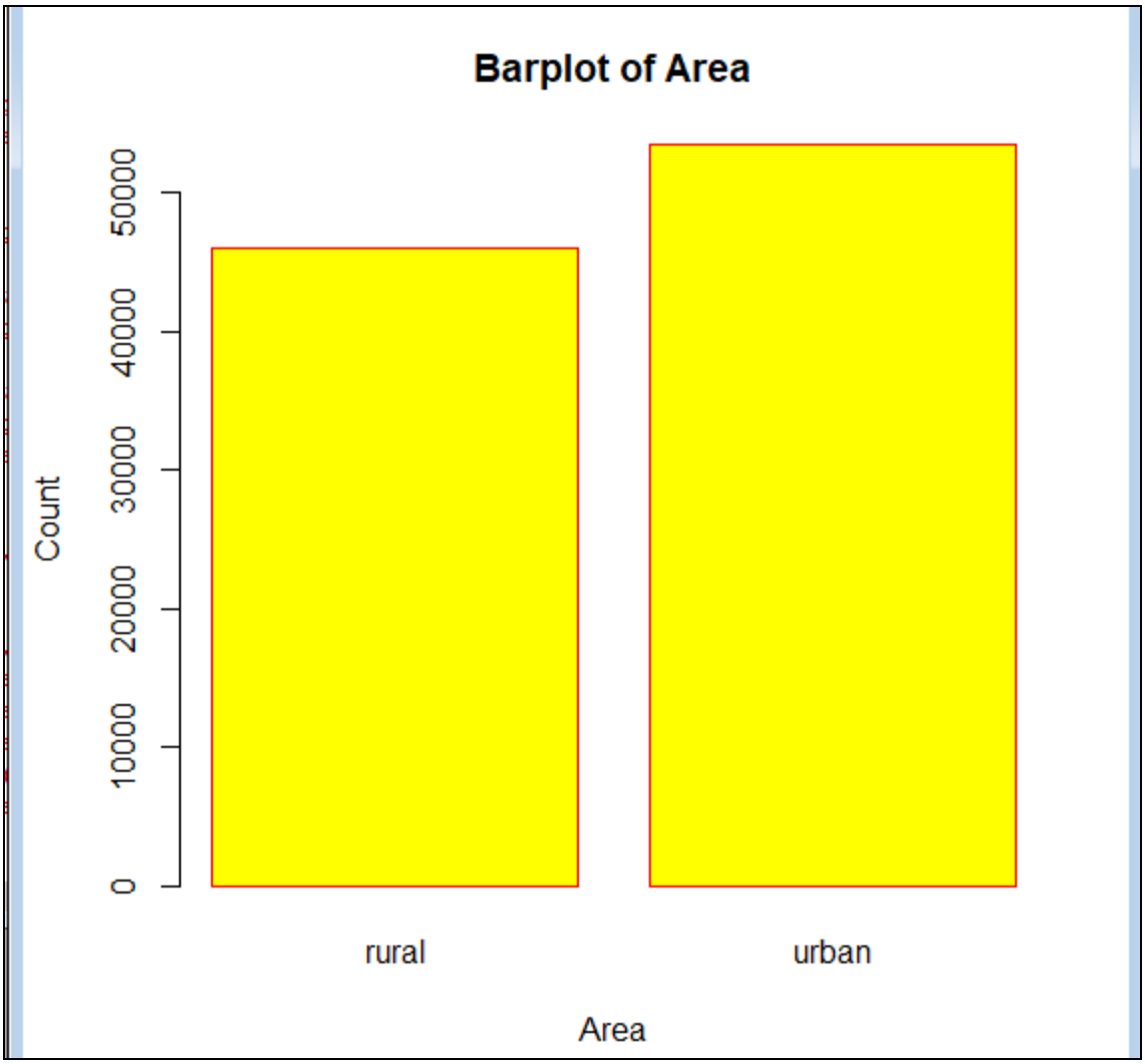


Bar plot for area

Code:

```
>barplot(table(CleanCdemoses$area),main="Barplot of Area", xlab="Area", ylab="Count", border="red", col="yellow")
```

Output:

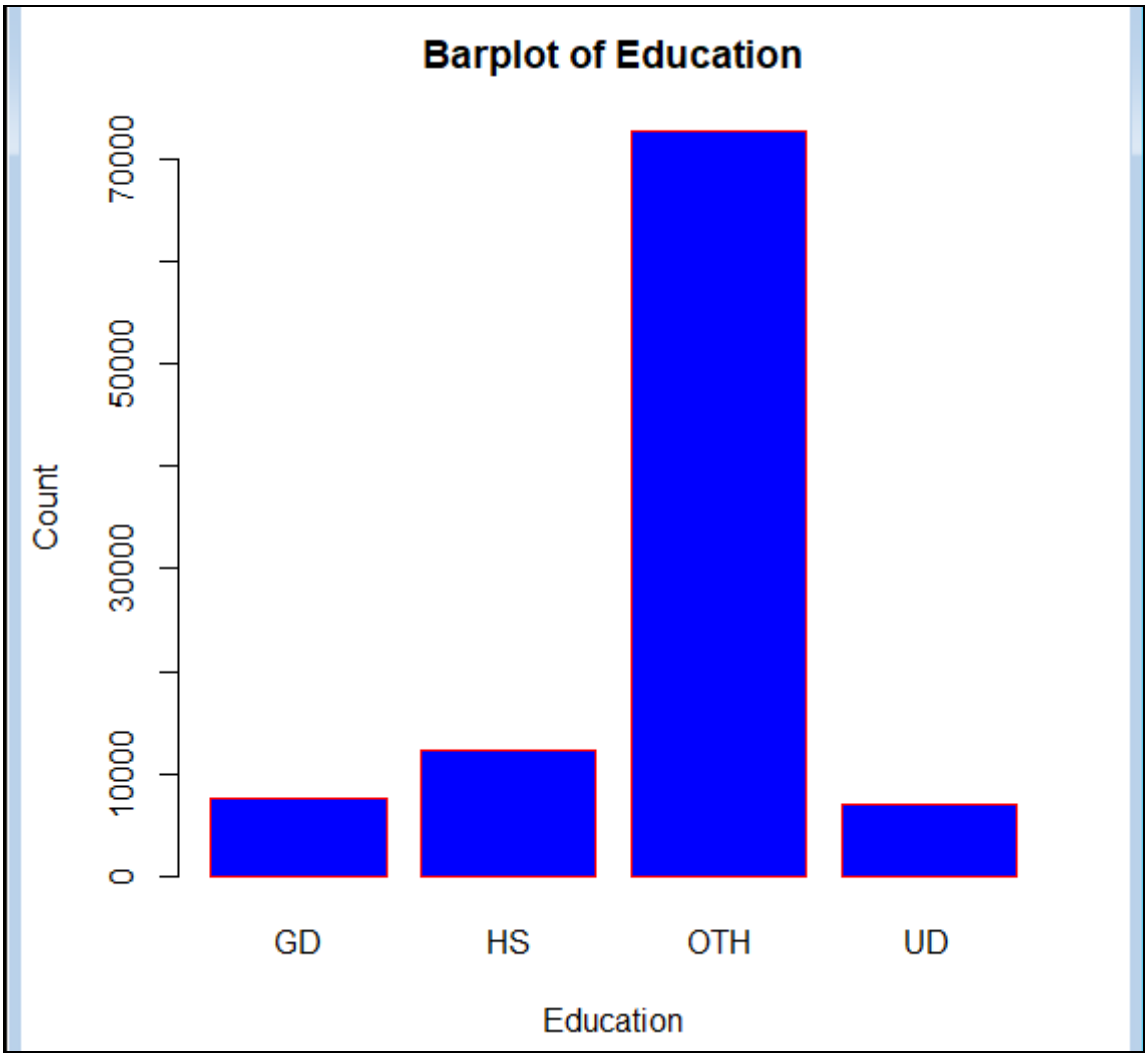


Bar plot for education

code:

```
>barplot(table(CleanCdemoses$education),main="Barplot of Education", xlab="Education", ylab="Count", border="red", col="blue")
```

Output:



Chealth

Summary Statistics

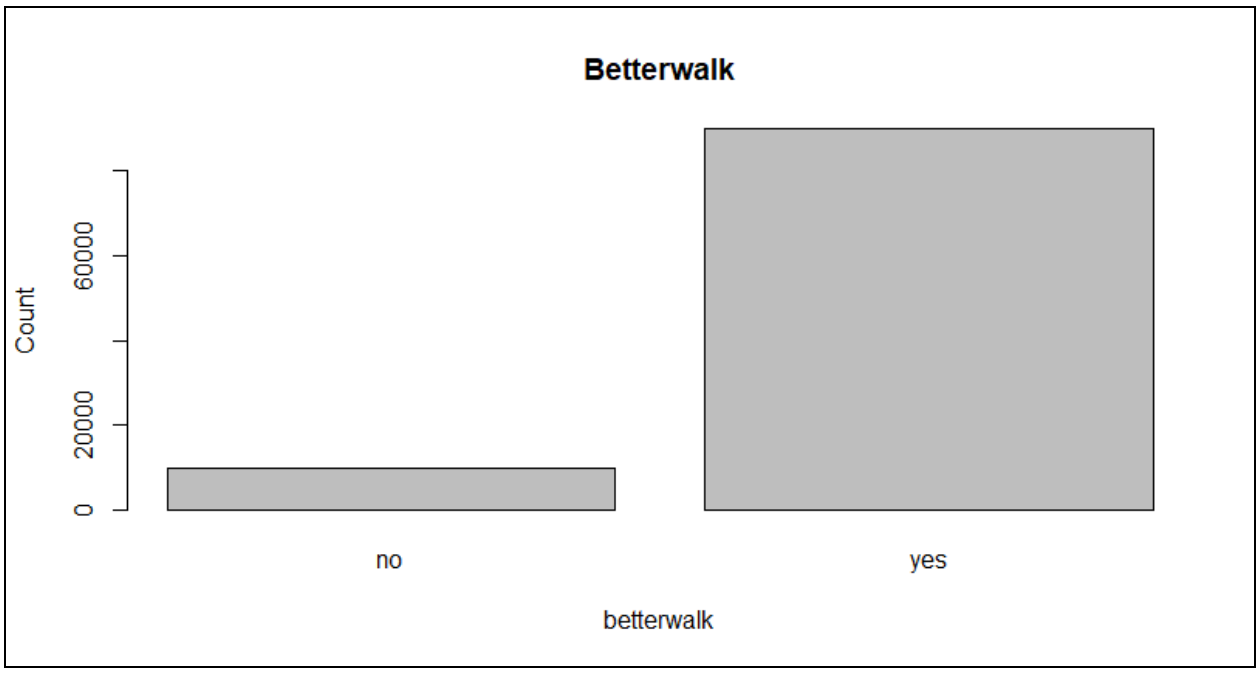
```
df))
  mortscore       betterwalk  betterbed   betterbath  bettermove  betterbreath betterheal  bettertaking adverseevent  drugu       alcolu
 Min.   :-221.033  no : 9812   no :49755   no :  413   no :49526   no : 1192    no : 1197   no : 1591    AE   :19525   no :59466   no :59912
 1st Qu.:  -6.549  yes:89814   yes:49871   yes:99213   yes:50100   yes:98434    yes:98429   yes:98035    AR   : 3779   yes:40160   yes:39714
 Median :  33.777                                                                                         SAE  :27791
 Mean   :  33.850                                                                                         SSAR :11463
 3rd Qu.:  74.362                                                                                         SUSAR:37068
 Max.   : 295.973
```

Bar plot for betterwalk

Code:

>barplot(table(df$betterwalk), main="Betterwalk", xlab="betterwalk", ylab="Count", col="red", border="black")

Output:



Bar plot for betterbed

Code:

```
>barplot(table(df$betterbed), main="BetterBed", xlab="betterbed", ylab="Count", col="blue",
border="black")
```

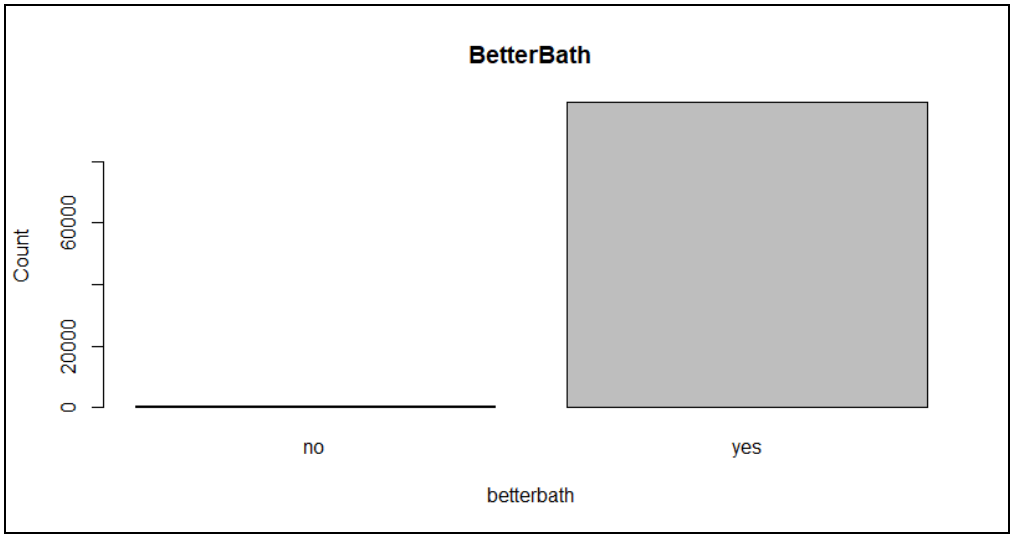
Output:

Bar plot for betterbath

Code:

barplot(table(df$betterbath), main="BetterBath", xlab="betterbath",ylab="Count", col="blue", border="black")

Output:

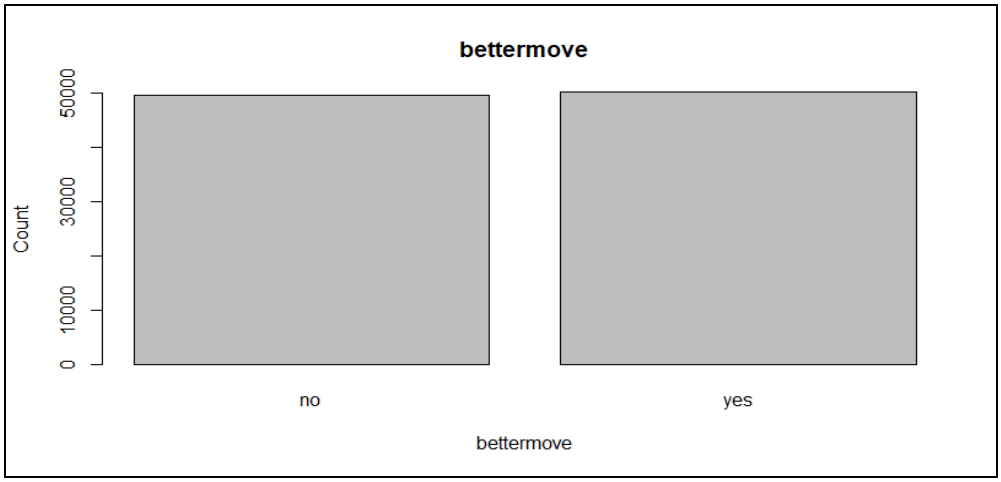


Bar plot for bettermove

Code:

barplot(table(df$bettermove), main="bettermove", xlab="bettermove", ylab="Count", col="blue", border="black")

Output:

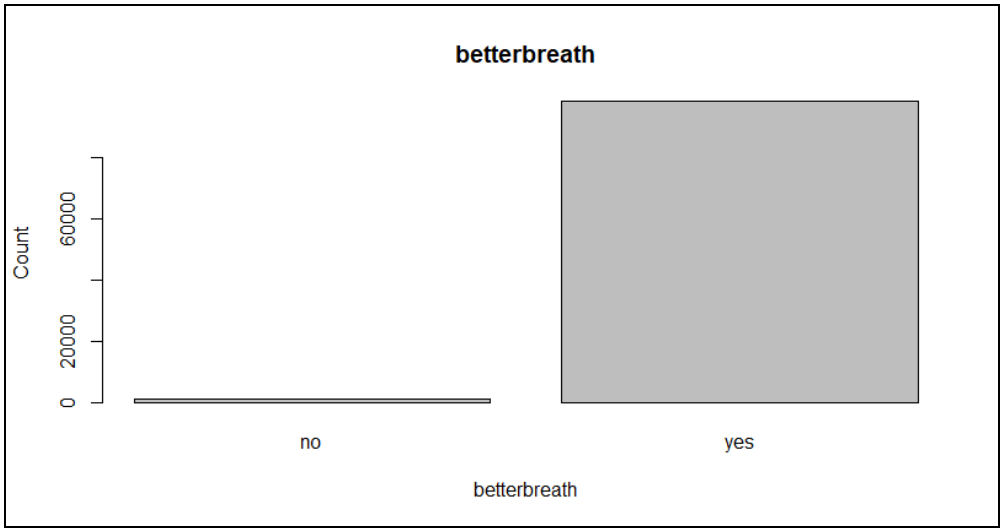


Bar plot betterbreath

Code:

barplot(table(df$betterbreath), main="betterbreath", xlab="betterbreath", ylab="Count", col="blue", border="black")

Output:

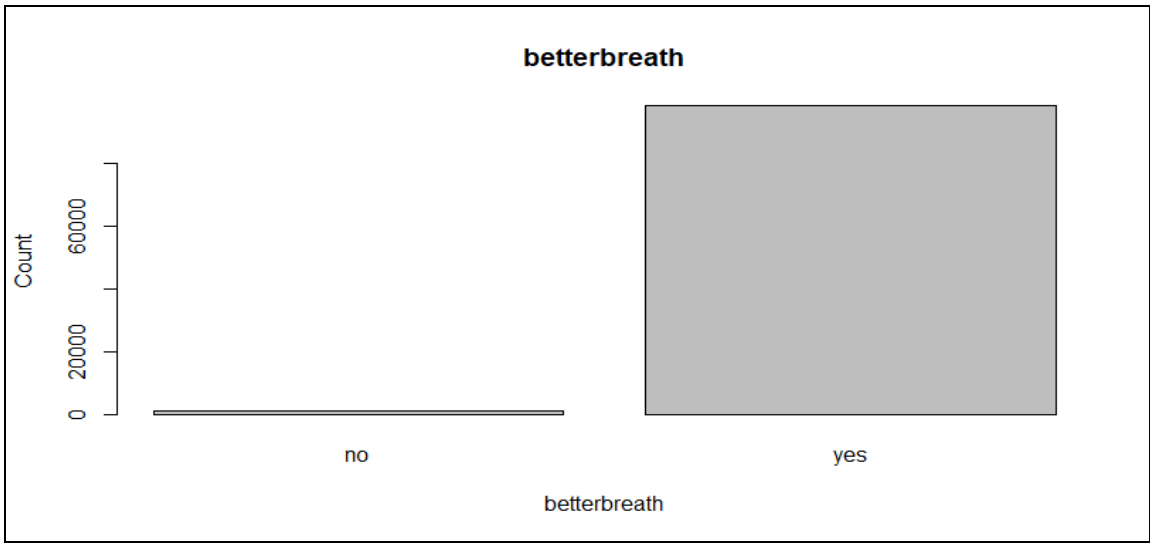


Better Heal:

Code:

barplot(table(df$betterheal), main="betterheal", xlab="betterheal", ylab="Count", col="blue", border="black")

Output:

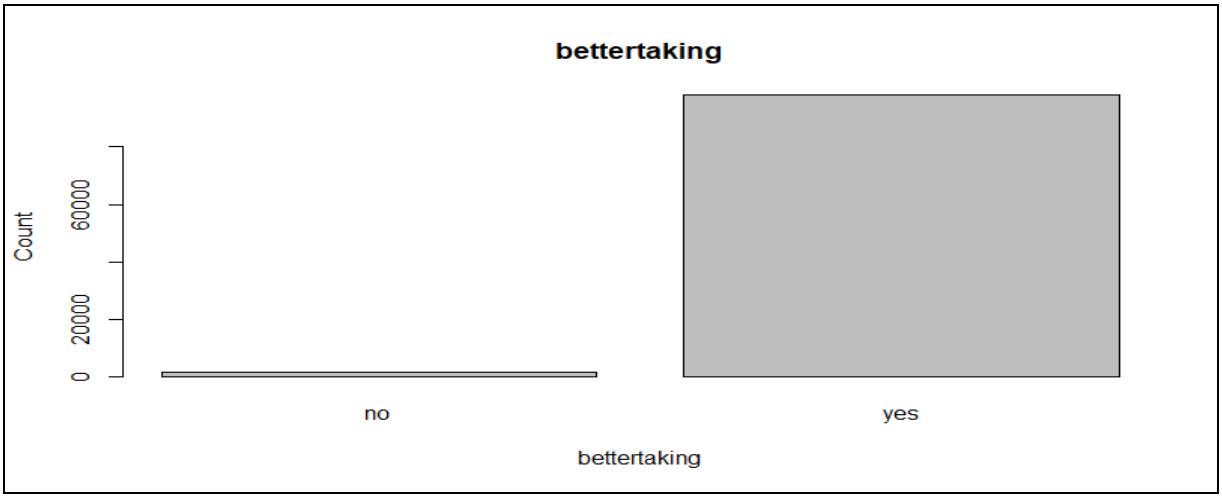


Bar plot for Better Taking:

Code:

barplot(table(df$bettertaking), main="bettertaking", xlab="bettertaking", ylab="Count", col="blue", border="black")

Output:

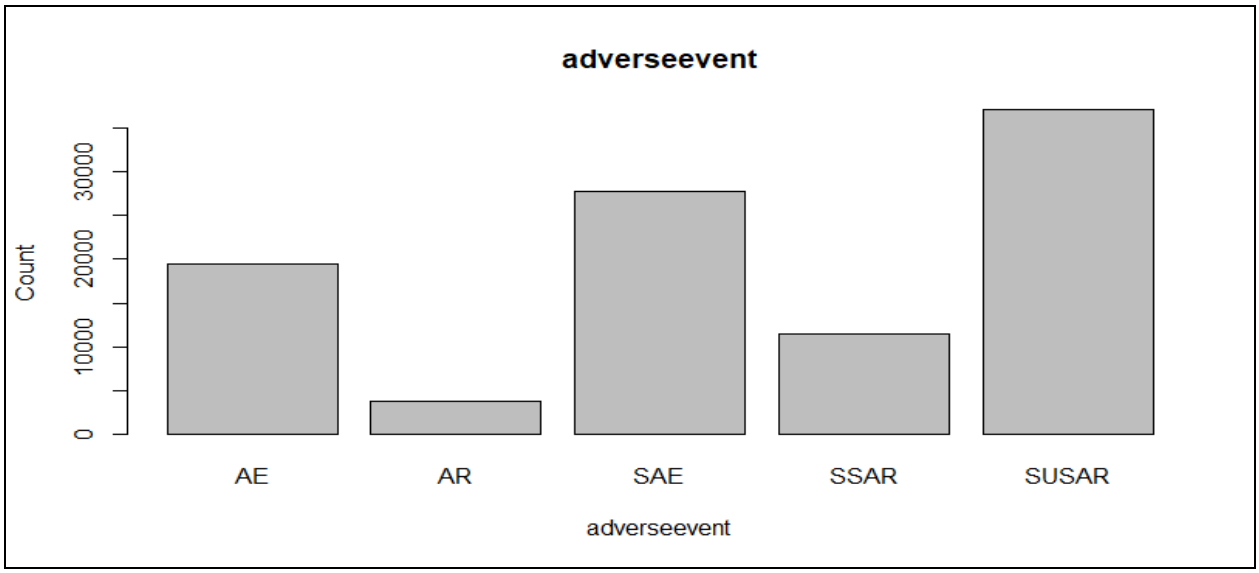


Adverse Event:

Code:

barplot(table(df$adverseevent), main="adverseevent", xlab="adverseevent", ylab="Count", col="blue", border="black")

Output:

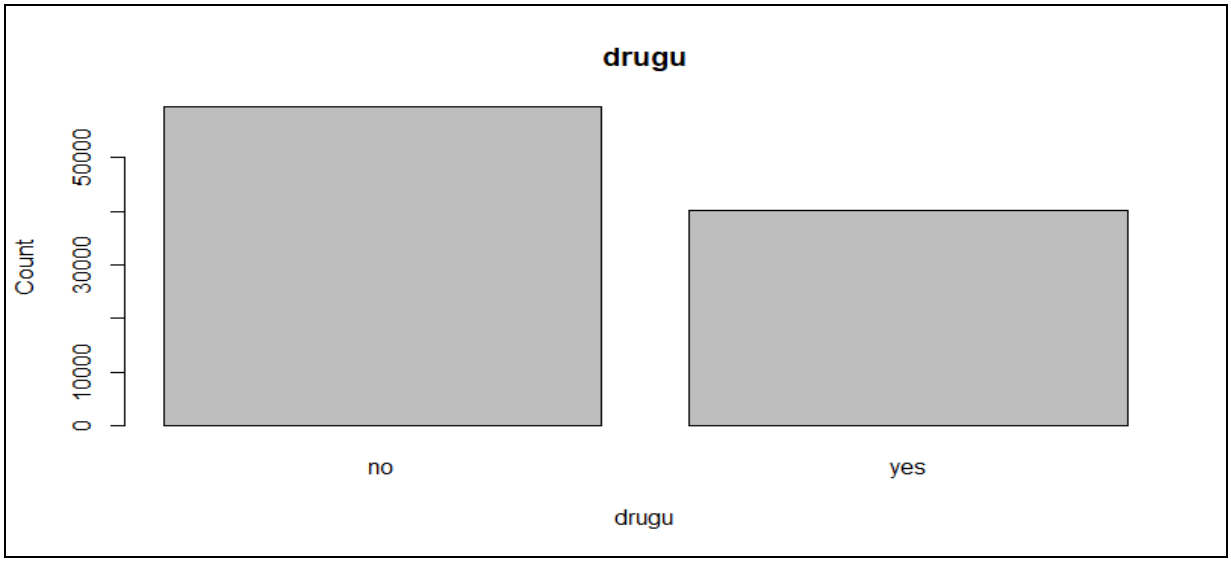


Drugu:

Code:

```
>barplot(table(df$drugu), main="drugu", xlab="drugu", ylab="Count", col="blue", border="black")
```

Output:

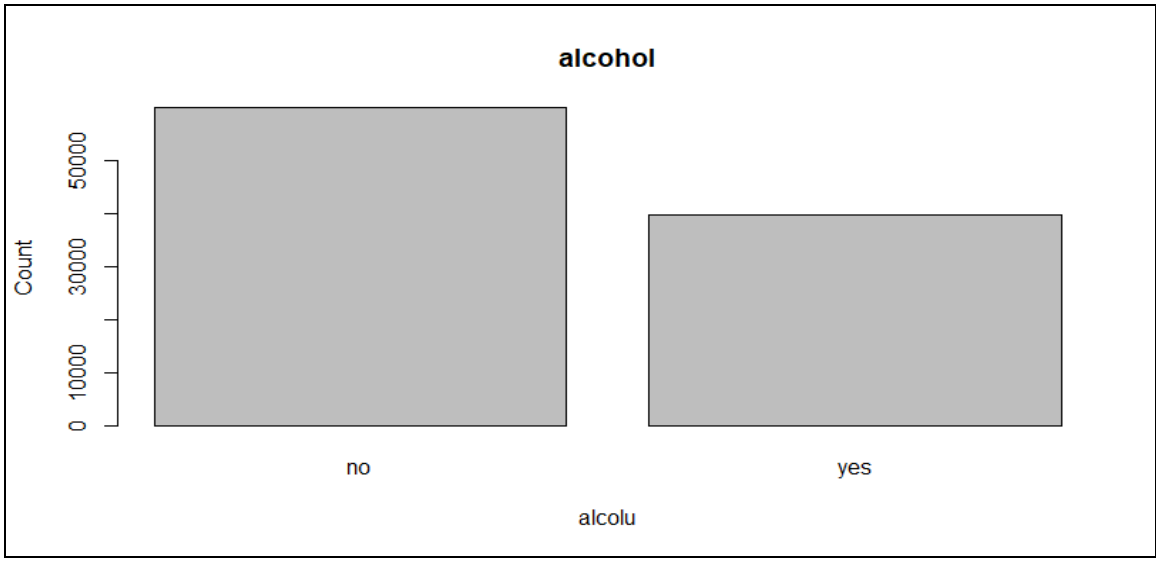


Alcohol

Code:

```
barplot(table(df$alcolu), main="alcolu", xlab=" alcohol", ylab="Count", col="blue",
border="black")
```

Output:

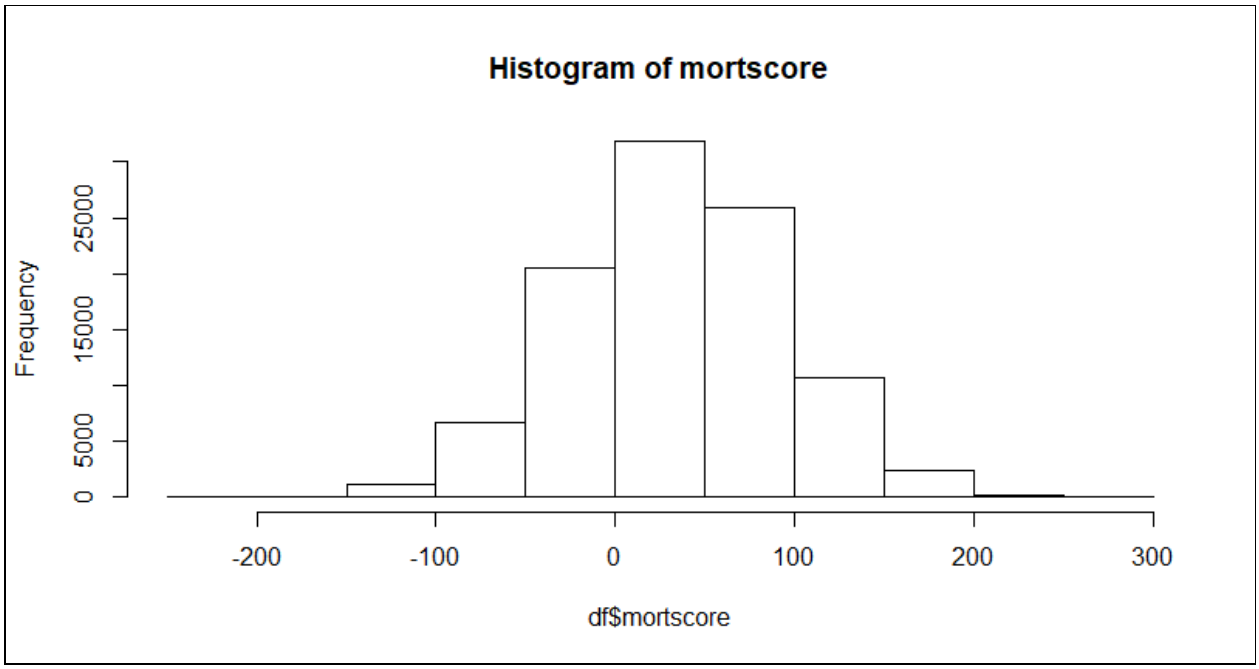


MortScore (Histogram):

Code:

```
>hist(df$mortscore, main="Histogram of mortscore")
```

Output:



D3: Predictive Modeling Explorations

Response Variable and Predictor Variables

Response variable: LOS
Predictor Variables:
Table name-Cquality: checkvaccine, checkflushot, checkdepression,
Table name-Cdemoses: Income, Age, Education
Table name-Chealth: betterwalk,bettermove,betterbreath,betterheal, Mortscore, drugu, alcolu
Table name-Cutilization: readmittedhospital, emergencyhospital, admittedhospital, urgent

Analysis Plan

After carefully examining and visualizing our data we plan to use the chi-squared test to determine a correlation between non-numeric values. We plan to use the following function to compare the non-numeric to the numeric values:

```
compare_histogram_barplot <- function(numeric, non_numeric) {
  data<-data.frame(numeric, non_numeric)
                                                                    ggplot(data,
aes(x=numeric))+geom_histogram()+facet_grid(~non_numeric)+theme_bw()
}
```

```
compare_non_numerics <- function(input_1, input_2) {
  data <- data.frame(input_1, input_2)
  data <- table(data$input_1, data$input_2)
  barplot(data, beside = TRUE, legend=rownames(data),
          args.legend = list(title=deparse(substitute(input_1))),
          xlab = deparse(substitute(input_1)))
}
```

Also, we can use additional libraries such as `PerformanceAnalytics` for correlation between numeric values and functions such as `hetcor` for finding correlations between numeric values and non-numeric values.

Our group has decided to take a two-pronged approach to the selection of our variables. The first strategy is to create a clean model which contains only variables that are statistically significant. These significant coefficients are found by removing all variables which have a p-value greater than 5%. Our second approach is to select the variables based on their importance to the model's ability to predict the variance of the dependent variable.

Predictive Modeling Exploration

Requirement

D3 will allow your team to present your initial exploratory modeling efforts on the data sets. You should now explore the relationships between predictors (e.g. correlation matrices and pairs plots). Depending on the type of your outcome variable, you should choose the appropriate modeling approach. You should experiment with the models with different number of predictors. You should document model performance on both training and test data sets by dividing your data into two halves. Include interaction terms as appropriate. Basically, apply all the techniques explained upto and including Chapter 4 of the textbook which apply to your selected modeling approach.

Rather than only a one snapshot final model, this deliverable asks you to document how you learned and what you learned from the process of statistical modeling. In your specific problem, consider what important questions can be asked and answered by building predictive models. In your specific problem, what could be potential problems in building predictive models.

Your D3 report should include all the meaningful steps you went through as a team. This document should look like the lab sections in the book. It should include the narrative component explaining what you are doing and why along with the R code snippets, textual R outputs, and plots as appropriate.

Your deliverable uploaded by the project manager should include a zip file and uploaded by only the project manager before the deliverable due date and time. Your zip file MUST include only two files: One file will be for your D3 document (pdf is preferrable); the other will be for your R script including the R statements used in the analysis. All figures and tables should be embedded in your report.

1) Introduction about the data

For this deliverable, which requires us to explore different models on the dataset we have Cdemoses, Chealth, Cquality and Cutilization, we have merged all the 4 files into a CSV and have removed ID from the data because it is certainly insignificant for modelling. In the past deliverables we searched for the defects in DQT, we cleaned it using “R” and we tried to plot all the predictors and the response variable (LOS).
Now, we are moving further with this clean data which is merged into a unified file for modelling, so that we can find the correlation between predictors and response variable.

2) Data merge process

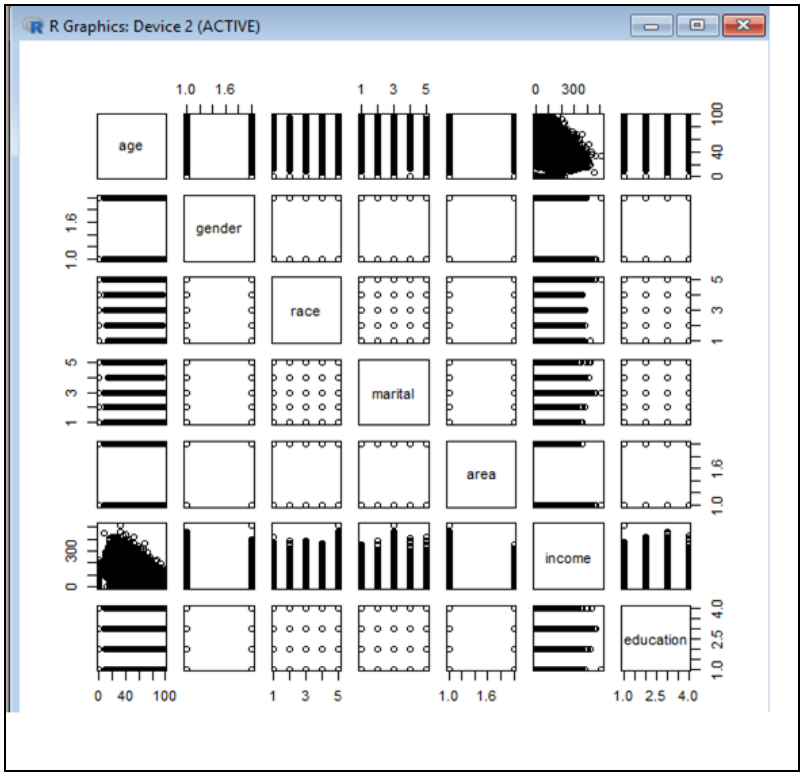
In order to merge the 4 data files in R we use the following commands which includes importing all the csv files

```
Cdemoses = read.csv(file="Cdemoses.csv", header=TRUE, sep="\t")
Chealth = read.csv(file="Chealth.csv", header=TRUE, sep="\t")
Cquality = read.csv(file="Cquality.csv", header=TRUE, sep="\t")
Cutilization = read.csv(file="Cutilization.csv", header=TRUE,
sep="\t")
```

After importing, we need to merge them using **`merge`** function in R:

```
Merged_Cdemoses_Chealth <- merge(Cdemoses,Chealth,by="id",
all.x=TRUE)
Merged_CQuality <- merge(Merged_Cdemoses_Chealth,Cquality,by="id",
all.x=TRUE)
Merged_Dataset <- merge(Merged_CQuality,Cutilization,by="id",
all.x=TRUE)
write.table(Merged_Dataset, file="Merged_Dataset.csv", sep="\t",
row.names=FALSE, quote=FALSE)
```

After merging the 4 files, we tried to generate plot of combined/merged dataset, but due to system memory limit it is not able to generate for all variables so here is plot for Cdemoses:



3) Normalization steps:

We decided to normalize our dataset so that we could fit our variables on a scale of zero to one. This allows us to more easily compare the data and perform analysis on it.

First we created the normalization function.

```
normalize <- function(x) {  return ( ( x- min(x) ) / (max(x) -
min(x)) ) }
```

Next we normalized our numeric values using the code below:

```
Normalized_Var <-  as.data.frame(lapply(Merged_Dataset[,c(2, 7, 9)],
normalize))
```

Then we output the normalized dataset to confirm that it is indeed normalized.

```
str(Normalized_Var)
```

Now we combined our two datasets.

```
Normalized_DS <- cbind(DS_Numeric, Normalized_Var)
```

We then checked the structure of this combined dataset.

```
str(Normalized_DS)
```

The KNN function required variables with some values, so we have removed all rows with the value of 'NA'.

```
Normalized_DS <- Normalized_DS[complete.cases(Normalized_DS), ]
```

Finally, we check the Normalized_DS dataset after removing the NA's.

```
summary(Normalized_DS)
```

4) Converted class variables to 0/1 and why:

After merging our dataset, we decided to convert all of our qualitative predictors. For columns whose possible values are 'yes' and 'no' we decided to replace them with 1 and 0. For our other qualitative attributes we chose to normalize the different responses by changing them to integers based on the total number of possible values. We made the decision to do this so that we could use these predictors in our model.
To achieve this with our merged dataset, we wrote an R script with examples of code below (full script will be uploaded as well):

```
# Importing merged data
dataMerged = read.delim('Merged_Dataset_NO_ID.csv')
# Delete NAs
dataMerged<-na.omit(dataMerged)

# Factoring qualitative columns
dataMerged$gender = factor(dataMerged$gender,
                           levels = c('male','female'),
                           labels = c(1,0))
dataMerged$race = factor(dataMerged$race,
                           levels = c('AFA','ASA','EUA','HLA','OTH'),
                           labels = c(4,3,2,1,0))
```

For mortscore:

In the merged dataset mortscore variable showing 99123 classes, need to convert it to numeric.

```
Merged_Dataset$mortscore = as.numeric(Merged_Dataset$mortscore)
```

In KNN, we were getting error due to factor classes, so created a function to convert it to numeric.

```
convert_to_numeric <- function(x) {  return (as.numeric(x)) }
```

Now, get the dataset that need to convert to numeric values.

```
DS_for_Numeric_Conv = subset(Merged_Dataset, select = -c(age, income,
mortscore) )
str(DS_for_Numeric_Conv)
```

Here we apply, convert to numeric function on Merged_dataset

```
DS_Numeric = as.data.frame(lapply(DS_for_Numeric_Conv,
convert_to_numeric))
```

Check structure of numeric converted dataset to make sure it is converted properly:

```
str(DS_Numeric)
```

5) linear regression, explanation of R square, f-statistics, t-statistics of model, Accuracy

Linear regression is a useful tool for predicting a quantitative response. Simple linear regression has a simple linear approach for predicting a quantitative response Y on the basis of a single regression predictor variable X.

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination for linear regression, or the coefficient of multiple determination for multiple regression. The definition of R-squared is the percentage of the response variable variation that is explained by a linear model. For a good fit of a model R-square should be higher. In the summary of a particular model it is shown as Multiple R-squared and Adjusted R-squared.

F-statistic:
An F statistic is a value you get when you run a regression analysis to find out if the means between two populations are significantly different. It's similar to a T statistic from a T-Test; A-T test will tell you if a single variable is statistically significant and an F test will tell you if a group of variables are jointly significant.
The approach of using an F-statistic to test for any association between the predictors and the response works when p is relatively small, and certainly small compared to n.

6) KNN

Because we don't know the conditional distribution of Y given X, computing the Bayes classifier. Therefore, we used the K-nearest neighbor method, which allows us to estimate the distribution of Y given X and then classify a given observation to the class with the highest estimated probability. The KNN classifier first identifies the K points in the training data that are closest to the test observation. We then estimate the conditional probability as class j, the fraction of points in the training data closest to the test observation who response values equal j. Last, KNN uses Bayes rule to classify the test observation to the class with largest probability.

Below is how we used KNN to derive knowledge from our dataset:

KNN Classification Model:

```
#Using caTools package to create training and test datasets based on
percentage.
install.packages("caTools")
library(caTools)

#seed is used to reproduce same results.
set.seed(100)

#0.75 indicate that we are using 75% data for training purpose of our
model and 25% for testing purpose.
sample = sample.split(Normalized_DS$id, SplitRatio = .75)

str(sample)
```

```
#below is training set
DS_Train = subset(Normalized_DS, sample == TRUE)
nrow(DS_Train)


#Testing dataset.
DS_Test = subset(Normalized_DS, sample == FALSE)

#check the summary of both datasets, Training set has 74833 rows
while testing has 24945 rows.
summary(DS_Train)
summary(DS_Test)

#For knn model we need training target variable, so save it in
DS_Train_Target.
DS_Train_Target = subset(Normalized_DS$LOS, sample == TRUE)
summary(DS_Train_Target)

#similarly we need test target variable.
DS_Test_Target = subset(Normalized_DS$LOS, sample == FALSE)
summary(DS_Test_Target)

#To get the square root of total observations, ideally we should set
k =  sqrt(no of observations), convert to odd number
require(class)

sqrt(nrow(Normalized_DS))

#now build knn model
KNN_ML1 <- knn(train= DS_Train, test = DS_Test, cl = DS_Train_Target,
k = 315)

#Check KNN model
KNN_ML1

#Now apply gendered model on test dataset to see how well our model
works on test dataset.
#for this we will generate confusion matrix table.
```

```
table(DS_Test_Target, KNN_ML1)
```

Limits:
Since we have too many values for classification that is causing issue, what we learned is basically if there are few classes may be 5 or 10 max then you will get cleaner confusion matrix and can conclude something from it.

KNN Regression:

K-Nearest Neighbour regression is a non-parametric supervised algorithm.This can be applied when dependent variable is continuous. In KNN regression, the predicted value is the average of the values of its k nearest neighbors.

Here, we have considered the features which we identified as important in linear regression. And then varying the K value to find the optimum K value for the model. After which the algorithm learning rate is low. We have used K=3,5,10,100 ,500 and 700.
We found that KNN regression algorithm with K value of 500, gives accuracy of 66.46%, above which the variation in the increase of accuracy is less.

