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 3b-d and Supplemental Figures 5a-c. Includes the following data: *fig3b - NMDS1 by source and IBD* fig3c - ratio of Bacteroides to Prevotella 9 *fig3d - ROC for serum chemistry and CBCs* Supplemental figure 5a - microbiome data by source and IBD status *Supplemental figure 5b - serum chemistry by source* Supplemental figure 5c - CBC by source

```
load("fig3_sfig5_data.RData")
```

```
#for figure 3 images
library(ggplot2)
library(ggthemes)
library(grid)
library(scales)
library(RColorBrewer)
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(ape)
library(phyloseq)
library(vegan)
```

```
## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.5-6
```

```
library(reshape2)


# for ML algorithms
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:vegan':
##
##     tolerance
```

```
library(ROCR)
library(rpart)
library(rattle)
```

```
## 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.
```

```
library(ellipse)
```

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

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

```
library(ggfortify)
library(vioplot)
```

```
## Loading required package: sm

## Package 'sm', version 2.2-5.6: type help(sm) for summary information

##
## Attaching package: 'sm'

## The following object is masked from 'package:rattle':
##
##     binning

## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(plotrix)
```

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

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

```
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")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:phyloseq':
##
##     sampleNames
```

```
library(RColorBrewer)
library(kmed)


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  grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] kmed_0.3.0          gcrma_2.58.0        affy_1.64.0
##  [4] Biobase_2.46.0      BiocGenerics_0.32.0 plotrix_3.7-8
##  [7] vioplot_0.3.5       zoo_1.8-8           sm_2.2-5.6
## [10] ggfortify_0.4.10    ellipse_0.4.2       rattle_5.4.0
## [13] bitops_1.0-6        tibble_3.0.3        rpart_4.1-15
## [16] ROCR_1.0-11         caret_6.0-86        reshape2_1.4.4
## [19] vegan_2.5-6         lattice_0.20-38     permute_0.9-5
## [22] phyloseq_1.30.0     ape_5.4             dplyr_1.0.0
## [25] RColorBrewer_1.1-2  scales_1.1.1        ggthemes_4.2.0
## [28] ggplot2_3.3.2
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-144          lubridate_1.7.9       tools_3.6.3
##  [4] affyio_1.56.0         R6_2.4.1              mgcv_1.8-31
```

```
##  [7] colorspace_1.4-1      ade4_1.7-15           nnet_7.3-12
## [10] withr_2.2.0           tidyselect_1.1.0      gridExtra_2.3
## [13] preprocessCore_1.48.0 compiler_3.6.3        stringr_1.4.0
## [16] digest_0.6.25         rmarkdown_2.3         XVector_0.26.0
## [19] pkgconfig_2.0.3       htmltools_0.5.0       rlang_0.4.7
## [22] generics_0.0.2        jsonlite_1.7.0        ModelMetrics_1.2.2.2
## [25] magrittr_1.5          biomformat_1.14.0     Matrix_1.2-18
## [28] Rcpp_1.0.5            munsell_0.5.0         S4Vectors_0.24.4
## [31] Rhdf5lib_1.8.0        lifecycle_0.2.0       stringi_1.4.6
## [34] pROC_1.16.2           yaml_2.2.1            MASS_7.3-51.6
## [37] zlibbioc_1.32.0       rhdf5_2.30.1          plyr_1.8.6
## [40] recipes_0.1.13        crayon_1.3.4          Biostrings_2.54.0
## [43] splines_3.6.3         multtest_2.42.0       knitr_1.29
## [46] pillar_1.4.6          tcltk_3.6.3           igraph_1.2.5
## [49] codetools_0.2-16      stats4_3.6.3          glue_1.4.1
## [52] evaluate_0.14         BiocManager_1.30.10   data.table_1.12.8
## [55] vctrs_0.3.1           foreach_1.5.0         gtable_0.3.0
## [58] purrr_0.3.4           tidyr_1.1.1           xfun_0.16
## [61] gower_0.2.2           prodlim_2019.11.13    class_7.3-15
## [64] survival_3.1-8        timeDate_3043.102     iterators_1.0.12
## [67] IRanges_2.20.2        cluster_2.1.0         lava_1.6.7
## [70] ellipsis_0.3.1        ipred_0.9-9
```

Figure 3b and Supplemental Figure 5a - IBD

Figure 3b & 5a code

Figure 3b plots the NMDS1 value for each sample divided by source or IBD status

```
# create otu table
otu_lower_allx_hvi = otu_table(lower_allx_hvi, taxa_are_rows = TRUE)

#create a taxa table
taxonomy = tax_table(taxo)

#convert to sample data matrix
meta_lower_allx_hvi <-sample_data(map_lower_allx_hvi)

# #check
# head(colnames(otu_lower_allx_hvi),20)
# head(rownames(meta_lower_allx_hvi),20)
#making sure rownames are colnames
colnames(otu_lower_allx_hvi)<-rownames(meta_lower_allx_hvi)
#create the phyloseq object
phylo_lower_allx_hvi <- phyloseq(otu_lower_allx_hvi, taxonomy, meta_lower_allx_hvi)

# create tree object
set.seed(1234)
tree_lower_allx_hvi <-rtree(ntaxa(phylo_lower_allx_hvi), rooted = TRUE,
                            tip.label = taxa_names(phylo_lower_allx_hvi))

#incorporate the tree object
```

```
phylo_lower_allx_hvi <- phyloseq(otu_lower_allx_hvi, taxonomy,
                                 meta_lower_allx_hvi, tree_lower_allx_hvi)

######### FILTERING
# subset by removing OTUs appearing < x times in < 1% of samples
wh0 = genefilter_sample(phylo_lower_allx_hvi, filterfun_sample(function(x) x>10),
                        A=0.01*nsamples(phylo_lower_allx_hvi))

phylo_lower_allx_hvi_filt = prune_taxa(wh0, phylo_lower_allx_hvi)
#obtain even sampling depth
phylo_lower_allx_hvi_filt = transform_sample_counts(phylo_lower_allx_hvi_filt,
                                                    function(x) 1E6 * x/sum(x))

# ntaxa(phylo_lower_allx_hvi)
# ntaxa(phylo_lower_allx_hvi_filt)

##########function stressbar
stressbar<-function(physeq){
  n = 20
  stress <- vector(length = n)
  for (i in 1:n) {
    stress[i] <- ordinate(physeq, "NMDS", "bray",k=i)$stress
  }
  names(stress) <- paste0(1:n, "Dim")

  par(mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 2)
  barplot(stress, ylab = "stress")
}

# ############################## ORDINATE
#### ABUNDANCE FILTER
ord.plh.f <-ordinate (phylo_lower_allx_hvi_filt, "NMDS", "bray",
                      trymax = 50, k=3)
p.f = plot_ordination(phylo_lower_allx_hvi_filt, ord.plh.f,
                      type="taxa", color="Phylum", title="taxa")
p.f = p.f + facet_wrap(~Phylum, 3)
p.fb = plot_ordination(phylo_lower_allx_hvi_filt, ord.plh.f,
                       type="samples", color="dev_ibd_src",shape="dev_ibd")
p.fb = p.fb + geom_point(size=1) + ggtitle("samples")

#### get coordinates for the ordination
a<-cbind(p.fb$data)
# write.csv(a,"all_hvi_filtered.csv", row.names = FALSE)

a<-cbind(p.f$data)
# write.csv(a,"all_hvi_filtered_taxa.csv", row.names = FALSE)

scrs <-scores(ord.plh.f)
cent <-aggregate(scrs~dev_ibd_src, data=map_lower_allx_hvi, FUN="mean")
# write.csv(cent,"centroids.txt", row.names = FALSE)
b<-p.fb$data
vio_b<- ggplot(b, aes(x=dev_ibd_src,y=NMDS1,fill=Source)) +
  geom_violin(trim=FALSE) +
```

```
  stat_summary(fun=mean, geom="point",size=2, color="red") +
  scale_color_brewer(palette="Dark2") +
  labs(title="Plot of NMDS1 by source and IBD status",x="Source and IBD status", y="NMDS1") +
  theme(axis.text.x = element_text(size=7)) +
  scale_y_continuous(limits=c(-1.5,2.5), breaks=seq(-1.5,2.5,0.5))
```

