Analysis of High Dimensional Data

Wilson Tendong, Luis CampoverdeReinoso, Omkar Kulkarni

18 May 2016

PART II

NB: The data exploration could be found on a separate document (Part I) and codes included in Rmarkdown script.

All predictive model building was done using the data containing the relative frequency counts of the microbiomes (OTUTableRel.RData) as its row sums equal to 1. Moreover it is mentioned that it is biologically more relevant for prediction.

2) Prediction Model : Principal component regression (PCR)

Principal component regression is a multivariate regression method usually employed in high dimensional data analysis; situations where there is relatively fewer number of observations (n) compared to predictor variables (p). This technique also adjust for the effect of multicollinearity of predictors which may lead to biased variances of least square estimates.

In performing PCR, the dimensionality of the data set is reduced and the independent variables are transformed into their principal components. Each principal component is independent from the others due to their orthogonal property, hence zero correlation with each other. Note that in PCR, the response variable is regress on the PCAs which contain information on the variability in the independent variables alone. Therefore there is no guarantee of having a good prediction model.

In this exercise, our aim is to construct a predictive model using PCR that would allows prediction of the age of a child given his/her microbiome composition. The 'OTUTableRel.RData' data set would be used.

```
# read data
load("OTUTable.RData") # data in OTUTable
load("OTUTableRel.RData") # data in OTUTableRel
Data1 <- OTUTable
Data2 <- OTUTableRel
library(PMA)
library(glmnet)
library(MASS)
library(nsprcomp)
library(boot)
library(knitr)
```

Model building

Data was split into 70% training data and 30% test and both dependent (Age of child) and independent (microbiome) variables standardized. The training data was used for model building while model validation was done on the test data. The objective is to have a final model that minimizes prediction mean square error (MSE).

```
########################################
### Principal component regression    #
########################################
####
#a) Model building
####
#dim(Data2)
#dim(Data1)
varY <- Data2[,2241] #Extracting the response variable T1D and Age
varX <- Data2[,-(2240:2242)]
set.seed(2)
#Selecting 70% of the data as straining data set
trainID <- sample(dim(Data2)[1],ceiling(0.7*777))
#Training data set
trainY <- scale(varY)[trainID]
trainX <- scale(varX)[trainID,]
#30% Validation data set (test data set)
testY <- scale(varY)[-trainID]
testX <- scale(varX)[-trainID,]
#round(colMeans(trainX))
#round(mean(trainY))
```

Cost function

In order to estimate the prediction MSE, we need a cost function.

```
#Cost functions for estimation of MSE
MSE=function(observedY,predictedY){
  n=length(observedY)
  MSE=(sum((observedY-predictedY)^2))/n
  return(MSE)
}
```

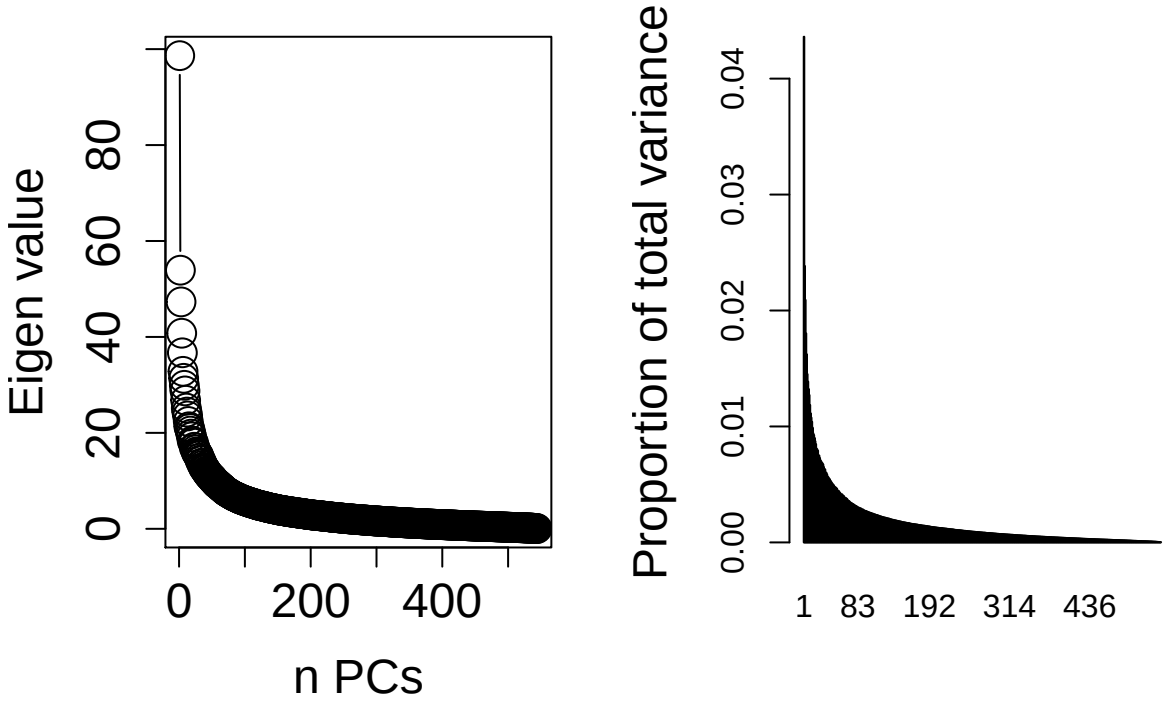
PCA analysis

The PCAs were obtained from the singular value decomposition (SVD) of the independent variables.