KNN regression Code:

#KNN

```
library(MASS)
library(caTools)
#Setting the working directory for the R-Script
setwd("D:\\Fall18 Classes\\IS777-DataAnalytics\\GroupC")

#Reading the CSV file as dataframe
df<-read.csv('Merged_Dataset_NO_ID_Factored.csv',sep='\t')


#Splitting Train and Test data
sample = sample.split(df,SplitRatio = 0.75) # splits the data in the
ratio mentioned in SplitRatio. After splitting marks these rows as
logical TRUE and the the remaining are marked as logical FALSE
```

```
train1 =subset(df,sample ==TRUE) # creates a training dataset named
train1 with rows which are marked as TRUE
test1=subset(df, sample==FALSE)


#Fitting KNN.reg model to the data
library(FNN)
#When K=3
pred_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k = 3)
pred_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k = 3)
#Error metrics for regression
library(DMwR)
#Train
cat("Error metrics on train data for k=3")
regr.eval(train1[,"LOS"], pred_train$pred)
cat("MAPE for k=3 on train data is",round(regr.eval(train1[,"LOS"],
pred_train$pred)[4],4)*100,"%")

#Test
cat("Error metrics on test data for k=3")
regr.eval(test1[,"LOS"], pred_test$pred)
cat("MAPE for k=3 is",round(regr.eval(test1[,"LOS"],
pred_test$pred)[4],4)*100,"%")

#Accuracy of the model is 45.13%

#When k=5
pred5_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k = 5)
pred5_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k = 5)
#Error metrics for regression

#Train
cat("Error metrics on train data for k=5")
regr.eval(train1[,"LOS"], pred5_train$pred)
cat("MAPE for k=5 on train data is",round(regr.eval(train1[,"LOS"],
pred5_train$pred)[4],4)*100,"%")

#Test
```

```
cat("Error metrics on test data for k=5")
regr.eval(test1[,"LOS"], pred5_test$pred)
cat("MAPE for k=5 is",round(regr.eval(test1[,"LOS"],
pred5_test$pred)[4],4)*100,"%")
#Accuracy is 47.33 %

#Further Increasing the K value
#When k=10
pred10_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k =
10)
pred10_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k = 10)
#Error metrics for regression

#Train
cat("Error metrics on train data for k=10")
regr.eval(train1[,"LOS"], pred5_train$pred)
cat("MAPE for k=10 on train data is",round(regr.eval(train1[,"LOS"],
pred5_train$pred)[4],4)*100,"%")

#Test
cat("Error metrics on test data for k=10")
regr.eval(test1[,"LOS"], pred5_test$pred)
cat("MAPE for k=10 is",round(regr.eval(test1[,"LOS"],
pred5_test$pred)[4],4)*100,"%")

#Accuracy is 47.33%

#When k=100
pred100_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k =
100)
pred100_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k =
100)
#Error metrics for regression

#Train
cat("Error metrics on train data for k=100")
regr.eval(train1[,"LOS"], pred100_train$pred)
cat("MAPE for k=100 on train data is",round(regr.eval(train1[,"LOS"],
```

```
pred100_train$pred)[4],4)*100,"%")

#Test
cat("Error metrics on test data for k=100")
regr.eval(test1[,"LOS"], pred100_test$pred)
cat("MAPE for k=100 is",round(regr.eval(test1[,"LOS"],
pred100_test$pred)[4],4)*100,"%")
#Model accuracy now increased to  63.87 %


#When K-500:
pred500_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k =
500)
pred500_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k =
500)
#Error metrics for regression

#Train
cat("Error metrics on train data for k=500")
regr.eval(train1[,"LOS"], pred500_train$pred)
cat("MAPE for k=500 on train data is",round(regr.eval(train1[,"LOS"],
pred500_train$pred)[4],4)*100,"%")

#Test
cat("Error metrics on test data for k=100")
regr.eval(test1[,"LOS"], pred500_test$pred)
cat("MAPE for k=500 is",round(regr.eval(test1[,"LOS"],
pred500_test$pred)[4],4)*100,"%")
#Model accuracy now increased to  66.46 %

#When K-700:
pred700_train<-knn.reg(train=train1, test=train1, y=train1$LOS, k =
700)
pred700_test<-knn.reg(train=train1, test=test1, y=train1$LOS, k =
700)
#Error metrics for regression

#Train
```

```
cat("Error metrics on train data for k=700")
regr.eval(train1[,"LOS"], pred700_train$pred)
cat("MAPE for k=700 on train data is",round(regr.eval(train1[,"LOS"],
pred700_train$pred)[4],4)*100,"%")

#Test
cat("Error metrics on test data for k=100")
regr.eval(test1[,"LOS"], pred700_test$pred)
cat("MAPE for k=700 is",round(regr.eval(test1[,"LOS"],
pred700_test$pred)[4],4)*100,"%")
#Model accuracy now increased to  66.67 %

#AS it can be seen there is much increase in the accuracy. We can
stop increasing the K value and take K=500 as optimum K for KNN Reg
model
```

7) Logistic regression

Logistic Regression is a parametric regression approach used to model a binomial outcome like probability of share market going up or down. The tasks performed by the model helps us in analyzing the correct predictions for a given binomial response variable.
There should be no outliers in the data and hence for our analysis purpose we have standardized the quantitative variables (predictive variables) in the dataset. The predictive variables in our dataset are ‘age’, ‘income’, ‘mortscore’.
Since the response variable should be binary we have binned the values of responsive variables in to two categories. The response variable identified in our dataset is ‘LOS’ and after binning the column the values are “Less” and “More”.

#Code to Standardize predictor variables

To have all the quantitative variables on a comparable scale for correlation purpose, all the variables are given a mean of zero and a standard deviation of one. The function used for this purpose is ‘Scale’.

1) Select subset of the imported file only for quantitative variables

```
v2<-subset(v1, select=c("age","income","mortscore"))
```

```
head(v2)
```

```
> head(v2)
    age    income mortscore
1 22.09 174.95024     96015
2 40.96  82.23359     36806
3 81.00  35.00407      4706
4 29.16 197.86215     41436
5 46.24 115.20864     42496
6 40.96 111.87287     80594
```

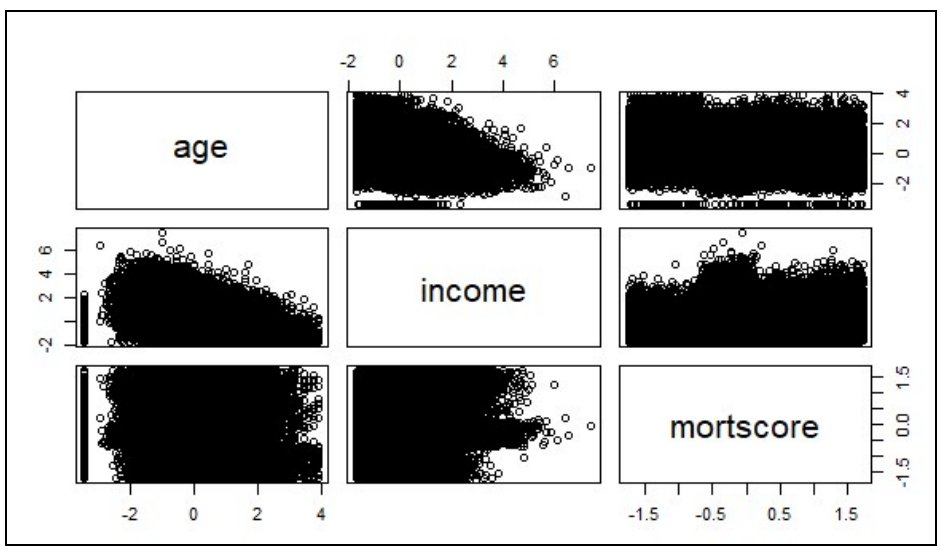
2)Standardize quantitative variables in merged file

```
standardized_data=data.frame(scale(v2))
head(standardized_data)
```

```
> head(standardized_data)
          age     income  mortscore
1 -1.7638948  1.4218281  1.6163903
2 -0.3561351 -0.2235198 -0.4487913
3  2.6309715 -1.0616537 -1.5684240
4 -1.2364512  1.8284223 -0.2872991
5  0.0377691  0.3616549 -0.2503268
6 -0.3561351  0.3024584  1.0785132
```

3) To produce correlation matrix and scatter plot for v2

```
#To produce matrix of scatter plot
pairs(v2)
```




```
#Correlation matrix
```

```
cor(v2)
```

```
> cor(v2)
                 age     income  mortscore
age        1.0000000 -0.3647689 -0.1901086
income    -0.3647689  1.0000000  0.2489634
mortscore -0.1901086  0.2489634  1.0000000
```

Explanation: The above correlation matrix between the variables helps us infer that mortscore and income have substantial correlation between them, whereas income and age have little correlation between them.

<u>8) Converting class label-'LOS' to binary, for Logistic Regression Purpose:</u>

```
#Cut function is used to bin the class label into binary values viz.
'Less' and 'more'
cut(v4,2, labels=c("Less","More"))
#Replace existing column in v1 with the binary values
v1$LOS<-cut(v4,2, labels=c("Less","More"))
#Selecting variables for logistic regression
v6<-subset(v1,select=c("age","income","mortscore","LOS"))
head(v6)
```

```
> head(v6)
          age     income  mortscore  LOS
1 -1.7638948  1.4218281  1.6163903 Less
2 -0.3561351 -0.2235198 -0.4487913 Less
3  2.6309715 -1.0616537 -1.5684240 More
4 -1.2364512  1.8284223 -0.2872991 Less
5  0.0377691  0.3616549 -0.2503268 Less
6 -0.3561351  0.3024584  1.0785132 More
```

Explanation: The LOS column mentioned above is binned to replace existing numeric values with binary variables viz. 'Less' or 'More'.

5) Fit Logistic Regression model in order to predict response variable LOS

```
glm.fits=glm(LOS~income+mortscore+age,data=v6,family=binomial)
summary(glm.fits)
coef(glm.fits)
```

```
> summary(glm.fits)

Call:
glm(formula = LOS ~ income + mortscore + age, family = binomial,
    data = v6)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9792  -0.5883  -0.4130  -0.2541   3.2593

Coefficients:
             Estimate Std. Error  z value Pr(>|z|)
(Intercept) -2.050022   0.011393 -179.944  < 2e-16 ***
income      -0.496213   0.012786  -38.809  < 2e-16 ***
mortscore    0.038600   0.009718    3.972 7.12e-05 ***
age          0.792817   0.010288   77.059  < 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: 85720  on 99776  degrees of freedom
Residual deviance: 73722  on 99773  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 73730

Number of Fisher Scoring iterations: 5
```

Explanation: The negative coefficients for income suggests that income and LOS have little association between them.

```
> coef(glm.fits)
(Intercept)      income   mortscore         age
-2.05002173 -0.49621330  0.03860004  0.79281659
```

Explanation: To access just the coefficients for the fitted model, we use above code. It is evident from above summary that age is highly associated with LOS.

6) Predict function used to predict LOS,be less or more

```
glm.probs =predict (glm.fits,type ="response")
#Printing first 10 probabilities
glm.probs[1:10]
#Creates dummy variables
contrasts(v6$LOS)
glm.pred=rep("Less",99778)
glm.pred[glm.probs>.5]="More"
#Confusion matrix for LOS
```

```
table(glm.pred,v6$LOS)
```

(83162+1671)/99778

```
> glm.probs =predict (glm.fits,type ="response")
> #Printing first 10 probabilities
> glm.probs[1:10]
         1          2          3          4          5          6          7          8

0.01643769 0.09632156 0.62295778 0.01891548 0.09892751 0.08011211 0.05219801 0.02729520

         9         10
0.22030578 0.15813368
> #Creates dummy variables
> contrasts(v6$LOS)
     More
Less    0
More    1
> glm.pred=rep("Less",99778)
> glm.pred[glm.probs>.5]="More"
> #Confusion matrix for LOS
> table(glm.pred,v6$LOS)

glm.pred  Less  More
    Less 83162 13693
    More  1251  1671
> (83162+1671)/99778
[1] 0.8502175
```

Explanation: The predict function will help us accurately predict the training data for LOS variable. The type ”response” in glm.probs variable helps us define the output of probabilities to be conditional rather than logit.

The probabilities for first 10 observation is predicted, and we can conclude that these values correspond to the probability of LOS being ‘More’ as R has created dummy variable with 1 for ‘More’.

The table function is used to produce confusion matrix to determine how many observations were correctly or incorrectly classified.

As seen from the above snapshot, the true positives and true negatives obtained are 83162 and 1671 respectively out of total of 99778 observations. The mean function helps us understand the accuracy of logistic regression model to correctly predict the ‘LOS’ variable. The accuracy of the model is 85% which is a great fit for the given subset of variables.

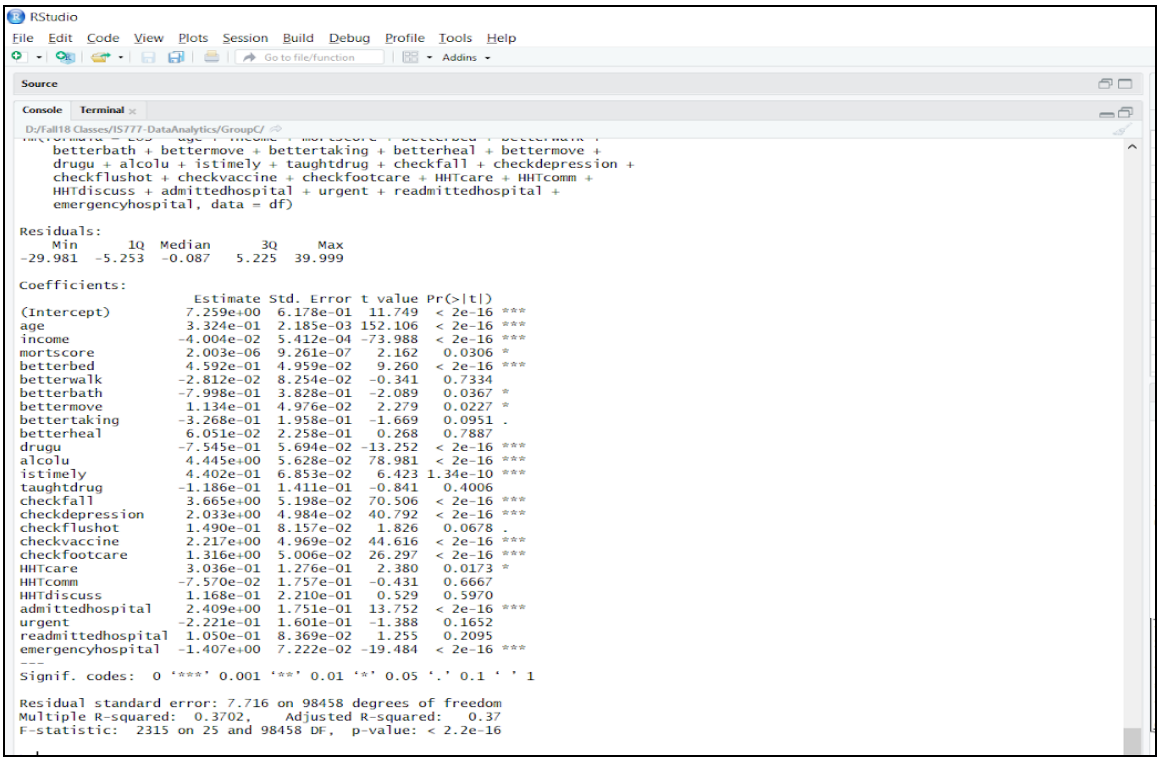
9) <u>Multiple Linear Regression</u>

The aim of linear regression is to model a continuous variable *Y* as a mathematical function of one or more *X* variable(s), so that we can use this regression model to predict the *Y* when only the *X* is known. This mathematical equation can be generalized as follows:

$Y = \beta_1 + \beta_2 X + \epsilon$

where, β_1 is the intercept and β_2 is the slope. Collectively, they are called *regression coefficients*. ϵ is the error term, the part of Y the regression model is unable to explain.

For our data set LOS is the response variable which is continuous. So, considering linear regression model and applying whole data to the lm model in R gives the Estimate, Std. Error, t value, Pr(>|t|). Analysing the summary of the model gives the important or highly correlated features with the response variable LOS.
We are using "Backward Elimination/ Selection Process" technique to identify the important features.



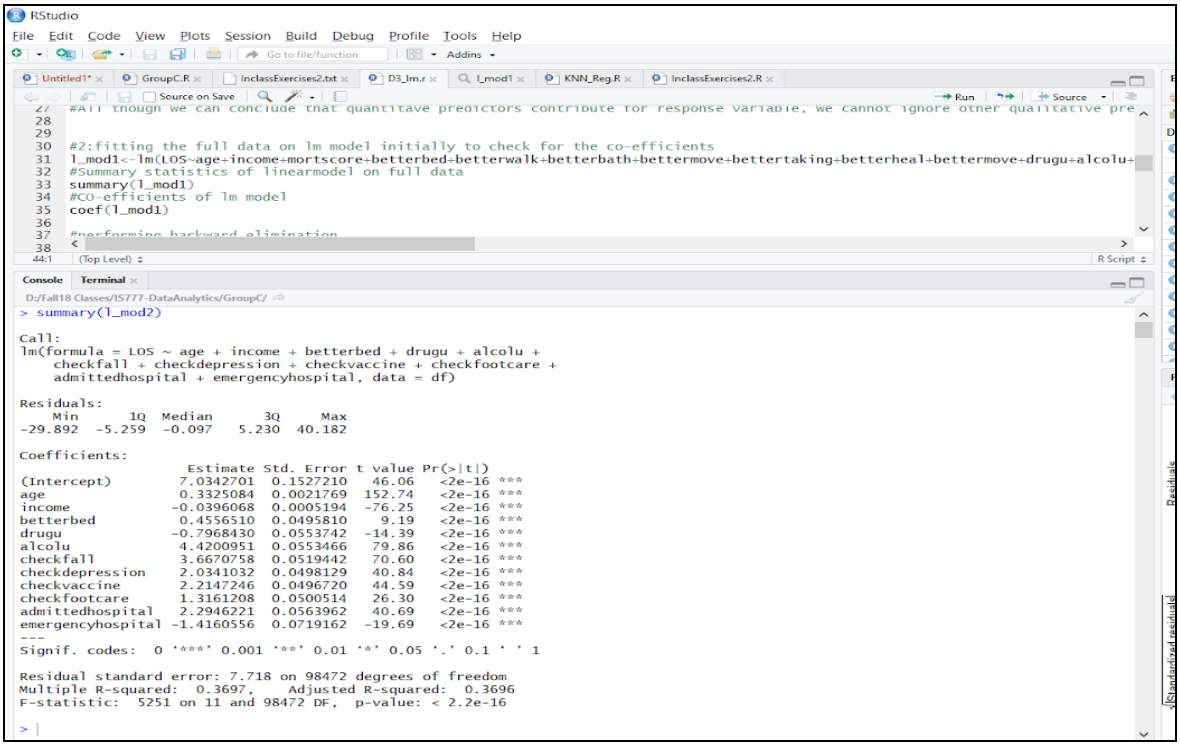
```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal
D:/Fall18 Classes/IS777-DataAnalytics/GroupC/
    betterbath + bettermove + bettertaking + betterheal + bettermove +
    drugu + alcolu + istimely + taughtdrug + checkfall + checkdepression +
    checkflushot + checkvaccine + checkfootcare + HHTcare + HHTcomm +
    HHTdiscuss + admittedhospital + urgent + readmittedhospital +
    emergencyhospital, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-29.981  -5.253  -0.087   5.225  39.999

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.259e+00  6.178e-01  11.749  < 2e-16 ***
age                 3.324e-01  2.185e-03 152.106  < 2e-16 ***
income             -4.004e-02  5.412e-04 -73.988  < 2e-16 ***
mortscore           2.003e-06  9.261e-07   2.162   0.0306 *
betterbed           4.592e-01  4.959e-02   9.260  < 2e-16 ***
betterwalk         -2.812e-02  8.254e-02  -0.341   0.7334
betterbath         -7.998e-01  3.828e-01  -2.089   0.0367 *
bettermove          1.134e-01  4.976e-02   2.279   0.0227 *
bettertaking       -3.268e-01  1.958e-01  -1.669   0.0951 .
betterheal          6.051e-02  2.258e-01   0.268   0.7887
drugu              -7.545e-01  5.694e-02 -13.252  < 2e-16 ***
alcolu              4.445e+00  5.628e-02  78.981  < 2e-16 ***
istimely            4.402e-01  6.853e-02   6.423 1.34e-10 ***
taughtdrug         -1.186e-01  1.411e-01  -0.841   0.4006
checkfall           3.665e+00  5.198e-02  70.506  < 2e-16 ***
checkdepression     2.033e+00  4.984e-02  40.792  < 2e-16 ***
checkflushot        1.490e-01  8.157e-02   1.826   0.0678 .
checkvaccine        2.217e+00  4.969e-02  44.616  < 2e-16 ***
checkfootcare       1.316e+00  5.006e-02  26.297  < 2e-16 ***
HHTcare             3.036e-01  1.276e-01   2.380   0.0173 *
HHTcomm            -7.570e-02  1.757e-01  -0.431   0.6667
HHTdiscuss          1.168e-01  2.210e-01   0.529   0.5970
admittedhospital    2.409e+00  1.751e-01  13.752  < 2e-16 ***
urgent             -2.221e-01  1.601e-01  -1.388   0.1652
readmittedhospital  1.050e-01  8.369e-02   1.255   0.2095
emergencyhospital  -1.407e+00  7.222e-02 -19.484  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.716 on 98458 degrees of freedom
Multiple R-squared:  0.3702,    Adjusted R-squared:   0.37
F-statistic:  2315 on 25 and 98458 DF,  p-value: < 2.2e-16
```

The output of the Linear regression model gives the estimates and probabilities (p-value which is usually less than 0.05).The important features are also marked with *.By looking at the summary, we can see from the above figure that, the features age,income,betterbed,drugu,alcolu,checkfall,checkdepression,checkvaccine, Checkfootcare,admittedhospital,emergencyhospital, have the high absolute values and also low p-value. The RSE in the model is 7.716 , which is relatively high.Multiple and adjusted R-squared is relatively very low-0.37.

Now, applying only the identified important features to linear model.

```
l_mod3<-lm(LOS~age+income+betterbed+drugu+alcolu+checkfall+checkdepre
```

```
ssion+checkvaccine+checkfootcare+admittedhospital+emergencyhospital,d
ata=train1)
```



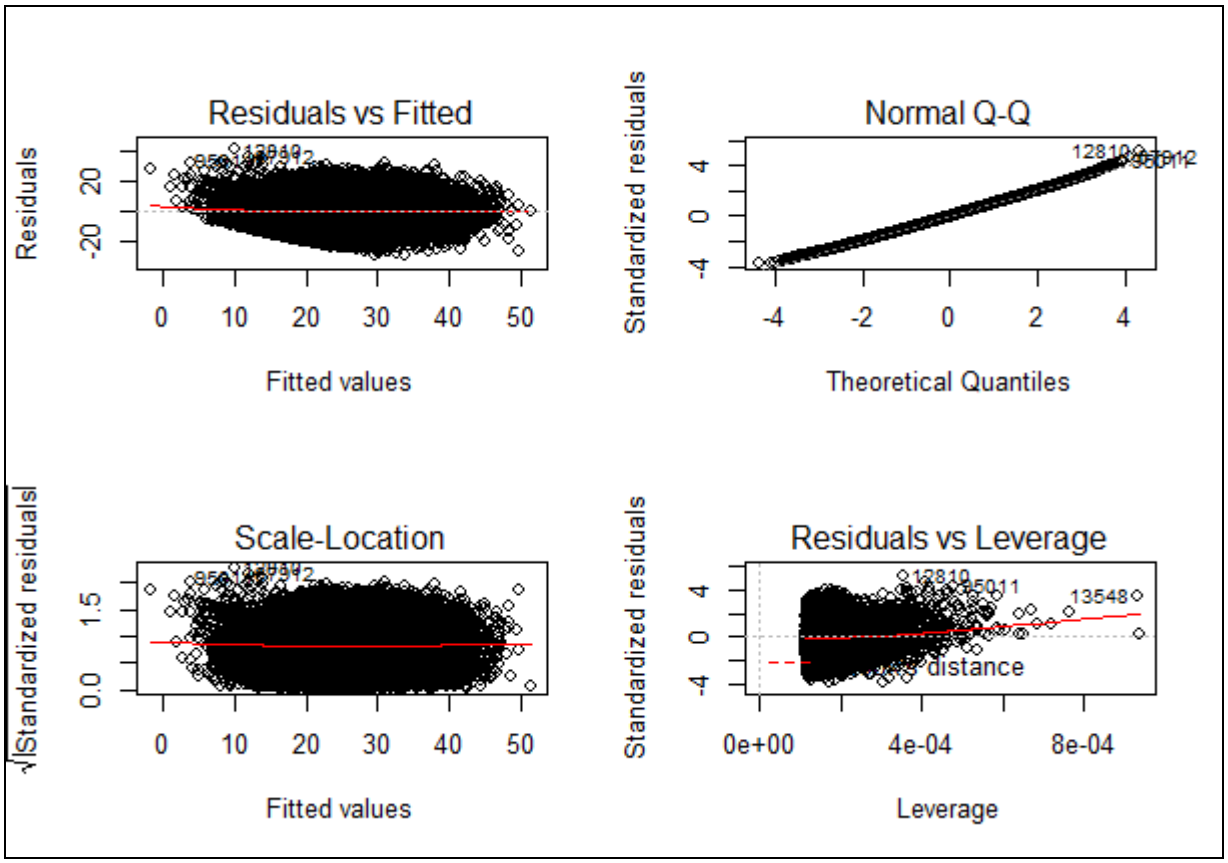
Model Accuracy

Fitting the model for test data to calculate the accuracy and performance of the model.We divided the data into train and test sample with the ratio of 75:25.