Supplemental Figure 5a

Ordination plot for microbiome data by source and IBD status

```
p.fb
```

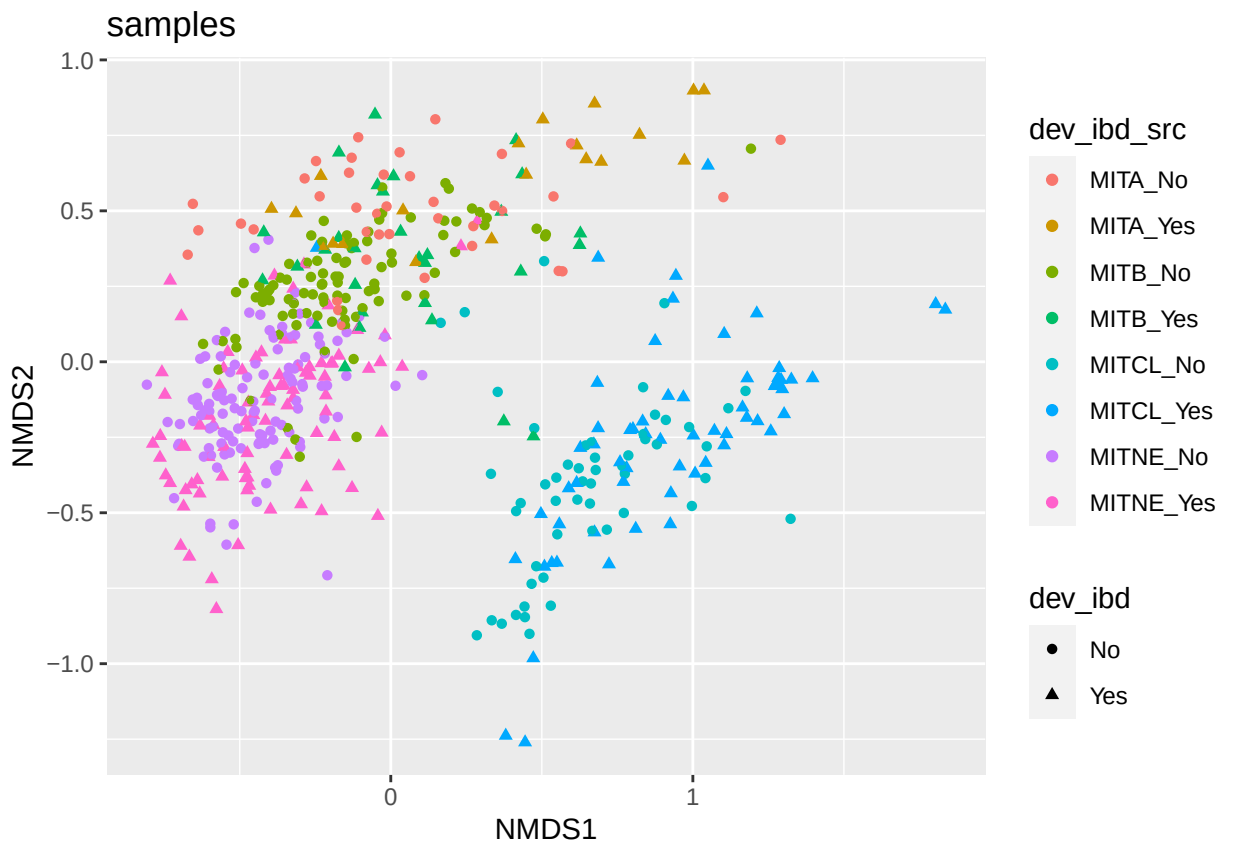


Figure 3b

```
vio_b
```

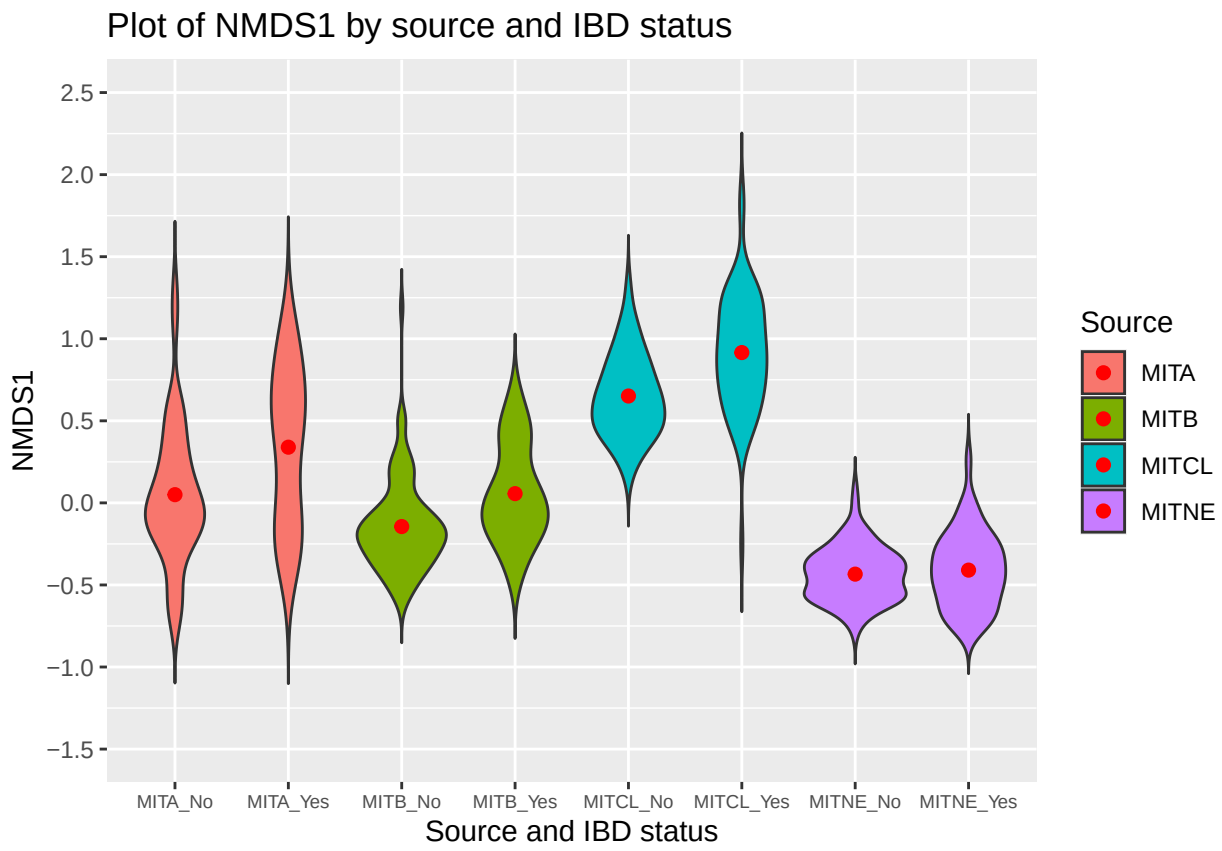
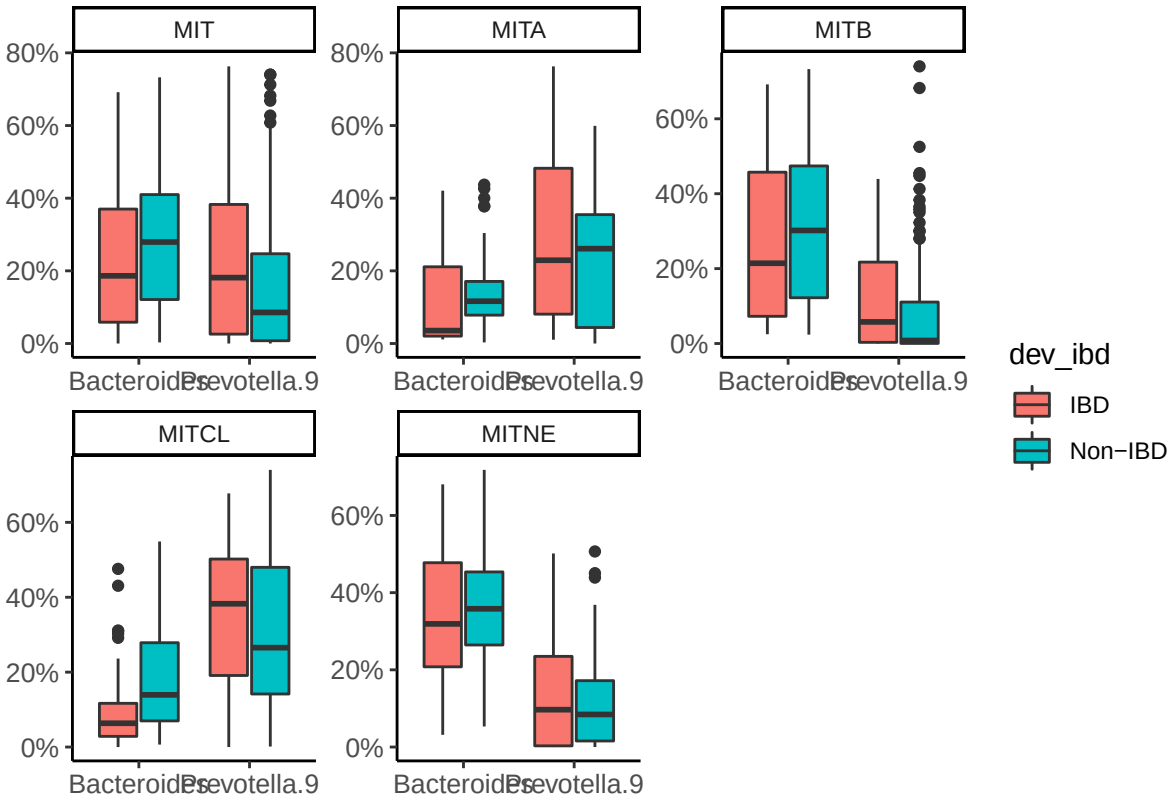


