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 Supplemental Figures 2&3. Includes the following data: *Supplemental figure 2a-d - PCA plots of microbiome data of healthy marmosets* Supplemental figure 3a - Comparison of multiple algorithms *Supplemental figure 3b - Evaluating stability by adding more ASVs* Supplemental figure 3c - Box and whisker plots for top 10 ASVs *Supplemental figure 3d - heatmap of microbiome data using top 10 ASVs selected by RF

```r
load("sfig23_data.RData")
```

```r
# for ML algorithms
library(caret)
```

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

```
## Loading required package: ggplot2
```

```r
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.
```

```r
library(ellipse)
```

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

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

```r
library(ggfortify)
library(plotrix)
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: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")'.
```

```r
library(RColorBrewer)
library(kmed)
library(DescTools)
```

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

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

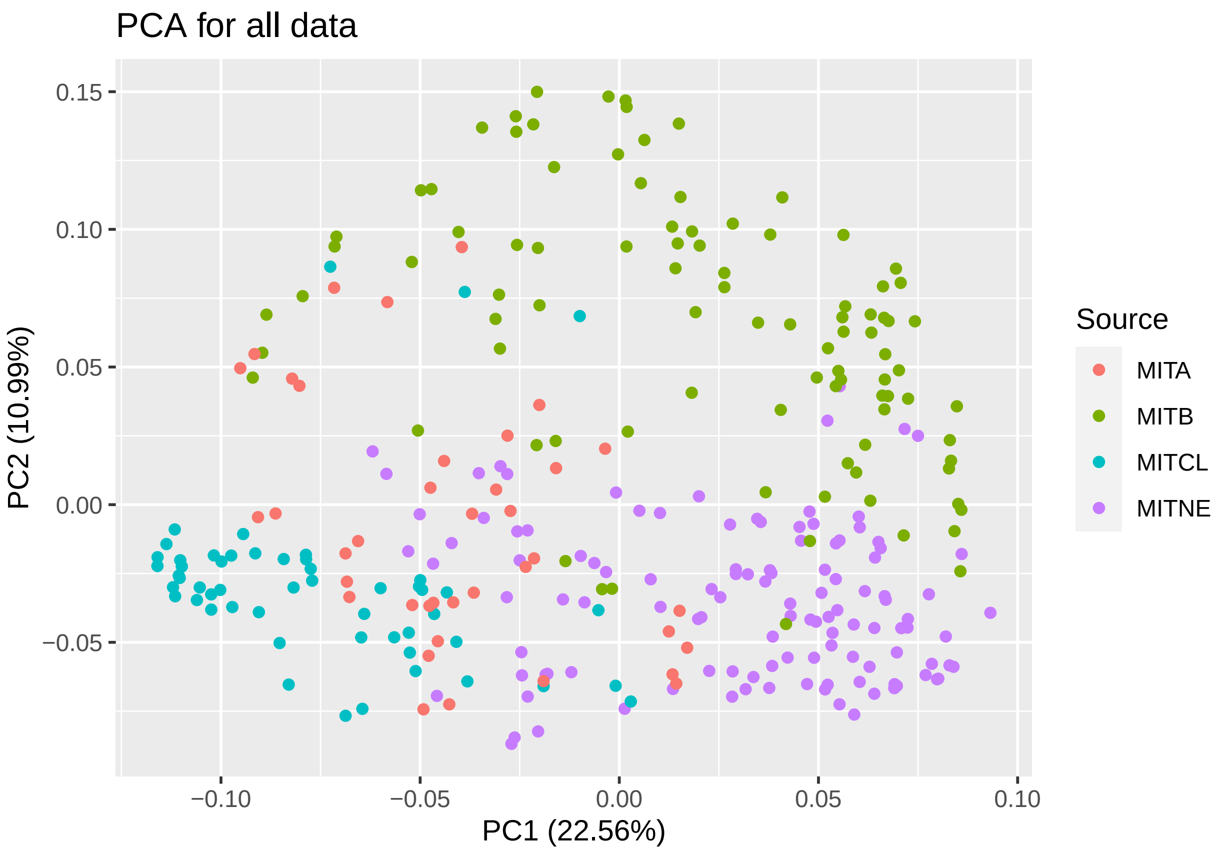
```r
sessionInfo()
```

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

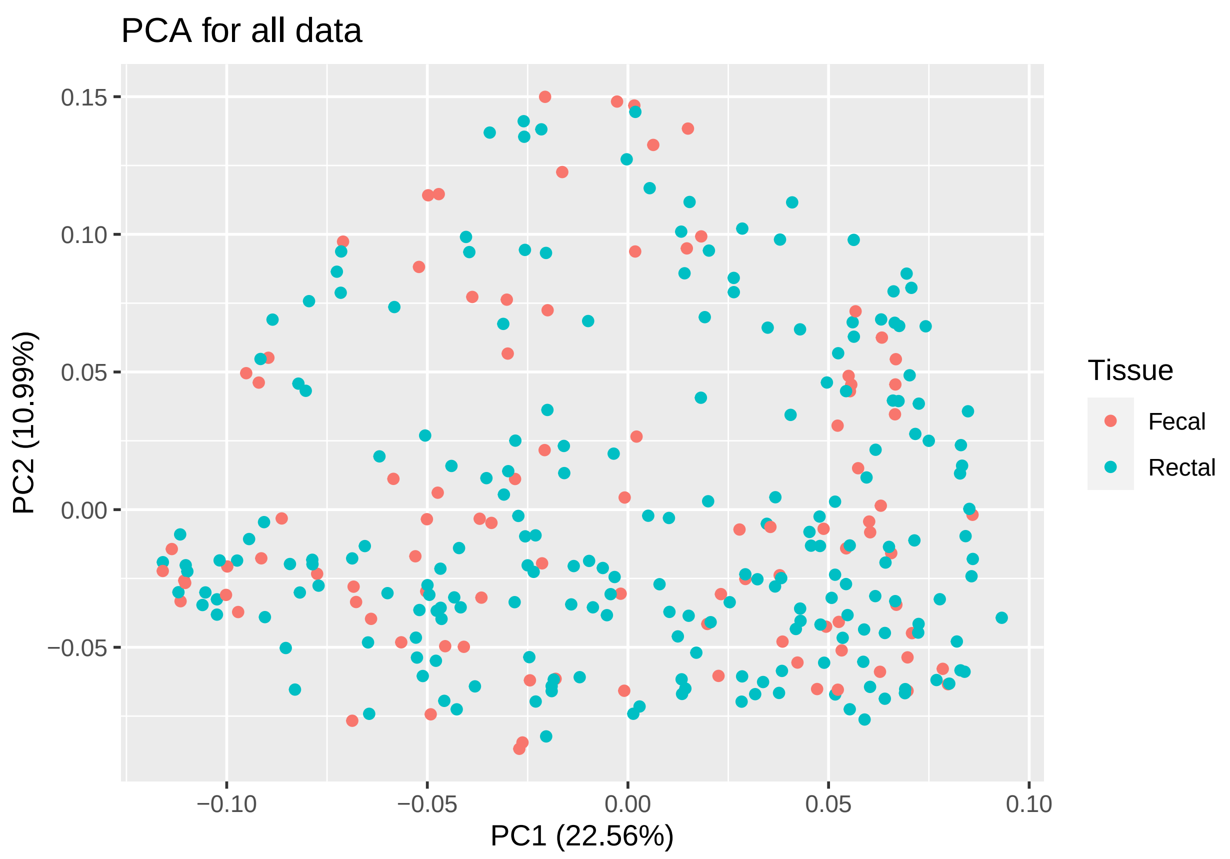
Supplemental Figure 2a-d - healthy

Uses different metadata to visualize healthy microbiome data

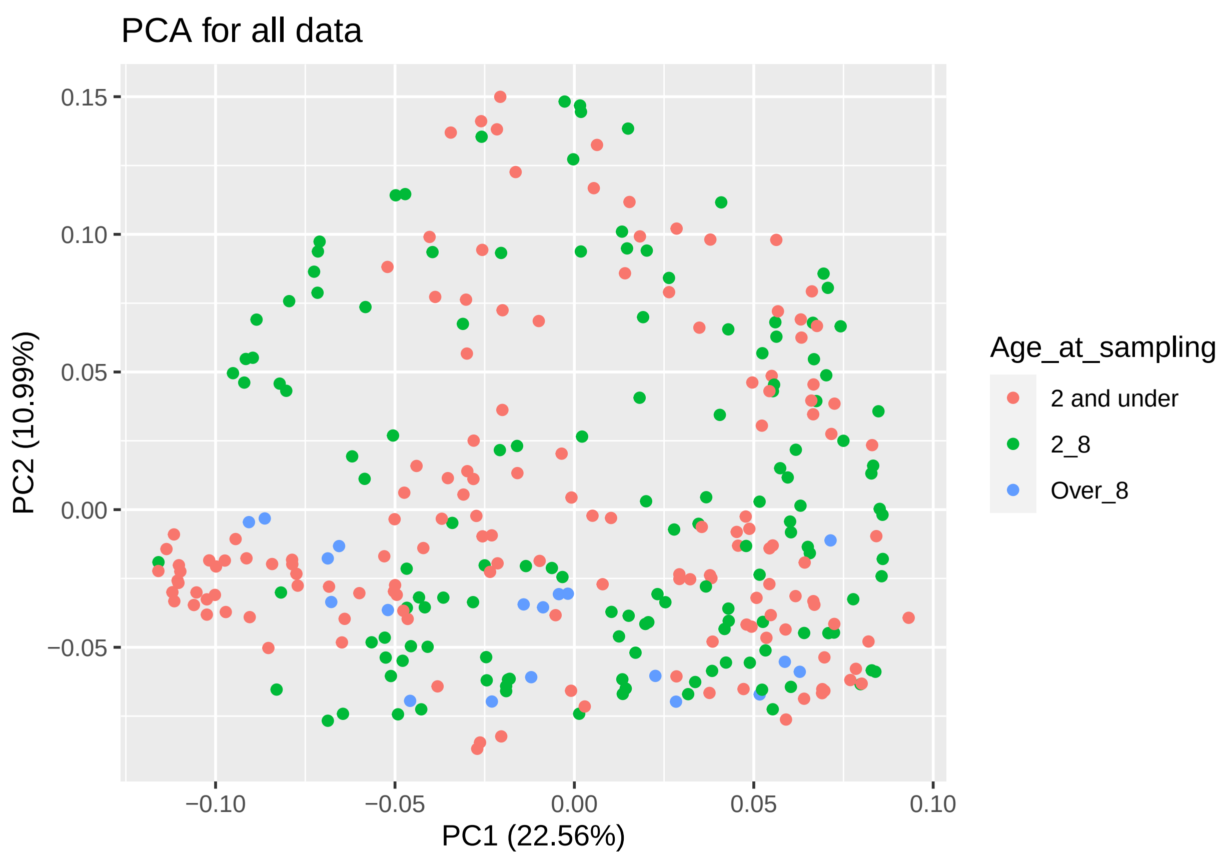
```
#boxcox to eliminate arch effect
Data <- BoxCox(hlt_otu,0.5)
conditions <- hlt_meta
pca<-prcomp(Data)
autoplot(pca, data = conditions, colour = 'Source', main ="PCA for all data")
```



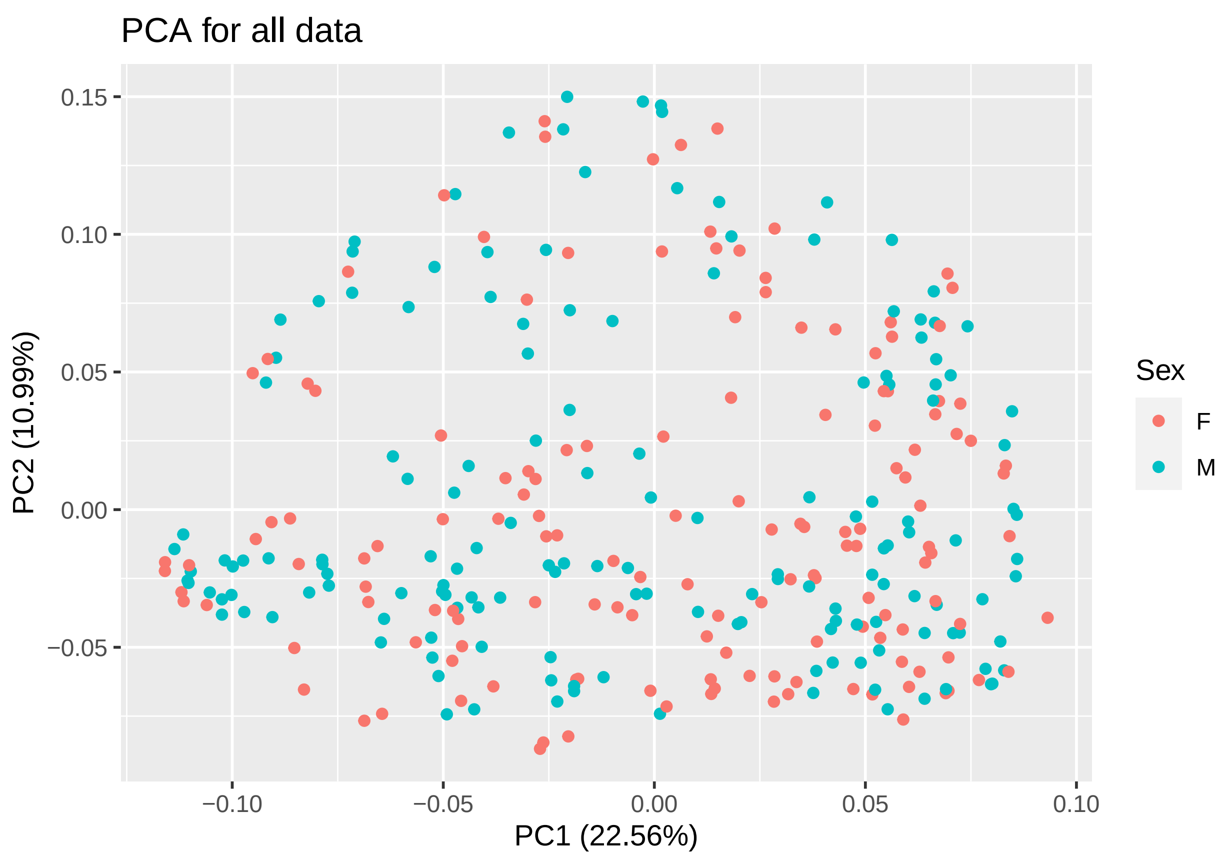
```
autoplot(pca, data = conditions, colour = 'Tissue', main ="PCA for all data")
```



