Host trait prediction from high-resolution microbial features

Giovanni Bacci1\*  
1Department of Biology, University of Florence, Via Madonna del Piano 6, Sesto Fiorentino, FI, I-50019, Italy  
\*Corresponding author: [giovanni.bacci@unifi.it](mailto:giovanni.bacci@unifi.it)

## Running head

Predictions from metagenomes

## Summary (500 characters)

Predicting host traits from metagenomes presents new challanges that can be difficult to overcome for researchers without a strong background in bioinformatics and/or statistics. Profiling bacterial communities usign shotgun metagenomics often leads to the generation of a large amount of data that cannot be used directly for training a model. In this chapter we provide a detailed description of how to build a working machine learning model based on taxonomic and functional features of bacterial communities inhabiting the lungs of cystic fibrosis patients. Models are build in the R environment by using different freely available machine learning algortihms.

## Keywords

machine learning, next generation sequencing, metagenomics, host trait prediciton, community profiling, taxonomic profiling, functional profiling

## 1. Introduction

The direct extraction and sequencing of genetic material from bacterial cells inhabiting a given environment—an approach called shotgun Metagenomics—has drastically increased our knowledge of the microbial world. In the last decade the sequencing cost for a magabase of DNA has dropped while the output of sequencing machines has rapidly increased. The advent of third-generation sequencing technologies (also known as long-read sequencing) enabled the production of long DNA sequences from single DNA molecules increasing the resolution power of ’omics techniques including Metagenomics; however all these technical advancements requires the development of specific analyses suitable for different applications.

Since the first study made by Antonie van Leeuwenhoek in 1680s where he compared microorganisms from fecal and oral samples of healthy and ill individuals [1] microbiologists have characterized thousands of different microbial strains in, almost, all districts of human body [2–4].

In more recent times, the interaction between microbes and plants has been intensively explored shading light on new possible methods of cultivation and defining groups of microorganisms associated with plant health [68]. In the last two decades, the widespread application of genetic and genomic approaches has revealed a bacterial world astonishing in its diversity and complexity. These new technologies have provided researchers with microbial data from the most different organisms (from the human body to the ant gut) altering our understanding of macroorganisms biology. In 2001 Joshua Lederberg coined the term microbiome referring to “the ecological community of commensal, symbiotic, and pathogenic microorganisms that literally share our body space” [11]. As time passes, the term “microbiome” has been referred to other types of macroorgansims such arthropods, fish, and plants. Many studies have been performed in animals and plants, reporting the description of the bacterial communities (microbiome) found in several districts as gut, roots, skin, and leafs [69, 70, 71, 72, 73]. Currently, many scientific articles distinguish “microbiome” and “microbiota” to describe either the collective genomes of the microorganisms that reside in an environmental niche or the microorganisms themselves. However, by the original definitions, these terms are largely synonymous. Bacteria took on a new role in animal nutrition, serving not only as prey but also as producers of digestible molecules in the animal gut. The evolution of gut itself was certainly influenced by bacteria. The advent of the coelom made gut elongation and regional specialization possible, facilitating both massive ingestion and storage for later digestion. Although the degree to which microbes have driven gut evolution is unknown, the radiation of several animal groups was undoubtedly enabled by alliances with their gut-associated microbiota (e.g. ruminants). The animal evolution has also influenced the distribution and diversification of bacteria promoting the proliferation of bacterial species exclusively in particular animals [85]. However, such specialization, comes with a cost: for every animal species that goes extinct, an unknown number of unique bacterial lineages that have evolved to depend on this animal niche disappear as well [86]. On a wider scale, the evolution of animals provided new environments for bacterial colonization (e.g. deep sediments resulting from animal burrowing). Finally, human activities, which produce a range of xenobiotic molecules, have driven selection on bacterial catabolic pathways, leaving a signature of our presence in microbial metabolism [87].

## 2. Materials

A working installation of R [5] is required for this tutorial to work, along with a set of libraries mainly used for building and validating the model. The workflow here proposed uses data coming from a metagenomic study on cystic fibrosis lung communities along time [6]. A complete description of datasets, hardware and software requirements is given below.