Figure 3c

Figure 3c creates boxplots comparing the abundance of Bacteroides and Prevotella 9 for the entire cohort and in subsets by source based on IBD status.

```
fig3c.melt <- melt(fig3c)
pm2 <- ggplot(fig3c.melt, aes(x=variable, y=value, fill=dev_ibd)) +
  geom_boxplot() +
  theme_classic() +
  theme(legend.position = "right",axis.text = element_text(size = 10),
        axis.title = element_text(size = 14, face = "bold")) +
  scale_y_continuous(labels = percent) +
  labs(x="", y = "")+
  facet_wrap(~Source, scale = "free")

pm2
```



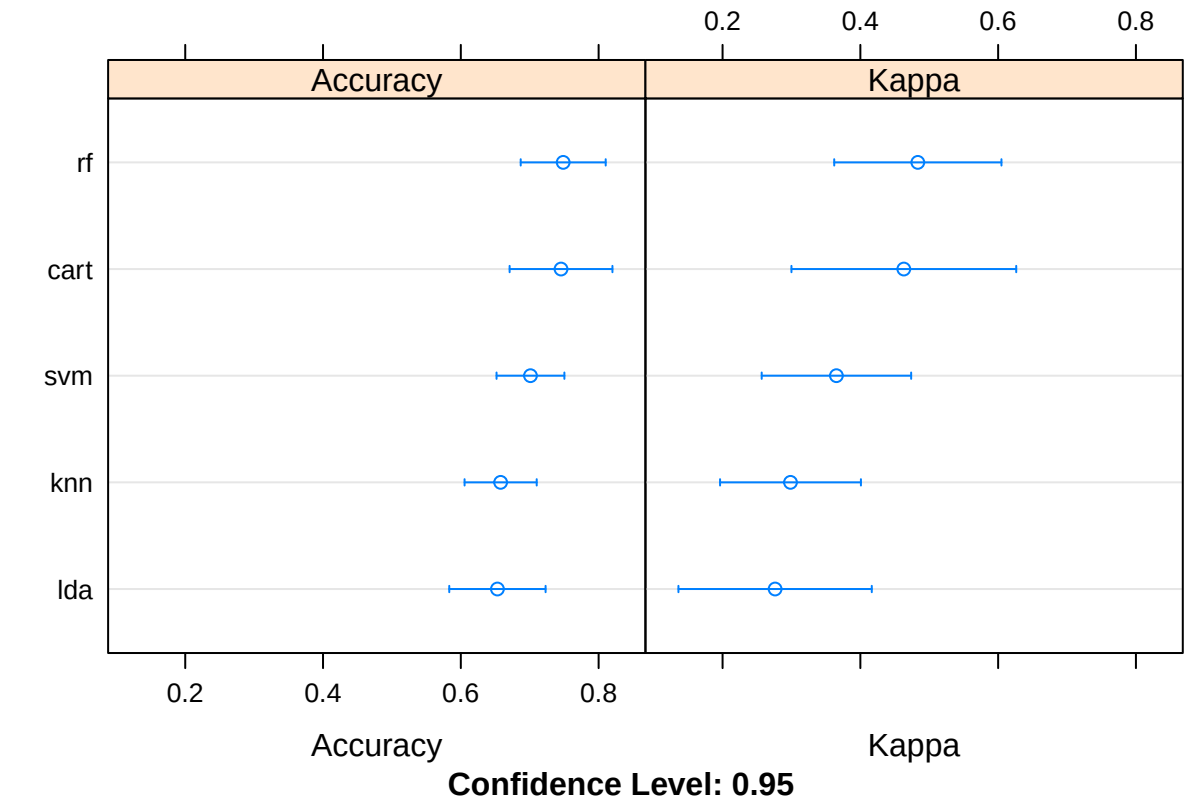
Machine Learning algorithm Machine learning algorithm originally developed by Jose Molina Mora and modified by Alex Sheh. Data normalized by min-max normalization prior to importation.

Figure 3d part I and Supplemental Figure 5b - ROC for serum chemistry

```
#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

# PART 1. LOAD DATA and metadata. Visualization of data
set.seed(5)
Data <- ibd_chem
conditions <-ibd_meta_chem
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Class', main ="PCA for all data")
```

