Microbial Data Analysis Course Part 1

Ece Kartal

Contents

This vignette contains the course schedule and material of the **Microbiome Data Analysis & Visualization Course**.

Loading packages and data in R

Before we start with any analysis, it is good practice to load all required libraries. This will also immediately identify libraries that may be missing. Note that for this course, we pre-installed all libraries for you. When you run your own analysis, you have to check which libraries are already available, and which are not. We use `suppressPackageStartupMessages` here to suppress the output messages of the various packages for reasons of brevity.

When using functions that sample pseudorandom numbers, each time you execute them you will obtain a different result. For the purpose of this vignette, this is not what we want. We therefore set a particular seed value(here: 1881) so that the result will always be the same. For more information, check out this webpage that explains this general concept in more detail.

```
suppressPackageStartupMessages({
library(knitr)
library(tidyverse)
library(ggrepel)
library(ggplot2)
library(pROC)
library(SIAMCAT)
})

set.seed(1881)
```

Next, we read the data tables that contain the count matrix and samples' metadata.

```
# set the working directory
folder <- gsub("/src", "", getwd())
folder.results <- paste0(folder, "/output/")
file.path(folder, 'data/mobi.Rdata')
```

```
## [1] "/Users/ecekartal/Documents/Academics-Work/Teaching/Microbiome_analysis_course_2023/data/mobi.Rda
```

```
# load data
load(file=file.path(folder, '/data/motu.relative.Rdata'))
```

Differential Abundance Analysis with Wilcoxin Test

Next, we can analyze the differences between the microbiome profiles of both groups. To do so, we use `wilcoxin test` which is a statistical test where all species in the count matrix are compared between the sample groups of interest. `wilcoxin test` is not the only option to perform Differential Abundance Analysis. Common alternatives include `edgeR` and `DESeq2`.

A **Wilcoxon test** estimates the difference in an outcome between two groups. It is a non-parametric alternative to a t-test, which means that the Wilcoxon test does not make any assumptions about the data.

```
featTable = motu.fil.rel
# subset meta accordingly
metaTable = meta[meta$ID %in% colnames(motu.fil.rel),]
# define cutoffs
p_cutoff = 0.05
################################################################################
p.cal <- tribble(~ID, ~pval, ~adj.pval, ~log10.adj, ~aucs.mat, ~fc, ~sig)

for (rowname in row.names(featTable)) {

    # define matrix to compare
    x <- as.numeric(featTable[rowname, metaTable %>% filter(status=='PC') %>% pull(ID)])
    y <- as.numeric(featTable[rowname, metaTable %>% filter(status=='CTR') %>% pull(ID)])

    # Fold change
    q.p <- quantile(log10(x+1e-05), probs=seq(.1, .9, .05))
    q.n <- quantile(log10(y+1e-05), probs=seq(.1, .9, .05))

    # create matrix
    p.cal=add_row(p.cal,
                  ID = rowname,
                  pval = wilcox.test(x, y, exact=FALSE)$p.value,
                  aucs.mat = roc(controls=y, cases=x, direction='<', ci=TRUE, auc=TRUE)$ci[2],
                  fc = sum(q.p - q.n)/length(q.p))
}

# p.adjust
p.cal <- p.cal %>%
  mutate(adj.pval = p.adjust(pval, method = "BH")) %>%
  mutate(sig = ifelse(adj.pval < p_cutoff, "p.adj < 0.05", "not sig")) %>%
  mutate(log10.adj = -log10(adj.pval))

# Lets have a look to results
p.cal
```

