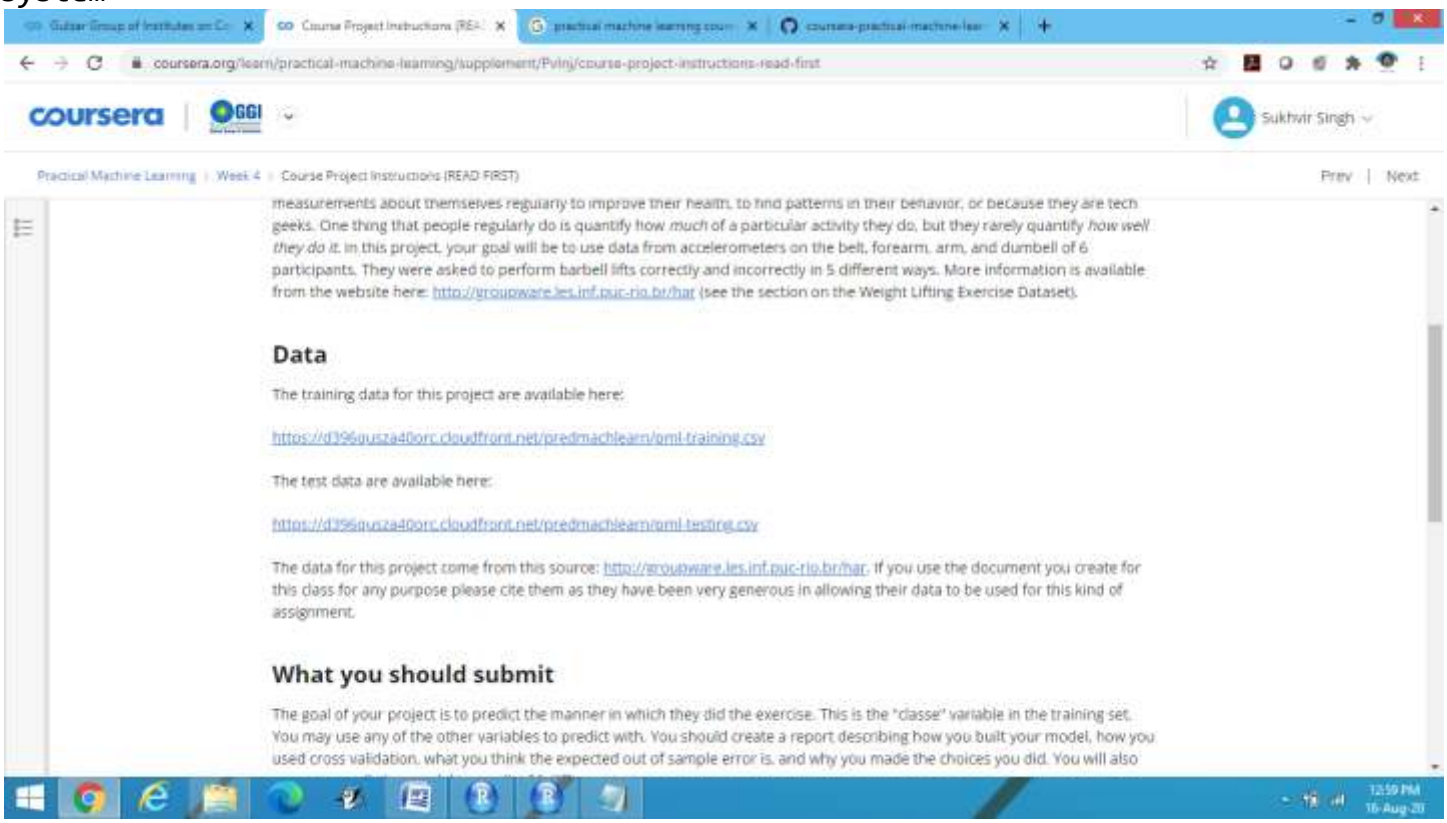
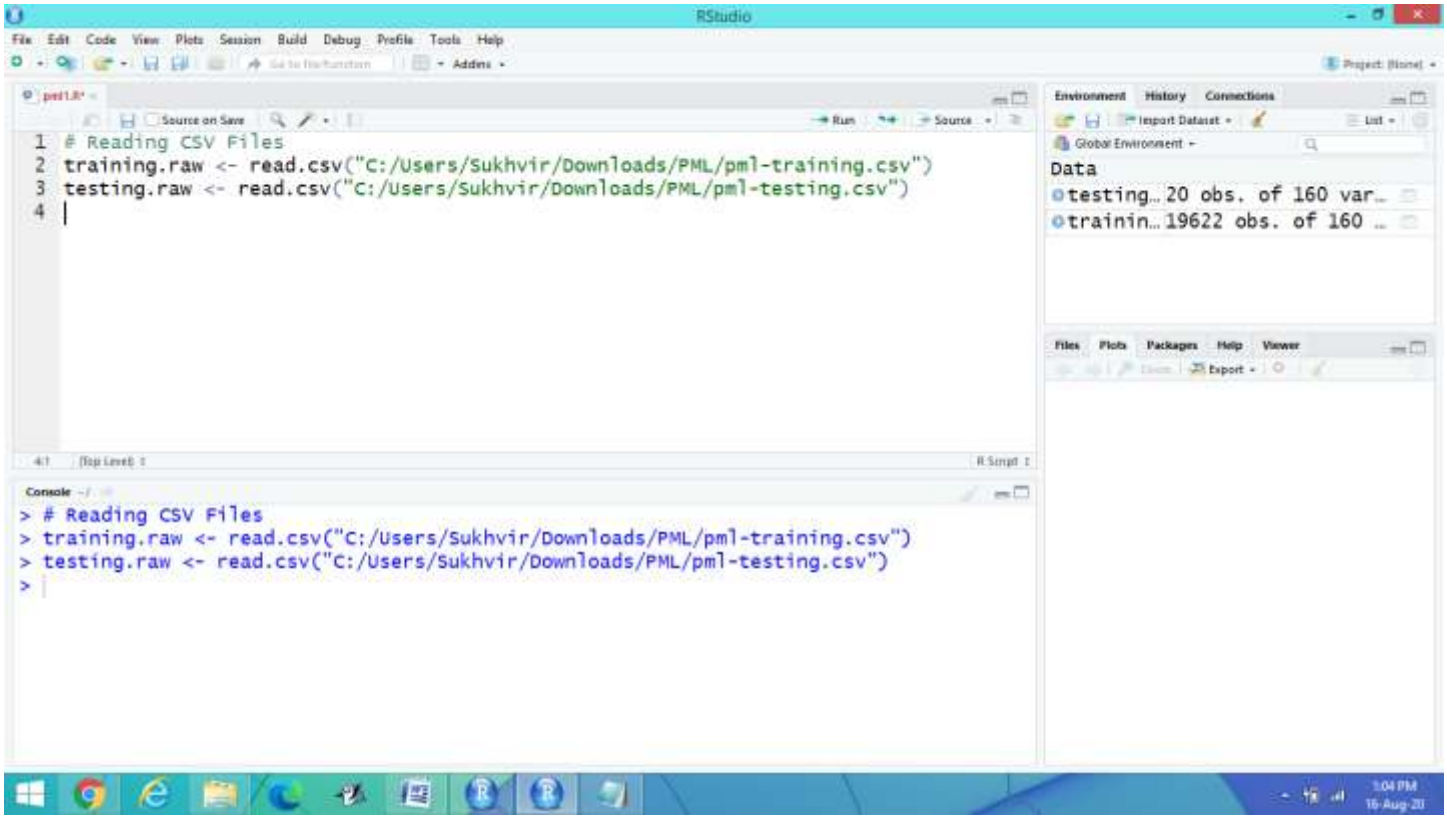
Predictions using Weight Lifting Exercise Dataset
Submitted by Sukhvir Singh
Date: August 16, 2020

Step 1:
Download the data i.e. csv files and place somewhere in some folder in your system