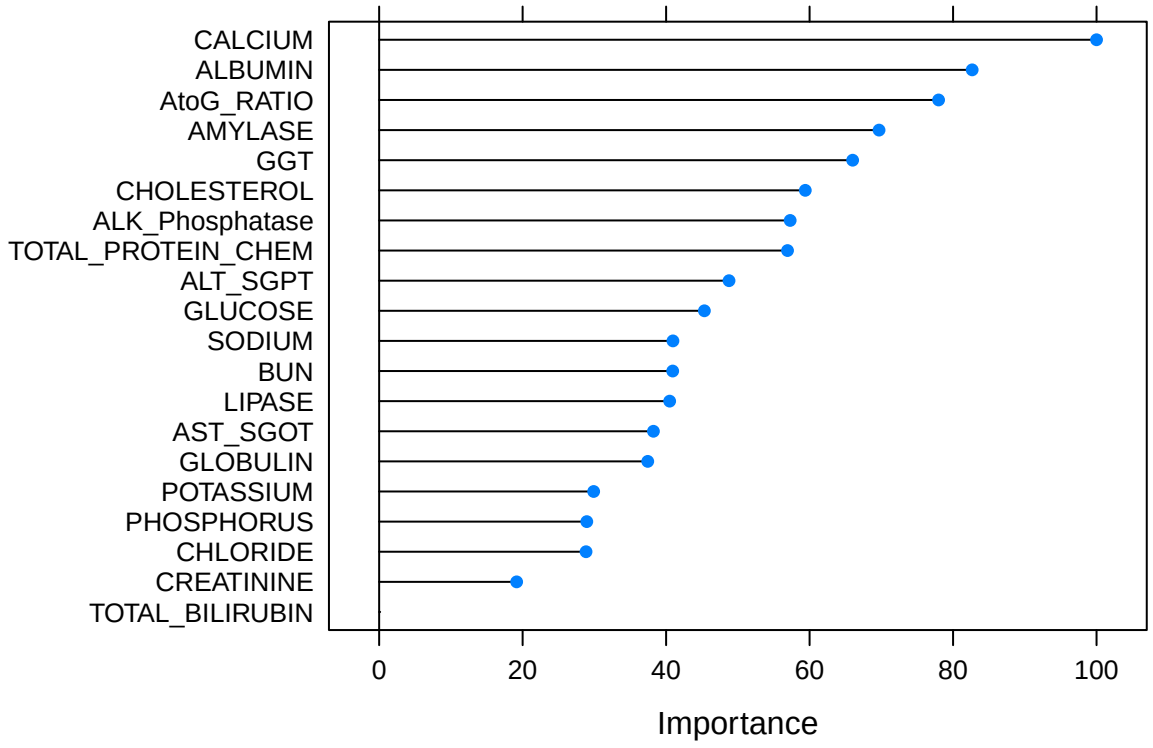

```
# Split the data into training (80%) and testing (20%) sets
test_index<-createDataPartition(conditions$Class, p=0.80, list = FALSE)
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]
Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

# Distribution in training set
percentage <- prop.table(table(Conditrain$Class)) * 100
#Distribution in testing set
percentage2 <- prop.table(table(Conditesting$Class))*100

# 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)
```



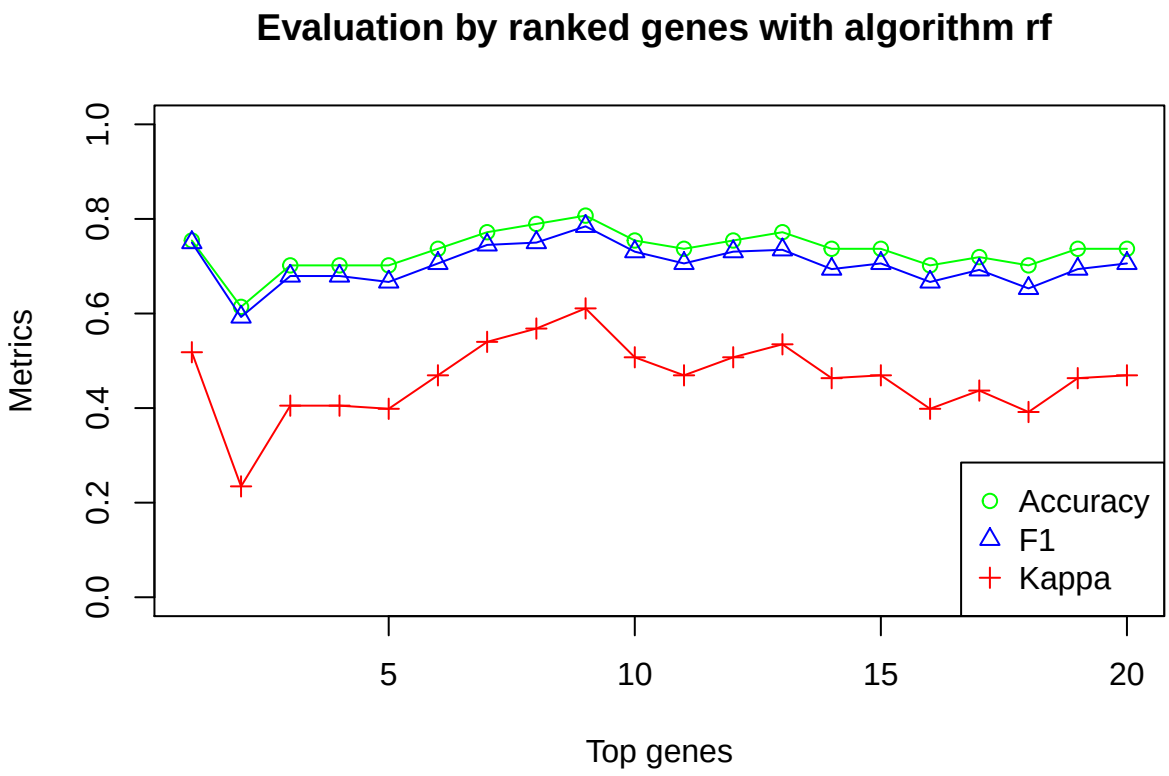

```
#RANKING
#model=fit.knn
```

```
#kalgoritmo="knn"
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
kalgoritmo="rf"
importance <- varImp(model, scale=TRUE)
plot(importance, main = paste("All variables with algorithm", kalgoritmo))
```

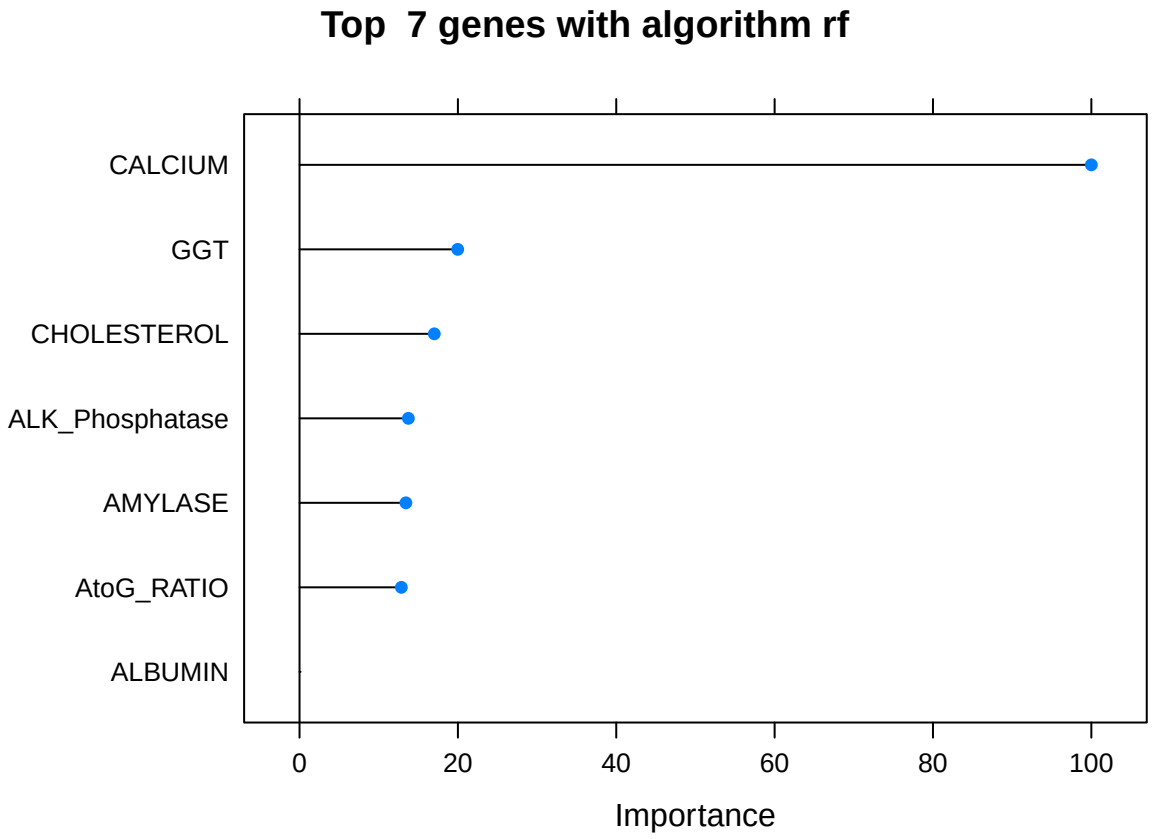



```
#INDEX
IndexRank <-data.frame(sort(importance$importance$Overall,
                            index.return = TRUE, decreasing = TRUE)[2])
Ranking<-t(IndexRank)

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

#EVALUATE
kvalue=20
kEvaluacion<-matrix(,nrow=kvalue, ncol=19)
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")
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)))
}
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"))
```




```
#choose K
ks=7
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]
DataRankedtopK<-DataRanked[,c(1:ks)]

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)

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

Supplemental Figure 5b

Ordination plot for serum chemistry for Healthy/IBD cohort based on source

