Common Marmoset Gut Microbiome Profiles in Health and Intestinal Disease

Alex Sheh and Jose Molina Mora

September 21, 2020

Loading R data file to generate figure 2a, b and d. Includes the following data: *fig2a - bar and pie charts* fig2b - boxplots for 9 ASVs that distinguish strictures vs non-strictures *fig2d - dotplot for relative abundance of Clostridium sensu stricto 1 in the duodenum

```
load("fig2_sfig4_data.RData")
```

```
#for figure 2 images
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(reshape2)
require(scales)
```

```
## Loading required package: scales
```

```
# for ML algorithms
#BiocManager::install(c("gcrma"))
library(caret)
```

```
## Loading required package: lattice
```

```
library(ROCR)    # for ROC curve
library(rpart)   # for decision tree in case RF is selected
library(rattle)  # for dataset and decision tree
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(ellipse)
```

```
##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs
```

```r
library(ggfortify)
library(plotrix)
```

```
##
## Attaching package: 'plotrix'

## The following object is masked from 'package:scales':
##
##     rescale
```

```r
library(gcrma)
```

```
## Loading required package: affy

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
library(RColorBrewer)
library(kmed)
library(DescTools) #
```

```
##
## Attaching package: 'DescTools'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
```

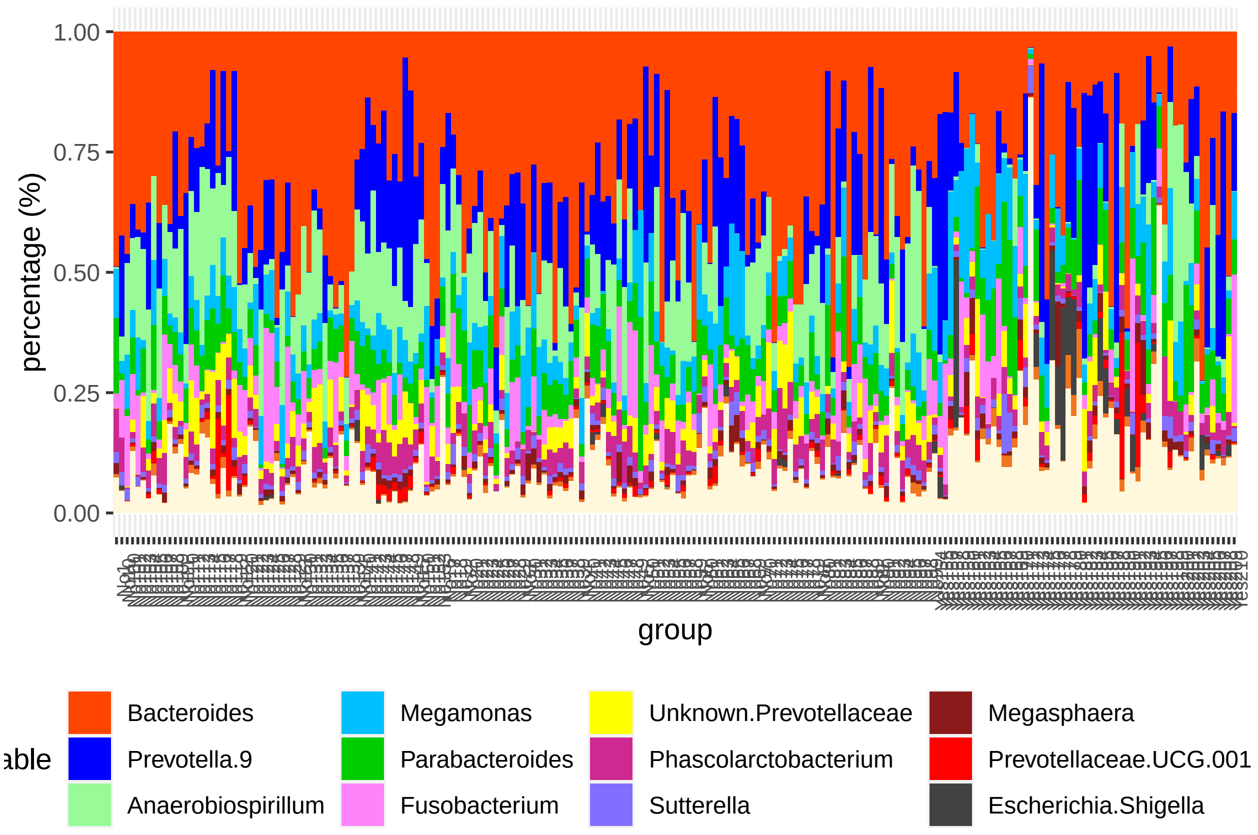
```
## other attached packages:
##  [1] DescTools_0.99.37   kmed_0.3.0          RColorBrewer_1.1-2
##  [4] gcrma_2.58.0        affy_1.64.0         Biobase_2.46.0
##  [7] BiocGenerics_0.32.0 plotrix_3.7-8       ggfortify_0.4.10
## [10] ellipse_0.4.2       rattle_5.4.0        bitops_1.0-6
## [13] tibble_3.0.3        rpart_4.1-15        ROCR_1.0-11
## [16] caret_6.0-86        lattice_0.20-38     scales_1.1.1
## [19] reshape2_1.4.4      ggplot2_3.3.2       dplyr_1.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.5            mvtnorm_1.1-1         lubridate_1.7.9
##  [4] tidyr_1.1.1           Biostrings_2.54.0     class_7.3-15
##  [7] digest_0.6.25         ipred_0.9-9           foreach_1.5.0
## [10] R6_2.4.1              plyr_1.8.6            stats4_3.6.3
## [13] evaluate_0.14         pillar_1.4.6          zlibbioc_1.32.0
## [16] rlang_0.4.7           Exact_2.0             rstudioapi_0.11
## [19] data.table_1.12.8     S4Vectors_0.24.4      Matrix_1.2-18
## [22] preprocessCore_1.48.0 rmarkdown_2.3         splines_3.6.3
## [25] gower_0.2.2           stringr_1.4.0         munsell_0.5.0
## [28] compiler_3.6.3        xfun_0.16             pkgconfig_2.0.3
## [31] htmltools_0.5.0       nnet_7.3-12           tidyselect_1.1.0
## [34] expm_0.999-5          gridExtra_2.3         prodlim_2019.11.13
## [37] IRanges_2.20.2        codetools_0.2-16      crayon_1.3.4
## [40] withr_2.2.0           MASS_7.3-51.6         recipes_0.1.13
## [43] ModelMetrics_1.2.2.2  grid_3.6.3            nlme_3.1-144
## [46] gtable_0.3.0          lifecycle_0.2.0       magrittr_1.5
## [49] pROC_1.16.2           stringi_1.4.6         XVector_0.26.0
## [52] affyio_1.56.0         timeDate_3043.102     ellipsis_0.3.1
## [55] generics_0.0.2        vctrs_0.3.1           boot_1.3-24
## [58] lava_1.6.7            iterators_1.0.12      tools_3.6.3
## [61] glue_1.4.1            purrr_0.3.4           survival_3.1-8
## [64] yaml_2.2.1            colorspace_1.4-1      BiocManager_1.30.10
## [67] knitr_1.29
```

Figure 2 stricture

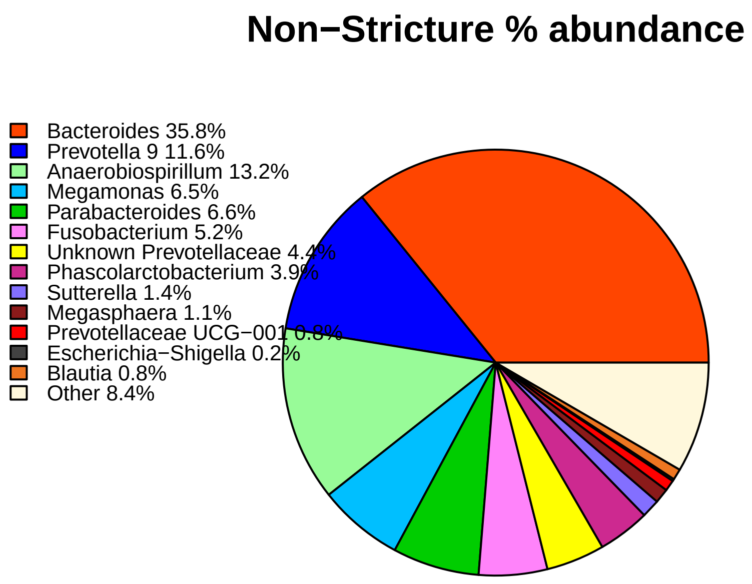
Figure 2a

Figure 2a creates a bar chart and a pie chart based on the relative abundances of genera for stricture progressors and non-progressors. Bar chart shows every individual sample while the pie chart presents an average based on stricture. Data is in variables “fig2a_bar” and “fig2a_pie”

```r
#figure 2a bar chart
fig2a.melt <- melt(fig2a_bar)
bar_2a <- ggplot(fig2a.melt, aes(x = Sample, y = value))
bar_2a <- bar_2a + geom_bar(aes(fill = variable), stat = "identity", width = 1) +
  labs(x = "group", y = "percentage (%)") +
  scale_fill_manual(values = c("orangered", "blue", "palegreen", "deepskyblue","green3",
                               "orchid1", "yellow", "maroon3", "slateblue1","firebrick4",
                               "red", "grey26", "chocolate2", "cornsilk")) +
  theme(axis.text.x = element_text(angle = 90,size=7), legend.position = "bottom")
bar_2a
```

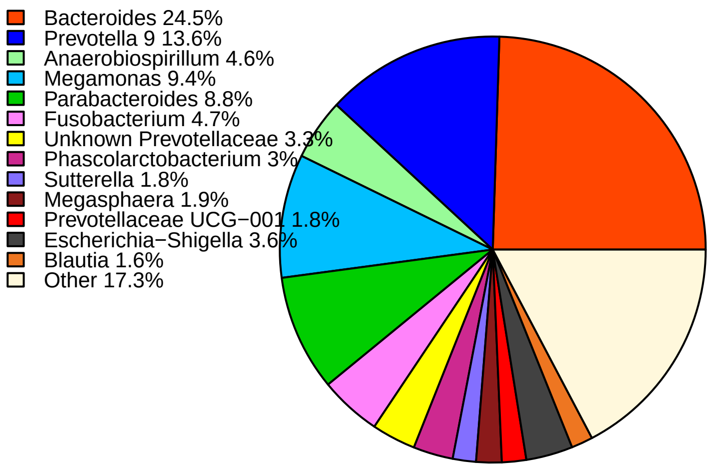


```
#stricture cases labeled as "YES" on the right side of figure

#figure 2a pie chart
non_stricturePCT<- round(100*fig2a_pie$Non.Stricture/sum(fig2a_pie$Non.Stricture), 1)
stricturePCT<- round(100*fig2a_pie$Stricture/sum(fig2a_pie$Stricture), 1)

#NON-stricture pie chart
pie(fig2a_pie$Non.Stricture,labels = "", radius = 0.9,
    main = "Non-Stricture % abundance", col = c("orangered", "blue", "palegreen",
                                               "deepskyblue","green3", "orchid1",
                                               "yellow", "maroon3", "slateblue1",
                                               "firebrick4", "red", "grey26",
                                               "chocolate2", "cornsilk"))
legend("topleft", cex = 0.7, y.intersp=0.75, bty = "n",
       legend = paste0(fig2a_pie$X," ", non_stricturePCT, "%"),
       fill = c("orangered", "blue", "palegreen", "deepskyblue",
                "green3", "orchid1", "yellow", "maroon3", "slateblue1",
                "firebrick4", "red", "grey26", "chocolate2", "cornsilk"))
```



```r
#Stricture pie chart
pie(fig2a_pie$Stricture,labels = "", radius = 0.9, main = "Stricture % abundance",
    col = c("orangered", "blue", "palegreen", "deepskyblue","green3", "orchid1",
            "yellow", "maroon3", "slateblue1","firebrick4", "red", "grey26",
            "chocolate2", "cornsilk"))
legend("topleft", cex = .7, y.intersp=0.75, bty = "n",
       legend = paste0(fig2a_pie$X," ", stricturePCT, "%"),
       fill = c("orangered", "blue", "palegreen", "deepskyblue",
                "green3", "orchid1", "yellow", "maroon3",
                "slateblue1","firebrick4", "red", "grey26",
                "chocolate2", "cornsilk"))
```

Stricture % abundance



## Figure 2b Figure 2b creates boxplots for the 9 ASVs identified as important in the random forest model to distinguish strictures vs non-strictures. RF model is run in full in section for "Fig 2c and Supp. Fig. 2B and 2C".

```
fig2b.melt <- melt(fig2b)
fig2b_boxplot <- ggplot(fig2b.melt, aes(x=Class, y=value, fill=variable)) +
  geom_boxplot() +
  facet_wrap(~variable, scale = "free")
fig2b_boxplot
```

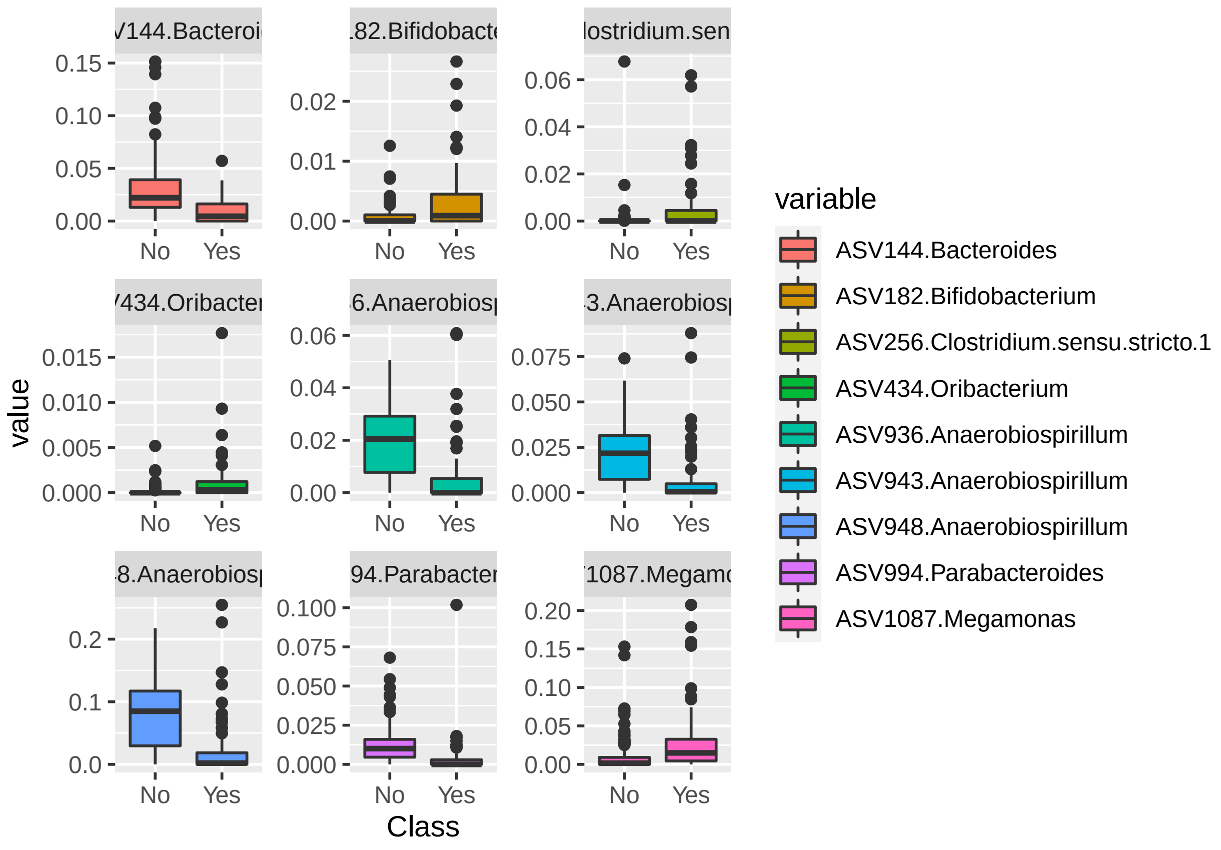


Figure 2d

Figure 2d creates the dot plot based on the relative abundances of Clostridium sensu stricto 1 in duodenum of stricture progressors and non-progressors. Data is in variable "fig2d"

```
figure_2d<- ggplot(fig2d, aes(x=Dx, y=PCT, fill=Dx)) +
  geom_dotplot(binaxis = 'y', stackdir = 'center') +
  stat_summary(fun=mean, geom="point", shape=18, size=6, color="black") +
  theme_classic() +
  theme(legend.position = "none",axis.text = element_text(size = 12),
        axis.title = element_text(size = 14, face = "bold")) +
  scale_y_continuous(labels = percent) +
  labs(x="", y = "% abundance in Duodenum")
figure_2d
```

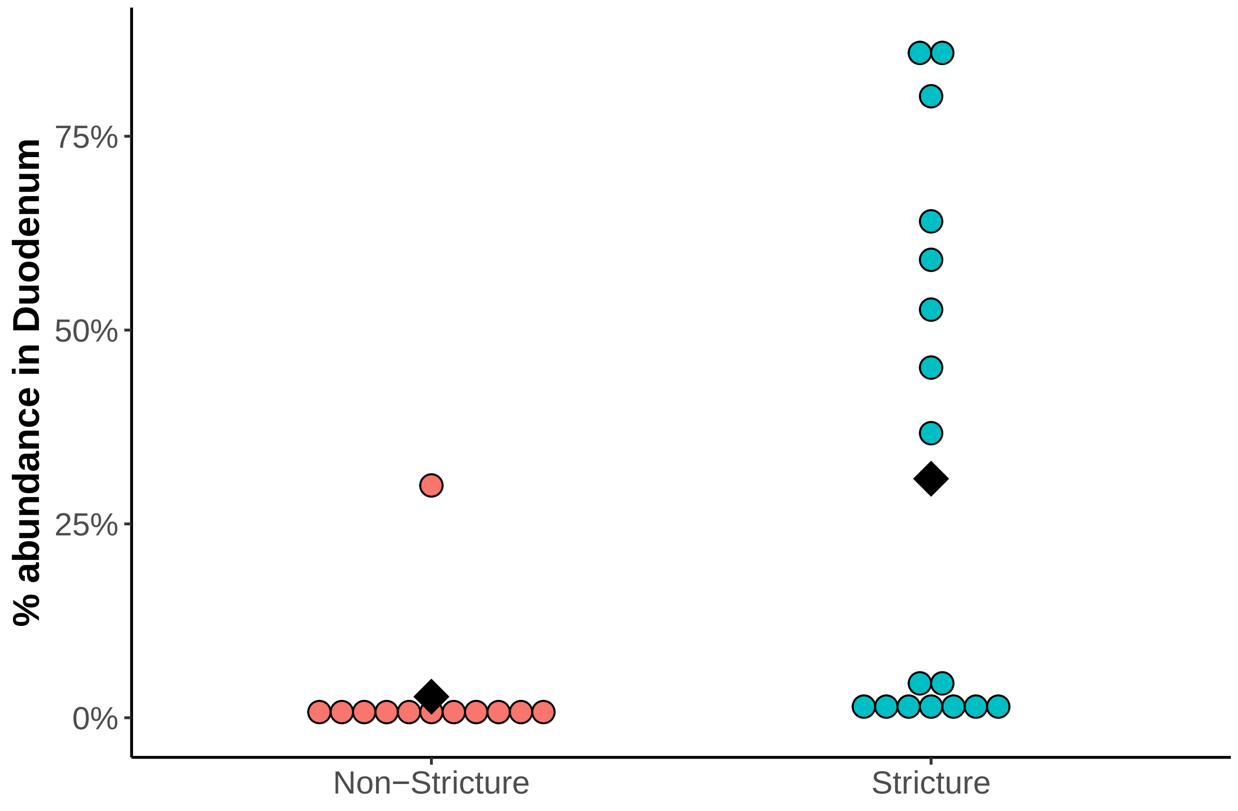


Figure 2c and Supplemental Figure 4B-I

Machine learning algorithm originally developed by Jose Molina Mora and modified by Alex Sheh. Data normalized by min-max normalization.

Figure 2c is a composite of the Receiver Operating Curves for OTU data, serum chemistry data and complete blood count data.

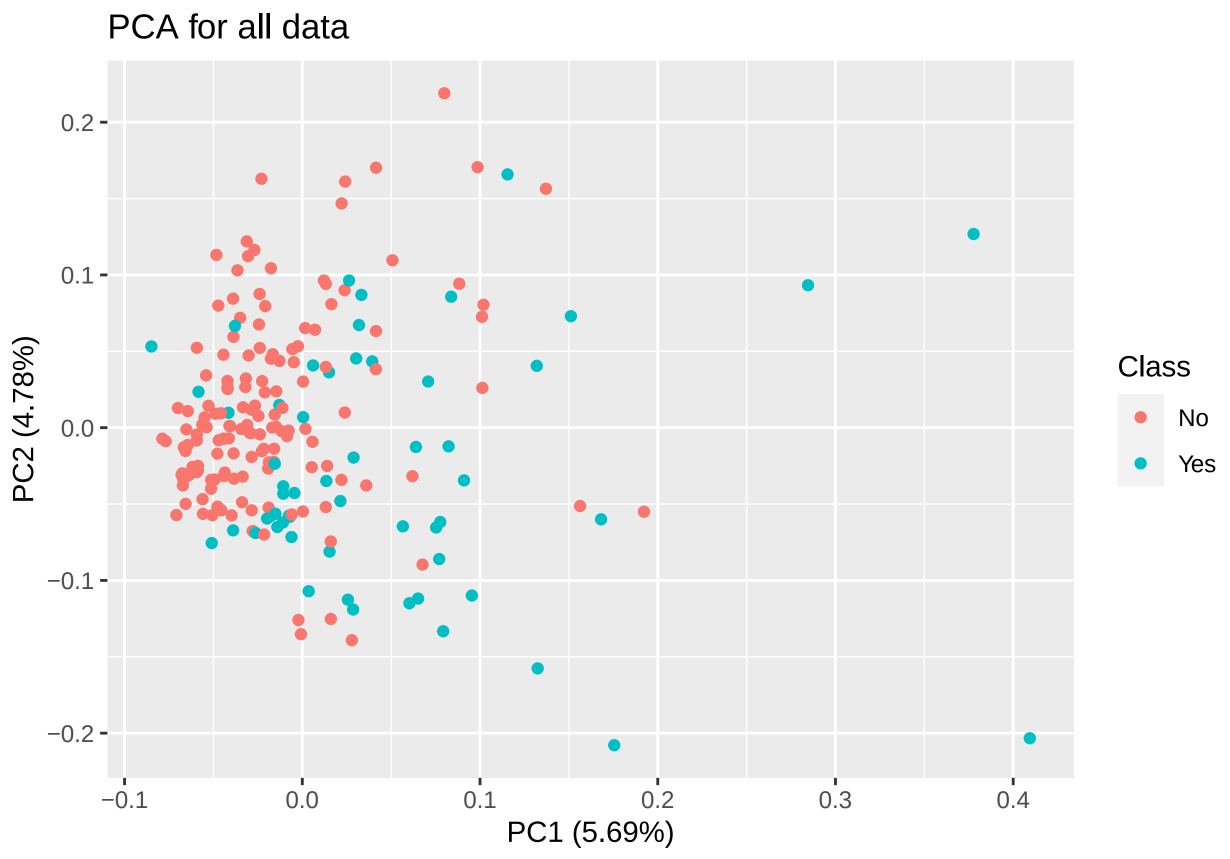
Supplemental Figure 4a was created by QIIME2. SFig 4B and C based on microbiome data in dataset "str_otu" with metadata in "str_meta" SFig 4D, E and F based on serum chemistry data SFig 4B and C based on CBC data