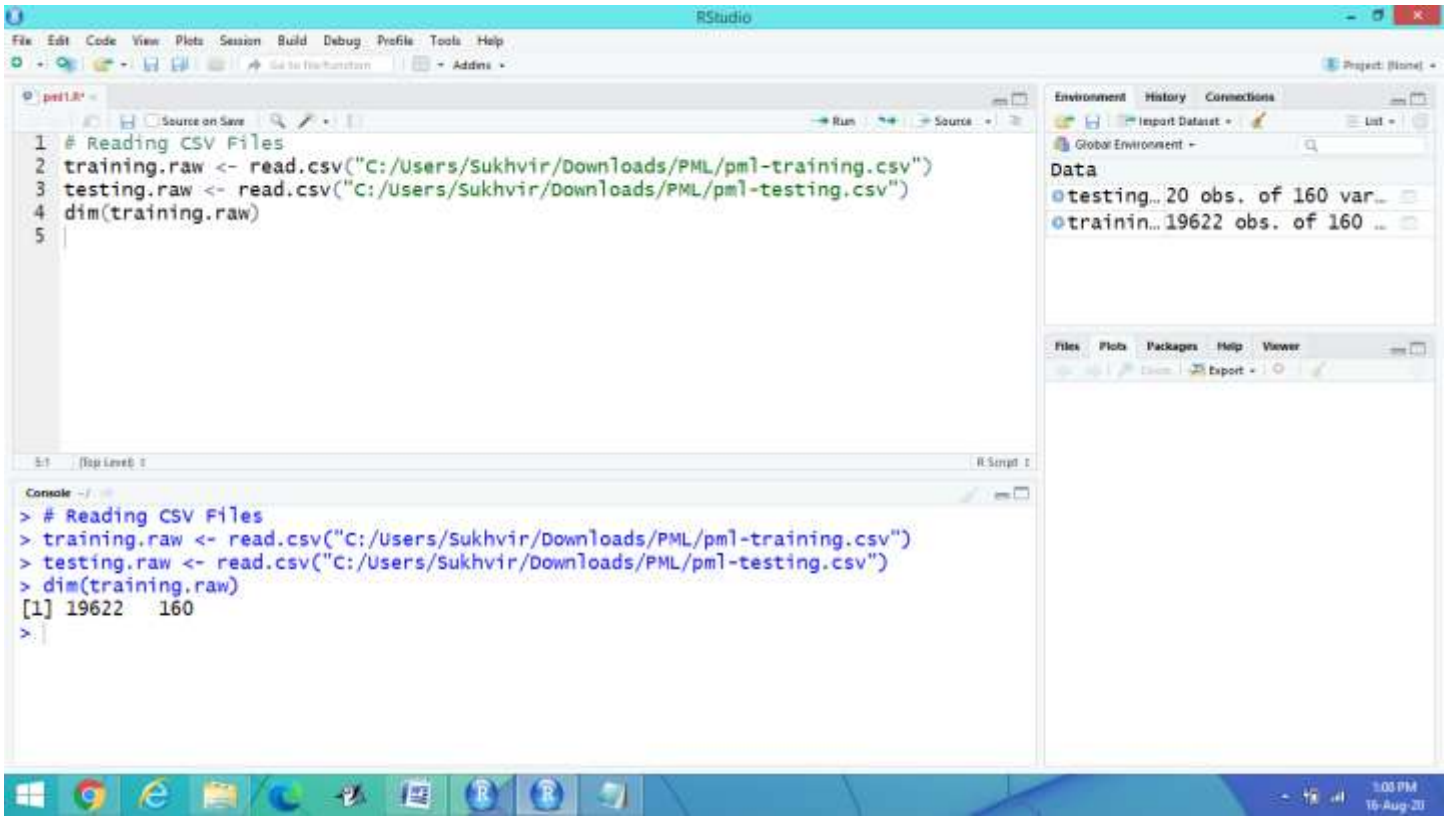
Step 2:
I place the data at the following path:
C:/Users/Sukhvir/Downloads/PML/pml-training.csv
C:/Users/Sukhvir/Downloads/PML/pml-testing.csv

Step 3:
```
# Now read the files from R studio
# Reading CSV Files
training.raw <- read.csv("C:/Users/Sukhvir/Downloads/PML/pml-training.csv")
testing.raw <- read.csv("C:/Users/Sukhvir/Downloads/PML/pml-testing.csv")
```



Step 4:

```
# Check the dimensions of data set
```

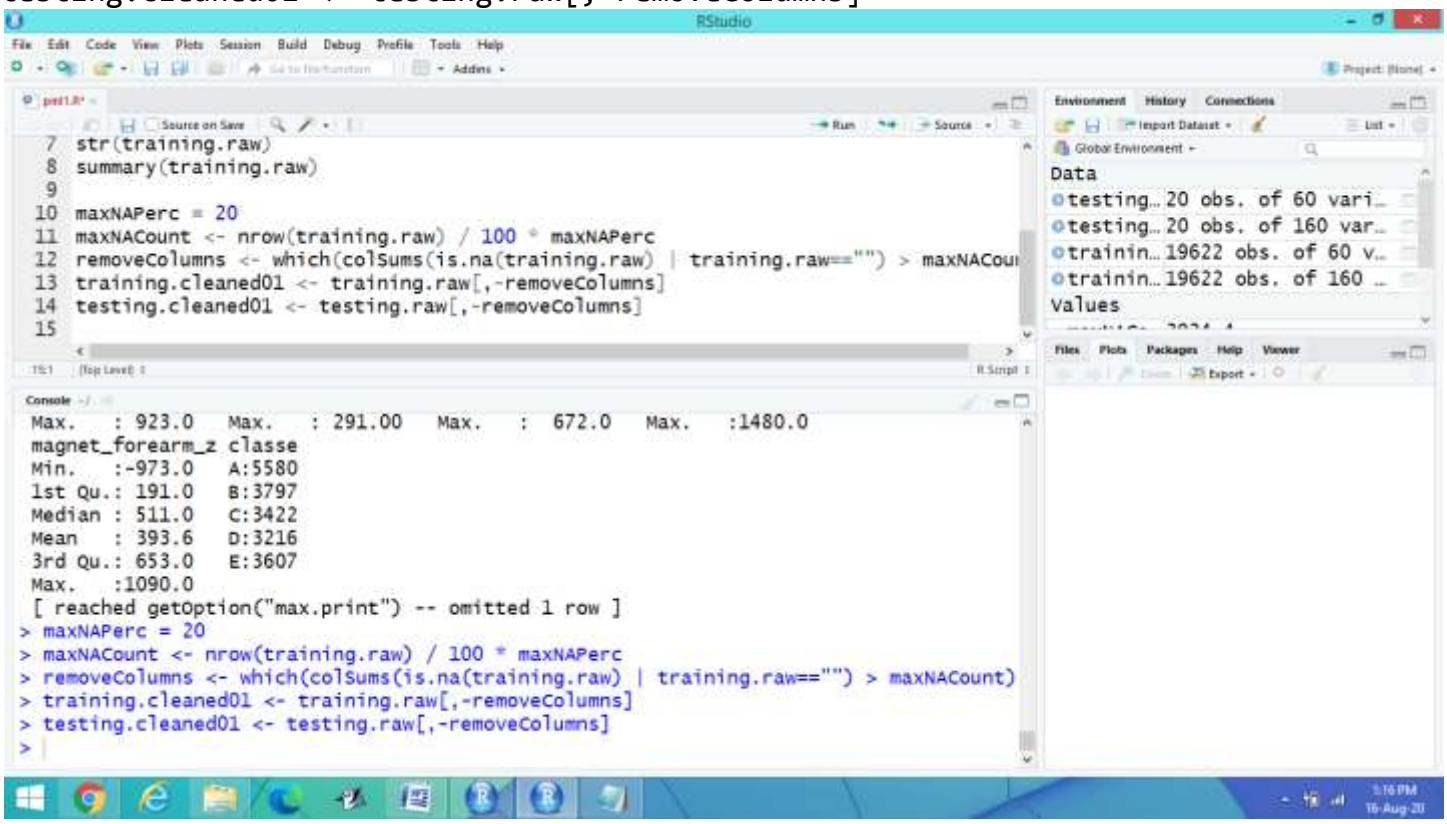


Step: 5

```
> head (training.raw)
> str (training.raw)
> summary (training.raw)
# by implementing the above commands in R you can see all the data.
```

