ML Course Project Part 1

Mayank

6/1/2018

About the data set:

- The data contain 384 features extracted from 53,500 CT images from 74 patients
- Each record characterizes a slice of an image
- Each CT slice is described by two histograms in polar space.
- The first histogram describes the location of bone structures in the image,
- the second the location of air inclusions inside of the body.
- The class output variable is numeric and denotes the relative location of the CT slice on
- the axial axis of the human body.

Data Description:

1. PatientId: Each ID identifies a different patient - each patient has multiple entries
2. From column number 2 - 241: Histogram describing bone structures
3. From column number 242 - 385: Histogram describing air inclusions
4. Column number 386: Reference: Relative location of the image on the axial axis (class value). Values are in the range [0; 180] where 0 denotes the top of the head and 180 the soles of the feet.

Problem objective: It is important to predict location of the slice using the features because when 2 or more scans are compared or in general it is necessary to navigate to a certain part of the body, the whole scan (about 1 Gb) needs to be loaded over clinical network and then the required slice is usually identified manually.

Compiling all the libraries

```
library(data.table)
library(pls)
library(factoextra)
#install.packages("relaimpo")
library(relaimpo)
suppressWarnings(library(relaimpo))
library(glmnet)
library(rpart)
library(rpart.plot)
```

Reading the data

```
datapath<-"/Users/mayank/Documents/College Documents/Q3 Courses/Machine Learning/Course Project/"
```

```
dat<-read.csv(file=paste(datapath,"slice_localization_data.csv",sep="/"))
head(colnames(dat))
```

```
## [1] "patientId" "value0"    "value1"    "value2"    "value3"    "value4"
```

```
tail(colnames(dat))
```

```
## [1] "value379"  "value380"  "value381"  "value382"  "value383"  "reference"
```

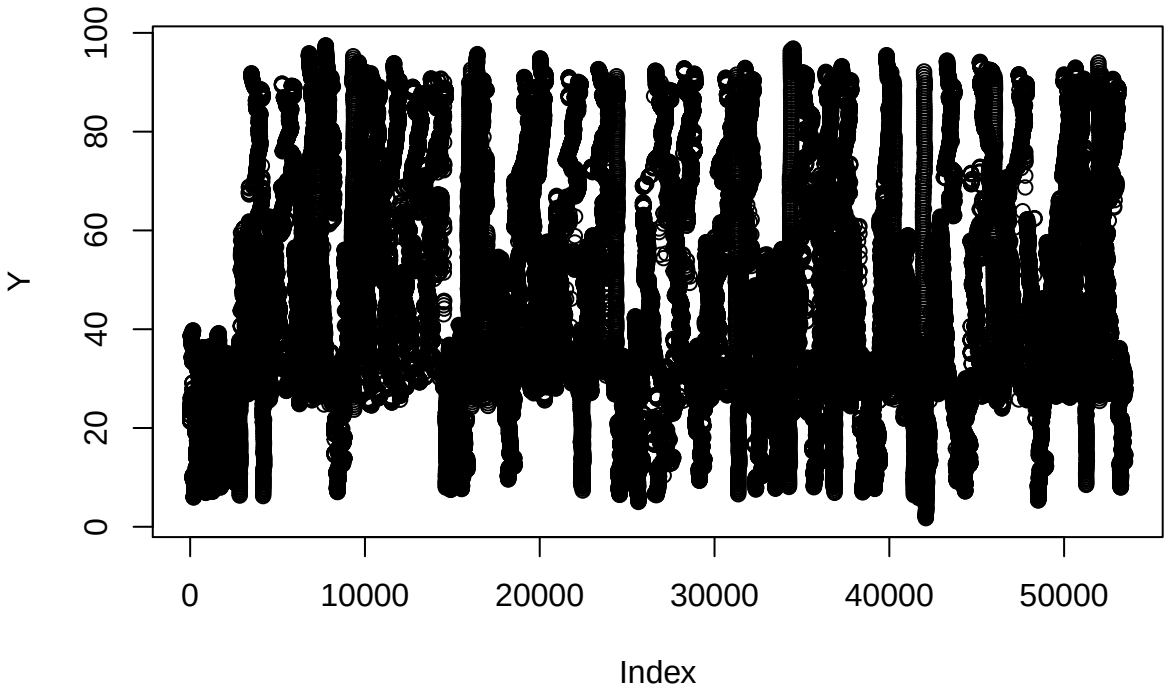
Conducting Exploratory Data Analysis

```
pred<-dat[,c(-1,-386)]
Y<-dat[,386]
head(colnames(pred))
```

```
## [1] "value0" "value1" "value2" "value3" "value4" "value5"
```

Checking the distribution of the dependent variable

```
plot(Y)
```



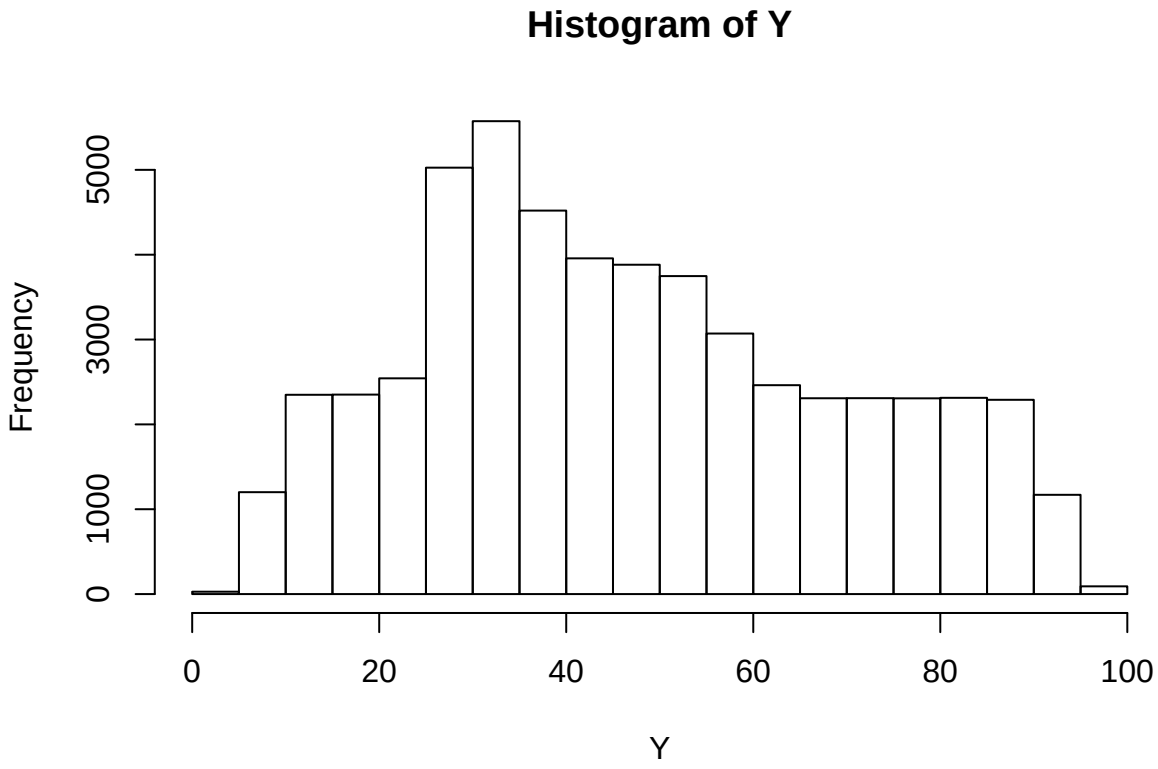
Checking the summary of Y

```
summary(Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.739  29.892  43.988  47.028  63.735  97.489
```

