Cryopreservation of a soil microbiome using a Stirling Cycle approach – a genomic (ITS data) assessment

Payton Yau

2024-02-05

Contents

0.1 Cyropreserve - ITS data

Soil microbiomes are responsive to seasonal and long-term environmental factors, impacting their composition and function. This manuscript explores cryopreservation techniques using a controlled rate cooler and assesses the genomic integrity and bacterial growth of an exemplar soil sample before and after cryopreservation. The study demonstrates that the controlled rate cooler effectively preserves the DNA content of the microbiome. Two cryopreservation methods were compared with control samples, and the results indicate successful cryopreservation using metabarcoding. Enrichment with liquid medium showed similar responses between cryopreserved and non-cryopreserved soil samples, supporting the efficacy of cryopreservation. This study represents the first report of cryopreservation of soil using a Stirling cycle cooling approach, highlighting its potential for future microbiome research.

0.1.1 Load the required packages

```r
# install.packages(c("ggplot2", "ggpubr", "dplyr",
#                   "rstatix", "purrr", "reshape2",
#                   "UpSetR","plyr", "dplyr", "RColorBrewer"))
library("ggplot2")
library("ggpubr")
library("dplyr")
library("rstatix")
library("purrr")
library("reshape2")
library("UpSetR")
library("plyr")
library("dplyr")
library("RColorBrewer")

# if (!require("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install(c("phyloseq", "DESeq2", "microbiome"))
library("phyloseq")
```

```r
library("DESeq2")
library("microbiome")

# if(!requireNamespace("devtools", quietly = TRUE)){install.packages("devtools")}
# devtools::install_github("jbisanz/qiime2R") # current version is 0.99.20
library("qiime2R")

# devtools::install_github("pmartinezarbizu/pairwiseAdonis/pairwiseAdonis")
library("pairwiseAdonis")
```

0.1.2 Qiime2 to Phyloseq

To work with QIIME2 outcomes in the R environment, it is beneficial to convert the data into the phyloseq object structure. This process involves importing and transforming the feature table and sample metadata, allowing for comprehensive analysis and visualization of microbial community profiles. The phyloseq package in R provides functions to organize and manipulate the data within the phyloseq object, enabling various analyses such as diversity assessments, differential abundance testing, and taxonomic profile visualization. By converting QIIME2 outcomes to phyloseq, researchers can leverage the capabilities of R for advanced statistical analysis, integration with other omics data, and gaining deeper insights into the microbiome datasets.

```r
# Convert qiime2 to phyloseq format
physeq <- qza_to_phyloseq(
  features = "qiime2/table-its-with-phyla-no-mitochondria-no-chloroplast.qza", # table.qza
  # tree = "inst/artifacts/2020.2_moving-pictures/rooted-tree.qza",
  taxonomy = "qiime2/taxonomy-its.qza",
  metadata = "meta-data-ITS.txt"
)
physeq ## confirm the object
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 4940 taxa and 15 samples ]
## sample_data() Sample Data:       [ 15 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 4940 taxa by 7 taxonomic ranks ]
```

0.1.3 import data and subgroup the data

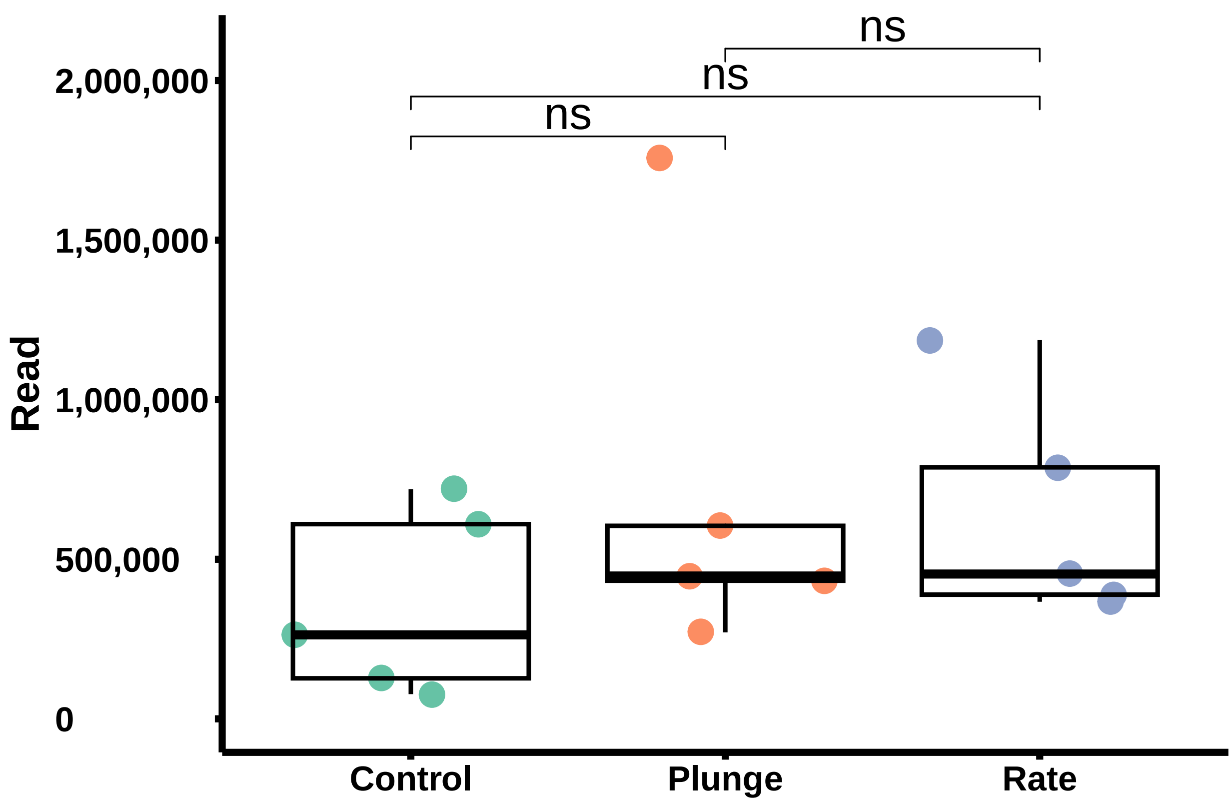
Normalise number of reads in each sample by using median sequencing depth

```r
## Normalise number of reads in each sample by using median sequencing depth
total = median(sample_sums(physeq))
standf = function(x, t=total) round(t * (x / sum(x)))
physeq.norm = transform_sample_counts(physeq, standf)

physeq.norm.group = merge_samples(physeq.norm, "Group") # Sum between replicate samples
sample_data(physeq.norm.group)$Group <- rownames(sample_data(physeq.norm.group))

rm(total, standf)
```

```r
meta <- data.frame(physeq.norm@sam_data)

# Now you can use 'meta_df' in your functions
stat.test1 <- meta %>%
  t_test(Raw_Reads ~ Group) %>%
  adjust_pvalue(method = "bonferroni") %>%
  add_significance()

print(stat.test1)
```

```
## # A tibble: 3 x 10
##   .y.       group1  group2    n1    n2 statistic    df     p p.adj p.adj.signif
##   <chr>     <chr>   <chr>  <int> <int>     <dbl> <dbl> <dbl> <dbl> <chr>
## 1 Raw_Reads Control Plunge     5     5    -1.15   5.76 0.296 0.888 ns
## 2 Raw_Reads Control Rate       5     5    -1.36   7.72 0.211 0.633 ns
## 3 Raw_Reads Plunge  Rate       5     5     0.211  6.45 0.84  1     ns
```

```r
# Plot a graph of the abundance of Fusarium for each sample grouped by Group:
Raw_Reads.Ori <- ggplot(subset(meta, Group %in% c("Control","Plunge","Rate")),
                        aes(x = Group, y = Raw_Reads, colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8) +
  scale_y_continuous(labels = scales::comma, limits=c(0, 2100000),
                     breaks = c(0, 500000, 1000000, 1500000, 2000000)) +
    stat_pvalue_manual(stat.test1,
                     y.position = c(1825000, 1950000, 2100000),
                     label = "p.adj.signif",
                     face="bold",
                     size = 6,
                     linetype = 1,
                     tip.length = 0.02,
                     inherit.aes = FALSE) +
  theme_classic() +
  labs(x = "", y = "Read") +
  theme(text = element_text(size=18, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.25),
        axis.line = element_line(colour = 'black', size = 1.25),
        axis.text.x = element_text(colour = "black",
                                   angle=0,
                                   size = 13, face="bold"),
        axis.text.y = element_text(angle=0, hjust=0, colour = "black",
                                   size = 13, face="bold"),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "none") +
  scale_color_brewer(palette="Set2")+
  scale_fill_brewer(palette="Set2")

# pdf(file = "Raw_Reads.ITS.pdf", width = 6, height = 5)
Raw_Reads.Ori
```




```r
# Close the PDF device and save the plot to a file
# dev.off()


# Clean up by removing objects that are no longer needed
rm(physeq.ori, meta, Raw_Reads.Ori, stat.test1)
```

