FMT

Farhad M. Panah

2024-11-14

Library

```
library(tidyverse)
library(phyloseq)
library(readr)
library(glue)
library(heatmaply)
library(dplyr) # dpyr masks select from plotly
library(readr)
library(stringr)
library(tidyr)
library(tibble)
library(ggplot2)
library(ggrepel)
library(ggbeeswarm)
library(pheatmap)
library(grid)
library(microbiome)
library(ape)
library(vegan)
library(useful)
library(kableExtra)
library(ggtree)
library(arrow)
library(yaml)
library(phyloseq)
library(tidytree) # overwrites tree from phyloseq

library(ape)
library(phytools)
library(KEGGREST)
```

```
setwd("/home/hackion/Dropbox/Old-2gb/postdoc/kenneth")

taxonomy_file = "./finals/gtdb_taxonomy.tsv"
tree_file  =  "./finals/gtdbtk.bac120.nwk"
tree_arch = "./finals/gtdbtk.ar53.nwk"
quality_file = "./finals/genome_quality.tsv"
counts_file = "./finals/counts_genomes.parquet"
```

```
abundance_file =  "./finals/median_coverage_genomes.parquet"
readstats_file =  "./finals/read_counts.tsv"
keggmodules_file = "./finals/kegg_modules.tsv"
dram= "./finals/dram_annotations.tsv"
dram_xlsx = "./finals/metabolism_summary.xlsx"
gene2genomes= "./finals/gene2genome.parquet"
bin2genome = "./finals/allbins2genome.tsv"

# Gene catalog
coverage_stats = "Genecatalog/counts/gene_coverage_stats.parquet"
coverage = "Genecatalog/counts/median_coverage.h5"
counts = "Genecatalog/counts/Nmapped_reads.h5"
sample_stats = "Genecatalog/counts/sample_coverage_stats.tsv"
geneinfo = "Genecatalog/clustering/orf_info.parquet"
eggnog = "Genecatalog/annotations/eggNOG.parquet"
kegg = "Genecatalog/annotations/dram/kegg.parquet"
cazy = "Genecatalog/annotations/dram/cazy.parquet"
pfam = "Genecatalog/annotations/dram/pfam.parquet"

# arrow::read_parquet("./finals/gene_catalog/cazy.parquet") %>% data.frame()
```

Kraken

```
tsvs = list.files('krak_out_noConf/', pattern = '\\.tsv')



df = list()
for(i in seq_along(tsvs)){
    name = gsub('\\.tsv$', '', tsvs[[i]])
    df[[name]] <- read_tsv(glue('{getwd()}/krak_out_noConf/{tsvs[[i]]}'), col_names = F)
%>% filter(str_count(X1, '\\|') >=6) %>% tidyr::separate(X1, into = c('Kingdom','Phylum',
'Class', 'Order', 'Family', 'Genus', 'Species'), sep = '\\|') %>%
mutate(across(c(Kingdom, Phylum, Class, Order, Family, Genus, Species), ~ gsub('^.__',
"", .))) %>%  mutate(X2 = as.numeric(X2)) %>% filter(X2 >0) %>% mutate(sample =
rep(glue("{name}"), length(X2)))

}

df = do.call(rbind, df)

df = pivot_wider(df,  names_from = sample, values_from = X2, values_fill = list(0))
```

1. Loading data

```
#Meta data
mt = read_csv(
```

```
    "./metadata.csv"
    ) %>%
    column_to_rownames(
        "Name"
    ) %>%
    mutate(Time = factor(paste0("w", Time), levels = c("w0", "w1"))) %>%
    mutate(Patient = as.factor(paste0("p", Patient))) %>%
  dplyr::mutate(across(where(is.character), as.factor)) %>%
rownames_to_column(
    "id"
) %>%
 mutate(
    new_id = ifelse( grepl("D27[0-9]+", id), "Don1",
    ifelse(
        grepl("D31[0-9]+", id), "Don2",
        as.character(Time)

    )

    ),
    new_var = ifelse( grepl("D27[0-9]+", Donor), "Rec_Don1",
    ifelse(
        grepl("D31[0-9]+", Donor), "Rec_Don2",
        as.character(new_id)
    )

 ),
 donor = ifelse(id %in% c(
        "B14", "B18", "B19", "B21", "A14", "A18", "A19", "A21"
    ),
    "Donor1",
    "Donor2"),
wd = ifelse(
    donor == "Donor2" & new_id == "w0",
    "w0_don2",
    ifelse(
        donor == "Donor1" & new_id == "w0",
        "w0_don1",
        ifelse(
            donor == "Donor2" & new_id == "w1",
            "w1_don2",
            ifelse(
                donor == "Donor1" & new_id == "w1",
                "w1_don1",
                new_id
            )
        )
    )
),
new_id = ifelse(
        grepl("Don[0-9]", new_id), "Donor",
        new_id
    )
```

```
  ) %>%
 filter(
    id != "MOCK"
 ) %>%
 column_to_rownames("id")
# tidy summary of the data
mt %>%
  skimr::skim()
kable(mt)



# count
# Relative abundance:
### For the relative abundance, we take the coverage over the genome, not the raw counts.
### This implicitly normalizes for genome size. The coverage is calculated as the median
of the coverage values calculated in 1kb blocks.

# ct = arrow::read_parquet(
#     counts_file, as_tibble = TRUE
#     ) %>%
#     column_to_rownames(
#         "Sample"
#     ) %>%
#     as.matrix()

ct <- arrow::read_parquet(
        abundance_file,
        as.tibble = TRUE
        ) %>%
        column_to_rownames(
            "index"
            ) %>%
        as.matrix() %>% t() %>% floor()

sum(ct[colSums(ct) ==0])
sum(ct[rowSums(ct) ==0])

range(colSums(ct))
range(rowSums(ct))

kable(topleft(ct, c = 8))

# Taxonomy
tx = read_tsv(
    taxonomy_file
    ) %>%
    column_to_rownames(
        "user_genome"
    ) %>%
    as.matrix()
kable(tx)
```

```r
# tax_fix(tax_table(tx))

# tidying the name of the taxonomic ranks
nm = c(
    "Kingdom",
    "Phylum",
    "Class",
    "Order",
    "Family",
    "Genus",
    "Species"
    )
colnames(tx) <- nm


# Tree
tr = ape::read.tree(
    tree_file
    )
tr_arc = ape::read.tree(
    tree_arch
)



combined_tree <- bind.tree(x = tr, y = tr_arc, where = 0) %>% root( outgroup = "MAG335",
resolve.root = TRUE)  #set the grapht position of the archea to




ps = phyloseq(
    otu_table(
        ct, taxa_are_rows = TRUE
        ),
    tax_table(
        tx
    ),
    phy_tree(
        combined_tree
    ),
    sample_data(
        mt
        )
)

ps = ps %>%  tax_fix()

ps@tax_table[sample_names(ps) == "MOCK",]
ps = subset_samples(ps, sample_names(ps) != "MOCK")

# ps_rec <- subset_samples(ps, Time %in% c("w0", "w1"))
```

```
# ps_don <- subset_samples(ps, !Time %in% c("w0", "w1"))
```

2. Preprocessing

```
prevdf = apply(
    ps@otu_table,
    ifelse(taxa_are_rows(ps), 1, 2),
    function(x){
        sum(
            x>0
        )
    }
)

wdf = data.frame(
    mag_prev = prevdf,
    taxabund = taxa_sums(ps),
    tax_table(ps)
)

plyr::ddply(
    wdf,
    "Phylum",
    function(x){
        cbind(
            means = round(
                mean(x$mag_prev), 2
            ),
            sums = round(
                sum(
                    x$mag_prev
                ), 2
            )
        )
    }
) %>%
mutate(
    sanity = ifelse(
        means == sums,
        "TRUE",
        "FALSE"
    )
)
```

```
asv.filter = function(asvtab, n.samples = 1 ){
  filter.threshold <- n.samples/ncol(asvtab) *100 # In how many samples out of total
samples an ASV should have occured
  table_count <- apply(otu_table(asvtab), 2, function(x) ifelse(x>0, 1, 0)) %>%
as.data.frame()
  suspected_ASV = table_count[which((rowSums(table_count)/ncol(table_count))*100 <
filter.threshold),] %>% rownames()
```

```
  return(suspected_ASV)
}

sus_otu = asv.filter(ps@otu_table, n.samples = 2)

zeros = ps@otu_table[rowSums(ps@otu_table) == 0, ] %>% rownames()

ps = subset_taxa(ps, !taxa_names(ps) %in% zeros)

sample_data(ps) %>% data.frame() %>%  filter(wd %in% c("w0_don1", "w1_don1", "Don1")) %>%
filter(wd == "w0_don1") %>% rownames()
```

finding singletones

```
single_bust = function(phyloseq, threshold =1, binwidth = 0.01) {

#Loading necessary pkgs
  pacman::p_load(glue, tidyverse, reshape2, ggrepel, S4Vectors) # nolint
#This function requires phyloseq, tidyverse and glue packages to be loaded.
    if (sum(colSums(otu_table(phyloseq)))/ncol(otu_table(phyloseq)) == 100 ) {#making the
    relative abundance table
                    rel_abund = as(t(otu_table(phyloseq)), "matrix")
    } else if (sum(colSums(otu_table(phyloseq)))/ncol(otu_table(phyloseq)) == 1) {
                    rel_abund = as(t(otu_table(phyloseq)), "matrix")
                    } else {
                    rel_abund = as(t(apply(otu_table(phyloseq),
                    ifelse(taxa_are_rows(phyloseq), 1,2),
                    function(x) x/sum(x))), "matrix")
                    }


      names.single = apply(rel_abund, 1, function(x){ifelse(x == threshold, TRUE,
                                                    ifelse(x == sum(x), TRUE, FALSE))})
                                                    %>% reshape2::melt() %>%
                                                    filter(value == TRUE) %>%
                                                    dplyr::select(2) %>%
                                                    pull%>% as.vector()

    if (length(names.single) == 0 ) {
          print(glue("WOW! {length(names.single)} singletones detected in this dataset"))
                qplot.noSing = qplot(rel_abund, geom = "histogram", binwidth = binwidth,

                 show.legend = F, main = "Frequency count of relative abundance, no
                 singletones detected") +
          xlab ("Relative abundance in samples") +
                   ylab("Frequency") + theme_bw()

    return(structure(list(qplot.noSing)))

                        } else {
```

```
single.ASV = rel_abund[rownames(rel_abund) %in% names.single,]
single.ASV[single.ASV == 0] <- NA # A separate dataset for annotation of singletones on
the barplot

         qplot.withSing = qplot(rel_abund, geom = "histogram", binwidth = binwidth,
        main = "Frequency count of relative abundance with singletones") +
        geom_bar(aes(single.ASV), fill = "red",  color = NA, width = binwidth)+
                       xlab ("Relative abundance in samples") + ylab("Frequency") +
                       geom_label_repel(aes(x = 1, y =length(rel_abund)/5),
                       label.padding =  unit(0.55, "lines"),
                       label = glue("{length(names.single)}\n Singletones"), color =
                       "black") +
           theme_bw()

                       qplot.rmSing = qplot(rel_abund[!rownames(rel_abund) %in%
names.single, ], geom = "histogram",
                       binwidth = binwidth, main = "Frequency count of relative abundance
                       without singletones") +
                       xlab ("Relative abundance in samples") + ylab("Frequency")+
                       theme_bw()

                       print(glue('Oh no..! {length(names.single)} singletones detected
                       in the dataset'))
                       return(structure(list(qplot.withSing, qplot.rmSing,
                       unlist(names.single))) )

                        }


        }

single.test = single_bust(phyloseq =ps, threshold = 1, binwidth = 0.01)
singletones = single.test[[3]] #here you can extract the names of the singletones


ps@otu_table[taxa_names(ps) %in% singletones,]

single.test[[1]]#to show the plot with singletones
single.test[[2]]

# I skipped this part
```

