HMP_2012_16S_gingival_V35_subset

```
library(benchdamic)
library(phyloseq)
library(MicrobiomeBenchmarkData)
library(dplyr)
```

```
dat_names <- getDataset()
tse <- getDataset(dat_names[4], dryrun = FALSE)[[1]]
```

Create prior knowledge info

```
biologicalInfo <- tse %>%
    rowData() %>%
    as.data.frame() %>%
    select(GENUS, BIOSIS) %>%
    mutate(
        newNames = paste0(rownames(.), "|", GENUS),
        BIOSIS = ifelse(is.na(BIOSIS), "unknown", BIOSIS)
    ) %>%
    rename(Type = BIOSIS)
```

Convert to phyloseq

```
## Biosis information is lost, but it's already stored in the biologicalInfo
## variable
ps <- mia::makePhyloseqFromTreeSummarizedExperiment(tse)
```

Useful variables

```
grp <- "hmp_body_subsite"
contrast <- c("subgingival_plaque", "supragingival_plaque")
```

Set reference

```
sample_data(ps)[[grp]] <- factor(sample_data(ps)[[grp]])
sample_data(ps)[[grp]] <- relevel(
    x = sample_data(ps)[[grp]], ref = contrast[1]
)
```

Prepare normalization

```
## Normalization methods
norm_pars <- tibble::tribble(
    ~fun, ~method,
    "norm_edgeR", "none",
    "norm_edgeR", "TMM",
    "norm_DESeq2", "poscounts",
    "norm_CSS", "median"
)

## set normalization
```

```
my_norm <- setNormalizations(fun = norm_pars$fun, method = norm_pars$method)

## Run normalization
ps <- runNormalizations(normalization_list = my_norm, object = ps)
```

```
##       + Running now: norm_edgeR
##         Parameters: method=none
##       + Running now: norm_edgeR
##         Parameters: method=TMM
##       + Running now: norm_DESeq2
##         Parameters: method=poscounts
##       + Running now: norm_CSS
##         Parameters: method=median
```

Create weights

```
zinbweights <- weights_ZINB(
    object = ps,
    K = 0,
    design = "~ 1"
)
```

Prepare DA methods

```
# edgeR
my_edger <- set_edgeR(
    group_name = grp,
    design = as.formula(paste0("~", grp)),
    norm = "TMM",
    coef = 2
)

# DESeq2
my_deseq2 <- set_DESeq2(
    contrast = c(grp, contrast),
    design = as.formula(paste0("~", grp)),
    norm = "poscounts"
)

# limma
my_limma <- set_limma( # I get a warning
    design = as.formula(paste0("~", grp)),
    norm = c("TMM", "CSSmedian"),
    coef = 2
)

# metagenomeSeq
my_metagenomeseq <- set_metagenomeSeq(
    design = as.formula(paste0("~", grp)),
    norm = "CSSmedian",
    coef = 2
)

# ALDEx2
my_aldex2 <- set_ALDEx2(
```

```
    conditions = grp,
    test = "t",
    norm = "none"
)

# corncob
my_corncob <- set_corncob(
    formula = as.formula(paste0("~", grp)),
    phi.formula = as.formula(paste0("~", grp)),
    formula_null = ~ 1,
    phi.formula_null = as.formula(paste0("~", grp)),
    test = "Wald",
    coefficient = paste0(grp, contrast[2]),
    norm = "none"
)

# MAST
my_mast <- set_MAST(
    rescale = "median",
    design = as.formula(paste0("~", grp)),
    coefficient = paste0(grp, contrast[2]),
    norm = "none"
)

# Seurat
my_seurat <- set_Seurat(
    test.use = "wilcox",
    contrast = c(grp, contrast),
    norm = "none"
)

my_methods <- c(
    my_edger, my_deseq2, my_limma, my_metagenomeseq, my_aldex2,
    my_corncob, my_mast, my_seurat
)
```

Prepare directions (for enrichment)

```
direction <- c(
    edgeR.TMM = "logFC",
    DESeq2.poscounts = "log2FoldChange",
    limma.CSSmedian = "logFC",
    limma.TMM = "logFC",
    metgenomeSeq.CSSmedian = paste0(grp, contrast[2]),
    ALDEx2.none = "effect",
    corncob.none = "Estimate",
    MAST.none = "logFC",
    Seurat.none = "avg_log2FC"
)
```