0.1.4 Beta diversity

Beta diversity is a measure used in ecological and microbial community studies to assess the dissimilarity of species or taxa compositions between different samples. It quantifies the variation in community structure and helps researchers understand the diversity and uniqueness of microbial communities. Various metrics, such as Bray-Curtis dissimilarity and Jaccard index, are employed to calculate beta diversity values, which can be visualized using techniques like Principal Coordinate Analysis or Non-Metric Multidimensional Scaling. Beta diversity analysis allows for comparisons of microbial communities across habitats, treatments, or environmental gradients, revealing factors influencing community variation and identifying key drivers of community structure. It provides insights into the functional and ecological significance of different microbial assemblages and their responses to environmental changes, aiding our understanding of microbial community dynamics and their roles in ecology, environmental science, and human health research.

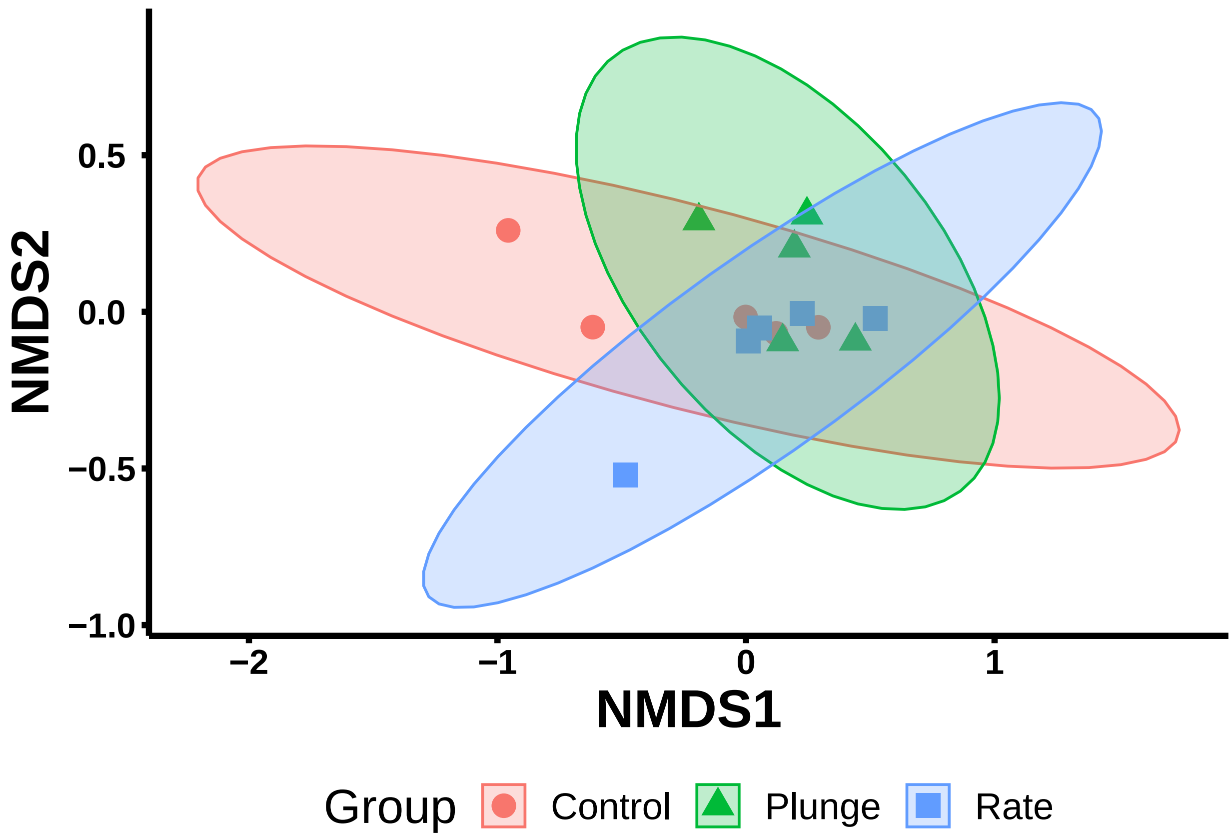
```r
nmds <- ordinate(physeq = physeq.norm, method = "NMDS", distance = "bray")
```

```
## Square root transformation
## Wisconsin double standardization
```

```
## Run 0 stress 0.04658859
## Run 1 stress 0.04658872
## ... Procrustes: rmse 3.605162e-05  max resid 6.5185e-05
## ... Similar to previous best
## Run 2 stress 0.04658862
## ... Procrustes: rmse 6.151511e-05  max resid 0.0001706606
## ... Similar to previous best
## Run 3 stress 0.1027285
## Run 4 stress 0.06610788
## Run 5 stress 0.04710306
## Run 6 stress 0.07877258
## Run 7 stress 0.06610821
## Run 8 stress 0.08494165
## Run 9 stress 0.04658859
## ... New best solution
## ... Procrustes: rmse 9.615247e-06  max resid 2.356616e-05
## ... Similar to previous best
## Run 10 stress 0.1027284
## Run 11 stress 0.06610828
## Run 12 stress 0.07813346
## Run 13 stress 0.04658861
## ... Procrustes: rmse 5.024713e-05  max resid 0.0001385339
## ... Similar to previous best
## Run 14 stress 0.07827761
## Run 15 stress 0.08505875
## Run 16 stress 0.0465886
## ... Procrustes: rmse 1.214252e-05  max resid 2.911525e-05
## ... Similar to previous best
## Run 17 stress 0.06610797
## Run 18 stress 0.06715053
## Run 19 stress 0.0661078
## Run 20 stress 0.06726516
## *** Best solution repeated 3 times
```

```r
Beta.its <- plot_ordination(
  physeq = physeq.norm,
  ordination = nmds,
  # title = "NMDS",
  color = "Group",
  shape = "Group") +
  # scale_x_discrete(name ="NMDS1 ()") +
  # scale_y_discrete(name ="NMDS2 ()") +
  theme_classic() +
  geom_point(aes(color = Group), alpha = 1, size = 4) +
  theme(text = element_text(size=18, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.1),
        axis.line = element_line(colour = 'black', size = 1.1),
        axis.text.x = element_text(colour = "black", angle=0,
                                   hjust=0.5, size = 13, face="bold"),
        axis.text.y = element_text(colour = "black", angle=0,
                                   hjust=0.5, size = 13, face="bold"),
        axis.title.y = element_text(color="black", size=20,face="bold"),
        axis.title.x = element_text(color="black", size=20,face="bold"),
                legend.position = "bottom") + # This line moves the legend to the bottom
```

```
  stat_ellipse(geom = "polygon", type="norm", alpha=0.25, aes(fill = Group))

# pdf(file = "Beta.its.pdf", width = 6,height = 6.1)
Beta.its
```