### 2.1 Data files

In this tutorial bacterial features obtained from shotgun metagenomics sequencing are used to build a machine learning model and to make infereces on bacterial community structure. The data consist of three main tables reporting quantitative and qualitative information about taxa detected in the lung of subjects included in the study, genes harbored by those taxa, and clinical characteristics of the subjects. Data can be downloaded from <https://github.com/GiBacci/predicting_from_metagenomes/tree/master/data>. Tables are available in the RDS format and can be easily imported into R using the function readRDS(). Since RDS is the native data file format for R, tables can be directly loaded into the R environment without worrying about additional parameters such as field separator, decimal separator, charcater encoding format, and so on. A description of the data files available is given below:

1. taxa\_ab.rds: taxa abundances in all subjects included in the study. Each row of the table is a different observation whereas each column represent a different taxa detected. The proportion of taxa is reported as relative abundance so that the sum of all taxa abundances in each observation is one.
2. gene\_counts.rds: counts of metagenomic reads mapping to bacterial genes recovered from lung communities. The same standard used for taxa abundance was used here with each reporting a different observation and each olumn reporting a different gene. In metagenomic studies genes are usually more than observations and they can be reported into rows instead of columns so to minimize the number of variables and reduce memory requirement.
3. sample\_meta.rds: characteristics of patients included in the study. The table is the table S1 of the paper reported above [6] and a complete description of columns is available in the work. In this chapter we will focus on the genotype of the patients aiming to build a working machine learnin algorithm able to predict genotype from bacterial features of lung communities.
4. gene\_meta.rds: characteristics of genes included in the gene count table reported in 2. This table is a slight modifyed version of the output produced by eggNOG mapper [7].

In principle any kind of metagenomics/transcriptomics study follow the scheme here proposed. The table reported in 1 and/or 2 could be replaced by the expression levels of the genes found in a transcriptomic study or by counts of reads coming from a metabarcoding based on 16S rRNA. Feel free to replace the tables reported above with any kind of data that fit the general scheme provided.

### 2.2 Software requirements

Models are generated using a free software environment for statistical computing called R. R is part of many Linux distributions but it can be freely downloaded and installed from <https://cran.r-project.org/> by choosing the appropriate operation system in the “Download and Install R” window. Additional packages needed are listed below (the version of each package used in this tutorial is reported between brackets):

1. compositions [8] (version 1.40.3): a collection of functions for analysing compositional data (quantitative data, strictly positive, which sum to a constant value).
2. vegan [9] (version 2.5.6): a package developed for studying multivariate data produced during ecological studies. It contains several functions for dimensionality reduction (such as correspondence analysis, non-metric multidimensional scaling, and others) and for diversity analysis (either alpha or beta diversity).
3. DESeq2 [10] (version 1.26.0): a suite for the analysis of count data from many biological assays. The package was developed for analysing RNA-seq data but it is also used for amplicon sequence data (such as 16S rRNA metabarcoding) or ChIP-Seq. It implements a transformation function (called variance stabilizing transformation or VST) useful to prepare count data for many machine learning approaches.
4. caret [11] (version 6.0.85): a collection of functions for training and validation of multiple machine learning algorithms. It contains methods for fine tuning classification and regression alorithms using a unified syntax. It also evaluates the performance of models produced using standard metrics such as root mean squared error, receiver operating characteristic curve, accuracy and others.
5. randomForest [12] (version 4.6.14): implementaion of the algorithm described by Breiman in 2001 [13]. This package is used in combination with caret to build the final model.
6. kernlab [14] (version 0.9.29): implementation of several kernel-based machine learning methods [15]. In this chapter we will use the radial basis function kernel in combination with caret.
7. gbm [16] (version 2.1.5): implementation of gradient boosting machines. A stochastic gradient boosting approach will be used within caret.
8. pROC [17] (version 1.16.1): tool for visualizing receiver operating characteristic (ROC) curves. It contains also function for comparing curves from different models. Curves are computed in sensitivity and specificity space defined as the probability to assign a true positive or a false negative given the model.
9. ggplot2 [18] (version 3.2.1): package for creating different types of graphics based on the book “The grammar of Graphics” by Leland Wilkinson [19].
10. ggbeeswarm [20] (version 0.6.0): this package provides ggplot2 geoms for plotting categorical data minimizing overlaps.
11. multcompView [21] (version 0.1.8): package that converts vectors of p-values into a letter-based visualization useful for multiple comparisons across categories.

## 3. Methods

A working installation of R can run all lines of code reported in this chapter. However, I would suggest to write codes into a script (a simple text file) by using an integrated development environment (IDE) like RStudio <https://rstudio.com/>.

### 3.1 Importing data

Before starting to build our models data must be imported into R. Several functions can do this depending on the input file format. The data suggested in this chapter were saved using a native R format called RDS and can be imported using the readRDS function. In case of text data the function read.table can be used as well as one of its sister functions (to see the help of read.table simply run: ?read.table).