Checking the histogram of the independent variable

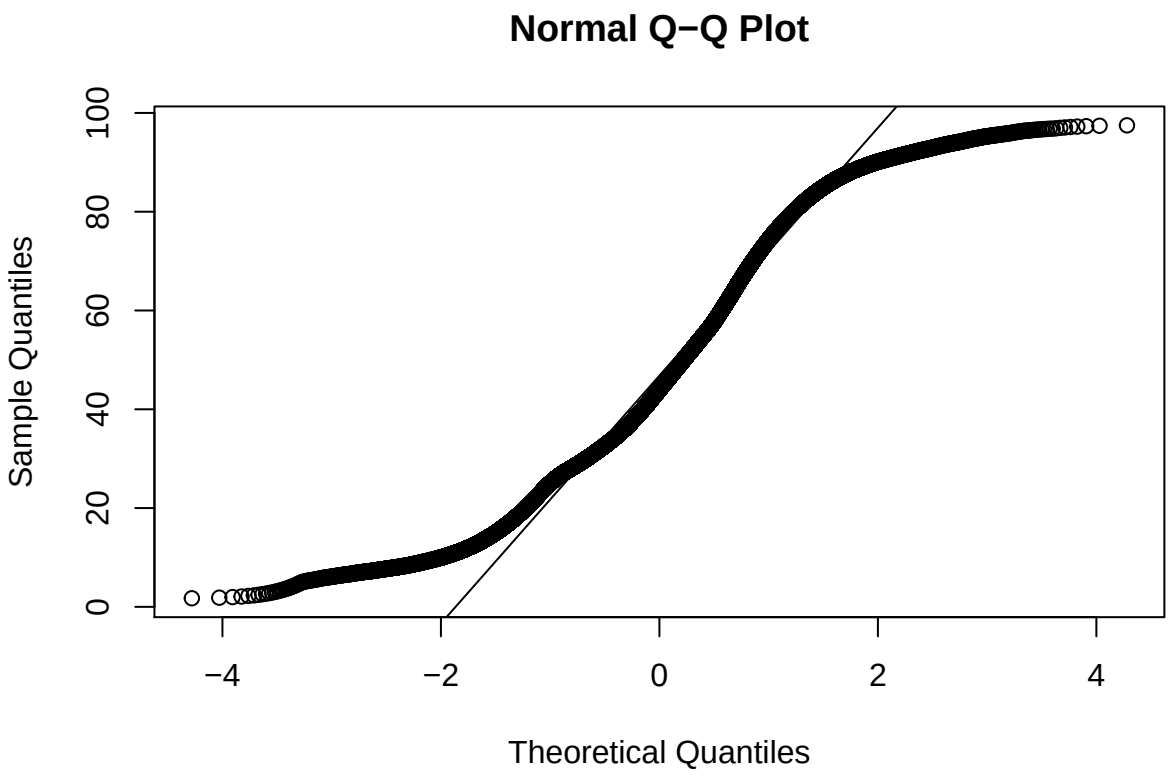
```
hist(Y)
```



The histogram in itself is representing that the distribution of the dependent variable is not gaussian, however, we check the q-q plot to ascertain it

Checking the q-q plot if the dependent variable follows gaussian distribution

```
qqnorm(Y)
qqline(Y)
```



The q-q plot ascertains our initial hypothesis of the dependent variable not following a normal distribution

As a first step for prediction we will try to fit a Linear Model. As a part of the assignment, we fit linear model with all the predictors.

```
#setting up the option to run the code on my system

lm_model<-lm(Y~., data=data.frame(Y,pred))

#creating a dataframe which has the values of all the beta coeffs and their respective p-vals

lm_beta<-data.frame(summary(lm_model)$coefficients[,1], summary(lm_model)$coefficients[,4])
lm_beta[,3]<-rownames(lm_beta)

colnames(lm_beta)<-c("coeff", "p.val", "predictor_var")

#removing the intercept value

lm_beta<-lm_beta[-1,]
lm_beta_2<-as.data.frame(lm_beta)
```

Eliminating the betas for which the p-value is greater than 5

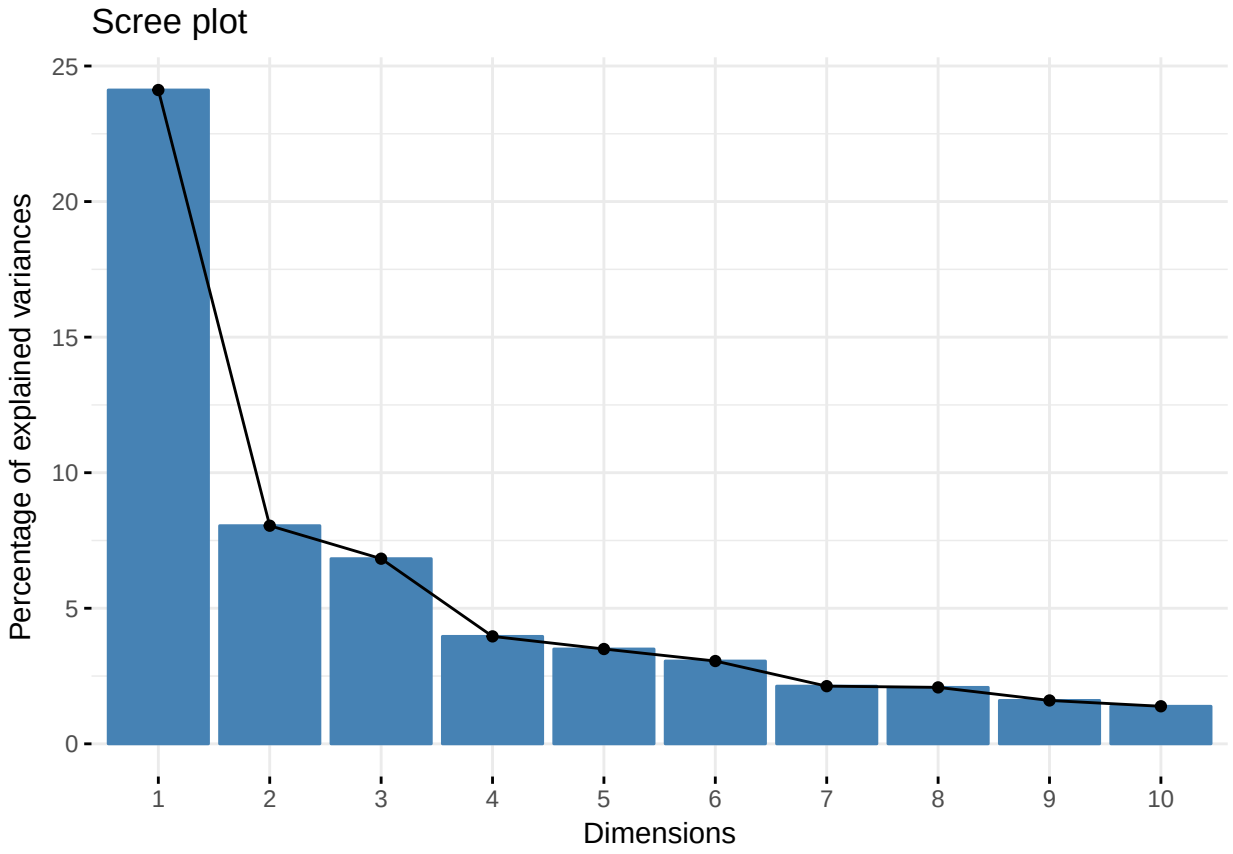
```
significant_betas<-subset(lm_beta_2, lm_beta_2$p.val<0.05)
```

```r
#subsetting the predictor list to fit the reduced model

pred_lm_reduced<-pred[,c(significant_betas[,3])]

#fitting linear regression to the reduced predictor list

reduced_lm_model<-glm(Y~., data.frame(Y, pred_lm_reduced), family = "gaussian")

#Creating vector of characteristics of the fit: AIC, R2, MSE, number of predictors in the model

R2<-1-(reduced_lm_model$deviance/reduced_lm_model$null.deviance)
MSE<-mean(reduced_lm_model$residuals^2)
(lm_fit_characteristics<-c(reduced_lm_model$aic, R2, MSE, ncol(pred_lm_reduced)))
```

```
## [1] 3.781033e+05 8.638282e-01 6.800158e+01 2.640000e+02
```

As the next steps we will be applying PCA-regression to the data as it was explained in our lecture

```r
#running pcr

#install.packages("pls")
pcr.fit<-pcr(Y~., data=data.frame(Y,pred))
#summary(pcr.fit)

#we see from the fit summary that still only 86% of the variance can be explained
#by the data set
#runnign PCA on the data set

pca<-prcomp(pred)

#installing and using the below mentioned library for visualizations
#of factors and their explained variances

#install.packages("factoextra")
fviz_eig(pca)
```



Relative Importance

Relative importance code wont work for all the factors as the amount of computation power required for it is more than that of my local machine, so we will subset the number of factors and then use the number of predictors from that list. For this we will check the initial analysis and on the basis of that decide what numnber of factors we should be proceeding forward with. Those factors need to be removed to prevent singular system error in linear model fitting process.

```
#lets check the importance of last 50 factors

summary(pca)$importance[,334:384]
```

```
##                            PC334      PC335      PC336      PC337
## Standard deviation     0.03365278 0.03305083 0.03144843 0.02987738
## Proportion of Variance 0.00003000 0.00003000 0.00003000 0.00003000
## Cumulative Proportion  0.99954000 0.99957000 0.99960000 0.99963000
##                            PC338      PC339      PC340     PC341
## Standard deviation     0.02922027 0.02847462 0.02811606 0.027752
## Proportion of Variance 0.00003000 0.00002000 0.00002000 0.000020
## Cumulative Proportion  0.99965000 0.99968000 0.99970000 0.999730
##                            PC342      PC343     PC344      PC345
## Standard deviation     0.02699929 0.02680002 0.0262389 0.02604765
## Proportion of Variance 0.00002000 0.00002000 0.0000200 0.00002000
## Cumulative Proportion  0.99975000 0.99977000 0.9997900 0.99981000
##                            PC346      PC347      PC348      PC349
## Standard deviation     0.02545333 0.02325134 0.02302374 0.02218115
## Proportion of Variance 0.00002000 0.00002000 0.00002000 0.00001000
```