MinMax tells you how far the model's prediction is off. For a perfect model, this measure is 1.0.The lower the measure, the worse the model, based on out-of-sample performance.The linear model accuracy is 77.9324.The accuracy is pretty good, the linear regression model performance on test is considerably better.

Adjusted and multiple R-squared value is still remains, but one cannot just consider R-squared value to access the model. It is significant from the accuracy that, linear model is performing well on test data.

Plots of Linear regression Model:



Code for Linear Regression Model:

```
library(MASS)
library(caTools)
#Setting the working directory for the R-Script
setwd("D:\\Fall18 Classes\\IS777-DataAnalytics\\GroupC")

#Reading the CSV file as dataframe
df<-read.csv('Merged_Dataset_NO_ID_Factored.csv',sep='\t')

#to check for some sample of data
head(df)
#Summary statistics of each object in df
summary(df)
```

```
#if there are null values in the column , displays NA
max(df)

#Exploratory Analysis for feature selection

#1:Checking only for qualitative predictors to response variable LOS.
Assuming that quantitative data has high correlation with LOS
df1<-subset(df,select=c("age","income","mortscore","LOS"))
str(df1)
df1$mortscore<-as.numeric(df1$mortscore)
linearMod <- lm(LOS~age+income+mortscore, data=df1)
summary(linearMod)
#All though we can conclude that quantitative predictors contribute
for response variable, we cannot ignore other qualitative predictors

#2:fitting the full data on lm model initially to check for the
coefficients
l_mod1<-lm(LOS~age+income+mortscore+betterbed+betterwalk+betterbath+b
ettermove+bettertaking+betterheal+bettermove+drugu+alcolu+istimely+ta
ughtdrug+checkfall+checkdepression+checkflushot+checkvaccine+checkfoo
tcare+HHTcare+HHTcomm+HHTdiscuss+admittedhospital+urgent+readmittedho
spital+emergencyhospital,data=df)
#Summary statistics of linear model on full data
summary(l_mod1)
#CO-efficients of lm model
coef(l_mod1)

#performing backward elimination
#As we can see there are few predictors which have low p-Value ,hence
we can reject the null hypothesis for those predictors
#--and consider those predictors for further analysis
l_mod2<-lm(LOS~age+income+betterbed+drugu+alcolu+checkfall+checkdepre
ssion+checkvaccine+checkfootcare+admittedhospital+emergencyhospital,d
ata=df)

summary(l_mod2)
```

```
#Splitting Train and Test data
sample = sample.split(df,SplitRatio = 0.75) # splits the data in the
ratio mentioned in SplitRatio. After splitting marks these rows as
logical TRUE and the the remaining are marked as logical FALSE
train1 =subset(df,sample ==TRUE) # creates a training dataset named
train1 with rows which are marked as TRUE
test1=subset(df, sample==FALSE)

#Fitting model on train data
l_mod3<-lm(LOS~age+income+betterbed+drugu+alcolu+checkfall+checkdepre
ssion+checkvaccine+checkfootcare+admittedhospital+emergencyhospital,d
ata=train1)

summary(l_mod3)
plot(l_mod3)
return()
par(mfrow = c(2, 2))
plot(l_mod3)

#predicting the LOS for test data
Pred <- predict(l_mod3, test1)

#showing the actual values with predicted values of LOS
actuals_preds<-data.frame(cbind(actuals=test1$LOS, predicteds=Pred))

#calculating the correlation between actual and pred values.
correlation_accuracy3<- cor(actuals_preds)

#MinMax tells you how far the model's prediction is off. For a
perfect model, this measure is 1.0.
#The lower the measure, the worse the model, based on out-of-sample
performance.
min_max_accuracya3 <- mean(apply(actuals_preds, 1, min) /
apply(actuals_preds, 1, max))

#calculating mean of
mapev <- mean(abs((actuals_preds$predicteds -
actuals_preds$actuals))/actuals_preds$actuals)
```

10) Findings from model:

No model is considered best model in statistics. Models are best analysed on the data provided. Hence for regression problem for our data set , we used Multiple linear regression and KNN regression model to check. For the features which were selected from the data, linear regression model show better performance in accuracy than the KNN regression.

For the classification problem, to identify the class for the LOS , logistic regression classifier predicts well with the accuracy of 85% compared to KNN classifier.

D4: Resampling

Model assessment

A. The entire data set as the training data.

In this approach the data is simply divided into train and test. Like other sampling approach, there is no validation set. So, the data is divided into train and set in the ratio of [66.66:33.33]. First, we train the regression model with training dataset and the model is assessed on the test data. The flexibility of the model is increased by increasing the model degree from 1 to 10.All the mean squared error for each degree is calculated and plotted against the degree of polynomial.

Steps for predictor variable Age

1.Setting the working directory and reading the dataset from local directory

```
library(ISLR)
library(MASS)
#Setting the working directory for the R-Script
setwd("D:\\Fall18 Classes\\IS777-DataAnalytics\\GroupC")

#Reading the CSV file as dataframe
v1<-read.csv('Merged_Dataset_NO_ID_Factored.csv',sep='\t')
```

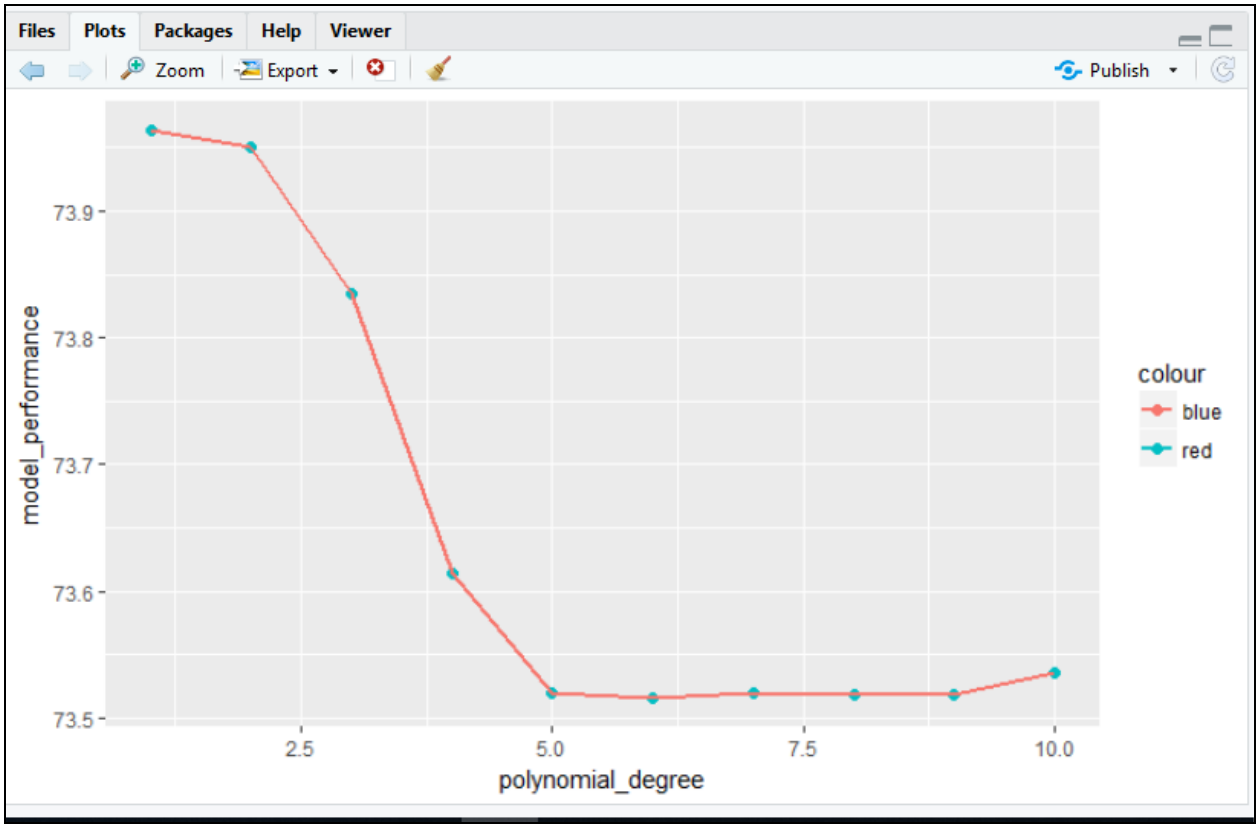
2.Now setting the seed and dividing the data to train and test.
Running the for loop to fit the “lm” model for each polynomial degree and calculating the MSE

```
set.seed(1)
train<-sample(dim(v1)[1],dim(v1)[1]/3)
lm.fit1=numeric()
mse=rep(0 ,10)
for (i in 1:10) {
  lm.fit1=lm(LOS~poly(age,i),data=v1,subset=train)
  mse[i]<-mean((v1$LOS-predict(lm.fit1,v1))[-train]^2)
  }
mse
```

```
> set.seed(1)
> train<-sample(dim(v1)[1],dim(v1)[1]/3)
> lm.fit1=numeric()
> mse=rep(0 ,10)
> for (i in 1:10) {
+   lm.fit1=lm(LOS~poly(age,i),data=v1,subset=train)
+   mse[i]<-mean((v1$LOS-predict(lm.fit1,v1))[-train]^2)
+   }
> mse
 [1] 73.96406 73.95087 73.83512 73.61475 73.52046 73.51546 73.52011 73.51919 73.51924 73.53638
>
```

3.Visualising the results by plotting the graph of MSE against Polynomial Degree.

```
#visualize MSE
model_performance<-mse
polynomial_degree<-c(1:10)
mse_p <- data.frame(polynomial_degree, model_performance)

ggplot()+
  geom_point(data=mse_p, aes(x=polynomial_degree, y=model_performance,color='red'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree, y=model_performance,color='blue'), size=1)
```



Steps for predictor variable Income

1.Setting the working directory and reading the dataset from local directory

```
library(ISLR)
library(MASS)
#Setting the working directory for the R-Script
setwd("D:\\Fall18 Classes\\IS777-DataAnalytics\\GroupC")

#Reading the CSV file as dataframe
v1<-read.csv('Merged_Dataset_NO_ID_Factored.csv',sep='\t')
```

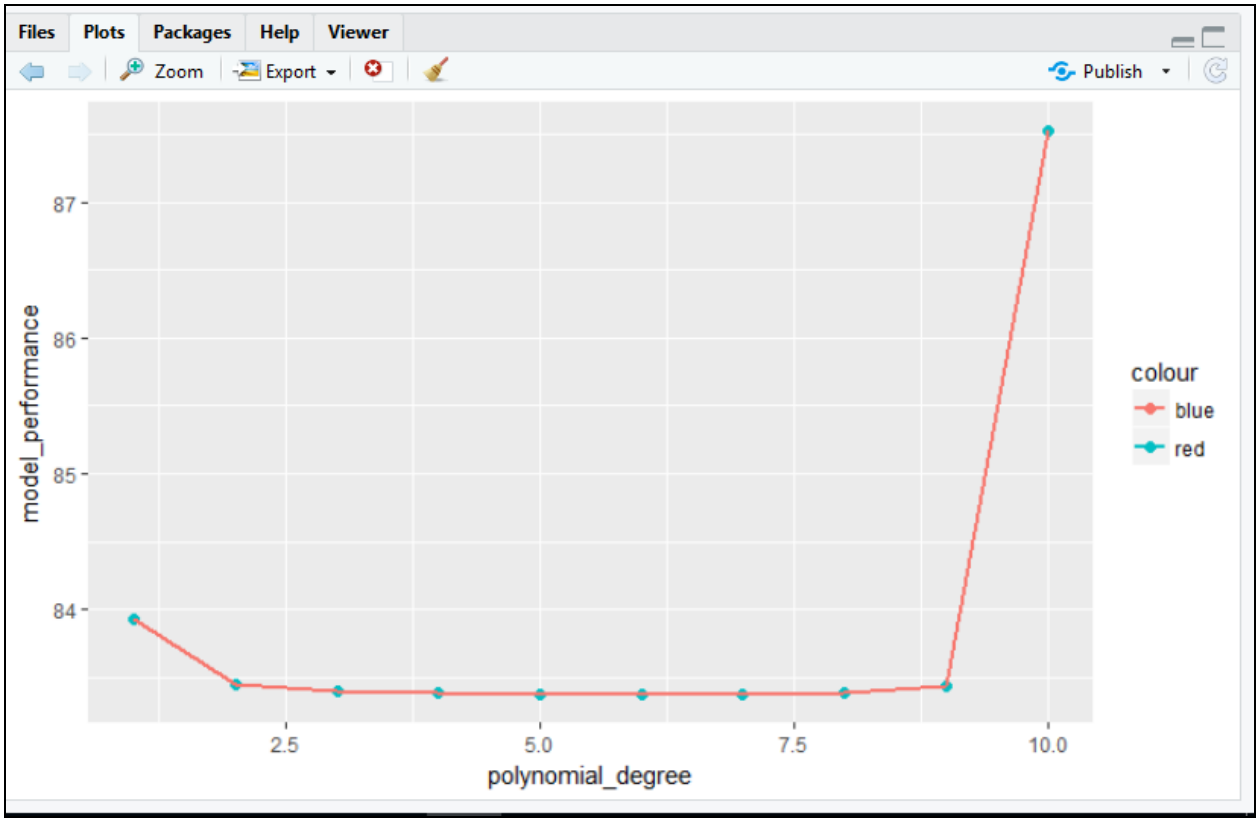
2.Now setting the seed and dividing the data to train and test.
Running the for loop to fit the “lm” model for each polynomial degree and calculating the MSE

```
set.seed(1)
train<-sample(dim(v1)[1],dim(v1)[1]/3)
lm.fit1=numeric()
mse=rep(0 ,10)
for (i in 1:10) {
  lm.fit1=lm(LOS~poly(income,i),data=v1,subset=train)
  mse[i]<-mean((v1$LOS-predict(lm.fit1,v1))[-train]^2)
}
mse
```

```
set.seed(1)
train<-sample(dim(v1)[1],dim(v1)[1]/3)
lm.fit1=numeric()
mse=rep(0 ,10)
for (i in 1:10) {
  lm.fit1=lm(LOS~poly(income,i),data=v1,subset=train)
  mse[i]<-mean((v1$LOS-predict(lm.fit1,v1))[-train]^2)
}
mse
[1] 83.93088 83.45325 83.39765 83.38096 83.37195 83.36833 83.36836 83.38355 83.43425 87.52943
```

3.Visualising the results by plotting the graph of MSE against Polynomial Degree.

```
#visualize MSE
model_performance<-mse
polynomial_degree<-c(1:10)
mse_p <- data.frame(polynomial_degree, model_performance)

ggplot()+
  geom_point(data=mse_p, aes(x=polynomial_degree, y=model_performance,color='red'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree, y=model_performance,color='blue'), size=1)
```



B. leave-one-out cross validation

Leave-one-out cross-validation puts the model repeated n times, suppose the dataset have 1000 records then model will consider 999 records (1000 – 1) as training set and validate result against leave out 1 record, Process will repeat itself n times so it get train on each record and get validated against all records. delta is the cross-validated prediction error where: The first number is the raw leave-one-out, or lieu cross-validation result. The second one is a bias-corrected version of it

Steps

1) Set working directory for R environment using setwd() command, it take path where file are placed.

```
setwd("C:\\Users\\shailendra.singh.kus\\Desktop\\UMBC
-MS\\IS-777\\Deliverables\\D4\\");
```

2) Verify that working directory is set properly using getwd() function, getwd() show the location of current working directory.

```
getwd();
```

3) Load the data from CSV file to in memory variable (named Cleaned), we used the cleaned and merged dataset from previous deliverables.

```
Clean_DF = read.csv(file="Merged_Dataset_NO_ID_Factored.csv",
header=TRUE, sep="\t");
```

CSV file data is separated by tab, sep parameter instruct R to consider new filed at tab, Since file as column header at first row so header parameter is set to TRUE, so R read first row as field name.

4) Check structure of loaded data, using summary and str function of R. Here is screen shot of summary and structure results.

```
str(Clean_DF)
```

```
> str(Clean_DF);
'data.frame':   98484 obs. of  33 variables:
 $ age               : num  22.1 41 81 29.2 46.2 ...
 $ gender            : int  1 1 1 0 0 0 0 0 0 1 ...
 $ race              : int  0 0 0 0 2 0 0 3 3 0 ...
```

```
summary(Clean_DF)
```

```
> summary(Clean_DF);
      age             gender             race            marital
 Min.   : 0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
 1st Qu.:36.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
 Median :44.89   Median :0.0000   Median :0.0000   Median :0.000
 Mean   :45.72   Mean   :0.4227   Mean   :0.7426   Mean   :0.395
 3rd Qu.:54.76   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.000
 Max.   :98.01   Max.   :1.0000   Max.   :4.0000   Max.   :4.000
```

summary function gives you closer look at data summary, provide detail as Min, Max, Min, Median, 1st Quartile and 3rd Quartile information.

5) As we need quantitative variables so we subset function of R to select specific variables from dataframe (Clean_DF) we selected "age","income","mortscore" and "LOS" for analysis purpose. subset( ) function in R is the basically the easiest way to select variables and observations. subsetted data values are saved in Quantitative data frame (Quant_DF).

```
Quant_DF <-
subset(Clean_DF,select=c("age","income","mortscore","LOS"));
```

6) Verify the new data frame using str() function.

7) Using as.numeric function converted mortscore to numeric values.

```
Quant_DF$mortscore<-as.numeric(Quant_DF$mortscore);
```

8) We want to reproduce the same results irrespective number to time script ran, so we use seed and set it to 100 (we can choose any number), in short seed is used to reproduce same results.

```
set.seed(100)
```

9) Since LOOCV in very resource intensive process, we were getting issue on laptop, It basically process the n-1 records (leaving 1 record for validation) n times, even trying multiple time process running for forever so try to split data and run with smaller dataset, split dataset using sample.split() function.

It split data into two sets in predefined ratio, used to split the data used during classification into train and test subsets

For this we need to install and load caTools library.
Below is the code to install libraries and execute sample.split function. We split data 75% and 25 %

```
install.packages("caTools");
library(caTools);

sample = sample.split(Quant_DF, SplitRatio = .75)
```

10) Now create training set and testing set using above created sample vector, which contain true false values.

```
#Training set, named as DS_Train
DS_Train = subset(Quant_DF, sample == TRUE);
nrow(DS_Train);

#Testing dataset named DS_Test
DS_Test = subset(Quant_DF, sample == FALSE);
```

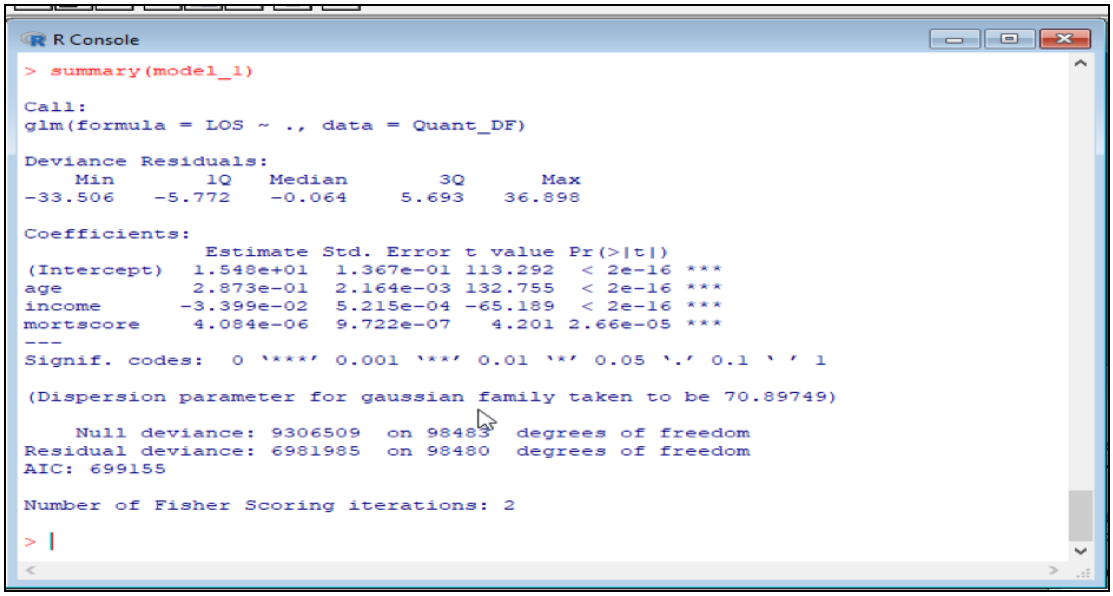
11) As per requirement 1.a, use entire data set as the training data, so train our model using full dataset.

glm() is used to fit generalized linear models without family="binomial" argument it perform as logistic regression.

```
#Generate model using full dataset.   "." Indicate generate model
```

```
using all variables of dataset.

model_1 <- glm(LOS~.,data=Quant_DF);
```

Check summary of model.



```
R Console
> summary(model_1)

Call:
glm(formula = LOS ~ ., data = Quant_DF)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-33.506   -5.772   -0.064    5.693   36.898

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.548e+01  1.367e-01 113.292  < 2e-16 ***
age          2.873e-01  2.164e-03 132.755  < 2e-16 ***
income      -3.399e-02  5.215e-04 -65.189  < 2e-16 ***
mortscore    4.084e-06  9.722e-07   4.201 2.66e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 70.89749)

    Null deviance: 9306509  on 98483  degrees of freedom
Residual deviance: 6981985  on 98480  degrees of freedom
AIC: 699155

Number of Fisher Scoring iterations: 2

> |
```

12) Apply generated trained model (generated using full dataset) on test dataset to see how it is performing. i.e. predicting the LOS on test dataset.

13) Create dataframe of actual values and predicted values, using below code, cbind function bind the provided columns, required the row numbers of the two datasets must be equal.

```
actuals_preds <- data.frame(cbind(actuals=DS_Test$LOS,
predicteds=Pred))
```

14) See actual prediction by inspecting actuals_preds variable

```
Actuals_preds
```

15) Min Max tells you how far the model's prediction is off. For a perfect model, this measure is 1.0.
The lower the measure, the worse the model, based on out-of-sample performance.

```
min_max_accuracy <- mean(apply(actuals_preds, 1, min) /
apply(actuals_preds, 1, max))

#check min_max_accuracy
```

```
min_max_accuracy
```

```
> #check min_max_accuracy
> min_max_accuracy
[1] 0.7670553
```

16) Calculate mean of error and check it

```
mapev<-mean(abs((actuals_preds$predicteds
actuals_preds$actuals))/actuals_preds$actuals)
```

check calculated mean of error
Mapev

```
> #check mapev
> mapev
[1] 0.5730876
```

17) For LOOCV we need boot library, install and load it, choose nearest cran mirror
Basically used for bootstrapping purpose.

```
install.packages("boot");
library (boot)
```

18) cv.glm() is used to apply LOOCV approach, we used model_1 generated above using full dataset.

cv.glm() function calculates the estimated K-fold cross-validation prediction error for generalized linear models. For k-fold cross validation k need to be set as number of fold you want to use. If k is not provided the default is set to the number of observations in data which gives the usual leave-one-out cross-validation.