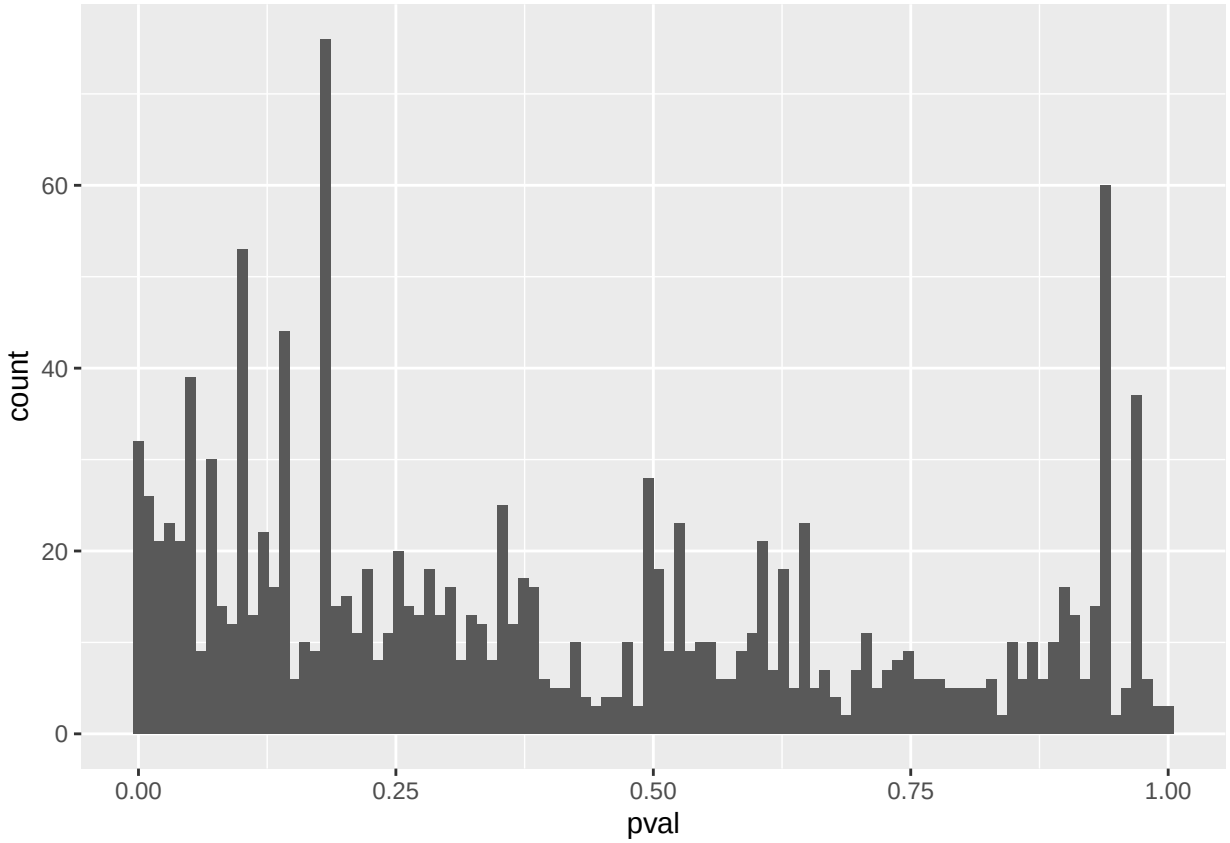
```
## # A tibble: 1,343 x 7
##    ID                          pval adj.pval log10.adj aucs.mat      fc sig
##    <chr>                      <dbl>    <dbl>     <dbl>    <dbl>   <dbl> <chr>
```

```
##  1 Abiotrophia defectiva [r_0~ 0.115      0.502   0.299      0.545  0.0155 not ~
##  2 Absiella dolichum [r_03694] 0.00852    0.250   0.602      0.438 -0.0545 not ~
##  3 Acetobacter sp. [r_03587]   0.423      0.729   0.137      0.466 -0.222  not ~
##  4 Acholeplasma sp. CAG:878 [~ 0.0677     0.482   0.317      0.469  0      not ~
##  5 Acidaminococcus fermentans~ 0.351      0.669   0.174      0.509  0      not ~
##  6 Acidaminococcus intestini ~ 0.559      0.817   0.0878     0.524  0.0848 not ~
##  7 Acidaminococcus massiliens~ 0.966      0.979   0.00916    0.499  0      not ~
##  8 Acinetobacter sp. [r_02372] 0.224      0.567   0.246      0.451 -0.241  not ~
##  9 Acinetobacter sp. [r_03673] 0.354      0.672   0.172      0.461 -0.189  not ~
## 10 Acinetobacter species  [m_~ 0.330      0.661   0.179      0.460 -0.237  not ~
## # i 1,333 more rows
```

```
# save file
write.table(p.cal, file=paste0(folder.results, 'wilcox.results.tsv'),
            sep='\t', row.names=TRUE, col.names=TRUE)
```

And look at the distribution of the adjusted P values:

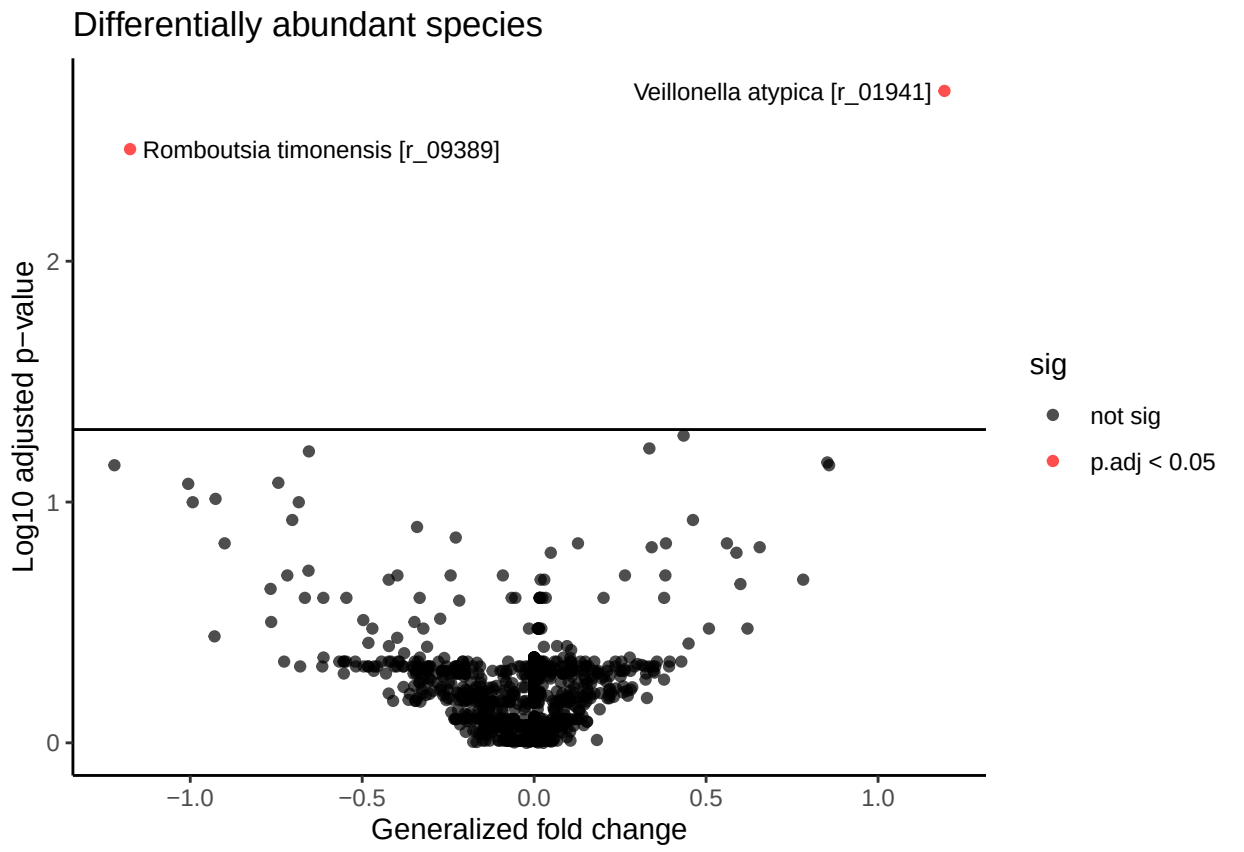
```
ggplot2::ggplot(p.cal, aes(x = pval)) +
  ggplot2::geom_histogram(bins = 100)
```



NOTE: Do you remember how the P-value distribution should look like? If not, please see here.

We can also visualize the abundance changes using one of the most common plots to explore Differential Abundance Analysis results, the **volcano plot**:

```
# Volcano plot
p.wilcox <- ggplot(p.cal, aes(x = fc, y = log10.adj)) +
  geom_point(aes(color = sig), alpha=0.7) +
  scale_color_manual(values = c("black","red")) +
  # add a line for significance
  geom_hline(yintercept=-log10(p_cutoff)) +
  # add ID for significant species
  geom_text_repel(data = subset(p.cal, adj.pval < p_cutoff), aes(label = ID), size = 3) +
  # add axis labels
  ggtitle("Differentially abundant species") +
  xlab("Generalized fold change") +
  ylab("Log10 adjusted p-value") +
  theme_classic()

p.wilcox
```




```
# save the plot
ggsave(p.wilcox, filename=paste0(folder.results, "wilcox.volcano.plot.pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

A volcano plot enables us to quickly visualize the magnitude (logFC) and significance (-log10(pvalue)) of DAA changes. Each point represent a species, and its color indicates whether they surpass or not a cutoff of an adjusted P value < 0.05.

QUESTION: What does the horizontal line mean in this plot? Take a look at the code and think about their meaning.

QUESTION: Try to modify the and `p_cutoff` variables in this chunk. What happens?

Modelling by SIAMCAT

`SIAMCAT` (Statistical Inference of Associations between Microbial Communities And host phenoTypes) is a comprehensive toolbox for comparative metagenome analysis using ML, statistical modeling, and advanced visualization approaches paper here.

`SIAMCAT` needs:

- a feature matrix (matrix or data.frame) that has features (in rows) samples (in columns)
- metadata in a data.frame, samples as row names

For microbiome studies, many issues arise from key characteristics of metagenomic data such as

- large technical and inter-individual variation
- experimental bias,
- compositionality of relative abundances,
- zero inflation,
- non-Gaussian distribution, all of which necessitate data normalization in order for ML algorithms to work well.

Please note that `SIAMCAT` is supposed to work with *relative abundances*.