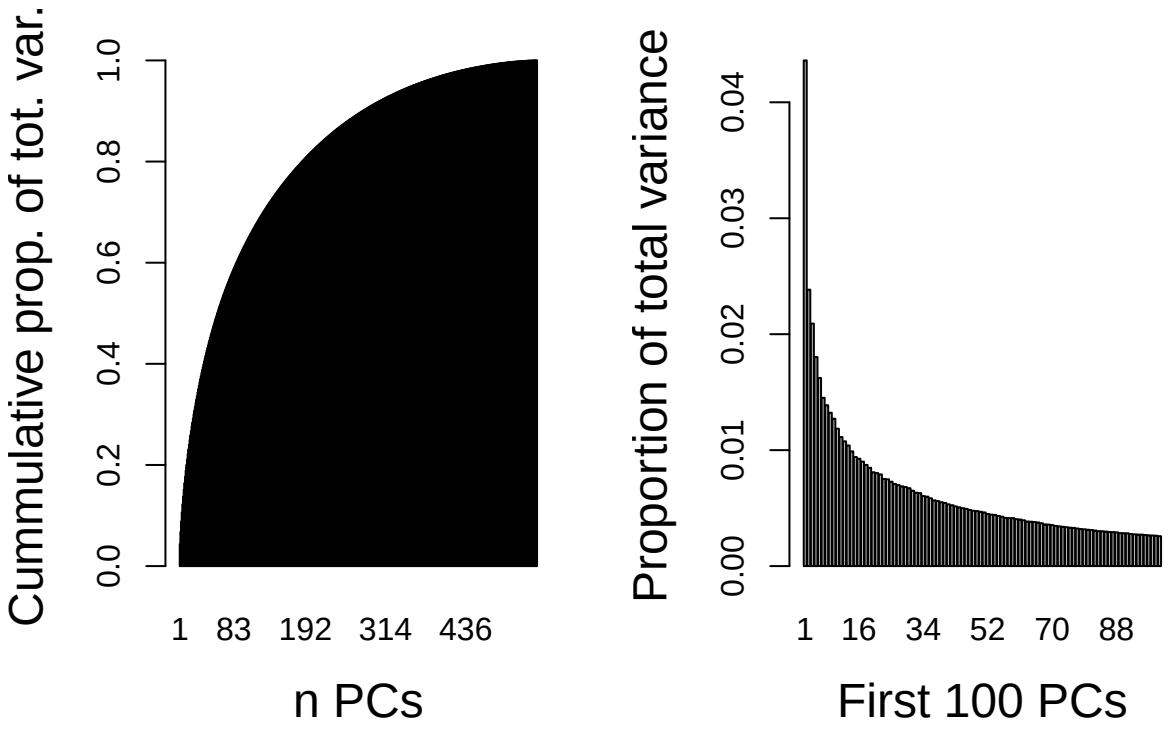
```
#SVD
X.svd <- svd(trainX)
#kable(dim(X.svd$v))
V <- X.svd$v
U <- X.svd$u
D <- diag(X.svd$d)
#Scores
Z <- U%*%D

#Scree plot of the relative variation in X explained by PCs.
par(mfrow=c(1,2))
totvar <- sum(X.svd$d^2)/(dim(trainX)[1]-1)
plot(X.svd$d^2/(dim(trainX)[1]-1), type="b",ylab="Eigen value",
     xlab="n PCs",cex=2, cex.axis=1.5, cex.lab=1.5)
```

```
barplot(X.svd$d^2/(dim(trainX)[1]-1)/totvar,names.arg = 1:dim(trainX)[1],
        ylab="Proportion of total variance", cex.lab=1.5)
```




```
barplot(cumsum(X.svd$d^2/(dim(trainX)[1]-1)/totvar), names.arg = 1:dim(trainX)[1],
        ylab="Cummulative prop. of tot. var.",xlab="n PCs", cex.lab=1.5)
#First 100 eigen values
barplot((X.svd$d^2/(dim(trainX)[1]-1)/totvar)[1:100], names.arg = 1:100,
        ylab="Proportion of total variance", xlab='First 100 PCs',cex.lab=1.5)
```