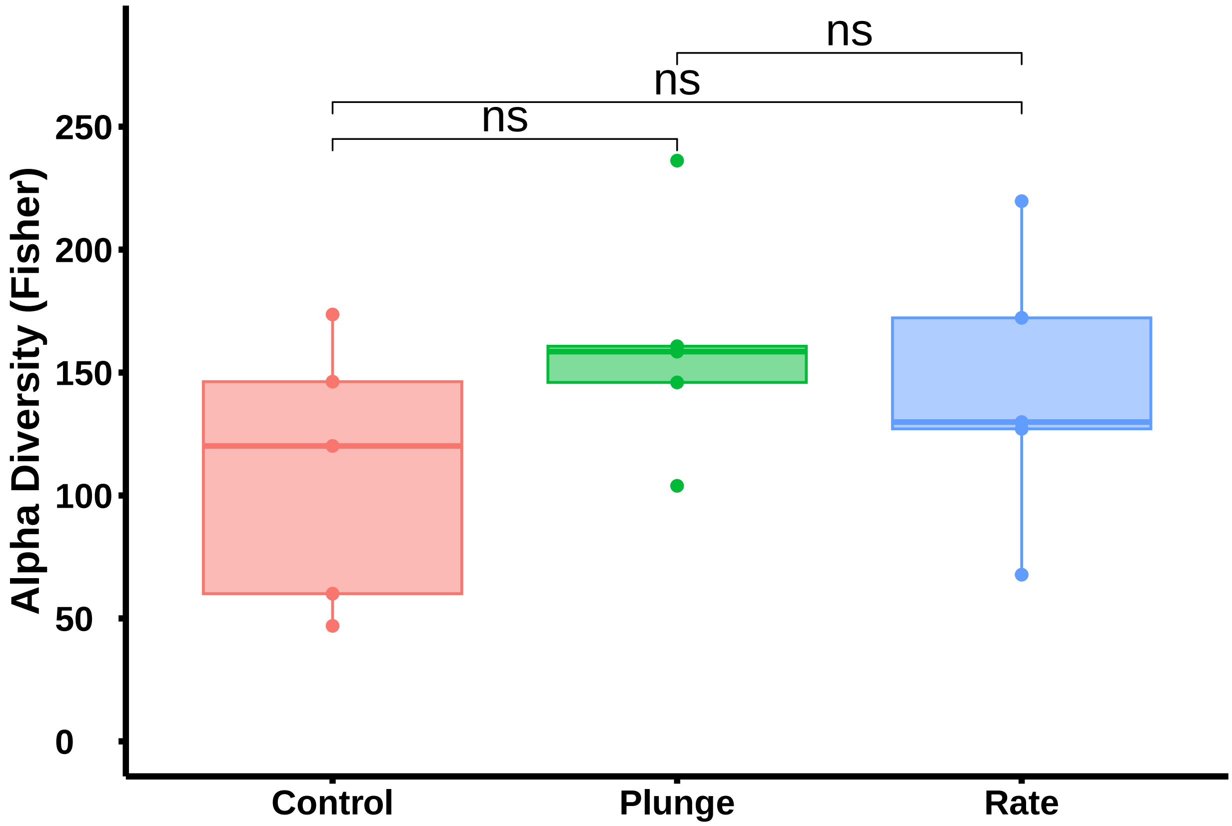



```
# Close the PDF device and save the plot to a file
# dev.off()

rm(nmds, Beta.its)
```

0.1.5 Alpha diversity

Alpha diversity is a fundamental concept in ecology and refers to the diversity or richness of species within a specific community or habitat. In the context of microbial ecology, alpha diversity represents the diversity of microorganisms within a given sample or microbiome. It provides insights into the variety and evenness of microbial species present in a particular environment. Common measures of alpha diversity include species richness, which counts the number of unique species, and evenness, which assesses the distribution of species abundances. Alpha diversity is crucial for understanding the stability, resilience, and functional potential of microbial communities. It can be influenced by various factors, including environmental conditions, host factors, and perturbations. By comparing alpha diversity across different samples or experimental groups, researchers can gain insights into the impact of factors such as disease, habitat changes, or interventions on microbial community structure.

```r
# available measurements [c("Observed", "Chao1", "ACE", "Shannon", "Simpson", "InvSimpson", "Fisher")]
tab = cbind(x = sample_data(physeq.norm),
            y = estimate_richness(physeq.norm, measures = 'Fisher'))

stat.test <- tab %>%
  # group_by(Neutrophils, GROUP1) %>%
  t_test(Fisher ~ x.Group) %>%
  adjust_pvalue(method = "bonferroni") %>%
  add_significance()

alpha.its <- ggplot(data = tab, aes(x = x.Group, y = Fisher, color = x.Group, fill = x.Group)) +
  theme_classic() +
  labs(# title = "IBD Patients",
    x = element_blank(),
    y = "Alpha Diversity (Fisher)") +
  geom_point(size = 1.75) +
  geom_boxplot(alpha = 0.5) +
  stat_pvalue_manual(stat.test,
                     y.position = c(245, 260, 280),
                     label = "p.adj.signif",
                     face="bold",
                     size = 6,
                     linetype = 1,
                     tip.length = 0.02,
                     inherit.aes = FALSE) +
  scale_y_continuous(limits=c(0 , 285), breaks = c(0, 50, 100, 150, 200, 250)) +
  theme(text = element_text(size=18, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.1),
        axis.line = element_line(colour = 'black', size = 1.1),
        axis.text.x = element_text(colour = "black",
                                   angle=0,
                                   size = 13, face="bold"),
        axis.text.y = element_text(angle=0, hjust=0, colour = "black",
                                   size = 13, face="bold"),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "none")

# pdf(file = "alpha.its.pdf", width = 6, height = 5)
alpha.its
```




```
# Close the PDF device and save the plot to a file
# dev.off()

rm(tab, stat.test, alpha.its)
```

0.1.5.1 Determine the count of taxa within each level and group

The purpose of this process is to visualise the distribution of the number of matched abundance across different groups and to identify any patterns in the distribution of the processed abundance within individual group.