Start Ranking algorithm section for microbiome

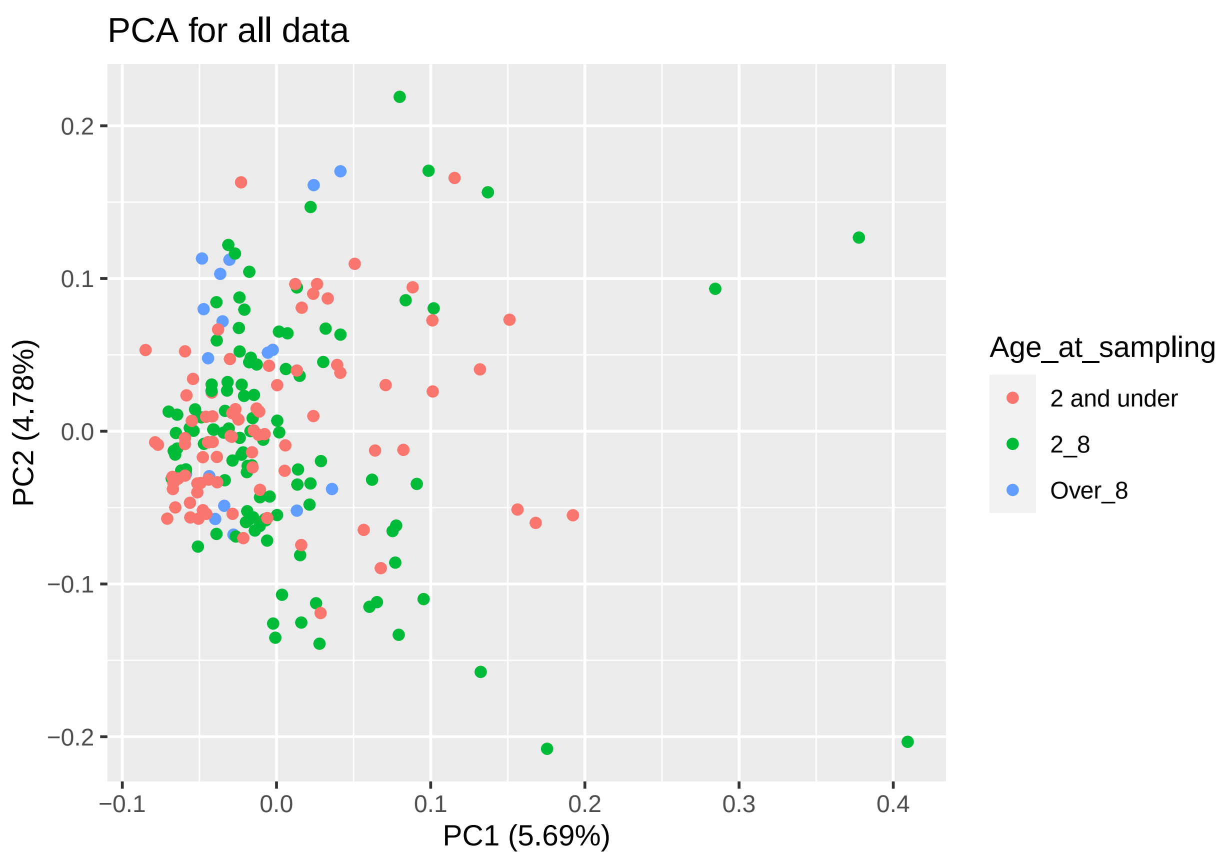
```
#ANALYSIS OF RANKING ALGORITHMS AND PARAMETER SELECTION
#Developed by Jose Molina Mora. Modified by Alex Sheh
# random forest models modeling stricture data based on microbiome

##########function minmax
minmax<-function(mat){
  mat.max = apply(mat,2,max)
  mat.min = apply(mat,2,min)
  for (i in 1:dim(mat)[2]){
    mat[,i]<-(mat[,i]-mat.min[i])/(mat.max[i]-mat.min[i])
  }
  #CHANGE NAs to 0
  mat[is.na(mat)] <- 0
  #REMOVE the NAs
  #mat0 <- (colSums(mat, na.rm=T) != 0) #some OTUs not present in this data subset
  #mat <- mat[, mat0] # all the non-zero columns
  return(mat)
}
###############################
```

evaluate the raw data prior to applying any algorithms

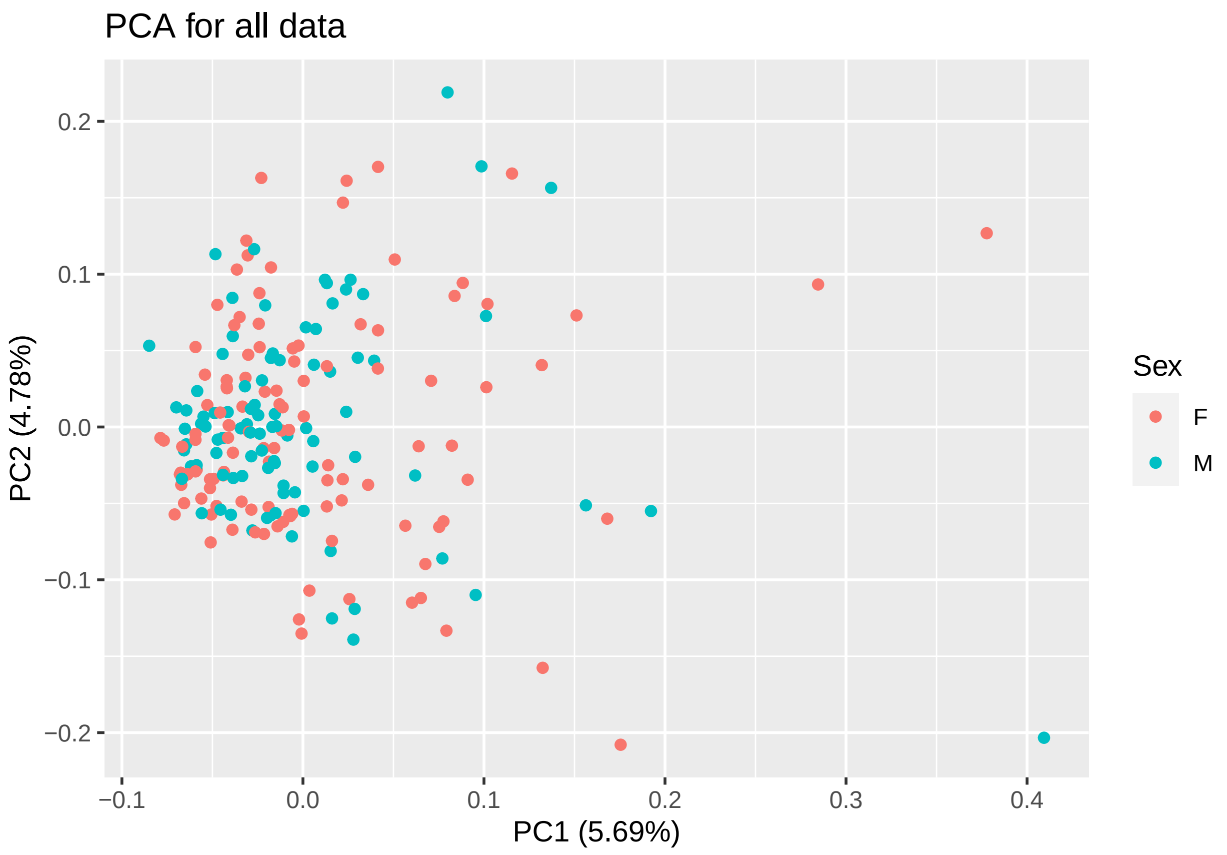
```
# PART 1. LOAD DATA and metadata. Visualization of data
Data <- minmax(str_otu)
conditions <- str_meta

##01_PCA - Class (Yes = Stricture and No = non-stricture)
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Class', main ="PCA for all data")
```



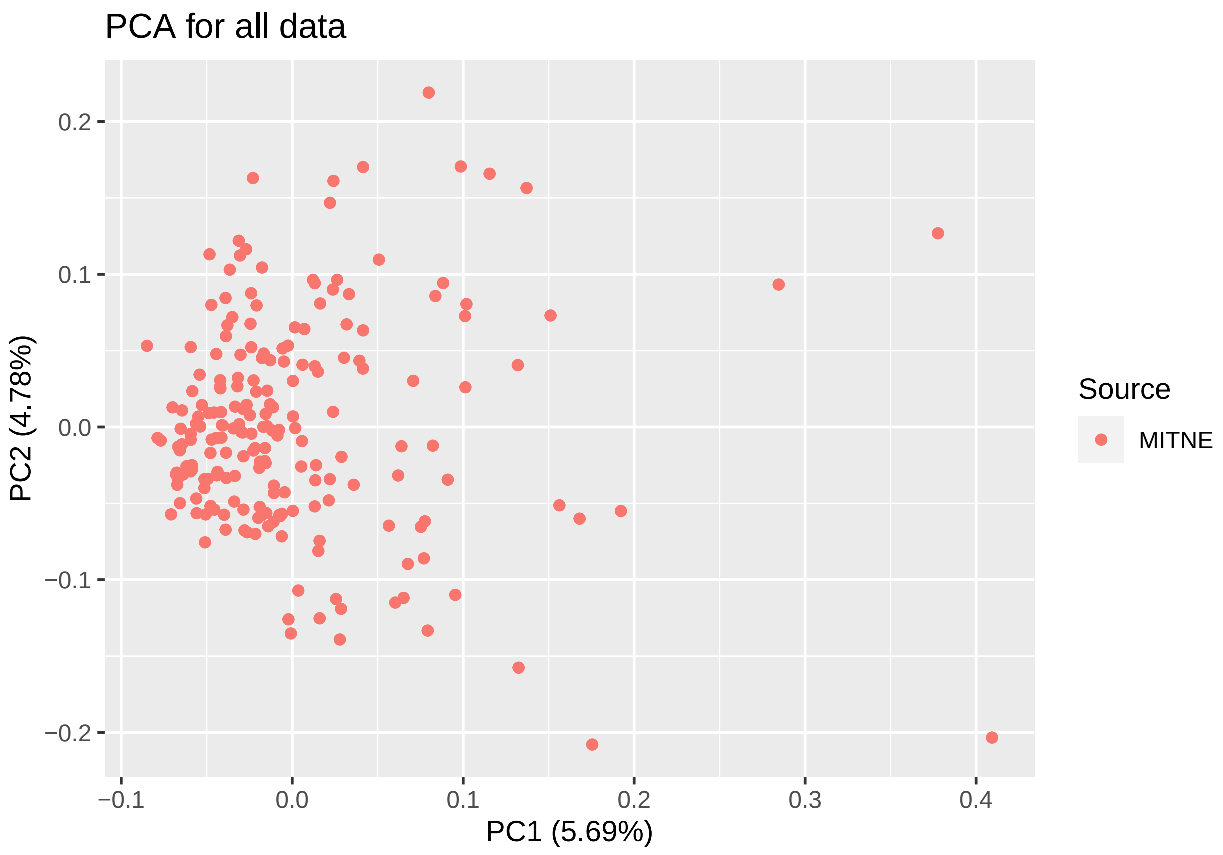
```
## sub-PCA plots using metadata
## 01-1_PCA - Age at sampling
autoplot(pca, data = conditions, colour = 'Age_at_sampling', main ="PCA for all data")
```



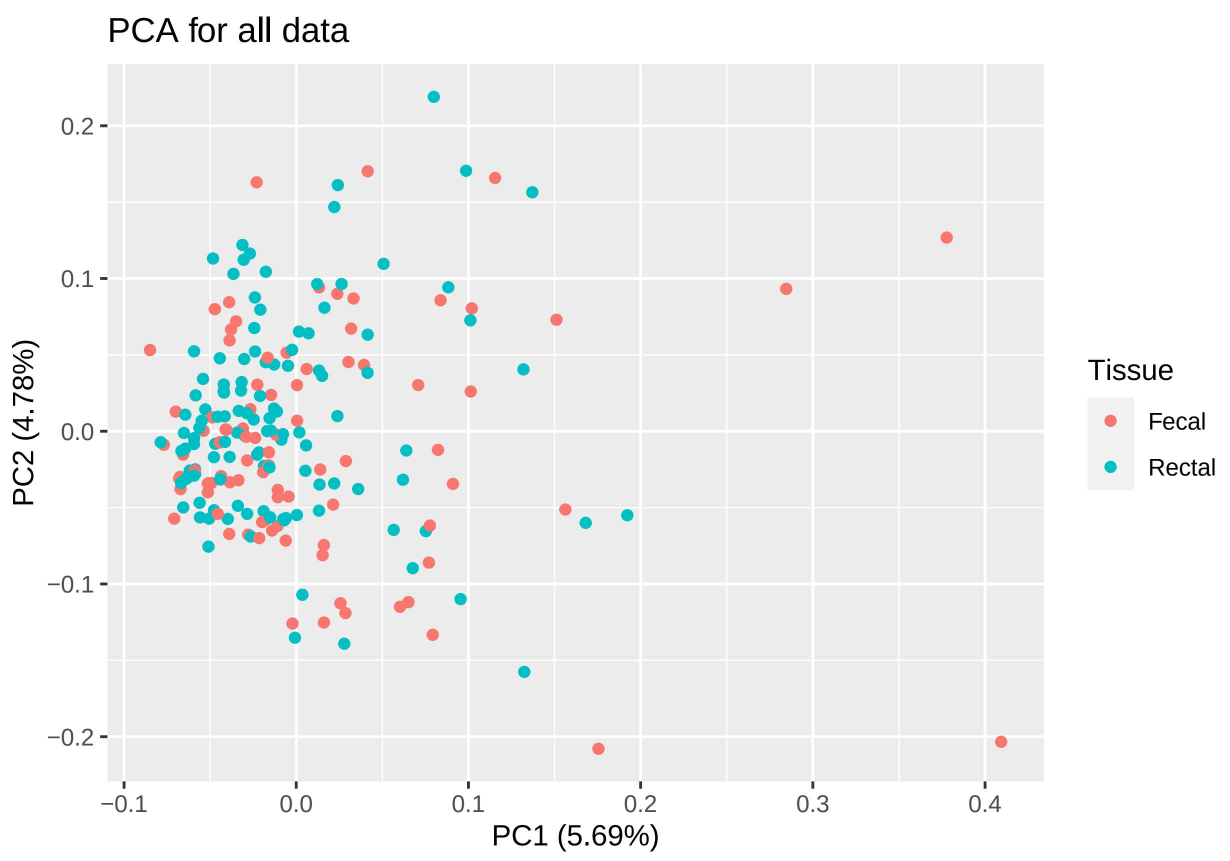
```
## 01-2_PCA - Sex
autoplot(pca, data = conditions, colour = 'Sex', main ="PCA for all data")
```



```
## 01-3_PCA - Source
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```

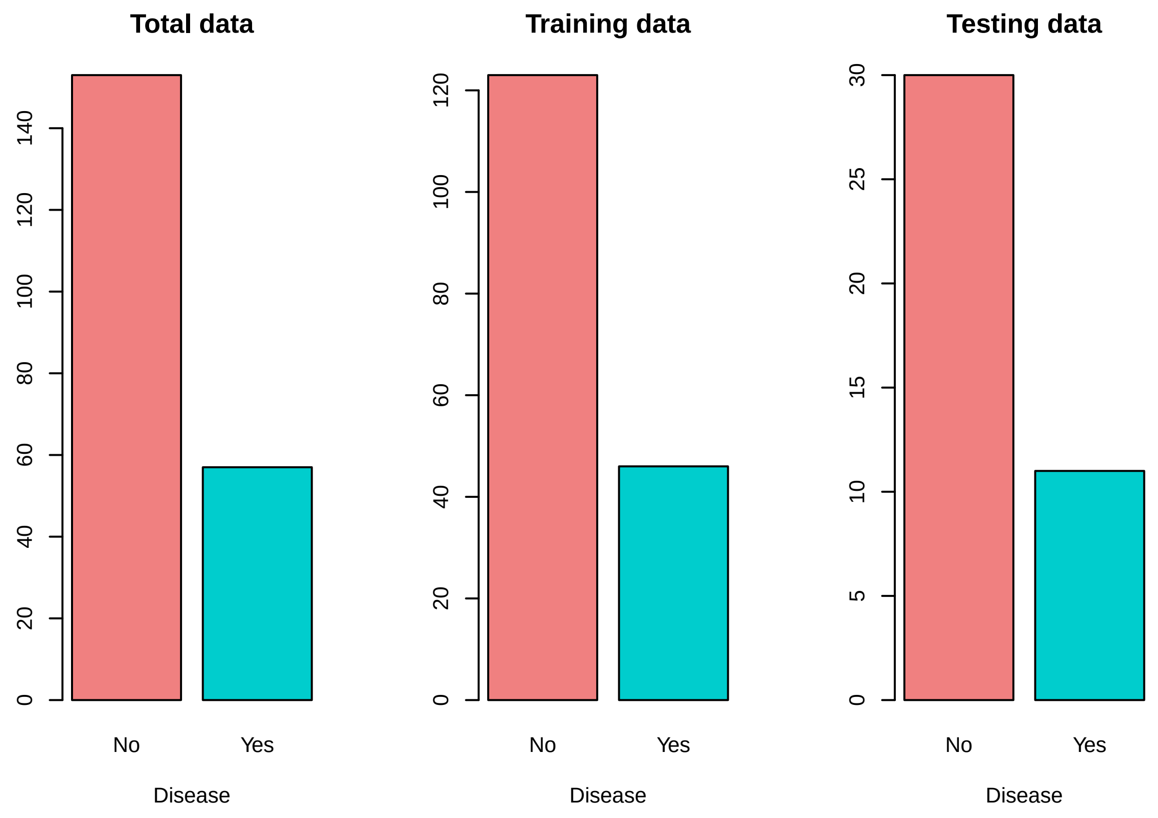


```
## 01-4_PCA - Tissue type
autoplot(pca, data = conditions, colour = 'Tissue', main ="PCA for all data")
```



```
# Split the data into training (80%) and testing (20%) sets
# set.seed(2)
test_index<-createDataPartition(conditions$Class, p=0.80, list = FALSE)

#Using D training to select features and then tested on Dtesting
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]

Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

#03_distribution of data by class in the entire dataset, the training set and testing set
par(mfrow=c(1,3))

group1<-conditions$Class
conteos1 <- table(group1)
barplot(conteos1, main="Total data",
        xlab="Disease", col=c("lightcoral","cyan3"))
        #, legend = c("No","Yes"))

group2<-Conditrain$Class
conteos2 <- table(group2)
barplot(conteos2, main="Training data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```

```
group3<-Conditesting$Class
conteos3 <- table(group3)
barplot(conteos3, main="Testing data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```



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

# Distribution in training set
percentage <- prop.table(table(Conditrain$Class)) * 100
# cbind(freq=table(Conditrain$Class), percentage=percentage)

#Distribution in testing set
percentage2 <- prop.table(table(Conditesting$Class))*100
# cbind(freq=table(Conditesting$Class),percentage=percentage2)

# PART 2. DISTRIBUTION OF VARIABLES BY CLASS
# Performed on training data but could be done on entire set or testing set
# split input and output
x <- Dtraining
y <- Conditrain$Class # class
sx <- Dtraining[,c(1:6)]
```

Set the seed

NOTE: this seed should be varied to test the robustness of results. Different seeds may alter the exact results in terms of accuracy, number of variables of importance, AUC, etc. Running the rest of the analysis over multiple conditions can help find the best algorithm for your analysis.

In our case we selected RF as it was consistently the best over multiple datasets but under certain conditions, SVM or other algorithms could also be considered as good classifiers

```
set.seed(2)
```

Apply the algorithms

```
#ALGORITHMS
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10, classProbs=TRUE)
metric <- "Accuracy"

#Classification algorithms
# a) linear algorithms
#fit.lda <- train(Dtraining, Conditrain$Class, method="lda", metric=metric,
#            trControl=control)

# b) nonlinear algorithms
# CART
fit.cart <- train(Dtraining, Conditrain$Class, method="rpart", metric=metric, trControl=control)
# kNN
fit.knn <- train(Dtraining, Conditrain$Class, method="knn", metric=metric,
                 trControl=control)
# c) advanced algorithms
# SVM
fit.svm <- train(Dtraining, Conditrain$Class, method="svmRadial", metric=metric,
                 trControl=control)
# Random Forest
fit.rf <- train(Dtraining, Conditrain$Class, method="rf", metric=metric,
                trControl=control)
```

SUPPLEMENTAL FIGURE 4B - compare multiple algorithms to see which classify the data best

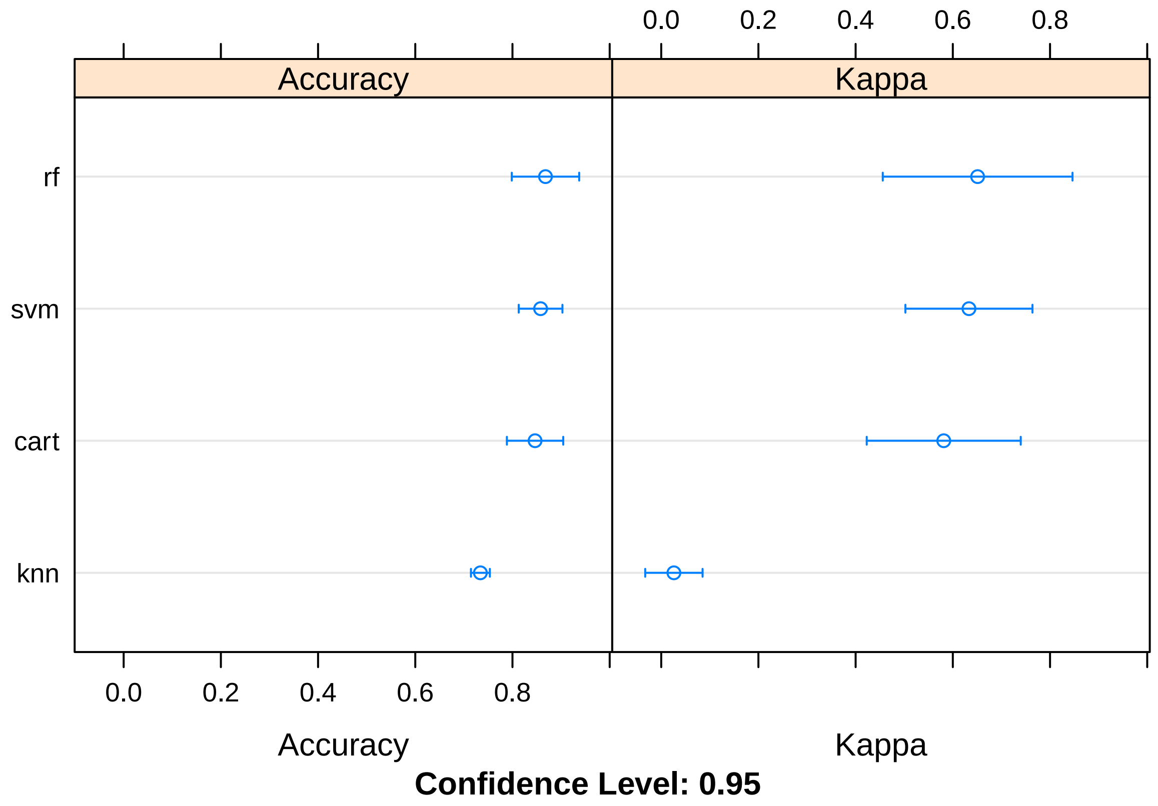
This is the type of plot used for Supplemental Figure 4B

```
##Supplemental Figure 4b - Classification
#summarize accuracy of models
results <- resamples(list(cart=fit.cart,svm=fit.svm,knn=fit.knn,rf=fit.rf,rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
```

```
##
## Models: cart, svm, knn, rf, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf   0.6875000 0.8235294 0.8856209 0.8679330 0.9411765 1.0000000    0
## cart 0.7500000 0.7794118 0.8284314 0.8464869 0.8854167 1.0000000    0
## svm  0.7777778 0.8152574 0.8492647 0.8578431 0.8823529 1.0000000    0
## knn  0.7058824 0.7058824 0.7361111 0.7339869 0.7610294 0.7647059    0
##
## Kappa
##            Min.   1st Qu.    Median       Mean   3rd Qu.      Max. NA's
## rf   0.09090909 0.5611107 0.7011611 0.65104143 0.8424313 1.0000000    0
## cart 0.34615385 0.4426407 0.4877073 0.58130452 0.7150685 1.0000000    0
## svm  0.36842105 0.4905154 0.6357511 0.63318163 0.7160714 1.0000000    0
## knn  0.00000000 0.0000000 0.0000000 0.02608696 0.0000000 0.2608696    0
```

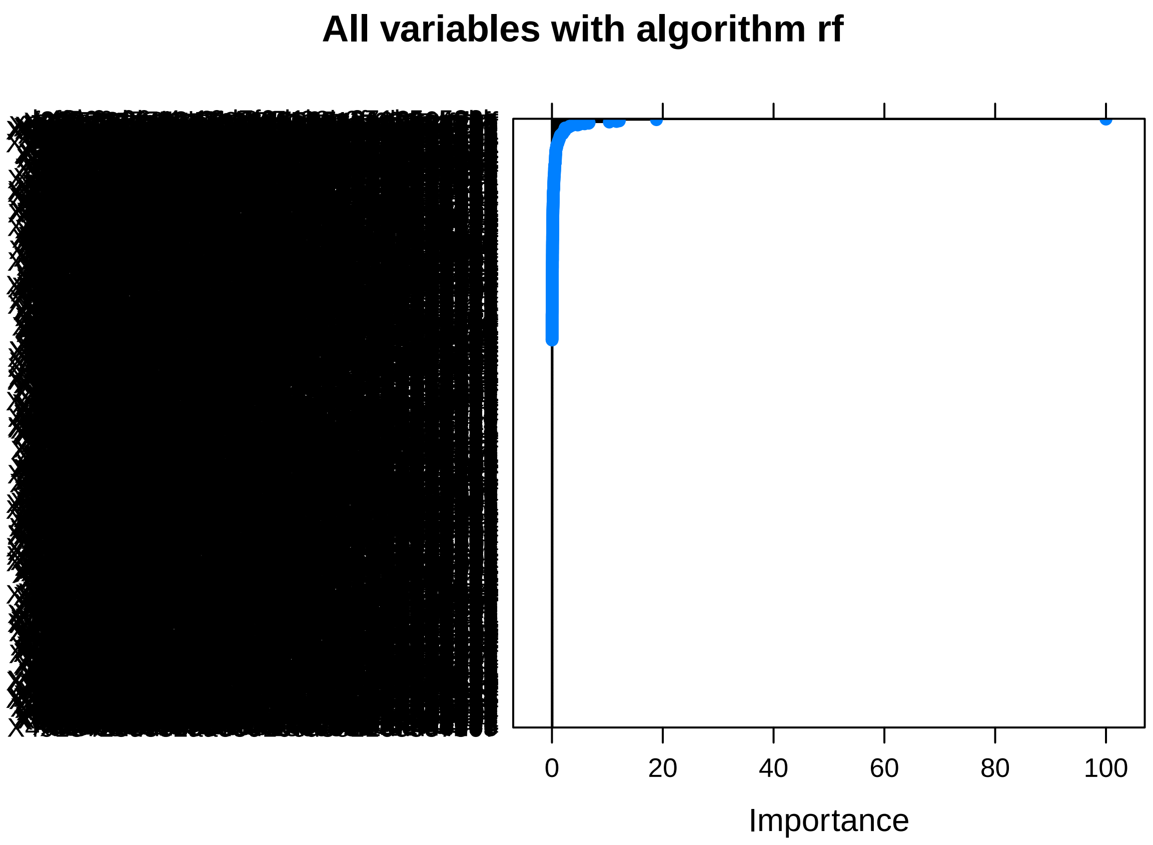
```
# compare accuracy of models
dotplot(results)
```