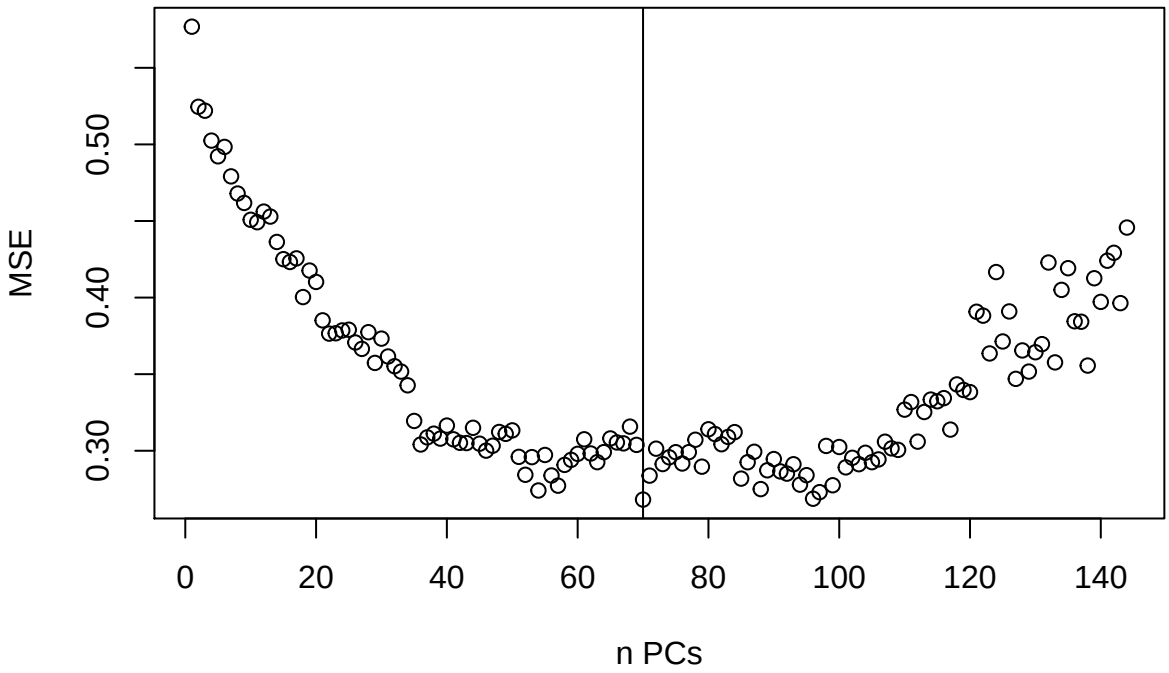



```
par(mfrow=c(1,1))
```

The above plots depict the percent of variance explained in the response variable as a function of the number of components. It could be seen that the variance information is not capture by a few PCs but is spread over the PCs. The first 100 PCs contain less that 70% of the total variability.

Model fitting and selection

The objective of PCR is to obtain an efficient and parsimonious model with fewer number of predictors. A 10 fold cross-validation (CV) technique was then implemented to determine the optimal number of components to be considered in the model. The motivation for choosing CV rather than LOOCV is due to the decent sample size (n=544) and computational efficiency. However, there was no significant difference between 10 fold CV and LOOCV. To assess the number of components to be retained, a plot of the prediction performance (MSE) against the number of components in the model is used. Usually the model with the least extra sample error (MSE) is chosen as the optimal model and in order to avoid over fitting, the local minimum is considered rather than the absolute minimum.



The results indicated that a model constituting the first 70 PCs gave the least MSE (0.27). This could be visualized by the vertical line in the above plot.