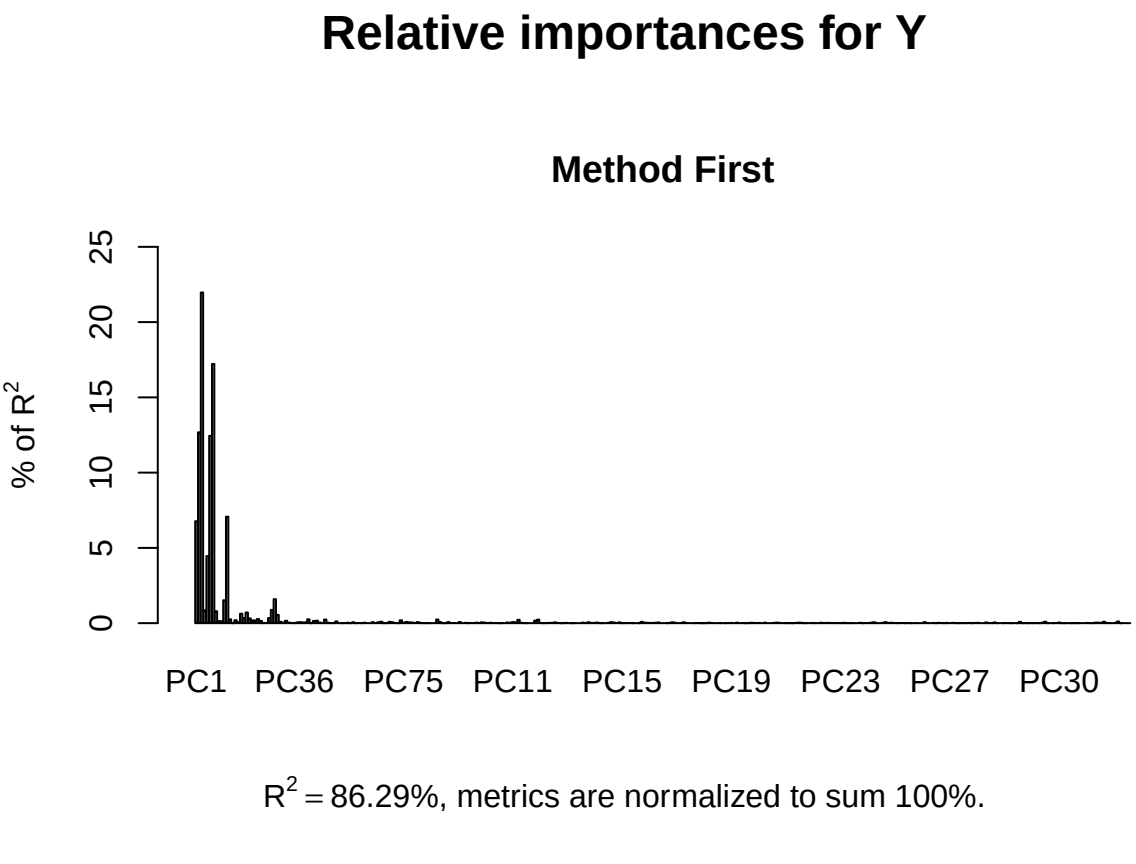
```
## Cumulative Proportion  0.99983000 0.99985000 0.99986000 0.99988000
##                             PC350      PC351     PC352      PC353
## Standard deviation     0.02188242 0.02129531 0.0212024 0.01952158
## Proportion of Variance 0.00001000 0.00001000 0.0000100 0.00001000
## Cumulative Proportion  0.99989000 0.99990000 0.9999200 0.99993000
##                             PC354      PC355     PC356     PC357
## Standard deviation     0.01868569 0.01699622 0.0156897 0.0150794
## Proportion of Variance 0.00001000 0.00001000 0.0000100 0.0000100
## Cumulative Proportion  0.99994000 0.99995000 0.9999500 0.9999600
##                             PC358      PC359     PC360      PC361
## Standard deviation     0.01423932 0.01271565 0.0119218 0.01149831
## Proportion of Variance 0.00001000 0.00000000 0.0000000 0.00000000
## Cumulative Proportion  0.99997000 0.99997000 0.9999800 0.99998000
##                             PC362      PC363      PC364       PC365
## Standard deviation     0.01146846 0.01035279 0.01021043 0.009336079
## Proportion of Variance 0.00000000 0.00000000 0.00000000 0.000000000
## Cumulative Proportion  0.99998000 0.99999000 0.99999000 0.999990000
##                              PC366       PC367       PC368      PC369
## Standard deviation     0.008548452 0.007291049 0.005852435 0.00508328
## Proportion of Variance 0.000000000 0.000000000 0.000000000 0.00000000
## Cumulative Proportion  1.000000000 1.000000000 1.000000000 1.00000000
##                             PC370       PC371       PC372       PC373
## Standard deviation     0.004140269 0.002958004 0.001562772 0.001073468
## Proportion of Variance 0.000000000 0.000000000 0.000000000 0.000000000
## Cumulative Proportion  1.000000000 1.000000000 1.000000000 1.000000000
##                               PC374        PC375        PC376        PC377
## Standard deviation     1.261121e-15 9.271477e-16 2.034379e-16 2.034379e-16
## Proportion of Variance 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##                               PC378        PC379        PC380        PC381
## Standard deviation     2.034379e-16 2.034379e-16 2.034379e-16 2.034379e-16
## Proportion of Variance 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##                               PC382        PC383        PC384
## Standard deviation     2.034379e-16 2.034379e-16 1.112165e-16
## Proportion of Variance 0.000000e+00 0.000000e+00 0.000000e+00
## Cumulative Proportion  1.000000e+00 1.000000e+00 1.000000e+00
```

```r
#segregating the first 334 factors for our analysis

num_factors<-334

#in case of prcomp the rotation is the factors loadings, i.e. they are the eigen vectors
#in case of prcomp the variable "x" contains the factor scores values, i.e. the
#coordinates of the individuals (observations) on the principal components.

factor_loading<-as.matrix(pca$rotation[,1:num_factors])
factor_scores<-as.matrix(pred)%*%factor_loading

#creating a dataframe with the above segregated number of factors

pca_filtered_factors<-data.frame(Y,factor_scores)
```

We see that how gradually the value of explained variance takes a toll. It is almost an exponential trend using all the factors as created below does not yield any results due inverse matrix calculation error

```
new_vars<-as.data.frame(pca$scores)

#running linear regression on pc comps
```

```
lm.pca<-lm(Y~., data=pca_filtered_factors)


#on the basis of the results depicted above I will go with 334 factors only

rel.imp2<-calc.relimp(lm.pca, type = "first", rela=TRUE)
plot(rel.imp2)
```