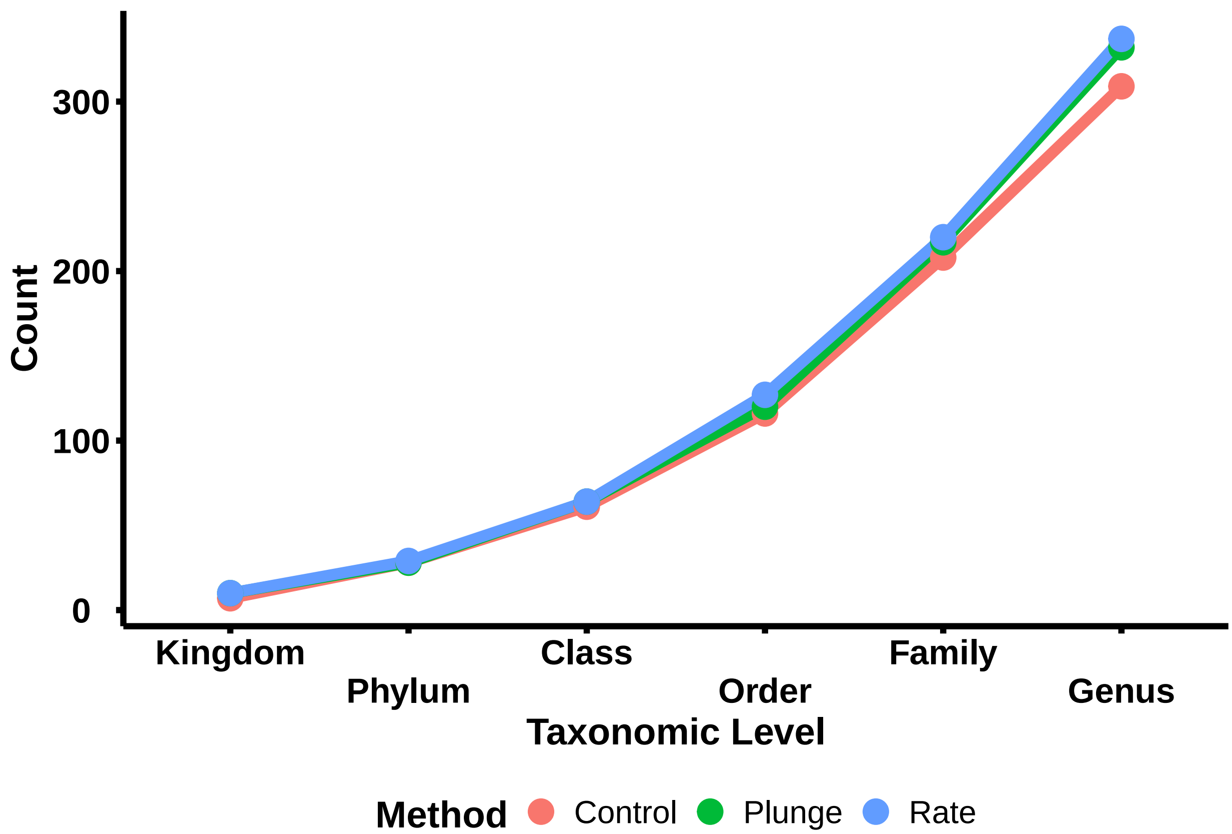
```
# Create an empty list to store genus-level abundance data for each taxonomic level
gentab_levels <- list()

# Set observation threshold
observationThreshold <- 15

# Define the taxonomic levels
genus_levels <- c("Kingdom", "Phylum", "Class", "Order",
                  "Family", "Genus", "Species")

# loop through all the taxonomic levels
for (level in genus_levels) {

  # create a factor variable for each level
  genfac <- factor(tax_table(physeq.norm.group)[, level])
```

```r
  # calculate the abundance of each genus within each sample
  gentab <- apply(otu_table(physeq.norm.group), MARGIN = 1, function(x) {
    tapply(x, INDEX = genfac, FUN = sum, na.rm = TRUE, simplify = TRUE)
  })

  # calculate the number of samples in which each genus is observed above the threshold
  level_counts <- apply(gentab > observationThreshold, 2, sum)

  # create a data frame of level counts with genus names as row names
  BB <- as.data.frame(level_counts)
  BB$name <- row.names(BB)

  # add the data frame to the gentab_levels list
  gentab_levels[[level]] <- BB
}

# Combine all level counts data frames into one data frame
B2 <- gentab_levels %>% purrr::reduce(dplyr::full_join, by = "name")

# Set row names and column names
rownames(B2) <- B2$name
B2$name <- NULL
colnames(B2)[1:7] <- genus_levels
B2$Species <- NULL
B2$name <- rownames(B2)

# Print the resulting data frame
print(B2)
```

```
##         Kingdom Phylum Class Order Family Genus    name
## Control       7     28    61   116    208   309 Control
## Plunge       10     28    64   120    217   332  Plunge
## Rate         10     29    64   127    220   337    Rate
```

```r
data_long <- melt(B2, id.vars = "name",
                  variable.name = "Dataset",
                  value.name = "Count")
colnames(data_long) = c("Method","Taxonomic.Level","Count")


tax.its <- ggplot(data_long, aes(x = Taxonomic.Level,
                                 y = Count,
                                 color = Method,
                                 group = Method)) +
  geom_line(size = 2) +
  geom_point(size = 4) +
  labs(x = "Taxonomic Level", y = "Count", color = "Method") +
  theme_classic() +
  theme(
    text = element_text(size = 19, colour = "black"),
    axis.ticks = element_line(colour = "black", size = 1.1),
    axis.line = element_line(colour = 'black', size = 1.1),
    axis.text.x = element_text(colour = "black", angle = 0, hjust = 0.5, size = 13, face = "bold"),
```

```r
    axis.text.y = element_text(colour = "black", angle = 0, hjust = 0.5, size = 13, face = "bold"),
    axis.title.y = element_text(color = "black", size = 14, face = "bold"),
    axis.title.x = element_text(color = "black", size = 14, face = "bold"),
    legend.title = element_text(size = 13.5, face = "bold"),
    legend.text = element_text(size = 12),
    legend.key.size=unit(0.4,"cm"),
    legend.position = "bottom") + # This line moves the legend to the bottom
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  scale_y_continuous(breaks=seq(0,600,by=100))

# pdf(file = "tax.its.pdf", width = 6, height = 6.1)
tax.its
```




```r
# Close the PDF device and save the plot to a file
# dev.off()

# Clean up by removing unnecessary objects
rm(gentab_levels, genus_levels, observationThreshold,
   BB, B2, data_long, gentab, tax.its, genfac, level, level_counts)
```