```
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```

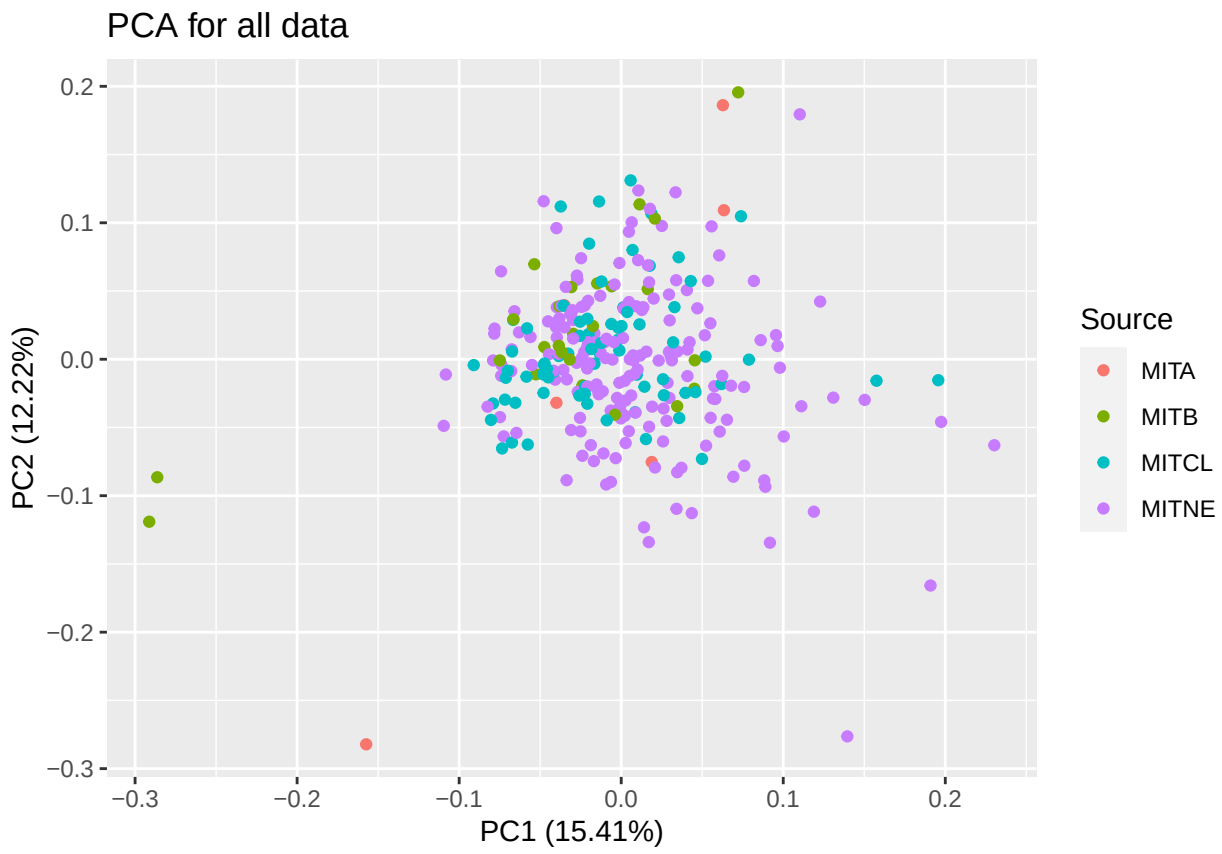


Figure 3d part I

ROC curve for serum chemistry

```
# ROC
# ------------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
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)))
```

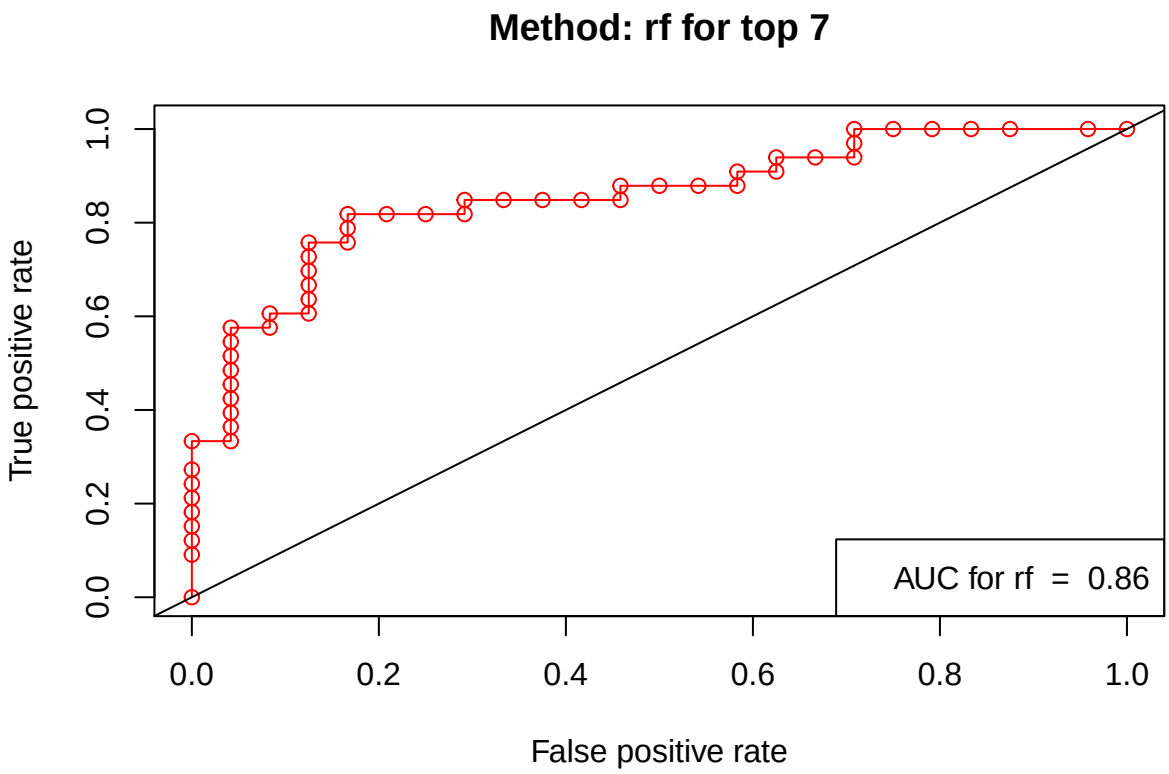
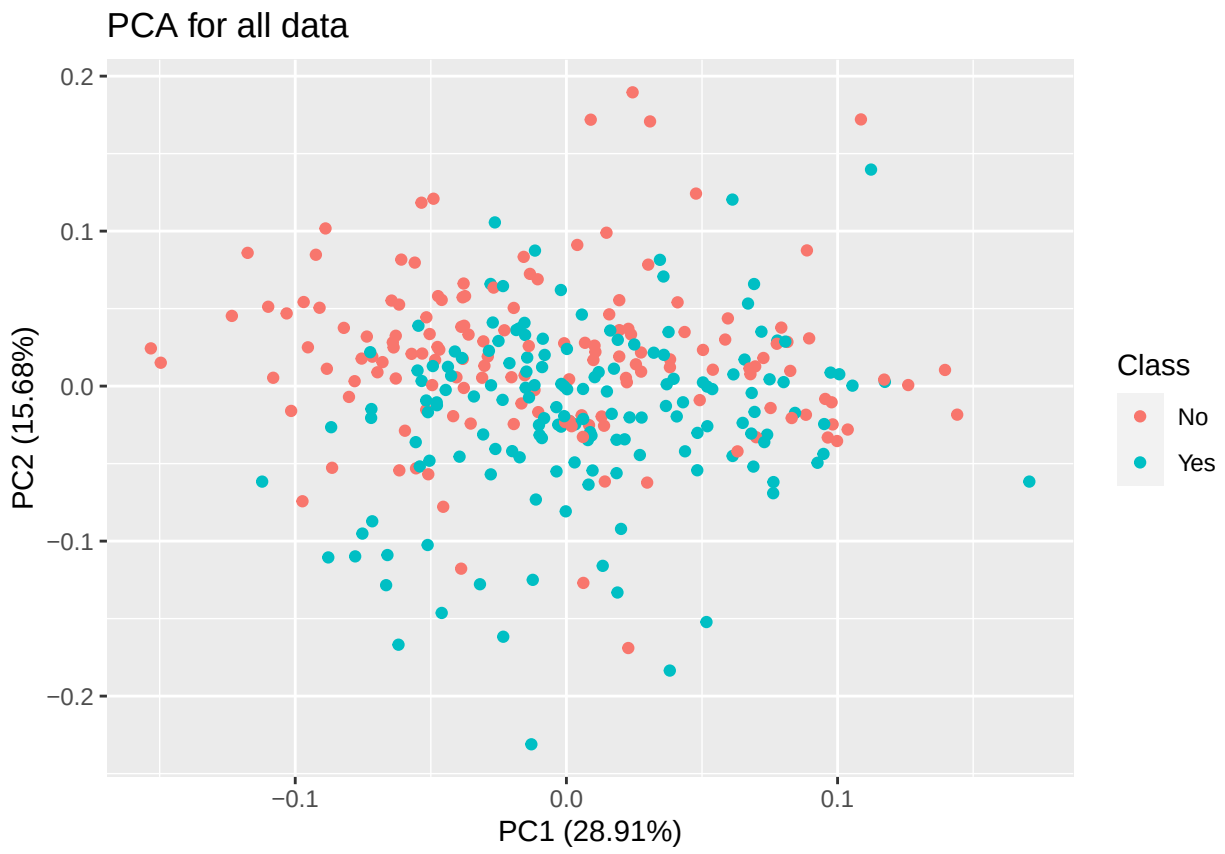