# importing gene counts  
genes <- readRDS("./data/gene\_counts.rds")  
  
# importing proportion of taxa  
taxa <- readRDS("./data/taxa\_ab.rds")  
  
# sample data  
meta <- readRDS("./data/sample\_meta.rds")

### 3.2 Data transformation

As described in section 2.1 of this chapter, our data consist of three tables (two of which were imported into R as “matrix” that is still a type of tabular format) reporting different features of a specific bacterial community: the lung of patients affected by cystic fibrosis. Data reported in these tables, however, differ from a statistical/mathematical point of view. The ‘taxa’ table for example reports abundances of bacterial organisms expressed as relative abundances or, in other words, as fractions. On the other hand, the ‘gene’ table contains counts of reads mapped to specific genes and the ‘meta’ table contains information about each patient includede in the original study.

Quantitative data must be handled in the proper way to generate reliable results. Relative abundace data (also called “compositional data”) are constrained so that their sum must be a constant (e.g. one in the case of relative abundaces and 100 in the case of percentages). The sample space for this kind of data is different from the real space associated with unconstrained data [22]. One of the main bias that affects biological analyses on compositional data is that they are negatively correlated. This influences standard statistical multivariate analyses where the abundance of different taxa may be positively or negatively correlated [23]. In addition, machine learning algorithms may benefit from reducing the number of correlated predictors as well as reducing the number of type I errors [24, 25]. Counts of reads obtained through high-throughput sequencing machines, on the other hand, are usually overdispersed (they have a variance much larger than the mean) and contain a high proportion of zero counts (sparsity). Besides, the total number of sequences obtained for a given sample does not match the real number of microbes present in that sample. Since the number of reads produced by modern sequencing machines is contrained by the capacity of the instrument, counts of microbial features can be considered compositional and thus constrained to sum to a given constant [26]. All these problems are still debated in the scientific community expecially regarding the study of bacterial community.

Taxa abundances will be transformed using centered log ratio transformation (CLR) whereas gene counts will be transformed using the so called variance stabilizing transformation (VST) provided in the DESeq2 package. Both transformations may help subsequent multivariate analyses such as principal component analysis (PCA) reducing the number of negative correlation (CLR) and the variance (VST). Both datasets will be transformed into orthogonal predictors using PCA to reduce type I errors (as explained above) prior to feed machine learning models. Only principal components with an eigenvalue higher than one will be retained. We will feed our model also with an alpha diversity index (namely the inverse Simpson index) to test the importance of bacterial diversity in predicting patient genotype.

# Centered log ratio transformation  
taxa.clr <- clr(taxa)  
class(taxa.clr) <- "matrix"  
  
# Variance stabilizing transformation  
genes.vst <- DESeqDataSetFromMatrix(t(genes), colData = meta, design = ~ 1)  
genes.vst <- estimateSizeFactors(genes.vst)  
genes.vst <- vst(genes.vst, blind = TRUE, fitType = "local")  
genes.vst <- t(assay(genes.vst))  
  
# Alpha diversity estimation  
alpha <- diversity(taxa, index = "invsimpson")  
alpha <- data.frame(A=(alpha - mean(alpha))/sd(alpha))  
  
# Principal component analysis  
taxa.pca <- prcomp(taxa.clr, center = T, scale. = T)  
genes.pca <- prcomp(genes.vst, center = T, scale. = T)  
  
# Retaining only PC with an eigenvalue higher than one  
taxa.x <- taxa.pca$x[,taxa.pca$sdev^2 >= 1]  
genes.x <- genes.pca$x[,genes.pca$sdev^2 >= 1]  
  
# Assembling the full dataset  
colnames(taxa.x) <- gsub("PC", "T", colnames(taxa.x))  
colnames(genes.x) <- gsub("PC", "F", colnames(genes.x))  
  
y <- meta$genotype  
full.data <- cbind(y = y, taxa.x, genes.x, alpha)  
full.data <- droplevels(full.data[full.data$y != "other",])

### 3.2 Training models

In this section, we are going to train three different machine learning models using caret and the respective machine learning packages defined in section 2.2. The function set.seed ensures reproducibility during random operations such as splitting our dataset into validation and trainin set. Data will be partitioned into a training set, used for training models, and a validation set, used for testing them. Machine learning algorithms need to be tuned in order to function correctly. The number and the type of parameters varies in each algorithm so it is reccommended to chose the best value programmatically. The so called cross-validation process attempts to da that by validating different parameters on a subset of the training set previously defined. The caret package has several different algorithms for cross-validation but in this chapter we will use the adaptive cross-validation algorithm due to its flexibility. In particular we will use a 10-fold cross-validation (number = 10) repeated 3 times (repeats = 3) with no grid specified (search = "random"), with adaptive parameters set to:

1. 5 minimum resamples used for each tuning parameter (min = 5)
2. A confidence level of 0.05 for removing parameter settings (alpha = 0.05)
3. A generalized linear model for tuning (method = "gls")
4. The algorithm stops if it finds an optimal solution before the end of resampling (complete = FALSE)