#save all data

```
write.table(x = otu_table(ps) %>% data.frame() %>% rownames_to_column("genome"), file =
"./count.tsv", sep = "\t", row.names = F)

write.table(x = tax_table(ps) %>% data.frame() %>% rownames_to_column("genome"), file =
"./taxa.tsv", sep = "\t", row.names = F)

write.table(x = sample_data(ps) %>% data.frame() %>% rownames_to_column("id"), file =
"./mt.tsv", sep = "\t", row.names = F)
```

3. Alpha diversity

```
ps_rel  = transform_sample_counts(ps, function(x){x/sum(x)*100})


Chao = estimate_richness(ps, split = TRUE, measures = "Chao1")$Chao1
shan =  estimate_richness(ps_rel, split = TRUE, measures = "Shannon")

sample_data(ps_rel) <- data.frame(
    sample_data(ps_rel),
    Chao1 = Chao,
    Shannon = shan
)
```

```
library(ggpubr, verbose = FALSE)
library(reshape2, verbose = FALSE)

alpha_df = sample_data(ps_rel)  %>% data.frame()
alpha_df = alpha_df %>% mutate(
    Patient = ifelse( new_var == "Don1", "pdon1",
    ifelse(
        new_var == "Don2", "pdon2",
        as.character(Patient)
)
    )
)


long_mtdat <- melt(alpha_df )
long_mtdat<- long_mtdat[long_mtdat$variable %in% c("Chao1", "Shannon"),]

my_comp = list(
    c("Don1", "Don2"),
    c("Don1", "w0_don1"),
    c("Don2", "w0_don2"),
    c("w1_don1", "w0_don1"),
    c("w1_don2", "w0_don2"),
    c("w1_don2", "Don2",
    c("w1_don1", "Don1"))

    )
# my_comp = list(
#     c("w1", "w0"),
#     c("Donor", "w0"),
#     c("Donor", "w1")
#     )
```

```
long_mtdat$variable <- factor(long_mtdat$variable , levels =  c("Chao1", "Shannon"))


 ggplot(long_mtdat, aes(x = wd, y = value)) +
  geom_violin(aes(fill = wd), trim = F, alpha = 0.55) +
  stat_compare_means(
    paired = FALSE,
    comparison = my_comp,
    method = "t.test",
    label = "p.signif"
    ) +
  geom_boxplot(width = 0.10, outlier.colour = NA) +
  geom_jitter(color = "black", alpha = 0.5, size = 2)+
  facet_wrap(~variable, scales = "free_y") +
  theme_bw()+
  scale_fill_manual(values = c("#d0e0e6", "#2b4557", "#f079d2", "#79def0", "#f07979",
  "#888788")) +
  labs(
    fill = "Groups",
    x = "",
    y = "Alpha diversity"
  ) +
  ggtitle(
    "Alpha diversity of taxa",
    "Donors compared to anorexic patients in pre- and post-treatment weeks.\nAn unpaired
    student t-test was used to derive statistics. Comparisions are labled with adjusted
    p-value"
  ) +
  theme(
    axis.text.x = element_text(size = 10, family = "Arial", face = "bold"),
    axis.text.y = element_text(size = 10, family = "Arial", face = "bold"),
    axis.title.y = element_text(size = 15, family = "Arial", face = "bold"),
    strip.text = element_text(face = "bold", size = 15, color = "white"),
    strip.background = element_rect(fill = "#679187")
  )

ggsave("./plots/alpha_treatment.jpeg", dpi = 500, width  = 15, height = 8)

## finding responders
long_mtdat$new_var <- factor(long_mtdat$new_var, levels = c("Don1", "Don2", "w0",
"Rec_Don1", "Rec_Don2"))

 ggplot(long_mtdat, aes(x = Patient, y = value)) +
  geom_col(aes(fill = as.factor(new_var)), trim = F, alpha = 0.9, position =
  position_dodge(width = 0.9)) +
  facet_wrap(~ variable, scales = "free") +
  stat_compare_means(
    paired = FALSE,
    comparison = my_comp,
    method = "t.test",
    label = "p.signif"
    ) +
  facet_wrap(~variable, scales = "free_y") +
```

```
  theme_bw()+
  scale_fill_manual(values = c("#4dbbe4", "#c7a719", "#e9c0df", "#3a9673", "#949394")) +
  labs(
    fill = "Treatment",
    x = "Tested individuals",
    y = "Alpha diversity"
  ) +
  ggtitle(
    "Alpha diversity of taxa",
    "All inidividuals"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1)
  )


ggsave("./plots/alpha_individuals.jpeg", dpi = 500, width  = 12, height = 8)


#Responders
my_comp = list(
    c("w0", "w1")
    )


long_mtdat %>% filter(! is.na(Responder ), !is.na(Time)) %>%

ggplot( aes(x = Time, y = value)) +
  geom_violin(aes(fill = Time), trim = F, alpha = 0.55) +
  stat_compare_means(
    paired = FALSE,
    comparison = my_comp,
    method = "t.test",
    label = "p.signif"
    ) +
  geom_boxplot(width = 0.10) +
  geom_jitter(color = "black", alpha = 0.5)+
  facet_grid(variable ~ Responder, scales = "free_y") +
  theme_bw()+
  scale_fill_manual(values = c("#d0e0e6", "#574f2b", "#f079d2")) +
  labs(
    fill = "Treatment",
    x = "",
    y = "Alpha diversity"
  ) +
  ggtitle(
    "Alpha diversity of taxa",
    "Donors compared to anorexic patients in pre- and post-treatment weeks.\nAn unpaired
    student t-test was used to derive statistics"
  ) +
  stat_compare_means(
    paired = FALSE,
    comparison = my_comp,
```

```
    method = "t.test",
    label = "p.signif"
    )
```

4. Beta diversity

Ordinations: based on dbrda model

```
library(vegan)
library(permute)

ps_spec = gloomer(ps, taxa_level = "Species")


ps_rel = transform_sample_counts(ps_spec, function(x){x/sum(x) *100})

ps_log = transform_sample_counts(ps_rel, function(x){  log( 1 +  x)})

# Calculate bray curtis dissimilarity coefficients
bray_log = phyloseq::distance(
    ps_log,
    method = "bray"
)


df = data.frame(sample_data(ps_log))

xtabs(~ donor + wd, df)

# Creating permutational plan
df$new_id <- ifelse(
    grepl("Don", df$new_id),
    "Donor",
    as.character(df$new_id )
) %>% as.factor()

h = with(
    data = df,
    how(blocks = donor,
    nperm = 9999)
)


# Checking the variance homogeniety around group means
bray.disp = vegan::betadisper(
    bray_log,
    group = df$wd,
    type = "centroid",

)
```

```
perm_test = permutest(
    bray.disp,
    permutation = h,
    pairwise = T
)

pval = perm_test$tab$'Pr(>F)'[[1]]
eigval = bray.disp$eig

jpeg(
    "./plots/beta_disper.jpeg",
    res = 500,
    units = "cm",
    width = 30,
    height = 25
)
plot(
    bray.disp,
    main = "Dispersion of variance around the means of the groups.\nBray-Curtis
    dissimilairty coefficients | Log-transformed",
    bty = "n",
    sub = NULL,
    xlab = sprintf("PCo1 [%s%%]", round(eigval/sum(eigval)*100,1)[1]),
    ylab = sprintf("PCo2 [%s%%]", round(eigval/sum(eigval)*100,2)[2])
    ); text(
        x = -0.5,
        y = -0.15,
        glue("p = {pval}"),
        col = "red"
    ); abline(v = 0, h = 0, lty = 2, col = "#b3c4c5")
dev.off()
```

dbrda: determining statistics for relationship between the group variable and the configuration of the distance metrix on MDS ordination plot

```
df$wd <- as.factor(df$wd)
df$donor <- as.factor(df$donor)

# Fitting the model
(
bray_db = dbrda(
    bray_log ~ wd + Condition(donor),
    permutation = h,
    data = df,
)
)
```

```r
# Test for groups
permutest(
    x = bray_db,
    by = "onedf",
    permutations = h
)

# Model omnibus test
permutest(
    x = bray_db,
    by = "terms",
    permutations = h
)

# anova(bray_db, by = "terms", permutations = h)


# Extracting sample scores
s_score = vegan::scores(bray_db, display = "sites", choices = c(1,2)) %>% as.data.frame()
s_score <- cbind(as.data.frame(s_score), group = df$wd)

# extracting mean scores
c_score = vegan::scores(bray_db, display = "cn") %>% as.data.frame()
rownames(c_score) <- levels(
    df$wd
)

# Or
c_score <- aggregate(cbind(dbRDA1, dbRDA2) ~ group, data = s_score, FUN = mean)


# the same as aggregate
c_score = s_score %>% group_by(group) %>% summarise_all(mean)

# Making segments
segs <- merge(s_score, setNames(c_score, c("group", "dbrda1", "dbrda2")),
by = 'group', sort = FALSE)


# Eigenvalues
eigval = bray_db$CCA$eig
interia_total = bray_db$tot.chi

x_fit = round(
    eigval[[1]]/sum(eigval)*100,
    1
)

y_fit = round(
    eigval[[2]]/sum(eigval)*100,
    1
)
```

```
x_tot = round(
    eigval[[1]]/interia_total*100,
    1
)

y_tot = round(
    eigval[[2]]/interia_total*100,
    1
)

#reportint statistics

stat_annot = "Omnibus dbRDA statistics of the model:\nP < 0.01\nPsudo-F = 3.09\nR2 =
21.0"


t = s_score %>% ggplot(
    aes(dbRDA1, dbRDA2)
)  +
geom_segment(
    data = segs,
    aes(
        xend = dbrda1,
        yend = dbrda2
        ),
        color = "#666666",
        lty = 1,
        alpha = 0.25
    ) +
geom_point(
    data = segs,
    aes(
        x = dbrda1,
        y = dbrda2
    ),
    pch = 21,
    stroke = 0.5,
    fill = "#ffffff",
    color = "#7c7c7c",
    size = 4
) +
geom_point(
    pch = 21,
    aes(
        fill= df$wd
    ),
    color = "white",
    stroke = 2,
    size = 10,
    alpha = 0.95
)+
theme_bw() +
geom_hline(
```

```
    yintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        0.5
        )
        ) +
geom_vline(
    xintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        alpha = 0.5
        )
        ) +
ggtitle(
    "PCoA plot of treatment effect on log-transfomred Bray-Curtis dissimilarity",
    "A distance-based redundancy analaysis was performed to analyse the variance between
    groups.\nVariance from individual participants is still a cofounding factor."
) +
scale_fill_manual(
    values = c(
    Don1 = "#a11818dd",
    Don2 = "#2b4557",
    w0_don1 = "#87236e",
    w0_don2 = "#79def0",
    w1_don1 = "#f07979",
    w1_don2 = "#888788"
    )


) +
labs(
    x = glue(
        "dbRDA1 [{x_fit} % of fitted and {x_tot} % of total variance ]"
    ),
    y = glue(
        "dbRDA2 [{y_fit} % of fitted and {y_tot} % of total variance ]"

    ),
    fill = "Sample points"
) +
# stat_ellipse(aes(
#     group = group,
# ),
#     geom = "polygon",
#     segments = 20,
#     alpha = 0.15,
#     level = 0.85,
#     type = "t",
#     show.legend = F,
#     fill  = "#f71414",
#     lty = 2
```

```
# ) +
geom_text(
    aes(
        x = 0.5,
        y  = 2.5
        ),
        label = stat_annot,
        color = "#0f0f34",
        hjust = 0
) +
 geom_text_repel(
    aes(
        label = glue("{df$Patient}")
        ),
    size = 4,
    box.padding = unit( 0.4, units ="cm"),
    point.padding = unit(0.4, "cm"),
    arrow = arrow(length = unit(0.009, "npc"))
)

#  coord_fixed( )

ggsave(
    "./plots/dbrda_pcoa_bray_withPatientVariance.jpeg",
    width = 15,
    height = 9,
    dpi = 300
)

library(plotly)
library(processx)

pl= plot_ly(z = ~ volcano) %>% add_surface()
orca(pl, "test.svg")
# Extracting rank-based changes from week 0 to week 1 in recipients
df1 = s_score%>% mutate(gr = ifelse(
    grepl("Rec_", group), "w1", as.character(group)
))  %>% filter(gr == "w0")

df2 = s_score%>% mutate(gr = ifelse(
    grepl("Rec_", group), "w1", as.character(group)
))  %>% filter(gr == "w1")


rank_dif = data.frame(
    dif_x = abs((df1 %>% select(1))- (df2 %>% select(1))),
    dif_y = abs((df1 %>% select(2))- (df2 %>% select(2)))

) %>%
 mutate(sums = dbRDA1 + dbRDA2) %>%
 setNames(c("dif_x", "dif_y", "sums")) %>%
 mutate(rank = min_rank(sums)) %>%
```

```
 arrange(desc(rank))

write.table(x = rank_dif, file = "./plots/ranks.tsv", sep = "\t")
```