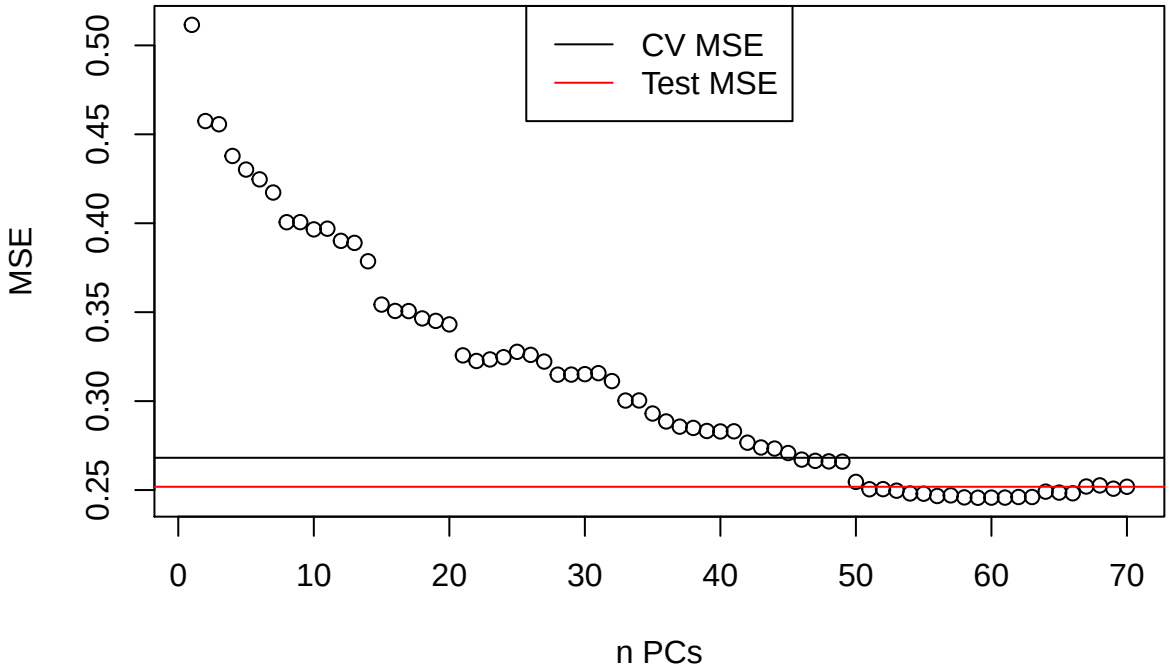
Model validation

To validate our model, it is use to predict the responses in the test data set. This is done by constructing scores (Z) from the test data set using the loadings (V) from the training data set. To evaluate the optimal model fit, a plot of predictive performance against increasing number of components up to 70 is assessed.

```
########################################
# Model validation
##################
#Creating the Z scores of the test data set using the loadings of the trainign data set
testX <- as.matrix(testX)
Ztest=testX%*%V
testX=data.frame(testX)
nm=names(testX)#We need to name our scores correctly

nPC <- 70  #Number of PCs that gave the least MSE
msetest=numeric()
for(i in 1:nPC){
  #Just making sure our data is a data frame
  data=data.frame(Z[,1:i])
  datatest=data.frame(Ztest[,1:i])
  #Create identifiable names
  names(datatest)=nm[1:i]
  names(data)=nm[1:i]
  pcr.mod1=glm(trainY~.,data=data)
  #predict the test data set
  ypred=predict(pcr.mod1,newdata = datatest)
  msetest[i]=MSE(testY,ypred)
  #cat("PC 1 to ",i,"\n")
```

```
}
nPC_at_min_MSE_test=c(1:nPC)[msetest==min(msetest)]
plot(msetest ,xlab = "n PCs",ylab = "MSE")
abline(h=min(cv_error))
abline(h=msetest[70],col="red",lwd=1)
legend('top', legend=c('CV MSE', 'Test MSE'), col=c('black','red'), lty=1)
```



The results indicates that a model with even lower number of components (59 PCs) did better in predicting the responses in our test data set(test MSE = 0.246). Nonetheless, there test MSE for the latter and the optimal model with 70 PCs (test MSE = 0.252) are somewhat similar. Comparing the latter to the CV MSE (0.268), there occur to be pretty close (2% difference) which signifies that our prediction model is a good fit. Also, considering the fact that the MSE is an estimate and the standard errors are not estimated by CV, it could be likely that the test MSE falls within the 95% CI of the model MSE. However, since the test data is assumed to be future data and unobserved, we would consider the model with 70 PCs as our final model.

3) Prediction Model : Penalized Least Square Regression - Lasso

Before building the model, we split the data in the ratio of 70:30 ending up with 543 observations for the training data set, which is a decent number of observations to build the model. We build the model using the train data set and check for MSE on the "test" data set.

```
## set the seed to make your partition reproductible
set.seed(13)
## 70% of the sample size
smp_size <- floor(0.70 * nrow(OTUTableRel))
train_ind <- sample(seq_len(nrow(OTUTableRel)), size = smp_size)
train <- OTUTableRel[train_ind, ]
test <- OTUTableRel[-train_ind, ]
rm(train_ind,smp_size)    #remove the intermediate variables
```

Our objective is to predict 'Age' in days based on OTUs as regressors. With 2239 regressors this is a high dimensional setting. Hence one of the ways to penalize the regression is L1 norm, i.e Lasso.

Lasso model building

Lasso is also a penalized regression, like ridge regression. The lasso estimator of β is the solution to minimizing the penalized SSE.