We will train three models: a random forest model (method = "rf"), a generalized boosted model (method = "gbm"), and a supporting vector machine model (method = "svmRadial"). The receiver operating characteristic metrics will be used to evaluate models after the training so the metric “ROC” is selected and predictions made during cross-validation are stored in the model object (metric = "ROC" in training function and savePredictions = TRUE in the train control function).

set.seed(1239)  
train <- createDataPartition(full.data$y, times = 1,   
 list = FALSE, p = .8)  
  
model.train <- full.data[train,]  
model.test <- full.data[-train,]  
  
cntr <- trainControl(method = "adaptive\_cv",  
 number = 10, repeats = 3,  
 search = "random",  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary,  
 sampling = "up",  
 savePredictions = TRUE,  
 adaptive = list(min = 5, alpha = 0.05,   
 method = "gls",   
 complete = FALSE))  
# TODO  
# REMOVE IF WHEN FINISHED  
out <- "model.list.rds"  
if(!file.exists(out)){  
 set.seed(12345)  
 rf <- train(y ~ ., data = model.train,   
 method = "rf",  
 trControl = cntr, tuneLength = 15,  
 verbose = FALSE, scale = FALSE,  
 metric = "ROC")  
   
 set.seed(12345)  
 gbm <- train(y ~ ., data = model.train,   
 method = "gbm",  
 trControl = cntr, tuneLength = 15,  
 verbose = FALSE,  
 metric = "ROC")  
   
 set.seed(12345)  
 svm <- train(y ~ ., data = model.train,   
 method = "svmRadial",  
 trControl = cntr, tuneLength = 15,  
 verbose = FALSE, scale = FALSE,  
 metric = "ROC")  
}

### 3.3 Model evaluation

Since we built classification models based on two classes (also known as binary classifiers) we will evaluate their performances using receiver operating characteristic curves (ROC curves). To do that, we could use the *ad-hoc* function reported below called rocCurve that will extract the best model based on tuned parameters and will produce two ROC curves: one using the same cross validation data (also known as out-of-bag estimates), and the other using the test data. A ROC curve is a graphical representation of the performances of a binary classifier at different threshold settings. Two parameters are displyed in a ROC curve: sensitivity (normally displayed on the y-axis) and specificity (normally displayed on the x-axis). Sensitivity, also called true positive rate, is the proportion of positive observations correctly classified by the model (in this case the proportion of homozygous patients for the F508 mutation correctly classified as such). Specificity, also called true negative ratio, is the proportion of negative observations correctly classified by the model (in this case the proportion of heterozygous patients for the F508 mutation correctly classified as such). The x-axis of ROC curves is usually inverted unlsess specificity is replaced by also called the false positive rate (namely the number of negative observations wrongly classified as positives).

# TODO  
# REMOVE IF WHEN FINISHED  
if(file.exists(out)){  
 model.list <- readRDS(out)  
}else{  
 model.list <- list(RF = rf,  
 GBM = gbm,  
 SVM = svm)  
 saveRDS(model.list, "model.list.rds")  
}  
  
rocCurve <- function(model, data){  
 best <- model$bestTune  
 pred <- model$pred  
 d <- data.frame(pred[,names(best)])  
 extr <- apply(d, 1, function(x) all(x == best))  
 pred <- pred[extr,]  
   
 p <- predict(model, data, type = "prob")  
 roc\_obj <- roc(data$y, p$heterozygote\_F508,   
 levels = levels(data$y))  
 plot.roc(pred$obs, pred$heterozygote\_F508)  
 plot.roc(roc\_obj)  
}

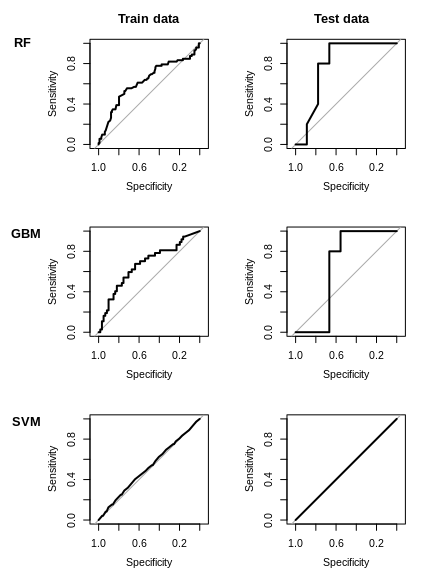


Figure 1: Receiver operating characteristic curves based on training and test data. The algorithm used to build the models is reported at the top left of each plot (RF, random forest; GBM, generalized boosted model; SVM, supporting vector machine).