5. Barchart

Color picker: a function for picking distinct colors

```
## Color brewere
disc_color = function(n, method = "brewer", seed = 1990){
    if(method == "rainbow"){
        return(rainbow(n))
    } else if(method == "brewer"){


        if(n<=11){
            cols = RColorBrewer::brewer.pal(n = 11, name = "Set3")[1:n]



        } else if( n>11 && n <=74){
            sets_name  =  RColorBrewer::brewer.pal.info %>% data.frame() %>%
rownames_to_column("names") %>% filter(category == "qual") %>% select(names) %>% pull
            sets_num = RColorBrewer::brewer.pal.info %>% data.frame() %>%
rownames_to_column("names") %>% filter(category == "qual") %>% select(maxcolors) %>% pull


            pooled = list()
            for(i in 1:length(sets_name)){
                pooled[[i]] <- RColorBrewer::brewer.pal(n = as.numeric(sets_num[i]), name
= sets_name[i])

            }
            cols = do.call(c, pooled)[1:n]

        } else if( n > 74){
            sets_name  =  RColorBrewer::brewer.pal.info %>% data.frame() %>%
rownames_to_column("names") %>% select(names) %>% pull
            sets_num = RColorBrewer::brewer.pal.info %>% data.frame() %>%
rownames_to_column("names")  %>% select(maxcolors) %>% pull
            pooled = list()
            for(i in 1:length(sets_name)){
                pooled[[i]] <- RColorBrewer::brewer.pal(n = as.numeric(sets_num[i]), name
= sets_name[i])

            }
            cols = do.call(c, pooled)
            set.seed(seed)
            cols = cols[shuffle(cols)][1:n]
```

```
        } else if(!is.numeric(n)){
            stop(
                "Please use a valid number!"
            )
        }
    } else if (method == "viridis"){
        cols = viridis::viridis(n = n, option = "D")
    }
return(cols)
}
```

```
ps_spec = gloomer(ps, taxa_level = "Species")
ps_rel = transform_sample_counts(ps_spec, function(x){
    x/sum(x) *100
})

set.seed(10)

ps_rel %>%
psmelt() %>%
dplyr::select(
    Species,
    Sample,
    Abundance,
    wd,
    Order
    ) %>%
    group_by(
        Species,
        Order,
        Sample,
        Abundance,
        wd
    ) %>%
    reframe(
        mean = mean(Abundance)
    ) %>%
    ggplot() +
    geom_col(
        aes(
            y = mean,
            x = Sample,
            fill = Order
        ),
        show.legend = T
    ) +
    facet_wrap(
        ~ factor(wd, levels = c(
            "Don1",
            "Don2",
            "w0_don1",
            "w0_don2",
            "w1_don1",
```

```
            "w1_don2"

        )
        ),
        scales = "free",
        ncol = 2,
        dir = "h"


    ) +
    theme_bw() +
    labs(
        x = "Individuals",
        y = "Relative abundance, %"
    )  +
    scale_fill_manual(
        values = sample(colors(), size = 32)
        ) +
    theme(
       axis.text.x = element_text(
        angle = 0
       )
    ) + guides(fill = guide_legend(ncol = 1))+
    coord_flip()

ggsave(
    "./plots/stack_bplot_individuals.jpeg",
    width = 15,
    height = 10,
    dpi = 500
    )



# stacked variable
cols = disc_color(n = length(unique(tax_table(ps_rel)[,4])) + 10, method = "brewer")

# hist(1:length(cols), col = cols,  breaks = length(cols), ylim = c(0,0.5))

ps_rel %>%
psmelt() %>%
select(
    Species,
    Abundance,
    wd,
    Order
    ) %>%
    group_by(
        Species,
        Order,
        wd
    ) %>%
    mutate(
```

```
        wd = factor(
            wd,
            levels = c(
                "Don1",
                "w0_don1",
                "w1_don1",
                "Don2",
                "w0_don2",
                "w1_don2"
            )
        )
    )%>%
    reframe(
        mean = mean(Abundance)
    ) %>%
    ggplot() +
    geom_col(
        aes(
            y = mean,
            x =wd,
            fill = Order
        ),
        show.legend = T
    ) +
    theme_bw() +
    labs(
        x = "Individuals",
        y = "Relative abundance, %"
    )  +
    scale_fill_manual(
        values = cols[1:length(cols)]
        ) +
    theme(
       axis.text.x = element_text(
        angle = 0
       )
    ) +
    coord_flip() +
    guides(
        fill = guide_legend(ncol = 2)
    ) +
    theme(
        axis.text.x = element_text(size = 12, face = "bold"),
        axis.text.y = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size = 15, face = "bold")
    )

ggsave(
    "./plots/stack_bplot_gropus.jpeg",
    width = 15,
    height = 8,
    dpi = 500
    )
```

6. Differential abundance analysis

Functions for deseq

```
library(DESeq2)

# A function to help deseq

deseq_fit = function(
    ps,
    design,
    fit.type = "parameteric",
    test = "Wald",
    taxa_level = "Species"
){

    ps = gloomer(ps, taxa_level = taxa_level, NArm = TRUE)

    #Converting phyloseq to deseq2
    design = as.formula(glue("~ {design}"))
    ds = phyloseq_to_deseq2(ps, design = design)

    # calculating the geometric means
    gm_mean = function(x, na.rm = TRUE){
        exp(
            sum(
                log(x[x>0]),
                na.rm = na.rm
            )/length(x)
        )
    }

    geo_mean = apply(counts(ds), 1, gm_mean)
    ds = estimateSizeFactors(ds, geoMeans = geo_mean)

    #Fitting the model
    ds = DESeq(
        ds, test = "Wald",
        fitType = "parametric"
    )

    return(ds)

}

# Function for visualization



deseq_vis = function(
    contrast = NULL,
    alpha = 0.05,
```

```
    deseq_obj,
    ps,
    colored = "Phylum",
    taxa_level = "Species",
    plot_type = "water_fall",
    target_variable = NULL,
    break_fraction = 3,
    x_lim_coef = 2
    ){


        df = DESeq2::results(
            object=deseq_obj,
            contrast = contrast,

        ) %>% data.frame() %>% filter(padj <= alpha)

        if(dim(df)[1] == 0){
            stop(
                "This combination of variables doesn't return anything, please try other
                combinations!"
            )
        }
        # ps = gloomer(ps, taxa_level = taxa_level)
        df = data.frame(df, tax_table(ps)[taxa_names(ps) %in% rownames(df),])

        df = df %>% rownames_to_column('taxa')

# Condition for the x axis limit
Condition = range(df$log2FoldChange)[1] < 0 & range(df$log2FoldChange)[2] > 0
lims = if(Condition){
                round(c(min(
                    df$log2FoldChange
                    ) - abs(x_lim_coef *log10(
                        max(
                            df$lfcSE
                            )
                        )),
                        max(
                            df$log2FoldChange
                            )+ abs(x_lim_coef *log10(
                                max(
                                    df$lfcSE
                                    )
                                ))
            ))
            } else {
                if(range(df$log2FoldChange)[2] > 0){
                    c(0, ceiling(range(df$log2FoldChange)[2] + abs(x_lim_coef *
                    log10(max(df$lfcSE)))))
                }else if(range(df$log2FoldChange)[2] < 0){
                    c(ceiling(range(df$log2FoldChange)[1] - abs(x_lim_coef *
                    log10(max(df$lfcSE)))), 0)
```

```
        } else {
            stop("Log2FodChange cannot be zero range!")
        }



    }
cls = c(disc_color(n = length(unique(df[[colored]])), method = "brewer"))

df$cols <- df[[colored]]

p = df %>%
arrange(desc(log2FoldChange)) %>%
mutate( #setting factor level for sorting
    taxa = factor(taxa, levels = unique(taxa), label = unique(taxa)),
    max = log2FoldChange + log10( lfcSE),
    min = log2FoldChange - log10( lfcSE)

) %>%
ggplot(aes(
    x = log2FoldChange,
    y = taxa,
    fill = cols
)) +
labs(
    y = glue("{taxa_level}"),
    fill = colored
) +
geom_col(
    #width = 0.1
) +
scale_fill_manual(
    values = cls
) +
theme_minimal() +
theme(
    panel.grid.minor = element_blank(),
    axis.text.y = element_text(size = 8)
) +
ggtitle(
   glue("Differential abundance of {taxa_level} in {contrast[2]} vs.
   {contrast[3]}"),
    glue("E.g., increased one taxa on the plot represents higher abundance of
    {contrast[2]} vs. {contrast[3]}")
) +
geom_vline(
    xintercept = 0, #ifelse(range(df$log2FoldChange)[1] < 0 &
    range(df$log2FoldChange)[2] > 0 , 0, ceiling(min(abs(df$log2FoldChange)))),
    alpha = 0.25
) +
geom_errorbar(
    aes(
        xmin = min,
```

```
                xmax= max
            ),
            width = 0.2,
            linewidth = 0.15
        ) +
        geom_point(
            pch = 21,
            stroke = 0.5,
            show.legend =  F
        ) +
        scale_x_continuous(
            limits = lims,
           n.breaks = floor(length(seq(floor(min(df$log2FoldChange)), by = 1,
           ceiling(max(df$log2FoldChange)))) / break_fraction)
)

    return(p)
}
```

```
library(DESeq2)

model.matrix(~ wd, data.frame(sample_data(ps_spec)))


ps_spec <- gloomer(ps, taxa_level = "Species", TRUE)
spec_rel <- transform_sample_counts(ps_spec, function(x){x/sum(x)*100})

sp_ds = deseq_fit(
    ps = ps_spec,
    design = "wd",
    taxa_level = "Species"
)


resultsNames(sp_ds)

#donor 2
df = results(sp_ds, contrast =  c('wd', "w1_don2", "w0_don2")) %>% data.frame() %>%
filter(!is.na(padj)) %>% rownames_to_column('Species')

df = cbind(df, tax_table(ps_spec)[rownames(tax_table(ps_spec)) %in% df$Species , c(1:6)])

dt <- sample_data(spec_rel) %>% data.frame()

for(i in seq_along(colnames(dt))) {

  if (is.character(dt[[i]])) {
    dt[[i]] <- factor(dt[[i]], levels = unique(dt[[i]]))
  }
}

tb = otu_table(spec_rel)[,sample_data(spec_rel)$wd %in% c("w1_don2", "w0_don2", "Don2")]
```

```
df1 = tb %>% data.frame() %>% rownames_to_column("Species") %>% pivot_longer(cols =
-Species, names_to = "Sample", values_to = 'relabund')


df1 = left_join(df1, dt %>% rownames_to_column('Sample'), by = 'Sample')  %>% filter(wd
%in% c("w0_don2", "w1_don2", "Don2"))

df1 =  left_join(df1, tax_table(ps_spec)[rownames(tax_table(ps_spec)) %in% df1$Species ,
c(1:6)] %>% data.frame() %>% rownames_to_column("Species"), by = "Species")

df1 = df1 %>% group_by(Species, Family, wd) %>% summarise(rel = mean(relabund)) %>%
ungroup() %>% pivot_wider(id_cols = c(Species, Family), names_from = wd, values_from =
rel) %>% mutate(col_code = ifelse(w0_don2 == 0 & Don2 > 0, "DonYes|RecNo",
ifelse(w0_don2 > 0 & Don2 > 0, "DonYes|RecYes", ifelse(w0_don2 > 0 & Don2 == 0,
"DonNo|RecYes", ifelse(w0_don2 == 0 & Don2 == 0, "DonNo|RecNo", NA)))))

df_w0 = df1 %>% select(1,6,3)


df_w1 = df1 %>% select(1,6,4)

df_don =  df1  %>% select(1,6,5)

d_w0 <-left_join(df_w0, df, by = 'Species') %>% mutate(index = "W0", rel_abund = w0_don2,
w0_don2 = NULL)
d_w1 <-left_join(df_w1, df, by = 'Species')  %>% mutate(index = "W1", rel_abund =
w1_don2, w1_don2=NULL)
d_d2 = left_join(df_don, df, by = 'Species')  %>% mutate(index = "Don2", rel_abund =
Don2, Don2=NULL) %>% mutate(shp = "Donor2")

df_d2<- rbind(d_w0, d_w1, d_d2 %>% mutate(shp = NULL))  %>% mutate(don = "Donor2")


#for donor1
d_d1 <- results(sp_ds, contrast =  c('wd', "w1_don1", "w0_don1")) %>% data.frame() %>%
filter(!is.na(padj)) %>% rownames_to_column('Species')

d_d1 = cbind(d_d1, tax_table(ps_spec)[rownames(tax_table(ps_spec)) %in% d_d1$Species ,
c(1:6)])

tb_d1 = otu_table(spec_rel)[,sample_data(spec_rel)$wd %in% c("w1_don1", "w0_don1",
"Don1")]

df1_d1 = tb_d1 %>% data.frame() %>% rownames_to_column("Species") %>% pivot_longer(cols =
-Species, names_to = "Sample", values_to = 'relabund')



df1_d1 = left_join(df1_d1, dt %>% rownames_to_column('Sample'), by = 'Sample')  %>%
filter(wd %in% c("w0_don1", "w1_don1", "Don1"))

df1_d1 =  left_join(df1_d1, tax_table(ps_spec)[rownames(tax_table(ps_spec)) %in%
df1_d1$Species , c(1:6)] %>% data.frame() %>% rownames_to_column("Species"), by =
"Species")
```

```
df1_d1 = df1_d1 %>% group_by(Species, Family, wd) %>% summarise(rel = mean(relabund),
.groups = 'drop') %>% ungroup() %>% pivot_wider(id_cols = c(Species, Family), names_from
= wd, values_from = rel) %>% mutate(col_code = ifelse(w0_don1 == 0 & Don1 > 0,
"DonYes|RecNo",
ifelse(w0_don1 > 0 & Don1 > 0, "DonYes|RecYes", ifelse(w0_don1 > 0 & Don1 == 0,
"DonNo|RecYes", ifelse(w0_don1 == 0 & Don1 == 0, "DonNo|RecNo", NA)))))

df_w0_d1 = df1_d1 %>% select(1,6,3)


df_w1_d1 = df1_d1 %>% select(1,6,4)

df_don1 =  df1_d1  %>% select(1,6,5)

d_w0_d1 <-left_join(df_w0_d1, d_d1, by = 'Species') %>% mutate(index = "W0", rel_abund =
w0_don1, w0_don1 = NULL) %>% filter(!is.na(baseMean))
d_w1_d1 <-left_join(df_w1_d1, d_d1, by = 'Species')  %>% mutate(index = "W1", rel_abund =
w1_don1, w1_don1=NULL) %>% filter(!is.na(baseMean))
d_don1 = left_join(df_don1, d_d1, by = 'Species')  %>% mutate(index = "Don1", rel_abund =
Don1, Don1 = NULL) %>% mutate(shp = "Donor1")  %>% filter(!is.na(baseMean))



df_d1 <- rbind(d_w0_d1, d_w1_d1, d_don1 %>% mutate(shp = NULL)) %>% mutate(don =
"Donor1")

merged_df <- rbind(df_d1, df_d2)


 ggplot() +
 geom_point(data  =  merged_df %>% filter(padj > 0.05), aes( x = rel_abund, y =
 log2FoldChange), color = '#9a014e', alpha = 0.05) +
 geom_point(data  =  d_d2 %>% filter(padj <= 0.05), aes( x = rel_abund, y =
 log2FoldChange, color = col_code, shape = shp), size = 5) +

 geom_point(data  =  d_don1 %>% filter(padj <= 0.05), aes( x = rel_abund, y =
 log2FoldChange, color = col_code, shape = shp), size = 5, alpha = 0.8) +
geom_label_repel(data  =  merged_df %>% filter(padj <= 0.05, index %in% c('Don1',
'Don2')), aes(label = Species, x = rel_abund, y = log2FoldChange, group = don),
max.overlaps = 100, size = 2) +
scale_color_manual(values = c('#ff8c00', '#43b2b2')) + theme_bw() +
geom_hline(yintercept = 0, alpha = 0.6, lty  = 2, color = 'red') +
ggtitle(
    "Dif abund of different speices from w0 to w1",
    "X axis is the relative abundance of taxa in Donors"
) +
labs(
    fill = "Presence/absence",
    y = 'Log2FoldChange, w1 vs. w0',
    x = 'Relative abundance of taxa in Donors'
)
```

```
ggsave("./plots/diffabund_donor2_w0_w1_Spec(D1_D2).jpeg", device = 'jpeg', width = 20,
height = 12, dpi = 700)
```