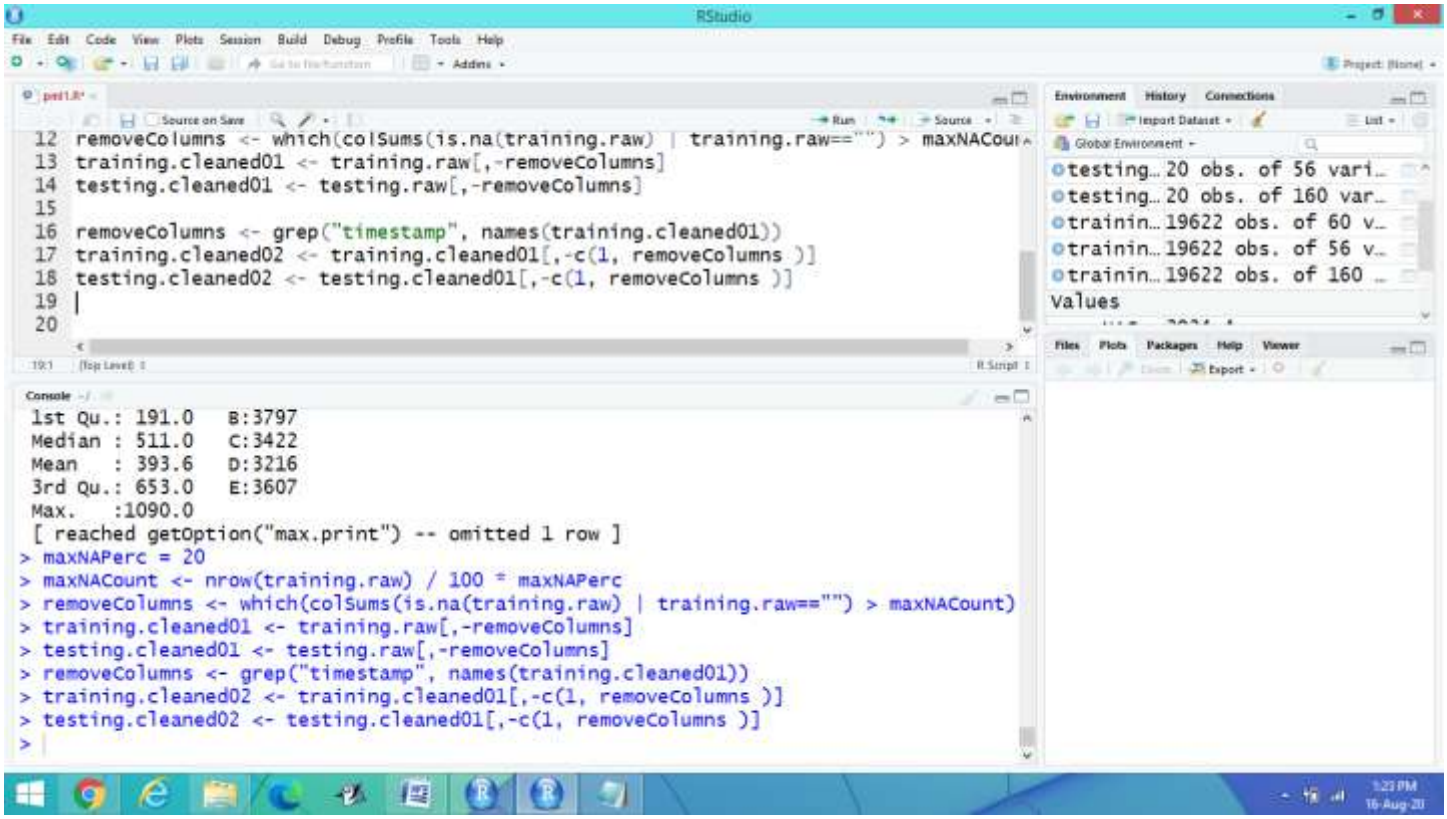
Step: 6

```
# let us remove the unnecessary NA values from the above display
maxNAPerc = 20
maxNACount <- nrow(training.raw) / 100 * maxNAPerc
removeColumns <- which(colSums(is.na(training.raw) | training.raw==="") > maxNACount)
training.cleaned01 <- training.raw[,-removeColumns]
testing.cleaned01 <- testing.raw[,-removeColumns]
```



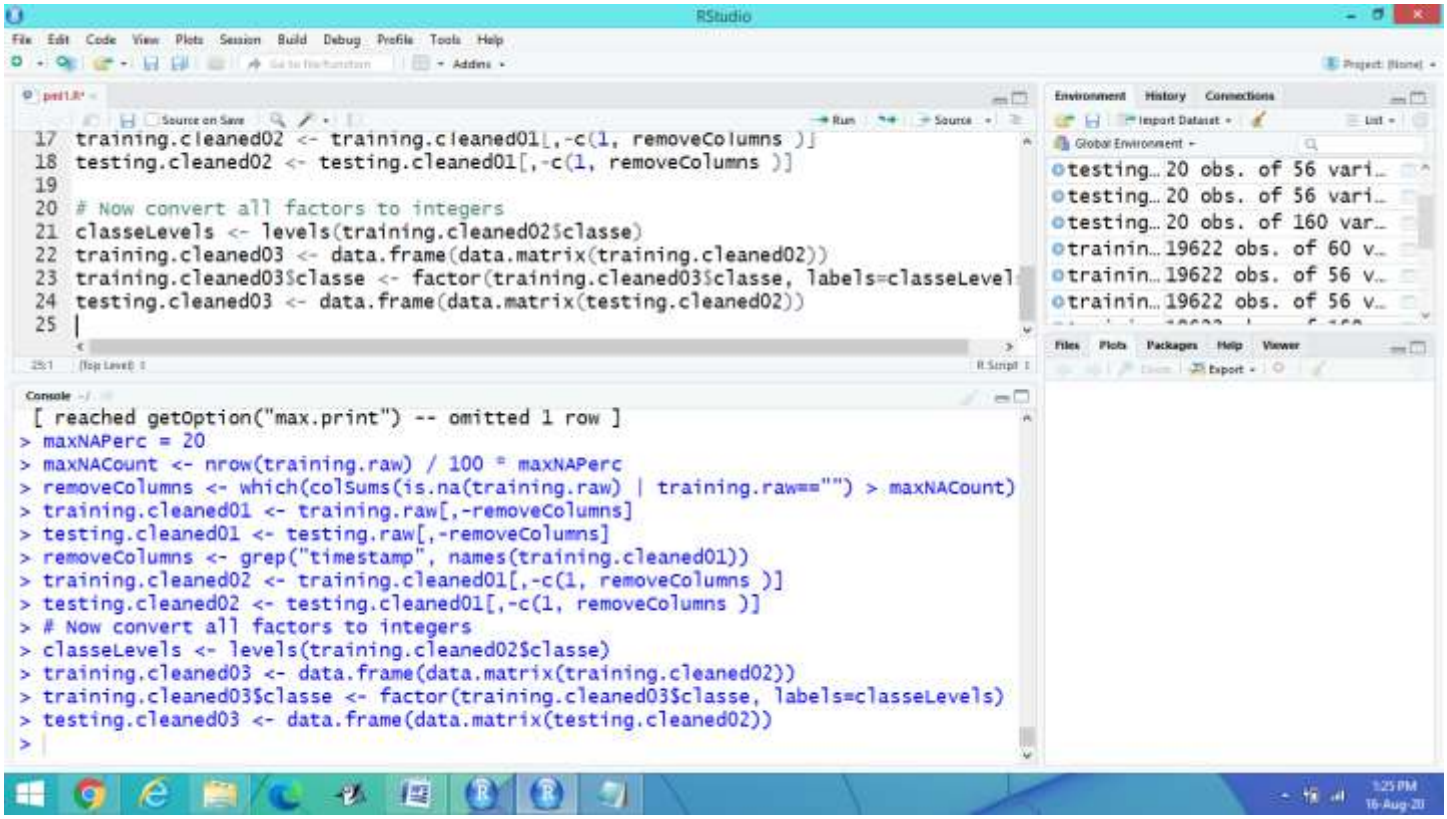
Step: 7

```
# Removal of all Time Related Data
removeColumns <- grep("timestamp", names(training.cleaned01))
training.cleaned02 <- training.cleaned01[,-c(1, removeColumns )]
testing.cleaned02 <- testing.cleaned01[,-c(1, removeColumns )]
```



Step: 8

```
# Now convert all factors to integers
classeLevels <- levels(training.cleaned02$classe)
training.cleaned03 <- data.frame(data.matrix(training.cleaned02))
training.cleaned03$classe <- factor(training.cleaned03$classe, labels=classeLevels)
testing.cleaned03 <- data.frame(data.matrix(testing.cleaned02))
```