0.1.6 Upset plot using UpsetR

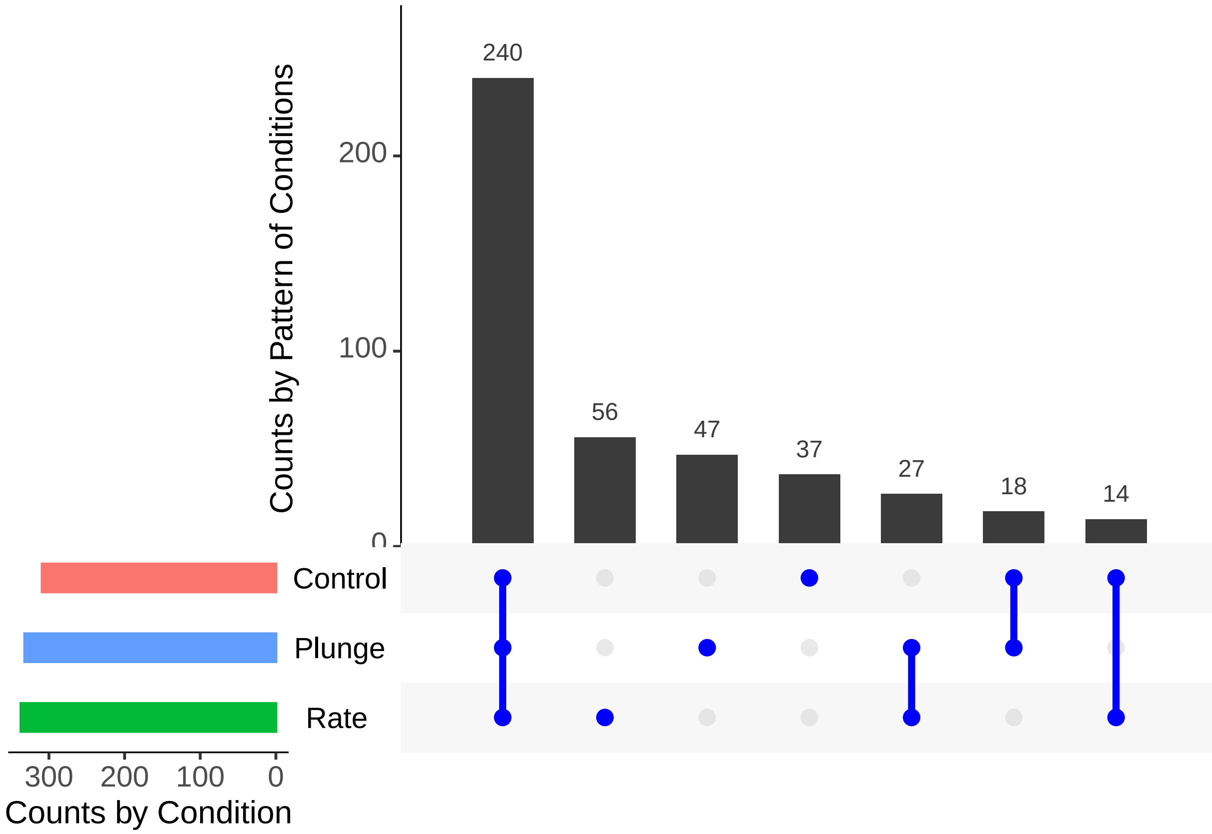
Venn diagrams are commonly used for visualizing sets, but they can become complex with more than five sets. UpSet graphs, on the other hand, offer a more efficient way to display intersections and complements, especially for larger or multiple datasets. They provide a more intuitive and informative data representation.

```r
# Aggregate taxa at the genus level
B <- aggregate_taxa(physeq.norm.group, "Genus", verbose = TRUE)
```

```
## [1] "Remove taxonomic information below the target level"
## [1] "Mark the potentially ambiguous taxa"
## [1] "-- split"
## [1] "-- sum"
## [1] "Create phyloseq object"
## [1] "Remove ambiguous levels"
## [1] "-- unique"
## [1] "-- Rename the lowest level"
## [1] "-- rownames"
## [1] "-- taxa"
## [1] "Convert to taxonomy table"
## [1] "Combine OTU and Taxon matrix into Phyloseq object"
## [1] "Add the metadata as is"
```

```r
# Remove undesired genera
# B2 <- subset_taxa(B, !get("Genus") %in% c("uncultured", "Unknown"))

# Remove unwanted taxon names
taxa_to_remove <- c("uncultured", "Unknown")
B2 <- subset_taxa(B, !get("Genus") %in% taxa_to_remove)

# Extract relevant data from the phyloseq object
sample_data <- sample_data(B2)
otu_table <- otu_table(B2)
abundance <- as.vector(otu_table)

# Create a tibble with the extracted data
D <- tibble(
  Sample = rep(sample_data$Group, each = nrow(otu_table)),
  ASV = rep(rownames(otu_table), times = ncol(otu_table)),
  Abundance = abundance
) %>%
  group_by(Sample) %>%
  mutate(rank = rank(plyr::desc(Abundance))) %>%
   filter(Abundance > 15) %>%
  ungroup() %>%
  select(Sample, Abundance, ASV)

# Remove the Abundance column
D$Abundance <- NULL

# Rename the second column to "ASV"
names(D)[2] <- "ASV"
names(D)[1] <- "Direct"

# Convert data from long to wide format
E <- dcast(D, ASV ~ Direct)

# Define a binary function
binary_fun <- function(x) {
```

```r
  x[is.na(x)] <- 0
  ifelse(x > 0, 1, 0)
}

col = c("#00BA38","#619CFF","#F8766D")

# Apply the binary function to columns 2 to 4
temp_df <- apply(E[2:4], 2, binary_fun)
temp_df <- as.data.frame(temp_df)

# Create an UpSet plot
upset_plot <- upset(temp_df,
                    sets = colnames(temp_df),
                    sets.bar.color = (col),
                    order.by = "freq",
                    empty.intersections = "on",
                    mainbar.y.label = "Counts by Pattern of Conditions",
                    sets.x.label = "Counts by Condition",
                    matrix.color="blue",
                    mb.ratio = c(0.65, 0.35),
                    point.size= 2.75,
                    line.size = 1.25,
                    text.scale = 1.5
)


# Open a new PDF graphics device
# pdf(file = "UpSet_ITS.pdf", width=6.5,height=4.5)

# Print the UpSet plot
print(upset_plot)
```




```r
# Close the PDF device and save the plot to a file
# dev.off()
```

0.1.7 Pairwise comparison using PERMANOVA

Pairwise PERMANOVA (Permutational Multivariate Analysis of Variance) is a statistical method used in microbial community studies to examine differences between groups or treatments. It assesses the dissimilarity between samples, allowing for the comparison of multivariate data. This approach is useful to focus on specific group comparisons rather than comparing all groups simultaneously. It enables the investigation of the effects of specific treatments on microbial communities, helping to determine if there are significant differences in community composition between selected groups. By considering variation within and between groups, pairwise PERMANOVA offers a robust statistical assessment of dissimilarity, providing insights into community structure differences.

```r
metdat = as.data.frame(as.matrix(physeq.norm@sam_data))
dat = as.data.frame(t(as.data.frame(physeq.norm@otu_table)))
pairwise.adonis(dat, metdat$Group, sim.function = "vegdist",
                sim.method = "bray", p.adjust.m = "bonferroni",
                reduce = NULL, perm = 100000)
```

```
##                pairs Df  SumsOfSqs F.Model        R2    p.value p.adjusted sig
## 1 Control vs Plunge  1 0.05745805 1.125904 0.1233746 0.23120769  0.6936231
## 2   Control vs Rate  1 0.07640305 1.380438 0.1471614 0.03931961  0.1179588
## 3    Plunge vs Rate  1 0.08478784 1.653684 0.1713008 0.05527945  0.1658383
```

```r
rm(metdat, dat)
```