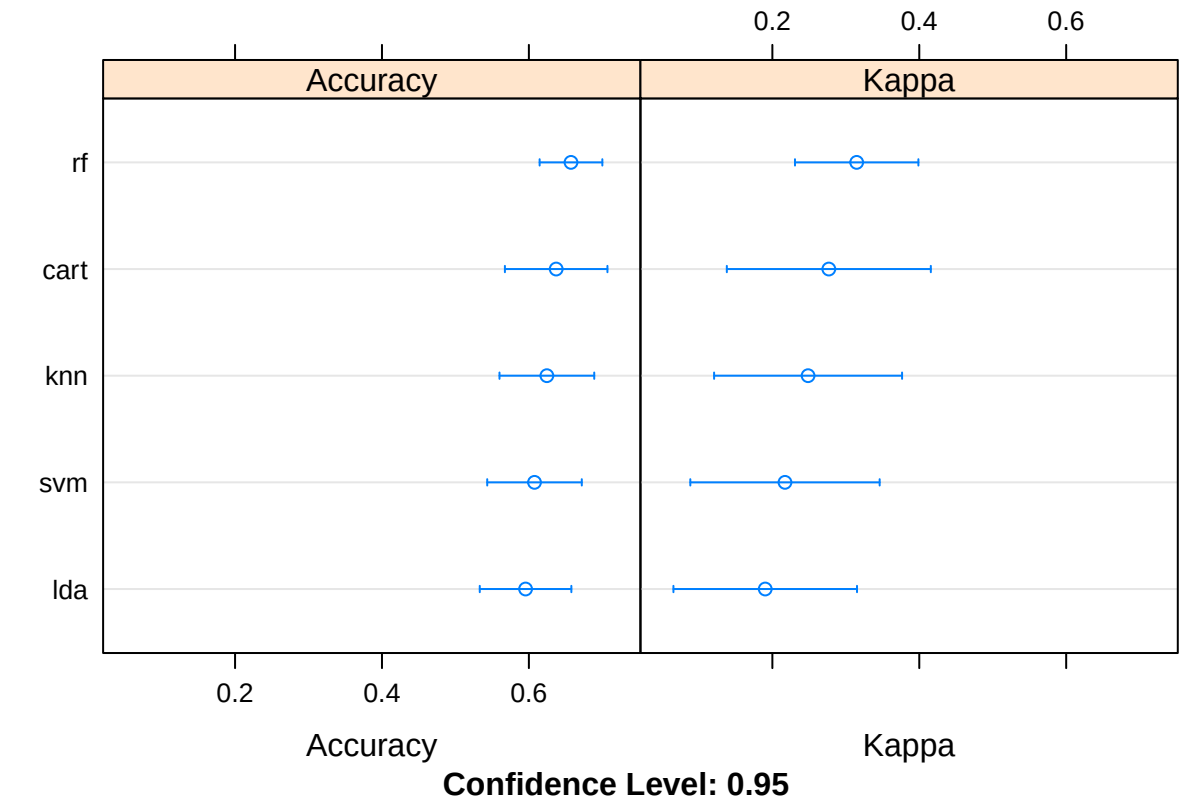
Figure 3d part II and Supplemental Figure 5c - ROC for CBC

```
#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

# PART 1. LOAD DATA and metadata. Visualization of data
set.seed(5)
Data <- ibd_cbc
conditions <-ibd_meta_cbc
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Class', main ="PCA for all data")
```