```
# define files and parameters
meta.train <- meta
feat.train <- motu.rel
test.meta <- c("age",  "center", "smoking", "periodontitis", "gender", "alcohol_status")

# start modelling
siamcat <- siamcat(feat=feat.train, meta=meta.train, label = 'status', case='PC')
```

```
## + starting create.label

## Label used as case:
##    PC
## Label used as control:
##    CTR

## + finished create.label.from.metadata in 0.003 s

## + starting validate.data

## +++ checking overlap between labels and features

## + Keeping labels of 107 sample(s).

## +++ checking sample number per class
```

```
## +++ checking overlap between samples and metadata

## + finished validate.data in 0.293 s
```

```
# SIAMCAT builds on the phyloseq data structure
show(siamcat)
```

```
## siamcat-class object
## label()                Label object:         50 CTR and 57 PC samples
##
## contains phyloseq-class experiment-level object @phyloseq:
## phyloseq@otu_table()   OTU Table:            [ 14212 taxa and 107 samples ]
## phyloseq@sam_data()    Sample Data:          [ 107 samples by 38 sample variables ]
```

Since we have quite a lot of microbial species in the dataset at the moment, we can perform *unsupervised feature selection* using the function `filter.features`

```
# filter based on abundance
siamcat <- filter.features(siamcat, filter.method = 'abundance',
                           cutoff=0.001, verbose=3)
```

```
## + starting filter.features

## +++ before filtering, the data have 14212 features

## +++ applying abundance filter

## +++ checking for unmapped reads

## +++ tried to remove unmapped reads but could not find any. Continue anyway.

## +++ removed 13171 features whose values did not exceed 0.001 in any sample (retaining 1041)

## +++ saving filtered features

## + finished filter.features in 0.021 s
```

```
# The check.confounders function provides the option to test the associated metadata
# variables for potential confounding influence.
check.confounders(siamcat, fn.plot = paste0(folder.results, 'confounders.pdf'),
                  meta.in = test.meta, verbose = 3)
```

```
## + starting check.confounders

## ++ metadata variables:
##  ID
## ++ have too many levels and have been removed from this analysis

## +++ plotting conditional entropies for metadata variables
```

```
## +++ building logistic regression classifiers for metadata

## +++ plotting regression coefficients

## +++ plotting regression coefficient significance

## +++ plotting au-roc values

## +++ checking Age as a potential confounder

## ++++ continuous variable, using a Q-Q plot

## ++++ panel 1/4: Q-Q plot

## ++++ panel 2/4: X histogram

## ++++ panel 3/4: X boxplot

## ++++ panel 4/4: Y histogram

## +++ checking Center as a potential confounder

## ++++ discrete variable, using a bar plot

## ++++ plotting barplot

## ++++ drawing contingency table

## +++ checking Smoking as a potential confounder

## ++++ discrete variable, using a bar plot

## ++++ plotting barplot

## ++++ drawing contingency table

## +++ checking Periodontitis as a potential confounder

## ++++ discrete variable, using a bar plot

## ++++ plotting barplot

## ++++ drawing contingency table

## +++ checking Gender as a potential confounder

## ++++ discrete variable, using a bar plot
```

```
## ++++ plotting barplot

## ++++ drawing contingency table

## +++ checking Alcohol status as a potential confounder

## ++++ discrete variable, using a bar plot

## ++++ plotting barplot

## ++++ drawing contingency table

## +++ computing variance explained by label

## +++ computing variance explained by age

## +++ computing variance explained by center

## +++ computing variance explained by smoking

## +++ computing variance explained by periodontitis

## +++ computing variance explained by gender

## +++ computing variance explained by alcohol_status

## + finished check.confounders in 1.18 s
```

```
# normalise count matrix
siamcat <- normalize.features(siamcat, norm.method = "log.clr",
                              norm.param = list(log.n0 = 1e-05, sd.min.q = 1),
                              verbose=3)
```

```
## + starting normalize.features

## +++ performing de novo normalization using the  log.clr  method

## +++ checking if parameters are compatible with each other

## + feature sparsity before normalization: 79.38%

## +++ performing normalization

## +++ feature sparsity after normalization:      0 %

## + finished normalize.features in 0.007 s
```

```
# Associations between microbial species and the label can be tested with the
# *check.associations* function
siamcat <- check.associations(siamcat,
                              feature.type = 'normalized')
```

SIAMCAT contains functions for data normalization, splitting the data into cross-validation folds, training the model, and making predictions based on cross-validation instances and the trained models.Here, we choose a 10 times-repeated 10-fold cross-validation scheme.

The models are saved in the **model_list** slot of the *SIAMCAT* object. The model building is performed using the `mlr` R package. All models can easily be accessed.