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



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

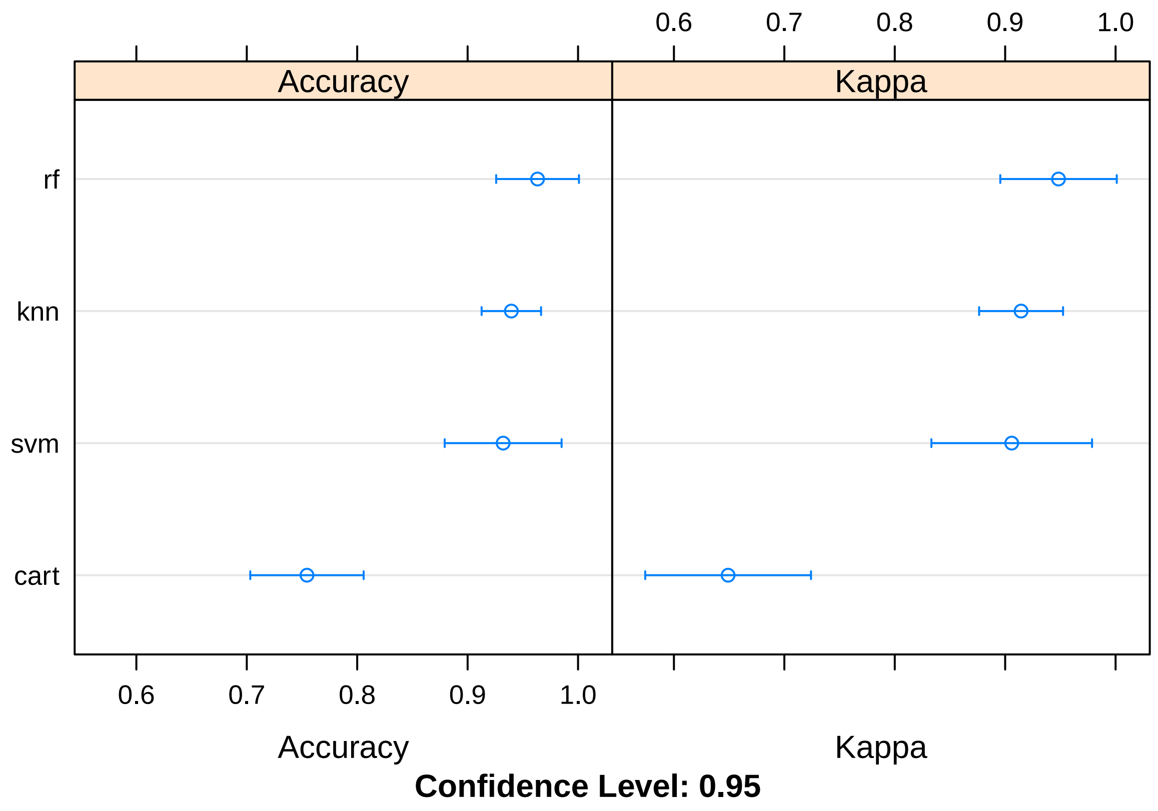


# Supplemental Figure 3

```
set.seed(2)
test_index<-createDataPartition(conditions$Source, p=0.80, list = FALSE)
Dtraining <- Data[test_index, ]
Dtesting<- Data[-test_index,]
Conditrain<-conditions[test_index, ]
Conditesting<-conditions[-test_index,]

percentage <- prop.table(table(Conditrain$Source)) * 100
percentage2 <- prop.table(table(Conditesting$Source))*100
# 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$Source, method="lda",
#metric=metric, trControl=control)
# b) nonlinear algorithms
# CART
fit.cart <- train(Dtraining, Conditrain$Source, method="rpart", metric=metric,
                  trControl=control)
# kNN
fit.knn <- train(Dtraining, Conditrain$Source, method="knn", metric=metric,
                 trControl=control)
# c) advanced algorithms
# SVM
fit.svm <- train(Dtraining, Conditrain$Source, method="svmRadial", metric=metric,
                 trControl=control)
# Random Forest
fit.rf <- train(Dtraining, Conditrain$Source, method="rf", metric=metric,
                trControl=control)

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

Supplemental Figure 3a

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

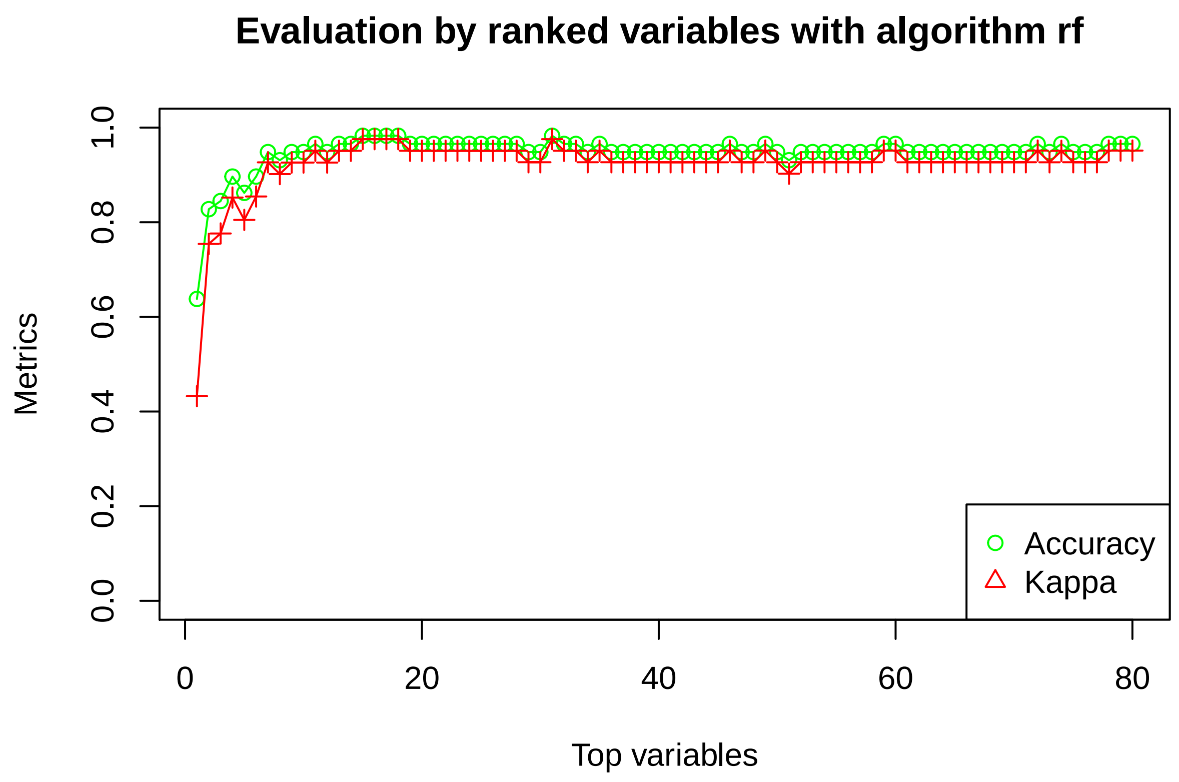