## RF performed best in most scenarios but SVM also had good performance with this data

```
#SELECT THE RF ALGORITHM BASED ON RESULTS
#model=fit.knn
#kalgoritmo="knn"
```

```
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
kalgoritmo="rf"

importance <- varImp(model, scale=TRUE)
# head(importance)

#05_import_all
#Graph importance
plot(importance, main = paste("All variables with algorithm", kalgoritmo))
```

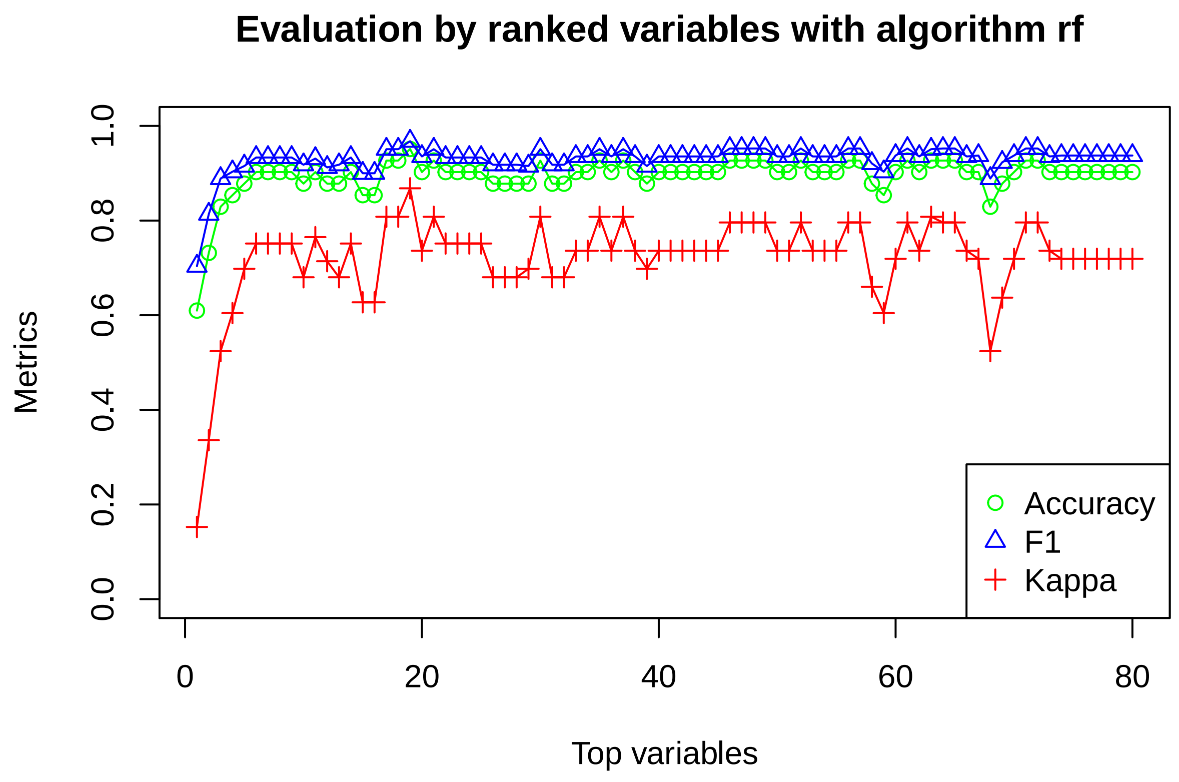


```
#INDEX
IndexRank <-data.frame(sort(importance$importance$Overall,
                            index.return = TRUE, decreasing = TRUE)[2])
#For SVM and KNN importance$importance$Control
#For RF/cart use $Overall (RF only Overall)
#IndexRank <-data.frame(sort(importance$importance$Yes, index.return = TRUE,
#           decreasing = TRUE)[2])
Ranking<-t(IndexRank)

DtrainingRanked<-Dtraining[,Ranking[1,]]
DtestingRanked<-Dtesting[,Ranking[1,]]
DataRanked<-Data[,Ranking[1,]]
#write.csv(DataRanked, "All_data_ranked.csv")
```

```r
#EVALUATE TOP VARIABLES
kvalue=80
kEvaluacion<-matrix(,nrow=kvalue, ncol=19) # 18 parameters to calculate
colnames(kEvaluacion)<-c("Accuracy","Kappa","AccuracyLower","AccuracyUpper",
                         "AccuracyNull", "AccuracyPValue","McnemarPValue",
                         "Sensitivity", "Specificity", "Pos Pred Value",
                         "Neg Pred Value", "Precision", "Recall","F1",
                         "Prevalence" ,"Detection Rate",
                         "Detection Prevalence","Balanced Accuracy", "K")

# For k=1 it is calculated separately to define vector
k=1
DtrainK<-as.data.frame(DtrainingRanked[,1])
colnames(DtrainK)<-colnames(DtrainingRanked)[1]
DtestK<-as.data.frame(DtestingRanked[,1])
colnames(DtestK)<-colnames(DtestingRanked)[1]

fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                    trControl=control)
predictionsK <- predict(fit.algorK, DtestK)
StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
kEvaluacion[1,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))


for (k in 2:kvalue){
  DtrainK<-DtrainingRanked[,c(1:k)]
  DtestK<-DtestingRanked[,c(1:k)]
  fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                      trControl=control)
  predictionsK <- predict(fit.algorK, DtestK)
  StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
  kEvaluacion[k,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))
}

# StatisticsK$byClass
EvalK<-as.data.frame(kEvaluacion)
```

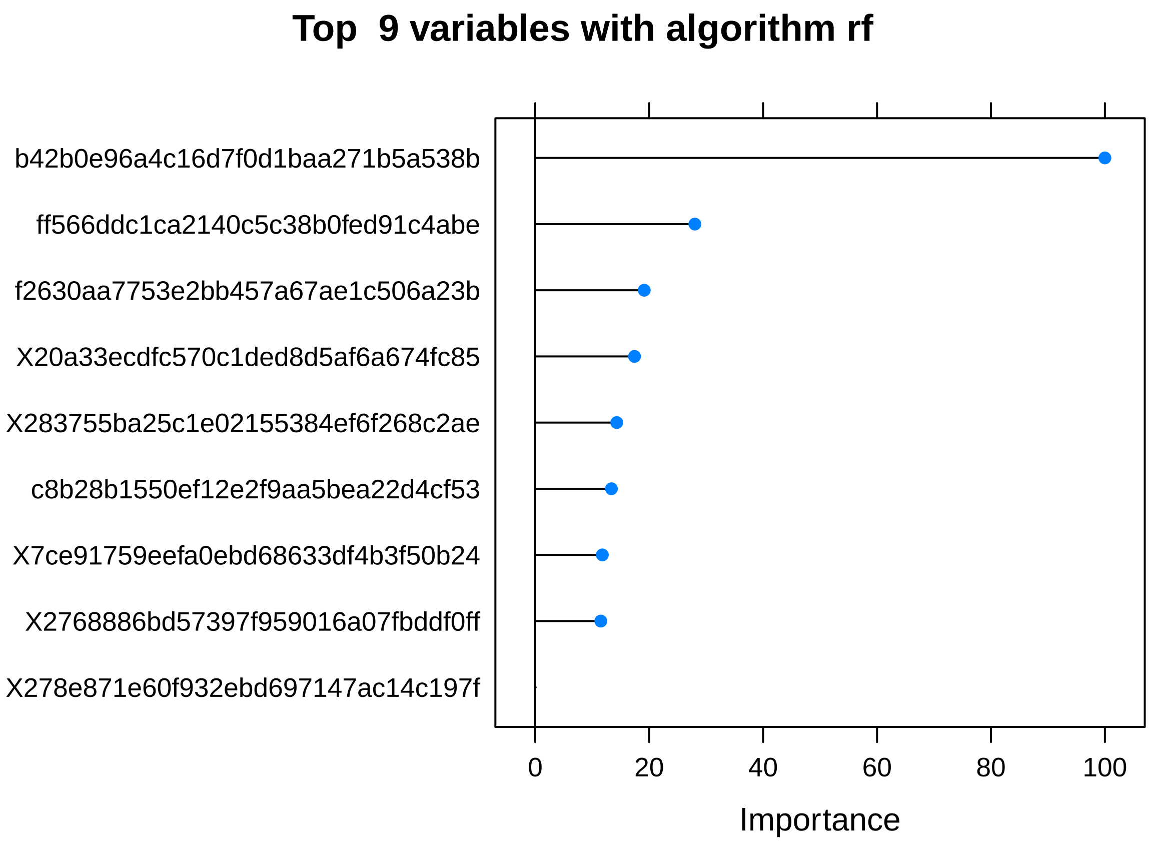
Supplemental Figure 4C

```r
#11_Metrics_per_top
par(mfrow = c(1,1))
plot(EvalK$Accuracy,type="o",pch=1,col="green",
     main = paste("Evaluation by ranked variables with algorithm",
                  kalgoritmo,sed=""),xlab = "Top variables",
                  ylab = "Metrics", ylim=c(0,1))
lines(EvalK$F1,type = "o", pch=2, col="blue")
lines(EvalK$Kappa,type = "o", pch=3, col="red")
legend("bottomright",legend=c("Accuracy","F1","Kappa"),pch=c(1,2,3),
       col=c("green","blue","red"))
```

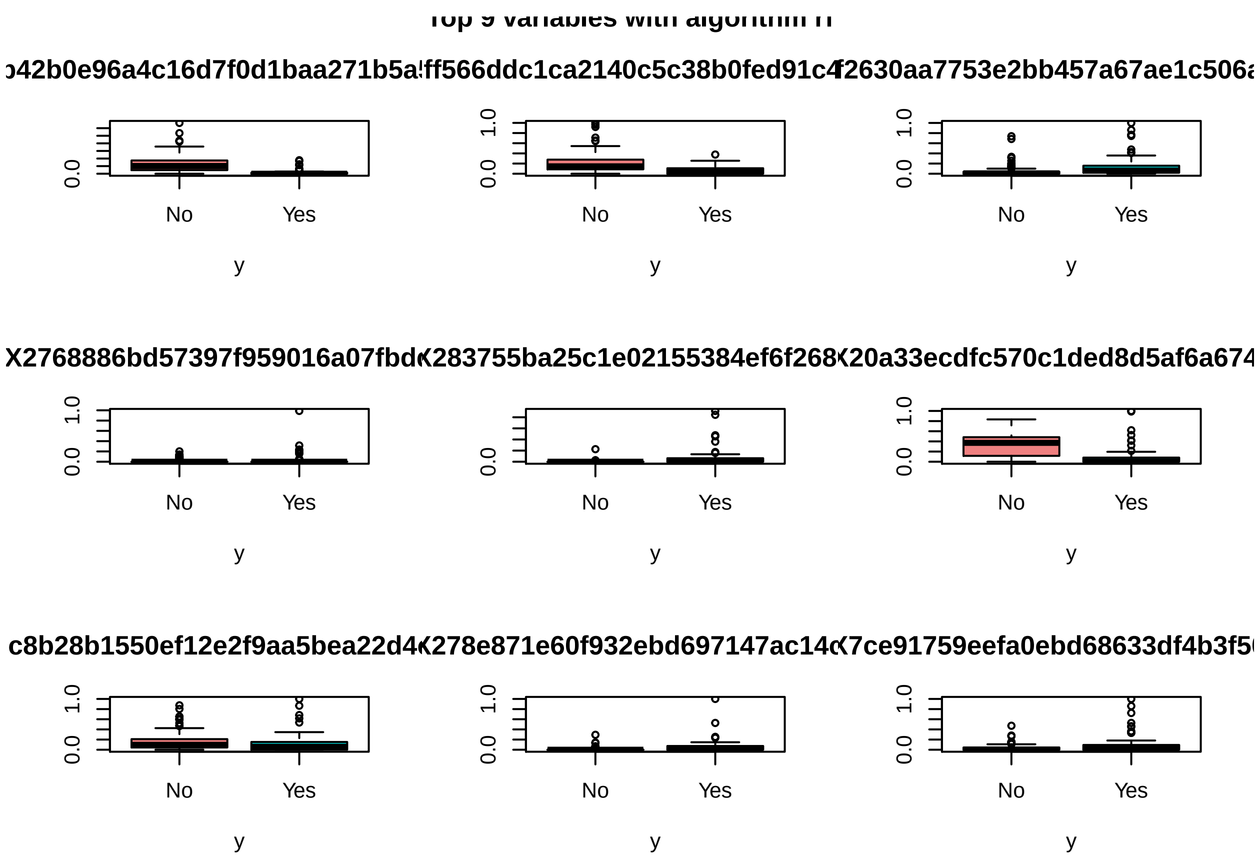


## for the seed set the number of variables is 9 when the accuracy reaches the plateau Here we visualize what using only the 9 variables will look like using boxplots, PCA, etc. as we did in the begining.

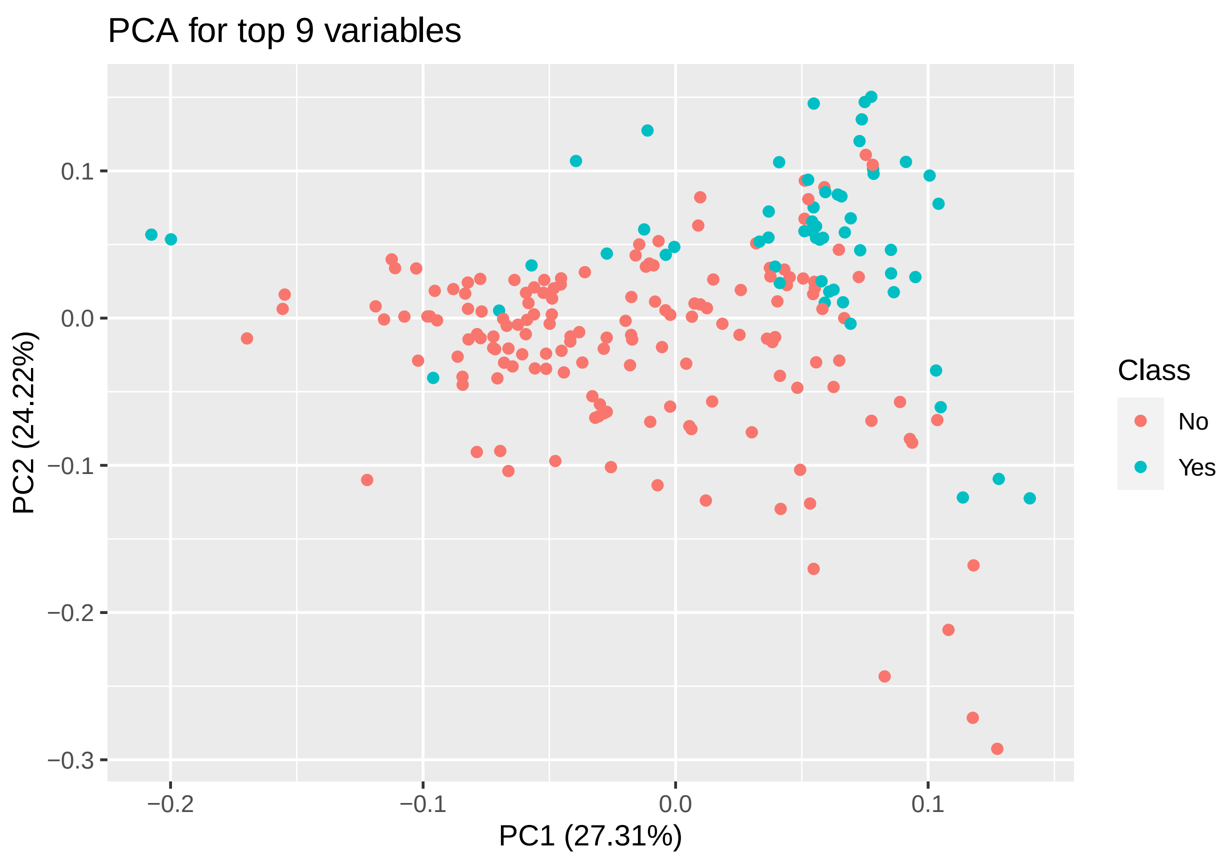
```
#For chosen value of K

ks=9
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]

DataRankedtopK<-DataRanked[,c(1:ks)]
# write.csv(DataRankedtopK, paste0("DataRankedtop",ks,"-",kalgoritmo,".csv"))

set.seed(3)
fit.algorKS <- train(DtrainKS, Conditrain$Class, method=kalgoritmo, metric=metric,
                     trControl=control)
importance <- varImp(fit.algorKS, scale=TRUE)
plot(importance, main = paste('Top ', ks, "variables with algorithm", kalgoritmo))
```



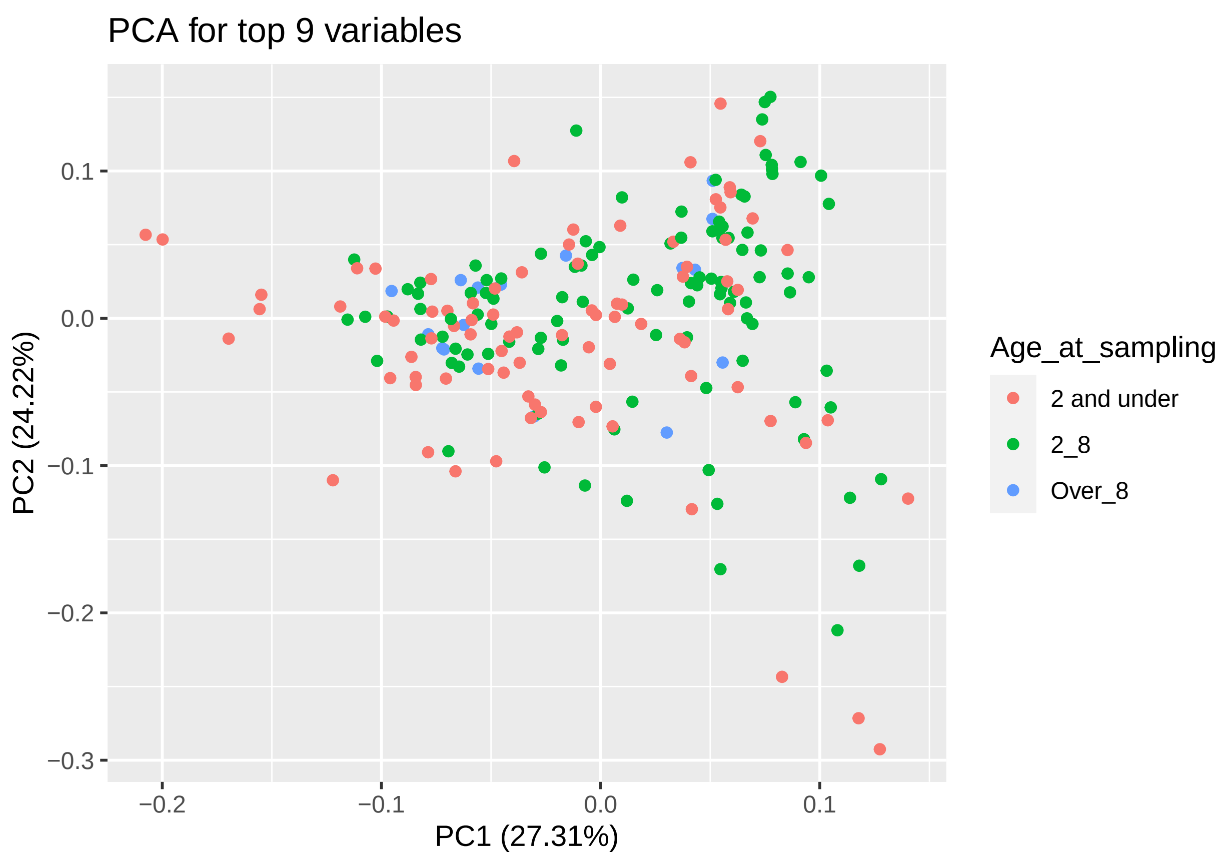
```
predictionsKS <- predict(fit.algorKS, DtestKS)
StatisticsKS<-confusionMatrix(predictionsKS, Conditesting$Class)
StatisticsKS
StatisticsKS$overall
StatisticsKS$byClass

xR <- DtrainKS
yR <- Conditrain$Class
sxR <- DtrainingRanked[,c(4,5,6,1,2,3)]

#boxplot
par(mfrow=c(3,3))

for(i in 1:ks) {
  boxplot(xR[,i]~y, main=names(xR)[i],col=c("lightcoral","cyan3"),ylab = " ")
}
title(paste("Top 9 variables with algorithm", kalgoritmo), outer=TRUE)
```



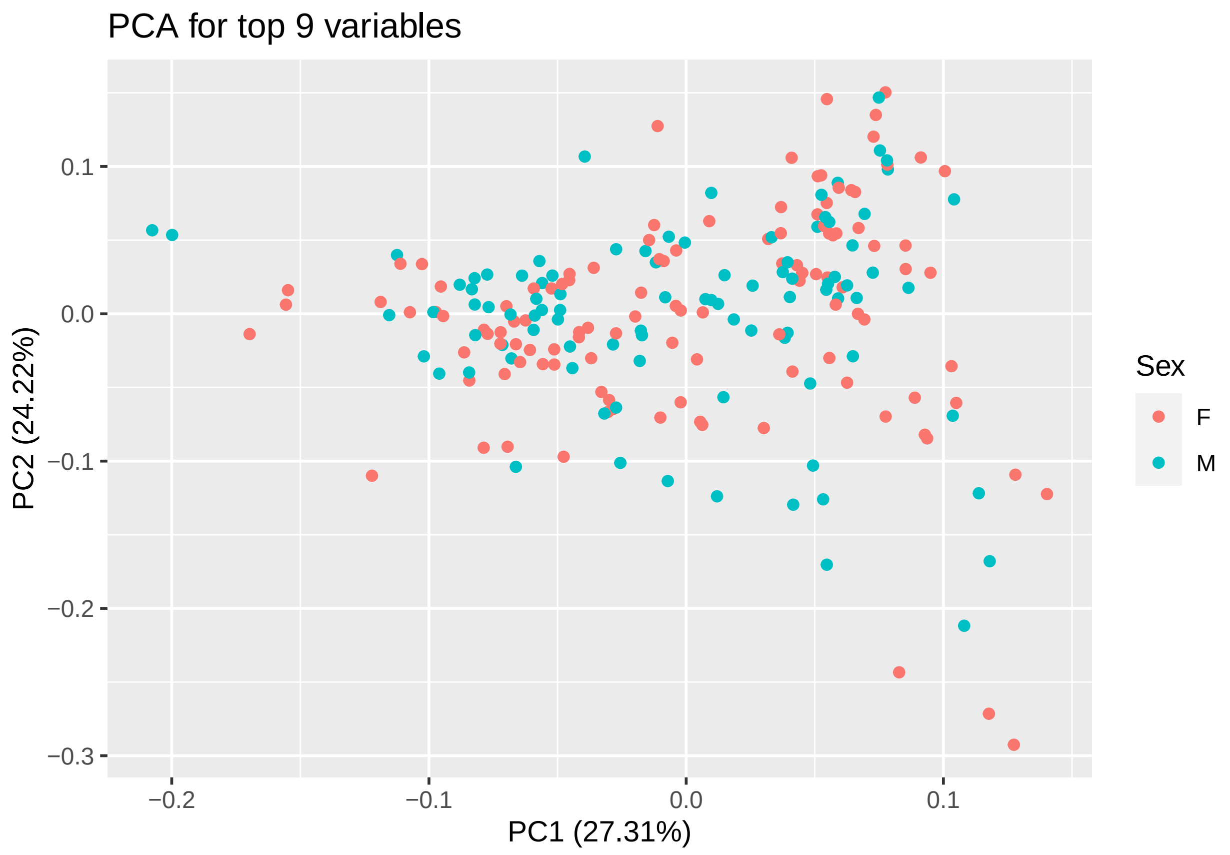
```r
par(mfrow=c(1,1))

pca<-prcomp(DataRanked[,1:ks])
#12_a_PCA-topK_Class
autoplot(pca, data = conditions, colour = 'Class', main =paste("PCA for top",
                                                              ks, "variables"))
```