DESeq plots

```
#plots
conts = list(
    c('wd', "Don2", "Don1"),
    c('wd', "w0_don1", "w1_don1"),
    c('wd', "w0_don2", "w1_don2"),
    c('wd', "Don1", "w0_don1"),
    c('wd', "Don2", "w0_don2"),
   c("wd", "Don2", "w1_don2")
)

pix = list()
tx_lev = 'Species'

ps_spec <- gloomer(ps, taxa_level = tx_lev, TRUE)
spec_rel <- transform_sample_counts(ps_spec, function(x){x/sum(x)*100})



sp_ds = deseq_fit(
    ps = ps_spec,
    design = "wd",
    taxa_level = tx_lev
)




for(i in 1:length(conts)){
    p = deseq_vis(
    contrast = conts[[i]],
    deseq_obj = sp_ds,
    ps = ps_spec,
    taxa_level = tx_lev,
    colored = "Phylum",
    break_fraction =  2

    )

    d1 <- sp_ds %>% results(contrast = conts[[i]]) %>% data.frame() %>% filter(padj
<=0.05)
    d2 <- tax_table(ps_spec)[rownames(tax_table(ps_spec)) %in% rownames(d1),] %>%
data.frame
```

```
    d_j <- left_join(rownames_to_column(d1, tx_lev), d2, by = tx_lev)

    pix[[i]] <- p
    # ggsave(plot = p, filename =
    glue("./plots/difabund_{conts[[i]][2]}_vs_{conts[[i]][3]}_{tx_lev}.jpeg"), width =
    15, height = 8, dpi = 500)

    write.table(d_j, file =
    glue('./plots/difabund_{conts[[i]][2]}_vs_{conts[[i]][3]}_{tx_lev}.tsv'), sep = '\t',
    row.names = F)
}
```

Chord diagram of diff abund taxa

```
library(hrbrthemes)
library(circlize)
library(chorddiag)
library(networkD3)
library(shadowtext)



ps_rel <- transform_sample_counts(ps_spec, function(x){x/sum(x)})

ots <- otu_table(ps_spec) %>% data.frame

ots <- ots[rownames(ots) %in% d_j$Species, ]
ots = ots %>% rownames_to_column('Species') %>% pivot_longer(cols = -Species, names_to =
'Samples', values_to = 'value')



# With networkD3, connection must be provided using id, not using real name like in the
links dataframe.. So we need to reformat it.
ots$IDsource <- match(ots$Species, nodes$name)-1
ots$IDtarget <- match(ots$Samples, nodes$name)-1

nodes <- data.frame(
  name=c(as.character(ots$Species), as.character(ots$Samples)) %>%
    unique()
  )
my_color <- 'd3.scaleOrdinal() .domain(["a", "b"]) .range(["#69b3a2", "steelblue"])'
# Load energy projection data
URL <- "https://cdn.rawgit.com/christophergandrud/networkD3/master/JSONdata/energy.json"
Energy <- jsonlite::fromJSON(URL)


p = sankeyNetwork(Links = ots$Species, Nodes = nodes$Samples,
                      Source = "IDsource", Target = "IDtarget",
                      Value = "value", NodeID = "name",
```

```
                    sinksRight=FALSE, colourScale = my_color)
saveNetwork(p, file = "sankey_plot.html")



circos.clear()
circos.par(start.degree = 90, gap.degree = 4, track.margin = c(-0.1,0.1),
points.overflow.warning = F)

par(mar = rep(0, 4))
mycolor <- viridis(10, alpha = 1, begin = 0, end = 1, option = "D")
mycolor <- mycolor[sample(1:10)]


chordDiagram(
    x = ots %>% select(-c(4,5)) ,
    # grid.color = mycolor,
     transparency = 0.25,
  directional = 1,
  direction.type = c( "diffHeight"),
  diffHeight  = -0.04,
#   annotationTrack = "grid",
  annotationTrackHeight = c(0.05, 0.1),
  link.arr.type = "big.arrow",
  link.sort = TRUE,
  link.largest.ontop = TRUE
)

circos.trackPlotRegion(
  track.index = 1,
  bg.border = NA,
  panel.fun = function(x, y) {

    xlim = get.cell.meta.data("xlim")
    sector.index = get.cell.meta.data("sector.index")

    # Add names to the sector.
    circos.text(
      x = mean(xlim),
      y = 3.2,
      labels = sector.index,
      facing = "downward",
      cex = 1
      )

    # Add graduation on axis
    circos.axis(
      h = "top",
      major.at = seq(from = 0, to = xlim[2], by = ifelse(test = xlim[2]>10, yes = 2, no =
      1)),
      minor.ticks = 1,
      major.tick.percentage = 0.5,
      labels.niceFacing = FALSE)
```

```
  }
)

circos.savet
```

Species upset plot

```
# Load required libraries
library(UpSetR)
library(dplyr)


df <-  df1 #%>% group_by(Species, Family, wd) %>% summarise(rel = mean(relabund)) %>%
ungroup() %>% pivot_wider(id_cols = c(Species, Family), names_from = wd, values_from =
rel)


df <- df %>%
  mutate(
    w0_present = ifelse(w0_don2 > 0, 1, 0),
    w1_present = ifelse(w1_don2 > 0, 1, 0),
    Don2_present = ifelse(Don2 > 0, 1, 0)
  )


upset_data <- df  %>% dplyr::select( w0_present, w1_present, Don2_present) %>%
as.data.frame

# Create the UpSet plot
jpeg('./plots/upset_plot_don2_w0vsw1.jpeg', units = 'cm', height = 15, width = 20, res =
1000)
UpSetR::upset(upset_data, sets = c("w0_present", "w1_present", "Don2_present"), order.by
= "freq")
dev.off()



df %>% filter(Don2_present ==1 & w1_present == 1 & w0_present == 0) %>% pull(Species)
```