Run DA

```
DA <- runDA(method_list = my_methods, object = ps, weights = zinbweights)
```

```
##       * Running now: DA_edgeR
##         Parameters: pseudo_count=FALSE, group_name=hmp_body_subsite, design=~hmp_body_subsite, coef=
##       * Running now: DA_DESeq2
##         Parameters: pseudo_count=FALSE, design=~hmp_body_subsite, contrast=hmp_body_subsite.subgingi
##       * Running now: DA_limma
##         Parameters: pseudo_count=FALSE, design=~hmp_body_subsite, coef=2, norm=CSSmedian, weights=FA
##       * Running now: DA_limma
##         Parameters: pseudo_count=FALSE, design=~hmp_body_subsite, coef=2, norm=TMM, weights=FALSE
##       * Running now: DA_metagenomeSeq
##         Parameters: pseudo_count=FALSE, design=~hmp_body_subsite, coef=2, norm=CSSmedian
## it= 0, nll=121.47, log10(eps+1)=Inf, stillActive=892
## it= 1, nll=132.30, log10(eps+1)=0.03, stillActive=85
## it= 2, nll=132.07, log10(eps+1)=0.05, stillActive=64
## it= 3, nll=132.16, log10(eps+1)=0.03, stillActive=15
## it= 4, nll=132.13, log10(eps+1)=0.04, stillActive=5
## it= 5, nll=132.07, log10(eps+1)=0.02, stillActive=4
## it= 6, nll=132.03, log10(eps+1)=0.00, stillActive=1
## it= 7, nll=131.99, log10(eps+1)=0.00, stillActive=1
## it= 8, nll=131.97, log10(eps+1)=0.00, stillActive=1
## it= 9, nll=131.95, log10(eps+1)=0.00, stillActive=1
## it=10, nll=131.94, log10(eps+1)=0.00, stillActive=1
## it=11, nll=131.93, log10(eps+1)=0.00, stillActive=1
## it=12, nll=131.92, log10(eps+1)=0.00, stillActive=1
## it=13, nll=131.92, log10(eps+1)=0.00, stillActive=1
## it=14, nll=131.91, log10(eps+1)=0.00, stillActive=1
## it=15, nll=131.91, log10(eps+1)=0.00, stillActive=1
## it=16, nll=131.91, log10(eps+1)=0.00, stillActive=1
## it=17, nll=131.91, log10(eps+1)=0.00, stillActive=1
## it=18, nll=131.90, log10(eps+1)=0.01, stillActive=1
## it=19, nll=131.90, log10(eps+1)=0.02, stillActive=1
## it=20, nll=131.89, log10(eps+1)=0.02, stillActive=1
## it=21, nll=131.90, log10(eps+1)=0.00, stillActive=0
##       * Running now: DA_ALDEx2
##         Parameters: pseudo_count=FALSE, conditions=hmp_body_subsite, mc.samples=128, test=t, denom=i
## |------------(25%)----------(50%)----------(75%)----------|
##       * Running now: DA_corncob
##         Parameters: pseudo_count=FALSE, formula=~.hmp_body_subsite, formula_null=~.1, phi.formula=~.
##       * Running now: DA_MAST
##         Parameters: pseudo_count=FALSE, design=~.hmp_body_subsite, coefficient=hmp_body_subsitesupra
##       * Running now: DA_Seurat
##         Parameters: pseudo_count=FALSE, contrast=hmp_body_subsite.subgingival_plaque.supragingival_p
```

Run ANCOMBC

```
ancombc <- function(ps, formula, group) {
    out <- ANCOMBC::ancombc(phyloseq = ps, formula = formula,
        p_adj_method = "bonferroni", zero_cut = 0.90, lib_cut = 1000,
        group = group, struc_zero = TRUE, neg_lb = TRUE,
```

```
        tol = 1e-5, max_iter = 100, conserve = TRUE, alpha = 0.05,
        global = TRUE)
    res <- out$res
    ### extract important statistics ###
    vector_of_pval <- res$p_val[[1]] # contains the p-values
    vector_of_adjusted_pval <- res$q_val[[1]] # contains the adjusted p-values
    name_of_your_features <- rownames(res$p_val) # contains the OTU, or ASV, or other feature
    # names. Usually extracted from the rownames of
    # the count data
    vector_of_logFC <- res$beta[[1]] # logos the logFCs
    vector_of_statistics <- res$beta[[1]] # contains other statistics

    ### prepare the output ###
    pValMat <- data.frame("rawP" = vector_of_pval,
        "adjP" = vector_of_adjusted_pval)
    statInfo <- data.frame("logFC" = vector_of_logFC,
        "statistics" = vector_of_statistics)
    name <- "ANCOMBC"
    # Be sure that your method hasn't changed the order of the features. If it
    # happens, you'll need to re-establish the original order.
    rownames(pValMat) <- rownames(statInfo) <- name_of_your_features

    # Return the output as a list
    return(list("pValMat" = pValMat, "statInfo" = statInfo, "name" = name))
}
my_ancombc <-  ancombc(ps, grp, grp)
DA$ancombc <- my_ancombc
```