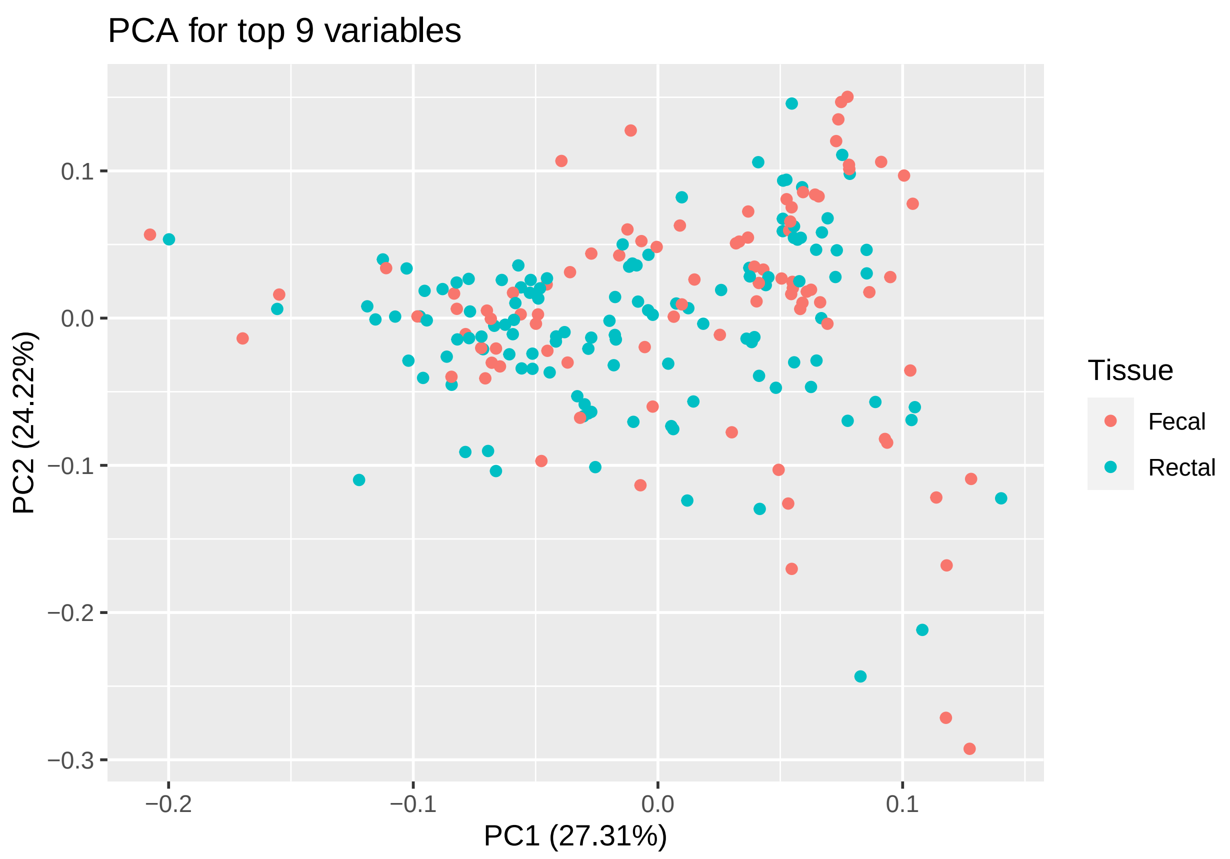
```
#12-b_PCA-topK_age
autoplot(pca, data = conditions, colour = 'Age_at_sampling',
         main =paste("PCA for top",ks, "variables"))
```



```
#12-c_PCA-topK_sex
autoplot(pca, data = conditions, colour = 'Sex', main =paste("PCA for top",
                                                             ks, "variables"))
```

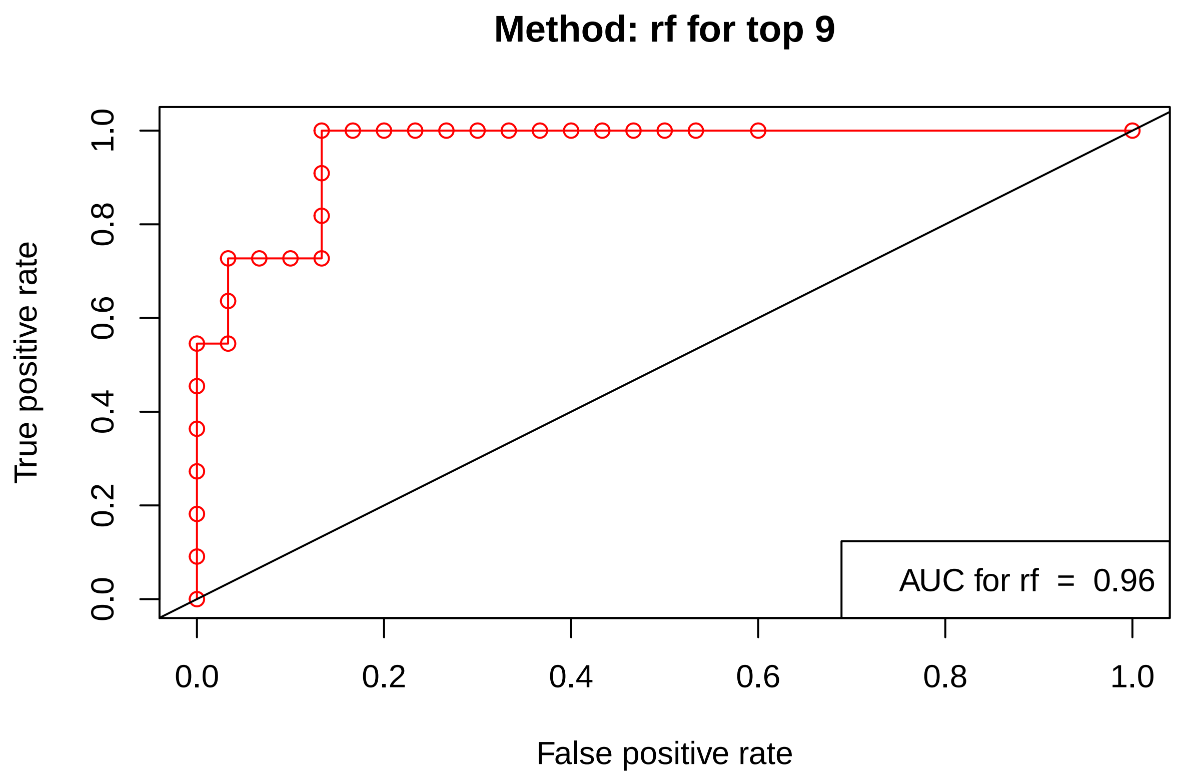


```
#12-d_PCA-topK_tissue
autoplot(pca, data = conditions, colour = 'Tissue', main =paste("PCA for top",
                                                              ks, "variables"))
```



## Figure 2C for the microbiome

```
#PREDICTION
#For SVM, use:
#control <- trainControl(method="cv", number=10, classProbs=TRUE)

#fit.algorKS <- train(Class~., data=DtrainKS, method=kalgoritmo, metric=metric,
#   trControl=control)
# ------------------------------------------------------------------------------
predictionsKSroc<- predict(fit.algorKS, DtestKS,type = "prob")[,2] #prob. clase=yes
predict.rocr  <- prediction (predictionsKSroc,Conditesting$Class)
perf.rocr     <- performance(predict.rocr,"tpr","fpr") #True/False positive.rate

#14_ROC-topK
# GRAPH ROC curve
# ------------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
# auc
par(mfrow = c(1,1))
plot(perf.rocr,type='o', col = "red",main = paste("Method:",
                                                  kalgoritmo, "for top", ks), ylim=c(0,1.01),xlim=c(0,1
abline(a=0, b=1)
legend("bottomright",legend= paste('AUC for', kalgoritmo, ' = ', round(auc,2)))
```



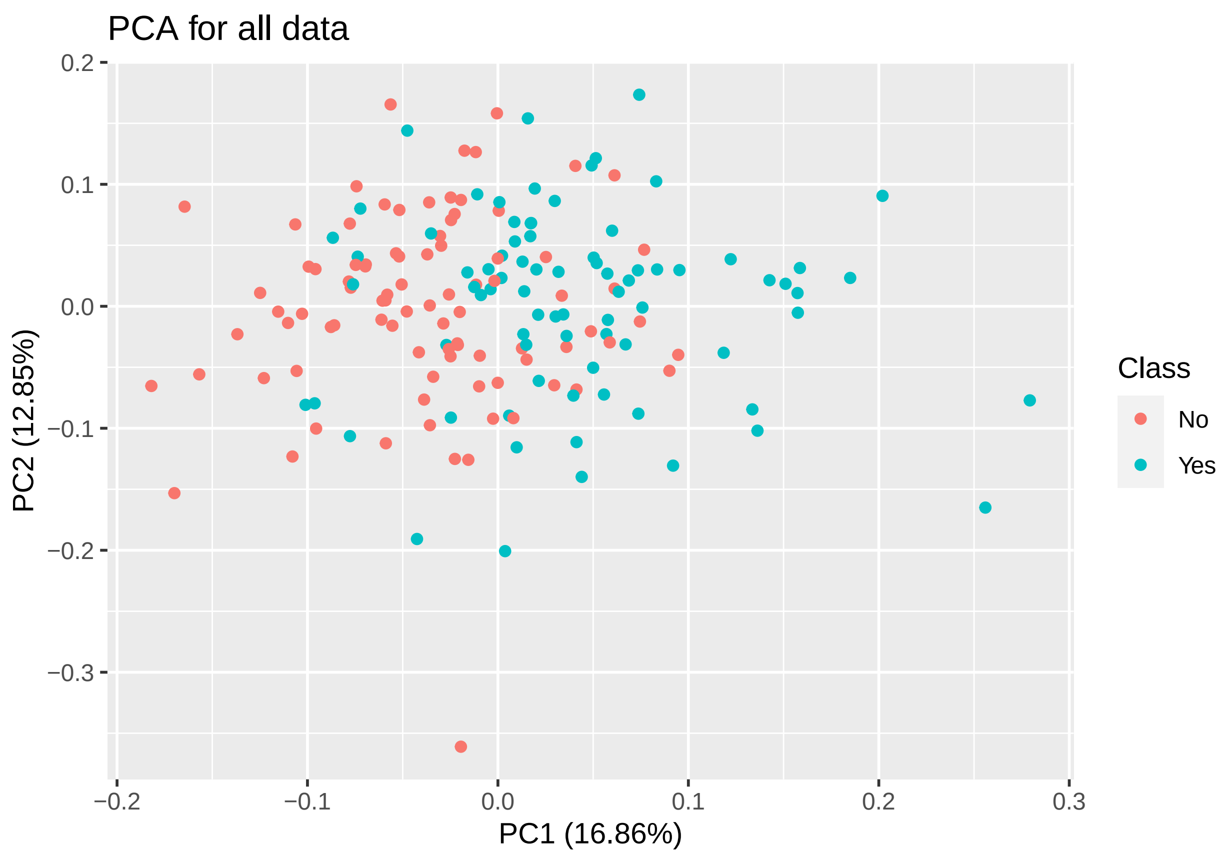
```
#COMPARISON CASE BY CASE for testing set
PRED<-as.data.frame(c(predictionsKS))
PRED2<-as.data.frame(c(Conditesting$Class))
juntos<-as.data.frame(c(PRED,PRED2))
# View(juntos)

#Extracting lists
ListaVariablestop<-as.data.frame(colnames(DtrainKS))
# write.csv(ListaVariablestop, file=paste(kalgoritmo,"Lista_top_variables.csv"))

#Extracting lists
ListaVariablesall<-as.data.frame(colnames(DtrainingRanked))
# write.csv(ListaVariablesall, file=paste(kalgoritmo,"Lista_all_variables.csv"))

#SAVING METRICS
# write.csv(EvalK, file="Metrics.csv")
```

With the current settings using the 9 variables selected by the model we calcutate the AUC. ##This ROC plot was used in Figure 2C.

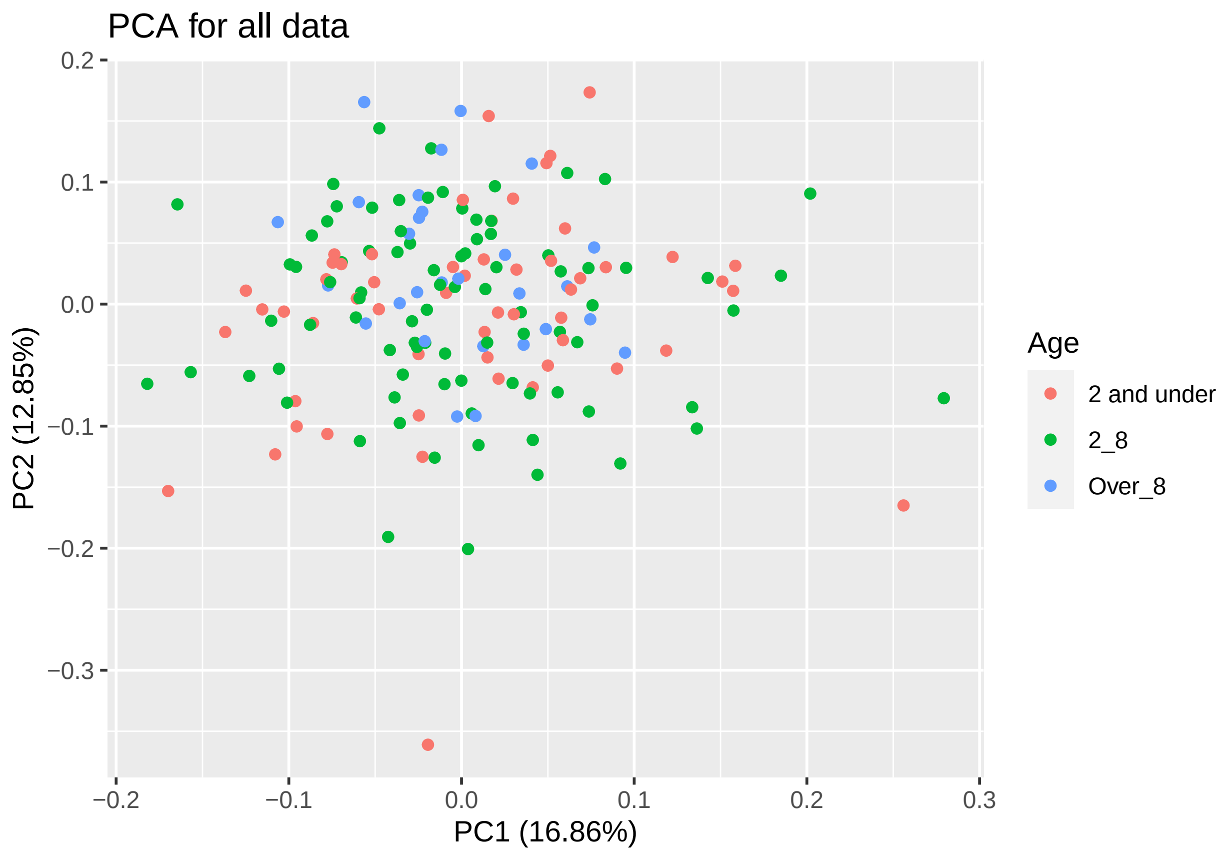
The ASVs are labeled based on the naming convention set forth by QIIME2. Upon importation in R, the ASVs whose names begin with a number have an X added to the front.

Analysis of serum chemistry

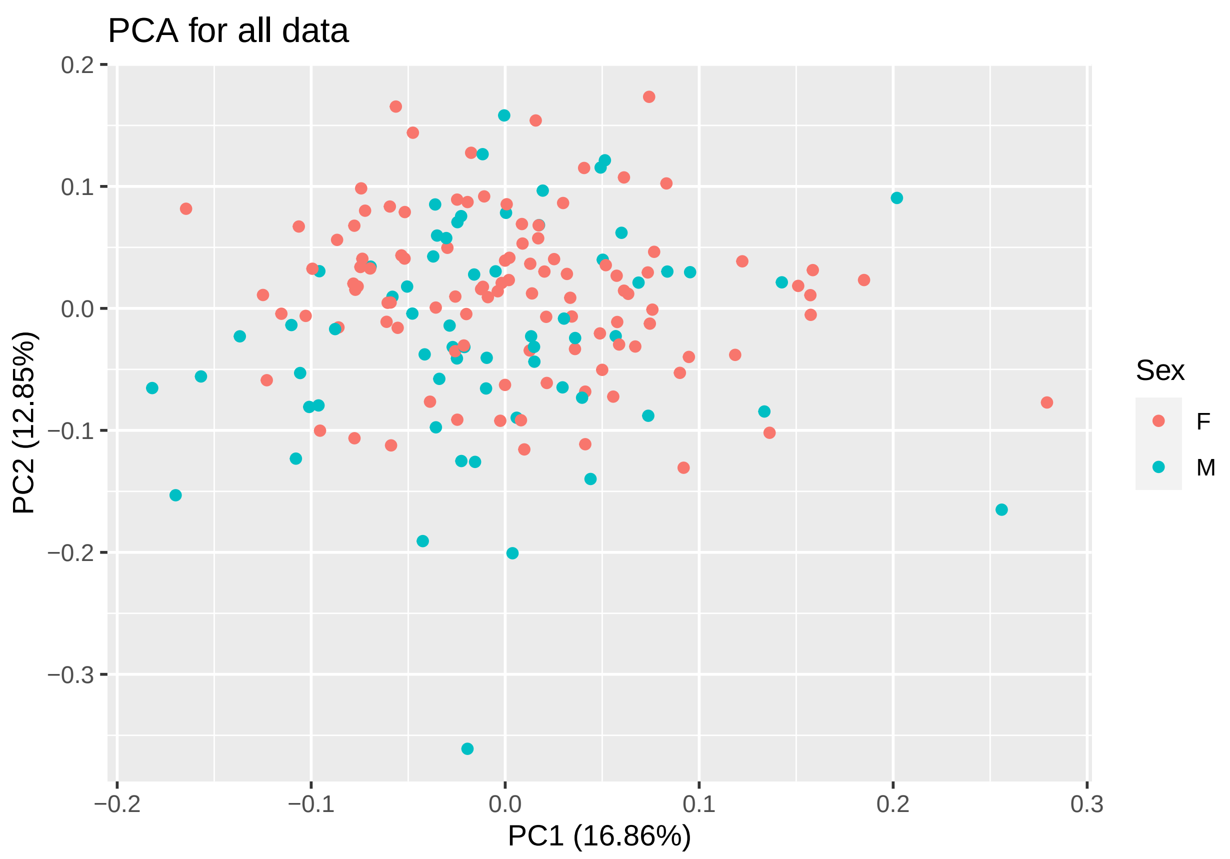
```r
# PART 1. LOAD DATA and metadata. Visualization of data
Data <- minmax(str_chem)
conditions <- str_meta_chem

##01_PCA - Class (Yes = Stricture and No = non-stricture)
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Class', main ="PCA for all data")
```



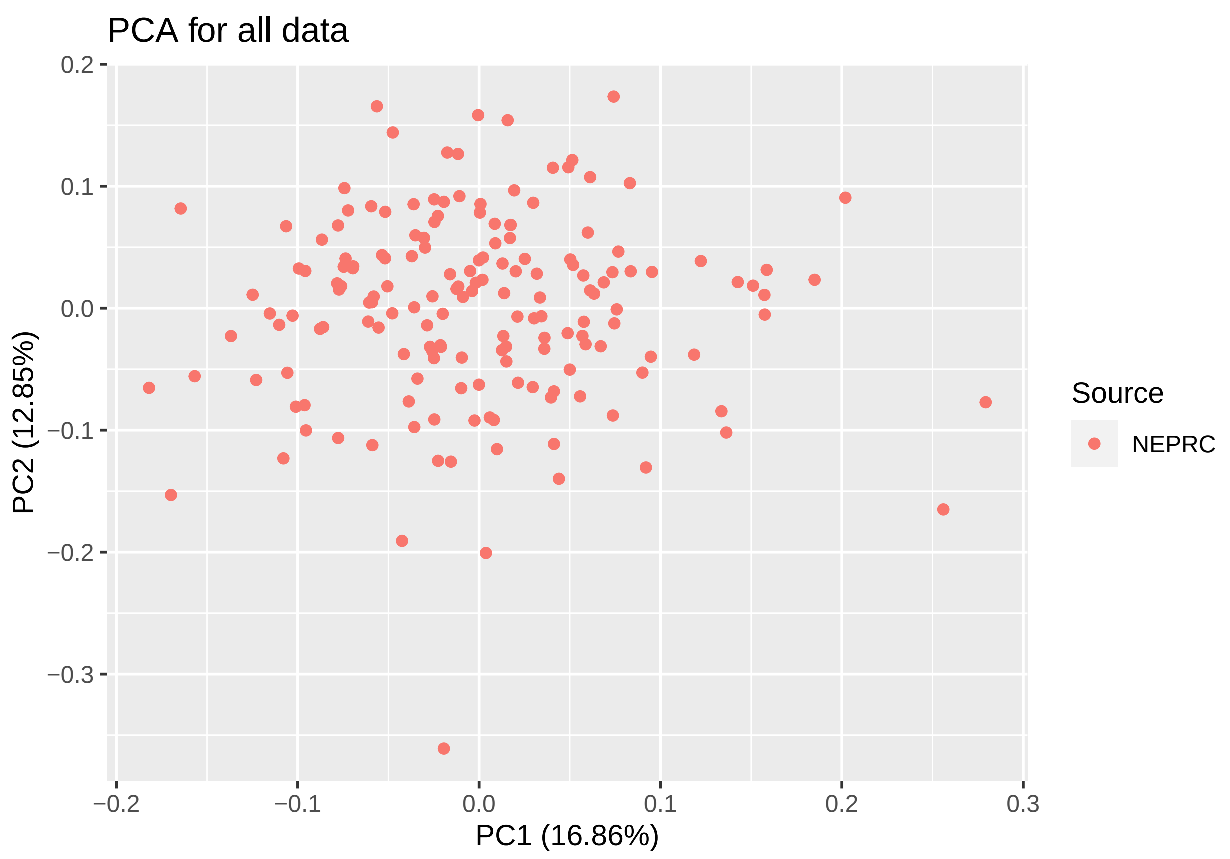
```r
## sub-PCA plots using metadata
## 01-1_PCA - Age
autoplot(pca, data = conditions, colour = 'Age', main ="PCA for all data")
```



```
## 01-2_PCA - Sex
autoplot(pca, data = conditions, colour = 'Sex', main ="PCA for all data")
```

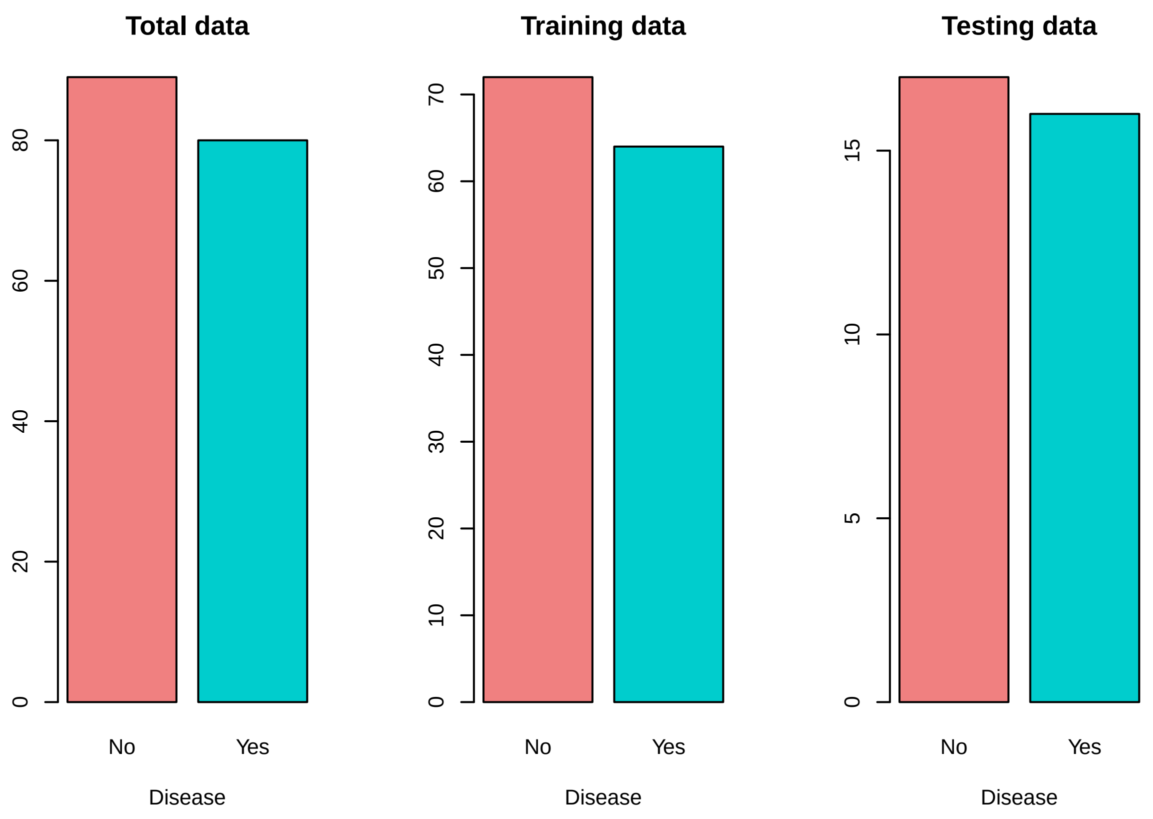


```
## 01-3_PCA - Source
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```



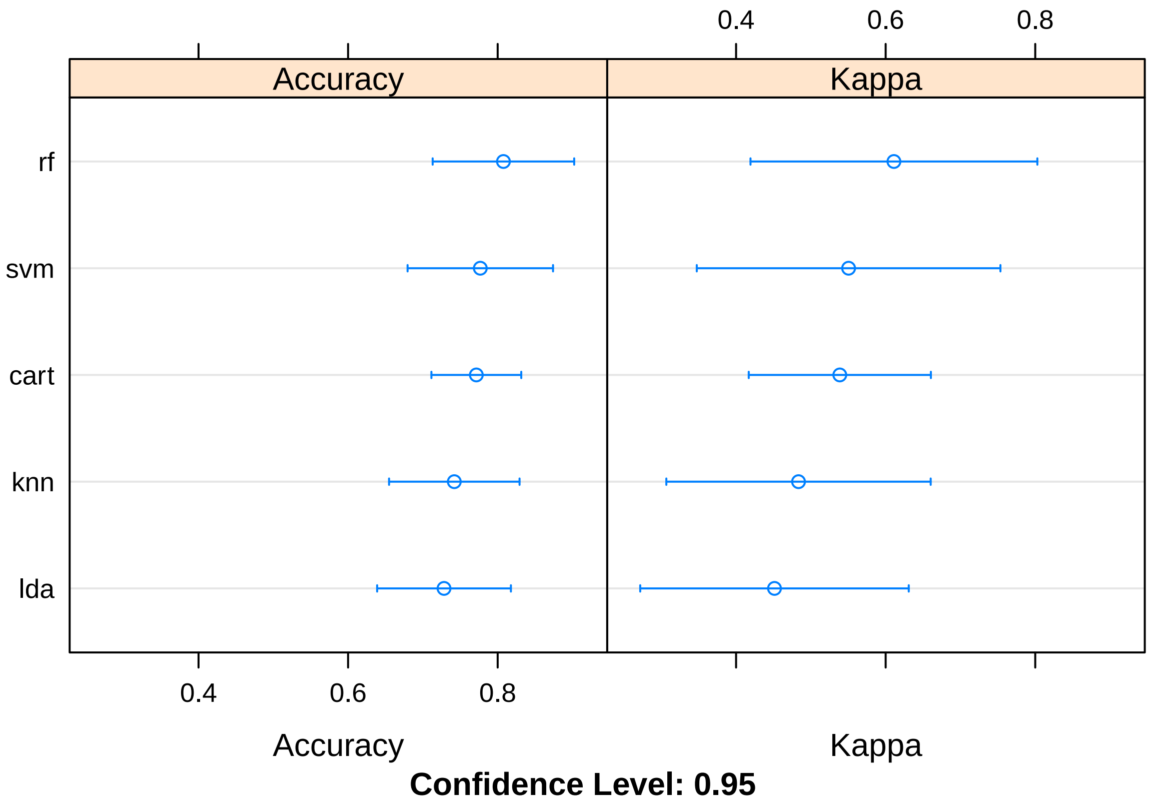
```
# Split the data into training (80%) and testing (20%) sets
set.seed(5)
test_index<-createDataPartition(conditions$Class, p=0.80, list = FALSE)