Phylogenetic tree

```
lays = c('rectangular', 'dendrogram', 'slanted', 'ellipse',
          'roundrect', 'fan', 'circular', 'inward_circular', 'radial',
          'equal_angle', 'daylight', 'ape')

set.seed(0)
ggtree(
    ps_spec,
    layout = "circular",
```

```
    open.angle = 0.5,
    branch.length = "none",
   # size = 0.5,
    aes(
        color = Phylum,

        )
    ) +
geom_tiplab(
    aes(
        label = Species,
        #color = Kingdom
        ),
        size = 2,
        check.overlap = F,
        offset = 0.3,
        color = "black"
        ) +
geom_point(
    size = 1,
    pch = 21,
    stroke = .2,
    fill = "deepskyblue",
    color= "white",
    alpha = 0.5
    ) +
scale_color_manual(
    values = sample(
        colors(), size = length(tax_table(ps)[,2]), FALSE),
        breaks = unique(tax_table(ps)[,2])
        )

if(!exists("./plots")){
    dir.create("./plots")
}

ggsave("./plots/tree.jpeg", device = "jpeg", width = 20, height = 20, dpi = 500)
```

7. Functional annotaions of genomes

Kegg modules produced by Dram

```
kegg_modules %>% filter(step_coverage >= 0.8, !is.na(step_coverage)) %>%
dplyr::select(1,3,6) %>% arrange(desc(step_coverage)) %>% mutate(module_name
=factor(module_name, levels  = rev(unique(module_name)))) %>% ggplot(aes(y = module_name,
x  = genome, fill = step_coverage)) + geom_tile(
 color = 'black'
) + scale_fill_viridis_c()  + theme_minimal() + theme(axis.text.x = element_text(angle =
45, hjust = 1))
```

```
ggsave('./plots/modules/plots/all_modules_all_genomes.jpeg', dpi = 500, limitsize = F,
width = 80, height = 15)
```

```
kegg_modules <- read_tsv(keggmodules_file, col_select = -1) %>% filter(step_coverage
>=0.8) # step coverage is the presence of enzymatic steps in the genome out of nessecary
steps for that module to function.
```

```
module_names <- kegg_modules %>%
  dplyr::select(c("module", "module_name")) %>%
  distinct() %>%
  column_to_rownames("module")
```

```
# step coverage matrix
stcov <- kegg_modules %>% filter(step_coverage !='NA') %>% pivot_wider(
    id_cols = genome,
    names_from = module,
    values_from = step_coverage

) %>% column_to_rownames("genome") %>% replace(is.na(.), 0) %>%#adding zero instead
as.matrix()
```

```
stcov = stcov[, colSums(stcov)>0]

stcov = stcov[rowSums(stcov)>0,]
```

```
# Heatmap
setHook("grid.newpage", function() pushViewport(viewport(x = 1, y = 1, width = 0.9,
height = 0.9, name = "vp", just = c("right", "top"))), action = "prepend")
hatmap <- pheatmap(stcov, show_colnames = F, show_rownames = F )
setHook("grid.newpage", NULL, "replace")
grid.text("Modules", y = -0.07, gp = gpar(fontsize = 16))
grid.text("Genomes", x = -0.07, rot = 90, gp = gpar(fontsize = 16))
grid.lines(x = 2, y = 30)
```

linking taxa to modules

```
# ps file for taxa

ps_rar <- phyloseq::rarefy_even_depth(ps, sample.size = 2600, replace = F)
# ps_spec <- gloomer(ps_rar, 'Species')
ps_rel <- transform_sample_counts(ps_rar, function(x){x/sum(x)})
```

```r
#ps file for modules
rel_ab <- otu_table(ps_rel)




df1 <- data.frame(tax_table(ps_rel)) %>% rownames_to_column('genome')
df2 <- data.frame(stcov) %>% rownames_to_column('genome')

d_tax <- left_join(df1, df2, by = 'genome') %>% column_to_rownames('genome')




head(d_tax)


annotations <- matrix("", nrow = nrow(stcov), ncol = ncol(stcov))
colnames(annotations) <- colnames(stcov)
rownames(annotations) <- rownames(stcov)

for (genome in rownames(annotations))
{
  for (module in colnames(annotations))
  {
    annotations[genome, module] <- paste0(
      "Name: ", d_tax[genome, "Species"],
      "\nPhylum: ", d_tax[genome, "Phylum"],
      "\nPathway: ", module_names[module, "module_name"]
    )
  }
}


names(d_tax)


heatmaply(stcov, title = "somehting",
          custom_hovertext = annotations,
          showticklabels = c(FALSE,FALSE),
          file = "./pathways.html" #saved as
          )

view(annotations)
```

Module abundance in genomes in samples

Calculate Abundance of pathways as the sum of abundance of species where a module is presence. This is equal to the matrix multiplication.

```
# making ps file with modules
rel_ab <- otu_table(ps_rel)
tmp = stcov[rownames(stcov) %in% rownames(rel_ab), ]

tmp = t(as(rel_ab, "matrix")) %*%  tmp %>% t() #module relabund per sample

ps_mod <- phyloseq(otu_table(as(tmp, "matrix"), taxa_are_rows = TRUE),
sample_data(ps_rel))


module_rel_ab = t(otu_table(ps_mod))

setHook("grid.newpage", function() pushViewport(viewport(x = 1, y = 1, width = 0.9,
height = 0.9, name = "vp", just = c("right", "top"))), action = "prepend")
pheatmap(module_rel_ab, show_rownames = T, show_colnames = T)

setHook("grid.newpage", NULL, "replace")
grid.text("Modules", y = -0.07, gp = gpar(fontsize = 16))
grid.text("Samples", x = -0.07, rot = 90, gp = gpar(fontsize = 16))
```

```
annotations <- matrix("", nrow = nrow(module_rel_ab), ncol = ncol(module_rel_ab))
colnames(annotations) <- colnames(module_rel_ab)
rownames(annotations) <- rownames(module_rel_ab)
for (sample in rownames(annotations))
{
  for (module in colnames(annotations))
  {
    annotations[sample, module] <- paste0(
      "Pathway: ", module_names[module, ]
    )
  }
}


mt <- data.frame(sample_data(ps_rel))

# clustering
mat <- t(module_rel_ab)
sds <- matrixStats::rowSds(mat)
o <- order(sds, decreasing = TRUE)
h_1 <- hclust(dist(mat[o,]))
h_2 <- hclust(dist(t(mat[o,])))


#making a phylum annotation and it only accepts one column dataframe
row.annot = fData(x)[rownames(fData(x)) %in% rownames(Biobase::exprs(x)[o,]),] %>%
dplyr::select(6, 2)
```

```r
#Making color index for the phylum annotation
library(RColorBrewer)

phyl.col = data.frame(Phylum = unique(row.annot[,2]), phyl.col = phylcol[1:11])

phyl.col = column_to_rownames(phyl.col, "Phylum") %>% as.matrix

#Making a color index for the Genus annotation, only if you use it
gen.col <- list(RColorBrewer::brewer.pal(9, name = "Set1"), RColorBrewer::brewer.pal(8,
"Accent"),
     RColorBrewer::brewer.pal(8, "Dark2"), RColorBrewer::brewer.pal(12, "Paired"),
RColorBrewer::brewer.pal(8, "Set2"), RColorBrewer::brewer.pal(12, "Set3")) %>% unlist

set.seed(10)
gen.col = data.frame(Genus = row.annot[,1], Gen.col = sample(gen.col, replace = F, size =
50)) %>% as.matrix
rownames(gen.col) <- gen.col[,1]



# healthy vs. sick week 0
subset_sp <- rownames(mt[mt$new_id %in% c("w0", "Donor"),])
mt_0 <- mt[rownames(mt)%in% subset_sp,]
mat_0 <- module_rel_ab[rownames(module_rel_ab)%in% subset_sp,]

mat <- t(mat_0)
sds <- rowSds(mat)
o <- order(sds, decreasing = TRUE)
h_1 <- hclust(dist(mat[o,]))
h_2 <- hclust(dist(t(mat[o,])))

rowSums(mat) %>% range

heatmaply(log(t(mat)+1),
colors= RColorBrewer::brewer.pal(11, "RdBu"),
  custom_hovertext = annotations,
  showticklabels = c(TRUE, TRUE),
  row_side_colors = list(Week_donor = mt_0[rownames(t(mat)), "new_id"]),
  file = "./module_sample.html",
  Rowv = h_2,
  Colv = h_1,
  k_row = 2,
  k_col = 5
)



pheat = pheatmap( angle_col = 45, t(mat[o,]), cellheight = 15,
                      #annotation_colors = list(Phylum = phyl.col[,1]), #Genus =
                      gen.col[,2]),
                      border_color = NA, annotation_row = mt_0%>% dplyr::select(new_id,
                      Responder),
```

```
                      cellwidth = 15, cutree_cols = 7, cutree_rows = 6, number_color =
                      "black",
                     display_numbers = round(t(mat[o,]),2), fontsize_number = 5,
                      Colv = as.dendrogram(h_1), cutcluster_rows = T, cluster_cols = T,
                      Rov = as.dendrogram(h_2),
                      col = RColorBrewer::brewer.pal(11, "Spectral"), width = 8, height =
                      15,
                      main = "Module relative abundance in different samples")

ggsave(plot = pheat, filename = "./plots/heat_module_sample_w0.jpeg", device = "jpeg",
dpi = 300, height = 10, width = 32)



# sick vs. sick week 0 vs w1
subset_sp <- rownames(mt[mt$Responder %in% c("YES"),])
mt_0 <- mt[rownames(mt)%in% subset_sp,]
mat_0 <- module_rel_ab[rownames(module_rel_ab)%in% subset_sp,]

mat <- log(t(mat_0)+1)
sds <- rowSds(mat)
o <- order(sds, decreasing = TRUE)[1:50]
h_1 <- hclust(dist(mat[o,]), method = "ward.D2")
h_2 <- hclust(dist(t(mat[o,])), method = "ward.D2")


heatmaply(log(t(mat)+1),
colors= RColorBrewer::brewer.pal(11, "RdBu"),
  custom_hovertext = annotations,
  showticklabels = c(TRUE, TRUE),
  row_side_colors = list(Week_donor = mt_0[, "new_id"]),
  file = "./module_sample.html",
  Rowv = h_2,
  Colv = h_1,
  k_row = 2,
  k_col = 5
)


pheatmap( angle_col = 45, t(mat[o,]), cellheight = 15,
                      #annotation_colors = list(Phylum = phyl.col[,1]), #Genus =
                      gen.col[,2]),
                      border_color = NA, annotation_row = mt_0%>% dplyr::select(new_id),
                      cellwidth = 15, cutree_cols = 7, cutree_rows = 2, number_color =
                      "black",
                     display_numbers = round(t(mat[o,]),2), fontsize_number = 5,
                      Colv = h_1, cutcluster_rows = F, cluster_cols = T, Rov = h_2,
                      col = RColorBrewer::brewer.pal(11, "Spectral"), width = 8, height =
                      15,
                      main = "Module relative abundance in different samples, w0 vs w1")


pheat = pheatmap( angle_col = 45, t(mat[o,]), cellheight = 15,
```

```
                      #annotation_colors = list(Phylum = phyl.col[,1]), #Genus =
                      gen.col[,2]),
                      border_color = NA, annotation_row = mt_0%>% dplyr::select(new_id),
                      cellwidth = 15, cutree_cols = 7, cutree_rows = 2, number_color =
                      "black",
                     display_numbers = round(t(mat[o,]),2), fontsize_number = 5,
                      Colv = as.dendrogram(h_1), cutcluster_rows = T, cluster_cols = T,
                      Rov = as.dendrogram(h_2),
                      col = RColorBrewer::brewer.pal(11, "Spectral"), width = 8, height =
                      15, clustering_method = 'ward.D2',
                      main = "Module relative abundance in different samples, w0 vs w1")

ggsave(plot = pheat, filename = "./plots/heat_module_sample_w0vsw1.jpeg", device =
"jpeg", dpi = 300, height = 10, width = 32)



mat <- t(mat[o,])

s_m <- scale(mat)
```

```
# get most abundant modules
abundance_per_module <- data.frame(abundance = colMeans(module_rel_ab)) %>%
arrange(desc(abundance))
abundance_per_module <- cbind(abundance_per_module,
module_names[rownames(abundance_per_module), ])
colnames(abundance_per_module) <- c("Average_abundance", "Description")




ggplot(abundance_per_module, aes(x = Average_abundance)) +
  geom_histogram() +
  labs(x = "Average module abundance", y = "counts") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))
```

