Cryopreservation of a soil microbiome using a Stirling Cycle approach – a genomic (16s data) assessment

2023-11-01

Cyropreserve CABI

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 of 16S and ITS rRNA. 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.

Load the required packages

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

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

# 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")
```

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 visualisation 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.

```
# Convert qiime2 to phyloseq format
physeq <- qza_to_phyloseq(
  features = "qiime2/430_327_213_table-with-phyla-no-mitochondria-no-chloroplast.qza", # table.qza
  # tree = "inst/artifacts/2020.2_moving-pictures/rooted-tree.qza",
  taxonomy = "qiime2/430_327_213_taxonomy.qza",
  metadata = "16s-meta-data.txt"
)
physeq ## confirm the object
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 14243 taxa and 29 samples ]
## sample_data() Sample Data:       [ 29 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 14243 taxa by 7 taxonomic ranks ]
```

import data and subgroup the data

Normalising the number of reads in each sample is an important step in analysing sequencing data, as it helps to remove any biases introduced by differences in sequencing depth across samples. One commonly used method for normalisation is to scale the read counts by the median sequencing depth.

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

total <- median(sample_sums(physeq)) # Calculate the median sequencing depth
standf <- function(x, t = total) round(t * (x / sum(x))) # Define a scaling function
physeq.norm <- transform_sample_counts(physeq, standf) # Normalise the sample counts using the scaling
```

Sub-grouping

Separate analysis is necessary for the Original and Enriched experiments since the data contained in each subset differs and requires distinct examination.

```
## Subgroup
physeq.norm.ori <- subset_samples(physeq.norm, Comparison=="Original")
physeq.norm.rich <- subset_samples(physeq.norm, Comparison=="Enriched")

## (3A) Merge the replicate samples for each Group
physeq.norm.ori.group = merge_samples(physeq.norm.ori, "Group2") # Sum between replicate samples
sample_data(physeq.norm.ori.group)$Group2 <- rownames(sample_data(physeq.norm.ori.group))

physeq.norm.rich.group = merge_samples(physeq.norm.rich, "Group2") # Sum between replicate samples
sample_data(physeq.norm.rich.group)$Group2 <- rownames(sample_data(physeq.norm.rich.group))
```

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.