#Using D training to select features and then tested on Dtesting
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]

Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

#03_distribution of data by class entire dataset, training set and testing set
par(mfrow=c(1,3))

group1<-conditions$Class
conteos1 <- table(group1)
barplot(conteos1, main="Total data",
        xlab="Disease", col=c("lightcoral","cyan3"))
        #, legend = c("No","Yes"))

group2<-Conditrain$Class
conteos2 <- table(group2)
barplot(conteos2, main="Training data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```

```
group3<-Conditesting$Class
conteos3 <- table(group3)
barplot(conteos3, main="Testing data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```

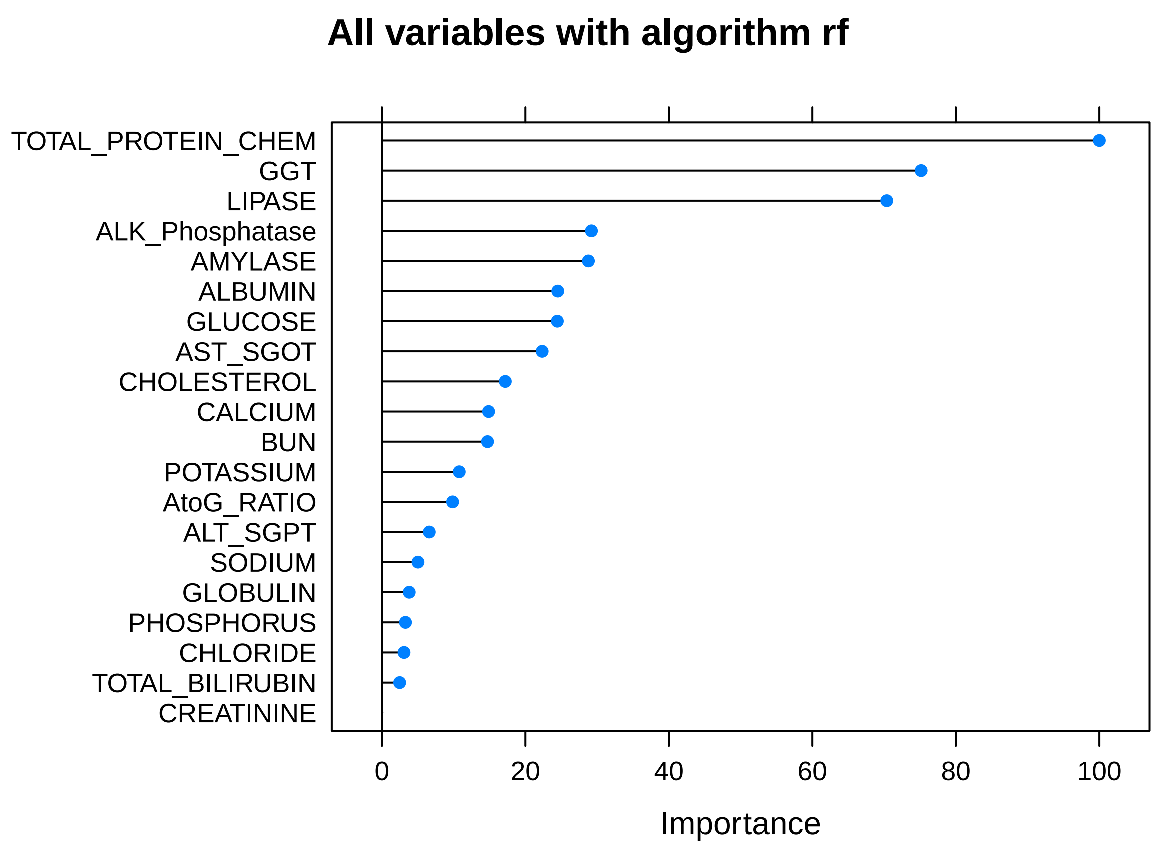


```
par(mfrow=c(1,1))

# Distribution in training set
percentage <- prop.table(table(Conditrain$Class)) * 100
# cbind(freq=table(Conditrain$Class), percentage=percentage)

#Distribution in testing set
percentage2 <- prop.table(table(Conditesting$Class))*100
# cbind(freq=table(Conditesting$Class),percentage=percentage2)

# PART 2. DISTRIBUTION OF VARIABLES BY CLASS
# Performed on training data but could be done on entire set or testing set
# split input and output
x <- Dtraining
y <- Conditrain$Class # class
sx <- Dtraining[,c(1:6)]

#ALGORITHMS

# Run algorithms using 10-fold cross validation
```

```
control <- trainControl(method="cv", number=10, classProbs=TRUE)
metric <- "Accuracy"

#Clasification algorithms
# a) linear algorithms
fit.lda <- train(Dtraining, Conditrain$Class, method="lda", metric=metric,
                 trControl=control)
# b) nonlinear algorithms
# CART
fit.cart <- train(Dtraining, Conditrain$Class, method="rpart", metric=metric,
                  trControl=control)
# kNN
fit.knn <- train(Dtraining, Conditrain$Class, method="knn", metric=metric,
                 trControl=control)
# c) advanced algorithms
# SVM
fit.svm <- train(Dtraining, Conditrain$Class, method="svmRadial", metric=metric,
                 trControl=control)
# Random Forest
fit.rf <- train(Dtraining, Conditrain$Class, method="rf", metric=metric,
                trControl=control)

#summarize accuracy of models
results <- resamples(list(lda=fit.lda,cart=fit.cart,svm=fit.svm,
                          knn=fit.knn,rf=fit.rf,rf=fit.rf))
summary(results)

# compare accuracy of models
dotplot(results)
```

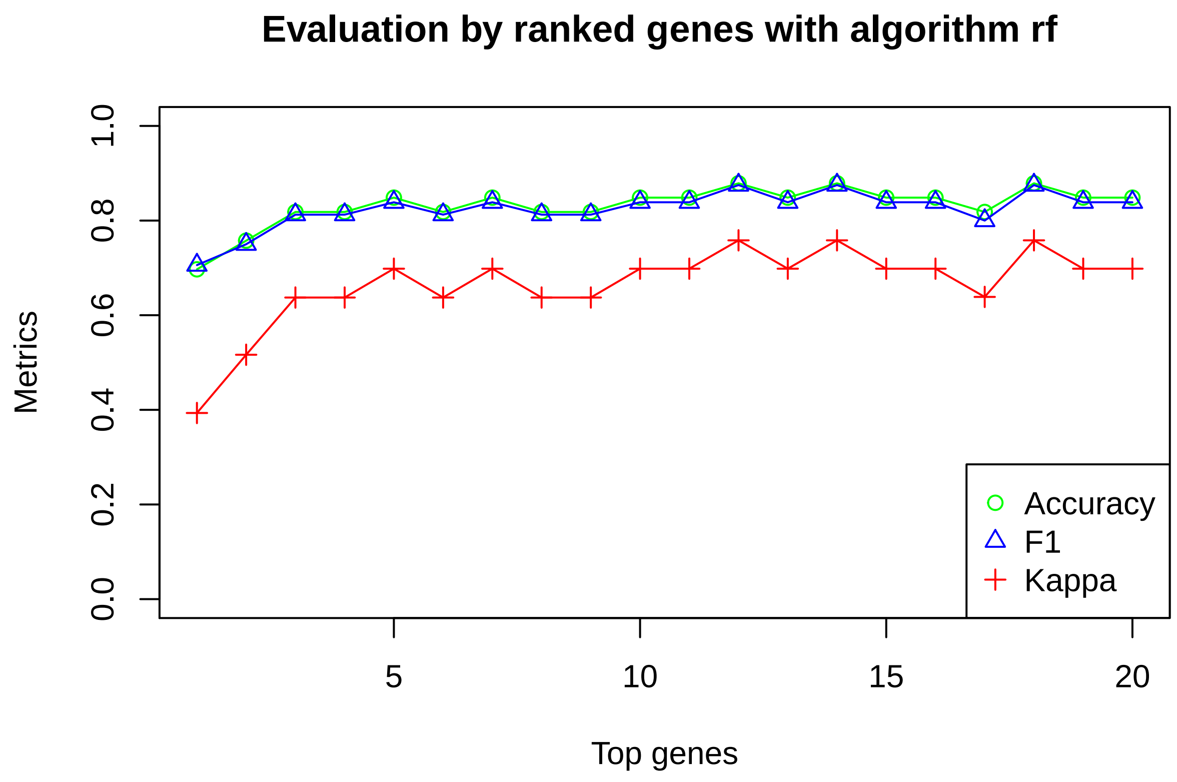


```
#RANKING FOR ALGORITHM

#ALGORITHM SELECTED
#model=fit.knn
#kalgoritmo="knn"
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
kalgoritmo="rf"

importance <- varImp(model, scale=TRUE)
# head(importance)

#Graph importance
plot(importance, main = paste("All variables with algorithm", kalgoritmo))
```

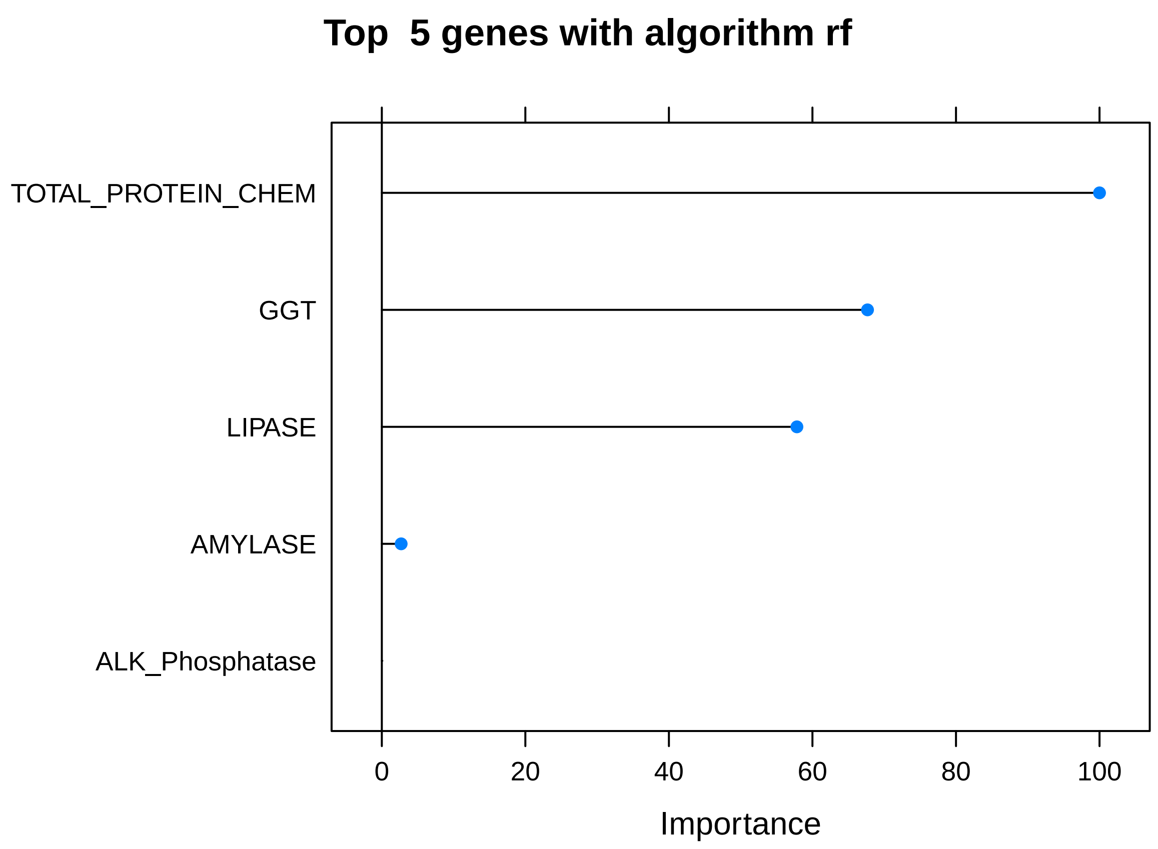


```
#INDEX
IndexRank <-data.frame(sort(importance$importance$Overall,
                            index.return = TRUE, decreasing = TRUE)[2])
#For SMV y KNN importance$importance$Control, For RF/cart use $Overall
#IndexRank <-data.frame(sort(importance$importance$Yes,
# index.return = TRUE, decreasing = TRUE)[2])
Ranking<-t(IndexRank)

DtrainingRanked<-Dtraining[,Ranking[1,]]
DtestingRanked<-Dtesting[,Ranking[1,]]

DataRanked<-Data[,Ranking[1,]]
# write.csv(DataRanked, "All_data_ranked.csv")

#EVALUATION OF DIF SUBSET OF TOP GENES
#Number of top genes to evaluate
kvalue=20


kEvaluacion<-matrix(,nrow=kvalue, ncol=19) # 18 parameters and 1 extra
colnames(kEvaluacion)<-c("Accuracy","Kappa","AccuracyLower",
                         "AccuracyUpper","AccuracyNull", "AccuracyPValue",
                         "McnemarPValue", "Sensitivity", "Specificity",
                         "Pos Pred Value", "Neg Pred Value", "Precision",
                         "Recall","F1","Prevalence" ,"Detection Rate",
                         "Detection Prevalence","Balanced Accuracy", "K")
```

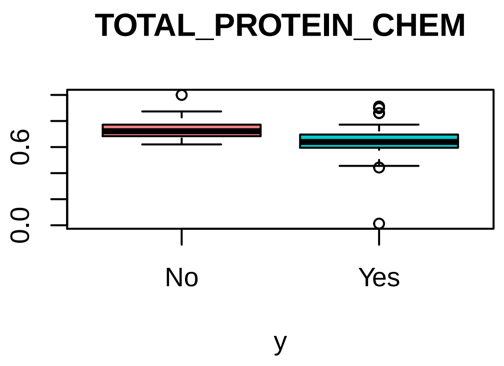
```r
# For k=1 it is calculated separately as it transforms data frame to vector
k=1 #
DtrainK<-as.data.frame(DtrainingRanked[,1])
colnames(DtrainK)<-colnames(DtrainingRanked)[1]
DtestK<-as.data.frame(DtestingRanked[,1])
colnames(DtestK)<-colnames(DtestingRanked)[1]

fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                    trControl=control)
predictionsK <- predict(fit.algorK, DtestK)
StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
kEvaluacion[1,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))


for (k in 2:kvalue){
  #k=2
  DtrainK<-DtrainingRanked[,c(1:k)]
  DtestK<-DtestingRanked[,c(1:k)]
  fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                      trControl=control)
  predictionsK <- predict(fit.algorK, DtestK)
  StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
  kEvaluacion[k,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))
  #if not adding extra columns you can use kEvaluacion[k,]<-t(as.data.frame(c(StatisticsK$overall,Stati
}

EvalK<-as.data.frame(kEvaluacion)

par(mfrow = c(1,1))
plot(EvalK$Accuracy,type="o",pch=1,col="green",
     main = paste("Evaluation by ranked genes with algorithm",
                  kalgoritmo,sed=""),xlab = "Top genes",
     ylab = "Metrics", ylim=c(0,1))
lines(EvalK$F1,type = "o", pch=2, col="blue")
lines(EvalK$Kappa,type = "o", pch=3, col="red")
legend("bottomright",legend=c("Accuracy","F1","Kappa"),pch=c(1,2,3),
       col=c("green","blue","red"))
```

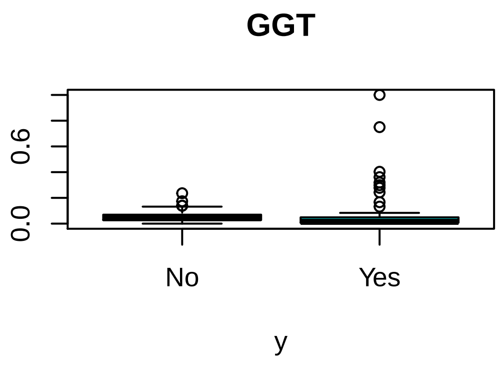


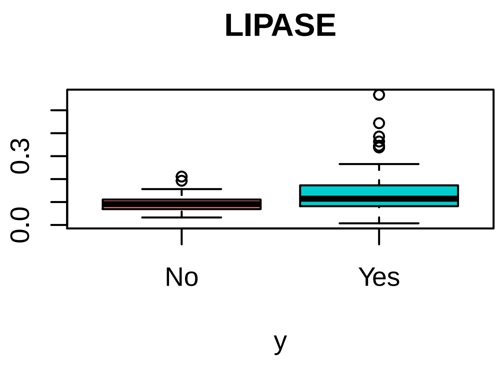
```
#for value of K chosen:
ks=5
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]

DataRankedtopK<-DataRanked[,c(1:ks)]
# write.csv(DataRankedtopK, paste0("DataRankedtop",ks,"-",kalgoritmo,".csv"))

fit.algorKS <- train(DtrainKS, Conditrain$Class, method=kalgoritmo, metric=metric,
                     trControl=control)
importance <- varImp(fit.algorKS, scale=TRUE)
plot(importance, main = paste('Top ', ks, "genes with algorithm", kalgoritmo))
```

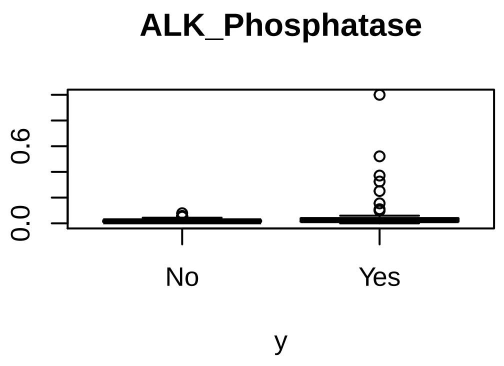


```
predictionsKS <- predict(fit.algorKS, DtestKS)
StatisticsKS<-confusionMatrix(predictionsKS, Conditesting$Class)

xR <- DtrainKS
yR <- Conditrain$Class
sxR <- DtrainingRanked[,c(3,4,1,2)]
#Here you can rerun particular parameter

#boxplot
par(mfrow=c(2,2))

for(i in 1:4) {
  boxplot(xR[,i]~y, main=names(xR)[i],col=c("lightcoral","cyan3"),ylab = " ")
}
```

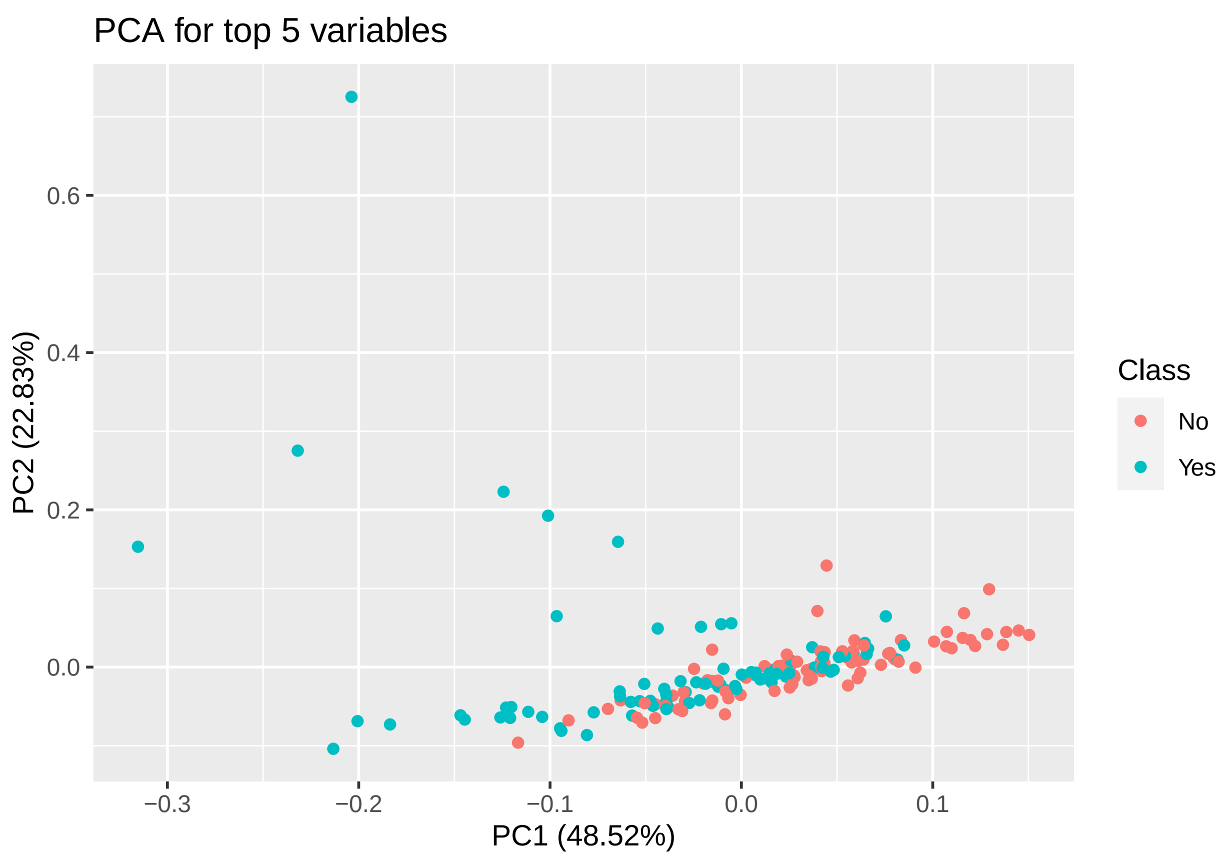




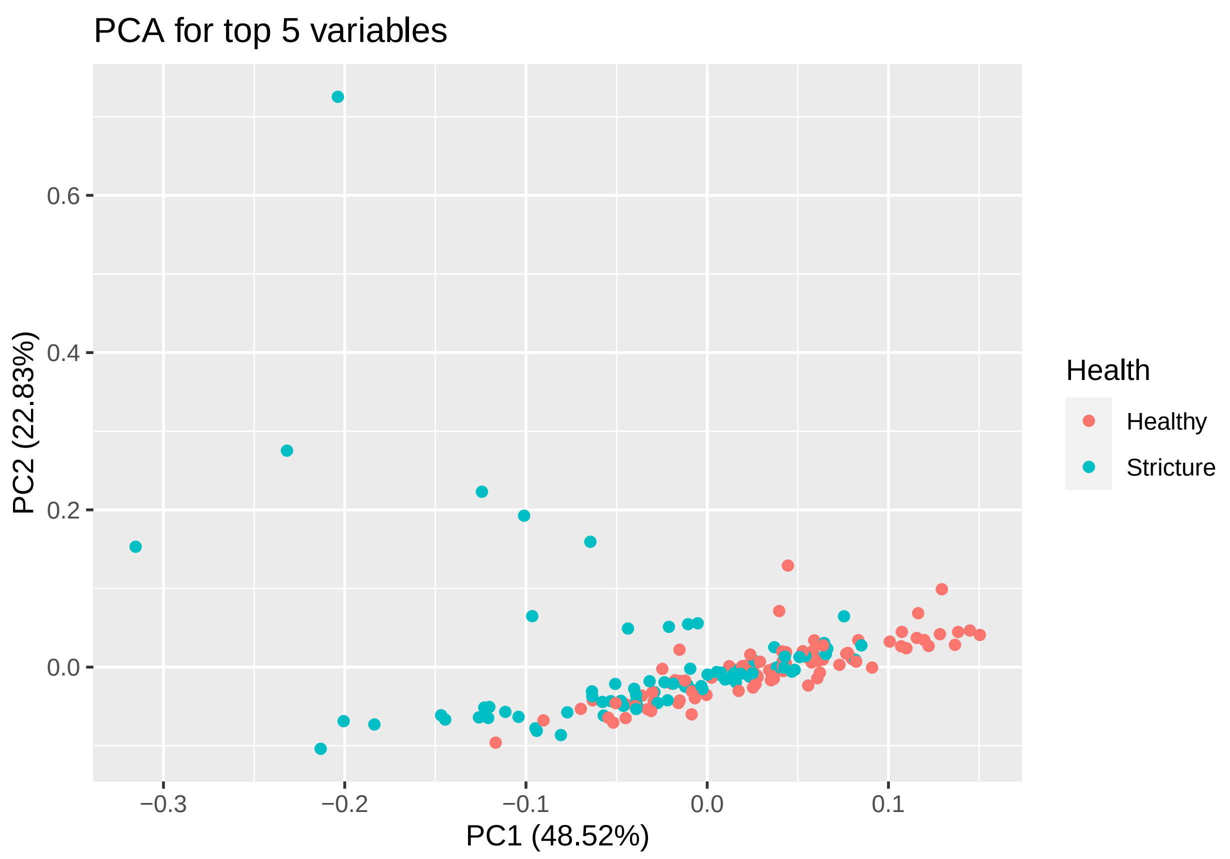




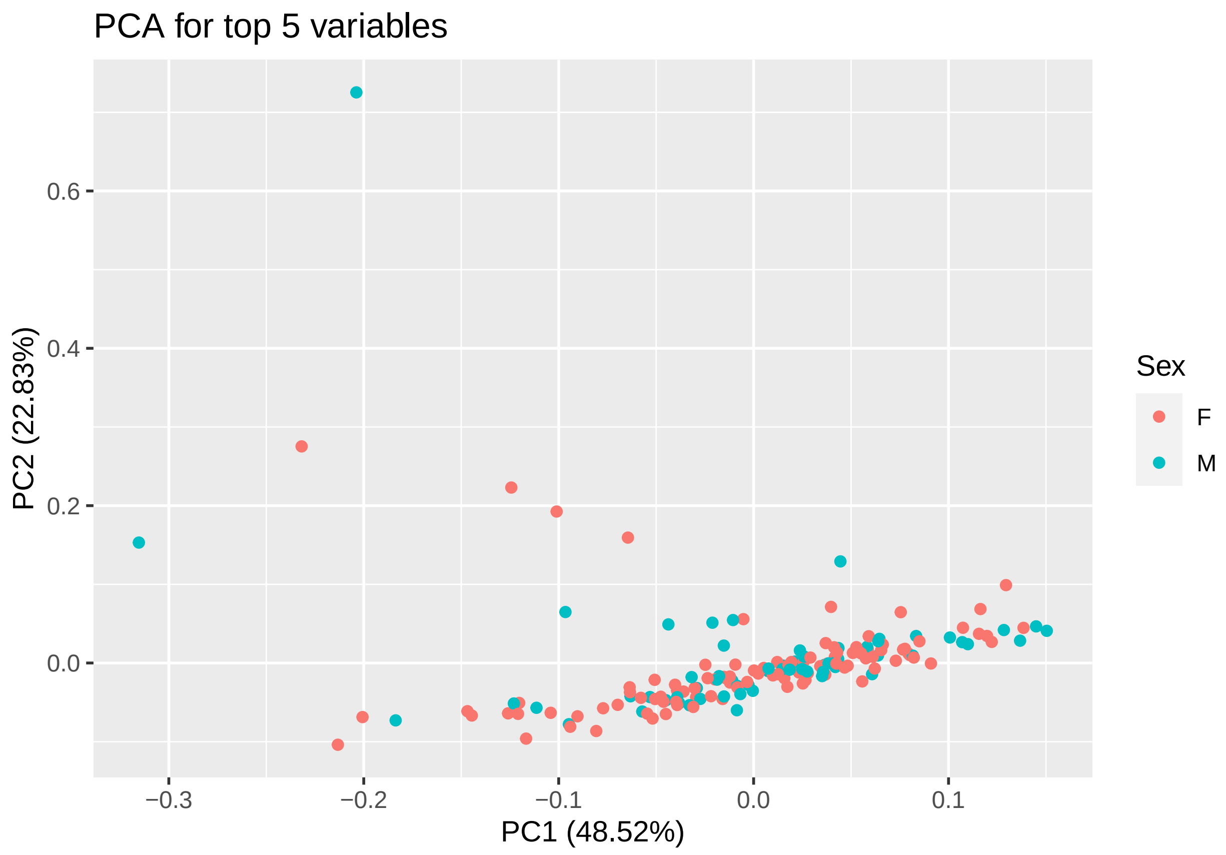
```
par(mfrow=c(1,1))