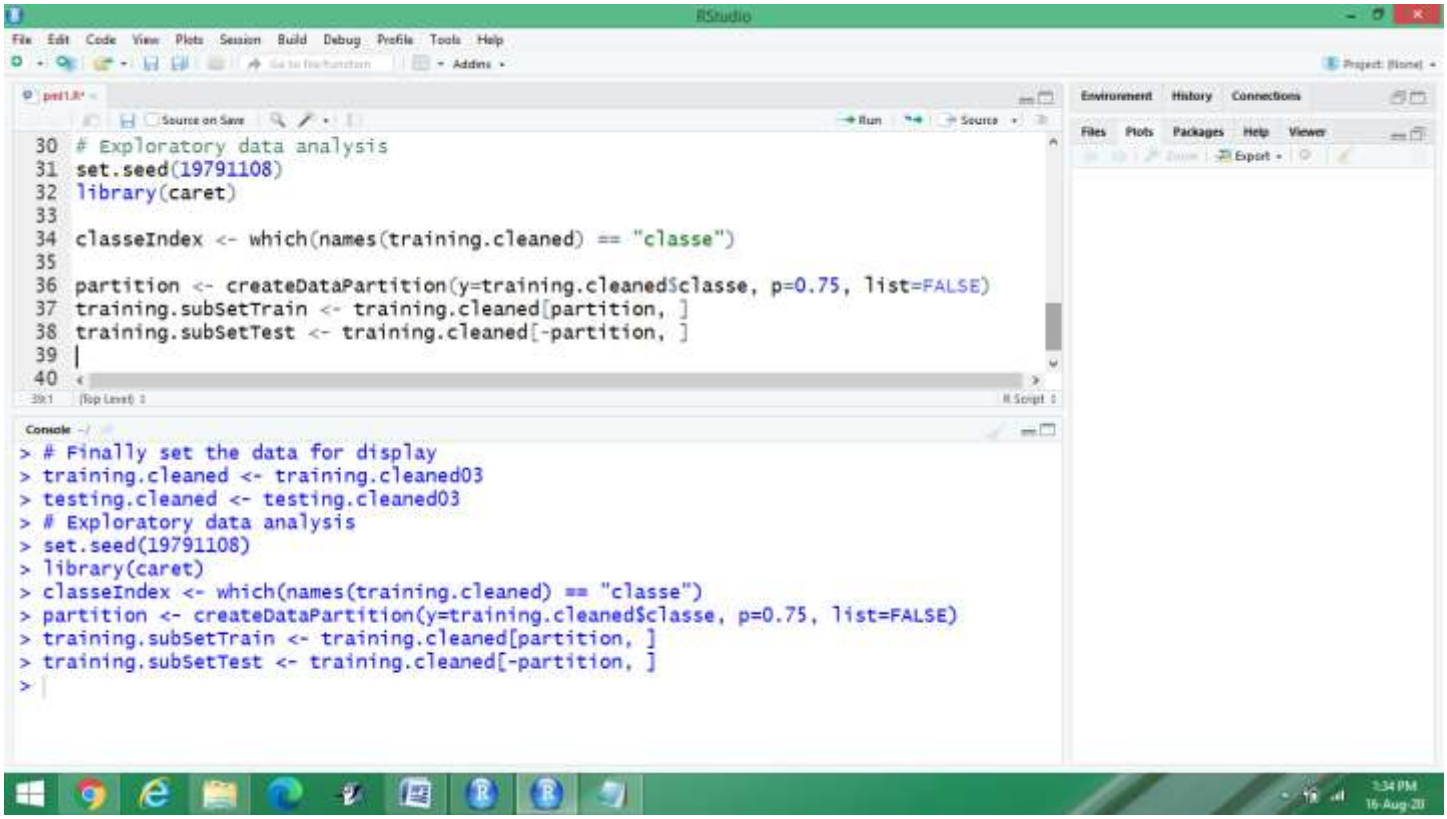


Step: 9

```
# Finally set the data for display
training.cleaned <- training.cleaned03
testing.cleaned <- testing.cleaned03
```

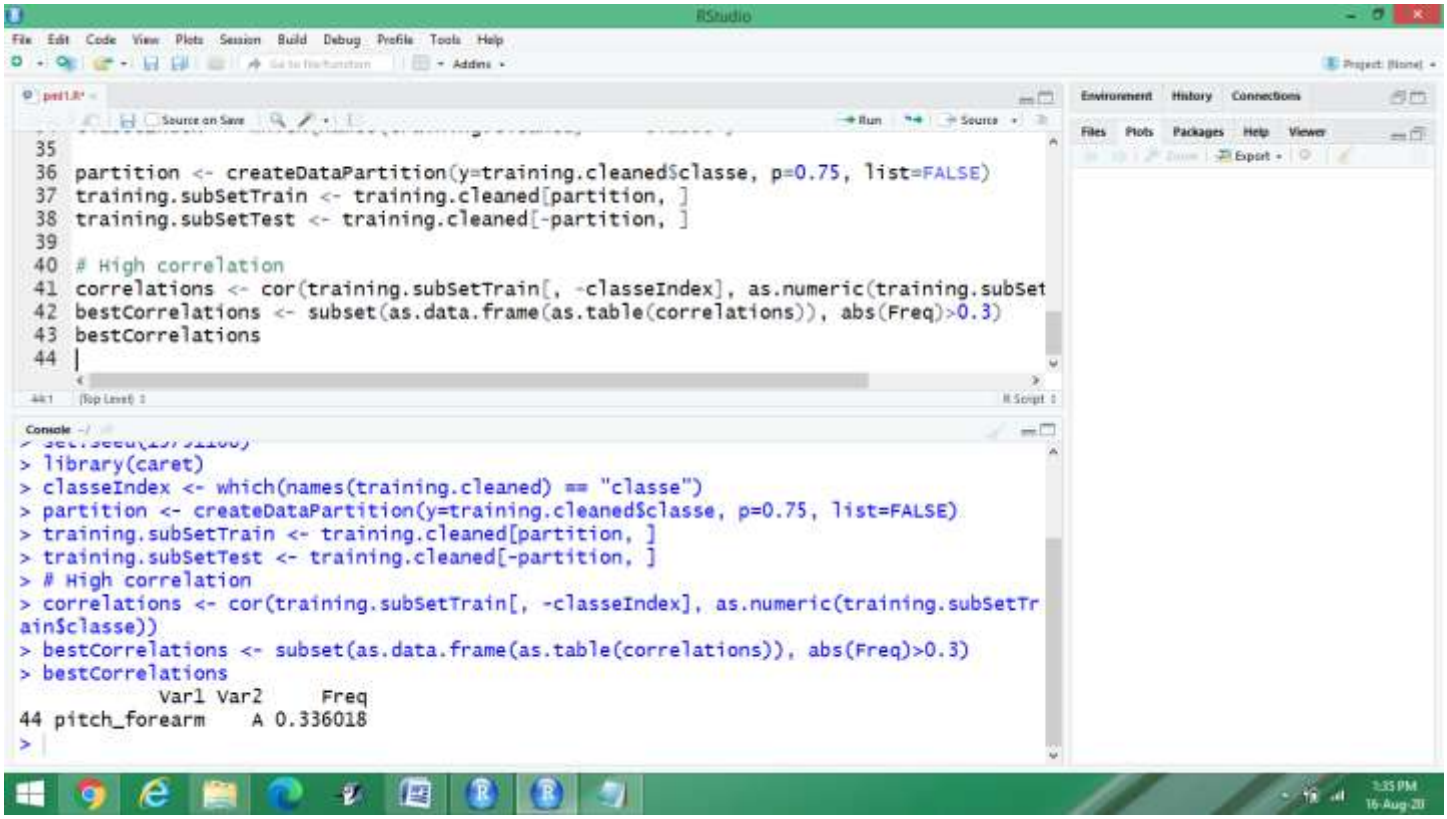
Step: 10

```
# Exploratory data analysis
set.seed(19791108)
library(caret)
classeIndex <- which(names(training.cleaned) == "classe")
partition <- createDataPartition(y=training.cleaned$classe, p=0.75, list=FALSE)
training.subSetTrain <- training.cleaned[partition, ]
training.subSetTest <- training.cleaned[-partition, ]
```



