Biomarker Discovery with Random Forest

Dr. Animesh Acharjee

3 January 2017

rm(list=ls(all=TRUE))  
  
 library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

library(ggplot2)  
  
 library(reshape2)

## Warning: package 'reshape2' was built under R version 3.3.2

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(varSelRF)

## Loading required package: parallel

############################################################################  
  
   
 data<-read.table("data.txt", header=T,sep="\t",row.names=1)  
   
 X<-data[,3:ncol(data)] # Sliced predictor set   
  
 x<-scale(X) # Pre-processing predictor set   
  
 y<-as.factor(data[,1]) # Response vector   
  
 dim(x) # Checking size of the matrix

## [1] 72 28

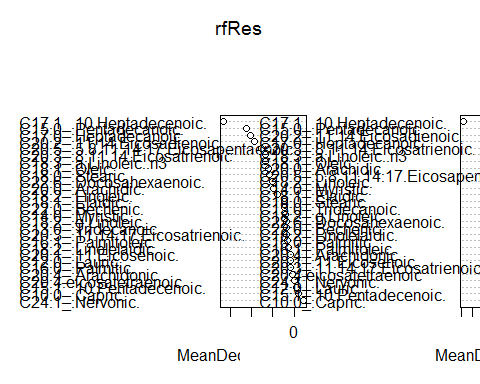
length(y) # Checking length of the Matrix

## [1] 72

##################################################  
  
# Random forest application in classification mode  
  
##################################################  
   
rfRes<-randomForest(x=x,y=y,importance=T)  
  
rfRes

##   
## Call:  
## randomForest(x = x, y = y, importance = T)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 55.56%  
## Confusion matrix:  
## C. C.R. High High R. Int. I. Int. II. Int. II. R. Low  
## C. 8 2 0 0 0 0 0 2  
## C.R. 5 1 0 0 0 0 0 0  
## High 0 0 8 0 0 1 0 0  
## High R. 3 0 0 1 0 0 0 1  
## Int. I. 0 0 0 0 5 3 0 4  
## Int. II. 0 0 2 0 7 2 0 0  
## Int. II. R. 3 1 0 0 0 0 1 1  
## Low 2 1 0 0 2 0 0 6  
## class.error  
## C. 0.3333333  
## C.R. 0.8333333  
## High 0.1111111  
## High R. 0.8000000  
## Int. I. 0.5833333  
## Int. II. 0.8181818  
## Int. II. R. 0.8333333  
## Low 0.4545455

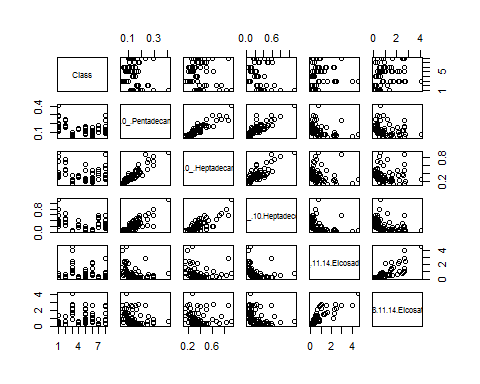
varImpPlot(rfRes) # Plotting all the predictors



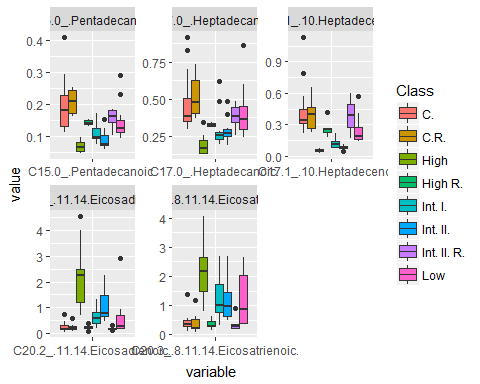
imp<-rfRes$importance  
  
OOB<-rfRes$err.rate\*100   
  
confusionMatrix<-rfRes$confusion  
  
# Storing all the results in the working directory  
  
write.csv(imp,"Imp\_Data.csv")  
  
write.csv(confusionMatrix,"confusionMatrix\_Data.csv")  
  
write.csv(OOB[500,1],"OOB\_error.csv")  
  
  
################### variable selection ##############################################  
  
 rf.vs1 <- varSelRF(x, as.factor(y), ntree = 500, ntreeIterat = 300,vars.drop.frac = 0.2)  
  
 rf.vs1

##   
## Backwards elimination on random forest; ntree = 500 ; mtryFactor = 1   
##   
## Selected variables:  
## [1] "C15.0\_.Pentadecanoic." "C17.0\_.Heptadecanoic."   
## [3] "C17.1\_.10.Heptadecenoic." "C20.2\_.11.14.Eicosadienoic."   
## [5] "C20.3\_.8.11.14.Eicosatrienoic."  
##   
## Number of selected variables: 5

selected<-rf.vs1$selected.vars  
  
 Class<-data[,1]  
  
 newDat<-cbind(Class,data[,selected])  
   
 plot(newDat)