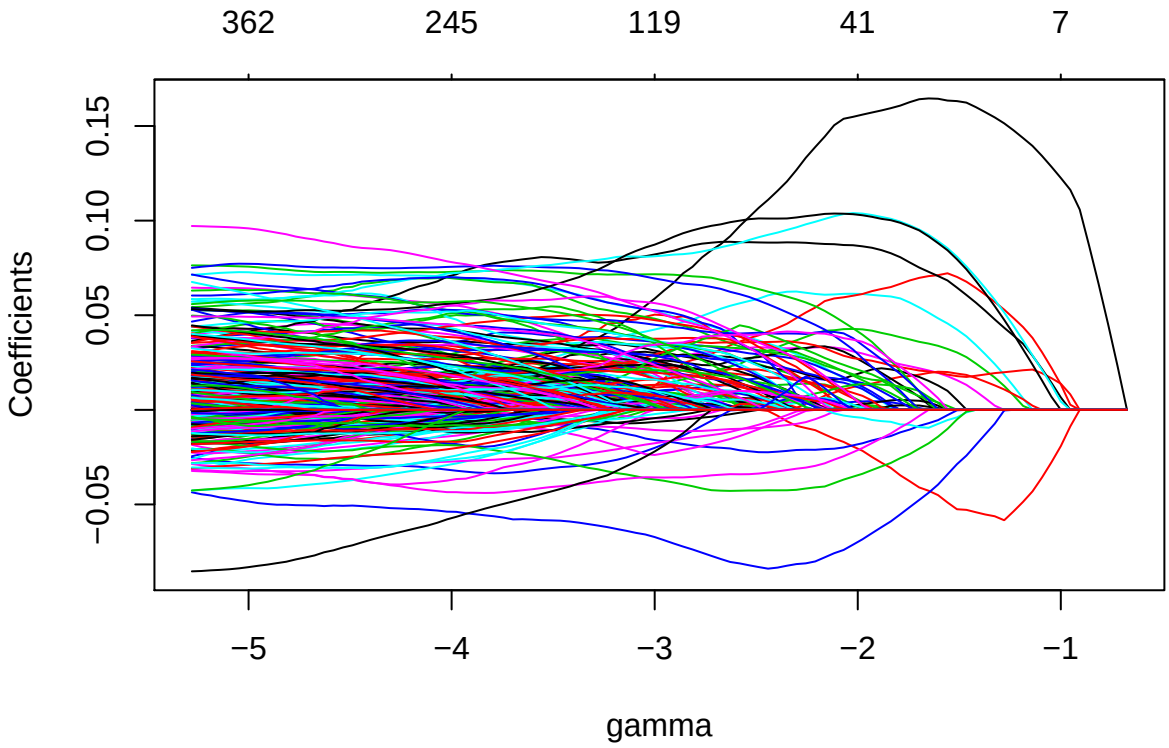
$$SSE = ||\mathbf{Y} - \mathbf{X}\beta||_2^2 \, subject\, to\, |\beta| \leq c$$

or, equivalently, minimizing

$$SSE = \sum_{i=1}^{n}(Y_i - x_i^t\beta)^2 + \gamma \sum_{j=1}^{p} |\beta_j|$$

Selecting the value of γ is part of model building and is aim at minimizing the MSE.

```
lasso.mod = glmnet(scX,y, alpha = 1)            #alpha = 1 defines lasso
plot(lasso.mod,xvar ="lambda", xlab ="gamma")
```