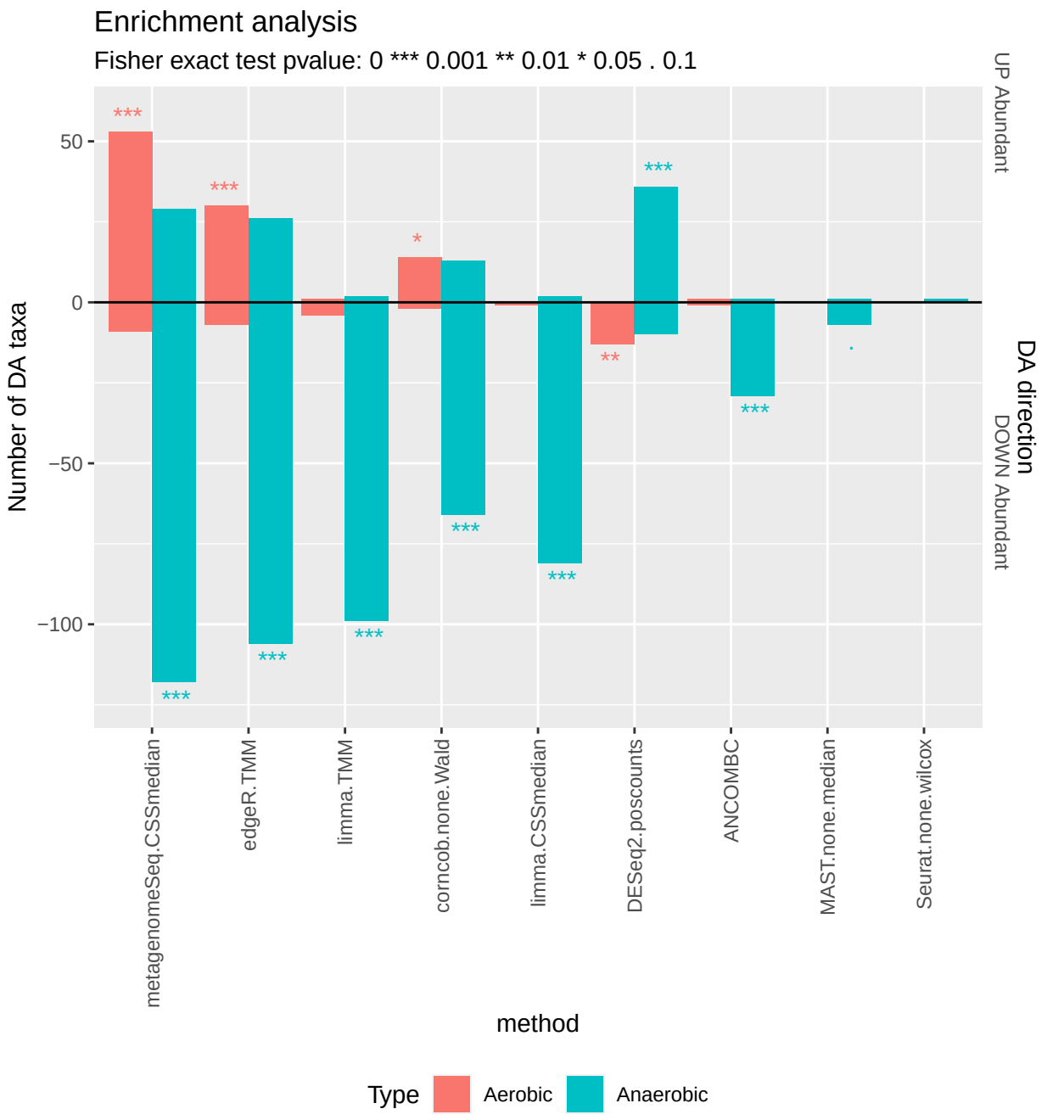
Enrichment

Run enrichment

```
enrichment <- createEnrichment(
    object = DA,
    priorKnowledge = biologicalInfo,
    enrichmentCol = "Type",
    namesCol = "newNames",
    slot = "pValMat",
    colName = "adjP",
    type = "pvalue",
    direction = direction,
    threshold_pvalue = 0.1,
    threshold_logfc = 0,
    top = NULL,
    alternative = "greater",
    verbose = TRUE
)
```

Plot enrichment

```
plotEnrichment(
    enrichment = enrichment,
    enrichmentCol = "Type",
    levels_to_plot = c("Aerobic", "Anaerobic")
)
```