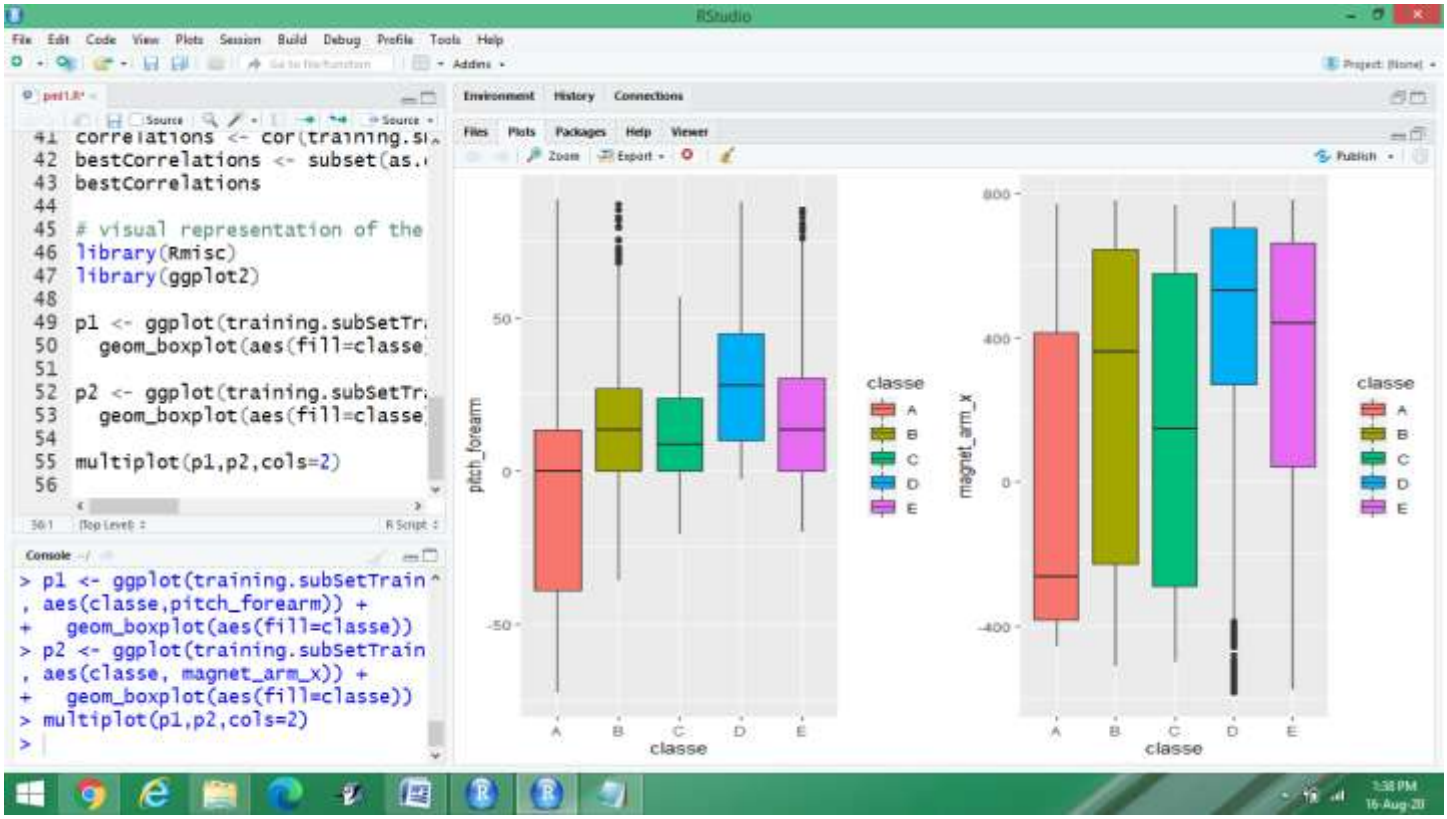
Step: 11

```
# High correlation
correlations <- cor(training.subSetTrain[, -classeIndex], as.numeric(training.subSetTrain$classe))
bestCorrelations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorrelations
```



Step: 12

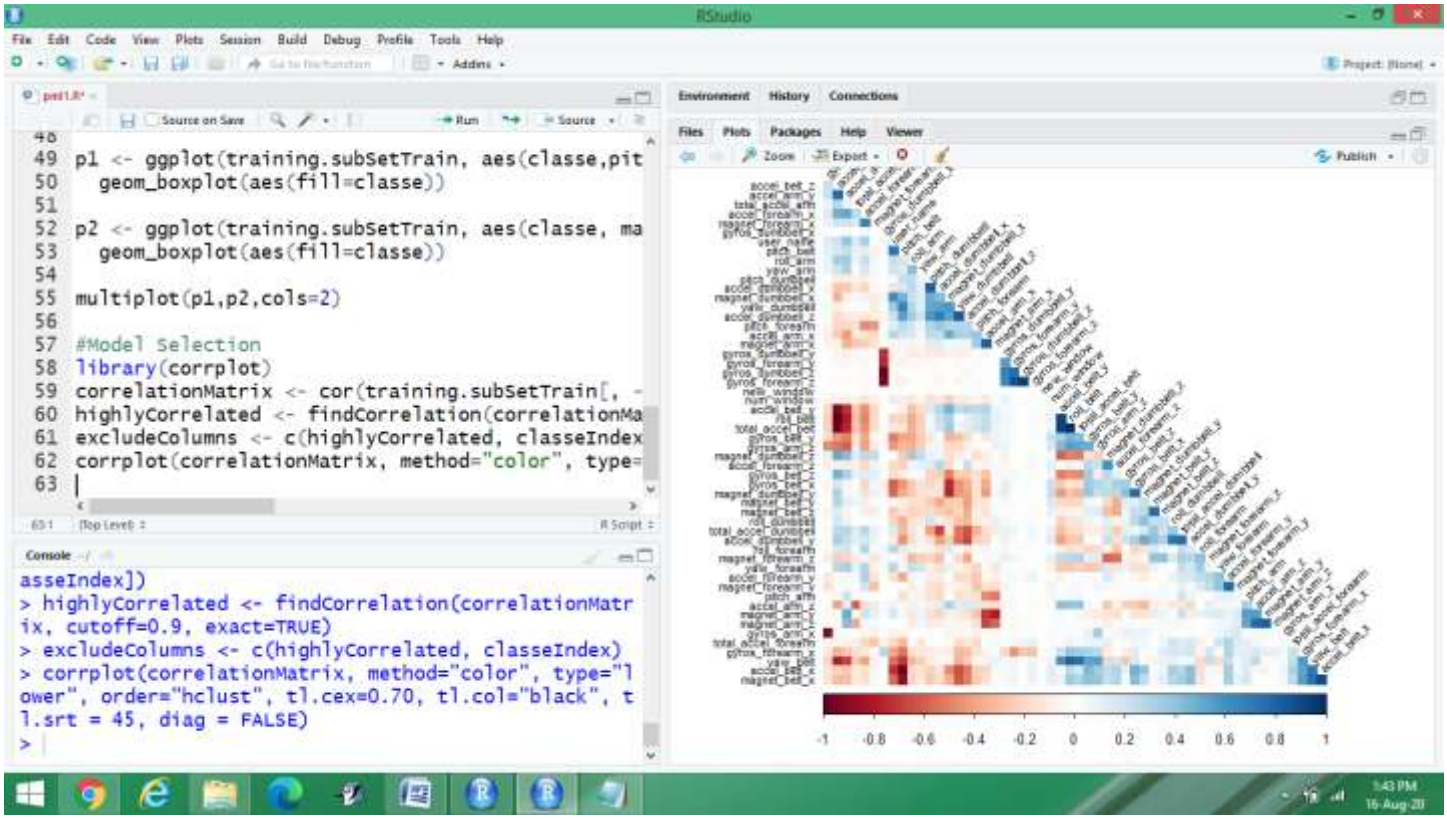
```
# visual representation of the above data
library(Rmisc)
library(ggplot2)

p1 <- ggplot(training.subSetTrain, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))

p2 <- ggplot(training.subSetTrain, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))

multiplot(p1,p2,cols=2)
```



As per from the above representation no hard separation of classes possible
Let's train some models for prediction with the installation of corrplot