From a first look at the curves reported in Figure 1, random forest and generalized boosted model outperformed supporting vector machines in terms of both sensitivity and specificity on train and test data. To confirm this hypotesis we can calculate the ROC confidence interval at 95% for each model using the function ci of the pROC package. The lower, median, and upper values of ROC were reported in Table 1 for all models. Based on results reported, the random tree model is the one with the best performances for classifyng the genotype of CF patients based on microbial characteristics.

get\_roc <- function(model, data){  
 roc\_obj <- roc(data$y,   
 predict(model, data, type = "prob")[,1],  
 levels = levels(model.test$y))  
 setNames(as.vector(ci(roc\_obj)),   
 c("lower", "ROC", "upper"))  
}  
roc <- t(sapply(model.list, get\_roc, model.test))

Table 1: Confidence interval of ROC curves reported in Figure 1. Models were reported in the first column whereas the lower, median, and upper values of ROC were reported in the second, third, and fourth column respectively.

|  |  |  |  |
| --- | --- | --- | --- |
|  | lower | ROC | upper |
| RF | 0.533 | 0.789 | 1.000 |
| GBM | 0.323 | 0.644 | 0.966 |
| SVM | 0.500 | 0.500 | 0.500 |

## 3.4 Selecting bacterial features

The importance of variables included in the model can be computed in different ways. The caret package provides a series of metrics that can be used to evaluate the importance of predictors in both regression and classification models. However, in this chapter we are going to use a model-related metric called the “Gini coefficient” provided by the randomForest package (varImp function of caret package with useModel = T). The Gini coefficient is a measure of nodes’ “impurity” (or heterogeneity) that ranges from 0 (homogeneous) to 1 (heterogeneous). The mean decrease in Gini coefficient is thus a measure of a variable importance to the classification performances of the model: variables with higher purity cause higher decrease in Gini coefficient. We can extract variable importance from our model to graphically represent the distribution of variables according to their type (T for taxa, F for functional, and A for alpha-diversity). To avoid overlaps we will use the r package ggbeeswarm version 0.6.0. Figure 2 shows the distribution of variable importances per type: 6 gene-related and 3 taxonomic-related predictors have an importance higher than the 90% of predictors whereas alpha diversity has a low impact in model performances.

# Get variable importance  
v.imp <- varImp(model.list$RF, useModel = T,  
 scale = F, type = 2)$importance  
# Build a data frame for plotting  
v.imp <- data.frame(vars = rownames(v.imp),  
 imp = v.imp[[1]],  
 row.names = NULL)  
# Get predictor type (the first character)  
v.imp$var.type = substr(v.imp$vars, 1, 1)  
  
# Top 10% variables (90th percentile)  
top10 <- quantile(v.imp$imp, .9)  
top.vars <- as.character(v.imp[v.imp$imp >= top10,"vars"])  
  
# Plotting  
ggplot(v.imp, aes(x = var.type, y = imp)) +  
 geom\_quasirandom(shape = 1) +  
 geom\_hline(yintercept = top10,  
 linetype = 2,  
 color = "red") +  
 theme\_bw(base\_size = 10, base\_family = "Helvetica",  
 base\_line\_size = 0.25) +  
 xlab("Predictor type") +  
 ylab("Mean decrease in Gini coefficient")

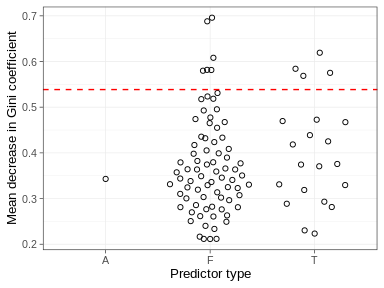


Figure 2: Importance of predictors (variables) used in the model. The importance of each predictor was reported according the type of predictor: T for taxa, F for functional, and A for alpha-diversity. The red dashed line represent the 90th percentile of the importance distribution.

Since predictors we used to train our models are orthogonal representiations of the original microbial features, we need to extract the contribution of each original feature to the most important predictors defined above (Figure 2). To get variable contributions we need to calculate their coordinates on principal components (PCs). To do this, we multiply each PC by its standard deviation; then we use resulting coordinates to obtain the representation quality of a variable on a given PC called squared cosine [27]. The relative contribution of a variable can finally be obtained by dividing the squared cosine by its sum over each PC. All these steps are included in a single function called varContrib. To finally inspect the contribution of taxa and genes on selected PCs we use the R base function rowMeans after selecting top components only (all implemented in the *ad-hoc* funciton formatDataFromContrib).