```
# split data for nested cross validation
siamcat <- create.data.split(siamcat, num.folds = 5, num.resample = 5)
```

```
## Features splitted for cross-validation successfully.
```

```
# run model
siamcat <- train.model(siamcat, method = "lasso_ll")
```

```
## Trained lasso_ll models successfully.
```

We want to find out how well the model performed and which microbial species had been selected in the model. In order to do so, we first calculate how well the predictions fit the real data using the function `evaluate.predictions`. This function calculates the Area Under the Receiver Operating Characteristic (ROC) Curve (**AU-ROC**) and the Precision Recall (PR) Curve for each resampled cross-validation run.

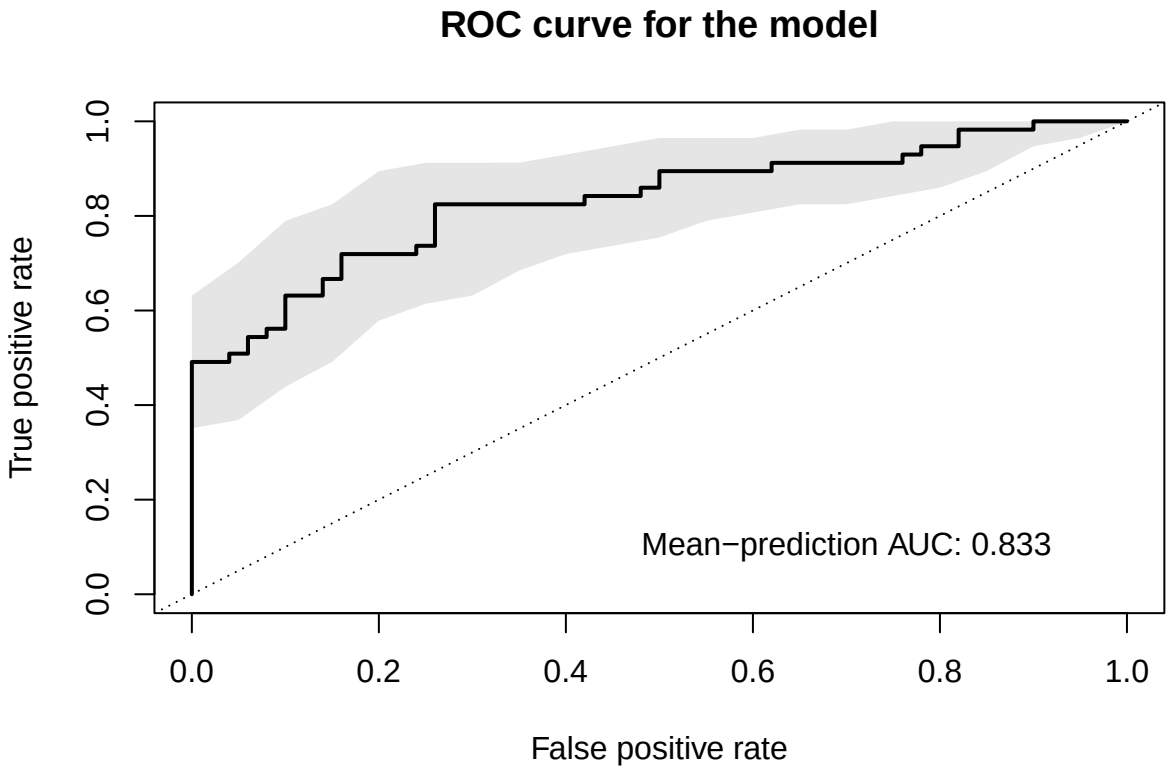
```
siamcat <- make.predictions(siamcat)
```

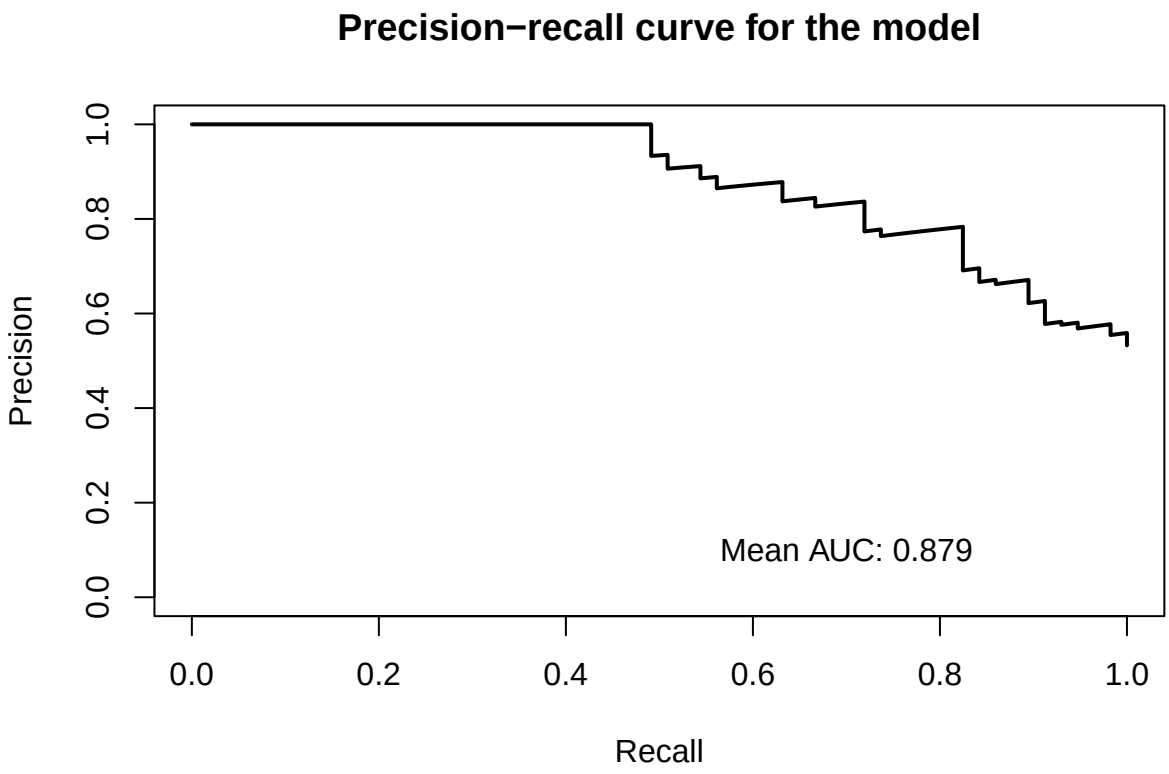
```
## Made predictions successfully.
```