0.1.8 Top 10 samples

In microbiome analysis, identifying the top 10 bacteria in the top 10 family level and their corresponding percentages provides a snapshot of the microbial community's composition.

```r
# Merge reads by groups
AyBCode <- merge_samples(physeq, "Group", fun = sum)

## Normalised number of reads in percentage
standf = function(x) x / sum(x) * 100
AyBCode.percent = transform_sample_counts(AyBCode, standf)

##### unwanted taxon names
taxa_to_remove <- c("uncultured", "Unknown")
# Remove unwanted taxon names
AyBCode.percent.B <- subset_taxa(AyBCode.percent, !get("Family") %in% taxa_to_remove)
## Aggregate
AyBCode.percent.B <- aggregate_taxa(AyBCode.percent.B, "Family", verbose = TRUE)
```

```
## [1] "Remove taxonomic information below the target level"
## [1] "Mark the potentially ambiguous taxa"
## [1] "-- split"
## [1] "-- sum"
## [1] "Create phyloseq object"
## [1] "Remove ambiguous levels"
## [1] "-- unique"
## [1] "-- Rename the lowest level"
## [1] "-- rownames"
## [1] "-- taxa"
## [1] "Convert to taxonomy table"
## [1] "Combine OTU and Taxon matrix into Phyloseq object"
## [1] "Add the metadata as is"
```

```r
top10otus = names(sort(taxa_sums(AyBCode.percent.B), TRUE)[1:10])
taxtab10 = cbind(tax_table(AyBCode.percent.B), Family = NA)
taxtab10[top10otus, "Family"] <- as(tax_table(AyBCode.percent.B)[top10otus, "Family"],"character")
tax_table(AyBCode.percent.B) <- tax_table(taxtab10)

top10plot = prune_taxa(top10otus, AyBCode.percent.B)
print(top10plot@otu_table)
```

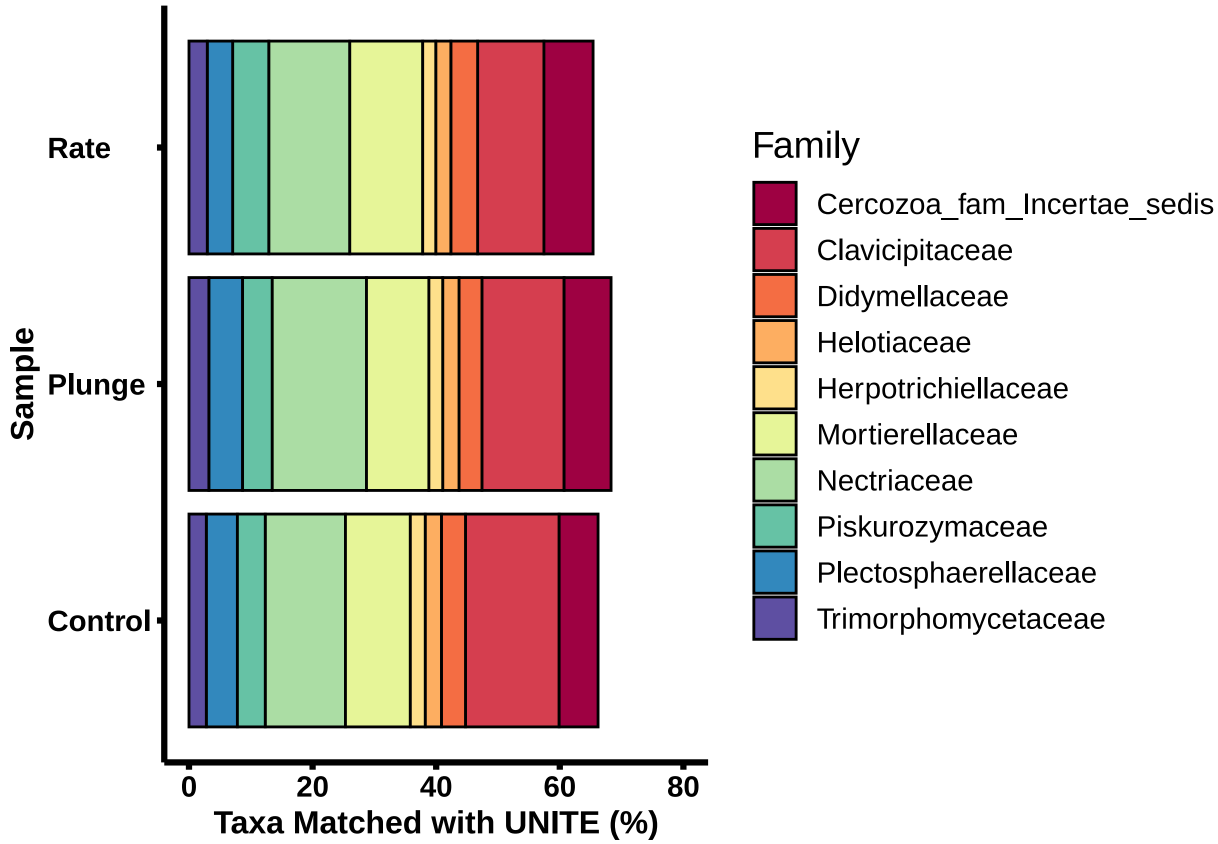
```
## OTU Table:          [10 taxa and 3 samples]
##                      taxa are rows
##                               Control    Plunge      Rate
## Didymellaceae                3.892923  3.704637  4.326121
## Herpotrichiellaceae          2.413237  2.236401  2.113900
## Helotiaceae                  2.622411  2.642670  2.444422
## Plectosphaerellaceae         5.027540  5.453886  4.105144
## Clavicipitaceae             15.154996 13.293230 10.731729
```

```
## Nectriaceae                 12.975410 15.262021 13.099461
## Piskurozymaceae              4.507682  4.787017  5.848620
## Trimorphomycetaceae          2.803656  3.216637  2.959132
## Mortierellaceae             10.498054 10.096468 11.804246
## Cercozoa_fam_Incertae_sedis  6.293547  7.596847  7.951049
```

```r
# Calculate the sum of each column
col_sums <- colSums(as.data.frame(top10plot@otu_table))

# Add a new row with the sums
top10plot.df <- rbind('SUM' = col_sums, as.data.frame(top10plot@otu_table))

# Print the dataframe
print(top10plot.df)
```

```
##                               Control    Plunge      Rate
## SUM                         66.189455 68.289814 65.383824
## Didymellaceae                3.892923  3.704637  4.326121
## Herpotrichiellaceae          2.413237  2.236401  2.113900
## Helotiaceae                  2.622411  2.642670  2.444422
## Plectosphaerellaceae         5.027540  5.453886  4.105144
## Clavicipitaceae             15.154996 13.293230 10.731729
## Nectriaceae                 12.975410 15.262021 13.099461
## Piskurozymaceae              4.507682  4.787017  5.848620
## Trimorphomycetaceae          2.803656  3.216637  2.959132
## Mortierellaceae             10.498054 10.096468 11.804246
## Cercozoa_fam_Incertae_sedis  6.293547  7.596847  7.951049
```

```r
top10.its <- plot_bar(top10plot, fill = "Family") + coord_flip() +
  ylab("Taxa Matched with UNITE (%)") + ylim(0, 80) +
  theme_classic() +
  theme(text = element_text(size=14, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.1),
        axis.line = element_line(colour = 'black', size = 1.1),
        axis.text.x = element_text(colour = "black", angle=0, size = 11, face="bold"),
        axis.text.y = element_text(angle=0, hjust=0, colour = "black", size = 11, face="bold"),
        axis.title.y = element_text(color="black", size=12,face="bold"),
        axis.title.x = element_text(color="black", size=12,face="bold"),
        legend.position = "right") +
  scale_color_brewer(palette="Spectral")+
  scale_fill_brewer(palette="Spectral")

# pdf(file = "top10.its.pdf", width = 6.75, height = 5)
top10.its
```