############################# Some vizualization with selected data #################  
  
 df.m <- melt(newDat, id.var = "Class")  
  
 p <- ggplot(data = df.m, aes(x=variable, y=value)) + geom\_boxplot(aes(fill=Class))  
  
 p1<-p + facet\_wrap( ~ variable, scales="free")  
   
 p1



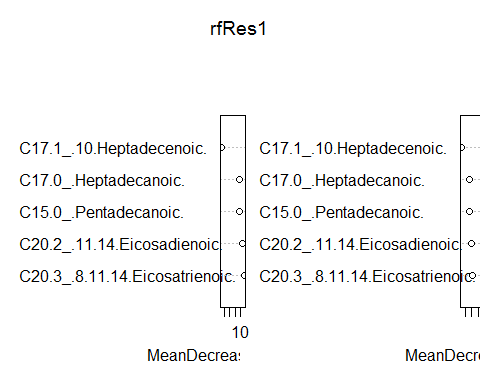
# Save all the results and plots in working directory  
  
 ggsave(p1, file="boxPlot with selected.pdf")

## Saving 5 x 4 in image

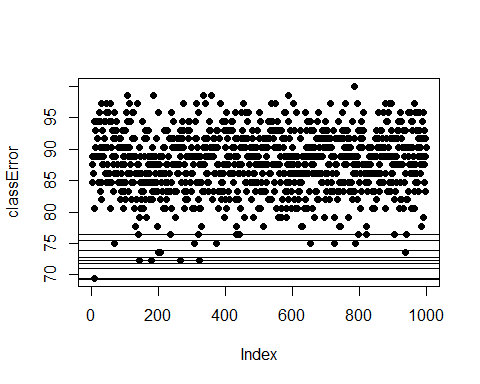
write.csv(selected,"selected.csv")  
  
 write.csv(data[,selected],"selected\_with\_data\_1\_2.csv")  
  
 write.csv(newDat,"selected\_with\_y.csv")  
  
  
########################################################################################  
  
## itarate the model with only selected variables and check how much variation is explained   
  
#######################################################################################  
  
  
rfRes1<-randomForest(x=data[,selected],y=y,importance=T)  
  
rfRes1

##   
## Call:  
## randomForest(x = data[, selected], y = y, importance = T)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 56.94%  
## Confusion matrix:  
## C. C.R. High High R. Int. I. Int. II. Int. II. R. Low  
## C. 7 2 0 0 0 0 1 2  
## C.R. 5 0 0 0 0 0 1 0  
## High 0 0 7 0 0 2 0 0  
## High R. 3 0 0 1 0 0 0 1  
## Int. I. 0 0 0 0 4 5 0 3  
## Int. II. 0 0 2 0 6 3 0 0  
## Int. II. R. 1 0 0 2 0 0 2 1  
## Low 1 2 0 0 0 0 1 7  
## class.error  
## C. 0.4166667  
## C.R. 1.0000000  
## High 0.2222222  
## High R. 0.8000000  
## Int. I. 0.6666667  
## Int. II. 0.7272727  
## Int. II. R. 0.6666667  
## Low 0.3636364

varImpPlot(rfRes1)



imp1<-rfRes1$importance  
  
OOB1<-rfRes1$err.rate\*100   
   
# Save all the values and results  
  
write.csv(imp1,"Imp\_Data\_after\_Selection.csv")  
  
write.csv(confusionMatrix,"confusionMatrix\_Data\_after\_Selection.csv")  
  
write.csv(OOB1[500,1],"OOB\_error\_after\_Selection.csv")  
  
  
########################################################  
##   
## permutation test for metabolomics data  
##  
########################################################  
  
 totiter<-1000  
  
 err<-matrix(data=NA,nrow=dim(X)[2],ncol=totiter)  
 RF\_Imp<-matrix(data=NA,nrow=dim(X)[2],ncol=totiter)  
# selected<-matrix(data=NA,nrow=dim(X)[2],ncol=totiter)  
 y2<-matrix(data=NA,nrow=dim(X)[1],ncol=totiter)  
 RF\_Imp1<-matrix(data=NA,nrow=dim(X)[2],ncol=totiter)  
 RF\_Imp2<-matrix(data=NA,nrow=dim(X)[2],ncol=totiter)  
  
 for(l in 1:totiter)  
 {  
  
 index<-l  
  
 #print(l)  
  
 y1<-sample(y) # randomize response variable  
  
 y2[,index]<-y1  
  
 mydata<-cbind(y1,X)  
  
 dim(mydata)  
  
 rfres<-randomForest(y=y1,x=X,data=mydata,importance=T,na.action=na.omit)  
  
 err[,index]<-rfres$err.rate[500,1][1]  
  
 imp<-rfres$importance  
 RF\_Imp1[,index]<-imp[,1]  
 RF\_Imp2[,index]<-imp[,2]  
  
 classError<-as.vector(t(err[1,]\*100))  
  
}   
   
 plot(classError, pch=19)  
  
 abline(h=OOB[500:1])



# Store all the values  
  
 write.csv(classError,"ClassError\_perm.csv")  
  
 write.csv(RF\_Imp1,"RF\_imp\_1\_perm.csv")  
  
 write.csv(RF\_Imp1,"RF\_imp\_2\_perm.csv")