varContrib <- function(data.pca){  
 # Get coordinates for variables (loadings X standard deviation)  
 var.coord <- t(t(data.pca$rotation) \* data.pca$sdev)  
 # Get quality of representation on the factor map (coordinates^2)  
 var.cos2 <- var.coord^2  
 # Get variable contributions to the PCs (relative importance)  
 t(t((var.cos2\*100)) / colSums(var.cos2))  
}  
  
# Creates a dataset of variable contributions from a prcomp  
# object and a set of selected variables.  
formatDataFromContrib <- function(pca, PC.name, top.vars){  
 # getting variable contributions  
 contrib <- varContrib(pca)  
 # change names according to PC.name  
 colnames(contrib) <- gsub("PC", PC.name, colnames(contrib))  
 # building dataset  
 contrib <- rowMeans(contrib[, grep(PC.name, top.vars, value = T)])  
 contrib <- sort(contrib, decreasing = T)  
 lvls <- names(contrib)  
 data.frame(vars = factor(lvls, levels = lvls), contrib = contrib,  
 row.names = NULL)  
}  
  
# get gene contribution  
genes.contrib <- formatDataFromContrib(genes.pca, "F", top.vars)  
names(genes.contrib) <- c("best.og", "contrib")  
  
taxa.contrib <- formatDataFromContrib(taxa.pca, "T", top.vars)  
names(taxa.contrib) <- c("taxa", "contrib")

To inspect how different functional categories contribute to selected PCs we conduct a one-way analysis of variance (ANOVA) using the aov function. First, we need to merge gene informations to gene contribution using the merge function and then we build our model. As shown in Table 2 different COG categories affect the actual contribution of selected PCs with a p-value lower than 0.01.

meta.genes <- readRDS("data/gene\_meta.rds")  
meta.genes <- merge(meta.genes, genes.contrib)  
  
genes.aov <- aov(contrib ~ COG\_cat, data = meta.genes)  
s <- summary(genes.aov)

Table 2: Analysis of variance results showing the effect of different COG categories on gene contributions. Df, degrees of freedom; Sum Sq, sum of squares (deviance); Mean Sq, mean square (variance); F value, value of F statistic; Pr(>F), p-value associated to the F statistic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| COG\_cat | 20 | 0.0008765 | 4.38e-05 | 9.922579 | 0 |
| Residuals | 51641 | 0.2280746 | 4.40e-06 | NA | NA |

At this stage we may be interested in which categories have an higher contribution on our model and we can inspect that using the Tukey post-hoc test implemented in the TukeyHSD function. The p-values of all the pairways comparisions returnde by the function are converted into a character-based encoding using the multcompLetters function of the package multcompView. Results are then plotted with ggplot2 (Figure 3).

tuk <- TukeyHSD(genes.aov)  
labs <- multcompLetters(tuk$COG\_cat[,4])[['Letters']]  
labs <- data.frame(COG\_cat = names(labs),  
 labs = labs)  
genes.mean <- with(meta.genes, do.call(rbind, by(contrib, COG\_cat, mean\_cl\_boot)))  
genes.mean <- data.frame(COG\_cat = rownames(genes.mean),  
 genes.mean)  
genes.mean$COG\_cat <- reorder(genes.mean$COG\_cat, genes.mean$y)  
genes.mean <- merge(genes.mean, labs)  
  
ggplot(genes.mean, aes(x = COG\_cat, y = y, ymin = ymin, ymax = ymax)) +  
 geom\_errorbar(width = .3) +  
 geom\_col() +  
 geom\_text(aes(label = labs, y = ymax), vjust = -1,  
 size = 3) +  
 scale\_y\_continuous(expand = expand\_scale(add = c(0,0.0003))) +  
 theme\_bw(base\_size = 10, base\_family = "Helvetica",  
 base\_line\_size = 0.25) +  
 ylab("Contribution on selected components (%)") +  
 xlab("COG categories")

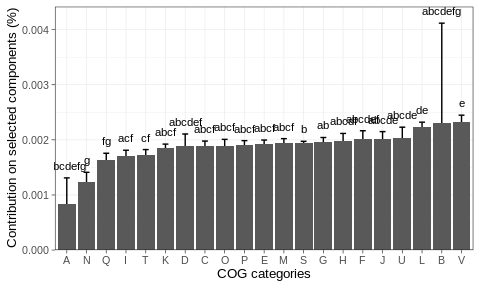


Figure 3: Tukey post-hoc test results. Different COG categories are reported following standard notation. Bars represent the mean contribution for each category whereas error bars represent the upper confidence limit for the observations.