Running LOOCV with full dataset (which is having 98484 rows) is taking forever, I ran cv.glm on full dataset it ran for 5+ hours but did not get result.
Since LOOCV consider n-1 records as training dataset and 1 record as validation and repeat the process n times, it is very resource intensive (need high configuration systems/server) and not able to run on my machine. So we decided to consider 25% data of full dataset for LOOCV approach.

Below is commented code we tried.

```
cv.err =cv.glm(Quant_DF,model_1);
```

So we consider created new dataframe LOOCV_DF which is copy of DS_Test (sampled at 25%), below is the code to create copy of existing dataset.

```
LOOCV_DF <- DS_Test
```

Now apply cv.glm on reduced dataset (LOOCV_DF)

```
cv.err =cv.glm(LOOCV_DF,model_1);
```

delta vector of cv.err contain cross validation result.

```
cv.err$delta;
```

we got values 118.44505 71.19881

After that we used entire dataset and got 70.90045 70.90045

19. Now as per requirement apply polynomial degree of 1:10, to apply polynomial degree of 1:10 we use for loop and iterate it 1 to 10 times, computes the associated cross-validation error, and stores it in the ith element of the vector cv.error.

```
cv.error=rep (0,10)
#glm.fit= numeric()
for (i in 1:10){
 glm.fit=glm(LOS~poly(age,i) ,data=LOOCV_DF)
 cv.error[i]=cv.glm(LOOCV_DF,glm.fit)$delta [1]
}

cv.error
```

below is the screen shot of cv.error generated by polynomial of 1:10

```
> #delta vector of cv.err contain cross validation result.
> cv.err$delta;
[1] 118.44505  71.19881
>
> #apply polynomial degree of 1:10, to apply polynomial degree of 1:10 we use f$
> #it 1 to 10 times,computes the associated cross-validation error, and stores $
> #the ith element of the vector cv.error.
>
> cv.error=rep (0,10)
> #glm.fit= numeric()
> for (i in 1:10){
+  glm.fit=glm(LOS~poly(age,i) ,data=LOOCV_DF)
+  cv.error[i]=cv.glm(LOOCV_DF,glm.fit)$delta [1]
+ }
>
> cv.error
 [1] 73.51256 73.52053 73.27052 73.03669 72.94438 72.94660 72.94584 72.94096
 [9] 72.94647 72.94643
```

20. We even tried to apply polynomial degree of 1:10 combining all variables but due to system limitation, we are not getting results, below is the commented code.

```
cv.error=rep (0,10)
glm.fit= numeric()
for (i in 1:10){
 glm.fit=glm(LOS~poly(age,i) + poly(income,i) + poly(mortscore,i)
,data=LOOCV_DF)
 cv.error[i]=cv.glm(LOOCV_DF,glm.fit)$delta [1]
}
```

C. Validation set approach

Cross validation is a resampling method that fits the model on certain observations from dataset, called as training dataset and finds out test error results using MSE on validation set. Validation set doesn't include the observations considered in training dataset.

In validation set Approach we divide the data in half and half. By selecting random set of observations we divide the data into training dataset (50%) and validation dataset (50%)

For the given dataset after applying fitting linear regression model on following variables the result obtained are as follows:

The dataset used for fitting the model is merged (as reported in D3).

LOS is the response variable. The predictors considered for the response variable are mortscore and age respectively.

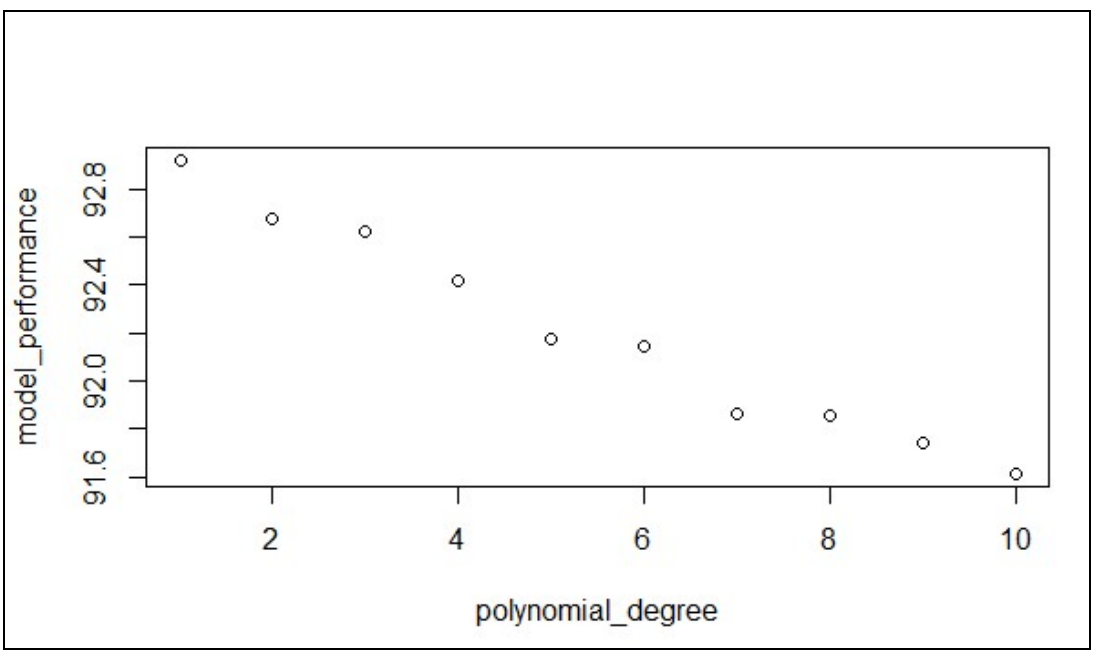
```
> set.seed(1)
> #random sample selection to split training and validation set
> train<-sample(dim(v1)[1],dim(v1)[1]/2)
> class(train)
[1] "integer"
> #Linear Regression Function to select observations in training s
e
> lm.fit=lm(LOS~mortscore,data=v1,subset=train)
> coef(lm.fit)
(Intercept)   mortscore
   25.66535    -1.09688
```

In above image linear regression is fit on the model LOS~mortscore , and the coefficient summary is mentioned. Negative coefficient indicates that for decrease in LOS stay , mortscore would decrease.

Also, the subset of the data used is training data set containing half of the observations.

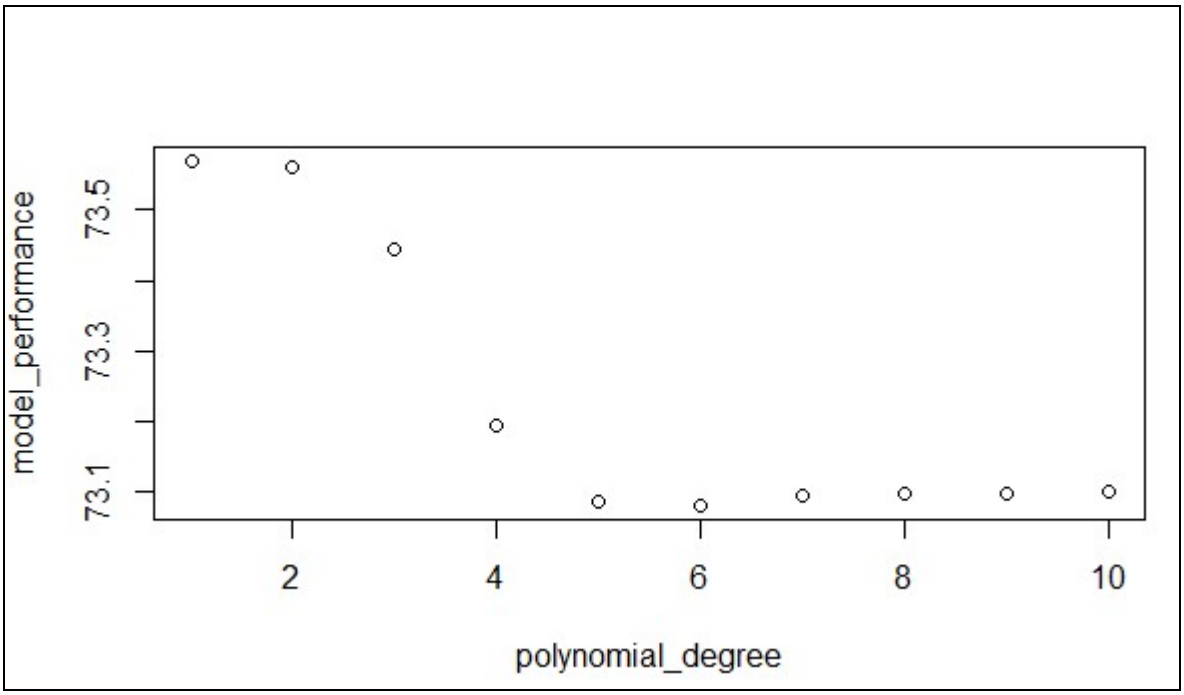
Then we calculate the MSE for observations in validation set by increasing the flexibility of models.

```
> #mean to calculate the MSE for observations in validation set
> r1<-mean((v1$LOS-predict(lm.fit,v1))[-train]^2)
> #use of poly function to estimate test error for quadratic and cubic regression
> lm.fit2=lm(LOS~poly(mortscore,2),data=v1,subset=train )
> r2<-mean((v1$LOS-predict(lm.fit2,v1))[-train]^2)
> lm.fit3=lm(LOS~poly(mortscore,3),data=v1,subset=train )
> r3<-mean((v1$LOS-predict(lm.fit3,v1))[-train]^2)
> lm.fit4=lm(LOS~poly(mortscore,4),data=v1,subset=train )
> r4<-mean((v1$LOS-predict(lm.fit4,v1))[-train]^2)
> lm.fit5=lm(LOS~poly(mortscore,5),data=v1,subset=train )
> r5<-mean((v1$LOS-predict(lm.fit5,v1))[-train]^2)
> lm.fit6=lm(LOS~poly(mortscore,6),data=v1,subset=train )
> r6<-mean((v1$LOS-predict(lm.fit6,v1))[-train]^2)
> lm.fit7=lm(LOS~poly(mortscore,7),data=v1,subset=train )
> r7<-mean((v1$LOS-predict(lm.fit7,v1))[-train]^2)
> lm.fit8=lm(LOS~poly(mortscore,8),data=v1,subset=train )
> r8<-mean((v1$LOS-predict(lm.fit8,v1))[-train]^2)
> lm.fit9=lm(LOS~poly(mortscore,9),data=v1,subset=train )
> r9<-mean((v1$LOS-predict(lm.fit9,v1))[-train]^2)
> lm.fit10=lm(LOS~poly(mortscore,10),data=v1,subset=train )
> r10<-mean((v1$LOS-predict(lm.fit10,v1))[-train]^2)
> model_performance<-c(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10)
> polynomial_degree<-c(1,2,3,4,5,6,7,8,9,10)
> plot(polynomial_degree,model_performance)
>
```



By plotting a graph of varying polynomial degrees of model v/s model performance we observe that, MSE decreases as we increase the polynomial degree of the model. If we look at the MSE values on y axis it is worth noting that all values range in 91 to 93, indicating there is not much benefit in including higher order polynomial terms in model. Although it is worth noting that all the 10 different degrees result in different error rates for the same linear regression model.

On similar lines, when we fit the model for LOS~age the graph for the same is as follows:



We can conclude from above plot, that the MSE for predictor age is less as compared to MSE for predictor mortscore for same response variable LOS. However for above plot , if we observe carefully the MSE starts increasing for the model when the polynomial degree goes beyond 5.

D. 5-fold or 10-fold cross validation

K-fold Cross Validation is an alternative to LOOCV. This approach involves randomly k-fold Cross Validation dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. The mean squared error, MSE1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, MSE1, MSE2,..., MSEk. The k-fold Cross Validation estimate is computed by averaging these values.

The advantage of K-fold cross validation over LOOCV is it's computational capability. LOOCV requires fitting the statistical learning method n times. This has the potential to be computationally expensive whereas the cross-validation is a very general approach that can be applied to almost any statistical learning method.

We have performed 5-fold and 10-fold cross validation on our merged dataset.

#5-fold Cross validation approach on merged dataset for income

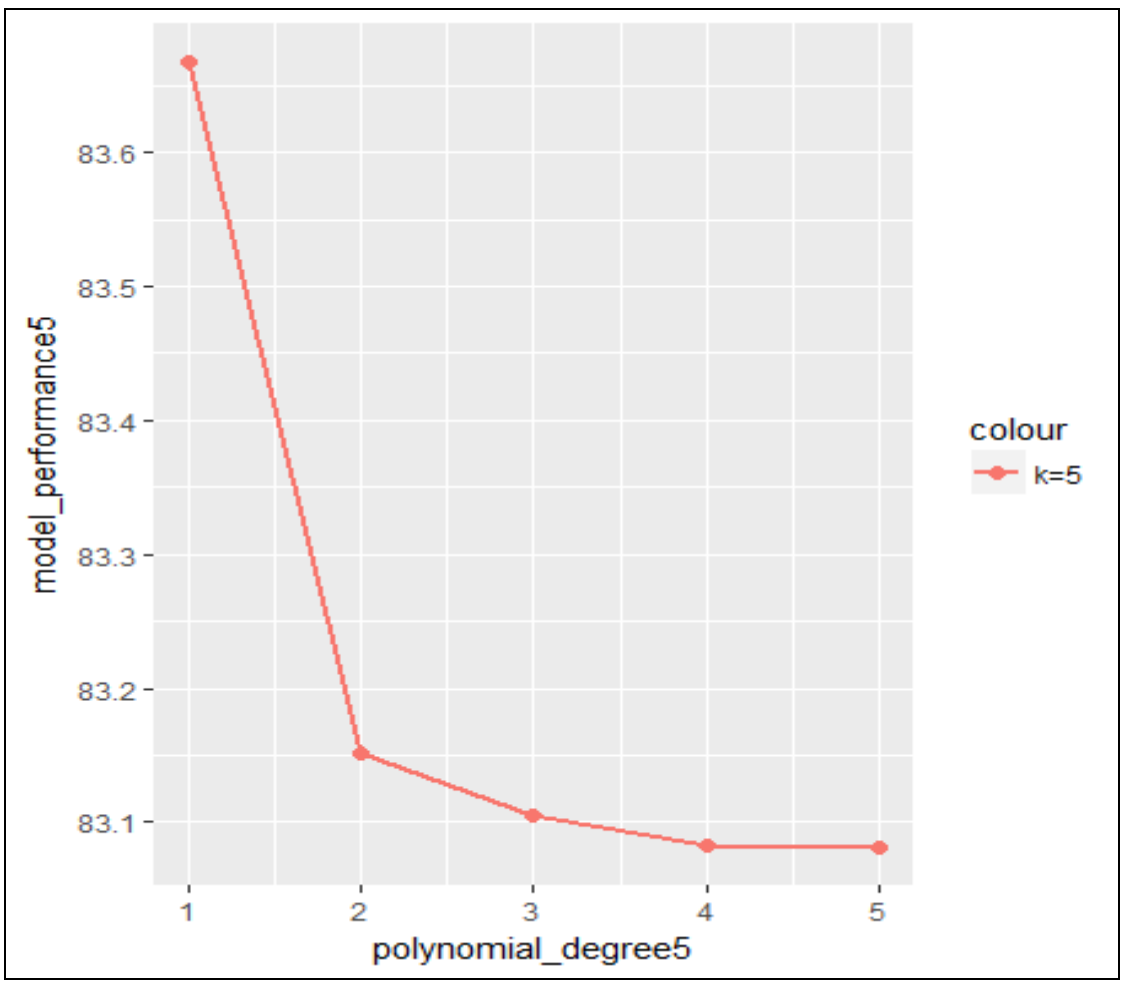
```
library(ISLR)
library(boot)
v1<-read.csv("D:\\Fall              2018\\IS              777              Data
Analytics\\D4\\Merged_Dataset_NO_ID_Factored.csv" , sep = '\t')

head(v1)
summary(v1)

#K=5 cross fold
set.seed(17)
cv.error.5=rep(0 ,5)
glm.fit=numeric()
for (i in 1:5){
  glm.fit=glm(LOS~poly(income,i),data=v1)
  cv.error.5[i]=cv.glm(v1, glm.fit ,K=5) $delta [1]
}
cv.error.5
library(ggplot2)
#plot for 5-fold Cross validation approach on merged dataset for income
polynomial_degree5=c(1:5)
model_performance5=cv.error.5
mse_p <- data.frame(polynomial_degree5,model_performance5)
```

```
ggplot()     +          geom_point(data=mse_p,     aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=2) +
                     geom_line(data=mse_p,          aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=1)
```

output:

```
 Max.   :1.0000     Max.   :1.0000     Max.   :68.45
> #K=5 cross fold
> set.seed(17)
> cv.error.5=rep(0 ,5)
> glm.fit=numeric()
> for (i in 1:5){
+   glm.fit=glm(LOS~poly(income,i),data=v1)
+   cv.error.5[i]=cv.glm(v1, glm.fit ,K=5) $delta [1]
+ }
> cv.error.5
[1] 83.66723 83.15118 83.10507 83.08325 83.08140
```

plot:

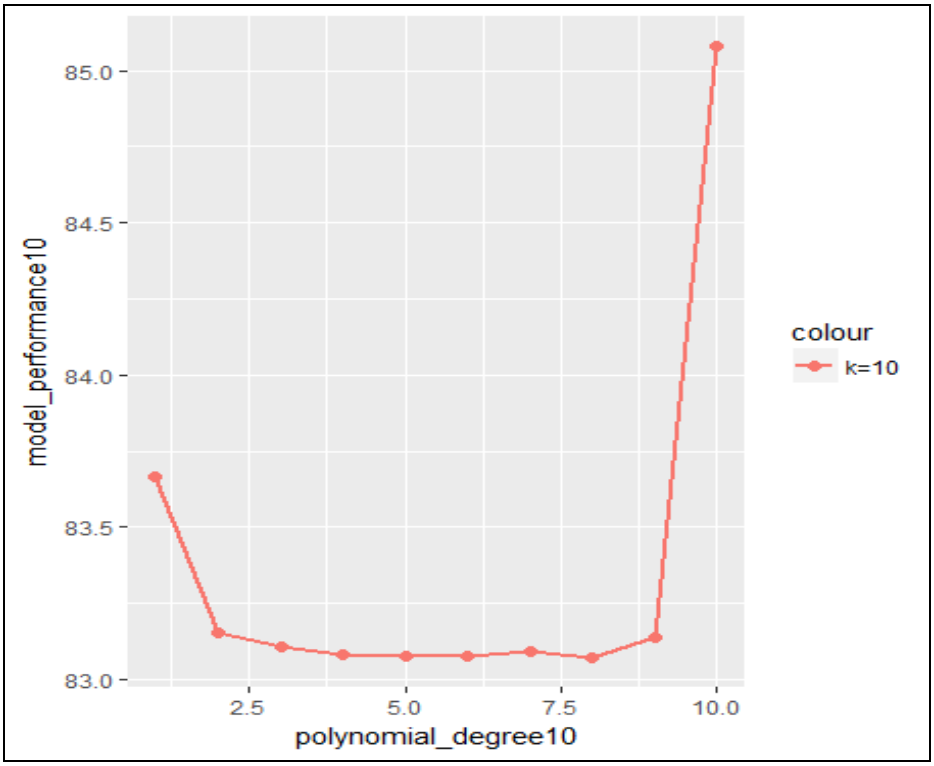


```
#K=10 cross fold Cross validation approach on merged dataset for income
polynomial_degree5=c(1:5)
set.seed(17)
cv.error.10=rep(0 ,10)
for (i in 1:10){
  klm=glm(LOS~poly(income,i), data=v1)
  cv.error.10[i]=cv.glm(v1, klm ,K=10) $delta [1]
}
cv.error.10
#plot
polynomial_degree10=c(1:10)
model_performance10=cv.error.10
mse_p <- data.frame(polynomial_degree10,model_performance10)

ggplot()+
                    geom_point(data=mse_p,          aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=2)+
                     geom_line(data=mse_p,          aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=1)
```

Output:

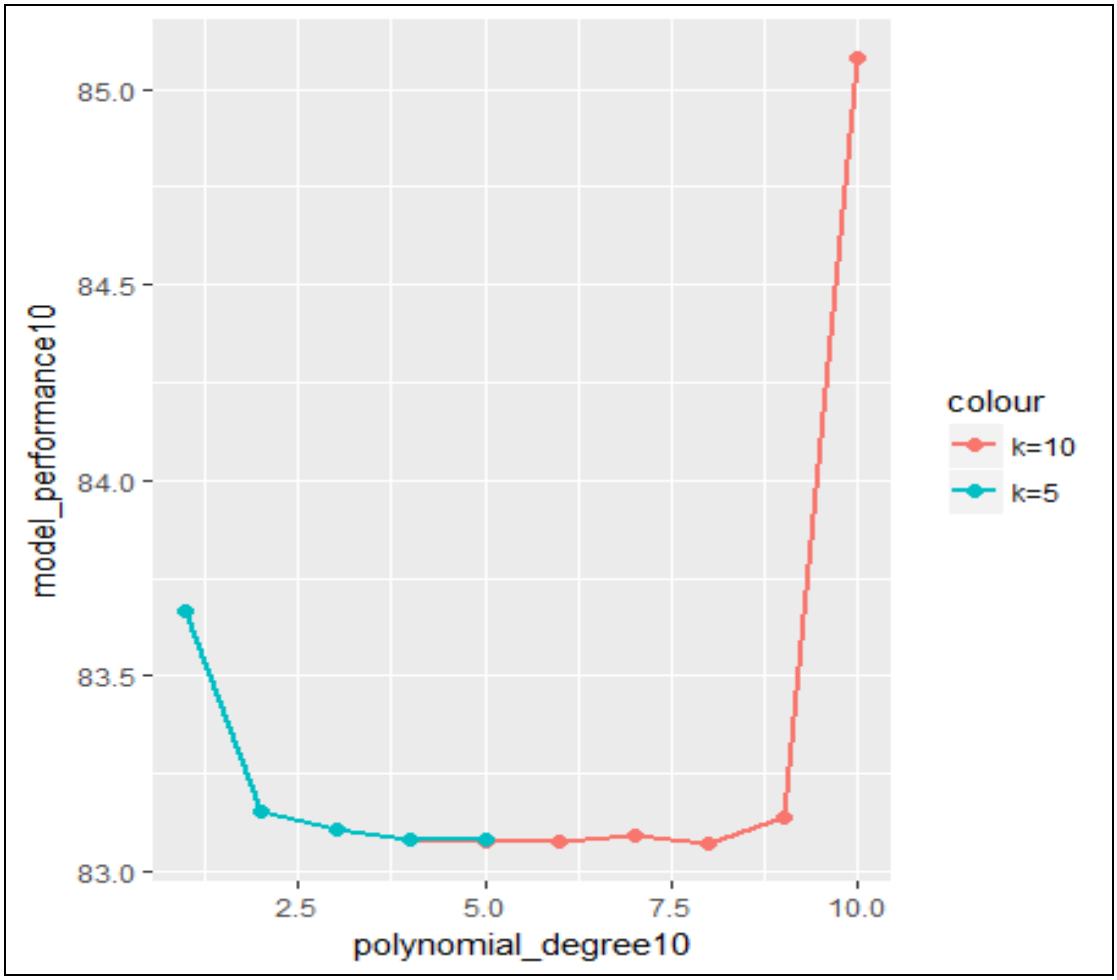
```
> for (i in 1:10){
+   klm=glm(LOS~poly(income,i), data=v1)
+   cv.error.10[i]=cv.glm(v1, klm ,K=10) $delta [1]
+ }
> cv.error.10
 [1] 83.66485 83.15212 83.10713 83.07960 83.07408 83.07680 83.09041 83.07272 83.13820 85.07848
> #K=10 cross fold Cross validation approach on merged dataset for income
```