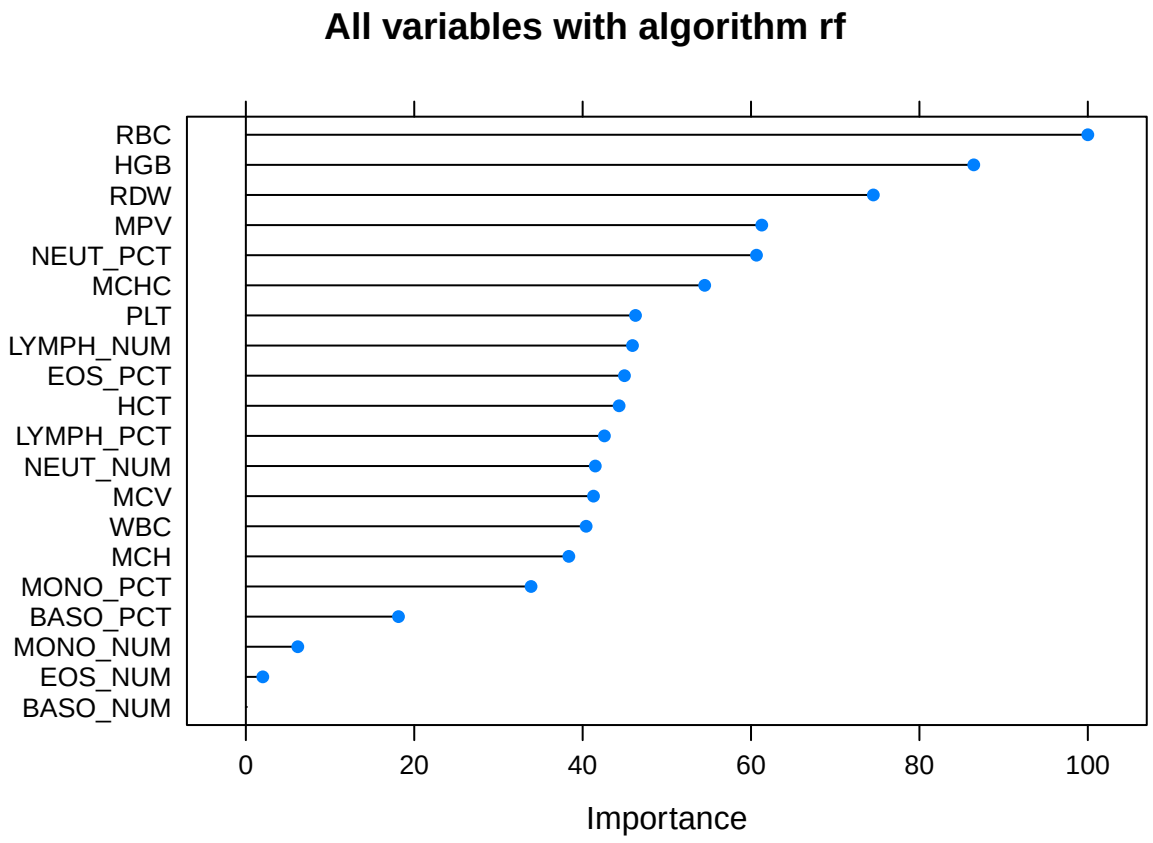

```
# Split the data into training (80%) and testing (20%) sets
test_index<-createDataPartition(conditions$Class, p=0.80, list = FALSE)
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]
Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

# Distribution in training set
percentage <- prop.table(table(Conditrain$Class)) * 100
#Distribution in testing set
percentage2 <- prop.table(table(Conditesting$Class))*100

# 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)
```



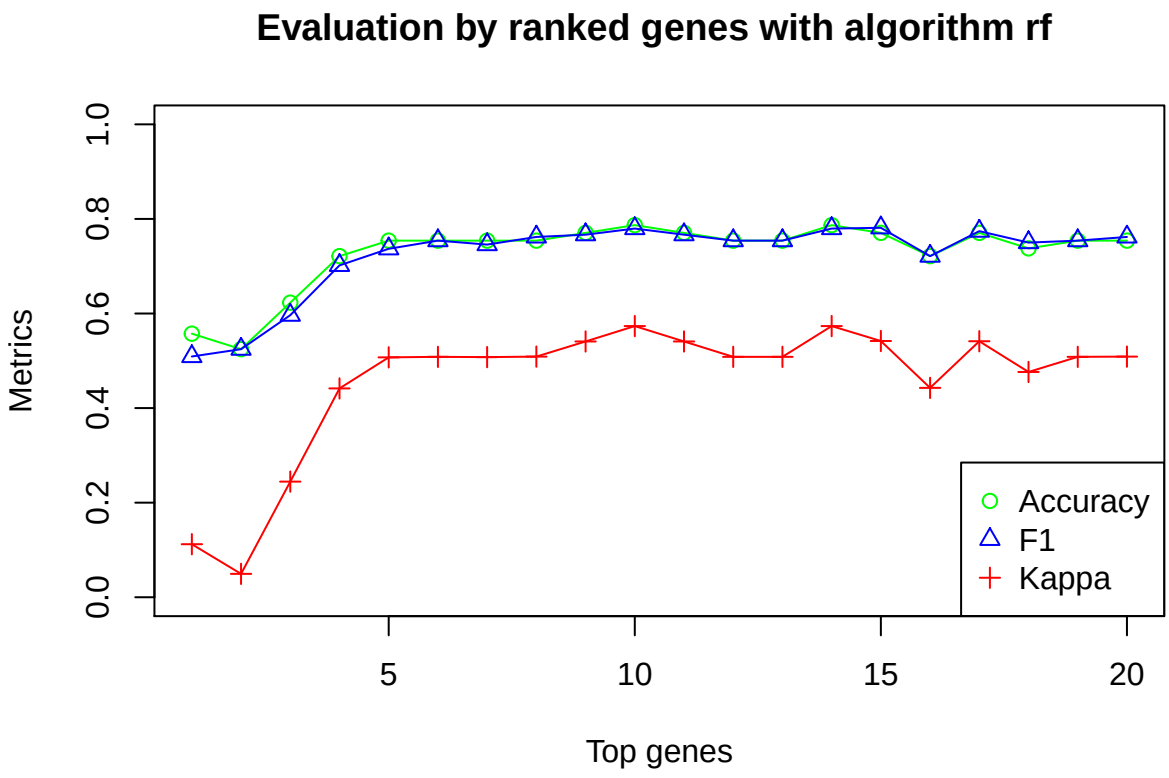

```
#RANKING
#model=fit.knn
```

```
#kalgoritmo="knn"
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
kalgoritmo="rf"
importance <- varImp(model, scale=TRUE)
plot(importance, main = paste("All variables with algorithm", kalgoritmo))
```

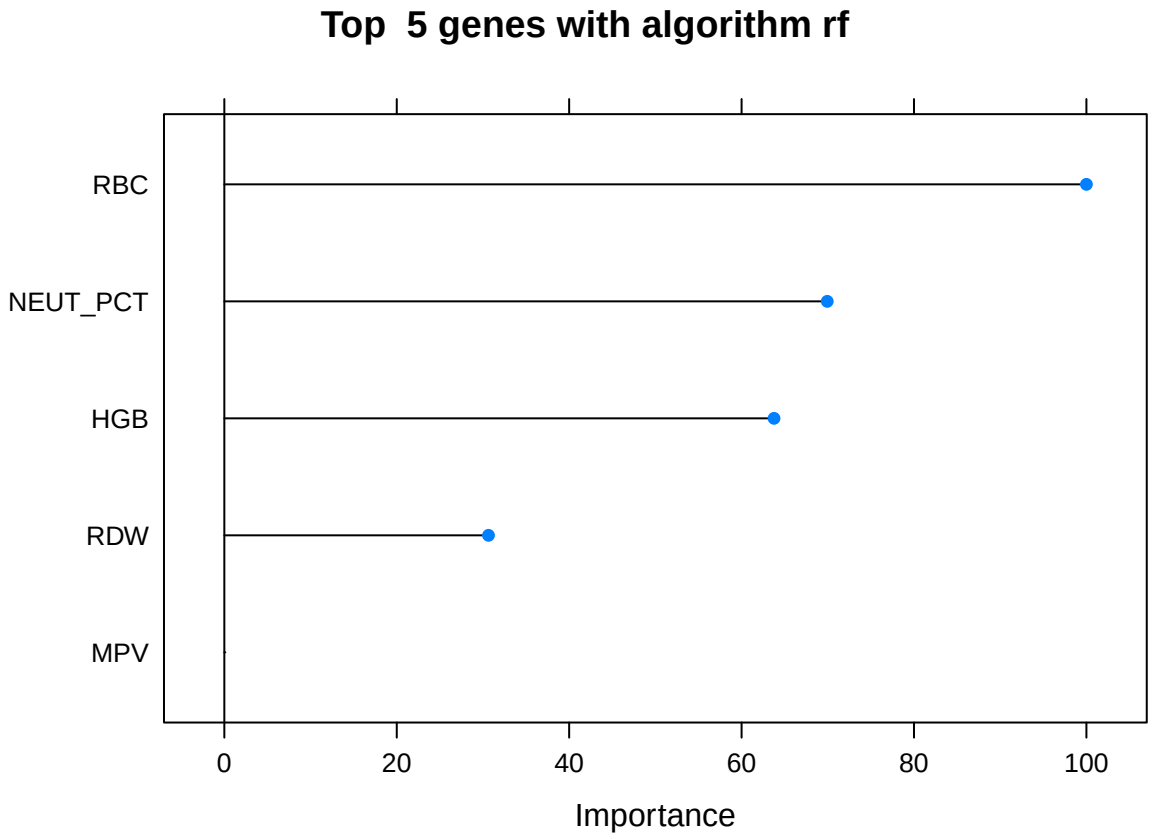



```
#INDEX
IndexRank <-data.frame(sort(importance$importance$Overall, index.return = TRUE,
                            decreasing = TRUE)[2])
Ranking<-t(IndexRank)

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

#EVALUATE
kvalue=20
kEvaluacion<-matrix(,nrow=kvalue, ncol=19)
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")
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)))
}
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"))
```




```
#choose K
ks=5
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]
DataRankedtopK<-DataRanked[,c(1:ks)]

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)

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

Supplemental Figure 5c

Ordination plot for CBCs for Healthy/IBD cohort based on source