TRUE and FALSE positives

Create table of positives

```
positives <- createPositives(
    object = DA,
    priorKnowledge = biologicalInfo,
    enrichmentCol = "Type",
    namesCol = "newNames",
    slot = "pValMat",
    colName = "rawP",
    type = "pvalue",
    direction = direction,
    threshold_pvalue = 1,
    threshold_logfc = 0,
    top = seq.int(from = 0, to = 50, by = 5),
    alternative = "greater",
    verbose = FALSE,
    TP = list(c("DOWN Abundant", "Anaerobic"), c("UP Abundant", "Aerobic")),
    FP = list(c("DOWN Abundant", "Aerobic"), c("UP Abundant", "Anaerobic"))
)
```

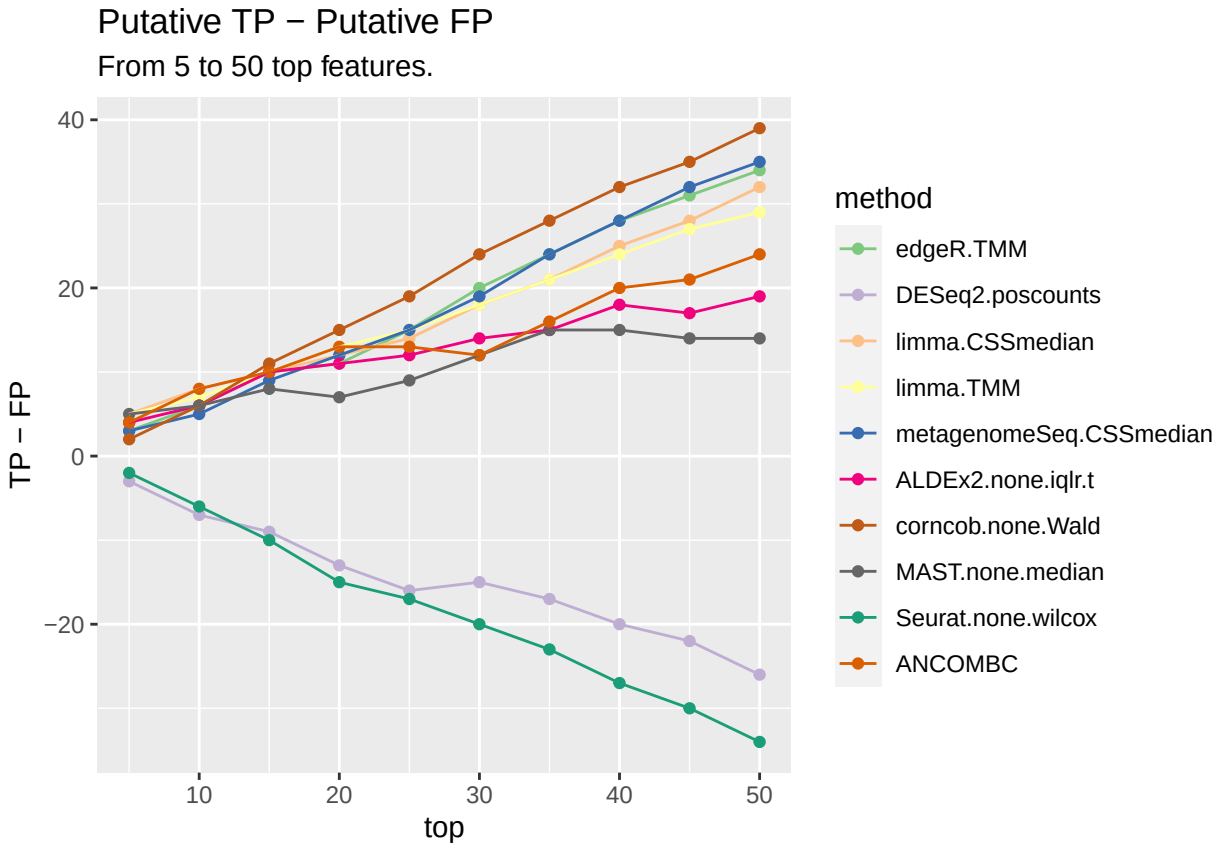
Table of positives:

```
head(positives)
```

```
##   top                  method TP FP
## 1   5               edgeR.TMM  4  1
## 2   5        DESeq2.poscounts  0  3
## 3   5         limma.CSSmedian  5  0
## 4   5               limma.TMM  5  0
## 5   5 metagenomeSeq.CSSmedian  3  0
## 6   5      ALDEx2.none.iqlr.t  4  0
```

Plot positives

```
plotPositives(positives)
```


Putative TP − Putative FP
From 5 to 50 top features.
TP − FP
40
20
0
−20
10
20
30
40
50
top
method
edgeR.TMM
DESeq2.poscounts
limma.CSSmedian
limma.TMM
metagenomeSeq.CSSmedian
ALDEx2.none.iqlr.t
corncob.none.Wald
MAST.none.median
Seurat.none.wilcox
ANCOMBC