Alpha diversity of modules

```
# dir.create('./modules')

write.table(x = otu_table(ps_mod) %>% data.frame() %>% rownames_to_column('id'),
"./modules/count_module.tsv", sep = "\t", row.names = F, col.names =  T)

write.table(x = sample_data(ps_mod) %>% data.frame() %>% rownames_to_column('id'),
"./modules/met_module.tsv", sep = "\t", row.names = F, col.names =  T)
```

```r
library(ggpubr, verbose = FALSE)
library(reshape2, verbose = FALSE)

otu_table(ps_mod) <- round(otu_table(ps_mod, taxa_are_rows = T))
shan = estimate_richness(ps_mod, measures = "Shannon")
chao  = estimate_richness(ps_mod, measures = "Chao1")

df = cbind(sample_data(ps_mod), shan, Chao1 = chao[,1])

my_comp <- list(c("Donor", "w0"), c("Donor", "w1"), c("w0", "w1"))

alpha_df = df %>% data.frame()
df =  melt(alpha_df) %>% filter(variable %in% c("Chao1", "Shannon")) %>% mutate(variable
= factor(variable, levels = c('Chao1', 'Shannon')))

df%>%
ggplot(
    aes(
        x = new_id,
        y = value
    )
) +
facet_wrap(
    ~ variable, scales = "free"
) +
geom_violin(trim = F, aes (fill = new_id), color = NA, show.legend = F, alpha = 0.25) +
geom_boxplot(width = 0.15) +
geom_jitter(aes(group = new_id), pch = 21, color= "#435252", fill = "#483a1f", size = 4,
alpha = 0.25, stroke = 0.95, position = position_jitter(width = 0.1)) +
geom_line(data = df[df$new_id != 'Donor',],aes(group = Patient),  size = 0.5, lty = 2,
alpha  =5, color = '#00d5ff')+
theme_bw() +
scale_fill_manual(values = c("salmon", "cyan4", "grey")) +
ggtitle(
    "Module alpha diversity"
) +
labs(
    x = "Groups",
    y = "Alpha diversity"
)



# dir.create("./modules/plots")
ggsave("./modules/plots/alpha_module.jpeg", devic = "jpeg", width  = 10, height = 8, dpi
= 500)
```

GLMER on alpha diversity

```r
library(lme4)
library(lmerTest)
library(useful)
library(nlme)
library(broom.mixed)
library(car)
library(lsmeans)
library(postHoc)
library(multcomp)
library(here)
library(DHARMa)
library(broom)
library(marginaleffects)


df = alpha_df %>% mutate(Patient= factor(Patient))

lm1 = glmer(Shannon ~ new_id + (1| Patient), data = df[df$new_id != "Donor",], family =
Gamma(link = "log"))
summary(lm1)
car::Anova(lm1)
```

Beta diversity on modules

```r
plog <- transform_sample_counts(ps_mod, function(x){log(x+1)})
plog <- subset_samples(plog, sample_data(plog)$new_id != "Donor")

df <- sample_data(plog) %>% data.frame()
mat <- otu_table(plog) %>% t()


# br_dis <- distance(plog, method = 'bray')







cm <- cca(
    mat ~ new_id + Condition(Patient),
    data = df
)

permutest(cm)


plot(cm)


module_names [rownames(module_names)%in%c('M00159', 'M00122', 'M00144'),] # important
modules
```

```r
# Extracting sample scores
s_score = vegan::scores(cm, display = "sites", choices = c(1,2)) %>% as.data.frame()
s_score <- cbind(as.data.frame(s_score), group = df$new_id)


# extracting mean scores
c_score = vegan::scores(cm, display = "cn") %>% as.data.frame()
sp_score <-  vegan::scores(cm, display = "species") %>% as.data.frame()

sp_score = sp_score %>% mutate(names = module_names[rownames(module_names) %in%
rownames(sp_score),1])

rownames(c_score) <- unique(
    df$new_id
)

# Or
c_score <- aggregate(cbind(CCA1, CA1) ~ group, data = s_score, FUN = mean)


# the same as aggregate
c_score = s_score %>% group_by(group) %>% summarise_all(mean)

# Making segments
segs <- merge(s_score, setNames(c_score, c("group", "CCA1_mean", "CA1_mean")),
by = 'group', sort = FALSE)


# Eigenvalues
eigval = cm$CCA$eig
interia_total = cm$tot.chi

x_fit = round(
    eigval[[1]]/sum(eigval)*100,
    1
)

y_fit = round(
    eigval[[2]]/sum(eigval)*100,
    1
)

x_tot = round(
    eigval[[1]]/interia_total*100,
    1
)

y_tot = round(
    eigval[[2]]/interia_total*100,
    1
)
```

```r
#reportint statistics

stat_annot = "Omnibus CCA statistics of the model:\nP = 0.62\nPsudo-F = 0.81\nR2 = 1.66"
s_score %>% group_by(group) %>% mutate(cca = CCA1[group == "w0"] - CCA1[group=="w1"])




s_score %>% ggplot(
    aes(CCA1, CA1)
)  +
# geom_segment(
#     data = segs,
#     aes(
#         xend = CCA1_mean,
#         yend = CA1_mean
#         ),
#         color = "#666666",
#         lty = 1,
#         alpha = 0.25
#     ) +
# geom_point(
#     data = segs,
#     aes(
#         x = CCA1_mean,
#         y = CA1_mean
#     ),
#     pch = 21,
#     stroke = 0.5,
#     fill = "#ffffff",
#     color = "#7c7c7c",
#     size = 4
# ) +
geom_point(
    pch = 21,
    aes(
        fill= df$new_id
    ),
    color = "white",
    stroke = 2,
    size = 10,
    alpha = 0.95
) +
theme_bw() +
geom_hline(
    yintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        0.5
        )
```

```
        ) +
geom_vline(
    xintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        alpha = 0.5
        )
        ) +
ggtitle(
    "PCoA plot of treatment effect on log-transfomred Bray-Curtis dissimilarity for
    modules/",
    "A constrained correspondence analaysis was performed to analyse the variance between
    groups.\nVariance from individual participants is factored out as conditional
    factor."
) +
scale_fill_manual(
    values = c(
    w0 = "#a11818dd",
    w1 = "#2b4557"
    )


) +
labs(
    x = glue(
        "CCA1 [{x_fit} % of fitted and {x_tot} % of total variance ]"
    ),
    y = glue(
        "CA1 (residuals) {round(interia_total*100,2)} % of total variance ]"

    ),
    fill = "Sample points"
) +
# stat_ellipse(aes(
#     group = group,
# ),
#     geom = "polygon",
#     segments = 20,
#     alpha = 0.15,
#     level = 0.85,
#     type = "t",
#     show.legend = F,
#     fill  = "#f71414",
#     lty = 2

# ) +
geom_text(
    aes(
        x = -5.5,
        y  = 2.5
        ),
        label = stat_annot,
```

```
        color = "#0f0f34",
        hjust = 0
) +
 geom_text_repel(
    aes(
        label = glue("{df$Patient}")
        ),
    size = 4,
    box.padding = unit( 0.4, units ="cm"),
    point.padding = unit(0.4, "cm"),
    arrow = arrow(length = unit(0.009, "npc"))
) +
stat_ellipse(aes(group = group, color = group), lty = 2, show.legend = F) +
geom_point(
    data = subset(sp_score, rownames(sp_score) %in% c("M00159", "M00122", "M00144")) ,
    aes(CCA1, CA1),
    pch = 21,
    fill = 'cyan4',
    color = '#c4cfc6',
    stroke = 3,

    size = 8
) +
geom_text_repel(
    data = subset(sp_score, rownames(sp_score) %in% c("M00159", "M00122", "M00144")),
    aes(
        x = CCA1, CA1, label = names
    ),
    size = 5,
    box.padding = unit( 0.4, units ="cm"),
    point.padding = unit(0.4, "cm"),
     label.padding = unit(0.25, "lines"),
     fill = "white",
     max.overlaps = 200
    #  label.size = 0.5
) +
scale_color_manual(
    values = c(
    w0 = "#a11818dd",
    w1 = "#2b4557"
    ))

ord <- metaMDS(as(mat, 'matrix'))
envfit(ord, env = df, perm = 999)

# ggsave("./modules/plots/beta_module_labeled.jpeg", device = "jpeg", width  = 15, height
= 10)

ggsave("./modules/plots/beta_module.jpeg", device = "jpeg", width  = 15, height = 10)
```

Subsetting data based on beta diversity individuals

```
ids <- s_score %>%
  dplyr::select(-CA1) %>%
  rownames_to_column('sample') %>%
  mutate(sample_id = sub("^[AB]", "", sample)) %>%
  pivot_wider(id_cols = sample_id, names_from = group, values_from = CCA1) %>%
  mutate(coef = w0 - w1) %>%
  data.frame() %>% arrange(desc(abs(coef))) %>% filter(abs(coef) >=2) %>%
  dplyr::select(sample_id) %>% mutate(ids = sub('^0+', '', sample_id), ids = paste0("p",
  ids)) %>% pull(ids)

sub_p <- subset_samples(plog, sample_data(plog)$Patient %in% ids)
ps_alpha <- subset_samples(ps_mod, sample_data(ps_mod)$Patient %in% ids)
otu_table(ps_alpha) <- round(otu_table(ps_alpha, taxa_are_rows = T))
```

Repeat alpha

```
library(ggpubr, verbose = FALSE)
library(reshape2, verbose = FALSE)

shan = estimate_richness(ps_alpha, measures = "Shannon")
chao  = estimate_richness(ps_alpha, measures = "Chao1")

df = cbind(sample_data(ps_alpha), shan, Chao1 = chao[,1])

my_comp <- list(c("Donor", "w0"), c("Donor", "w1"), c("w0", "w1"))

alpha_df = df %>% data.frame()
df =  melt(alpha_df) %>% filter(variable %in% c("Chao1", "Shannon")) %>% mutate(variable
= factor(variable, levels = c('Chao1', 'Shannon')))

df%>%
ggplot(
    aes(
        x = new_id,
        y = value
    )
) +
facet_wrap(
    ~ variable, scales = "free"
) +
geom_violin(trim = F, aes (fill = new_id), color = NA, show.legend = F, alpha = 0.25) +
geom_boxplot(width = 0.15) +
geom_jitter(aes(group = new_id), pch = 21, color= "#435252", fill = "#483a1f", size = 4,
alpha = 0.25, stroke = 0.95, position = position_jitter(width = 0.1)) +
geom_line(data = df[df$new_id != 'Donor',],aes(group = Patient),  size = 0.5, lty = 2,
alpha  =5, color = '#00d5ff')+
theme_bw() +
scale_fill_manual(values = c("salmon", "cyan4", "grey")) +
```

```
ggtitle(
    "Module alpha diversity, subseted"
) +
labs(
    x = "Groups",
    y = "Alpha diversity"
)
```

```
# dir.create("./modules/plots")
ggsave("./modules/plots/alpha_module_subseted.jpeg", devic = "jpeg", width  = 10, height
= 8, dpi = 500)

library(lme4)
library(lmerTest)
library(useful)
library(nlme)
library(broom.mixed)
library(car)
library(lsmeans)
library(postHoc)
library(multcomp)
library(here)
library(DHARMa)
library(broom)
library(marginaleffects)


df = alpha_df %>% mutate(Patient= factor(Patient))

lm1 = glmer(Shannon ~ new_id + (1| Patient), data = df[df$new_id != "Donor",], family =
Gamma(link = "log"))
summary(lm1)
car::Anova(lm1)
```