```
#ALGORITMO SELECCIONADO
#model=fit.knn
#kalgoritmo="knn"
#model=fit.svm
#kalgoritmo="svmRadial"
model=fit.rf
```

```r
kalgoritmo="rf"

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

#EVALUACION DE DIF SUBSET DE variables TOP
kvalue=80

kEvaluacion<-matrix(,nrow=kvalue, ncol=52)
colnames(kEvaluacion)<-c("Accuracy","Kappa","AccuracyLower","AccuracyUpper",
                         "AccuracyNull", "AccuracyPValue","McnemarPValue",
                         "Sensitivity_B", "Specificity_B", "Pos Pred Value_B",
                         "Neg Pred Value_B", "Precision_B", "Recall_B","F1_B",
                         "Prevalence" ,"Detection Rate_B","Detection Prevalence_B",
                         "Balanced Accuracy_B", "Sensitivity_C", "Specificity_C",
                         "Pos Pred Value_C", "Neg Pred Value_C", "Precision_C",
                         "Recall_C","F1_C","Prevalence" ,"Detection Rate_C",
                         "Detection Prevalence_C","Balanced Accuracy_C",
                         "Sensitivity_E", "Specificity_E", "Pos Pred Value_E",
                         "Neg Pred Value_E", "Precision_E", "Recall_E","F1_E",
                         "Prevalence" ,"Detection Rate_E","Detection Prevalence_E",
                         "Balanced Accuracy_E", "Sensitivity_N", "Specificity_N",
                         "Pos Pred Value_N", "Neg Pred Value_N", "Precision_N",
                         "Recall_N","F1_N","Prevalence" ,"Detection Rate_N",
                         "Detection Prevalence_N","Balanced Accuracy_N", "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$Source, method=kalgoritmo, metric=metric,
                    trControl=control)
predictionsK <- predict(fit.algorK, DtestK)
StatisticsK<-confusionMatrix(predictionsK, Conditesting$Source)
kEvaluacion[1,1:51]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass[1,],
                                       StatisticsK$byClass[4,])))

for (k in 2:kvalue){
  DtrainK<-DtrainingRanked[,c(1:k)]
  DtestK<-DtestingRanked[,c(1:k)]
  fit.algorK <- train(DtrainK, Conditrain$Source, method=kalgoritmo, metric=metric,
                      trControl=control)
  predictionsK <- predict(fit.algorK, DtestK)
  StatisticsK<-confusionMatrix(predictionsK, Conditesting$Source)
  kEvaluacion[k,1:51]<-t(as.data.frame(c(StatisticsK$overall,StatisticsK$byClass[1,],
                           StatisticsK$byClass[2,],StatisticsK$byClass[3,],
                           StatisticsK$byClass[4,])))