Relative importance measure "first" makes perfect decomposition of R^2. It is also mentioned in the graph above. Now, I will reorder the PCA factors according to their relative importance as predictors explaining the output and calculate the R^2 sequence.

```
ordered.PC<-rel.imp2@first.rank
ordered.PC
```

```
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11  PC12
##     6     3     1    11     7     4     2    12    35    36     9     5
##  PC13  PC14  PC15  PC16  PC17  PC18  PC19  PC20  PC21  PC22  PC23  PC24
##    21   246    26    60    14    16    13    18    28    29    19    34
```

```
##  PC25  PC26  PC27  PC28  PC29  PC30  PC31  PC32  PC33  PC34  PC35  PC36
##   227   144    17    10     8    15    42   106    32    82   135   225
##  PC37  PC38  PC39  PC40  PC41  PC42  PC43  PC44  PC45  PC46  PC47  PC48
##    74    56    75    70    20   254    33    31    78   206    23   102
##  PC49  PC50  PC51  PC52  PC53  PC54  PC55  PC56  PC57  PC58  PC59  PC60
##   193   215    37   287   261   169    97   273    57   231   156   220
##  PC61  PC62  PC63  PC64  PC65  PC66  PC67  PC68  PC69  PC70  PC71  PC72
##    98   292   323    54   158    66    39   252   200    43    61   142
##  PC73  PC74  PC75  PC76  PC77  PC78  PC79  PC80  PC81  PC82  PC83  PC84
##   267    27    95    46    68    77   162    49   123   207   242   224
##  PC85  PC86  PC87  PC88  PC89  PC90  PC91  PC92  PC93  PC94  PC95  PC96
##   290   318    22    47   272   236    51   320   164   306    48   286
##  PC97  PC98  PC99 PC100 PC101 PC102 PC103 PC104 PC105 PC106 PC107 PC108
##   125   205   155   201    90   150    67    93   203   105   198   159
## PC109 PC110 PC111 PC112 PC113 PC114 PC115 PC116 PC117 PC118 PC119 PC120
##   276   279   304    79   111    50    64    25   100   234   216   326
## PC121 PC122 PC123 PC124 PC125 PC126 PC127 PC128 PC129 PC130 PC131 PC132
##   284    30    24   154   255   183   126   140    69   134   268   259
## PC133 PC134 PC135 PC136 PC137 PC138 PC139 PC140 PC141 PC142 PC143 PC144
##   127   305   251   221   301   325    84   175    58   121   138    81
## PC145 PC146 PC147 PC148 PC149 PC150 PC151 PC152 PC153 PC154 PC155 PC156
##   217   260   283   122    55    76   157    65   262   202   192   161
## PC157 PC158 PC159 PC160 PC161 PC162 PC163 PC164 PC165 PC166 PC167 PC168
##   228   291   281    45    88   108   148   130   104    73   322   312
## PC169 PC170 PC171 PC172 PC173 PC174 PC175 PC176 PC177 PC178 PC179 PC180
##   241   115    63   110   315   191    59   163   327   324   223   153
## PC181 PC182 PC183 PC184 PC185 PC186 PC187 PC188 PC189 PC190 PC191 PC192
##   257   266   243    87   165   316   314   248   313   237   189   137
## PC193 PC194 PC195 PC196 PC197 PC198 PC199 PC200 PC201 PC202 PC203 PC204
##   297    94   293   185   256   239   103   131   172   139   295    85
## PC205 PC206 PC207 PC208 PC209 PC210 PC211 PC212 PC213 PC214 PC215 PC216
##   311   303   136    83   113   299   210   176   265   277   143    89
## PC217 PC218 PC219 PC220 PC221 PC222 PC223 PC224 PC225 PC226 PC227 PC228
##   112   253   196   310   145   219   308    99   170   132   129   171
## PC229 PC230 PC231 PC232 PC233 PC234 PC235 PC236 PC237 PC238 PC239 PC240
##   181   179   197   114   229   199   213   298   296   109   232   166
## PC241 PC242 PC243 PC244 PC245 PC246 PC247 PC248 PC249 PC250 PC251 PC252
##   186    91    62   211   300   119    53   141   101   151   209   152
## PC253 PC254 PC255 PC256 PC257 PC258 PC259 PC260 PC261 PC262 PC263 PC264
##   167   275   182   250   334   190   302   309    52   187   289   147
## PC265 PC266 PC267 PC268 PC269 PC270 PC271 PC272 PC273 PC274 PC275 PC276
##   233   116   184   230   124   329   118   180   258   177   188   214
## PC277 PC278 PC279 PC280 PC281 PC282 PC283 PC284 PC285 PC286 PC287 PC288
##   133   264   149   107   317   319    72   168   194    71   245   333
## PC289 PC290 PC291 PC292 PC293 PC294 PC295 PC296 PC297 PC298 PC299 PC300
##   235   160   204   271   331   238    44   270   173   263   195   226
## PC301 PC302 PC303 PC304 PC305 PC306 PC307 PC308 PC309 PC310 PC311 PC312
##   174   280    92    40   274   321   244   178    80   269   282   285
## PC313 PC314 PC315 PC316 PC317 PC318 PC319 PC320 PC321 PC322 PC323 PC324
##   247   249   222   240   332   288   146   212   208    86    96   117
## PC325 PC326 PC327 PC328 PC329 PC330 PC331 PC332 PC333 PC334
##    41   128   218   294   120    38   278   328   330   307
```

We will try to use this ordering and then as per it we will re-run out regression analysis as per the ordering segregating the loadings and factor scores as we did that earlier, but this time as per the ordering of the components as per their importance.

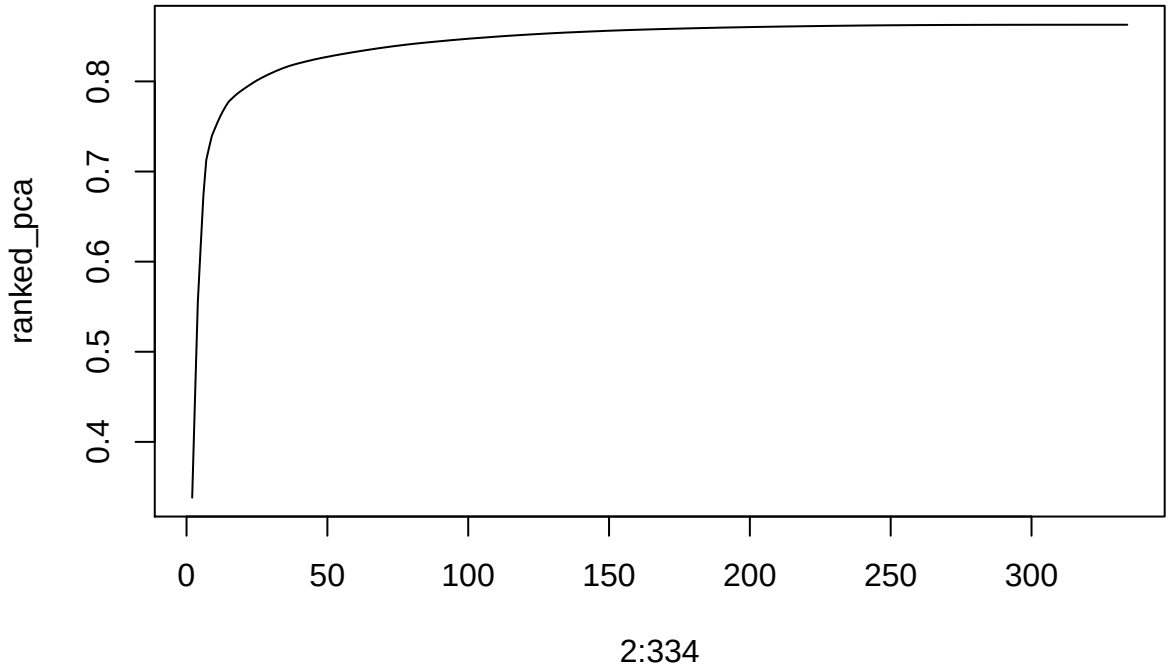
```
ranked_factors<-factor_scores[,order(ordered.PC)]
ranked_loadings<-factor_loading[,order(ordered.PC)]
head(colnames(ranked_factors))
```

```
## [1] "PC3"  "PC7"  "PC2"  "PC6"  "PC12" "PC1"
```

```
tail(colnames(ranked_factors))
```

```
## [1] "PC270" "PC333" "PC293" "PC317" "PC288" "PC257"
```

```
#creating a chart to check how the explained variance behaves when we use the
#ordered principal components

ranked_pca<-sapply(2:334,function(z)
  summary(lm(Y~.,data=data.frame(Y=dat[,386],ranked_factors[,1:z])))$r.squared)
plot(2:334,ranked_pca,type="l")
```



We see that the curve kind of diminishes or does not add any sufficient information after the first 150 principal components, we will now try to understand and echouate the optimal cur-off point for our analysis.

```
data_filter_2<-rbind(R2=c(0,ranked_pca),ranked_loadings)
head(data_filter_2[,1:10])
```

```
##                 PC3          PC7         PC2         PC6         PC12
## R2      0.000000000  0.338127186 0.447535590 0.554891775  0.61591463
## value0 -0.012618847 -0.066281698 0.009655732 0.006242987  0.06083645
## value1 -0.007778977 -0.040263968 0.016235972 0.058333219  0.04443881
## value2 -0.004028579 -0.007601155 0.044500741 0.145563901  0.04468455
```

```
## value3  0.018246320  0.037331782 0.071189186 0.183732635 -0.02771766
## value4 -0.022398739  0.170039785 0.027584180 0.051956730 -0.06975304
##                PC1          PC5          PC29         PC11         PC28
## R2      0.67433772  0.712776067  0.7265198776  0.739670399  0.747259880
## value0 0.01170951 -0.013990495  0.0005734488 -0.005540102  0.001421191
## value1 0.01613695 -0.002753690  0.0108987779 -0.028338290 -0.032390983
## value2 0.03596277 -0.006361954 -0.0092054289 -0.101541722 -0.064938327
## value3 0.04919846 -0.001902797  0.0816528748 -0.066008417 -0.024916503
## value4 0.04669315  0.053268944  0.0182709640  0.049926590  0.026043405
```

```
#Now we will look for the maximum level of accuracy we can reach for with the given set of ordered
#principal components

max(data_filter_2[1,])
```

```
## [1] 0.8628812
```

We see that the maximum accuracy level which we can achieve is 86%, thus now we will try to find the number of principal components requried to achieve that accuracy