Repeat Beta

```
df <- sample_data(sub_p) %>% data.frame()
mat <- otu_table(sub_p) %>% t()


# br_dis <- distance(plog, method = 'bray')




cm <- cca(
```

```
    mat ~ new_id + Condition(Patient),
    data = df
)

permutest(cm)


plot(cm)


mds <- module_names %>% rownames_to_column('id')%>% filter(id%in%c('M00028', 'M00122',
'M00140', "M00019", "M00527", "M00432", "M00020", "M00050", "M00570", "M00082")) #
important modules


# Extracting sample scores
s_score = vegan::scores(cm, display = "sites", choices = c(1,2)) %>% as.data.frame()
s_score <- cbind(as.data.frame(s_score), group = df$new_id)


# extracting mean scores
c_score = vegan::scores(cm, display = "cn") %>% as.data.frame()
sp_score <-  vegan::scores(cm, display = "species") %>% as.data.frame()

sp_score = sp_score %>% mutate(names = module_names[rownames(module_names) %in%
rownames(sp_score),1])

rownames(c_score) <- unique(
    df$new_id
)

# Or
c_score <- aggregate(cbind(CCA1, CA1) ~ group, data = s_score, FUN = mean)


# the same as aggregate
c_score = s_score %>% group_by(group) %>% summarise_all(mean)

# Making segments
segs <- merge(s_score, setNames(c_score, c("group", "CCA1_mean", "CA1_mean")),
by = 'group', sort = FALSE)


# Eigenvalues
eigval = cm$CCA$eig
interia_total = cm$tot.chi

x_fit = round(
    eigval[[1]]/sum(eigval)*100,
    1
)

y_fit = round(
```

```
    eigval[[2]]/sum(eigval)*100,
    1
)

x_tot = round(
    eigval[[1]]/interia_total*100,
    1
)

y_tot = round(
    eigval[[2]]/interia_total*100,
    1
)
cm
#reportint statistics

stat_annot = "Omnibus statistics of the CCA model:\nP = 0.01\nPsudo-F = 2.55\nR2 = 11.6"




s_score %>% ggplot(
    aes(CCA1, CA1)
)  +
# geom_segment(
#     data = segs,
#     aes(
#         xend = CCA1_mean,
#         yend = CA1_mean
#         ),
#         color = "#666666",
#         lty = 1,
#         alpha = 0.25
#     ) +
# geom_point(
#     data = segs,
#     aes(
#         x = CCA1_mean,
#         y = CA1_mean
#     ),
#     pch = 21,
#     stroke = 0.5,
#     fill = "#ffffff",
#     color = "#7c7c7c",
#     size = 4
# ) +
geom_point(
    pch = 21,
    aes(
        fill= df$new_id
    ),
```

```
    color = "white",
    stroke = 2,
    size = 10,
    alpha = 0.95
) +
theme_bw() +
geom_hline(
    yintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        0.5
        )
        ) +
geom_vline(
    xintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        alpha = 0.5
        )
        ) +
ggtitle(
    "PCoA plot of treatment effect on log-transfomred Bray-Curtis dissimilarity for
    modules/",
    "A constrained correspondence analaysis was performed to analyse the variance between
    groups.\nVariance from individual participants is factored out as conditional
    factor."
) +
scale_fill_manual(
    values = c(
    w0 = "#a11818dd",
    w1 = "#2b4557"
    )


) +
labs(
    x = glue(
        "CCA1 [{x_fit} % of fitted and {x_tot} % of total variance ]"
    ),
    y = glue(
        "CA1 (residuals) unexplained {round(0.273*100,2)} % of total fotted variance ]"

    ),
    fill = "Sample points"
) +
stat_ellipse(aes(
    color = group,
),
    geom = "polygon",
    segments = 6,
    alpha = 0.15,
```

```
    level = 0.8,
    type = "t",
    show.legend = F,
    fill  = "#dddddd",
    lty = 2

) +
geom_text(
    aes(
        x = -1.5,
        y  = 2.5
        ),
        label = stat_annot,
        color = "#0f0f34",
        hjust = 0
) +
#  geom_text_repel(
#     aes(
#         label = glue("{df$Patient}")
#         ),
#     size = 4,
#     box.padding = unit( 0.4, units ="cm"),
#     point.padding = unit(0.4, "cm"),
#     arrow = arrow(length = unit(0.009, "npc"))
# ) +
geom_point(
    data = subset(sp_score, rownames(sp_score) %in% mds$id) ,
    aes(CCA1, CA1),
    pch = 21,
    fill = 'cyan4',
    color = '#c4cfc6',
    stroke = 3,

    size = 8
) +
geom_text_repel(
    data = subset(sp_score, rownames(sp_score) %in% mds$id),
    aes(
        x = CCA1, CA1, label = names
    ),
    size = 3,
    box.padding = unit( 0.4, units ="cm"),
    point.padding = unit(0.4, "cm"),
     label.padding = unit(0.25, "lines"),
     fill = "white",
     max.overlaps = 200
    #  label.size = 0.5
) +
scale_color_manual(
    values = c(
    w0 = "#a11818dd",
    w1 = "#2b4557"
    ))
```

```
ord <- metaMDS(as(mat, 'matrix'))
envfit(ord, env = df, perm = 999)
plot(ord)
ggsave("./modules/plots/beta_module_subsetted.jpeg", device = "jpeg", width  = 15, height
= 10)

# ggsave("./modules/plots/beta_module_subsetted_annot.jpeg", device = "jpeg", width  =
15, height = 10)
```

finding differences in expression

```
ps_w <- subset_samples(ps_mod, sample_data(ps_mod)$Patient %in% ids)

df = otu_table(ps_w)[rownames(otu_table(ps_w))%in% mds[,2], ]

df = df %>% data.frame() %>% rownames_to_column('id') %>% pivot_longer(cols=-id, names_to
= "sample", values_to = 'expression') %>% mutate(new_id = ifelse(grepl("A", sample),
'w0', 'w1'))
df = df %>% mutate(patient = paste0("p",gsub('^[AB]', "", sample)))

#negative binomial
gl <- glmer(expression ~ new_id + (1|patient), data = df, family = binomial)
summary(gl)
car::Anova(gl)


gr_df <- df %>% group_by(id, new_id) %>% summarise(expre_count = sum(expression)) %>%
pivot_wider(names_from = new_id, values_from = expre_count, values_fill = 0)


#t-test and chi-square test

t.test(gr_df$w0, gr_df$w1)
chisq.test(gr_df$w0, gr_df$w1)$p.value[[1]]


gr_df %>% filter(w0 >0) %>%
pivot_longer(cols = -id, names_to = 'week', values_to = 'val') %>% ggplot(aes(x = id, y =
val)) +
geom_col(aes(fill = week), position = 'dodge') +
theme_bw() +
theme(axis.text.x = element_text(angle = 45,hjust = 1, face = 'bold')) +
scale_fill_manual(
    values = c(
    w0 = "#a11818dd",
    w1 = "#2b4557"
    ))+
labs(
    x = "KEGG modules",
    y = "Count of modules",
```

```
    fill = "Weeks of intervention"
) +
ggtitle(
    "Number of modules with differeces in beta diversity from week0 to week1",
    "Only in patients with differences found in beta diversity"
) +
geom_text(data = gr_df %>% dplyr::select(w1) %>% filter(id == "C1-unit interconversion,
prokaryotes"),
   aes( y = 6, label = glue("Chi-square p-value = {round(chisq.test(gr_df$w0,
   gr_df$w1)$p.value[[1]],2)}"))
)


ggsave("./modules/plots/module_difference.jpeg", device = "jpeg", width = 25, height =
12, dpi =500)
```

Differential abundance analaysis on modules

```
library(DESeq2)

ps_w <- subset_samples(ps_mod, sample_data(ps_mod)$Patient %in% ids)




rownames(otu_table(ps_w))<- module_names[rownames(module_names) %in%
rownames(otu_table(ps_w)),1]
rownames(otu_table(ps_mod))<- module_names[rownames(module_names) %in%
rownames(otu_table(ps_mod)),1]

    design = as.formula(glue("~ new_id"))
    ds = phyloseq_to_deseq2(ps_w, design = design)
    gm_mean = function(x, na.rm = TRUE){
        exp(
            sum(
                log(x[x > 0]),
                na.rm = na.rm
            ) / length(x)
        )
    }

    geo_mean = apply(counts(ds), 1, gm_mean)

    ds = estimateSizeFactors(ds, geoMeans = geo_mean)
    ds = estimateDispersionsGeneEst(ds)

    dispersions(ds) <- mcols(ds)$dispGeneEst

    ds = nbinomWaldTest(ds)
```

```
resultsNames(ds)

results(ds, contrast  = c('new_id', 'w1', 'w0')) %>% data.frame %>% filter(!is.na(padj))


results(ds, name  = 'new_id_w1_vs_w0') %>% data.frame() %>% filter(padj <=1) %>%
rownames_to_column('name') %>% filter(name %in% mds[,2])
```

using limma and edgeR

```
library(limma)
library(edgeR)

mat <- as(otu_table(ps_w), 'matrix')
df <- data.frame(sample_data(ps_w))

y = DGEList(counts = mat)
y = calcNormFactors(y)


# Convert counts to log-CPM using voom
design <- model.matrix(~ new_id, data = df)  # Fixed effect: new_id
v <- voom(y, design)
corfit <- duplicateCorrelation(v, design, block = df$Patient)
fit = lmFit(v, design, block = df$Patient, correlation = corfit$consensus.correlation)
fit <- eBayes(fit)

topTable(fit)
```

Using Gam

```
library(glmmTMB)
dim(mat)
mt = mat %>% data.frame() %>%
rownames_to_column('genome') %>%
pivot_longer(!genome, names_to = 'samples', values_to = 'count')

df1 <- rownames_to_column(df, 'samples')
df_joint <- left_join(mt, df1, by = 'samples')



ml <- glmmTMB(
    count ~ new_id + (1|Patient),
    data = df_joint,
    family = 'nbinom2'
)
```

```r
summary(ml)
coef(ml)
car::Anova(ml)
df
```

Contringency plot: see the shared and unique pathways

```r
donor_samples <- rownames(df)[df$new_id == "Donor"]
w0_samples <- rownames(df)[df$new_id == "w0"]

ct <- mat
 ct[ct>0] <- 1

unique_taxa_in_donors <- rownames(ct)[rowSums(ct[, donor_samples] > 0) > 0 &
                                        rowSums(ct[, w0_samples] > 0) == 0]

unique_taxa_data <- mat[unique_taxa_in_donors, donor_samples]

# Convert the data to long format for ggplot2
library(reshape2)
long_data <- melt(unique_taxa_data)
colnames(long_data) <- c("Taxa", "Sample", "Abundance")

# Add metadata information to the long_data for plotting
long_data <- merge(long_data, df, by.x = "Sample", by.y = "new_id")

library(ggplot2)

# Plotting the unique taxa in donor samples
ggplot(long_data, aes(x = Taxa, y = Abundance, fill = Sample)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  ggtitle("Unique Taxa in Donors (Absent in w0)") +
  xlab("Taxa") +
  ylab("Abundance") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```r
library(UpSetR)

data <- otu_table(ps_spec)

data = apply(data, 2, function(x){
    ifelse(
        x > 0,
        1,
        0
    )
}) %>% data.frame

dim(data)
```

```
upset(data )


cnt <- otu_table(ps_log)
```