}

EvalK<-as.data.frame(kEvaluacion)
```

Supplemental Figure 3b

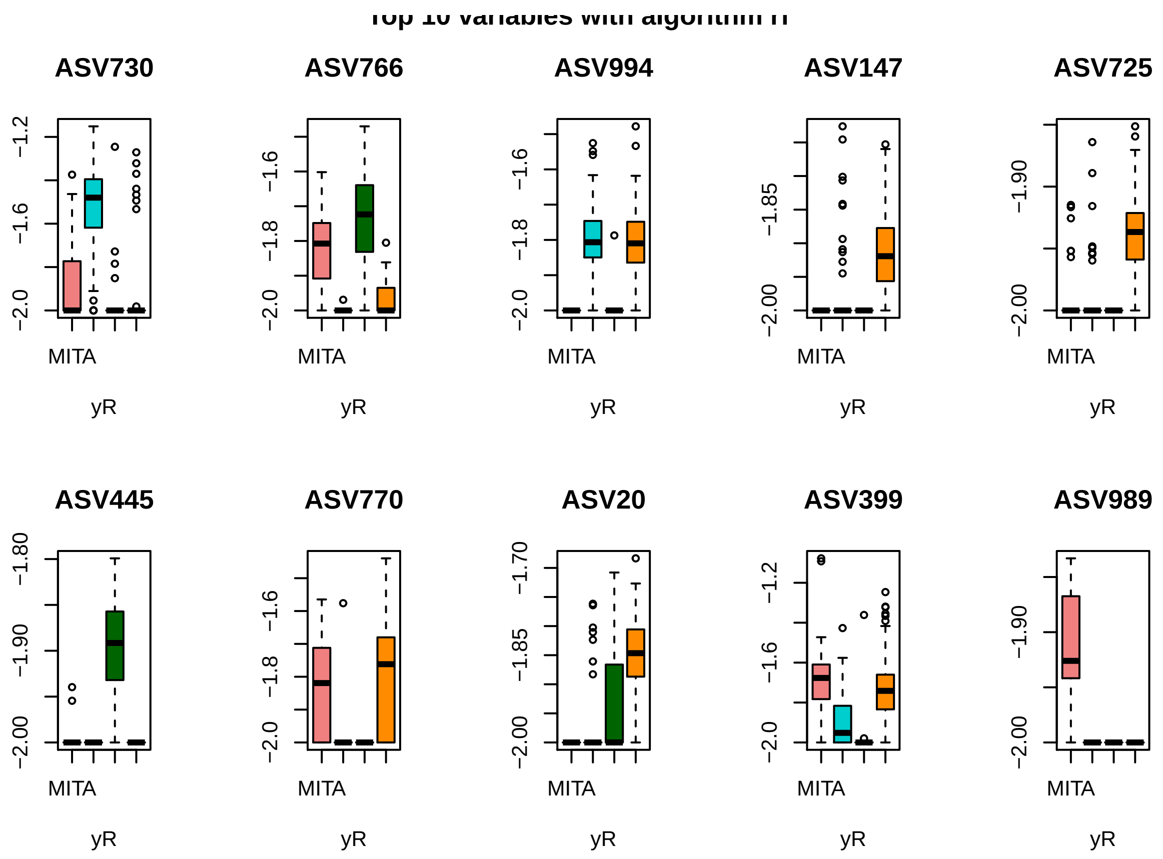
```
par(mfrow = c(1,1))
plot(EvalK$Accuracy,type="o",pch=1,col="green",
     main = paste("Evaluation by ranked variables with algorithm",kalgoritmo,sed=""),
     xlab = "Top variables",ylab = "Metrics", ylim=c(0,1))
lines(EvalK$Kappa,type = "o", pch=3, col="red")
legend("bottomright",legend=c("Accuracy","Kappa"),pch=c(1,2,3),col=c("green","red"))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



## Boxplots of top bacteria

```
ks=10
DtrainKS<-DtrainingRanked[,c(1:ks)]