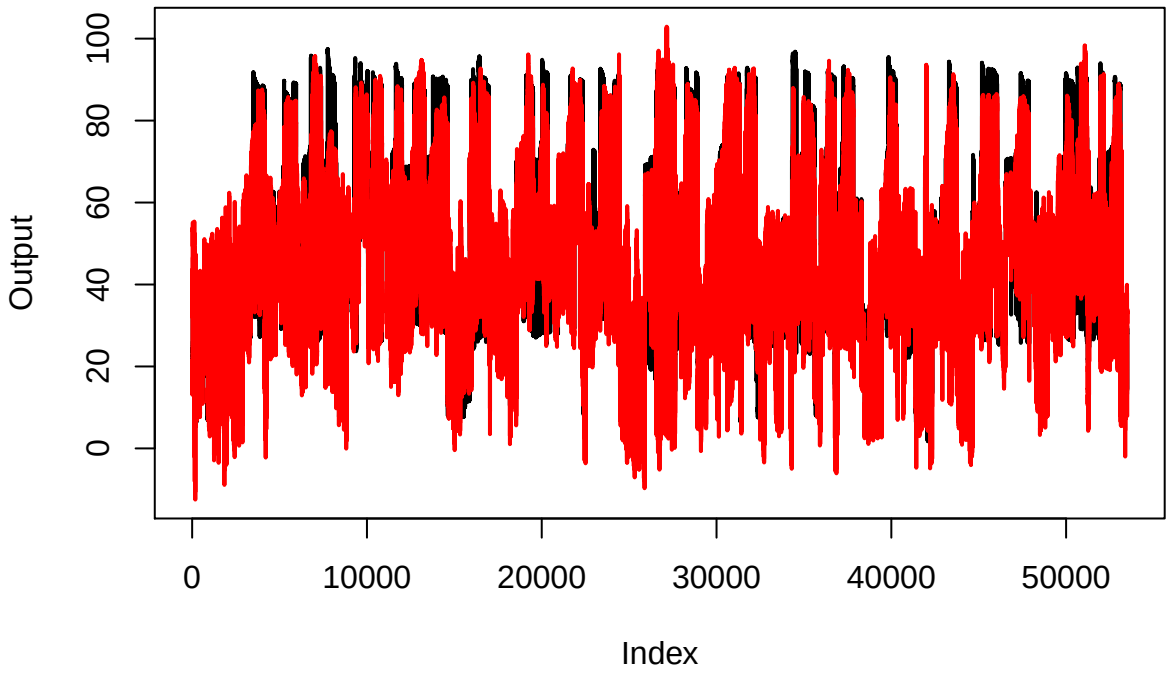
```
accuracy_level<-0.86
level_86_perc<-(data_filter_2[1,]>=accuracy_level)*(1:length(data_filter_2[1,]))
(num_factors_2<-min(level_86_perc[level_86_perc>0]))
```

```
## [1] 196
```

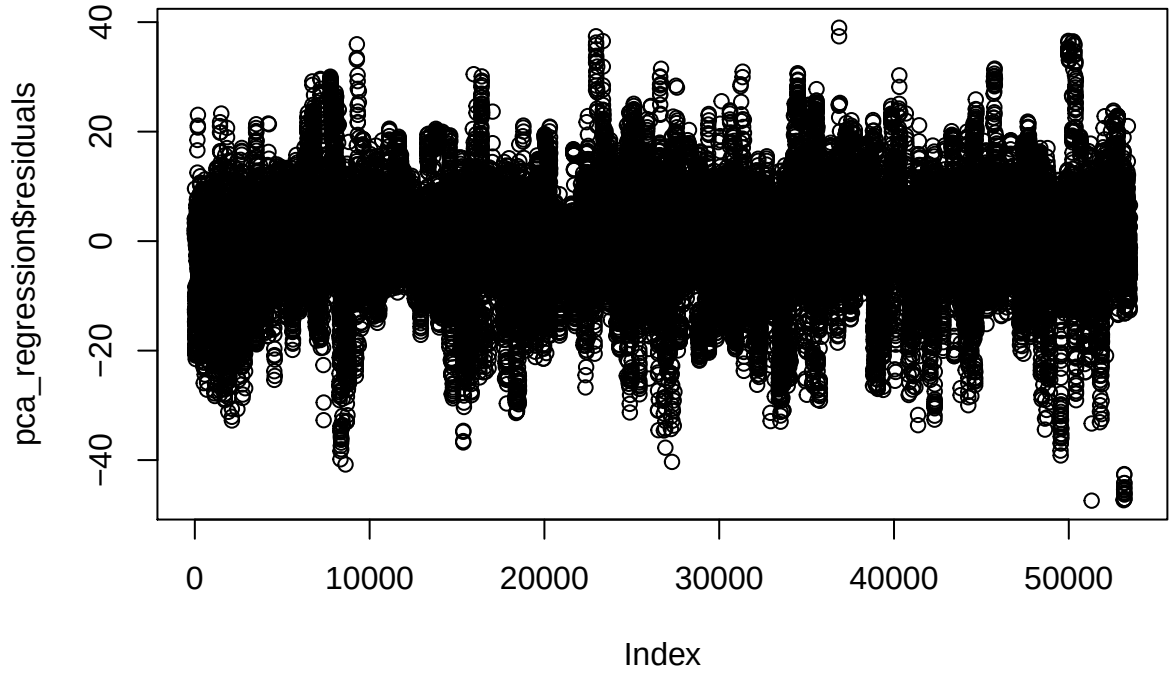
```
#thus we will now use only 196 ranked principal components to run our
#analysis and linear regression model again

final_pca_data<-cbind(Y,ranked_factors)
regression_data_pca<-as.data.frame(final_pca_data[,1:(num_factors_2+1)])
pca_regression<-lm(Y~.,data=regression_data_pca)

#Create vector of fit characteristics.

(pca_fit<-c(AIC=AIC(pca_regression),
            R2=summary(pca_regression)$r.squared,
            MSE=mean(pca_regression$residuals^2),
            nSlopes=pca_regression$rank-1))
```

```
##          AIC           R2          MSE      nSlopes
## 3.794285e+05 8.600580e-01 6.988438e+01 1.960000e+02
```

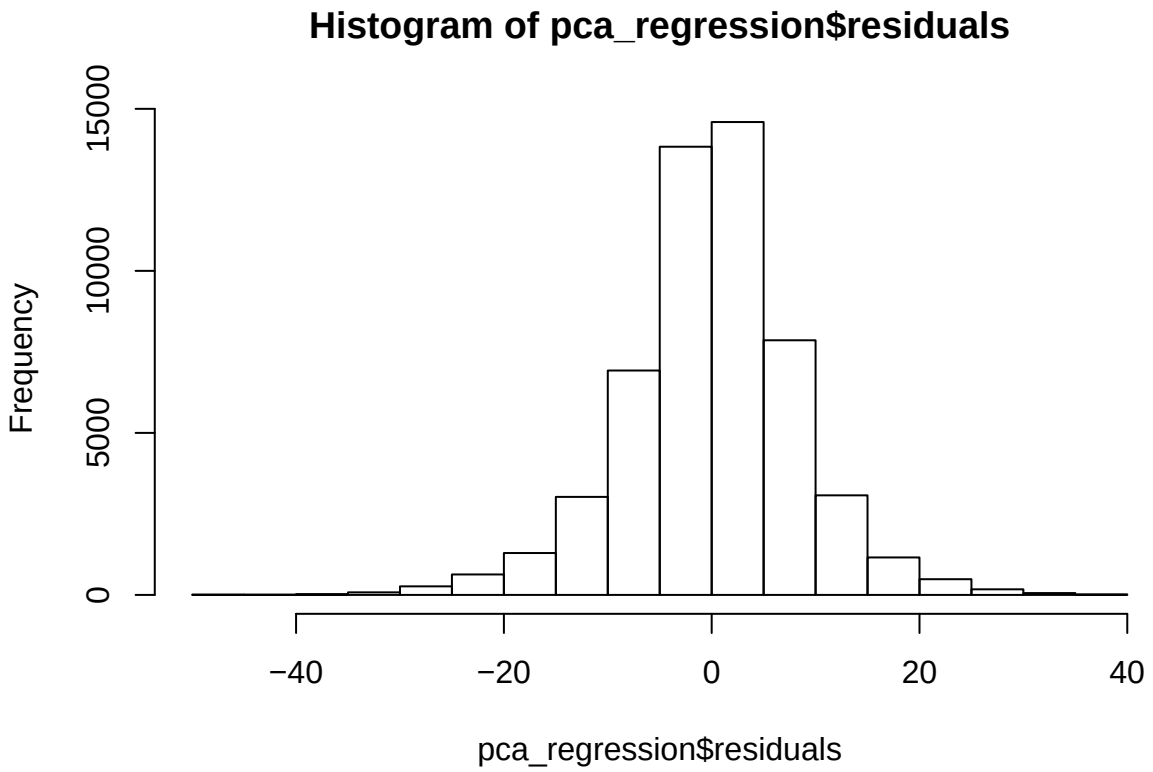
```
#Checking the predictions and the actual values

matplot(1:length(final_pca_data[,1]),cbind(final_pca_data[,1],pca_regression$fitted.values),type="l",
        lwd=2,lty=1,ylab="Output",xlab="Index")
```