pca<-prcomp(DataRanked[,1:ks])
autoplot(pca, data = conditions, colour = 'Class',
         main =paste("PCA for top",ks, "variables"))
```



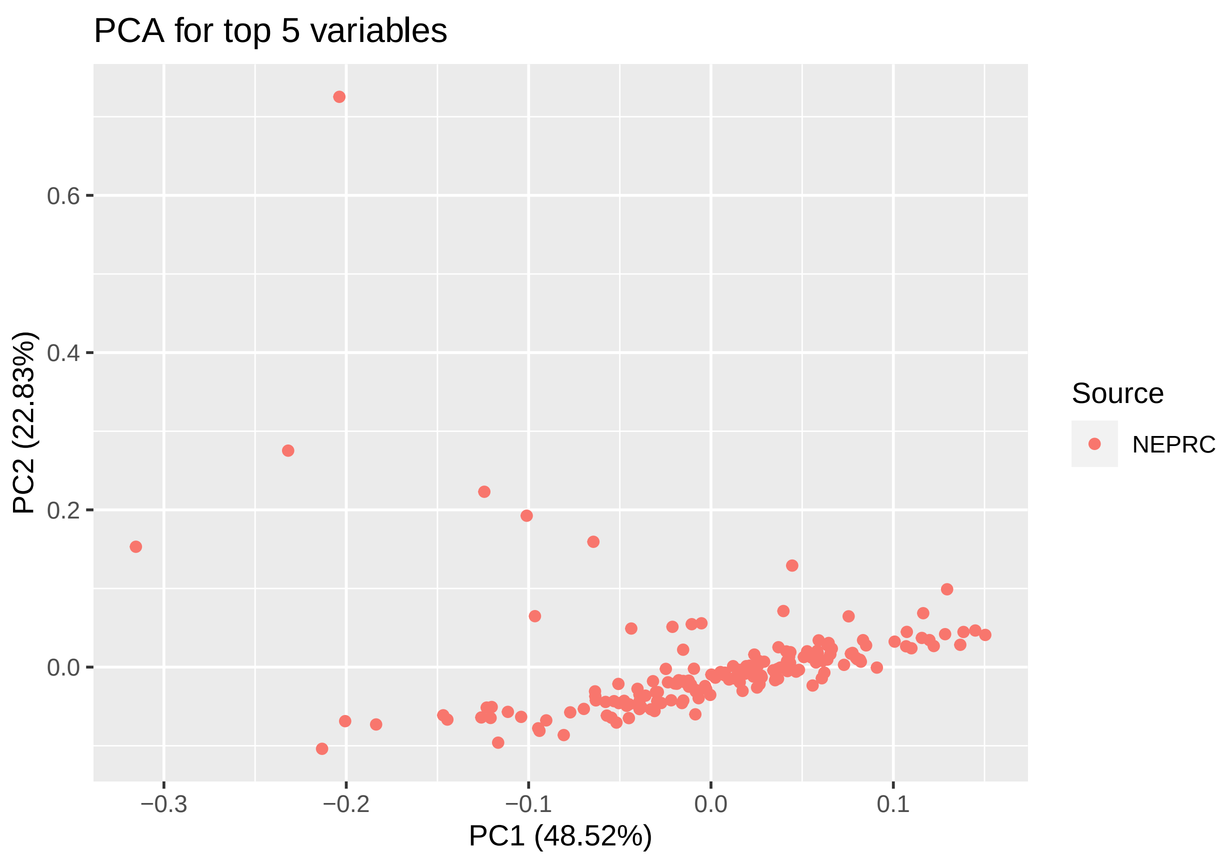
```
autoplot(pca, data = conditions, colour = 'Health',
         main =paste("PCA for top",ks, "variables"))
```



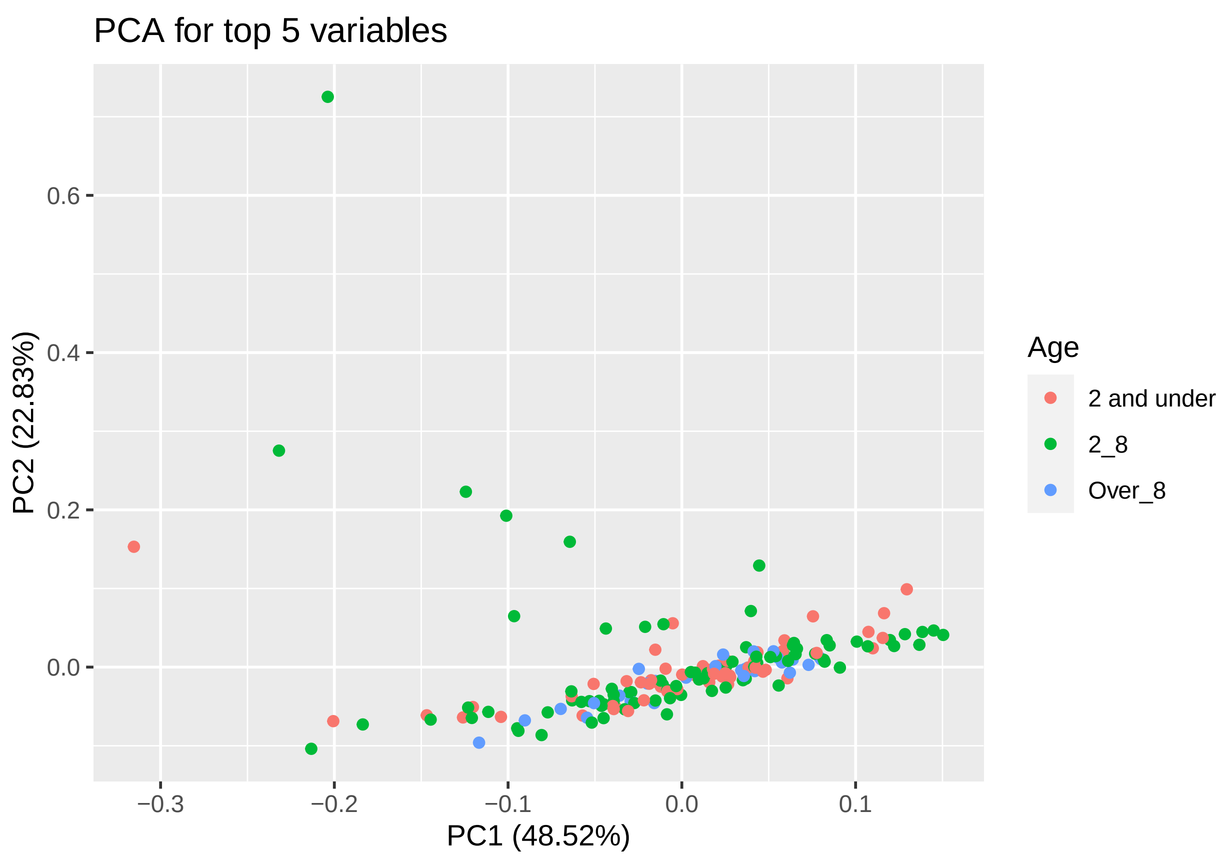
```
autoplot(pca, data = conditions, colour = 'Sex',
         main =paste("PCA for top",ks, "variables"))
```



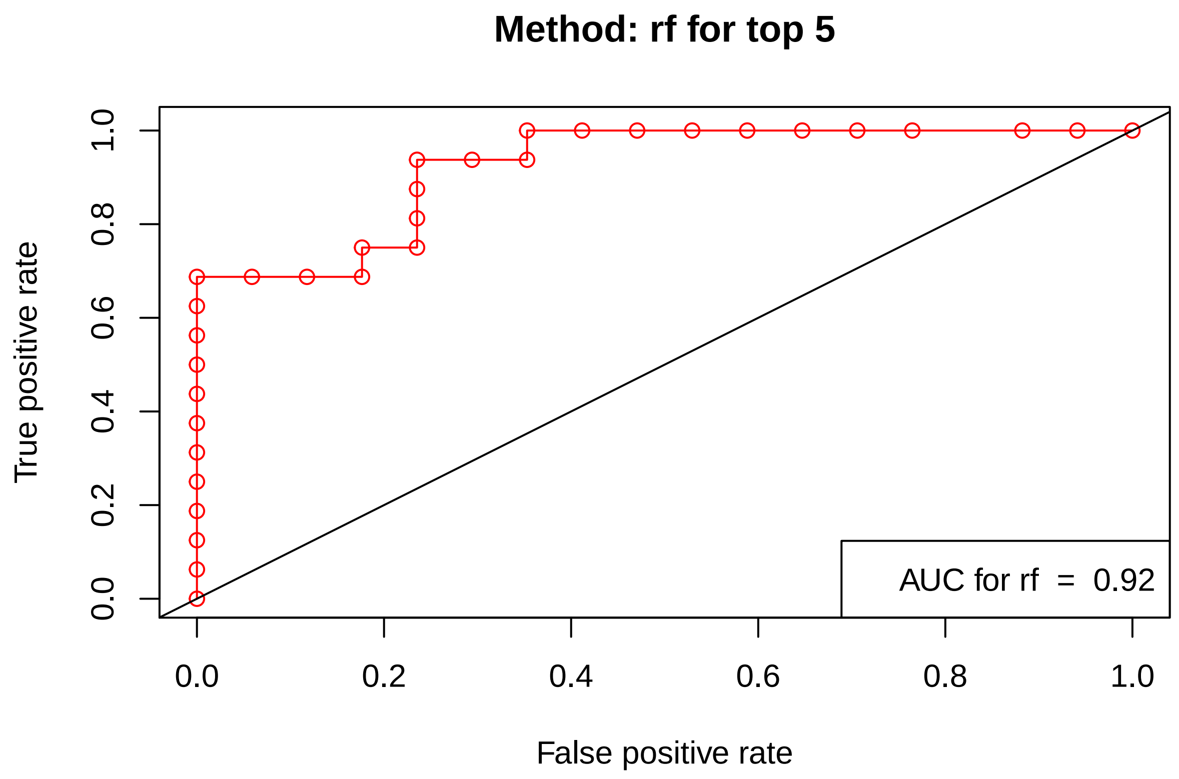
```
autoplot(pca, data = conditions, colour = 'Source',
         main =paste("PCA for top",ks, "variables"))
```



```r
autoplot(pca, data = conditions, colour = 'Age',
         main =paste("PCA for top",ks, "variables"))
```



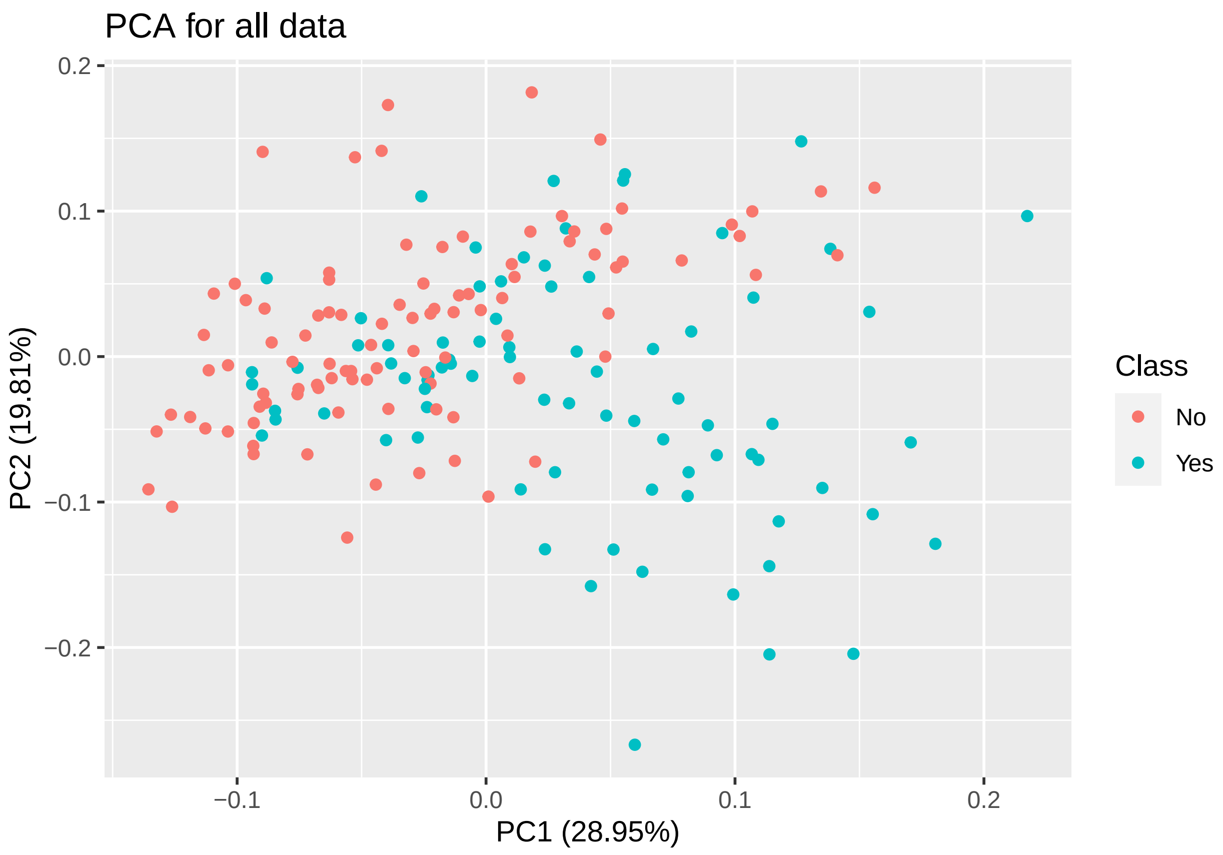
```
#PREDICTION
#FOR SVM, use:
#control <- trainControl(method="cv", number=10, classProbs=TRUE)

#fit.algorKS <- train(Class~., data=DtrainKS, method=kalgoritmo, metric=metric,
# trControl=control)
# ----------------------------------------------------------------------------
predictionsKSroc<- predict(fit.algorKS, DtestKS,type = "prob")[,2] #prob. clase=yes
predict.rocr  <- prediction (predictionsKSroc,Conditesting$Class)
perf.rocr     <- performance(predict.rocr,"tpr","fpr") #True/False positive.rate

# ROC FIGURE
# ----------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
# auc
par(mfrow = c(1,1))
plot(perf.rocr,type='o', col = "red",
     main = paste("Method:", kalgoritmo, "for top", ks), ylim=c(0,1.01),xlim=c(0,1))
abline(a=0, b=1)
legend("bottomright",legend= paste('AUC for', kalgoritmo, ' = ', round(auc,2)))
```



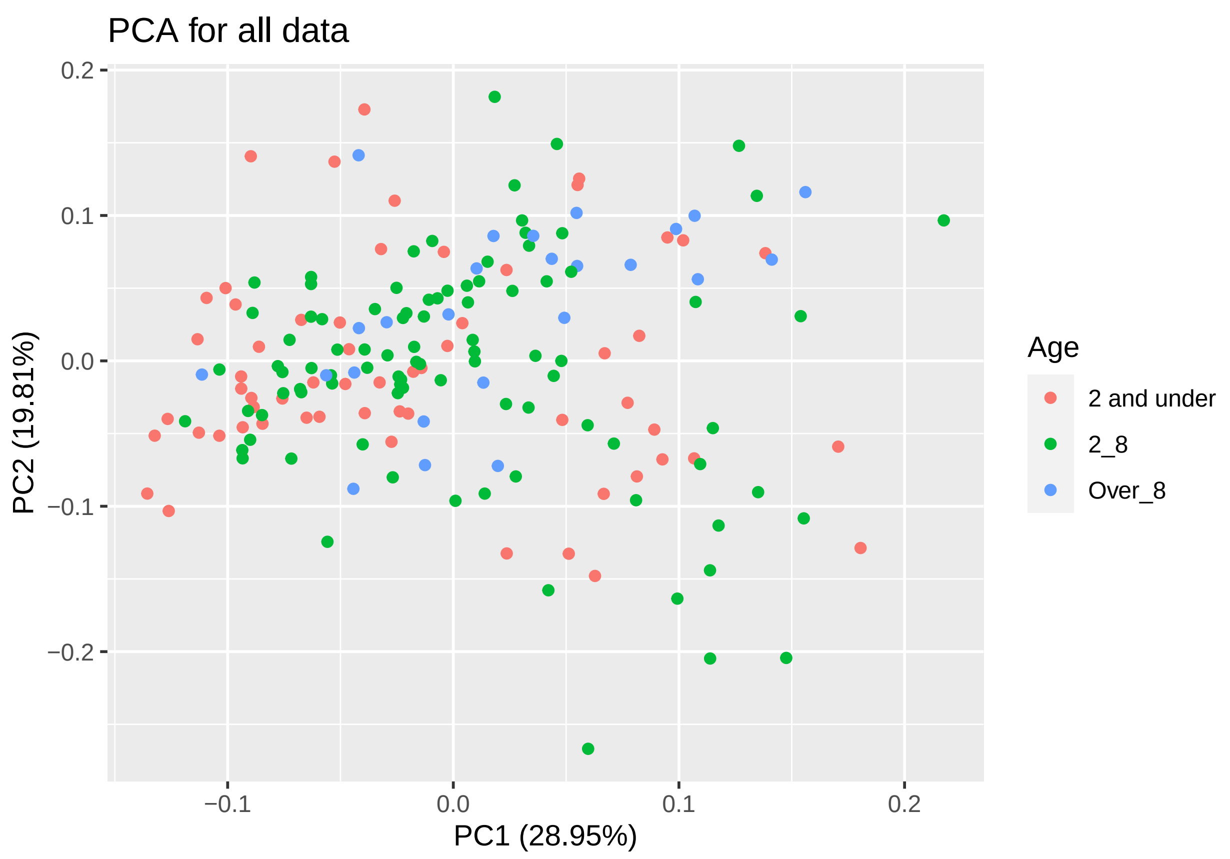
```
#COMPARISON CASE by BASE of test set
PRED<-as.data.frame(c(predictionsKS))
PRED2<-as.data.frame(c(Conditesting$Class))
juntos<-as.data.frame(c(PRED,PRED2))

#Extraction of lists
ListaVariablestop<-as.data.frame(colnames(DtrainKS))
# write.csv(ListaVariablestop, file=paste(kalgoritmo,"Lista_all_variables.csv"))

#Extraction of lists
ListaVariablesall<-as.data.frame(colnames(DtrainingRanked))
# write.csv(ListaVariablesall, file=paste(kalgoritmo,"Lista_all_variables.csv"))
```

Analysis of CBC

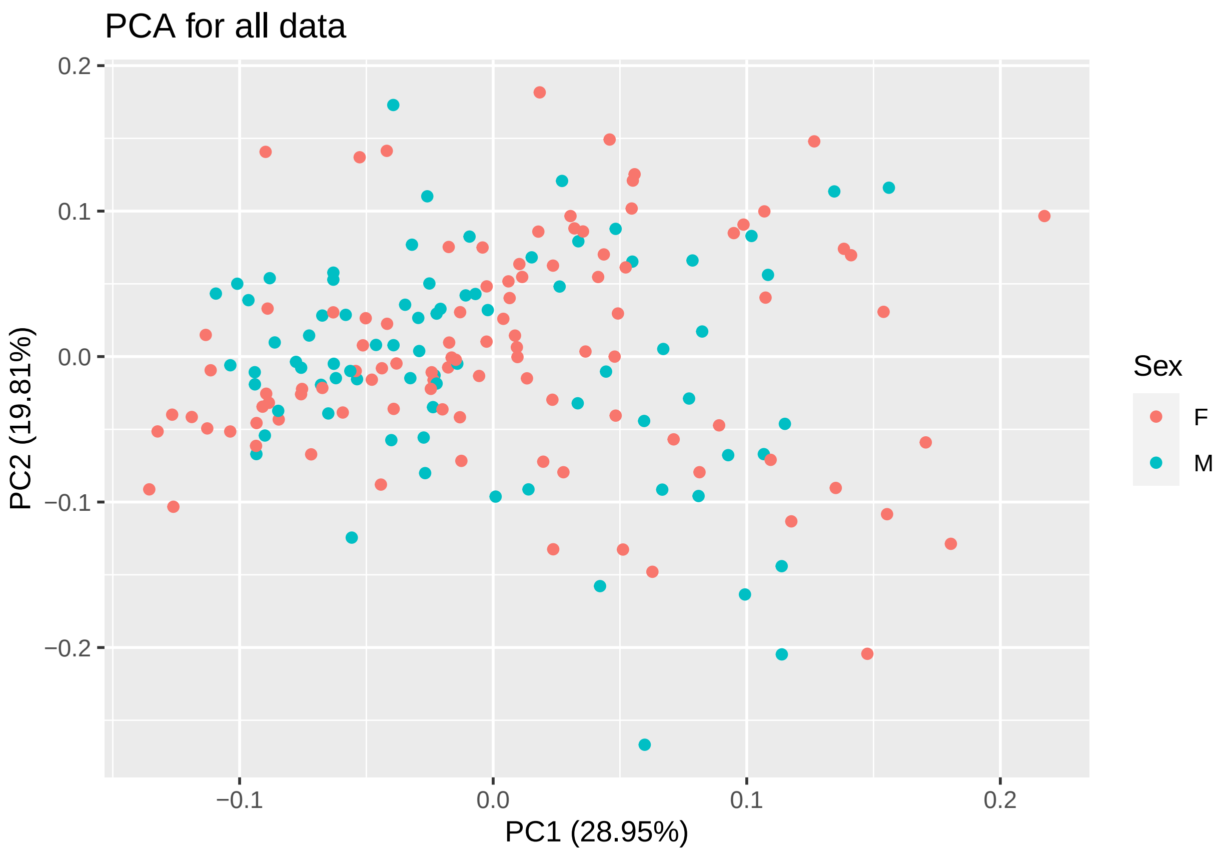
```
Data <- minmax(str_cbc)
conditions <- str_meta_cbc