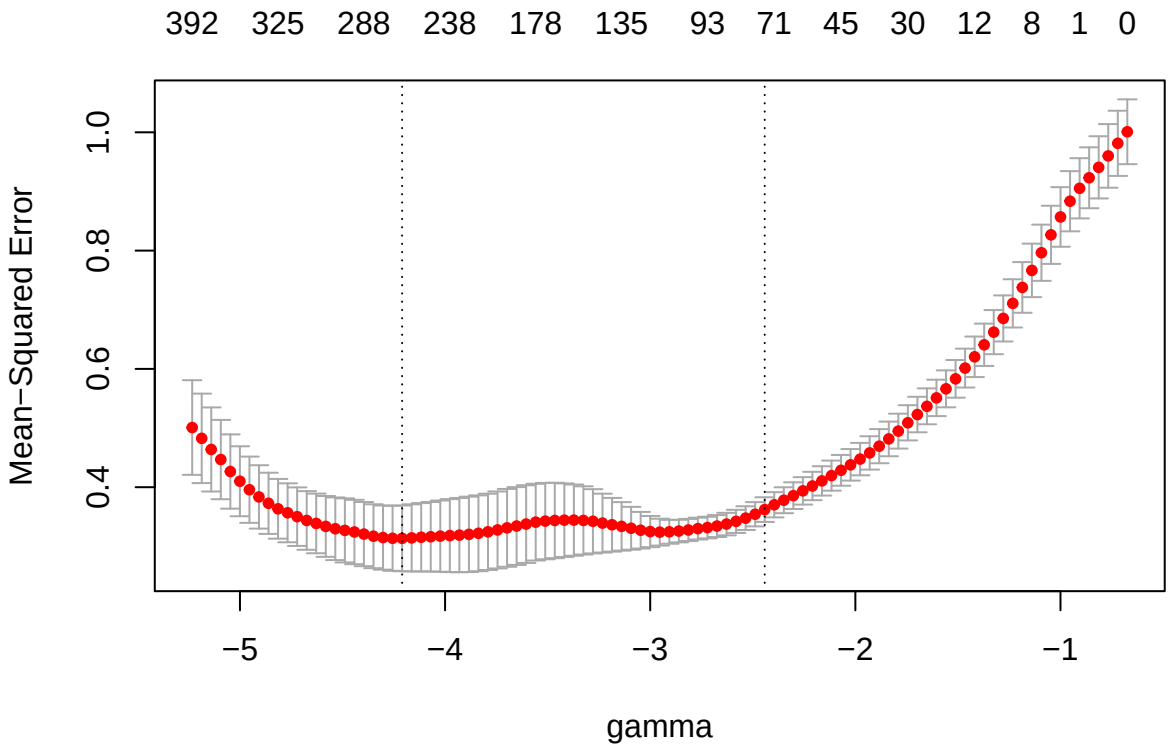


When γ increases the estimates are shrunken towards zero and once they hit zero, they remain zero on further increasing γ. A parameter estimate equal to zero, say $\hat{\beta}_j = 0$, implies that the corresponding predictor is no longer in the model. But we do not know yet what value of gamma is to be chosen to have minimum MSE. Our goal is to have minimum MSE, so that we get good predictions.

To build the model, we implement a 10 fold CV techniques on the training dataset. Given the sample size, LOOCV and 10 fold CV would end up with similar results.

```
set.seed(154)

mcv_10fold <- cv.glmnet(scX,y,alpha = 1 , nfolds = 10)
plot(mcv_10fold, xlab = "gamma")
```




```
cat(log(mcv_10fold$lambda.min))
```

```
## -4.209454
```

```
cat(log(mcv_10fold$lambda.1se))
```

```
## -2.441813
```

The above figure shows that from cross validation we get an optimal γ value of -4.2094541 for the minimum MSE and γ = -2.441813 for "Minimum MSE + 1 Standard Error" (in fact this would further reduce the number of predictors, with a slightly higher MSE; if the goal is to also consider the number of predictors this value for gamma would make more sense).

One may also consider an average of both γs to balance the number of predictors and MSE. However, our aim is prediction hence we concentrate on minimizing the MSE.

Cross validation

The CV gives us random values for lambda (even after setting seed); hence we perform 50 simulation runs and take the average value.

```
#HEAVY CALCULATIONS AHEAD

#The results of cv.glmnet are random, since the folds are selected at random
#Hence reduce this randomness by running cv.glmnet many times, and averaging the error $curves.
iteration=50
lambdas = NULL
for (i in 1:iteration)
```

```
{
    fit <- cv.glmnet(scX,y,alpha = 1 , nfolds = 10)
    errors = data.frame(fit$lambda,fit$cvm, fit$lambda.1se)
    lambdas <- rbind(lambdas,errors)
}

# take mean cvm for each lambda
lambdas <- aggregate(lambdas[, 2:3], list(lambdas$fit.lambda), mean)

# select the best one
bestindex = which(lambdas[2]==min(lambdas[2]))
bestlambda = lambdas[bestindex,1]

# bestindex = which(lambdas[3]==min(lambdas[3]))
# bestlambda.1se = lambdas[min(bestindex),1]  #can be more than one bestindex hence min

# NEED TO FURTHER LOOK INTO HOW TO CAPTURE  fit$lambda.1se from this

log(bestlambda)
```

```
## [1] -2.720914
```

```
# and now run glmnet once more with bestlambda
cv.final.model <- glmnet(scX,y,lambda=bestlambda)
paste('Number of predictors for Lasso =',dim(summary(coef(cv.final.model)))[1], sep=' ')
```

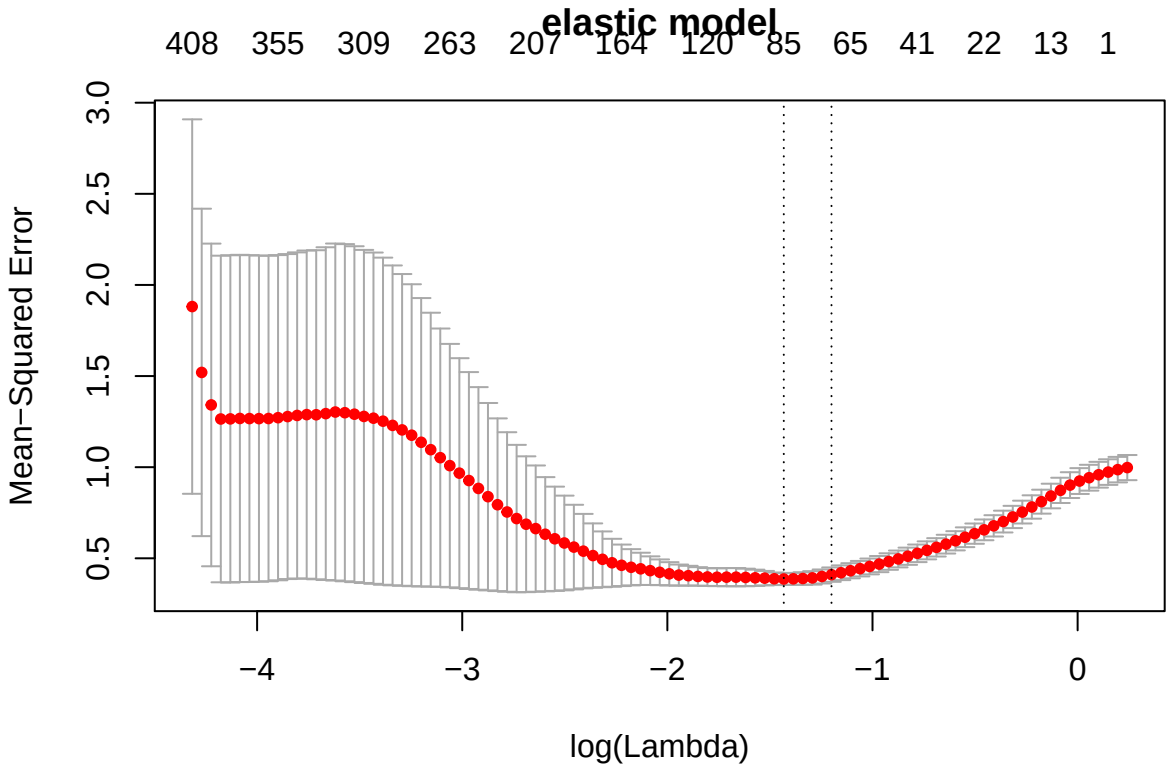
```
## [1] "Number of predictors for Lasso = 95"
```

```
# we have 95 predictors
```