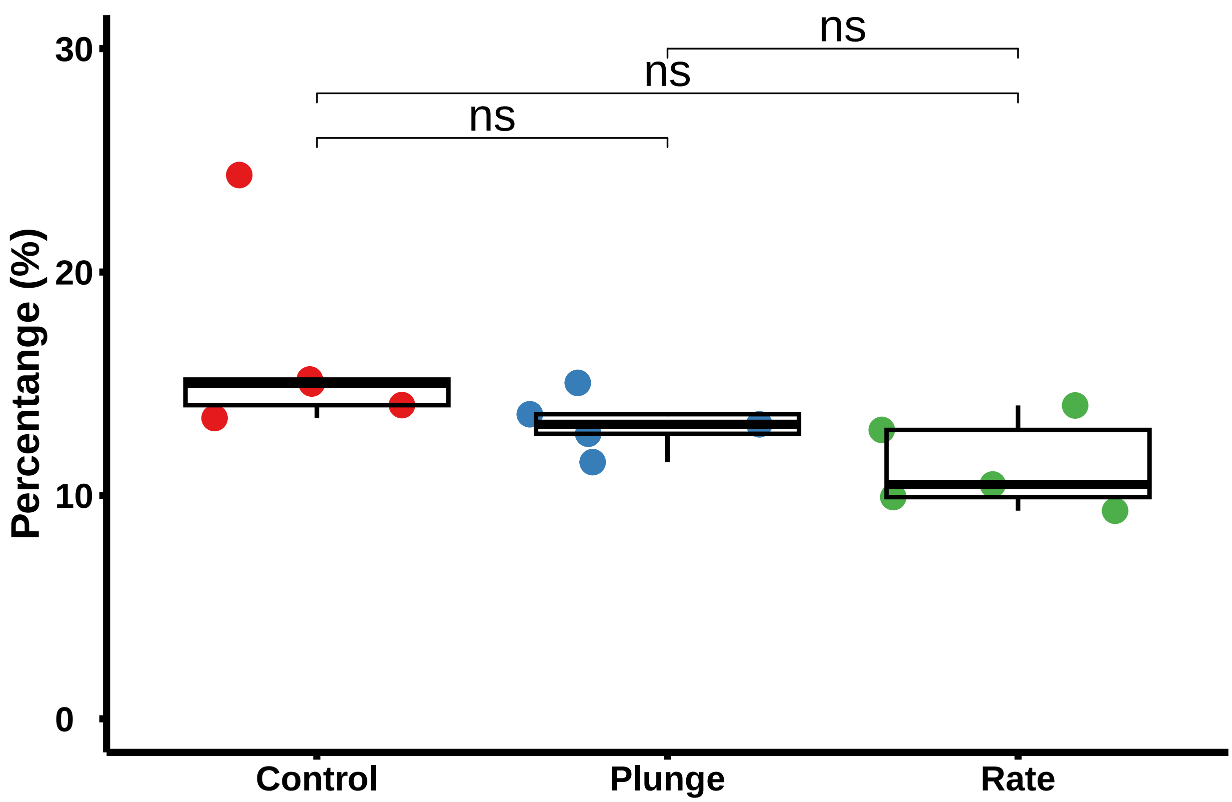



```r
# Close the PDF device and save the plot to a file
# dev.off()

# rm(physeq.ori, physeq.rich, AyBCode,
#    standf, AyBCode.percent, top10otus,
#    taxtab10, top10plot, top10.ori, top10.rich)
```

0.1.9 Plot the graph for Moraxellaceae, Enriched

```r
## Normalised number of reads in percentage
AyBCode.percent = transform_sample_counts(physeq.norm, standf)
physeq.a.genus <- subset_taxa(AyBCode.percent, Family == "Clavicipitaceae")

# Calculate the total abundance of Clavicipitaceae for each sample
meta = data.frame(AyBCode.percent@sam_data)
otudf = as.data.frame(t(as.data.frame(physeq.a.genus@otu_table)))
meta$Clavicipitaceae = rowSums(otudf)

stat.test1 <- meta %>%
  t_test(Clavicipitaceae ~ Group) %>%
  adjust_pvalue(method = "bonferroni") %>%
  add_significance()

# Plot a graph of the abundance of Clavicipitaceae for each sample grouped by Group:
```

```
Clavicipitaceae.Rich <- ggplot(subset(meta, Group %in% c("Control","Plunge","Rate")),
                              aes(x = Group, y = Clavicipitaceae, colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +
  labs(x = "", y = "Percentange (%)") +
  stat_pvalue_manual(stat.test1,
                   y.position = c(26, 28, 30),
                   label = "p.adj.signif",
                   face="bold",
                   size = 6,
                   linetype = 1,
                   tip.length = 0.02,
                   inherit.aes = FALSE) +
  scale_y_continuous(limits=c(0, 30), breaks = c(0, 10, 20, 30)) +
  theme(text = element_text(size=18, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.25),
        axis.line = element_line(colour = 'black', size = 1.25),
        axis.text.x = element_text(colour = "black",
                                   angle=0,
                                   size = 13, face="bold"),
        axis.text.y = element_text(angle=0, hjust=0, colour = "black",
                                   size = 13, face="bold"),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "none") +
  scale_color_brewer(palette="Set1")+
  scale_fill_brewer(palette="Set1")

# pdf(file = "Clavicipitaceae.ITS.pdf", width = 6, height = 5)
Clavicipitaceae.Rich
```

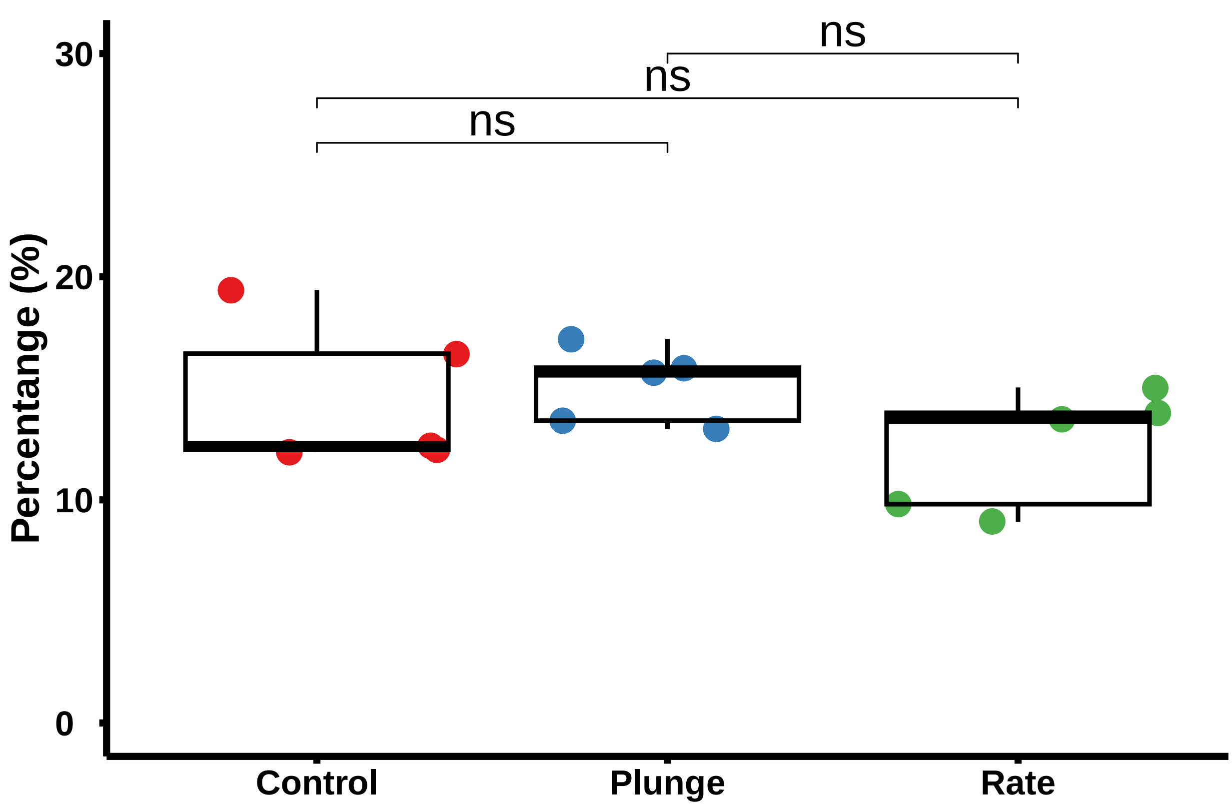



```
# Close the PDF device and save the plot to a file
# dev.off()

# Clean up by removing objects that are no longer needed
rm(physeq.a.genus, meta, otudf, AyBCode.percent)
```