##01_PCA - Class (Yes = Stricture and No = non-stricture)
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Class', main ="PCA for all data")
```



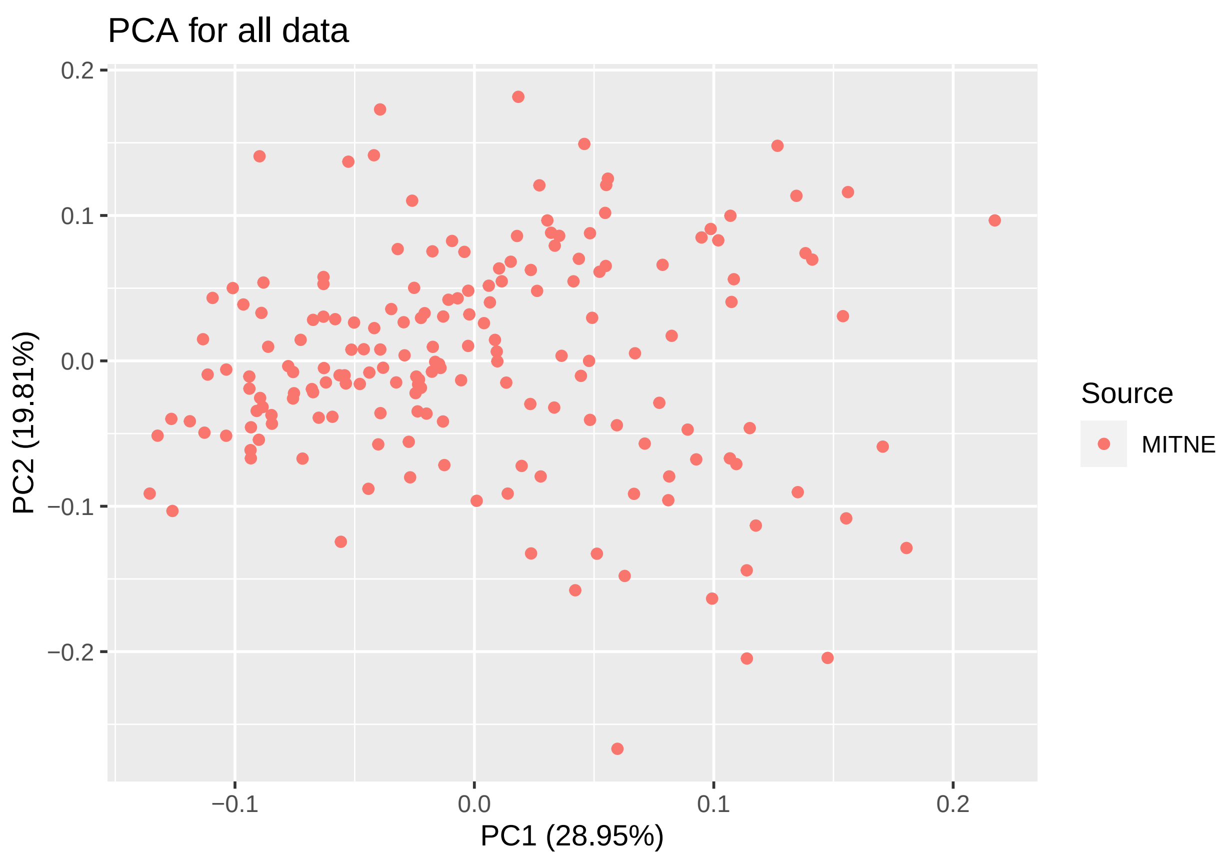
```
## sub-PCA plots using metadata
## 01-1_PCA - Age
autoplot(pca, data = conditions, colour = 'Age', main ="PCA for all data")
```



```
## 01-2_PCA - Sex
autoplot(pca, data = conditions, colour = 'Sex', main ="PCA for all data")
```

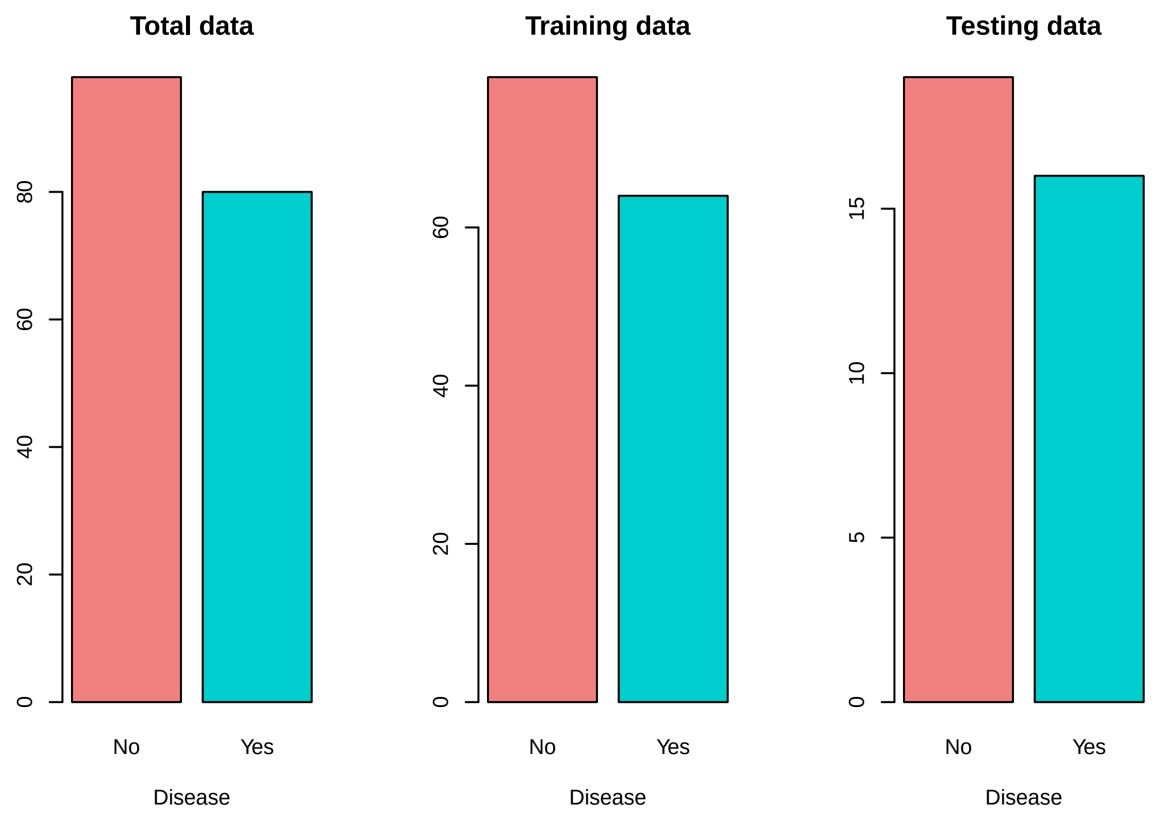


```
## 01-3_PCA - Source
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```



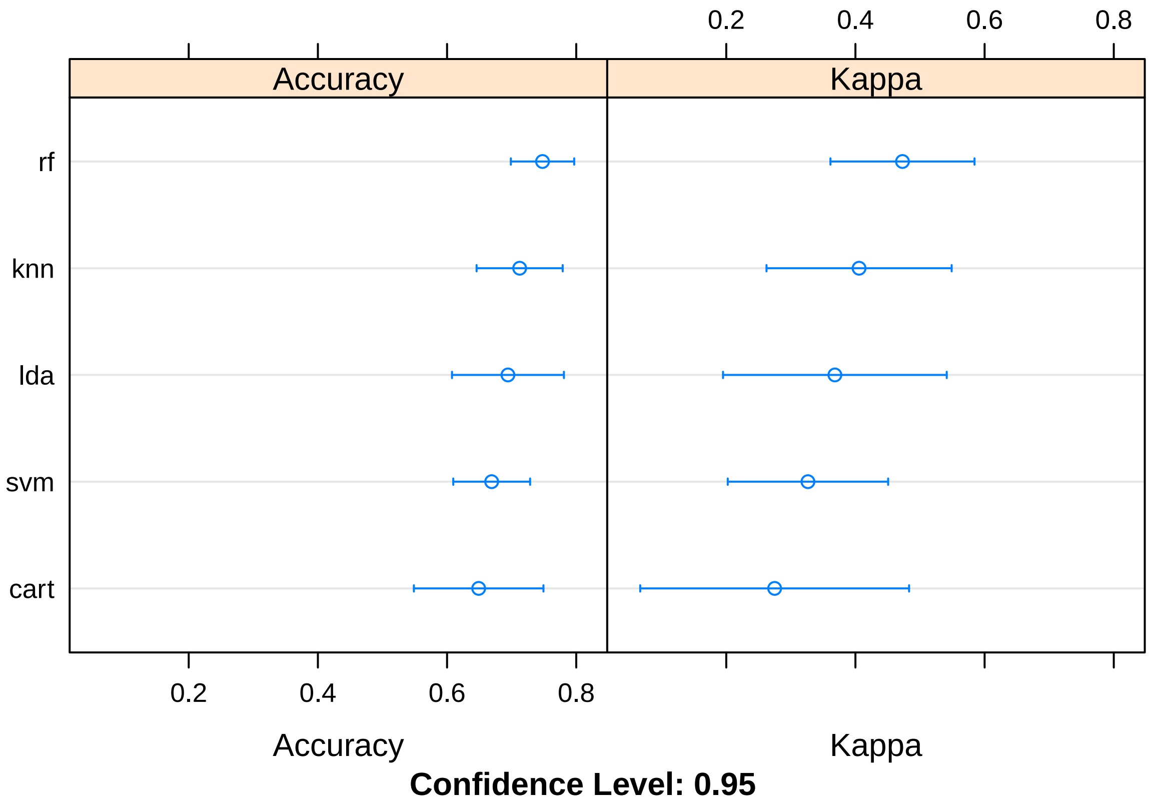
```
# Split the data into training (80%) and testing (20%) sets
set.seed(3)
test_index<-createDataPartition(conditions$Class, p=0.80, list = FALSE)

#Using D training to select features and then tested on Dtesting
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]

Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

#03_distribution of data by class in entire dataset, training set and testing set
par(mfrow=c(1,3))

group1<-conditions$Class
conteos1 <- table(group1)
barplot(conteos1, main="Total data",
        xlab="Disease", col=c("lightcoral","cyan3"))
        #, legend = c("No","Yes"))

group2<-Conditrain$Class
conteos2 <- table(group2)
barplot(conteos2, main="Training data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```

```
group3<-Conditesting$Class
conteos3 <- table(group3)
barplot(conteos3, main="Testing data",
        xlab="Disease", col=c("lightcoral","cyan3"))
```

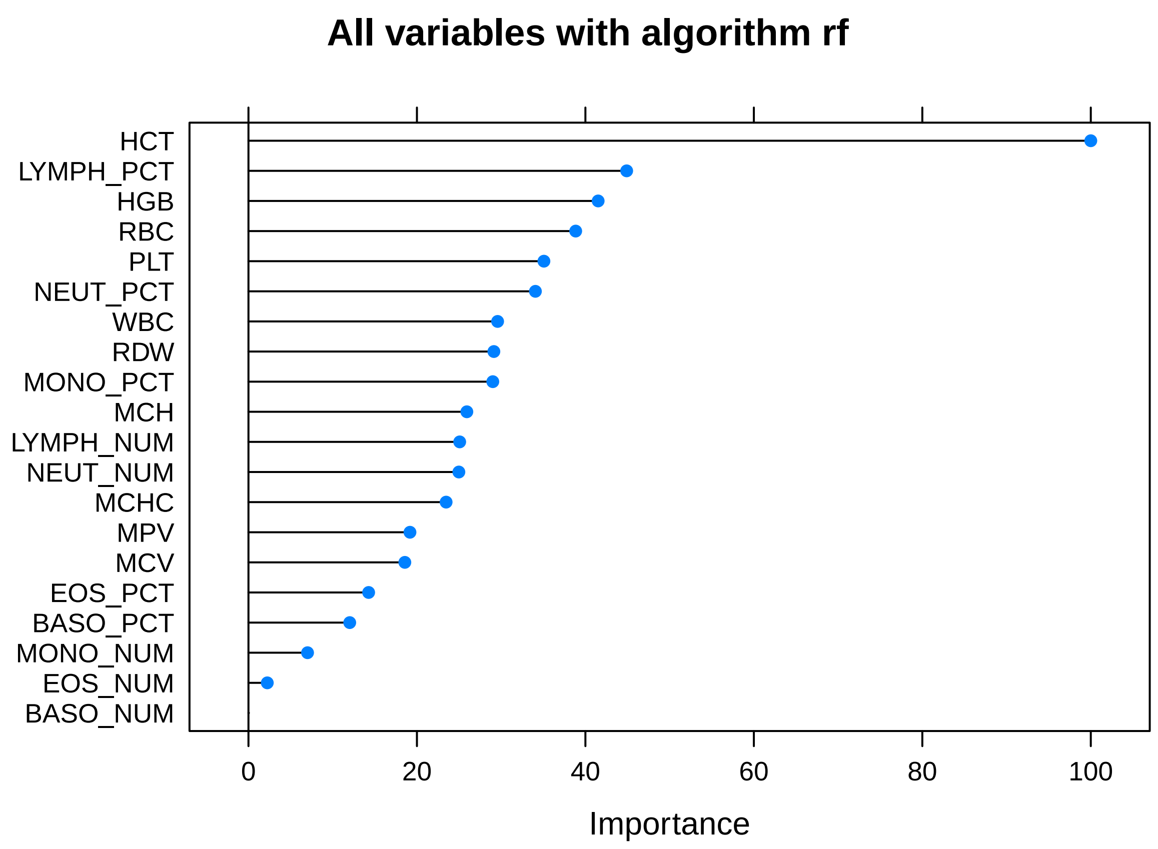


```
par(mfrow=c(1,1))

# Distribution in training set
percentage <- prop.table(table(Conditrain$Class)) * 100
# cbind(freq=table(Conditrain$Class), percentage=percentage)

#Distribution in testing set
percentage2 <- prop.table(table(Conditesting$Class))*100
# cbind(freq=table(Conditesting$Class),percentage=percentage2)

# PART 2. DISTRIBUTION OF VARIABLES BY CLASS
# Performed on training data but could be done on entire set or testing set
# split input and output
x <- Dtraining
y <- Conditrain$Class # class
sx <- Dtraining[,c(1:6)]

#ALGORITHMS

# Run algorithms using 10-fold cross validation
```

```r
control <- trainControl(method="cv", number=10, classProbs=TRUE)
metric <- "Accuracy"

#Clasification algorithms
# a) linear algorithms
fit.lda <- train(Dtraining, Conditrain$Class, method="lda", metric=metric,
                 trControl=control)
# b) nonlinear algorithms
# CART
fit.cart <- train(Dtraining, Conditrain$Class, method="rpart", metric=metric,
                  trControl=control)
# kNN
fit.knn <- train(Dtraining, Conditrain$Class, method="knn", metric=metric,
                 trControl=control)
# c) advanced algorithms
# SVM
fit.svm <- train(Dtraining, Conditrain$Class, method="svmRadial", metric=metric,
                 trControl=control)
# Random Forest
fit.rf <- train(Dtraining, Conditrain$Class, method="rf", metric=metric,
                trControl=control)

#summarize accuracy of models
results <- resamples(list(lda=fit.lda,cart=fit.cart,svm=fit.svm,
                          knn=fit.knn,rf=fit.rf,rf=fit.rf))
summary(results)

# compare accuracy of models
dotplot(results)
```

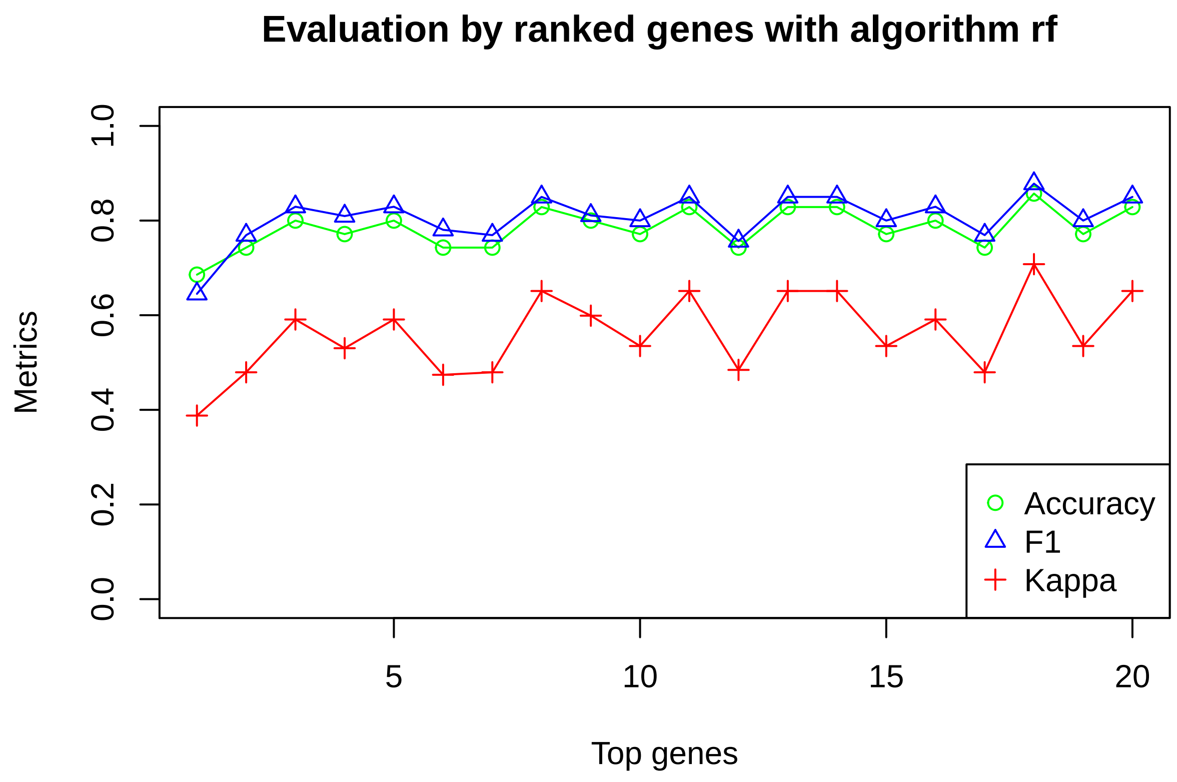


```
#RANKING ALGORITHM

#ALGORITHMO SELECTION
#model=fit.knn
#kalgoritmo="knn"
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
kalgoritmo="rf"

importance <- varImp(model, scale=TRUE)
# head(importance)

#Graph importance
plot(importance, main = paste("All variables with algorithm", kalgoritmo))
```

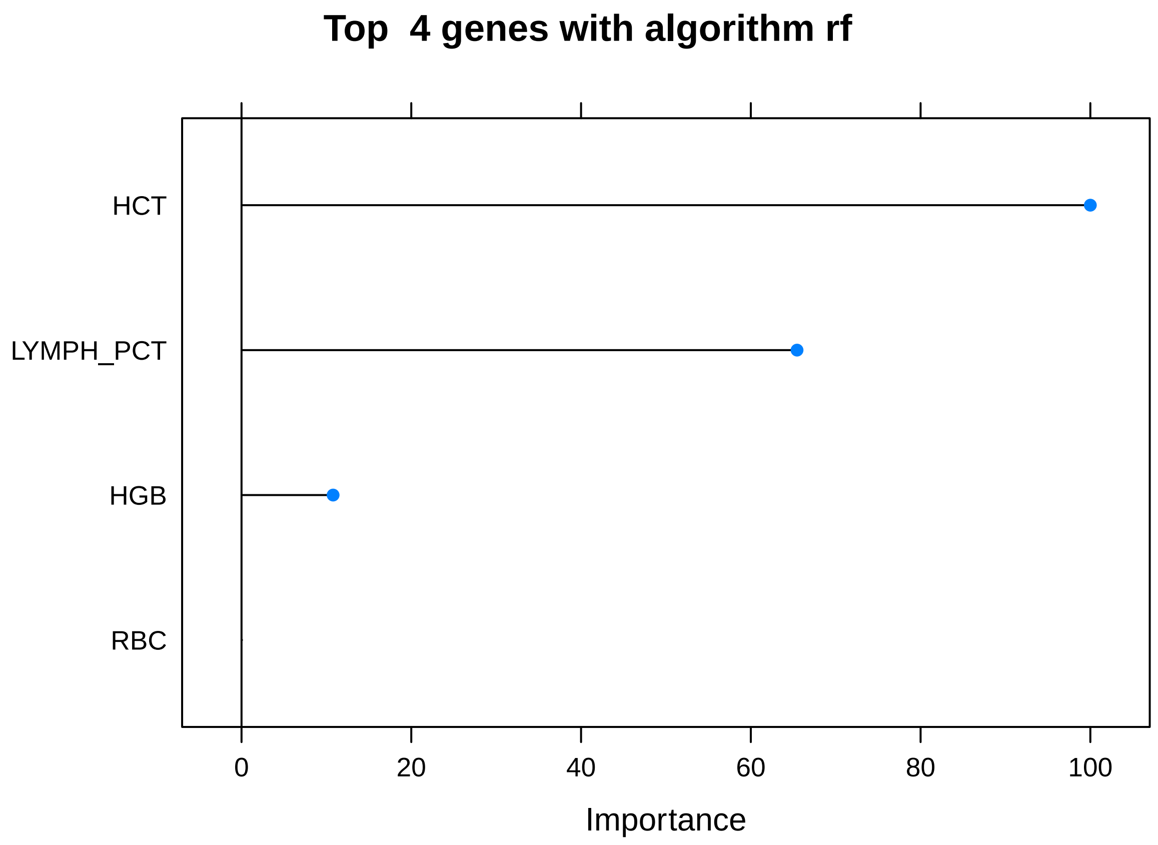


```
#INDEX
IndexRank <-data.frame(sort(importance$importance$Overall,
                          index.return = TRUE, decreasing = TRUE)[2])
#For SMV y KNN importance$importance$Control, para RF/cart usar $Overall
Ranking<-t(IndexRank)

DtrainingRanked<-Dtraining[,Ranking[1,]]
DtestingRanked<-Dtesting[,Ranking[1,]]

DataRanked<-Data[,Ranking[1,]]
# write.csv(DataRanked, "All_data_ranked.csv")

#EVALUATION OF DIF SUBSET OF GENES TOP
#Numero de genes top a evaluar
kvalue=20


kEvaluacion<-matrix(,nrow=kvalue, ncol=19) # 18 parametros and one extra for other measurements
colnames(kEvaluacion)<-c("Accuracy","Kappa","AccuracyLower",
                          "AccuracyUpper","AccuracyNull", "AccuracyPValue",
                          "McnemarPValue", "Sensitivity", "Specificity",
                          "Pos Pred Value", "Neg Pred Value", "Precision",
                          "Recall","F1","Prevalence" ,"Detection Rate",
                          "Detection Prevalence","Balanced Accuracy", "K")