Hence with the model selected, we get **95 coefficients** which are not shrunk to zero and are most significant after L1 norm.

We can also look at the elastic model, which is a combination of Ridge and Lasso. The model building for elastic would mean, finding the value of α and γ both, which together minimizes the prediction MSE.

```
mcv.elastic <- cv.glmnet(scX, y, alpha=0.4, nfolds = 10)
plot(mcv.elastic, main="elastic model")
```

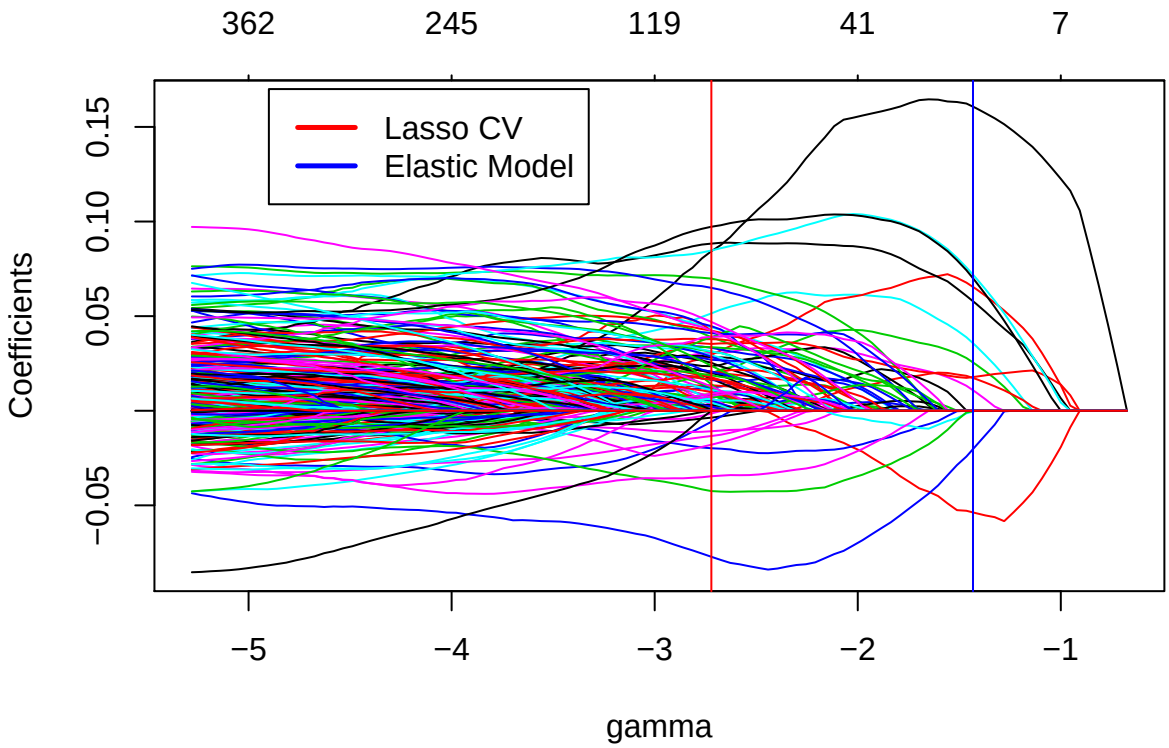



```
final.elastic.model <- glmnet(scX,y,lambda=mcv.elastic$lambda.min, alpha = 0.4)
paste('Number of predictors for elastic net =',
      dim(summary(coef(final.elastic.model)))[1], sep=' ')
```

```
## [1] "Number of predictors for elastic net = 87"
```