A similar approach can be applyed to taxonomic data by extracting the genus affiliation from species names. This time we will inspect if different genera differentially impact the contribution on selected PCs using the same approach that we used above bat with different data. This time differences are not significant (p-value higher than the significance level of 0.05, Table 3) so we will avoid doing any post-hoc test but we will plot the average contribution of each bacterial species to the selected PCs (Figure 4).

genus <- sapply(strsplit(as.character(taxa.contrib$taxa), "\_"), "[", 1)  
genus <- reorder(factor(genus), taxa.contrib$contrib)  
taxa.contrib$genus <- genus  
  
taxa.aov <- aov(contrib ~ genus, data = taxa.contrib)  
s <- summary(taxa.aov)

Table 3: Analysis of variance results showing the effect of different bacterial genera on taxa contributions. Df, degrees of freedom; Sum Sq, sum of squares (deviance); Mean Sq, mean square (variance); F value, value of F statistic; Pr(>F), p-value associated to the F statistic.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| genus | 29 | 40.61622 | 1.400559 | 1.146008 | 0.3523265 |
| Residuals | 32 | 39.10783 | 1.222120 | NA | NA |

genus <- sapply(strsplit(as.character(taxa.contrib$taxa), "\_"), "[", 1)  
genus <- reorder(factor(genus), taxa.contrib$contrib)  
taxa.contrib$genus <- genus  
  
taxa.aov <- aov(contrib ~ genus, data = taxa.contrib)  
summary(taxa.aov)

## Df Sum Sq Mean Sq F value Pr(>F)  
## genus 29 40.62 1.401 1.146 0.352  
## Residuals 32 39.11 1.222

ggplot(taxa.contrib, aes(x = taxa, y = contrib)) +  
 geom\_col() +  
 theme\_bw(base\_size = 10, base\_family = "Helvetica",  
 base\_line\_size = 0.25) +  
 scale\_y\_continuous(expand = expand\_scale(add = c(0,0.1))) +  
 theme(axis.text.x = element\_text(angle=90, hjust = 1, vjust = 0.5)) +  
 ylab("Contribution on selected components (%)") +  
 xlab("Bacterial species")

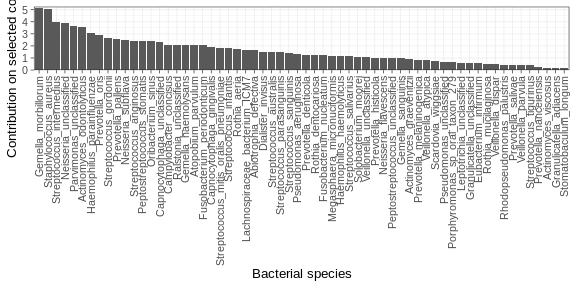


Figure 4: Average contribution of bacterial species on the selected variables.

## References

1. Van Leeuwenhoek A An abstract of a letter from antonie van leeuwenhoek, sep. 12, 1683. About animals in the scrurf of the teeth. Philos Trans R Soc Lond 14:568–574

2. Turnbaugh PJ, Ley RE, Hamady M, Fraser-Liggett CM, Knight R, Gordon JI (2007) The human microbiome project. Nature 449:804–810. doi:[10.1038/nature06244](https://doi.org/10.1038/nature06244)

3. Group TNHW, Peterson J, Garges S, Giovanni M, McInnes P, Wang L, Schloss JA, Bonazzi V, McEwen JE, Wetterstrand KA, Deal C, Baker CC, Di Francesco V, Howcroft TK, Karp RW, Lunsford RD, Wellington CR, Belachew T, Wright M, Giblin C, David H, Mills M, Salomon R, Mullins C, Akolkar B, Begg L, Davis C, Grandison L, Humble M, Khalsa J, Little AR, Peavy H, Pontzer C, Portnoy M, Sayre MH, Starke-Reed P, Zakhari S, Read J, Watson B, Guyer M (2009) The nih human microbiome project. Genome Research 19:2317–2323. doi:[10.1101/gr.096651.109](https://doi.org/10.1101/gr.096651.109)

4. Lloyd-Price J, Mahurkar A, Rahnavard G, Crabtree J, Orvis J, Hall AB, Brady A, Creasy HH, McCracken C, Giglio MG, McDonald D, Franzosa EA, Knight R, White O, Huttenhower C (2017) Strains, functions and dynamics in the expanded human microbiome project. Nature 550:61–66. doi:[10.1038/nature23889](https://doi.org/10.1038/nature23889)

5. R Core Team (2019) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria

6. Bacci G, Taccetti G, Dolce D, Armanini F, Segata N, Di Cesare F, Lucidi V, Fiscarelli E, Morelli P, Casciaro R, Negroni A, Mengoni A, Bevivino A (2020) Taxonomic variability over functional stability in the microbiome of cystic fibrosis patients chronically infected by pseudomonas aeruginosa. bioRxiv. doi:[10.1101/609057](https://doi.org/10.1101/609057)