DtestKS<-DtestingRanked[,c(1:ks)]
```

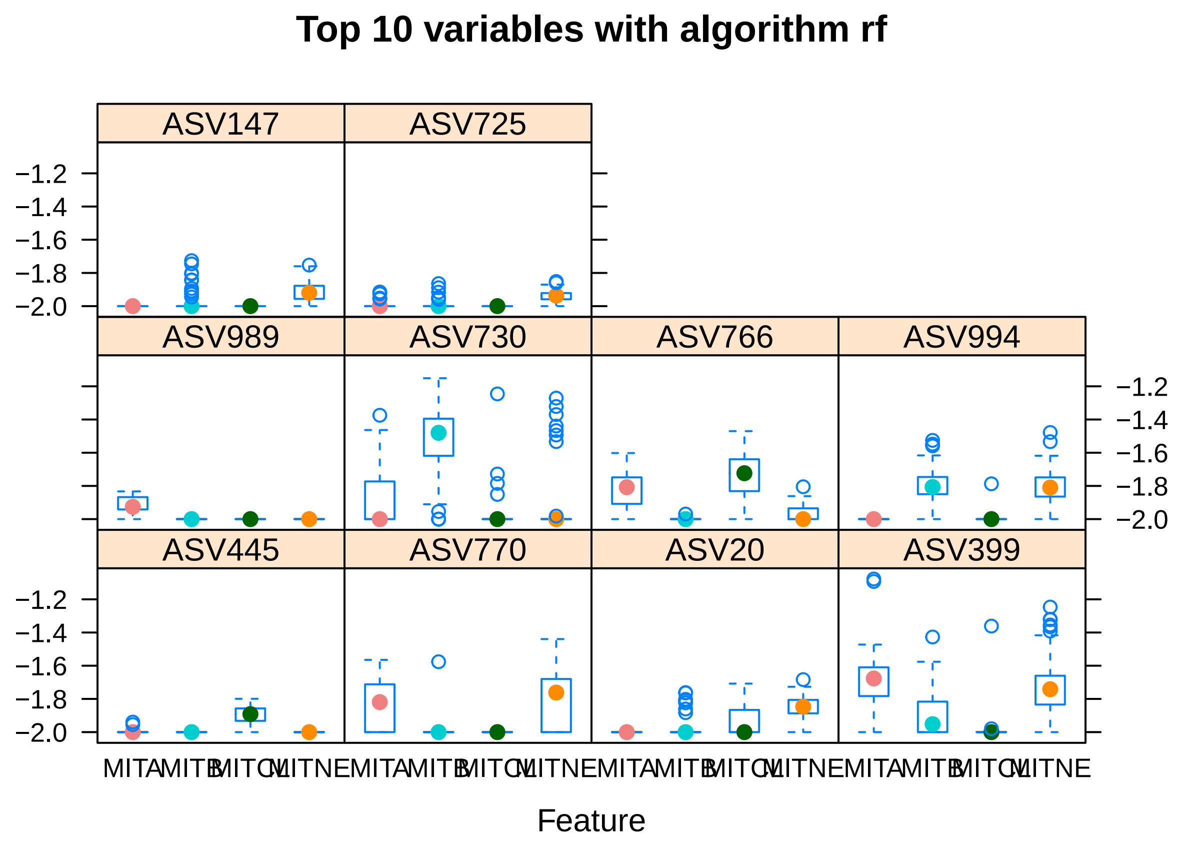
```r
DataRankedtopK<-DataRanked[,c(1:ks)]

set.seed(3)
fit.algorKS <- train(DtrainKS, Conditrain$Source, 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$Source)

xR <- DtrainKS
yR <- Conditrain$Source
sxR <- DtrainingRanked[,c(6,7,8,9,10,1,2,3,4,5)]

#boxplot
par(mfrow=c(2,5))

for(i in 1:10) {
  boxplot(xR[,i]~yR, main=names(xR)[i],
          col=c("lightcoral","cyan3","darkgreen","darkorange"),ylab = " ")
}
title(paste("Top 10 variables with algorithm", kalgoritmo), outer=TRUE)
```