0.1.10 Plot the graph for Nectriaceae, Enriched

```
## Normalised number of reads in percentage
AyBCode.percent = transform_sample_counts(physeq.norm, standf)
physeq.a.genus <- subset_taxa(AyBCode.percent, Family == "Nectriaceae")

# Calculate the total abundance of Nectriaceae for each sample
meta = data.frame(AyBCode.percent@sam_data)
otudf = as.data.frame(t(as.data.frame(physeq.a.genus@otu_table)))
meta$Nectriaceae = rowSums(otudf)

stat.test1 <- meta %>%
  t_test(Nectriaceae ~ Group) %>%
  adjust_pvalue(method = "bonferroni") %>%
  add_significance()

# Plot a graph of the abundance of Nectriaceae for each sample grouped by Group:
Nectriaceae.Rich <- ggplot(subset(meta, Group %in% c("Control","Plunge","Rate")),
```

```
                              aes(x = Group, y = Nectriaceae, colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +
  labs(x = "", y = "Percentange (%)") +
  stat_pvalue_manual(stat.test1,
                     y.position = c(26, 28, 30),
                     label = "p.adj.signif",
                     face="bold",
                     size = 6,
                     linetype = 1,
                     tip.length = 0.02,
                     inherit.aes = FALSE) +
  scale_y_continuous(limits=c(0, 30), breaks = c(0, 10, 20, 30)) +
  theme(text = element_text(size=18, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.25),
        axis.line = element_line(colour = 'black', size = 1.25),
        axis.text.x = element_text(colour = "black",
                                   angle=0,
                                   size = 13, face="bold"),
        axis.text.y = element_text(angle=0, hjust=0, colour = "black",
                                   size = 13, face="bold"),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "none") +
  scale_color_brewer(palette="Set1")+
  scale_fill_brewer(palette="Set1")

# pdf(file = "Nectriaceae.ITS.pdf", width = 6, height = 5)
Nectriaceae.Rich
```




```r
# Close the PDF device and save the plot to a file
# dev.off()

# Clean up by removing objects that are no longer needed
rm(physeq.a.genus, meta, otudf, AyBCode.percent)
```

```r
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## time zone: Europe/London
## tzcode source: internal
##
```

```
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] pairwiseAdonis_0.4.1        cluster_2.1.4
##  [3] vegan_2.6-4                 lattice_0.21-9
##  [5] permute_0.9-7               qiime2R_0.99.6
##  [7] microbiome_1.22.0           DESeq2_1.40.2
##  [9] SummarizedExperiment_1.30.2 Biobase_2.60.0
## [11] MatrixGenerics_1.12.3       matrixStats_1.0.0
## [13] GenomicRanges_1.52.1        GenomeInfoDb_1.36.4
## [15] IRanges_2.34.1              S4Vectors_0.38.2
## [17] BiocGenerics_0.46.0         phyloseq_1.44.0
## [19] RColorBrewer_1.1-3          plyr_1.8.9
## [21] UpSetR_1.4.0                reshape2_1.4.4
## [23] purrr_1.0.2                 rstatix_0.7.2
## [25] dplyr_1.1.3                 ggpubr_0.6.0
## [27] ggplot2_3.4.4
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-7            gridExtra_2.3           rlang_1.1.1
##  [4] magrittr_2.0.3          ade4_1.7-22             compiler_4.3.2
##  [7] mgcv_1.9-0              vctrs_0.6.3             stringr_1.5.0
## [10] pkgconfig_2.0.3         crayon_1.5.2            fastmap_1.1.1
## [13] backports_1.4.1         XVector_0.40.0          labeling_0.4.3
## [16] utf8_1.2.3              rmarkdown_2.25          xfun_0.40
## [19] zlibbioc_1.46.0         jsonlite_1.8.7          biomformat_1.28.0
## [22] rhdf5filters_1.12.1     DelayedArray_0.26.7     Rhdf5lib_1.22.1
## [25] BiocParallel_1.34.2     broom_1.0.5             parallel_4.3.2
## [28] R6_2.5.1                stringi_1.7.12          zCompositions_1.4.1
## [31] rpart_4.1.21            car_3.1-2               Rcpp_1.0.11
## [34] iterators_1.0.14        knitr_1.44              base64enc_0.1-3
## [37] nnet_7.3-19             Matrix_1.6-1.1          splines_4.3.2
## [40] igraph_1.5.1            tidyselect_1.2.0        rstudioapi_0.15.0
## [43] abind_1.4-5             yaml_2.3.7              codetools_0.2-19
## [46] tibble_3.2.1            withr_2.5.2             evaluate_0.22
## [49] Rtsne_0.16              foreign_0.8-85          survival_3.5-7
## [52] Biostrings_2.68.1       pillar_1.9.0            carData_3.0-5
## [55] DT_0.30                 checkmate_2.2.0         foreach_1.5.2
## [58] NADA_1.6-1.1            generics_0.1.3          RCurl_1.98-1.12
## [61] truncnorm_1.0-9         munsell_0.5.0           scales_1.3.0
## [64] glue_1.6.2              Hmisc_5.1-1             tools_4.3.2
## [67] data.table_1.14.8       locfit_1.5-9.8          ggsignif_0.6.4
## [70] rhdf5_2.44.0            grid_4.3.2              tidyr_1.3.0
## [73] ape_5.7-1               colorspace_2.1-0        nlme_3.1-163
## [76] GenomeInfoDbData_1.2.10 htmlTable_2.4.1         Formula_1.2-5
## [79] cli_3.6.1               fansi_1.0.4             S4Arrays_1.0.6
## [82] gtable_0.3.4            digest_0.6.33           farver_2.1.1
## [85] htmlwidgets_1.6.2       htmltools_0.5.6         multtest_2.56.0
## [88] lifecycle_1.0.4         MASS_7.3-60
```