```
# evaluate model
siamcat <- evaluate.predictions(siamcat)
```

```
## Evaluated predictions successfully.
```

```
# evaluation plot based on AUROC
model.evaluation.plot(siamcat)
```


ROC curve for the model
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean−prediction AUC: 0.833




```
# to save it
model.evaluation.plot(siamcat, fn.plot = paste0(folder.results, 'evaluation.plot.pdf'))
```

```
## Plotted evaluation of predictions successfully to: /Users/ecekartal/Documents/Academics-Work/Teachin
```

The final plot produced by `SIAMCAT` is the model interpretation plot, created by the `model.interpretation.plot` function. The plot shows for the top selected features the

- model weights (and how robust they are) as a barplot,
- a heatmap with the z-scores or fold changes for the top selected features, and
- a boxplot showing the proportions of weight per model which is captured by the top selected features.

```
# interpretation plot with heatmap
model.interpretation.plot(siamcat,
                          consens.thres = 0.5,
                          heatmap.type = 'zscore',
                          fn.plot = paste0(folder.results, 'interpret.plot.pdf'))
```

```
## Successfully plotted model interpretation plot to: /Users/ecekartal/Documents/Academics-Work/Teachin
```

```
# save modelling matrix
save(siamcat, file=paste0(folder.results, 'siamcat.lassoll.model.RData'))
```

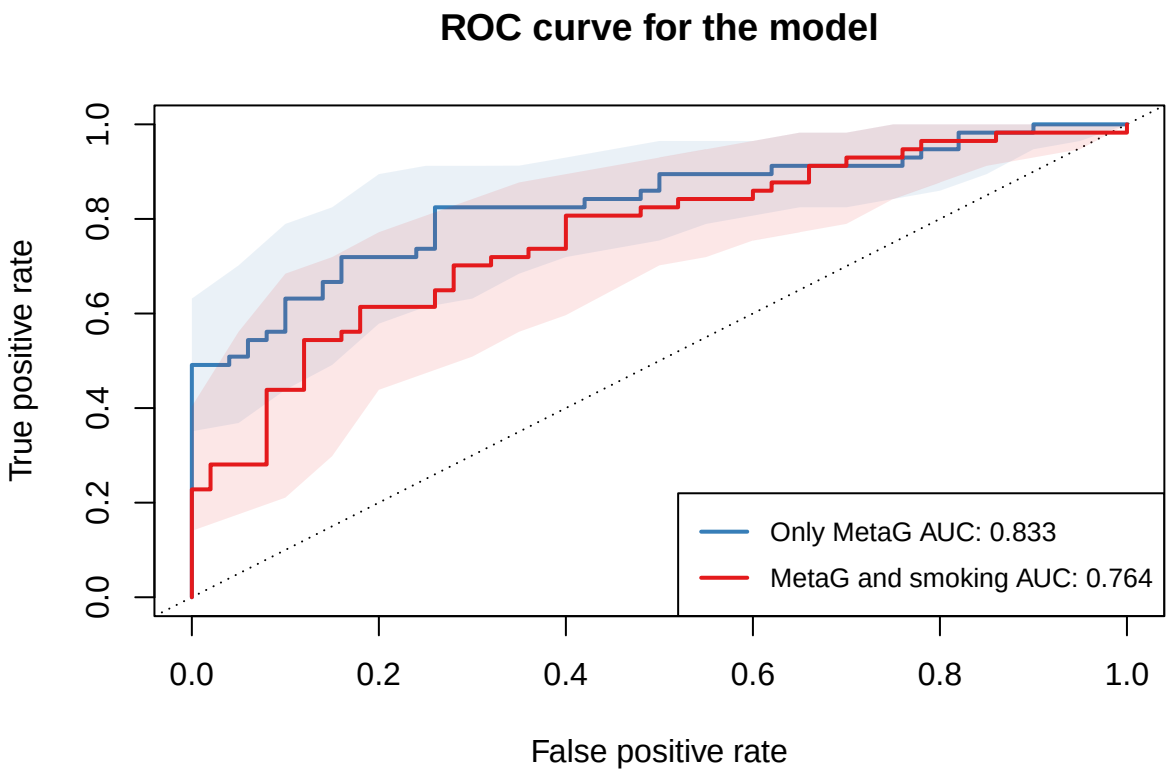
Adding metadata variables to microbial model

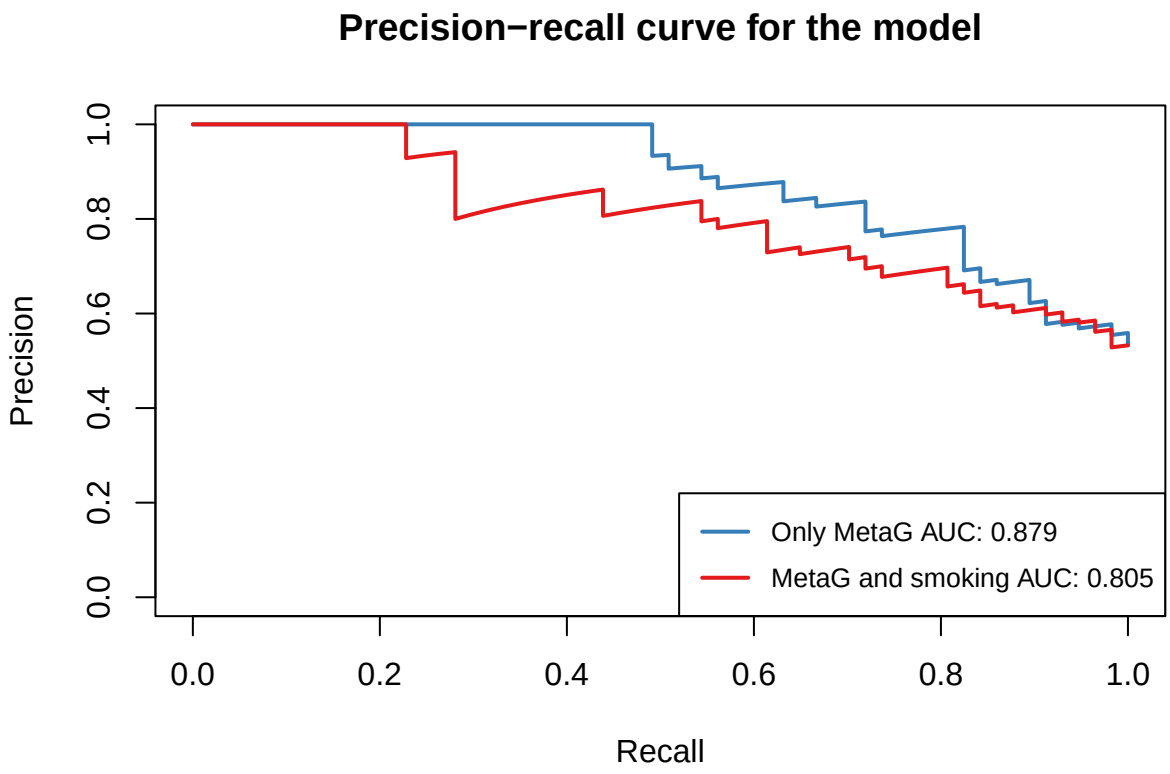