Step: 13

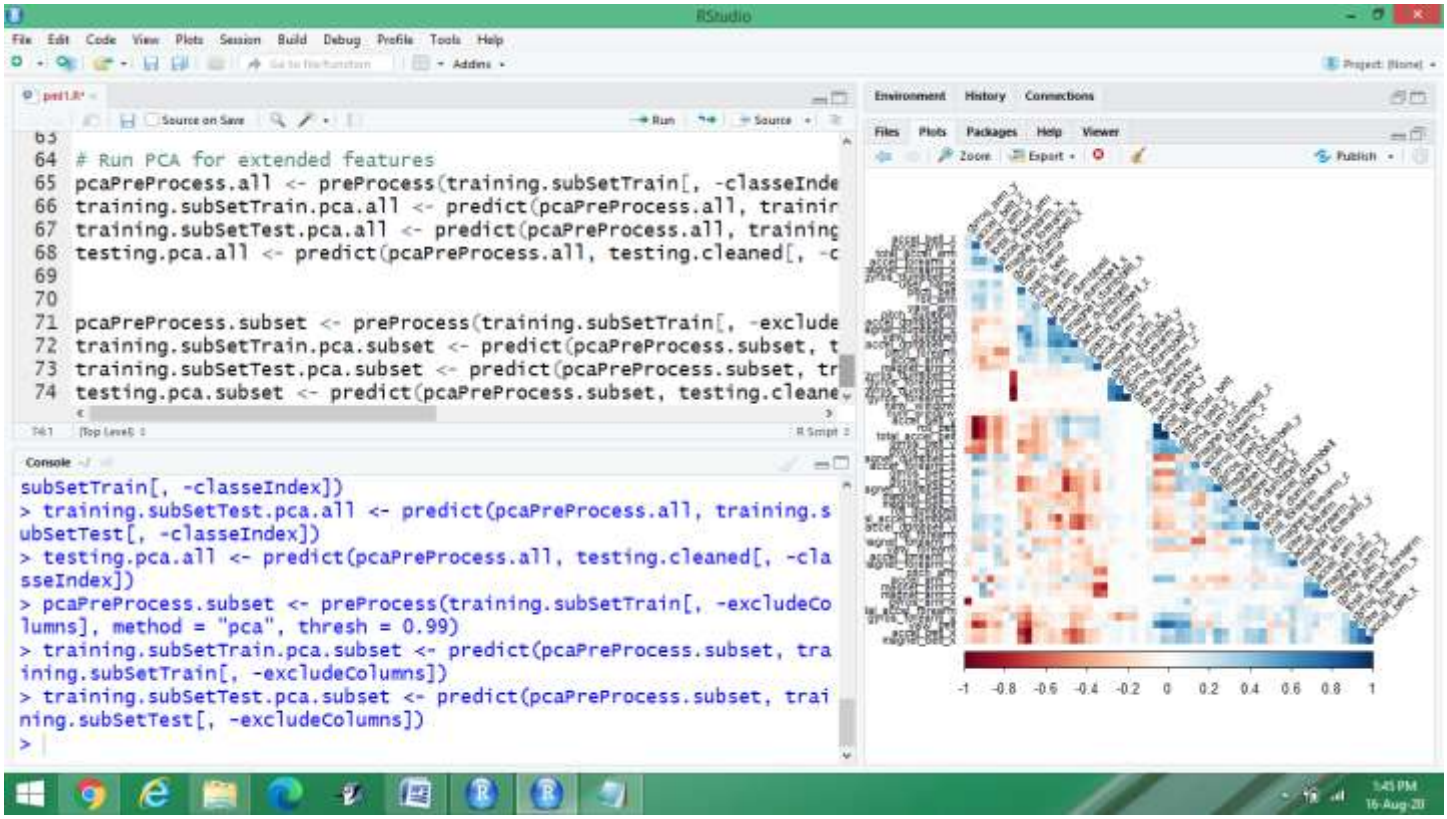
```
#Model Selection
library(corrplot)
correlationMatrix <- cor(training.subSetTrain[, -classeIndex])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.9, exact=TRUE)
excludeColumns <- c(highlyCorrelated, classeIndex)
corrplot(correlationMatrix,    method="color",    type="lower",    order="hclust",
tl.cex=0.70, tl.col="black", tl.srt = 45, diag = FALSE)
```



Step: 14

 # Run PCA for extended features

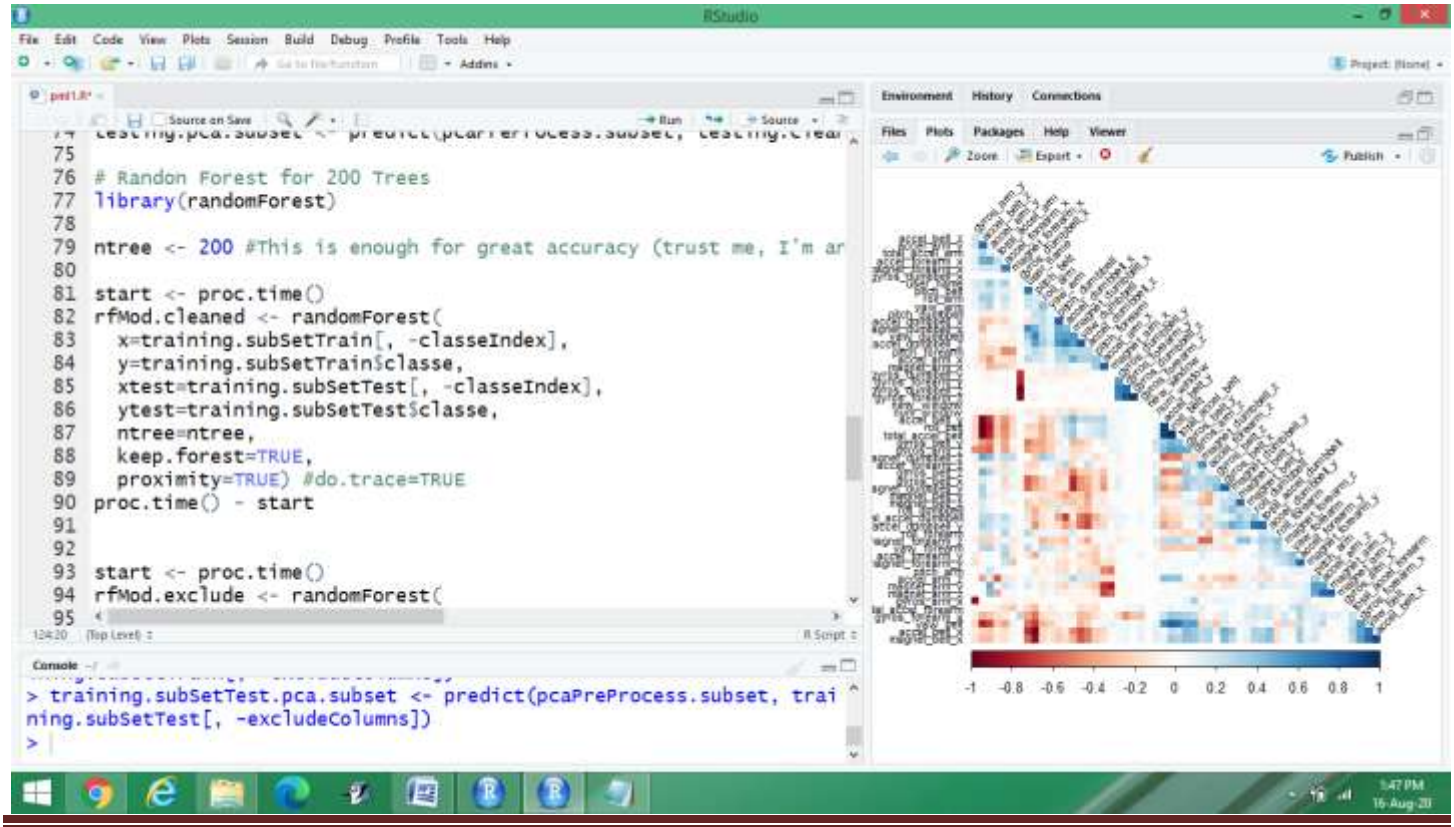
```
pcaPreProcess.all <- preProcess(training.subSetTrain[, -classeIndex], method = "pca", thresh = 0.99)
training.subSetTrain.pca.all <- predict(pcaPreProcess.all, training.subSetTrain[, -classeIndex])
training.subSetTest.pca.all <- predict(pcaPreProcess.all, training.subSetTest[, -classeIndex])
testing.pca.all <- predict(pcaPreProcess.all, testing.cleaned[, -classeIndex])


pcaPreProcess.subset <- preProcess(training.subSetTrain[, -excludeColumns], method = "pca", thresh = 0.99)
training.subSetTrain.pca.subset <- predict(pcaPreProcess.subset, training.subSetTrain[, -excludeColumns])
training.subSetTest.pca.subset <- predict(pcaPreProcess.subset, training.subSetTest[, -excludeColumns])
testing.pca.subset <- predict(pcaPreProcess.subset, testing.cleaned[, -classeIndex])
```