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

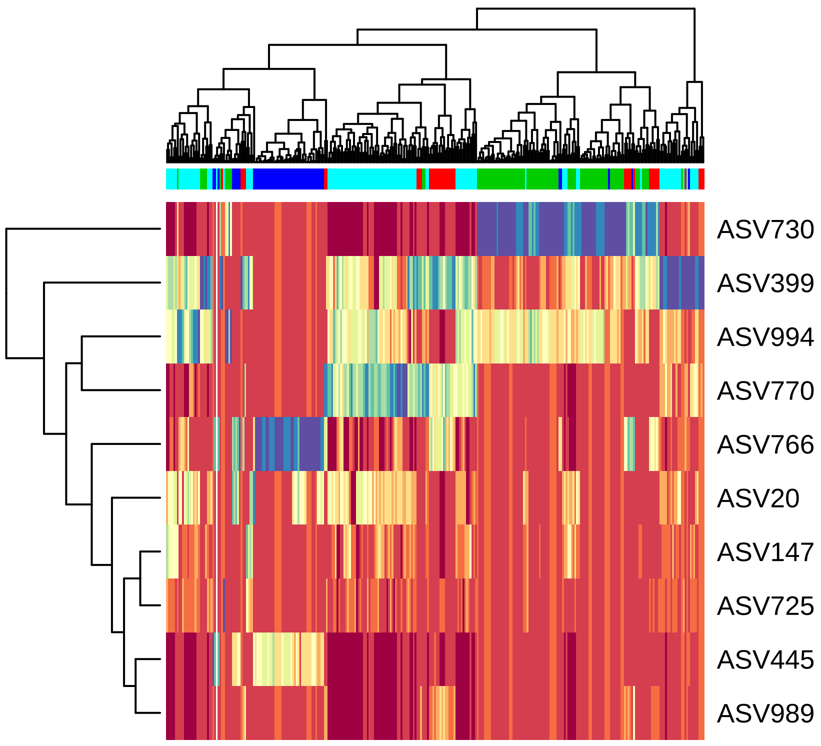
## Supplemental Figure 3c

```
# box and whisker plots for each attribute
featurePlot(x=sxR, y=yR, plot="box",
            main = paste("Top 10 variables with algorithm", kalgoritmo),
            col=c("lightcoral","cyan3","darkgreen","darkorange"))
```



# Supplemental Figure 3d

```
heatmap(as.matrix(t(DataRanked[,1:ks])), scale="column", col = brewer.pal(11,"Spectral"),
        ColSideColors=paste(as.numeric(conditions$Source)+1),,labCol = FALSE)
```


ASV730
ASV399
ASV994
ASV770
ASV766
ASV20
ASV147
ASV725
ASV445
ASV989