```
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```

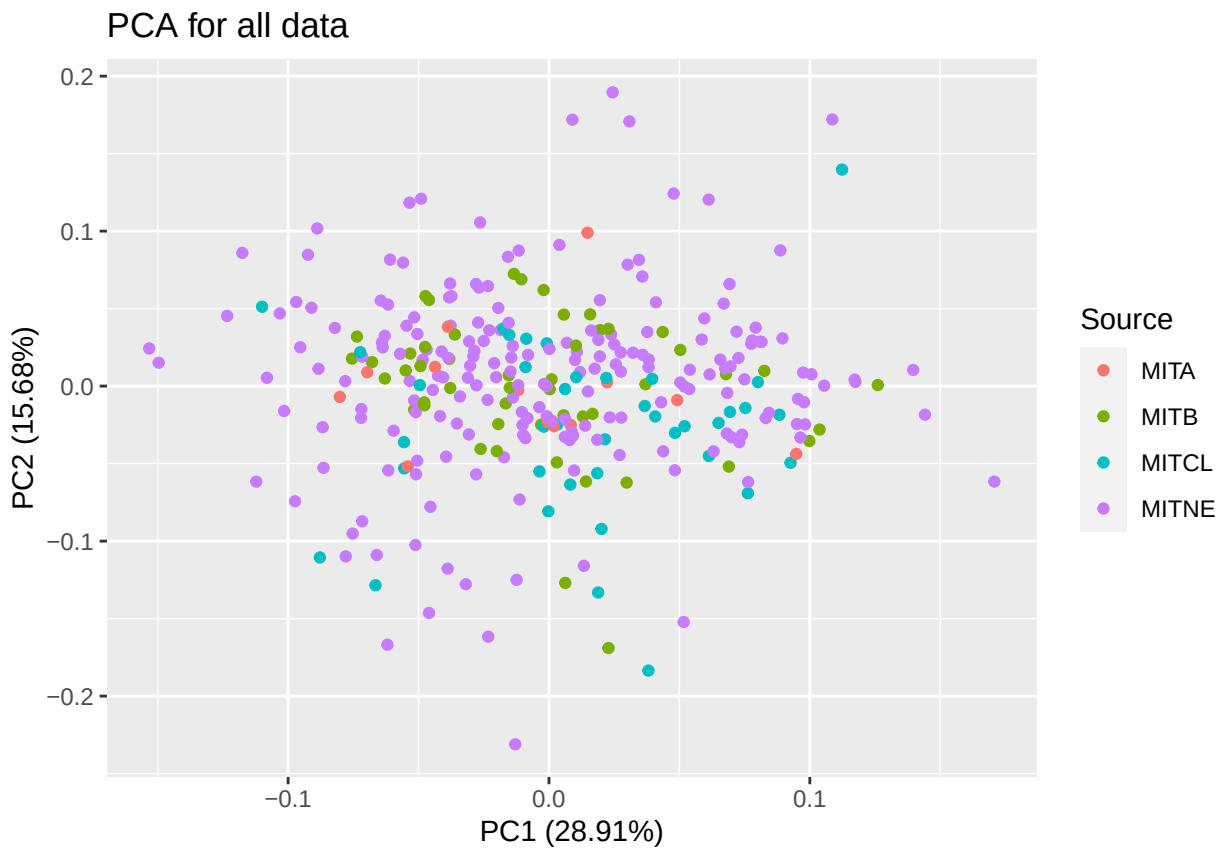
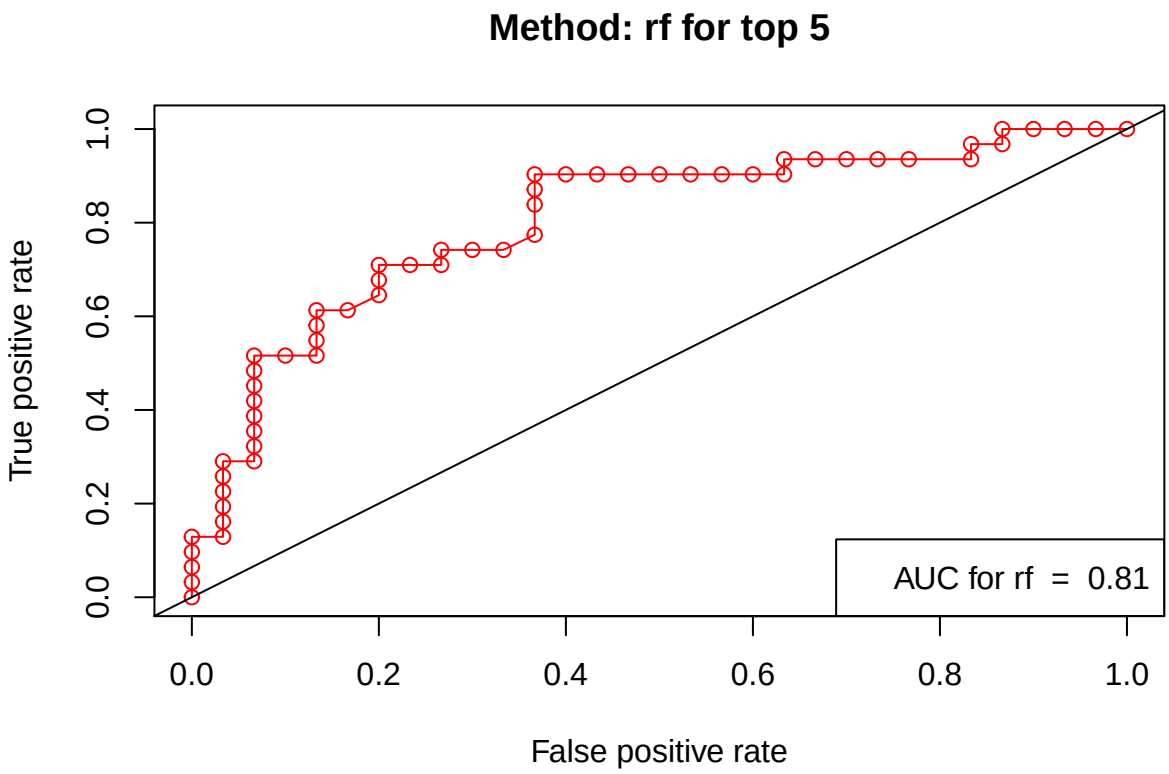


Figure 3d part II

ROC curve for CBC

```
# ROC
# ------------------------------------------------------------------------------
auc <- as.numeric(performance(predict.rocr ,"auc")@y.values)
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)))
```


Method: rf for top 5
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
AUC for rf = 0.81