`add.meta.pred` functions adds one or several metadata variables to the set of features, so that they can be included for model training. Numerical meta-variables are added as z-scores to the feature matrix unless specified otherwise.

```
# add metadata to the feature matrix to be later used as predictors
add.meta <- function(x, n){
  x <- add.meta.pred(x, pred.names = n, verbose=3,std.meta=FALSE)
  x <- train.model(x, method='lasso_ll', verbose=2, perform.fs = TRUE)

  x <- make.predictions(x)
  x <- evaluate.predictions(x)
  return(x)
}
# combine smoking information with naive microbiome model
siamcat.smoking<- add.meta(siamcat, 'gender')
```

```
## + starting add.meta.pred

## + starting to add metadata predictors

## +++ adding metadata predictor: gender

## +++ added 1 meta-variables as predictor to the feature matrix

## + finished add.meta.pred in 0.003 s

## + starting train.model

## + training lasso_ll models on 25 training sets

## + Performing feature selection with following parameters:

##     no_features = 100

##     method = AUC

##     direction = absolute

## + finished train.model in 47.8 s

## Made predictions successfully.

## Evaluated predictions successfully.
```

```
# lets see the comparison of only microbiome-based model and microbiome+smoking
model.evaluation.plot('Only MetaG'= siamcat,
                      'MetaG and smoking'= siamcat.smoking)
```


ROC curve for the model
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Only MetaG AUC: 0.833
MetaG and smoking AUC: 0.764