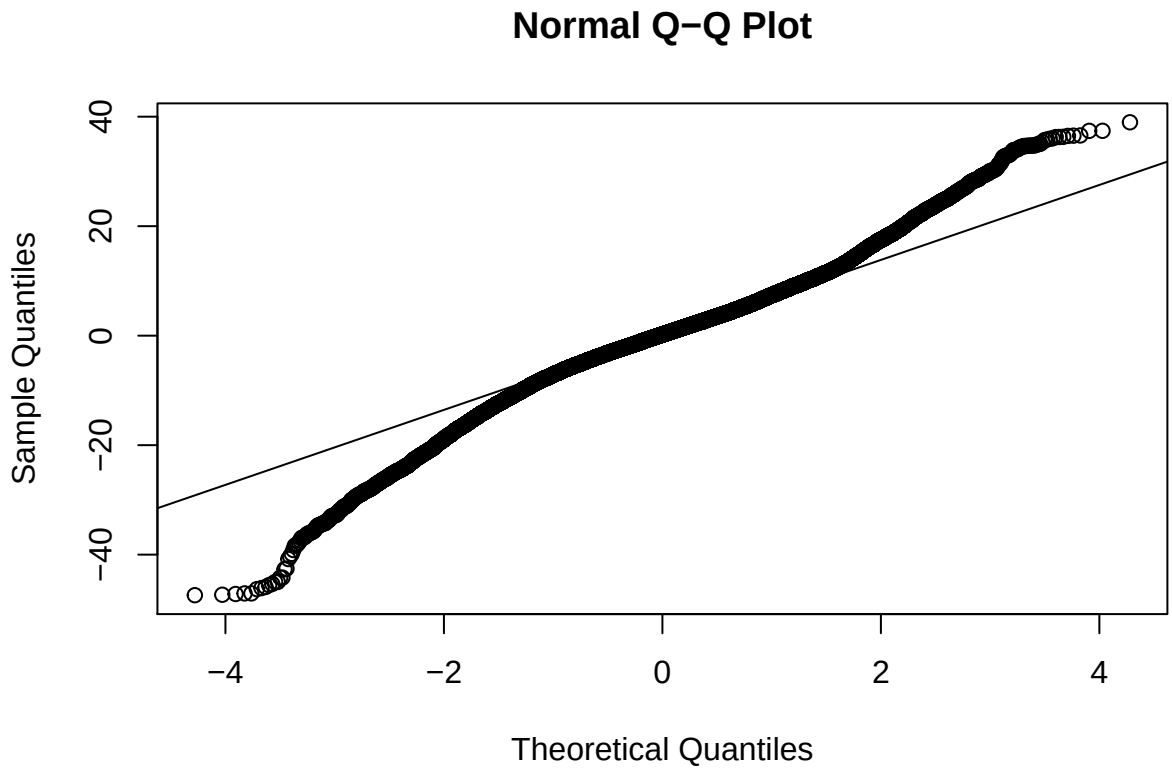

```
#checking the residuals

plot(pca_regression$residuals)
```




```
#checking the histogram of the residuals

hist(pca_regression$residuals)
```

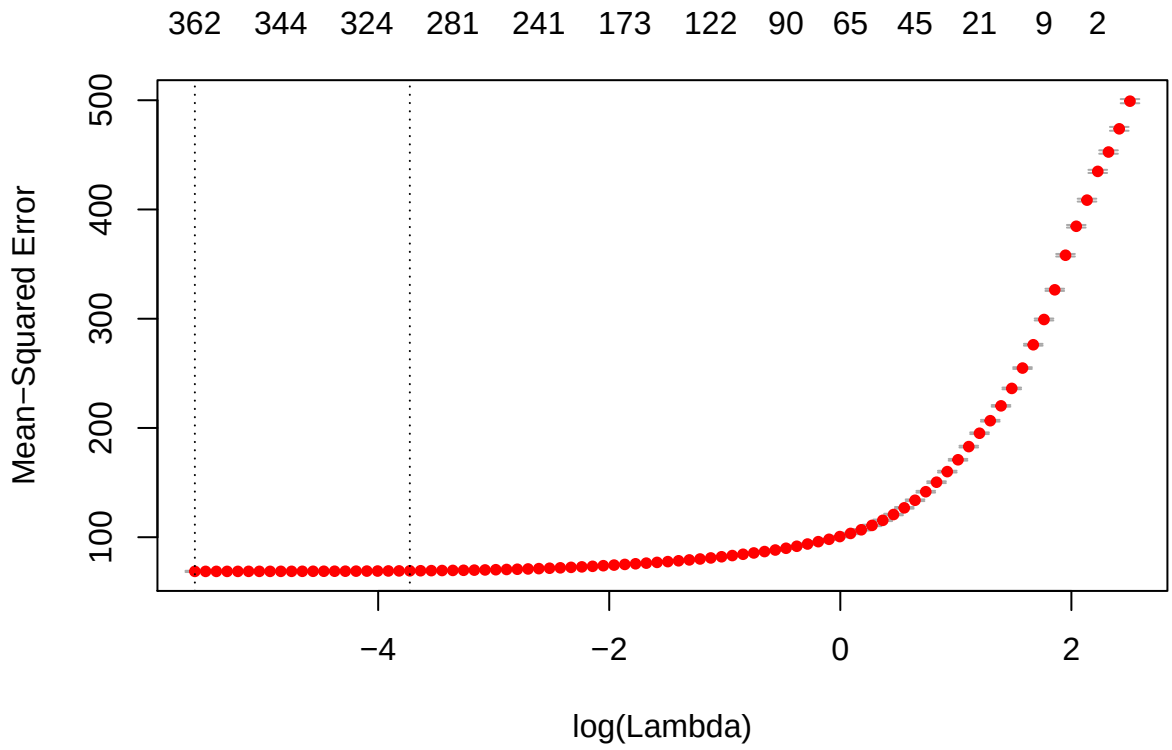



```r
#checking the Q-Q plots for residuals

qqnorm(pca_regression$residuals)
qqline(pca_regression$residuals)
```



Next we will be applying the lasso regression method on the data set

```
set.seed(1)

#running cross validation in-order to obtain the best lambda value

cv_out=cv.glmnet(x=as.matrix(dat[,c(-1,-386)]),y=as.matrix(dat[,386]),alpha=1)
plot(cv_out)
```

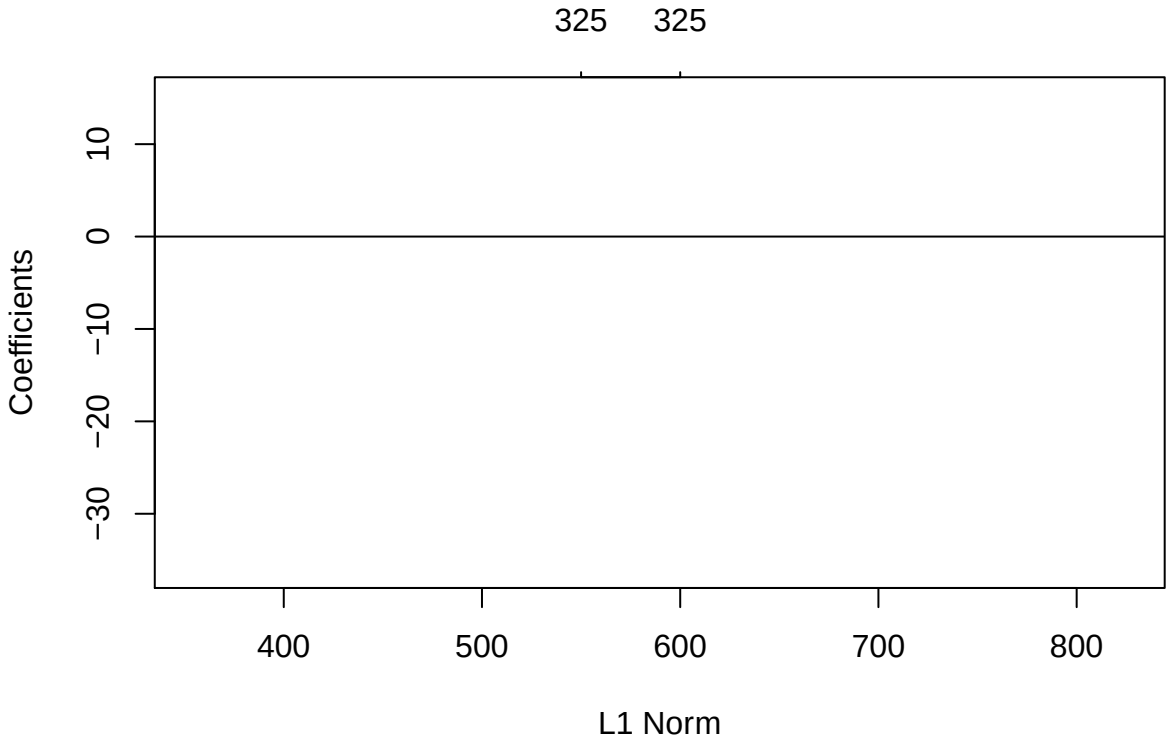



```
#the best lambda

(best_lam =cv_out$lambda.min)
```