Gene annotation profile

Using Prokka annotated results

```
library(pheatmap)
library(matrixStats)
library(ggdendro)
library(patchwork)


len = list.files('./finals/genes_tsv')
names = list.files('./finals/genes_tsv')


df = list()

for(i in seq_along(len)){
  name = gsub("\\.tsv$", "", names[i])
  df[[name]] <- read.csv(glue("{getwd()}/finals/genes_tsv/{names[[i]]}"), sep = "\t") %>%
filter(product !="")   %>% data.frame()  %>% dplyr::select(length_bp, gene, product) %>%
group_by(product) %>% summarise(counts = n(), .groups = "drop")  %>%  mutate(sample =
rep(glue("{name}"), length(counts)))


}



df = do.call(rbind, df) %>%
      data.frame() %>%
      pivot_wider(names_from = 'sample',
      values_from = 'counts', values_fill = list(counts = 0))  %>% filter(product !=
      "hypothetical protein") %>%
      column_to_rownames('product')



df_mat <- df %>% as.matrix

df_mat[df_mat > 0] <- 1

head(df_mat)


#Keeping only the genes that are not shared among all samples
```

```
mat = df_mat[rowSums(df_mat)!=ncol(df_mat),]
sds = rowSds(mat)
o = order(sds, decreasing = TRUE)[1:100]
h_1 = hclust(dist(mat[o,], method = 'manhattan'), method = "ward.D2")
h_2 = hclust(dist(t(mat[o,]), method = 'manhattan'), method = "ward.D2")
plot(h_2)

ggdendrogram(h_2, rotate = T)

df_annot <- sample_data(ps_rel) %>% data.frame %>% filter(wd %in% c("w0_don2", "w1_don2",
"Don2")) %>% mutate('Sample point' = wd)


disc_color(n = 3)
col_wd = c("w0_don2" = "#89dbcd", "w1_don2" = "#ffffb0", "Don2" = "#eac5f6")

mat = mat[,rownames(df_annot)]

#distance heatmap

p_mat = pheatmap(mat[o,], color = c( "#f3f3f3", "#6a6c6c"), cluster_cols = T,
cluster_rows = F,  border_color = "NA", legend_breaks = c(0,1), cellwidth =  15,
cellheight = 12, Cov = as.dendrogram(h_2),   cutree_cols = 3,  angle_col = 45, main =
"Contingency of annotated genes, for Donor 2 and its recipients. Ward.D2 ", show_colnames
=T, annotation_col = df_annot %>% dplyr::select('Sample point'), annotation_colors =
list('Sample point' = col_wd ))

ggsave(plot = p_mat, "./plots/contingency_genes_100.jpeg", device = "jpeg", limitsize =
F, height = 20, width = 20, dpi = 400)
```

constrained correspondence analysis on the annotated genes

```
mat_t <- as.matrix(df)


# once with patients before after
mt <- sample_data(ps_rel) %>% data.frame %>% filter(wd %in% c("w0_don2", "w1_don2"))

mat_t <- mat_t[,colnames(mat_t) %in% rownames(mt)]



# br = vegdist(t(mat_t), method = 'bray')
# dbr <- dbrda(br ~ wd + Condition (Patient), data = mt)
ccm_w0_w1 <- cca(t(mat_t) ~ wd + Condition(Patient), data = mt)

permutest(ccm_w0_w1, by = 'onedf')

# with patients w0 vs donor2
mt <- sample_data(ps_rel) %>% data.frame %>% filter(wd %in% c("w0_don2", "Don2"))
mat_t <- as.matrix(df)
```

```
mat_t <- mat_t[,colnames(mat_t) %in% rownames(mt)]
dim(mat_t)

ccm_w0_d2 <- cca(t(mat_t) ~ wd, data = mt)

permutest(ccm_w0_w1, by = 'onedf')


# with patients and donor2
mt <- sample_data(ps_rel) %>% data.frame %>% filter(wd %in% c("w0_don2","w1_don2",
"Don2"))
mat_t <- as.matrix(df)
mat_t <- mat_t[,colnames(mat_t) %in% rownames(mt)]


ccm_all <- cca(t(mat_t) ~ wd, data = mt)

permutest(ccm_all, by = 'terms')
```

Visualizing cca of annotaed genes

```
# Extracting sample scores
s_score = vegan::scores(ccm_all, display = "sites", choices = c(1,2)) %>% as.data.frame()
sm_score = vegan::scores(ccm_all, display = "species", choices = c(1,2)) %>%
as.data.frame()
s_score <- cbind(as.data.frame(s_score), group = mt$wd)
plot(ccm_all)

# extracting mean scores
c_score = vegan::scores(ccm_all, display = "cn") %>% as.data.frame()
sp_score <-  vegan::scores(ccm_all, display = "species") %>% as.data.frame()

sp_score = sp_score %>% mutate(names = module_names[rownames(module_names) %in%
rownames(sp_score),1])

rownames(c_score) <- c('Don2', 'Week0', 'Week1')

# Or
c_score <- aggregate(cbind(CCA1, CCA2) ~ group, data = s_score, FUN = mean)


# the same as aggregate
c_score = s_score %>% group_by(group) %>% summarise_all(mean)

# Making segments
segs <- merge(s_score, setNames(c_score, c("group", "CCA1_mean", "CCA2_mean")),
by = 'group', sort = FALSE)


# Eigenvalues
eigval = ccm_all$CCA$eig
```

```
interia_total = ccm_all$tot.chi

x_fit = round(
    eigval[[1]]/sum(eigval)*100,
    1
)

y_fit = round(
    eigval[[2]]/sum(eigval)*100,
    1
)

x_tot = round(
    eigval[[1]]/interia_total*100,
    1
)

y_tot = round(
    eigval[[2]]/interia_total*100,
    1
)

#reportint statistics

stat_annot = "Omnibus CCA statistics of the model:\nP = 0.01\nPsudo-F = 1.97\nR2 = 10.1"




s_score %>% ggplot(
    aes(CCA1, CCA2)
)  +
geom_segment(
    data = segs,
    aes(
        xend = CCA1_mean,
        yend = CCA2_mean
        ),
        color = "#666666",
        lty = 1,
        alpha = 0.25
    ) +
geom_point(
    data = segs,
    aes(
        x = CCA1_mean,
        y = CCA2_mean
    ),
    pch = 21,
    stroke = 0.5,
    fill = "#ffffff",
```

```
    color = "#7c7c7c",
    size = 4
) +
geom_point(
    pch = 21,
    aes(
        fill= group
    ),
    color = "white",
    stroke = 2,
    size = 10,
    alpha = 0.95
) +
theme_bw() +
geom_hline(
    yintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        0.5
        )
        ) +
geom_vline(
    xintercept = 0,
    lty = 2,
    color = ggplot2::alpha(
        colour = "orange",
        alpha = 0.5
        )
        ) +
ggtitle(
    "PCoA plot of intervention for Bray-Curtis dissimilarity of annotated genes",
    "A constrained correspondence analaysis was performed to analyse the variance between
    groups."
) +
scale_fill_manual(
    values = c(
    w0_don2 = "#0b5000dd",
    w1_don2 = "#c22790",
    Don2 = 'darkorange'
    )


) +
labs(
    x = glue(
        "CCA1 [{x_fit} % of fitted and {x_tot} % of total variance ]"
    ),
    y = glue(
        "CCA2  [{y_fit} % of fitted and {y_tot} % of total variance ]"

    ),
    fill = "Sample points"
```

```
) +
geom_text(
    aes(
        x = -2,
        y  = 2.5
        ),
        label = stat_annot,
        color = "#0f0f34",
        hjust = 0
) +
stat_ellipse(aes(group = group, color = group), type = 't', lty = 2, show.legend = F,
segments = 8, level = 0.95, fill = alpha(alpha = 0.05,'grey'), geom = 'polygon') +
scale_color_manual(
    values = c(
    w0_don2 = "#0b5000dd",
    w1_don2 ="#c22790",
    Don2 = 'darkorange'
    ))


ggsave("./plots/beta_annot_genes.jpeg", device = "jpeg", width  = 15, height = 10)
```

Difabund annotated genes

```
mat_t <- as.matrix(df)
mt_df <- mt #%>% filter(wd %in% "Don2")

dss <- DESeqDataSetFromMatrix(countData = mat_t[, colnames(mat_t) %in% rownames(mt_df)],
colData = mt_df, design = ~ wd)


dss <- DESeq(dss)
resultsNames(dss)

sig_df = results(dss, name = 'wd_w1_don2_vs_w0_don2') %>% data.frame() %>% filter(padj <=
0.05) %>% filter(abs(log2FoldChange) > 0) %>% rownames_to_column('Genes') %>%
arrange(desc(log2FoldChange)) %>% mutate(Genes = factor(Genes, levels = Genes), col =
ifelse(log2FoldChange > 0 , 'Increased', 'Decreased'))

sig_df %>% ggplot(aes(x = log2FoldChange, y = Genes)) +
# geom_col(width = 1, fill = 'grey', alpha = 0.6) +
#geom_point(aes(color = col), size = 7, show.legend = F) +
geom_col(width = 1, aes(fill = col), alpha = 0.4, show.legend = F)+
theme_minimal() +
ggtitle('Annotated genes in patients at W0 \ncompared to Donor (all donor 2)') +
# scale_color_manual(values = c(Increased = 'cyan4', Decreased = 'darkorange')) +
scale_fill_manual(values = c(Increased = 'cyan4', Decreased = 'darkorange')) +
geom_vline(xintercept = 0, lty = 3) +
geom_line() +
labs(y = 'Annotated genes')
```

```
ggsave('./plots/difabund_genes_w0_vs_don2.jpeg', dpi = 300, width = 10, height = 22)


sig_df =results(dss, contrast = c('wd', 'w1_don2', 'don2')) %>% data.frame() %>%
filter(padj <= 0.05) %>% filter(abs(log2FoldChange) > 0) %>% rownames_to_column('Genes')
%>% arrange(desc(log2FoldChange)) %>% mutate(Genes = factor(Genes, levels = Genes), col =
ifelse(log2FoldChange > 0 , 'Increased', 'Decreased'))


sig_df %>% ggplot(aes(x = log2FoldChange, y = Genes)) +
# geom_col(width = 1, fill = 'grey', alpha = 0.6)+
#geom_point(aes(color = col), size = 7, show.legend = F) +
geom_col(width = 1, aes(fill = col), alpha = 0.4, show.legend = F)+
theme_minimal() +
ggtitle('Annotated genes in patients after W0 compared to Donor (all donor 2)') +
# scale_color_manual(values = c(Increased = 'cyan4', Decreased = 'darkorange')) +
scale_fill_manual(values = c(Increased = 'cyan4', Decreased = 'darkorange')) +
geom_vline(xintercept = 0, lty = 3) +
geom_line() +
labs(y = 'Annotated genes')

ggsave('./plots/difabund_genes_w1_vs_w0_d2.jpeg', dpi = 300, width = 10, height = 22)
```

Contingency of annotated genes

```
library(UpSetR)
library(ComplexUpset)


df <-  df1 %>% group_by(Species, Family, wd) %>% summarise(rel = mean(relabund)) %>%
ungroup() %>% pivot_wider(id_cols = c(Species, Family), names_from = wd, values_from =
rel)


df <- df %>%
  mutate(
    w0_present = ifelse(w0_don2 > 0, 1, 0),
    w1_present = ifelse(w1_don2 > 0, 1, 0),
    Don2_present = ifelse(Don2 > 0, 1, 0)
  )


upset_data <- df %>%
  select(w0_present, w1_present, Don2_present) %>%
  as.data.frame()

# Create the UpSet plot
jpeg('./plots/upset_plot_don2_w0vsw1.jpeg', units = 'cm', height = 15, width = 20, res =
1000)
upset(upset_data, sets = c("w0_present", "w1_present", "Don2_present"), order.by =
"freq")
```

```r
dev.off()


# Create binary columns for each value in the 'wd' column
wd_sets <- table(mt$wd)
wd_sets_df <- as.data.frame.matrix(wd_sets)



mt_df <- mt

mat_t = mat_t[, colnames(mat_t) %in% rownames(mt_df)]

mat_t[mat_t>0] <- 1

m <- t(mat_t)



mat_t %>% data.frame %>% rownames_to_column('genes') %>% pivot_longer(cols = -genes,
names_to = 'Sample', values_to = 'count')
```