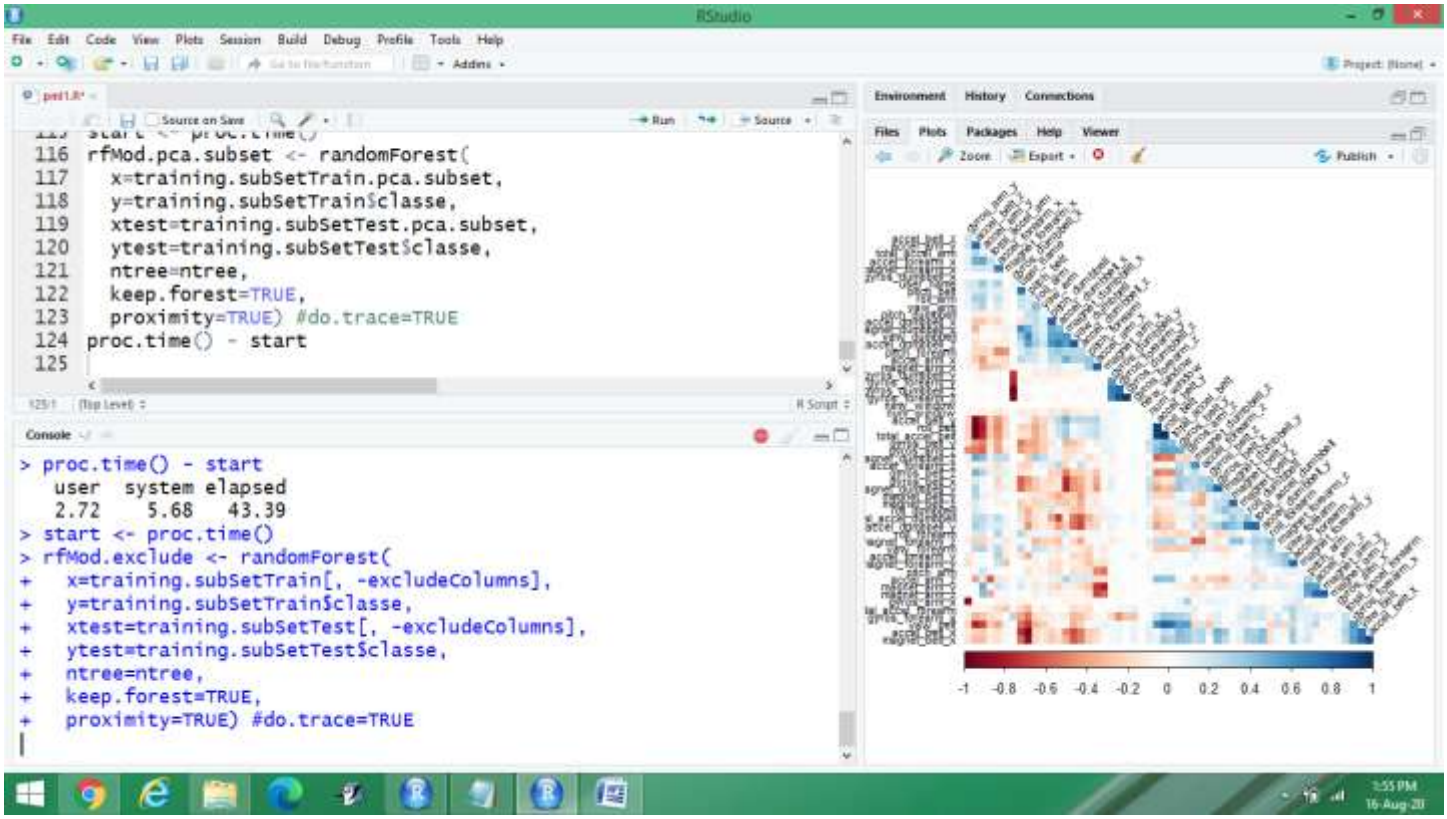


Step: 15

```
# Randon Forest for 200 Trees
library(randomForest)
ntree <- 200 #This is enough for great accuracy (trust me, I'm an engineer).

start <- proc.time()
rfMod.cleaned <- randomForest(
  x=training.subSetTrain[, -classeIndex],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -classeIndex],
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start

start <- proc.time()
rfMod.exclude <- randomForest(
  x=training.subSetTrain[, -excludeColumns],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -excludeColumns],
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
```

```
  proximity=TRUE) #do.trace=TRUE
proc.time() - start

start <- proc.time()
rfMod.pca.all <- randomForest(
  x=training.subSetTrain.pca.all,
  y=training.subSetTrain$classe,
  xtest=training.subSetTest.pca.all,
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start

start <- proc.time()
rfMod.pca.subset <- randomForest(
  x=training.subSetTrain.pca.subset,
  y=training.subSetTrain$classe,
  xtest=training.subSetTest.pca.subset,
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() – start
```