Plot:




```
#visualize for combined plot both k=5 and k=10 in the same graph

polynomial_degree10=c(1:10)
polynomial_degree5=c(1:5)
model_performance10=cv.error.10
model_performance5=cv.error.5
mse_p                                                                    <-
data.frame(polynomial_degree10,model_performance10,polynomial_degree5,model_p
erformance5)

ggplot()+
                    geom_point(data=mse_p,          aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=2)+
                     geom_line(data=mse_p,          aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=1)+
                     geom_point(data=mse_p,          aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=2)+
                      geom_line(data=mse_p,          aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=1)
```

Combined Plot:



#Code for K=5 fold Cross validation approach on merged dataset for age

```
library(ISLR)
library(boot)
v1<-read.csv("D:\\Fall 2018\\IS 777 Data
Analytics\\D4\\Merged_Dataset_NO_ID_Factored.csv" , sep = '\t')


head(v1)
summary(v1)


#K=5 cross fold
```

```
set.seed(17)
cv.error.5=rep(0 ,5)
glm.fit=numeric()
for (i in 1:5){
  glm.fit=glm(LOS~poly(age,i),data=v1)
  cv.error.5[i]=cv.glm(v1, glm.fit ,K=5) $delta [1]
}
Cv.error.5
```

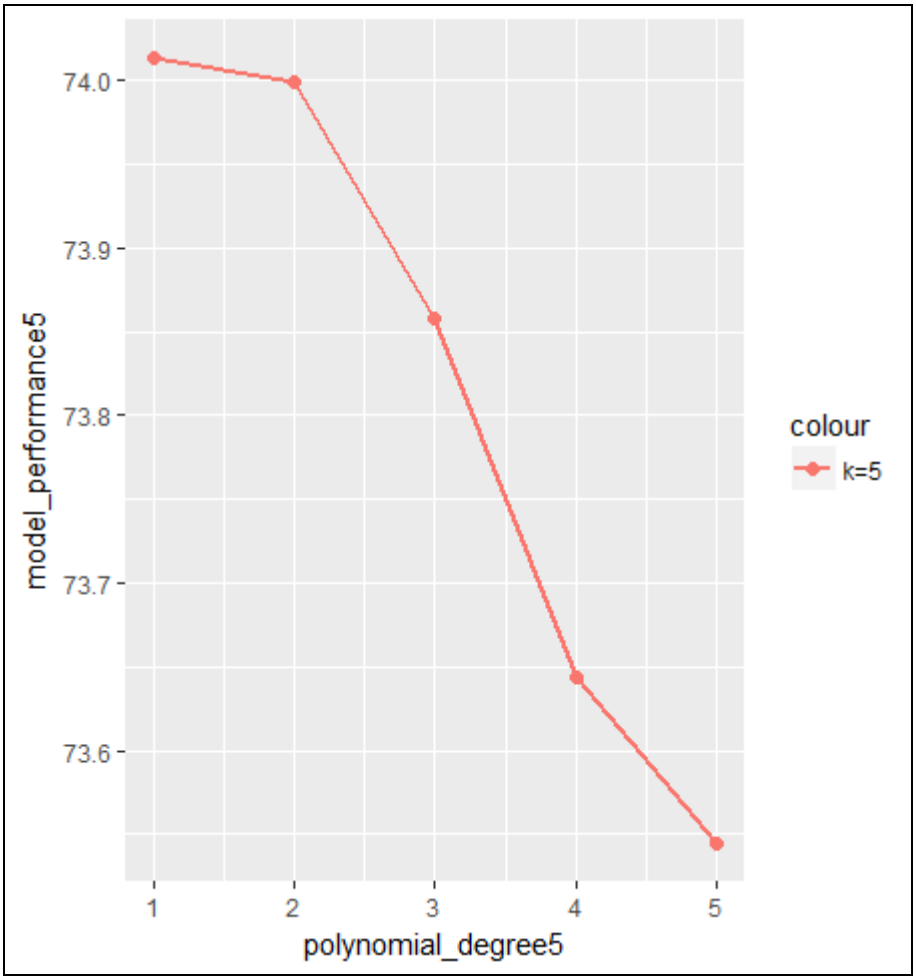
Output:

```
> #K=5 cross fold
> set.seed(17)
> cv.error.5=rep(0 ,5)
> glm.fit=numeric()
> for (i in 1:5){
+   glm.fit=glm(LOS~poly(age,i),data=v1)
+   cv.error.5[i]=cv.glm(v1, glm.fit ,K=5) $delta [1]
+ }
> cv.error.5
[1] 74.01301 73.99931 73.85817 73.64364 73.54467
>
```

#Code for plotting for K=5 fold Cross validation approach on merged dataset for age

```
polynomial_degree5=c(1:5)
model_performance5=cv.error.5
mse_p <- data.frame(polynomial_degree5,model_performance5)
ggplot()+
  geom_point(data=mse_p, aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=1)
```

Plot for K=5 fold Cross validation approach on merged dataset for age:



#Code for K=10 fold Cross validation approach on merged dataset for age

```
#K=10 cross fold
set.seed(17)
cv.error.10=rep(0 ,10)
for (i in 1:10){
  klm=glm(LOS~poly(age,i), data=v1)
  cv.error.10[i]=cv.glm(v1, klm ,K=10) $delta [1]
}
Cv.error.10
```

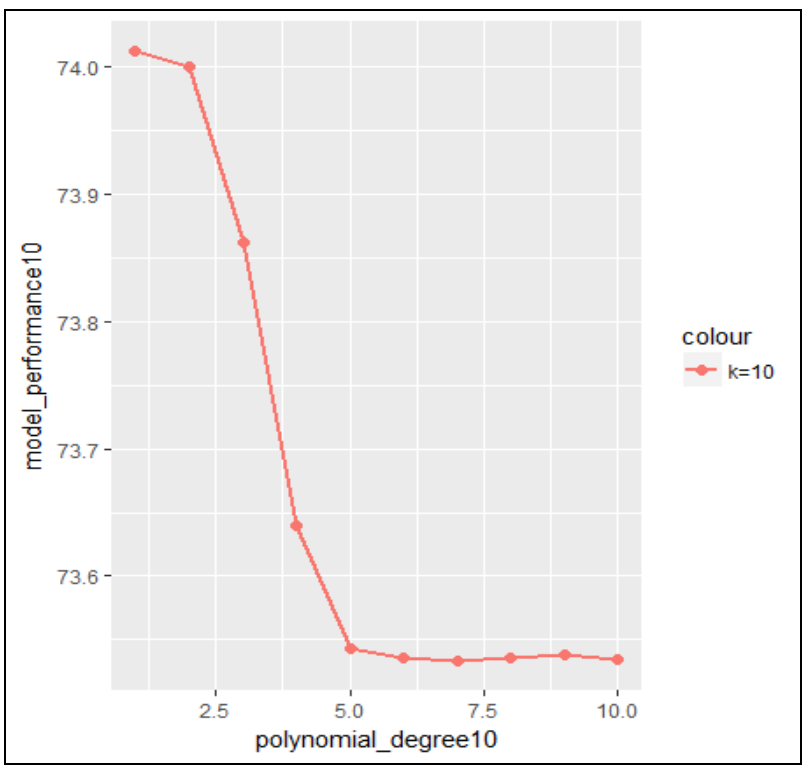
Output:

```
+   geom_line(data=mse_p, aes(x=polynomial_degree5, y=model_performance5,color='k=5'), size=1)
> #K=10 cross fold
> set.seed(17)
> cv.error.10=rep(0 ,10)
> for (i in 1:10){
+   klm=glm(LOS~poly(age,i), data=v1)
+   cv.error.10[i]=cv.glm(v1, klm ,K=10) $delta [1]
+ }
> cv.error.10
 [1] 74.01208 73.99971 73.86219 73.63911 73.54288 73.53586 73.53354 73.53566 73.53777 73.53497
```

#Code for plotting for K=10 fold Cross validation approach on merged dataset for age

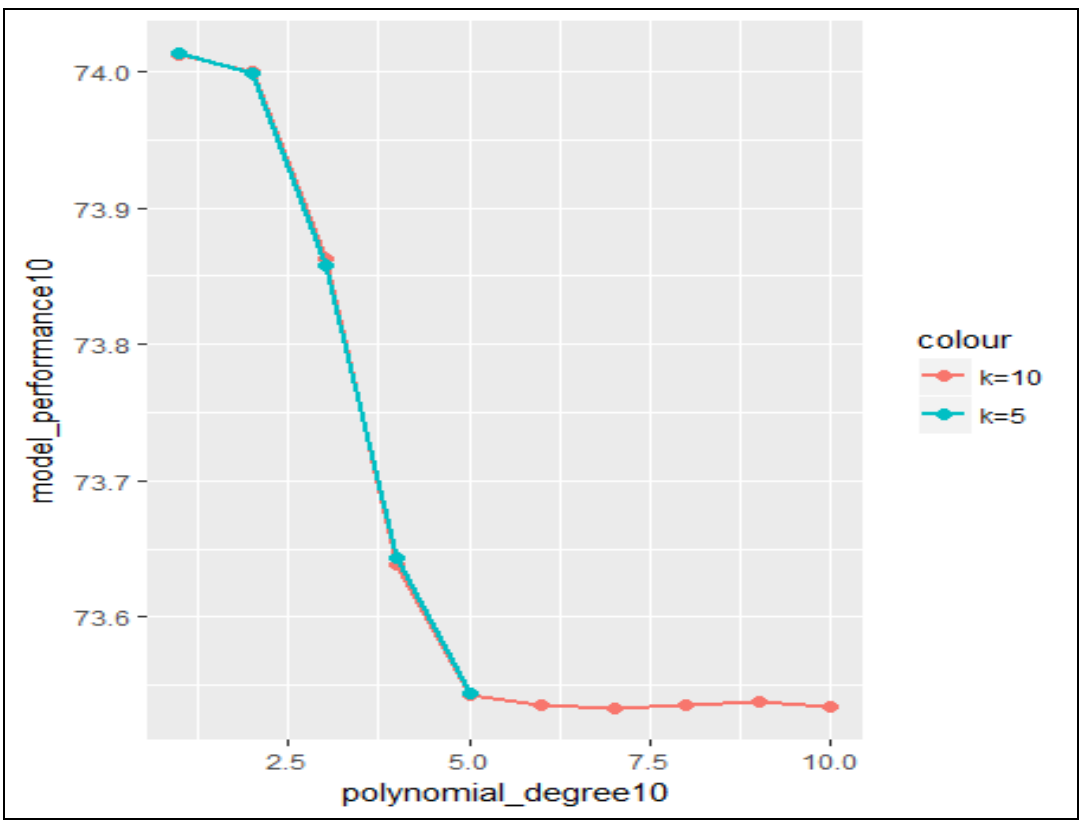
```
polynomial_degree10=c(1:10)
model_performance10=cv.error.10
mse_p <- data.frame(polynomial_degree10,model_performance10)
ggplot()+
  geom_point(data=mse_p, aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=1)
```

Plot:



```
#visualize combined plot for both k=5 and k=10 in the same graph
polynomial_degree10=c(1:10)
polynomial_degree5=c(1:5)
model_performance10=cv.error.10
model_performance5=cv.error.5
mse_p <-
data.frame(polynomial_degree10,model_performance10,polynomial_degree5,model_p
erformance5)

ggplot()+
  geom_point(data=mse_p, aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree10,
y=model_performance10,color='k=10'), size=1)+
  geom_point(data=mse_p, aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=2)+
  geom_line(data=mse_p, aes(x=polynomial_degree5,
y=model_performance5,color='k=5'), size=1)
```

Plot:



Results

After analysing the results of all the four approaches, K-fold approach with k=10 gives the best performance with low MSE comparatively for age as the predictor variable. The graphs have been plotted which shows the visual decrease or increase in the MSE values.Although in the report ,it seems like LOOCV has low MSE, but it cant be considered to assess. LOOCV takes longer time to run , hence a small set of data was considered in our analysis.

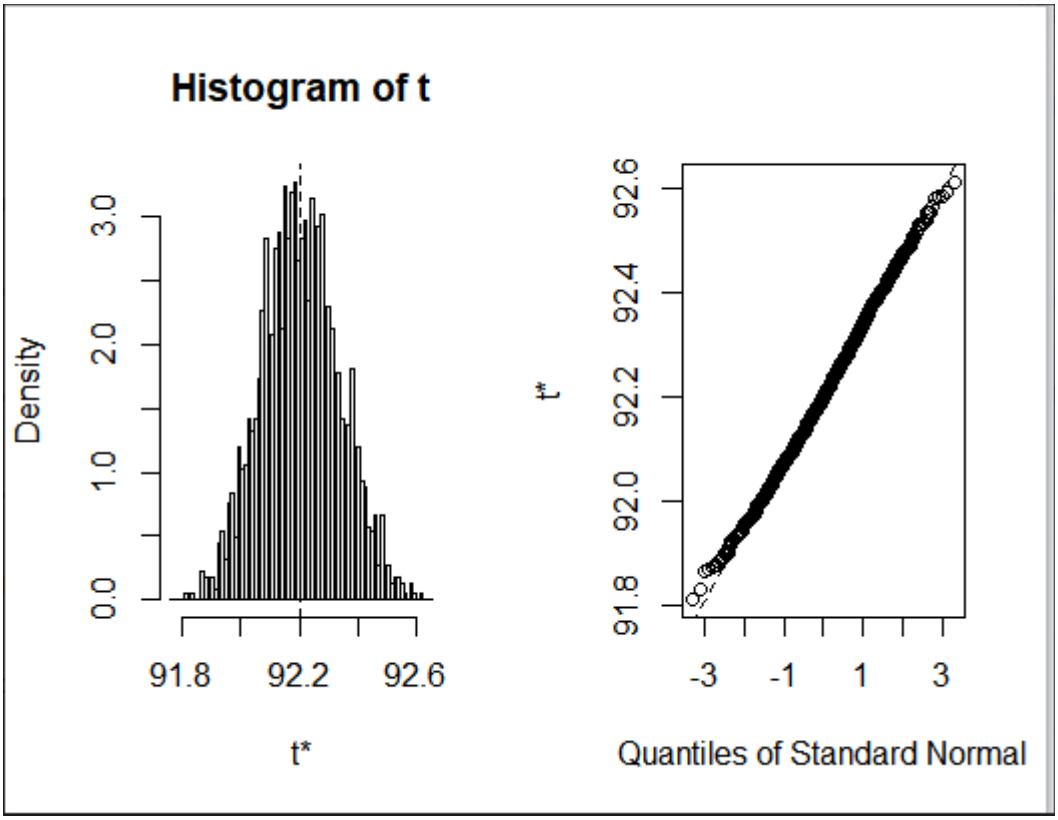
Quantifying parameter uncertainty

Out of our four models, we chose to apply bootstrapping to our linear regression model above comparing coefficients LOS and mortscore. Bootstrapping is a method used to estimate the confidence of a model when making predictions about the data. We chose to run the bootstrap on our model for 2,000 repetitions in order to generate a substantial sample size given the time and computing resources at our disposal. The bootstrap function in R returns intervals using four different methods. These intervals are the following:normal (calculated using normal approximation), basic (calculated using basic bootstrap method), percentile (calculated using bootstrap percentile method), and bca (calculated using the adjusted bootstrap percentile method).

The results of quantifying the uncertainty of our linear regression model comparing LOS and mortscore were the following, using the four methods described above:

Normal - Lower limit: 91.95 Upper limit: 92.46
Basic - Lower limit: 91.94 Upper limit: 92.45
Percentile - Lower limit: 91.96 Upper limit: 92.47
BCA - Lower limit: 91.97 Upper limit: 92.48

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = mean_results)

Intervals :
Level      Normal              Basic
95%   (91.95, 92.46 )   (91.94, 92.45 )

Level     Percentile            BCa
95%   (91.96, 92.47 )   (91.97, 92.48 )
Calculations and Intervals on Original Scale
```


Histogram of t
Density
0.0
1.0
2.0
3.0
91.8
92.2
92.6
t*
t*
91.8
92.0
92.2
92.4
92.6
-3
-1
1
3
Quantiles of Standard Normal

D5: Model Selection

Requirement

PLEASE NOTE: ONLY THE PROJECT MANAGER SHOULD UPLOAD A SINGLE POWERPOINT SLIDE BEFORE THE CLASS TIME.

Part A: MODEL SELECTION: By using the k-fold cross validation to assess predictive performance, choose the model with the size leading to the best prediction performance by using the following techniques:

1. Forward selection 2. Backwards selection 3. Ridge regression 4. Lasso

You should demonstrate the analysis steps in your report and explain the results obtained by using each method. Discuss what conclusion is reached by using each method as well as your overall conclusions after experimenting with these techniques. You can include higher degrees of the numeric variables in your models, if you have observed non-linear relationships in the analysis performed for the earlier deliverables.

Part B: ACCOMMODATING NON-LINEARITY

Use Generalized Additive Models in an exploratory fashion by relaxing the assumption of linearity for your numeric variables. You can use any splines technique to relax the assumption of linearity in your generalized additive models. After some experimentation, you should decide on how much non-linearity should be used for each variable. This decision should be based on predictive performance results. Present your final model along with the plots showing the relationship between x's and f(x)'s. Your D5 report should include all the meaningful steps you went through as a team from the beginning of the semester including the last analysis steps. This document should look like the lab sections in the book. It should include the narrative component explaining what you are doing and why along with the R code snippets, textual R outputs, and plots as appropriate. Your deliverable uploaded by the project manager should include a zip file and uploaded by only the project manager before the deliverable due date and time. Your zip file MUST include only two files: One file will be for your D5 document (pdf is preferrable) which is the cumulative and final report; the other will be for your R script including the R statements used in the analysis. All figures and tables should be embedded in your report.

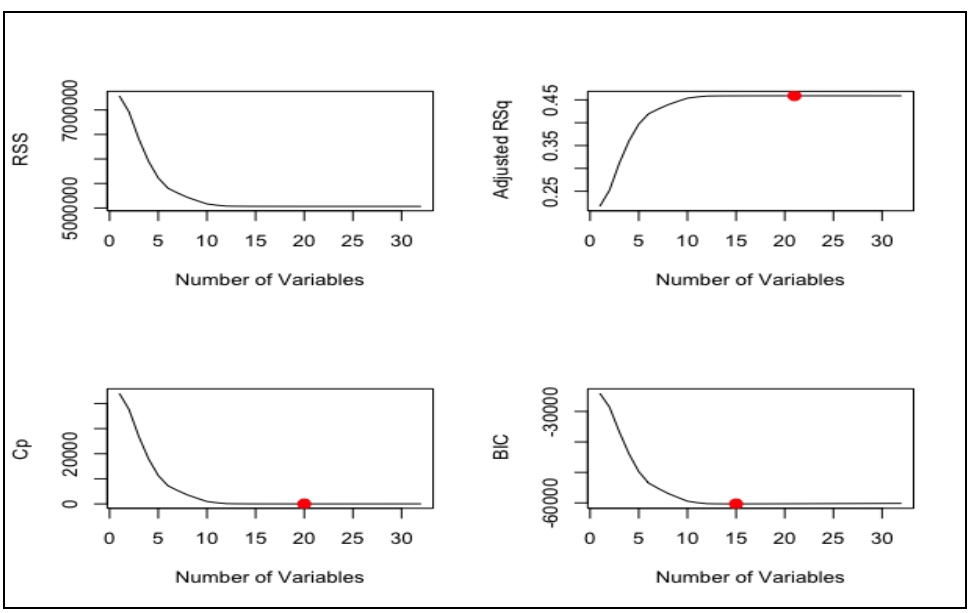
Part A: MODEL SELECTION

By using the k-fold cross-validation to assess predictive performance, choose the model with the size leading to the best prediction performance by using the following techniques:

1. Forward selection

In the forward selection, we start with the null model—a model that contains an intercept yet no indicators. We at that point fit p basic direct relapses and add to the invalid model the variable that outcomes in the least RSS. Further, We add to that demonstrate the variable that outcomes in the most reduced RSS for the new two-variable model. This methodology proceeds until the point that some stopping rule is satisfied.

Also, the forward stepwise selection is a computationally productive choice to best forward stepwise choice subset determination. While the best subset selection method considers all 2p conceivable models containing subsets of the p indicators, forward stepwise thinks about a lot littler arrangement of models. Forward stepwise determination starts with a model containing no indicators, and afterward adds indicators to the model, each one in turn until the majority of the indicators is in the model. Specifically, at each progression, the variable that gives the best extra enhancement to the fit is added to the model.



It is clear we are not getting best model using all variables, some data to support it as below:

1) RSS VS Number of variables 2) Adjusted R Square 3) Cp 4) BIC VS Number of variables

Regsubsets using full model (using all variables):

Maximum Adjusted R Square achieved with 21 variables:
which.max(reg.summary$adjr2) #gives 21 value 0.4590097 fwd 0.4588482 bwd 0.4590097

Minimum cp is at 20 variable point:
which.min(reg.summary$cp) #gives 20 value 14.63332 fwd 14.63332 bwd 14.63332

Minimum BIC is with variable 15:
which.min(reg.summary$bic) #gives 15 value -60319.26 fwd -60319.26 bwd -60319.26

Normal - Lower limit: 91.95 Upper limit: 92.46
Basic - Lower limit: 91.94 Upper limit: 92.45
Percentile - Lower limit: 91.96 Upper limit: 92.47
BCA - Lower limit: 91.97 Upper limit: 92.48

2. Backward selection

Similar to forward stepwise selection, backward stepwise selection starts with the full squares model including every predictor and then iteratively works backward by removing one predictor at a time. The backward selection can be applied situationally, where best subset selection would not work due to p being too great.

Backward stepwise selection also has no guarantee to produce the optimal set of p predictors. Backward selection also has the requirement that the number of samples are greater than the number of predictors, however forward stepwise selection can be used even when the number of samples is less than the number of predictors.

After running this approach on our merged dataset, we found the following subset of eleven predictors to be most favorable: Age, gender, area, income, drugu,

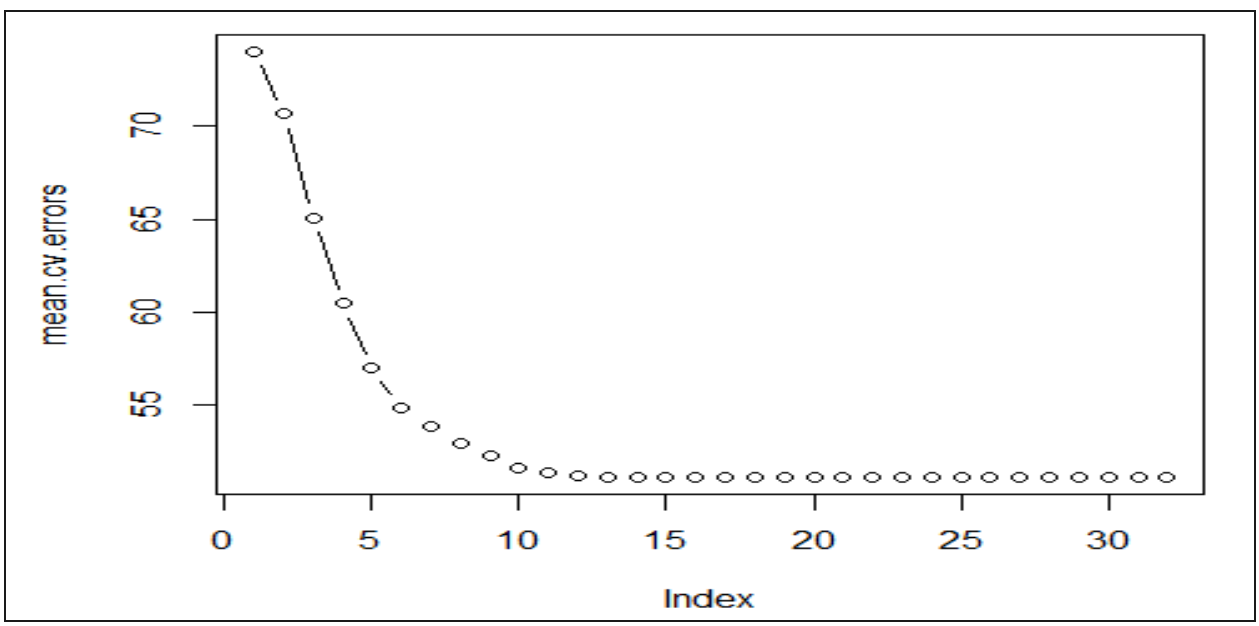
alcolu, checkfall, checkdepression, checkvaccine, checkfootcare, and admittedhospital.

Below you can see that this was found by plotting the mean cross-validation error against the number of predictors to find that eleven is right where the graph begins to plateau. The coefficients of our 11 variable model are found below.

Input:

```
#Apply K cross validation, here we create a vector that allocates each observation
#to one of k = 10 folds, and we create a matrix in which we will store the results.
k=10
set.seed(1)
folds=sample(1:k,nrow(Merged_DS),replace =TRUE)
cv.errors = matrix(NA ,k,32, dimnames =list(NULL,paste(1:32)))

#This will give us 10 by 32 matrix, of which the (i, j)th element corresponds
#to the test MSE for the ith cross-validation fold for the best j-variable model.
for(j in 1:k){
  best.fit =regsubsets(LOS ~ .,data=Merged_DS[folds !=j,],nvmax =32)
  for(i in 1:32) {
    pred=predict(best.fit,Merged_DS[folds ==j,], id=i)
    cv.errors[j,i]=mean((Merged_DS$LOS[folds ==j]-pred)^2)
  }
}

#use apply() function to average over the columns of this matrix in order to obtain a
#vector for which the jth element is the crossvalidation error for the j-variable model.
mean.cv.errors =apply(cv.errors ,2, mean)


#See mean error of cross validation
mean.cv.errors

par(mfrow =c(1,1))

#Plot error, type = 'b' indicate line.
plot(mean.cv.errors,type='b')
```

Output:



Input:

```
#We now perform best subset selection on the full data set in order to obtain the 11-variablen model.
reg.best=regsubsets(LOS ~ .,data=Merged_DS, nvmax =32)

#See the cofficient of 11 variable use for our best model selected by k fold cross validation.
coef(reg.best ,11)
```

Output:

```
> coef(reg.best ,11)
     (Intercept)             age          gender            area          income
      16.3320487       0.3134429      -3.9043440      -5.4247689      -0.0643890
           drugu          alcolu       checkfall  checkdepression     checkvaccine
      -1.8865419       3.6329150       3.3448770       1.6675606       2.0420682
   checkfootcare admittedhospital
       1.0443092       1.7956612
```

3. Ridge regression

Subset selection uses least squares to fit a linear model that contains subset of predictors. We can also fit the model containing all ‘p’ predictors that shrink the coefficients towards zero. One Such method is Ridge Regression.

Shrinking the coefficient estimates can reduce the coefficient variance. Ridge Regression works best in situations where the least squares estimates have high variance.

Least squares is a statistical method used to determine a line of best fit by minimizing the sum of squares created by mathematical function. Since other fitting procedures yield better prediction accuracy, we use ridge regression.

We first merge all the 4 datasets provided to group c. The merged datasets have 33 attributes and 98484 variables. We first fit the ridge regression model on the dataset using LOS as response variable and rest of the attributes as predictors. The grid is used to range values of lambda:

```
#Fitting ridge regression model
x=model.matrix(v1$LOS~.,v1)[,-1]
y=v1$LOS
grid=10^seq(10,-2,length =100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod ))
```

The value of coefficients when lambda is large and small is compared:

```
#value of coefficients when lambda is large
ridge.mod$lambda[50]
coef(ridge.mod)[,50]
sqrt(sum(coef(ridge.mod)[ -1 ,50]^2))
#value of coefficients when lambda is small
ridge.mod$lambda [60]
coef(ridge.mod)[,60]
sqrt(sum(coef(ridge.mod)[ -1 ,60]^2))
```

The output is:

```
coef(ridge.mod)[,50]
        (Intercept)                 age          gendermale
       2.558827e+01        2.852894e-04        -2.297590e-03
            raceASA             raceEUA             raceHLA
       9.244246e-05       -9.600117e-05        -6.747083e-05
            raceOTH          maritalMP          maritalOTH
       1.634582e-05       -2.308668e-04         6.290127e-05
          maritalSG          maritalWD            areaurban
      -9.062012e-05        1.163260e-04        -1.845300e-03
             income        educationHS        educationOTH
      -4.932755e-05        8.177546e-05        -6.963862e-05
       educationUD          mortscore       betterwalkyes
       1.177233e-04       -3.217726e-08         1.192257e-04
       betterbedyes       betterbathyes       bettermoveyes
       9.412508e-04       -6.351667e-04         5.060506e-04
   betterbreathyes       betterhealyes    bettertakingyes
       3.395374e-05        3.970322e-05        -9.871127e-05
     adverseeventAR    adverseeventSAE    adverseeventSSAR
       1.528578e-04       -6.926100e-05        -6.975902e-06
 adverseeventSUSAR           druguyes            alcoluyes
       1.464469e-04       -2.760475e-03         2.855886e-03
        istimelyyes       taughtdrugyes          checkfallyes
       3.758304e-04        9.269565e-05         8.422333e-04
checkdepressionyes    checkflushotyes     checkvaccineyes
       1.262649e-03        7.992021e-06         9.933629e-04
  checkfootcareyes          HHTcareyes          HHTcommyes
      -2.593946e-04        8.660759e-05        -2.605735e-06
      HHTdiscussyes  admittedhospitalyes           urgentyes
       3.071894e-04        4.733197e-04         4.042179e-04
admittedhospitalyes  emergencyhospitalyes
       3.793041e-04       -2.022046e-03
```

```
ridge.mod$lambda [60]
] 705.4802
coef(ridge.mod)[,60]
        (Intercept)                 age          gendermale
       2.547569e+01        4.573788e-03        -3.724342e-02
            raceASA             raceEUA              raceHLA
       1.470769e-03       -1.526137e-03        -1.065495e-03
            raceOTH           maritalMP           maritalOTH
       2.547838e-04       -3.685997e-03         1.000054e-03
          maritalSG           maritalWD             areaurban
      -1.448963e-03        1.866448e-03        -3.028640e-02
             income         educationHS         educationOTH
      -7.903890e-04        1.289244e-03        -1.096629e-03
        educationUD           mortscore        betterwalkyes
       1.871337e-03       -5.099791e-07         1.896355e-03
       betterbedyes       betterbathyes        bettermoveyes
       1.503475e-02       -1.018738e-02         8.004564e-03
    betterbreathyes       betterhealyes      bettertakingyes
       5.344793e-04        6.308541e-04        -1.610676e-03
     adverseeventAR     adverseeventSAE     adverseeventSSAR
       2.434682e-03       -1.089766e-03        -9.121733e-05
  adverseeventSUSAR            druguyes            alcoluyes
       2.331615e-03       -4.409182e-02         4.594691e-02
        istimelyyes       taughtdrugyes         checkfallyes
       6.037300e-03        1.450686e-03         1.400387e-02
 checkdepressionyes     checkflushotyes      checkvaccineyes
       2.038199e-02        1.540164e-04         1.613369e-02
   checkfootcareyes          HHTcareyes           HHTcommyes
      -3.910761e-03        1.416619e-03        -6.065392e-05
      HHTdiscussyes  admittedhospitalyes            urgentyes
       4.873132e-03        7.936832e-03         6.799286e-03
admittedhospitalyes emergencyhospitalyes
       6.345039e-03       -3.217372e-02
```

It is found that when the value of lambda is small the coefficients have higher value.

We want to shrink the value of coefficients to zero for which we need to find out effective lambda value. We split the dataset into test and training dataset:

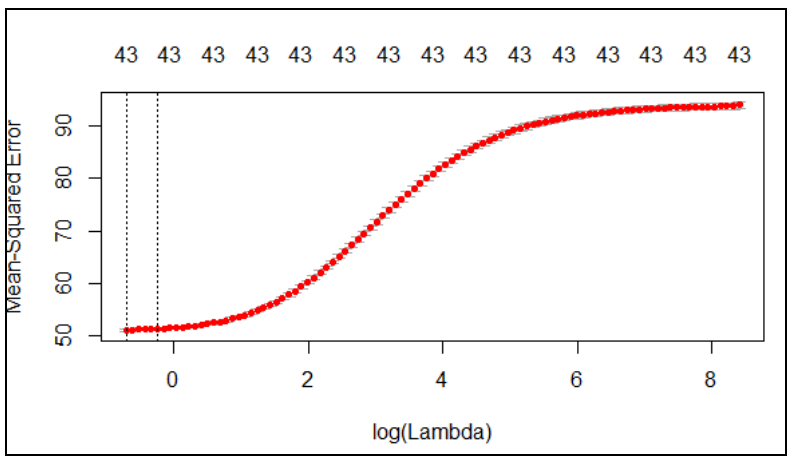
```
 #Splitting model into test set and training set, to estimate the test error of ridge
egression
 set.seed(1)
 train=sample(1:nrow(x),nrow(x)/2)
 test=(-train)
 y.test=y[test]
 #Fitting regression model on training set and evaluate its MSE on test set
 ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid,thresh=1e-12)
 ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
 mean((ridge.pred-y.test)^2)
1] 56.11325
 #Fitting regression model on training set and evaluate its MSE on test set, lambda=10
10
 ridge.pred=predict(ridge.mod ,s=1e10 ,newx=x[test,])
 mean(( ridge.pred -y.test)^2)
1] 95.06044
```

When λ.=4 the MSE on test dataset is 56.11 and when λ.=10^10 the MSE on test data set is 95.06.

By using cross validation we find the best λ. Value, which as mentioned below is 0.49:

```
#cross-validation to choose the tuning parameter
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
1] 0.4925447
```

The graph pasted below mentions different MSE for increasing value of λ. It is found that the best λ value lies between the two vertical lines:



By fitting the ridge model using the best λ. Value obtained above we found the MSE value for test dataset to be 51.52.

Output:

```
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)
] 51.52527
```

Now, we try to refit ridge regression on entire dataset using λ obtained from cross validation:

```
#refit ridge regression on full dataset using lambda value obtained by using cv
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlam )[1:33,]
    (Intercept)             age      gendermale         raceASA
   1.720703e+01    2.927359e-01   -3.659126e+00    2.844286e-02
        raceEUA         raceHLA         raceOTH       maritalMP
  -5.939498e-03   -2.065144e-02   -3.711772e-03   -2.014983e-01
     maritalOTH       maritalSG       maritalWD       areaurban
   8.840395e-03   -6.600573e-02    4.915623e-02   -4.949193e+00
         income     educationHS    educationOTH     educationUD
  -5.997227e-02    5.288037e-02    1.025338e-02    1.257894e-01
      mortscore   betterwalkyes    betterbedyes   betterbathyes
  -2.780515e-06    5.642831e-04    3.941144e-01   -6.028063e-01
  bettermoveyes  betterbreathyes   betterhealyes  bettertakingyes
  -1.321994e-01   -1.593140e-01    2.193594e-03   -1.549724e-01
 adverseeventAR  adverseeventSAE  adverseeventSSAR adverseeventSUSAR
   1.248218e-01    1.991193e-02    4.827465e-02    1.404940e-01
       druguyes       alcoluyes     istimelyyes    taughtdrugyes
  -1.792174e+00    3.395687e+00    3.136345e-01   -7.355181e-02
   checkfallyes
   3.054469e+00
```

As expected none of the coefficients have shrunk to zero. Thus ridge regression doesn't perform variable selection.

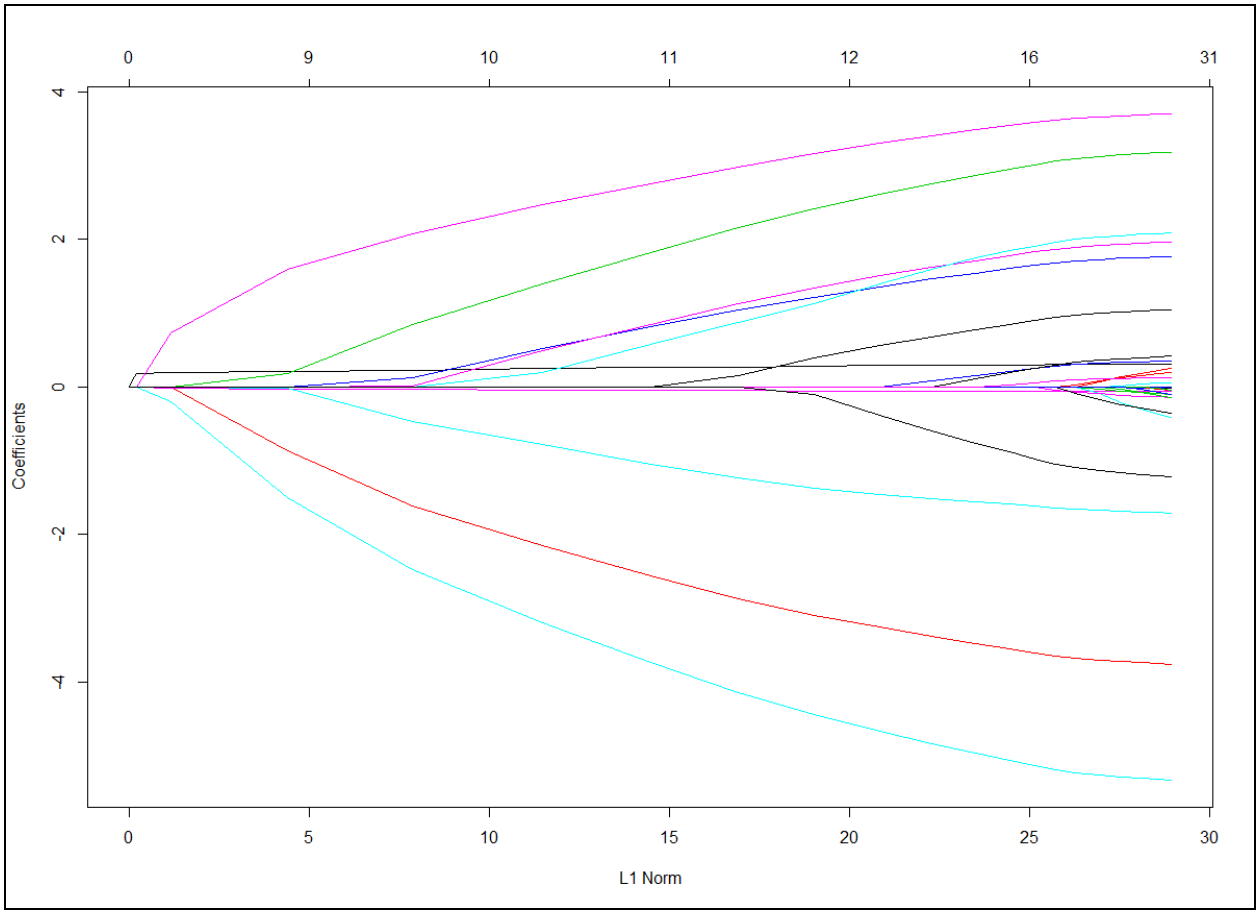
4. Lasso

Ridge regression has one disadvantage. Ridge regression includes all the p predictors in the model. The penalty factor shrinks all the coefficients towards zero, but not set any of the coefficients to zero. There is no effect on prediction accuracy but the challenge in the interpretation.

Lasso is a relative alternate for the ridge regression that overcomes the disadvantage. The lasso coefficient ˆ βLλ minimizes the coefficients and shrinks to zero. The lasso uses an l1 (pronounced "ell 1") penalty instead of an l2 penalty. The 1 norm of a coefficient vector β is given byβ1 =Σ|βj |. Lasso performs variable selection. As a result, the models are easy to interpret.

So applying the lasso model to our data set. Below are the steps followed in fitting lasso model to our data set:

```
#Splitting the dataset into train and test data to fit the lasso model
set.seed (1)
train=sample (1: nrow(x), nrow(x)/2)
test=(- train )
y.test=y[test]
#defining the range of lambda values
grid =10^ seq (10,-2, length =100)
lasso.mod =glmnet (x[train ,],y[train],alpha =1, lambda =grid)
#plotting the coefficient estimates of the lasso
plot(lasso.mod)
```



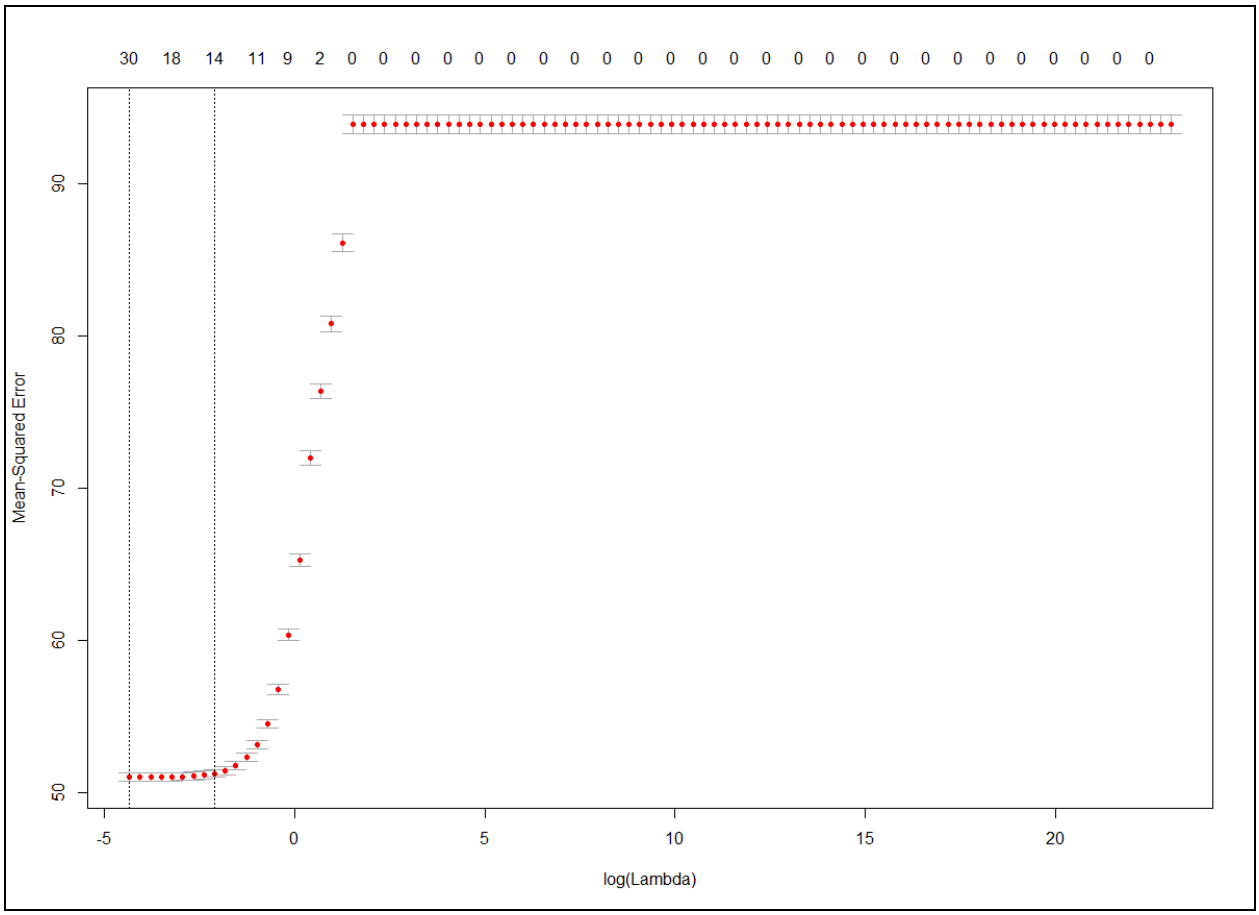
The above figure is the coefficient estimate graphs. Depending on the tuning parameter some of the coefficients will be exactly equal to 0.

```
> lasso.mod

Call:   glmnet(x = x[train, ], y = y[train], alpha =

        Df     %Dev     Lambda
  [1,]   0 0.00000 1.000e+10
  [2,]   0 0.00000 7.565e+09
  [3,]   0 0.00000 5.722e+09
  [4,]   0 0.00000 4.329e+09
  [5,]   0 0.00000 3.275e+09
  [6,]   0 0.00000 2.477e+09
  [7,]   0 0.00000 1.874e+09
  [8,]   0 0.00000 1.417e+09
  [9,]   0 0.00000 1.072e+09
 [10,]   0 0.00000 8.111e+08
 [11,]   0 0.00000 6.136e+08
 [12,]   0 0.00000 4.642e+08
 [13,]   0 0.00000 3.511e+08
 [14,]   0 0.00000 2.656e+08
 [15,]   0 0.00000 2.009e+08
 [16,]   0 0.00000 1.520e+08
 [17,]   0 0.00000 1.150e+08
 [18,]   0 0.00000 8.697e+07
 [19,]   0 0.00000 6.579e+07
 [20,]   0 0.00000 4.977e+07
 [21,]   0 0.00000 3.765e+07
 [22,]   0 0.00000 2.848e+07
 [23,]   0 0.00000 2.154e+07
 [24,]   0 0.00000 1.630e+07
 [25,]   0 0.00000 1.233e+07
 [26,]   0 0.00000 9.326e+06
 [27,]   0 0.00000 7.055e+06
 [28,]   0 0.00000 5.337e+06
 [29,]   0 0.00000 4.037e+06
 [30,]   0 0.00000 3.054e+06
 [31,]   0 0.00000 2.310e+06
 [32,]   0 0.00000 1.748e+06
 [33,]   0 0.00000 1.322e+06
```

Now applying K-cross-fold validation technique to the lasso model to compute the associated error. The function `cv.glmnet()` is an inbuilt function R which applies 10-cross-fold validation on the lasso model:

```
#Let us perform  cross validation and compute the associated error
set.seed (1)
cv.out =cv.glmnet (x[train ,],y[train],alpha =1,lambda = grid)
plot(cv.out)
bestlam =cv.out$lambda.min
lasso.pred=predict (lasso.mod ,s=bestlam ,newx=x[test ,])
mean(( lasso.pred -y.test)^2)
1] 51.32166
```

As it can be seen that the test set MSE for the lasso is lesser than the null model and of the least square errors. It is also similar to ridge regression MSE but lasso MSE is still less than the ridge regression MSE:



In the above lasso model, it can be interpreted that for the large lambda values, there is no change in the model. The values are all zero. But for lower values of the lambda, there is a drop in the MSE. We can choose any value between the 2 vertical lines as the least lambda value to fit the value. Taking the first deviation of the model,lambda approximately equal to -2, the test MSE is 51.32%.