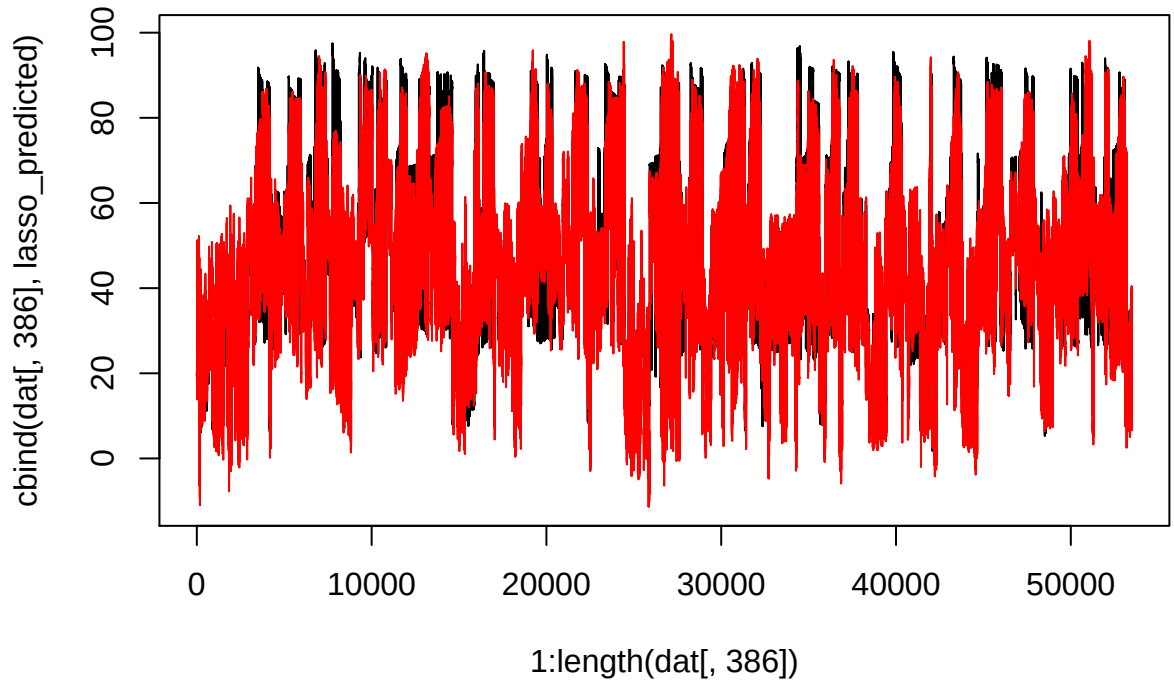
```
## [1] 0.003747753
```

```
#running the lasso regression model with the best value of lambda

lasso_model<-glmnet(x=as.matrix(dat[,c(-1,-386)]),y=as.matrix(dat[,386]), alpha = 1,
                    lambda = best_lam, standardize = F)

#Extracting all the betas in a data frame

lasso_beta<-as.data.frame(lasso_model$beta[,1])
colnames(lasso_beta)<-c("coeff_val")
lasso_beta$variable<-rownames(lasso_beta)


eliminated.preds<-subset(lasso_beta, lasso_beta$coeff_val==0)

# we see from the above analysis that 62 obs have been eliminated from our analysis
```

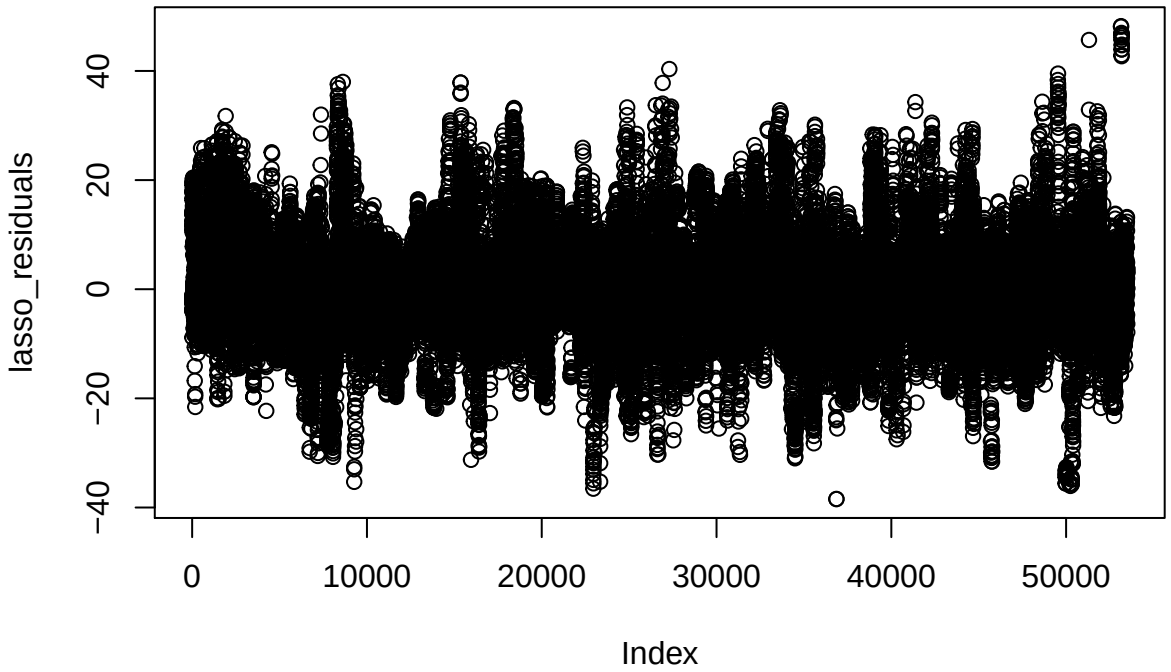
```r
#checking the plot of all the used variables in our table
#currently this plotis not working, check why

plot(lasso_model,lwd=1)
abline(h=0)
```




```r
# checking the plot of the predicted vs original outputs and how well or bad our model performed

lasso_predicted<-predict(lasso_model,type="response",newx=as.matrix(pred),s=best_lam)
matplot(1:length(dat[,386]),cbind(dat[,386],lasso_predicted),type="l",lty=1)
```




```
#checking the behavior of the residuals now

lasso_residuals<-lasso_predicted-dat[,386]
plot(lasso_residuals)
```




```
#finally running a linear regression model using the variables which
#havent been eliminated through lasso

lasso_pred_set<-pred[,!names(pred) %in% (eliminated.preds$variable)]
lasso_lm<-lm(Y~.,data=data.frame(cbind(Y=Y,as.matrix(lasso_pred_set))))
```

```
#Create vector of fit characteristics

(lasso_fit<-c(AIC=AIC(lasso_lm),
              R2=summary(lasso_lm)$r.squared,
              MSE=mean(lasso_lm$residuals^2),
              nSlopes=lasso_lm$rank-1))
```

```
##          AIC           R2          MSE      nSlopes
## 3.781200e+05 8.640757e-01 6.787800e+01 3.210000e+02
```