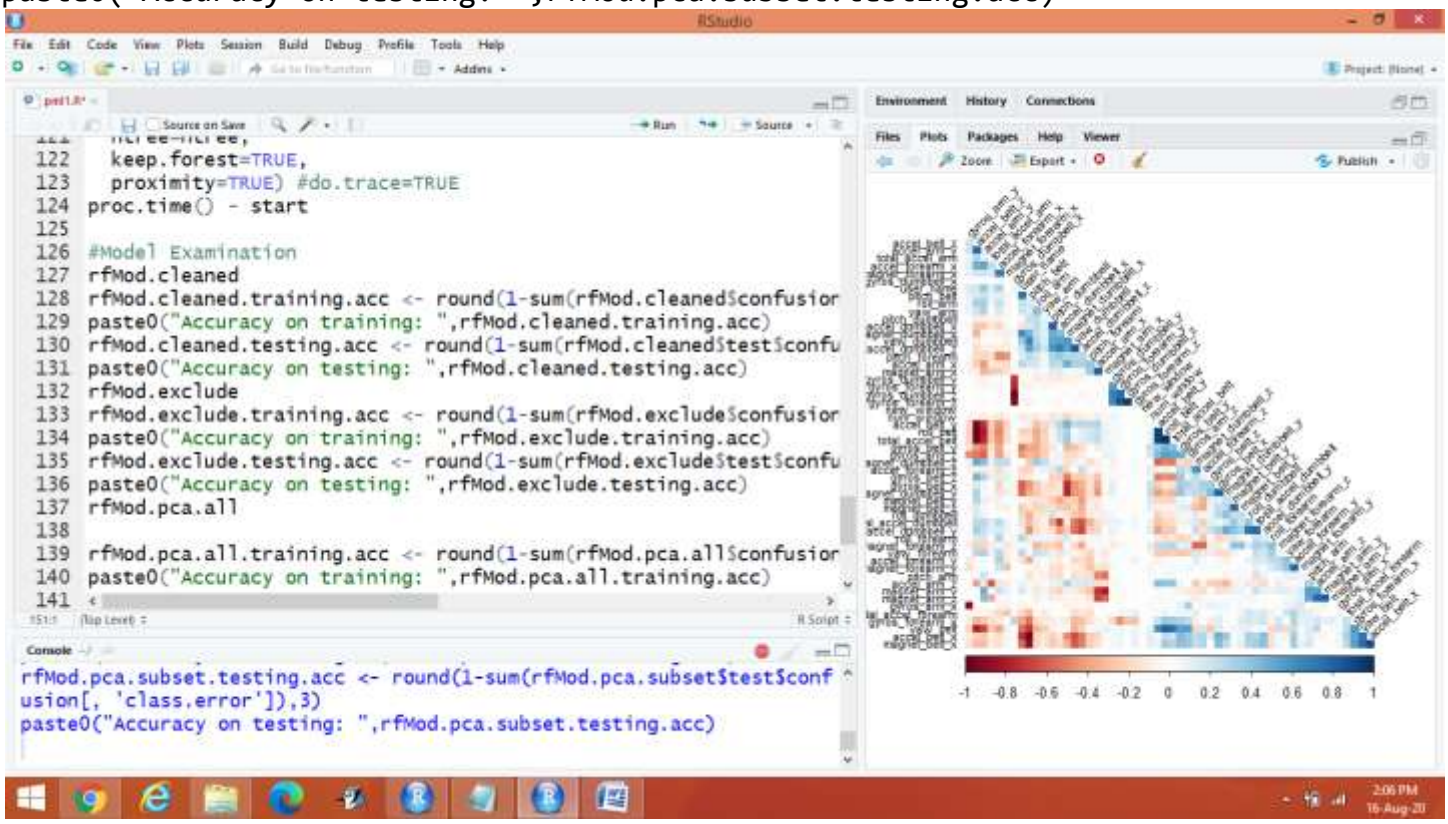




Step: 16

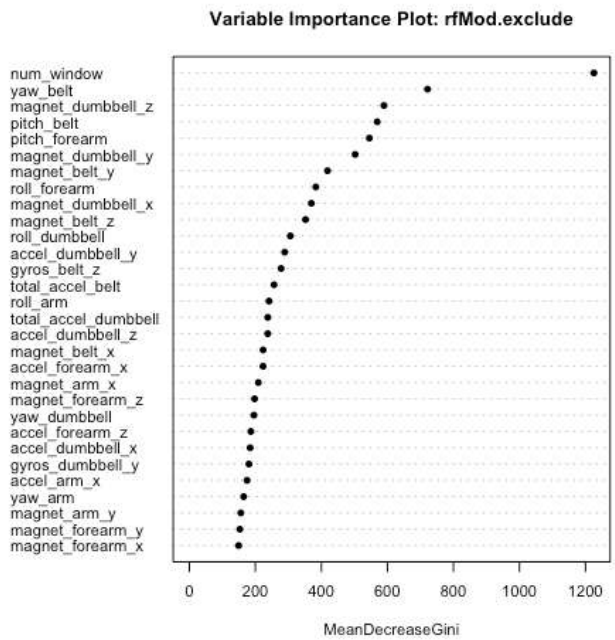
```
# Model examination
rfMod.cleaned
rfMod.cleaned.training.acc <- round(1-sum(rfMod.cleaned$confusion[,
'class.error']),3)
paste0("Accuracy on training: ",rfMod.cleaned.training.acc)
rfMod.cleaned.testing.acc <- round(1-sum(rfMod.cleaned$test$confusion[,
'class.error']),3)
paste0("Accuracy on testing: ",rfMod.cleaned.testing.acc)
rfMod.exclude
rfMod.exclude.training.acc <- round(1-sum(rfMod.exclude$confusion[,
'class.error']),3)
paste0("Accuracy on training: ",rfMod.exclude.training.acc)
rfMod.exclude.testing.acc <- round(1-sum(rfMod.exclude$test$confusion[,
'class.error']),3)
paste0("Accuracy on testing: ",rfMod.exclude.testing.acc)
rfMod.pca.all

rfMod.pca.all.training.acc <- round(1-sum(rfMod.pca.all$confusion[,
'class.error']),3)
paste0("Accuracy on training: ",rfMod.pca.all.training.acc)

rfMod.pca.all.testing.acc <- round(1-sum(rfMod.pca.all$test$confusion[,
'class.error']),3)
```

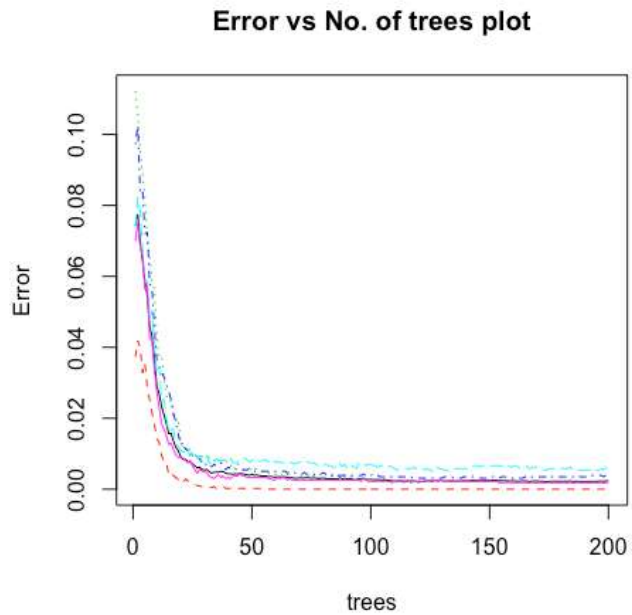
```
paste0("Accuracy on testing: ",rfMod.pca.all.testing.acc)

rfMod.pca.subset
rfMod.pca.subset.training.acc <- round(1-sum(rfMod.pca.subset$confusion[,
'class.error']),3)
paste0("Accuracy on training: ",rfMod.pca.subset.training.acc)

rfMod.pca.subset.testing.acc <- round(1-sum(rfMod.pca.subset$test$confusion[,
'class.error']),3)
paste0("Accuracy on testing: ",rfMod.pca.subset.testing.acc)
```



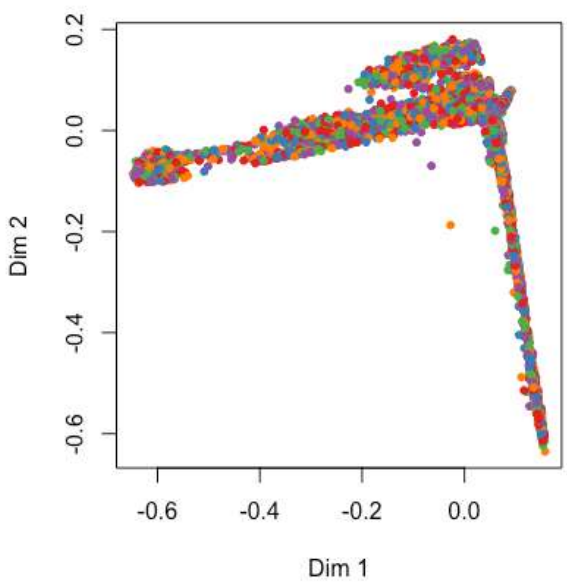
Step: 17

```
# Conclusion
Result is that nor PCA doesn’t have a +ve of the accuracy
rfmod.exclude is slightly better than rfmod.cleaned
Let’s draw some more plots before finalizing the answer

par(mfrow=c(1,2))
varImpPlot(rfMod.exclude, cex=0.7, pch=16, main='Variable Importance Plot:
rfMod.exclude')
plot(rfMod.exclude, , cex=0.7, main='Error vs No. of trees plot')
```



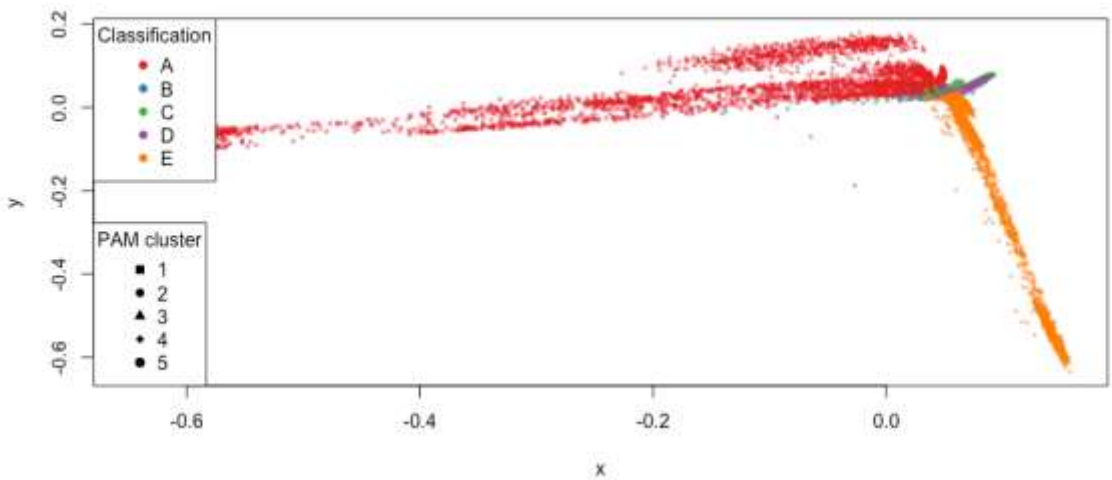



```
par(mfrow=c(1,1))
start <- proc.time()
library(RColorBrewer)
palette <- brewer.pal(length(classeLevels), "Set1")
rfMod.mds <- MDSplot(rfMod.exclude, as.factor(classeLevels), k=2, pch=20,
palette=palette)
```




```
library(cluster)
rfMod.pam <- pam(1 - rfMod.exclude$proximity, k=length(classeLevels), diss=TRUE)

plot(
```

```
  rfMod.mds$points[, 1],
  rfMod.mds$points[, 2],
  pch=rfMod.pam$clustering+14,
  col=alpha(palette[as.numeric(training.subSetTrain$classe)],0.5),
  bg=alpha(palette[as.numeric(training.subSetTrain$classe)],0.2),
  cex=0.5,
  xlab="x", ylab="y")
legend("bottomleft",                          legend=unique(rfMod.pam$clustering),
pch=seq(15,14+length(classeLevels)), title = "PAM cluster")
  legend("topleft",  legend=classeLevels,  pch  =  16,  col=palette,  title  =
"Classification")
```



Step: 18

```
# Test Result
predictions <- t(cbind(
    exclude=as.data.frame(predict(rfMod.exclude,        testing.cleaned[,        -
excludeColumns]), optional=TRUE),
    cleaned=as.data.frame(predict(rfMod.cleaned,                testing.cleaned),
optional=TRUE),
    pcaAll=as.data.frame(predict(rfMod.pca.all, testing.pca.all), optional=TRUE),
    pcaExclude=as.data.frame(predict(rfMod.pca.subset,        testing.pca.subset),
optional=TRUE)
))
predictions
```

```
##            1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
## exclude    "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## cleaned    "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## pcaAll     "B" "A" "C" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## pcaExclude "B" "A" "C" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
```