The above figure shows the Mean squared error vs log of lambda for the elastic model. The vertical line at the left represents minimum MSE and the one at left minimum MSE + 1 Standard error. With elastic model ($\alpha = 0.4$) we get 87 coefficients which is much more than that for Lasso. However, unlike for Lasso, the choice of $\alpha = 0.4$ is random. No cross validation (or other technique) was implemented to choose the best α weight for Ridge and $1 - \alpha$ weight for Lasso.

```
plot(lasso.mod, xvar ="lambda", xlab ="gamma")
abline(v = log(cv.final.model$lambda), col="red")
abline(v= log(mcv.elastic$lambda.min), col="blue")
legend(-4.9,0.17, c("Lasso CV","Elastic Model"),lty=c(1,1), lwd=c(2.5,2.5),col=c("red","blue"))
```



The vertical red line shows are final model for lasso. Blue line shows the value of gamma for the elastic model.

Model evaluation

As the predicted variable "Age" is a continuous variable, we check the efficiency of the model using **Expected prediction error in x**

$$Err(x) = E_{Y^*,Y}(\hat{Y}(x) - Y^*)^2$$

where Y^* is an outcome at predictor x, independent of the training data.

```
#Cost functions
#mse
MSE_function=function(observedY,predictedY){
n=length(observedY)
MSE=(sum((observedY-predictedY)^2))/n
return(MSE)
}
#model evaluation :
gamma = log(lasso.mod$lambda)

#prediction of TRAIN dataset using generic model
generic.pred = predict(lasso.mod, newx = scX)
generic.MSE = apply(generic.pred, 2,MSE_function, y)


#prediction of TRAIN dataset using final model
final.pred = predict(cv.final.model, newx = scX)     # using the final model
final.MSE = apply(final.pred, 2,MSE_function, y)
```

```
#prediction of TRAIN dataset using ELASTIC model
final.pred.elastic = predict(final.elastic.model, newx = scX)    # using the final model
final.MSE.elastic = apply(final.pred.elastic, 2,MSE_function, y)

final.MSE
```

```
##        s0
## 0.2016343
```

```
final.MSE.elastic
```

```
##        s0
## 0.2801235
```

Prediction error on Training Data
We see the Err(x) = 0.2016343 for the **Lasso**.
And Err(x) = 0.2801235 for the **Elastic model**.
Hence for prediction, the model with the lower MSE is selected.

Prediction error on Test Data

```
####Prediction on the test dataset####

#prediction on test using generic model
generic.pred.test = predict(lasso.mod, newx = xtest.scale)
generic.MSE.test = apply(generic.pred.test, 2,MSE_function, ytest.scale)

#prediction on test using final model
test.final.pred = predict(cv.final.model, newx = xtest.scale) # using the final model
final.MSE.test = apply(test.final.pred, 2,MSE_function, ytest.scale)

#prediction of TEST dataset using ELASTIC model
final.pred.elastic.test = predict(final.elastic.model, newx = xtest.scale)
final.MSE.elastic.test = apply(final.pred.elastic.test, 2,MSE_function, ytest.scale)


#MSE on test dataset with gamma from cross validation
final.MSE.test
```

```
##        s0
## 0.3469706
```

```
final.MSE.elastic.test
```

```
##        s0
## 0.4002039
```

Hence we finally see the errors on our "Test Dataset".

$\boldsymbol{Err(x) = 0.3469706}$..... Lasso

$\boldsymbol{Err(x) = 0.4002039}$..... Elastic net.

Hence we choose the model with minimum error. However, as we have not 'built' the model for elastic net, but randomly chosen the value of $\boldsymbol{\alpha = 0.4}$ we shall not further consider it, even if it gives a lower MSE. Therefore our final prediction model would be discriminated between Lasso and PCR and the one having a lower prediction MSE is selected.

Executive summary

On comparing the MSE from Lasso (extra MSE = 0.347) (as well as elastic; extra MSE = 0.4) model and PCR model we observe a lower MSE for the PCR model. Hence purely keeping a good prediction model as the main objective, amongst the PCR and Lasso model, we retain the PCR is a better model (extra MSE = 0.252).

One of the goals of constructing a prediction model is to obtain a model with just a subset of predictors that could predict the response accurately (parsimonious model). This is achievable with Lasso and elastic net regressions. In this study, amongst the models implemented, PCR proof to perform best in prediction with the smallest MSE.

However in PCR, variables may be included that are unnecessary for prediction because the components serve to describe the variability in the predictors and not the response. Therefore large weights for variables that are strongly correlated to the latter and not the former may be retained. In essence, no matter the number of components retained, the model would always depend on all the predictors.

An alternative technique would be to consider Partial Least Squares Regression (PLSR). Like PCR, this method is also implemented in modelling high dimensional data by constructing new predictor variables (components) as a linear combination of the original predictors but taken into account the correlation with the response as well. In this case only components that are both correlated to both the predictors and response would be included in the model hence may leading to a more parsimonious model.