# for k=1 this is done separately as it would transform into vector
```

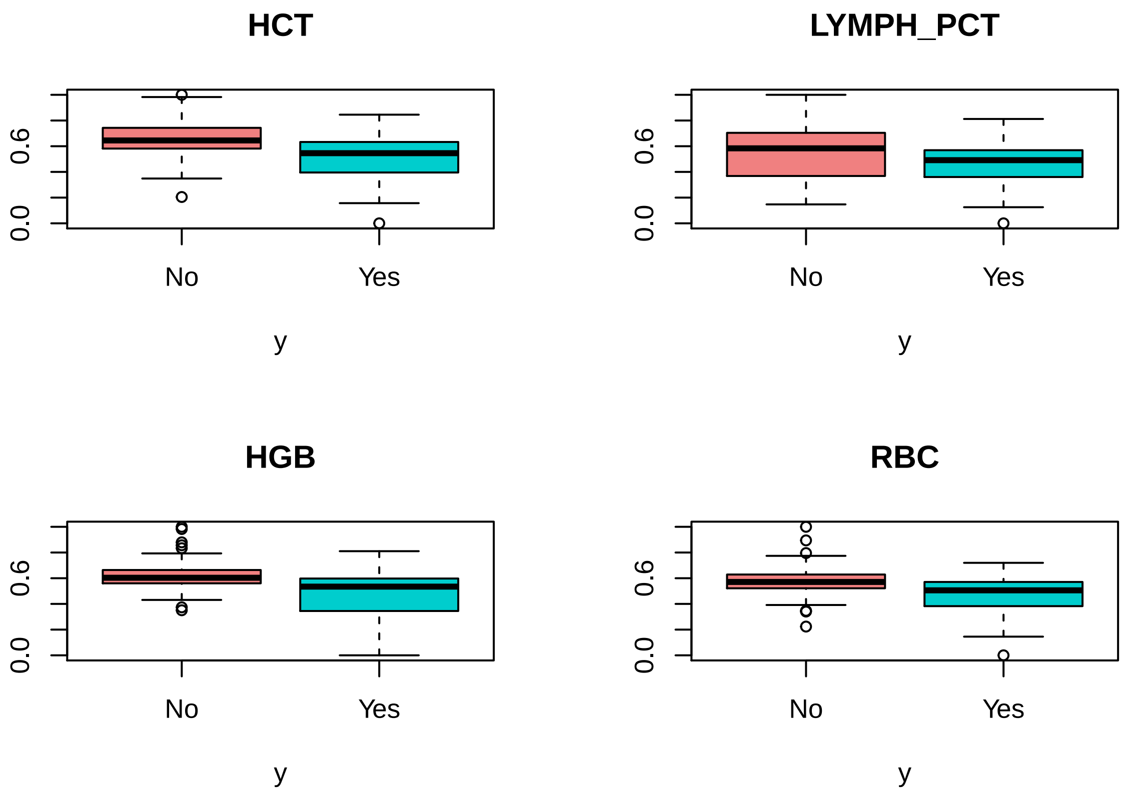
```r
k=1
DtrainK<-as.data.frame(DtrainingRanked[,1])
colnames(DtrainK)<-colnames(DtrainingRanked)[1]
DtestK<-as.data.frame(DtestingRanked[,1])
colnames(DtestK)<-colnames(DtestingRanked)[1]

fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                    trControl=control)
predictionsK <- predict(fit.algorK, DtestK)
StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
kEvaluacion[1,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))


for (k in 2:kvalue){
  #k=2
  DtrainK<-DtrainingRanked[,c(1:k)]
  DtestK<-DtestingRanked[,c(1:k)]
  fit.algorK <- train(DtrainK, Conditrain$Class, method=kalgoritmo, metric=metric,
                      trControl=control)
  predictionsK <- predict(fit.algorK, DtestK)
  StatisticsK<-confusionMatrix(predictionsK, Conditesting$Class)
  kEvaluacion[k,1:18]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))
  #not adding more columns
  #kEvaluacion[k,]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass)))
}

EvalK<-as.data.frame(kEvaluacion)

par(mfrow = c(1,1))
plot(EvalK$Accuracy,type="o",pch=1,col="green",
     main = paste("Evaluation by ranked genes with algorithm",
                  kalgoritmo,sed=""),xlab = "Top genes",
                  ylab = "Metrics", ylim=c(0,1))
lines(EvalK$F1,type = "o", pch=2, col="blue")
lines(EvalK$Kappa,type = "o", pch=3, col="red")
legend("bottomright",legend=c("Accuracy","F1","Kappa"),
       pch=c(1,2,3),col=c("green","blue","red"))
```



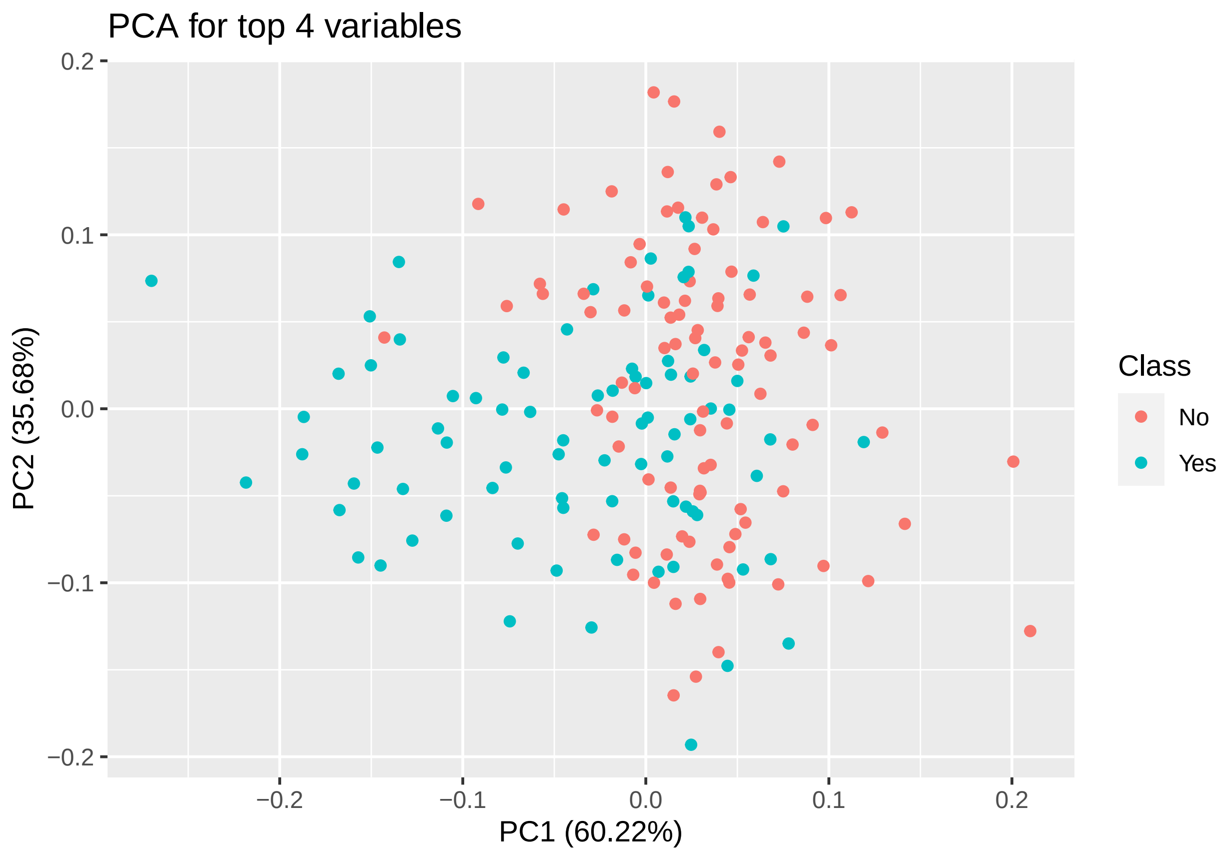
```
#for chosen value of k parameters:
ks=4
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]

DataRankedtopK<-DataRanked[,c(1:ks)]
# write.csv(DataRankedtopK, paste0("DataRankedtop",ks,"-",kalgoritmo,".csv"))

fit.algorKS <- train(DtrainKS, Conditrain$Class, method=kalgoritmo, metric=metric,
                     trControl=control)
importance <- varImp(fit.algorKS, scale=TRUE)
plot(importance, main = paste('Top ', ks, "genes with algorithm", kalgoritmo))
```



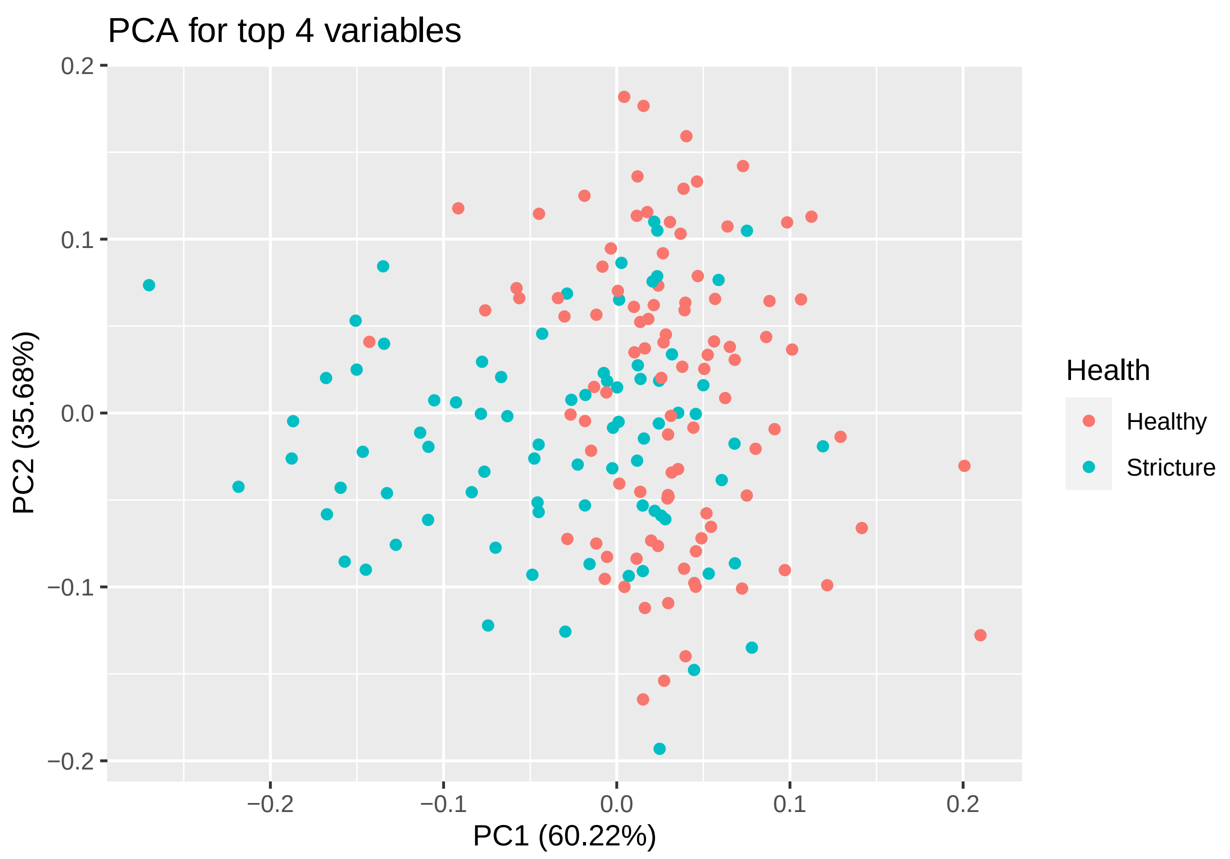
```
predictionsKS <- predict(fit.algorKS, DtestKS)
StatisticsKS<-confusionMatrix(predictionsKS, Conditesting$Class)

xR <- DtrainKS
yR <- Conditrain$Class
sxR <- DtrainingRanked[,c(3,4,1,2)]
#you can rerun specific parameters here

#boxplot
par(mfrow=c(2,2))

for(i in 1:4) {
  boxplot(xR[,i]~y, main=names(xR)[i],col=c("lightcoral","cyan3"),ylab = " ")
}
```



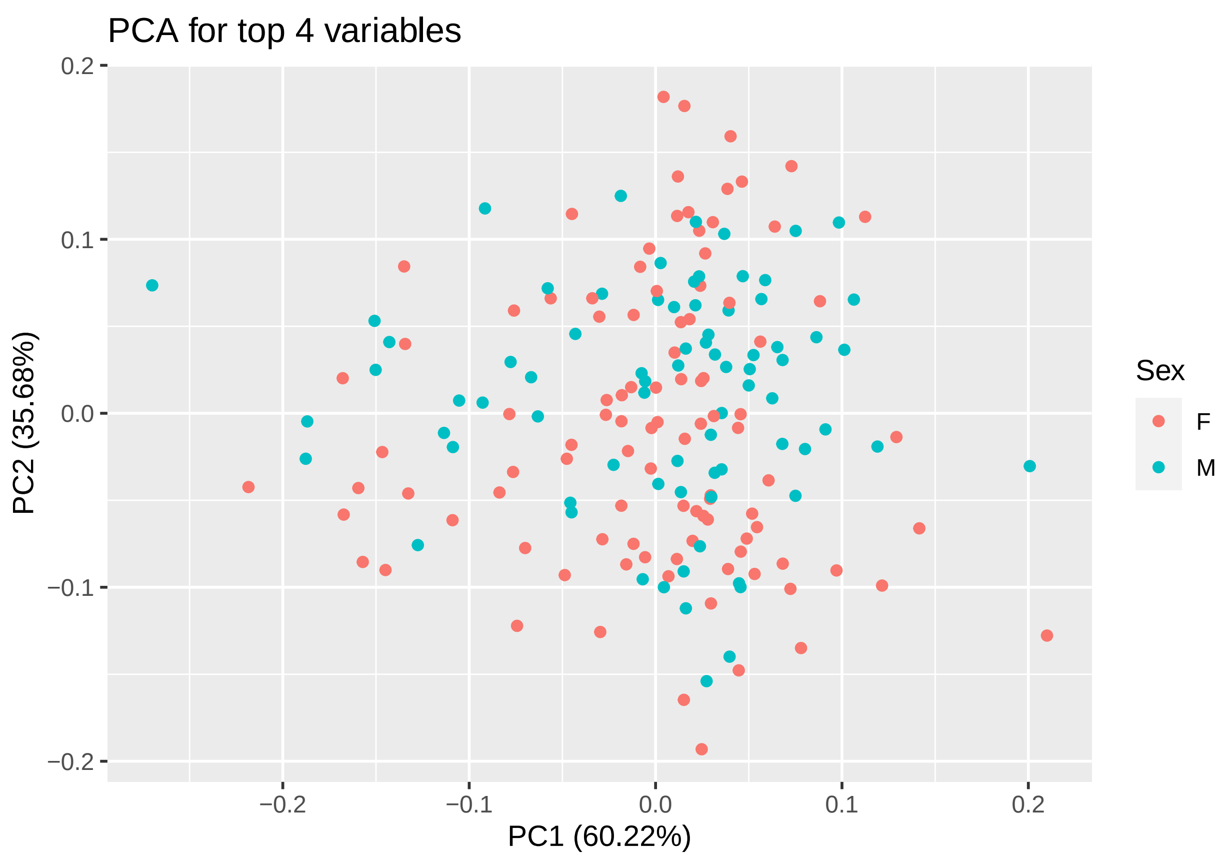
```
par(mfrow=c(1,1))


pca<-prcomp(DataRanked[,1:ks])
autoplot(pca, data = conditions, colour = 'Class',
         main =paste("PCA for top",ks, "variables"))
```



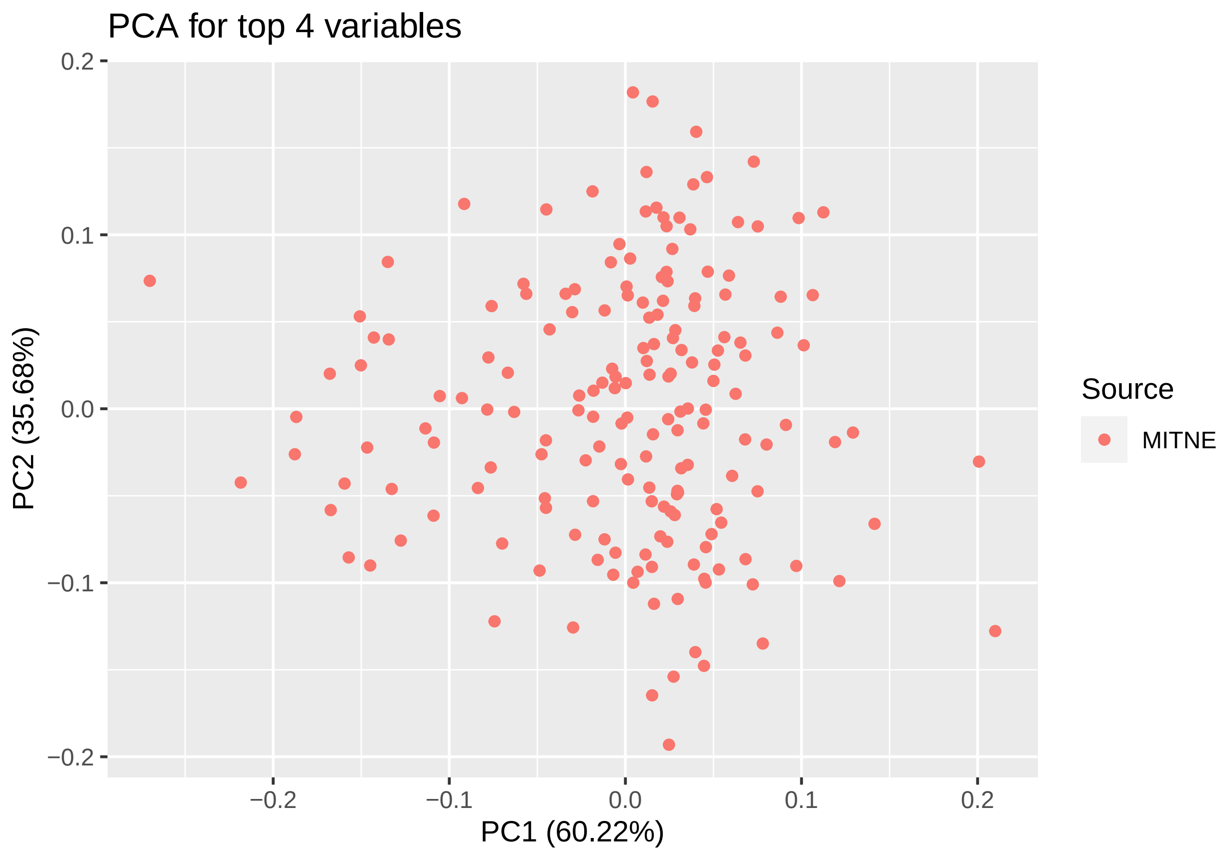
```
autoplot(pca, data = conditions, colour = 'Health',
        main =paste("PCA for top",ks, "variables"))
```



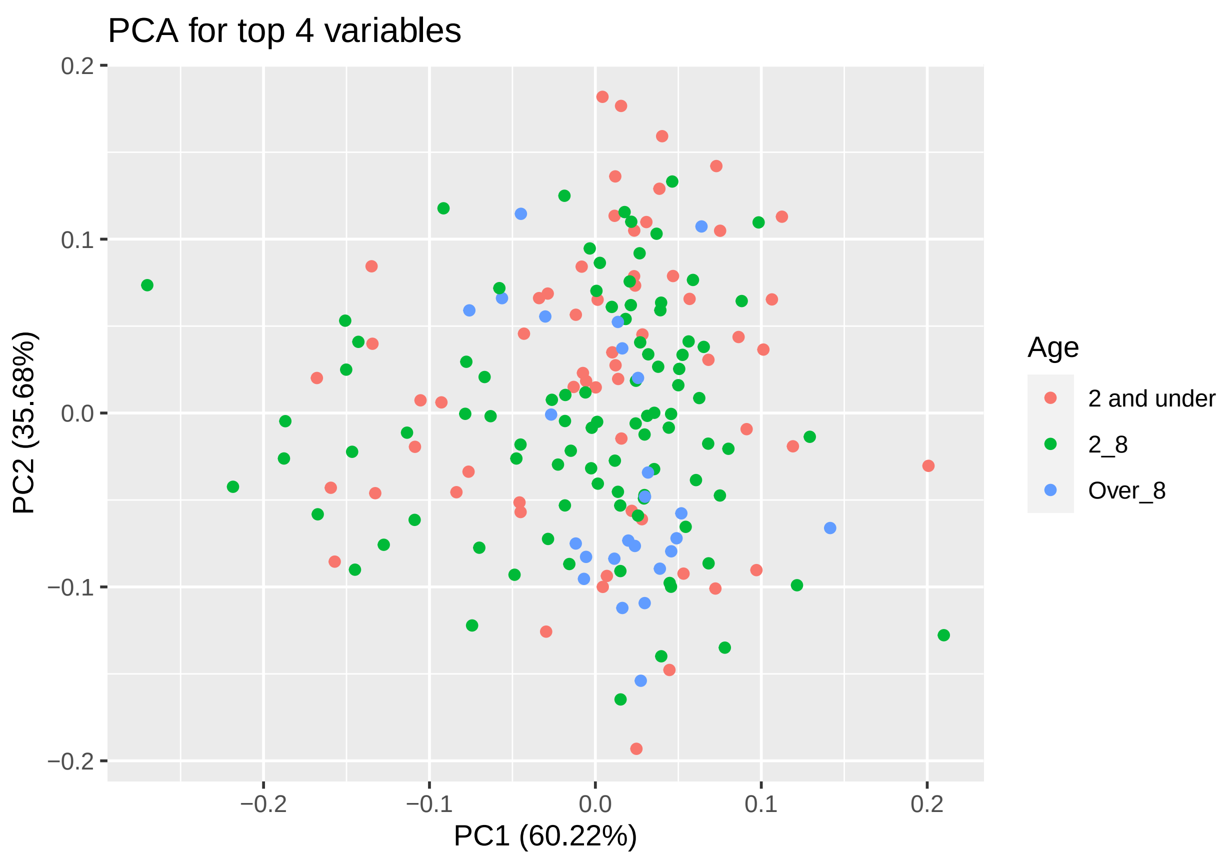
```r
autoplot(pca, data = conditions, colour = 'Sex',
         main =paste("PCA for top",ks, "variables"))
```



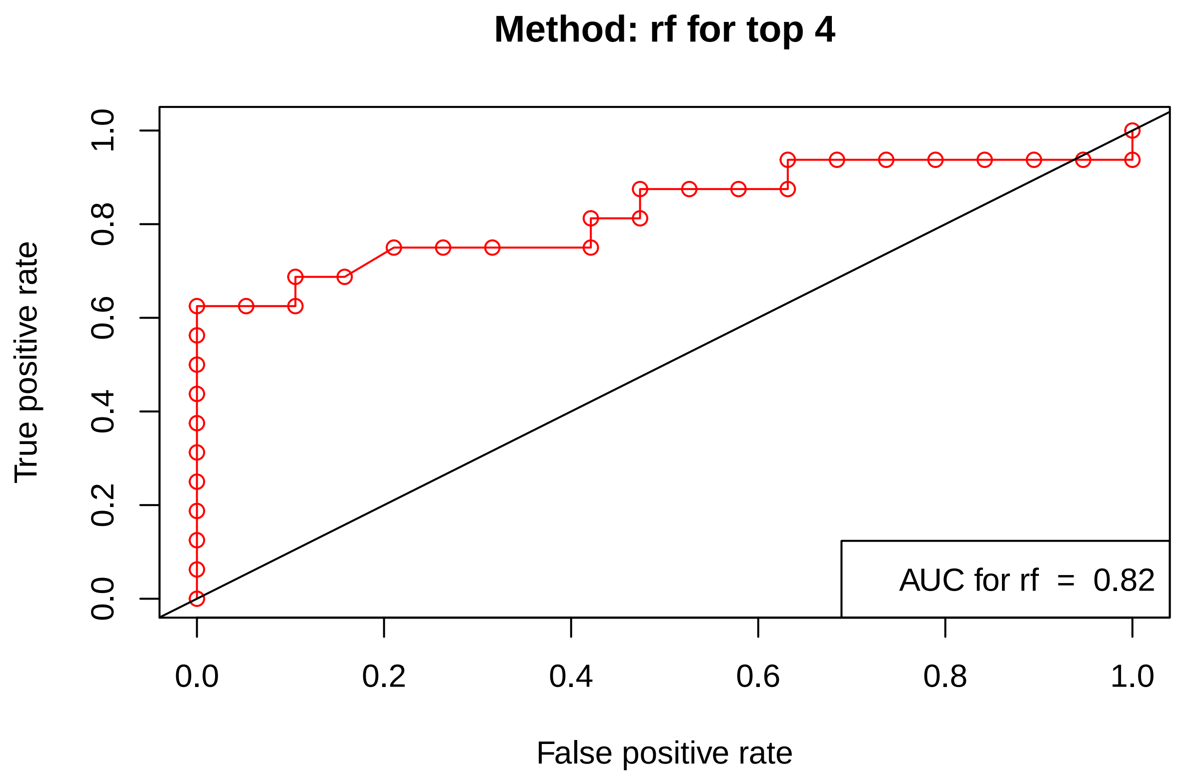
```
autoplot(pca, data = conditions, colour = 'Source',
         main =paste("PCA for top",ks, "variables"))
```



```
autoplot(pca, data = conditions, colour = 'Age',
         main =paste("PCA for top",ks, "variables"))
```



```
#PREDICTION
#FOR SVM, use:
#control <- trainControl(method="cv", number=10, classProbs=TRUE)

#fit.algorKS <- train(Class~., data=DtrainKS, method=kalgoritmo, metric=metric, trControl=control)
# -----------------------------------------------------------------------------
predictionsKSroc<- predict(fit.algorKS, DtestKS,type = "prob")[,2] #prob. clase=yes
predict.rocr  <- prediction (predictionsKSroc,Conditesting$Class)
perf.rocr     <- performance(predict.rocr,"tpr","fpr") #True/False positive.rate

# FIGURE ROC CURVE
# -----------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
# auc
par(mfrow = c(1,1))
plot(perf.rocr,type='o', col = "red",
     main = paste("Method:", kalgoritmo, "for top", ks),
     ylim=c(0,1.01),xlim=c(0,1))
abline(a=0, b=1)
legend("bottomright",legend= paste('AUC for', kalgoritmo, ' = ', round(auc,2)))
```



```
#COMPARISON CASE BY CASE OF TESTING SET
PRED<-as.data.frame(c(predictionsKS))
PRED2<-as.data.frame(c(Conditesting$Class))
juntos<-as.data.frame(c(PRED,PRED2))

#Extraction of lists
ListaVariablestop<-as.data.frame(colnames(DtrainKS))
# write.csv(ListaVariablestop, file=paste(kalgoritmo,"Lista_top_variables.csv"))

#Extraction of lists
ListaVariablesall<-as.data.frame(colnames(DtrainingRanked))
# write.csv(ListaVariablesall, file=paste(kalgoritmo,"Lista_all_variables.csv"))
```