7. Huerta-Cepas J, Forslund K, Coelho LP, Szklarczyk D, Jensen LJ, Mering C von, Bork P (2017) Fast Genome-Wide Functional Annotation through Orthology Assignment by eggNOG-Mapper. Molecular Biology and Evolution 34:2115–2122. doi:[10.1093/molbev/msx148](https://doi.org/10.1093/molbev/msx148)

8. Boogaart KG van den, Tolosana-Delgado R (2008) “Compositions”: A unified r package to analyze compositional data. Computers & Geosciences 34:320–338. doi:[10.1016/j.cageo.2006.11.017](https://doi.org/10.1016/j.cageo.2006.11.017)

9. Oksanen J, Blanchet FG, Friendly M, Kindt R, Legendre P, McGlinn D, Minchin PR, O’Hara RB, Simpson GL, Solymos P, Stevens MHH, Szoecs E, Wagner H (2019) Vegan: Community ecology package

10. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for rna-seq data with deseq2. Genome biology 15:550. doi:[10.1186/s13059-014-0550-8](https://doi.org/10.1186/s13059-014-0550-8)

11. Kuhn M (2015) Caret: Classification and regression training. Astrophysics Source Code Library

12. Liaw A, Wiener M (2002) Classification and regression by randomForest. R News 2:18–22

13. Breiman L (2001) Random forests. Machine learning 45:5–32. doi:[10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)

14. Karatzoglou A, Smola A, Hornik K, Zeileis A (2004) Kernlab – an S4 package for kernel methods in R. Journal of Statistical Software 11:1–20

15. Scholkopf B, Smola AJ (2001) Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge, MA, USA

16. Greenwell B, Boehmke B, Cunningham J, Developers G (2019) Gbm: Generalized boosted regression models

17. Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, Müller M (2011) PROC: An open-source package for r and s+ to analyze and compare roc curves. BMC Bioinformatics 12:77. doi:[10.1186/1471-2105-12-77](https://doi.org/10.1186/1471-2105-12-77)

18. Wickham H (2016) Ggplot2: Elegant graphics for data analysis. Springer-Verlag New York

19. Wilkinson L (2012) The grammar of graphics. In: Gentle JE, Härdle WK, Mori Y (eds) Handbook of computational statistics: Concepts and methods. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 375–414

20. Clarke E, Sherrill-Mix S (2017) Ggbeeswarm: Categorical scatter (violin point) plots

21. Graves S, Piepho H-P, Sundar Dorai-Raj LS with help from (2019) MultcompView: Visualizations of paired comparisons

22. AITCHISON J (1983) Principal component analysis of compositional data. Biometrika 70:57–65. doi:[10.1093/biomet/70.1.57](https://doi.org/10.1093/biomet/70.1.57)

23. Lovell VAE David AND Pawlowsky-Glahn (2015) Proportionality: A valid alternative to correlation for relative data. PLOS Computational Biology 11:1–12. doi:[10.1371/journal.pcbi.1004075](https://doi.org/10.1371/journal.pcbi.1004075)

24. Bacci G, Mengoni A, Fiscarelli E, Segata N, Taccetti G, Dolce D, Paganin P, Morelli P, Tuccio V, De Alessandri A, Lucidi V, Bevivino A (2017) A different microbiome gene repertoire in the airways of cystic fibrosis patients with severe lung disease. International Journal of Molecular Sciences 18: doi:[10.3390/ijms18081654](https://doi.org/10.3390/ijms18081654)

25. John R, Dalling JW, Harms KE, Yavitt JB, Stallard RF, Mirabello M, Hubbell SP, Valencia R, Navarrete H, Vallejo M, Foster RB (2007) Soil nutrients influence spatial distributions of tropical tree species. Proceedings of the National Academy of Sciences 104:864–869. doi:[10.1073/pnas.0604666104](https://doi.org/10.1073/pnas.0604666104)

26. Weiss S, Xu ZZ, Peddada S, Amir A, Bittinger K, Gonzalez A, Lozupone C, Zaneveld JR, Vázquez-Baeza Y, Birmingham A, Hyde ER, Knight R (2017) Normalization and microbial differential abundance strategies depend upon data characteristics. Microbiome 5:27. doi:[10.1186/s40168-017-0237-y](https://doi.org/10.1186/s40168-017-0237-y)

27. Abdi H, Williams LJ (2010) Principal component analysis. WIREs Computational Statistics 2:433–459. doi:[10.1002/wics.101](https://doi.org/10.1002/wics.101)