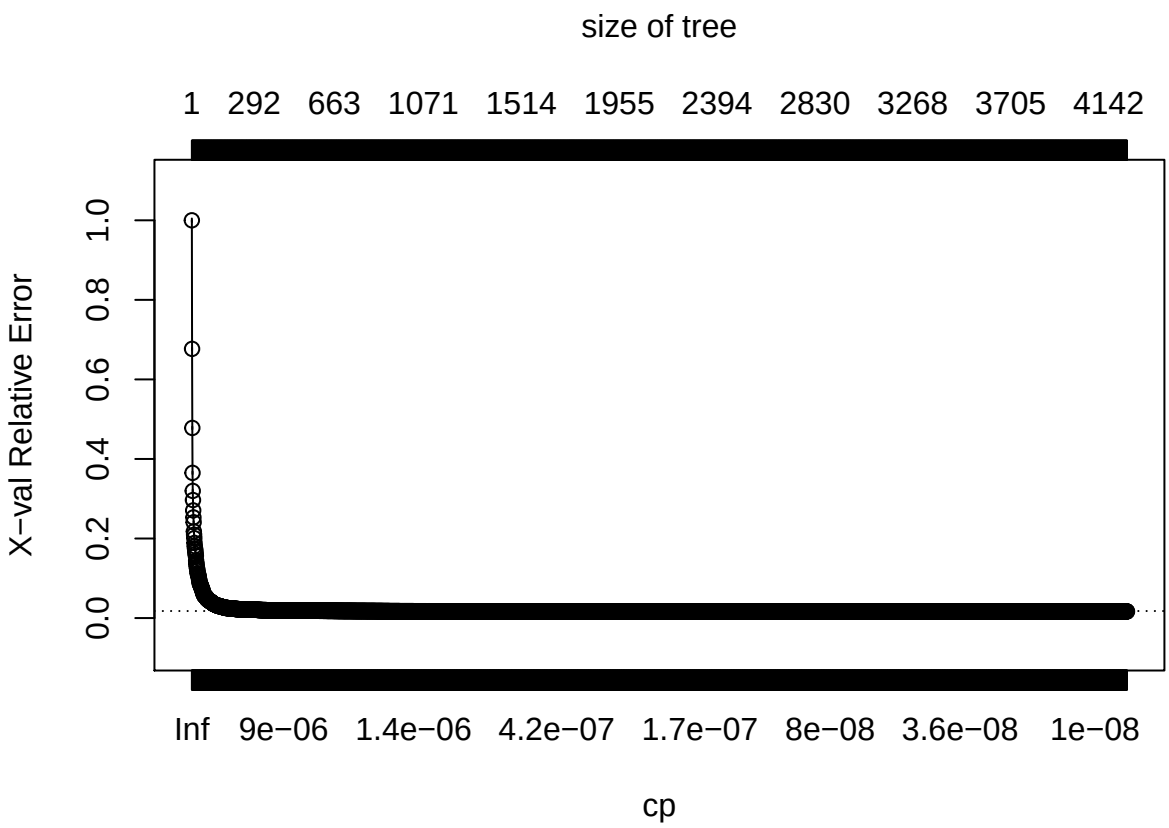
Next we will use the regression tree method on the above data set to estimate its predictive quality using 10 fold cross validation

```
#estimate its predictive quality using 10 fold cross validation
#rpart uses CV of 10 by default thus we dont need to set up that option anywhere
#create a vector of characteristics r.square, MSE, number of predictors in the model
```

Method is used to pass the splitting rule for each node. In the code below we are using method="anova" because we want to do linear regression. The other options are

- method= "Class", it is used for categorical data
- method="Poisson", it is used for counts or processes
- method= "Exp", it is used for survival modeling

```
set.seed(1)

FullTree <- rpart(Y ~ ., data=data.frame(Y,pred), method="anova", control = rpart.control(cp = 0))

#visualizing the tree

plotcp(FullTree)
```



Check out the cp table

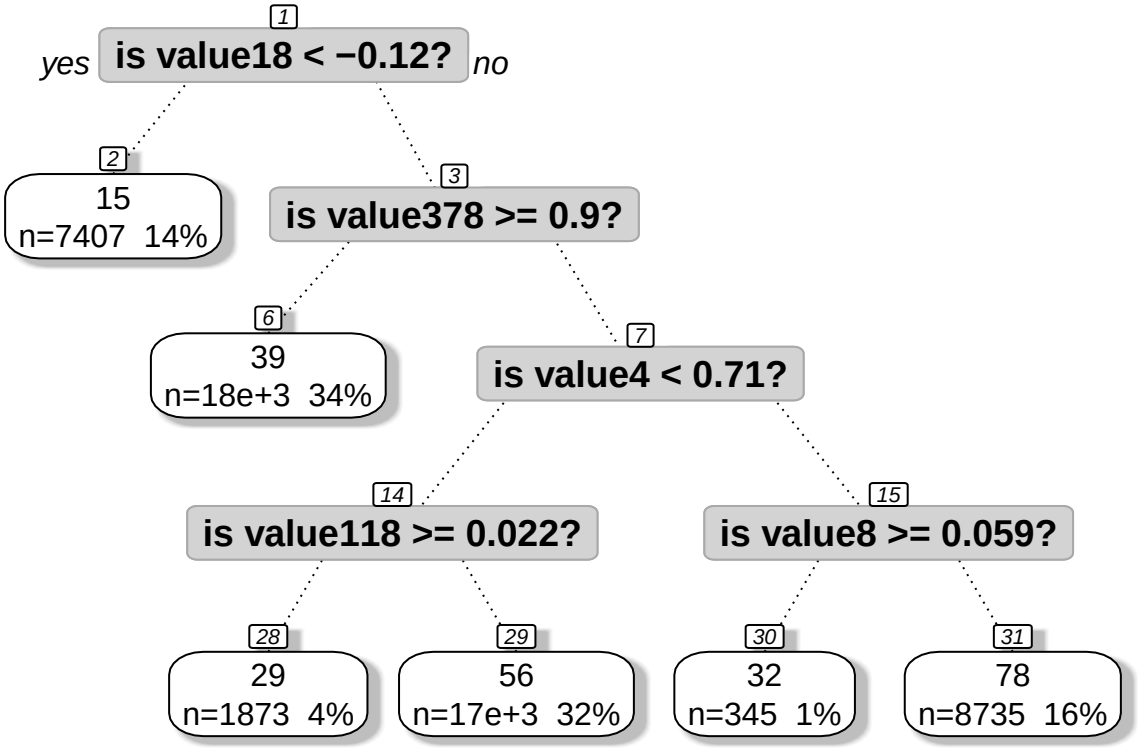
A rule of thumb is that tree needs to be pruned at a minimum level where rel error + xstd < xerror

```
#I am not showing it here as the table is pageS long
FullTree$cptable
```

```
#store the optimal value of the complexity parameter, so that we can use it
#to prune the tree later on
#also, not showing it here as it is pageS long

cpbest <- FullTree$cptable[6,1]
```

```
#fit the model with this best cp parameter for pruning

set.seed(1)
PrunedTree <- rpart(Y ~ ., data=data.frame(Y,pred), method="anova",
                    control = rpart.control(cp = cpbest))

#visualizing the tree

#install.packages("rpart.plot")
prp(PrunedTree,extra=101, # display the number of observations that fall in the node
    branch=.5, # change angle of branch lines
    shadow.col="gray", # shadows under the leaves
    branch.lty=3, # draw branches using dotted lines
    split.cex=1.2, # make the split text larger than the node text
    split.prefix="is ", # put "is " before split text
```

```
    split.suffix="?", # put "?" after split text
    split.box.col="lightgray", # lightgray split boxes (default is white)
    split.border.col="darkgray", # darkgray border on split boxes
    split.round=.5,
    nn=TRUE) # display the node numbers, default is FALSE
```



Analyze the tree. Which factors are the most important for size of compensation? Are there any questionable results of the tree model? Tree with too small terminals is unreliable and misleading. This tree is too deep. Analyze the depth of the tree. How many nodes are necessary to keep the tree stable and still accurate? Decision about appropriate depth of the tree is made based on parameter CP and the error columns.

```
#calculating the parameter values of this model
#calculating the rmse of this model

rmse <- function(x) sqrt(mean(x^2))
tree_Rmse <- rmse(resid(PrunedTree))
tree_Rmse
```

```
## [1] 12.08206
```

```
#calculating the MSE of this model

RSS<-sum(resid(PrunedTree)^2)
MSE<-RSS/nrow(pred)
R2<-1-(RSS/(sum(Y-mean(Y)^2)))

#Look at the residuals from this model, just as with a regular linear regression fit
```

```
#plot(predict(PrunedTree), jitter(resid(PrunedTree)))
#temp <- PrunedTree$frame[PrunedTree$frame$var == '<leaf>',]
#axis(3, at = temp$yval, as.character(row.names(temp)))
#mtext('leaf number', side = 3, line = 3)
#abline(h = 0, lty = 2)

#check out if there are any outliers in any of these trees and how many trees have outliers

#final parameters list
(tree_fit<-c(AIC=RSS, R2=R2, MSE=RSS/nrow(pred), nSlopes=NaN))
```

```
##          AIC           R2          MSE      nSlopes
## 7.809726e+06 1.067438e+00 1.459762e+02          NaN
```

FINALLY COMPARE THE FIT OF ALL THE METHODS USED ABOVE

```
#create a vector of characteristics r.square, MSE, number of predictors in the model
rbind(LM=lm_fit_characteristics,PCA=pca_fit,LASSO=lasso_fit,Tree=tree_fit)
```

```
##             AIC        R2       MSE nSlopes
## LM     378103.3 0.8638282  68.00158     264
## PCA    379428.5 0.8600580  69.88438     196
## LASSO  378120.0 0.8640757  67.87800     321
## Tree  7809725.8 1.0674377 145.97618     NaN
```