```
# to save it
model.evaluation.plot('Only MetaG'= siamcat,
                      'MetaG and smoking'= siamcat.smoking,
                      fn.plot = paste0(folder.results, 'metag.smoking.combined.pdf'))
```

```
## Plotted evaluation of predictions successfully to: /Users/ecekartal/Documents/Academics-Work/Teaching
```

Session info

It is good practice to print the so-called session info at the end of an R script, which prints all loaded libraries, their versions etc. This can be helpful for reproducibility and recapitulating which package versions have been used to produce the results obtained above.

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Sonoma 14.0
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
```

```
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] SIAMCAT_2.6.0   phyloseq_1.46.0 mlr3_0.17.0     pROC_1.18.5
##  [5] ggrepel_0.9.4   lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
##  [9] dplyr_1.1.4     purrr_1.0.2     readr_2.1.4     tidyr_1.3.0
## [13] tibble_3.2.1    ggplot2_3.4.4   tidyverse_2.0.0 knitr_1.45
##
## loaded via a namespace (and not attached):
##   [1] beanplot_1.3.1          bitops_1.0-7            gridExtra_2.3
##   [4] permute_0.9-7           rlang_1.1.2             magrittr_2.0.3
##   [7] gridBase_0.4-7          ade4_1.7-22             matrixStats_1.1.0
##  [10] compiler_4.3.1          mgcv_1.9-0              systemfonts_1.0.5
##  [13] vctrs_0.6.4             reshape2_1.4.4          pkgconfig_2.0.3
##  [16] shape_1.4.6             crayon_1.5.2            fastmap_1.1.1
##  [19] backports_1.4.1         XVector_0.42.0          labeling_0.4.3
##  [22] PRROC_1.3.1             utf8_1.2.4              rmarkdown_2.25
##  [25] tzdb_0.4.0              nloptr_2.0.3            ragg_1.2.6
##  [28] xfun_0.41               glmnet_4.1-8            zlibbioc_1.48.0
##  [31] mlr3misc_0.13.0         GenomeInfoDb_1.38.1     jsonlite_1.8.7
##  [34] progress_1.2.2          biomformat_1.30.0       highr_0.10
##  [37] rhdf5filters_1.14.1     uuid_1.1-1              Rhdf5lib_1.24.0
##  [40] mlr3measures_0.5.0      prettyunits_1.2.0       parallel_4.3.1
##  [43] cluster_2.1.4           R6_2.5.1                RColorBrewer_1.1-3
##  [46] stringi_1.8.1           boot_1.3-28.1           parallelly_1.36.0
##  [49] numDeriv_2016.8-1.1     Rcpp_1.0.11             iterators_1.0.14
##  [52] future.apply_1.11.0     IRanges_2.36.0          Matrix_1.6-3
##  [55] splines_4.3.1           igraph_1.5.1            timechange_0.2.0
##  [58] tidyselect_1.2.0        rstudioapi_0.15.0       yaml_2.3.7
##  [61] mlr3tuning_0.19.1       vegan_2.6-4             codetools_0.2-19
##  [64] listenv_0.9.0           lmerTest_3.1-3          lattice_0.22-5
##  [67] plyr_1.8.9              Biobase_2.62.0          withr_2.5.2
##  [70] evaluate_0.23           future_1.33.0           survival_3.5-7
##  [73] Biostrings_2.70.1       infotheo_1.2.0.1        pillar_1.9.0
##  [76] corrplot_0.92           checkmate_2.3.0         foreach_1.5.2
##  [79] stats4_4.3.1            generics_0.1.3          bbotk_0.7.3
##  [82] RCurl_1.98-1.13         S4Vectors_0.40.1        hms_1.1.3
##  [85] munsell_0.5.0           scales_1.2.1            minqa_1.2.6
##  [88] globals_0.16.2          glue_1.6.2              LiblineaR_2.10-22
##  [91] tools_4.3.1             data.table_1.14.8       lme4_1.1-35.1
##  [94] rhdf5_2.46.0            grid_4.3.1              ape_5.7-1
##  [97] colorspace_2.1-0        paradox_0.11.1          nlme_3.1-163
## [100] GenomeInfoDbData_1.2.11 palmerpenguins_0.1.1    cli_3.6.1
## [103] textshaping_0.3.7       fansi_1.0.5             gtable_0.3.4
## [106] digest_0.6.33           BiocGenerics_0.48.1     farver_2.1.1
## [109] lgr_0.4.4               htmltools_0.5.7         multtest_2.58.0
## [112] lifecycle_1.0.4         mlr3learners_0.5.7      MASS_7.3-60
```