```
>
> out=glmnet (x,y,alpha =1, lambda =grid)
> lasso.coef=predict (out ,type ="coefficients",s=bestlam )[1:33 ,]
> lasso.coef
      (Intercept)               age            gender              race            marital              area
     1.671726e+01      3.057191e-01      -3.869065e+00      0.000000e+00      -3.154161e-03      -5.359566e+00
           income         education          mortscore        betterwalk          betterbed        betterbath
    -6.452350e-02      -4.931239e-04      -1.779640e-07      0.000000e+00       3.498655e-01      -4.206844e-01
       bettermove       betterbreath         betterheal      bettertaking       adverseevent             drugu
    -1.462783e-01      -6.035009e-02       0.000000e+00      -5.871246e-02      -2.836902e-02      -1.751437e+00
           alcolu          istimely         taughtdrug          checkfall    checkdepression      checkflushot
     3.618111e+00       2.835912e-01      -6.993142e-03       3.275474e+00       1.677629e+00       1.151414e-01
     checkvaccine      checkfootcare            HHTcare           HHTcomm         HHTdiscuss   admittedhospital
     1.967960e+00       9.815050e-01       1.726059e-01      0.000000e+00       0.000000e+00       2.128556e+00
           urgent readmittedhospital  emergencyhospital
     0.000000e+00       9.069747e-02      -1.180688e+00
```

Here we see that the 6 predictor variables out of 32 variables, coefficient estimates are exactly zero. So the lasso model with cross-validation will only have 26 variables in the final model.

The 6 variables which are set to zero are: Race, Betterwalk, Betterheal, HHTcomm, HHTdiscuss, Urgent.

```
> lasso.coef[lasso.coef !=0]
      (Intercept)               age            gender           marital              area            income
     1.671726e+01      3.057191e-01      -3.869065e+00      -3.154161e-03      -5.359566e+00      -6.452350e-02
        education          mortscore          betterbed         betterbath         bettermove       betterbreath
    -4.931239e-04      -1.779640e-07       3.498655e-01      -4.206844e-01      -1.462783e-01      -6.035009e-02
     bettertaking       adverseevent             drugu            alcolu           istimely         taughtdrug
    -5.871246e-02      -2.836902e-02      -1.751437e+00       3.618111e+00       2.835912e-01      -6.993142e-03
        checkfall    checkdepression       checkflushot       checkvaccine      checkfootcare            HHTcare
     3.275474e+00       1.677629e+00       1.151414e-01       1.967960e+00       9.815050e-01       1.726059e-01
  admittedhospital readmittedhospital  emergencyhospital
     2.128556e+00       9.069747e-02      -1.180688e+00
>
```

Hence out of the four model techniques using the k-cross-fold validation approach, we can conclude that lasso model performs well with lower MSE resulting in the model which is easy to interpret.

Part B: ACCOMMODATING NON-LINEARITY

We use Generalized Additive Models (GAM) in an exploratory fashion by relaxing the assumption of linearity for numeric variables. We can use any splines technique to relax the assumption of linearity in our generalized additive models. After some experimentation, we decide on how much non-linearity should be used for each variable. This decision should be based on predictive performance results. Further, we present our final model along with the plots showing the relationship between x's and f(x)'s.

Generalized Additive Models with natural spline

To accommodate non-linearity of the model, we used generalized additive models (GAM) with the natural spline. We choose the natural spline because the natural spline extrapolates linearly beyond the boundary knots, so it makes the model less wiggled. We had an exploration with three numeric variables which are age, income, and mortscore. After a lot of trials and errors, we found the four most decent models. To compare these four models, firstly we had plotted the graph of each model and checked it with our eyes. However, there was a limit to analyze the graph only with our eyes, so we checked the summary and finally used ANOVA function to compare the F value of each model.

Experiment of models

We adjusted degrees of freedom of each variable for the experiment and found four most promising models.

```
gam1 = gam(LOS~ns(age,11) + ns(income, 1) + ns(mortscore, 4), data=datafile)
gam2 = gam(LOS~ns(age,11) + ns(income, 1) + ns(mortscore, 5), data=datafile)
gam3 = gam(LOS~ns(age,1) + ns(income, 1) + ns(mortscore, 4), data=datafile)
gam4 = gam(LOS~ns(age,1) + ns(income, 1) + ns(mortscore, 5), data=datafile)
```

We took care of three things when we inspected models in the experiments: Degree of wiggliness, tail wagging, and standard errors. Also, we inspected the various shape of the plotted model if it has less standard errors to make the best decision. For example,

we assumed that linear degree is the best for the age, but we also look into the square degree model because it represented low standard errors.

```
par(mfrow=c(1,3))
plot(gam1, se=TRUE,col="blue")
plot(gam2, se=TRUE,col="blue")
plot(gam3, se=TRUE,col="blue")
plot(gam4, se=TRUE,col="blue")
```

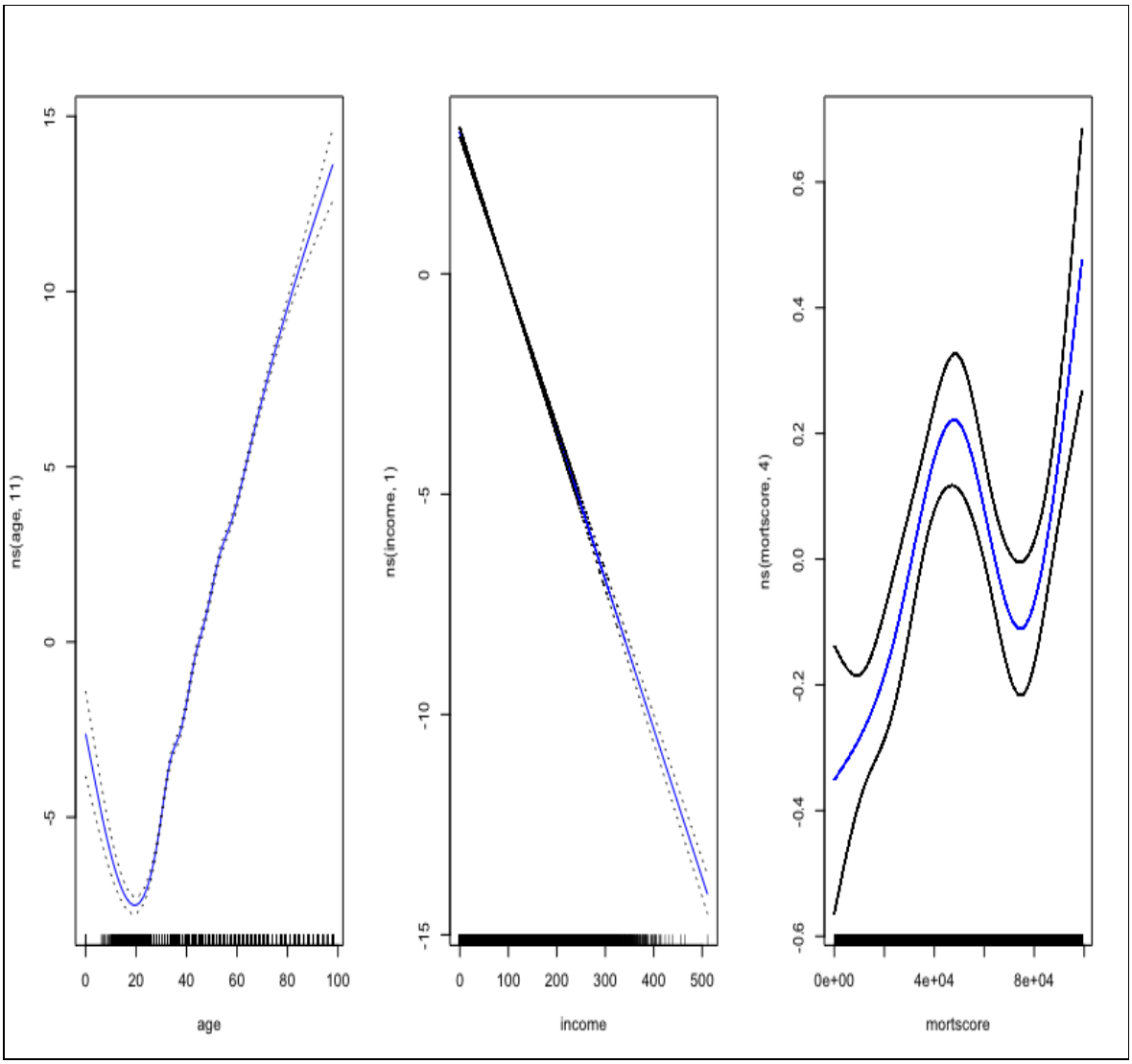


Fig: Graph of gam1

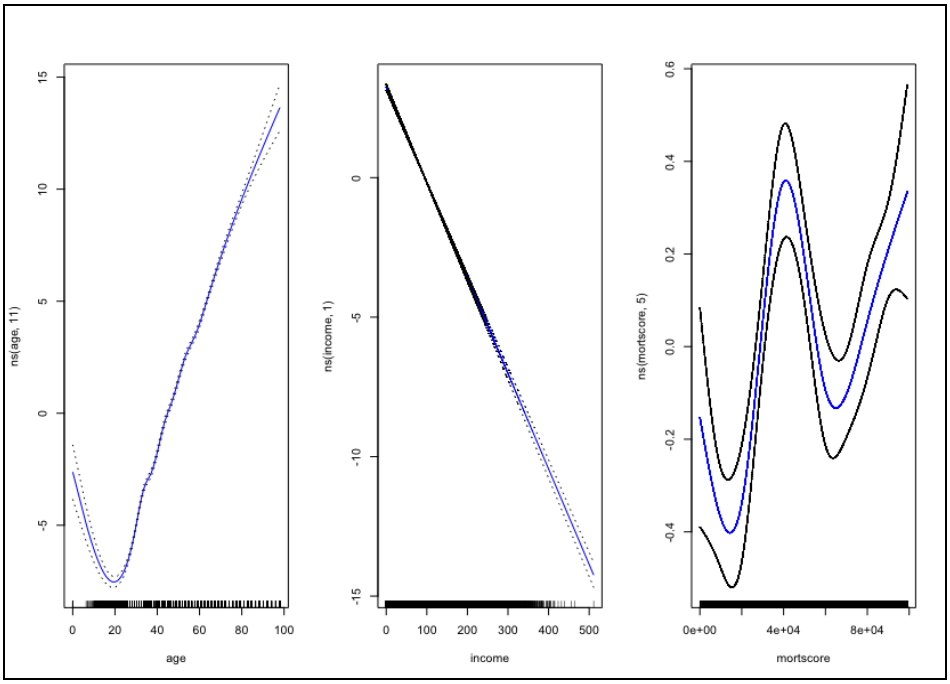


Fig: Graph of gam2

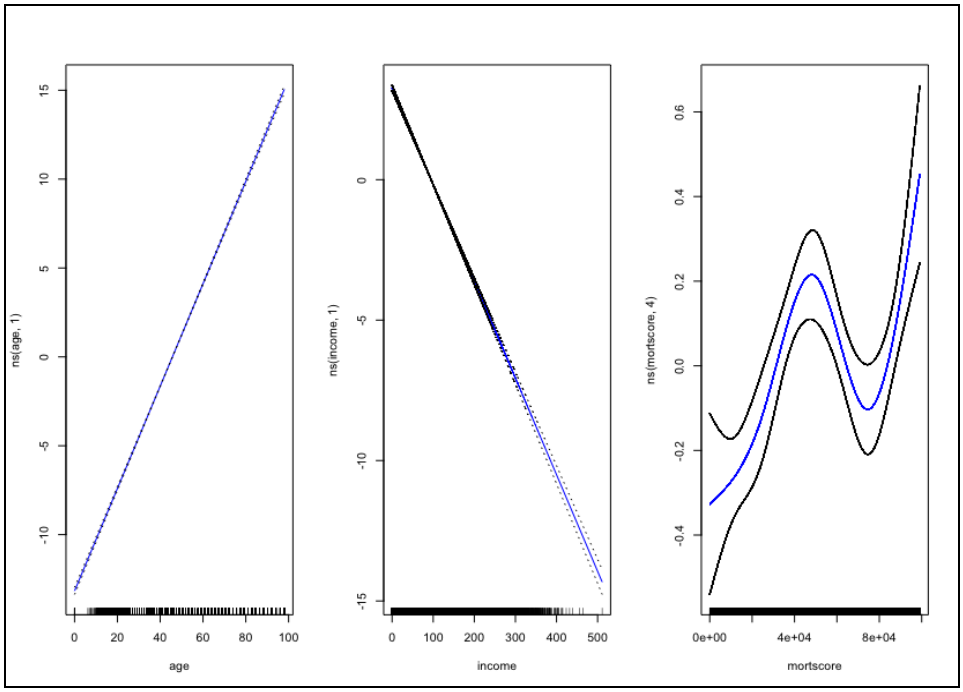


Fig: Graph of gam3

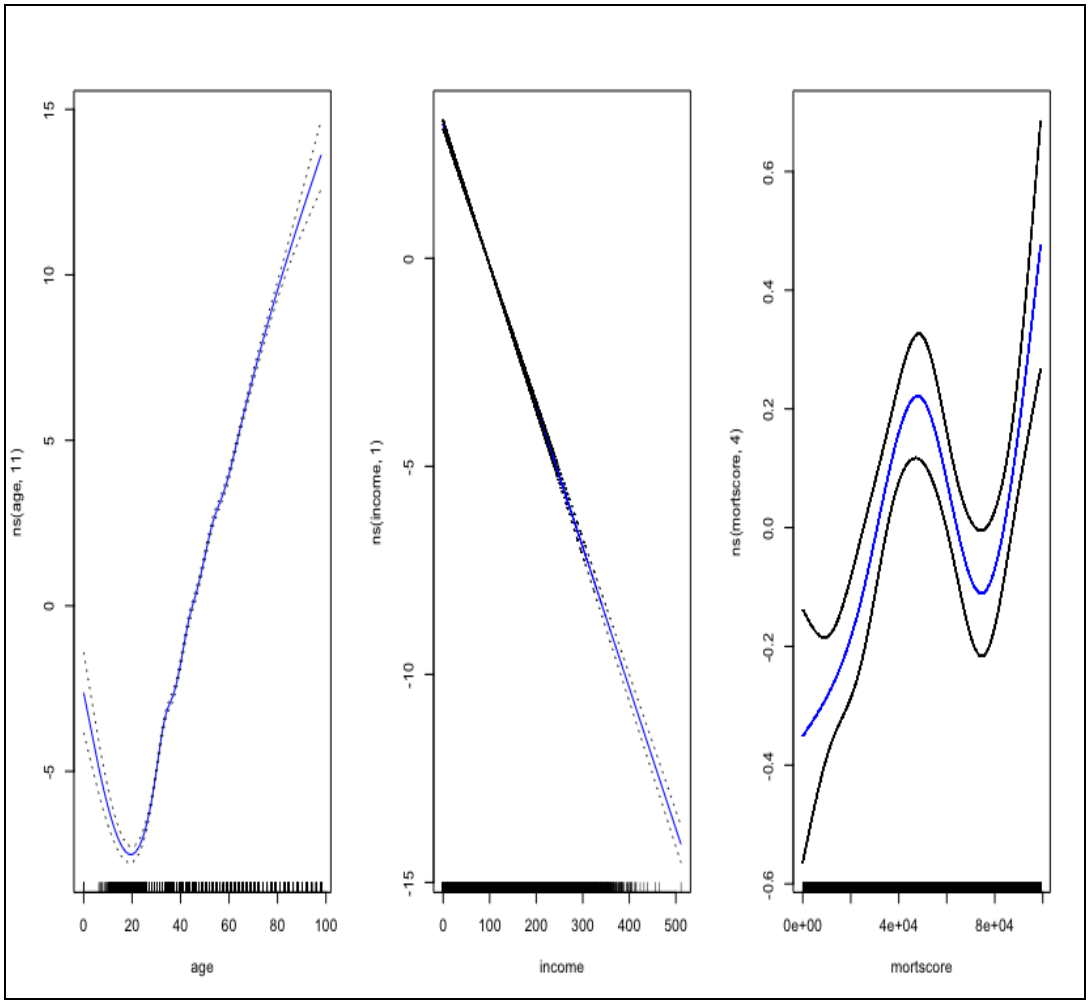


Fig: Graph of gam4

For a more precise analysis, we referred to the summary of each model and used the ANOVA function to compare the F value of each model:

```
summary(gam1)
summary(gam2)
summary(gam3)
summary(gam4)
```

```
> summary(gam1)

Call: gam(formula = LOS ~ ns(age, 11) + ns(income, 1) + ns(mortscore,
    4), data = datafile)
Deviance Residuals:
      Min        1Q    Median        3Q       Max
-33.38518  -5.74693  -0.06672   5.69326  35.99060

(Dispersion Parameter for gaussian family taken to be 70.6117)

    Null Deviance: 9306509 on 98483 degrees of freedom
Residual Deviance: 6952924 on 98467 degrees of freedom
AIC: 698770.2

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                    Df  Sum Sq Mean Sq  F value    Pr(>F)
ns(age, 11)         11 2057916  187083 2649.465 < 2.2e-16 ***
ns(income, 1)        1  292334  292334 4140.027 < 2.2e-16 ***
ns(mortscore, 4)     4    3334     834   11.805 1.379e-09 ***
Residuals        98467 6952924      71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig: summary of gam1

```
> summary(gam2)

Call: gam(formula = LOS ~ ns(age, 11) + ns(income, 1) + ns(mortscore,
    5), data = datafile)
Deviance Residuals:
      Min        1Q    Median        3Q       Max
-33.35188  -5.74718  -0.06696   5.68893  36.12740

(Dispersion Parameter for gaussian family taken to be 70.5996)

    Null Deviance: 9306509 on 98483 degrees of freedom
Residual Deviance: 6951664 on 98466 degrees of freedom
AIC: 698754.3

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                    Df  Sum Sq Mean Sq  F value    Pr(>F)
ns(age, 11)         11 2057916  187083 2649.918 < 2.2e-16 ***
ns(income, 1)        1  292334  292334 4140.735 < 2.2e-16 ***
ns(mortscore, 5)     5    4594     919   13.014 1.094e-12 ***
Residuals        98466 6951664      71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig: summary of gam2

```
> summary(gam3)

Call: gam(formula = LOS ~ ns(age, 1) + ns(income, 1) + ns(mortscore,
    4), data = datafile)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-33.5235  -5.7605  -0.0737   5.6854  36.7737

(Dispersion Parameter for gaussian family taken to be 70.8813)

    Null Deviance: 9306509 on 98483 degrees of freedom
Residual Deviance: 6980176 on 98477 degrees of freedom
AIC: 699135.4

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                    Df  Sum Sq Mean Sq   F value    Pr(>F)
ns(age, 1)           1 2017815 2017815 28467.523 < 2.2e-16 ***
ns(income, 1)        1  305458  305458  4309.433 < 2.2e-16 ***
ns(mortscore, 4)     4    3060     765    10.793 9.566e-09 ***
Residuals        98477 6980176      71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig: summary of gam3

```
> summary(gam4)

Call: gam(formula = LOS ~ ns(age, 1) + ns(income, 1) + ns(mortscore,
    5), data = datafile)
Deviance Residuals:
      Min        1Q    Median        3Q       Max
-33.49757  -5.76668  -0.06664   5.68050  36.91604

(Dispersion Parameter for gaussian family taken to be 70.8712)

    Null Deviance: 9306509 on 98483 degrees of freedom
Residual Deviance: 6979114 on 98476 degrees of freedom
AIC: 699122.4

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                    Df  Sum Sq Mean Sq   F value    Pr(>F)
ns(age, 1)           1 2017815 2017815 28471.567 < 2.2e-16 ***
ns(income, 1)        1  305458  305458  4310.045 < 2.2e-16 ***
ns(mortscore, 5)     5    4123     825    11.634 2.924e-11 ***
Residuals        98476 6979114      71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig: summary of gam4

anova(gam1,gam2,gam3,gam4,test="F")

```
> anova(gam1,gam2,gam3,gam4,test="F")
Analysis of Deviance Table

Model 1: LOS ~ ns(age, 11) + ns(income, 1) + ns(mortscore, 4)
Model 2: LOS ~ ns(age, 11) + ns(income, 1) + ns(mortscore, 5)
Model 3: LOS ~ ns(age, 1) + ns(income, 1) + ns(mortscore, 4)
Model 4: LOS ~ ns(age, 1) + ns(income, 1) + ns(mortscore, 5)
  Resid. Df Resid. Dev  Df Deviance      F    Pr(>F)
1     98467    6952924
2     98466    6951664   1   1259.7 17.842 2.402e-05 ***
3     98477    6980176 -11 -28511.5 36.713 < 2.2e-16 ***
4     98476    6979114   1   1062.4 15.048 0.0001049 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig: ANOVA

Conclusion

For the part A, model selection, out of the four model techniques using the k-cross-fold validation approach, we can conclude that lasso model performs well with lower MSE resulting in the model which is easy to interpret.

For the part B, we could find that the F value of the model 3 is the highest from the ANOVA function:

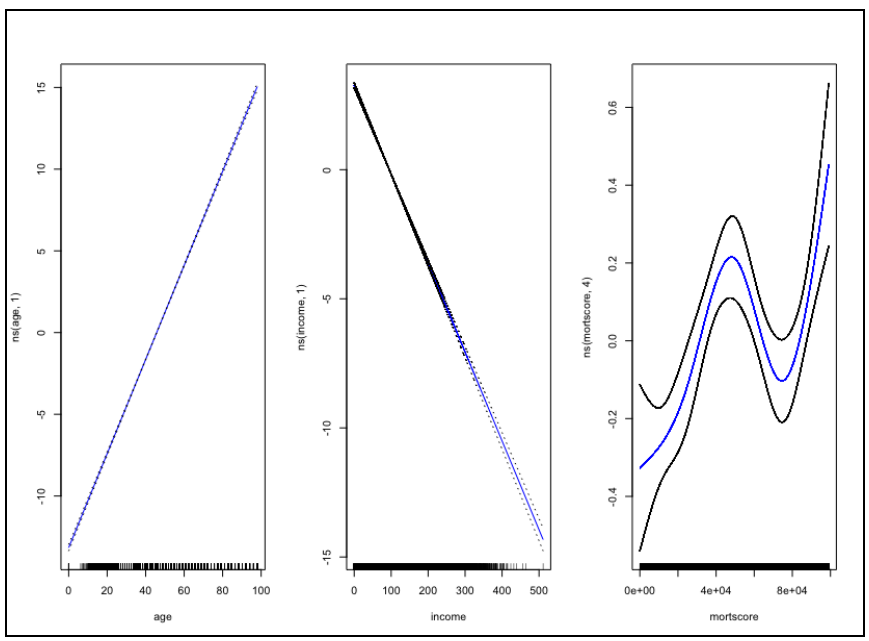


Fig: Graph of gam3

Therefore, we concluded age and income should be linear while mortscore should be cubic referring to the shape of gam3.