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

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.04166808
## Run 1 stress 0.05276263
## Run 2 stress 0.05276263
## Run 3 stress 0.04166808
## ... New best solution
## ... Procrustes: rmse 7.35552e-06  max resid 1.805382e-05
## ... Similar to previous best
## Run 4 stress 0.04825551
## Run 5 stress 0.05451056
## Run 6 stress 0.05088141
## Run 7 stress 0.04394385
## Run 8 stress 0.04825556
## Run 9 stress 0.04825551
## Run 10 stress 0.04825552
## Run 11 stress 0.04857385
## Run 12 stress 0.04166809
## ... Procrustes: rmse 2.489648e-05  max resid 5.562324e-05
## ... Similar to previous best
## Run 13 stress 0.05912191
## Run 14 stress 0.04561031
## Run 15 stress 0.05741067
## Run 16 stress 0.04166809
## ... Procrustes: rmse 2.774898e-05  max resid 6.561634e-05
## ... Similar to previous best
## Run 17 stress 0.04561031
## Run 18 stress 0.04166811
## ... Procrustes: rmse 5.324361e-05  max resid 0.0001254059
## ... Similar to previous best
## Run 19 stress 0.05364061
## Run 20 stress 0.0494635
## *** Best solution repeated 4 times
```

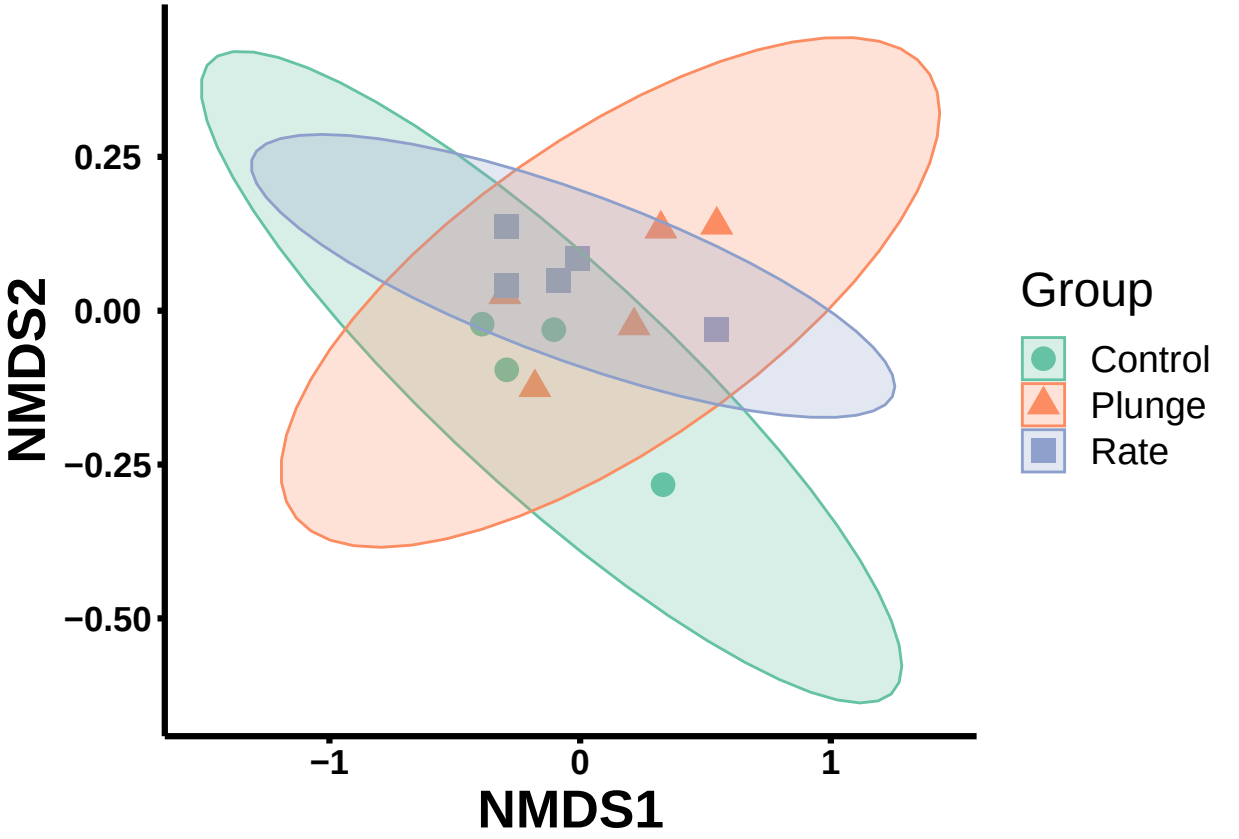
```
nmds.rich <- ordinate(physeq = physeq.norm.rich, method = "NMDS", distance = "bray")
```

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

```
## Run 0 stress 0.08844091
## Run 1 stress 0.09117322
## Run 2 stress 0.09575812
## Run 3 stress 0.0882692
## ... New best solution
## ... Procrustes: rmse 0.01373221  max resid 0.03591984
## Run 4 stress 0.08795252
## ... New best solution
## ... Procrustes: rmse 0.05259181  max resid 0.142621
## Run 5 stress 0.09117333
## Run 6 stress 0.08823141
## ... Procrustes: rmse 0.02173455  max resid 0.06063795
## Run 7 stress 0.08795257
## ... Procrustes: rmse 6.243234e-05  max resid 0.000128365
## ... Similar to previous best
## Run 8 stress 0.09361876
## Run 9 stress 0.08835016
## ... Procrustes: rmse 0.02329398  max resid 0.06210878
## Run 10 stress 0.0882314
## ... Procrustes: rmse 0.02173521  max resid 0.06065356
## Run 11 stress 0.2388587
## Run 12 stress 0.09361869
## Run 13 stress 0.09117349
## Run 14 stress 0.09171857
## Run 15 stress 0.09117318
## Run 16 stress 0.08795251
## ... New best solution
## ... Procrustes: rmse 1.636877e-05  max resid 2.716283e-05
## ... Similar to previous best
## Run 17 stress 0.08878583
## Run 18 stress 0.08799876
## ... Procrustes: rmse 0.05829963  max resid 0.1473168
## Run 19 stress 0.0874603
## ... New best solution
## ... Procrustes: rmse 0.02993509  max resid 0.1044524
## Run 20 stress 0.09117318
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     19: stress ratio > sratmax
##      1: scale factor of the gradient < sfgrmin
```

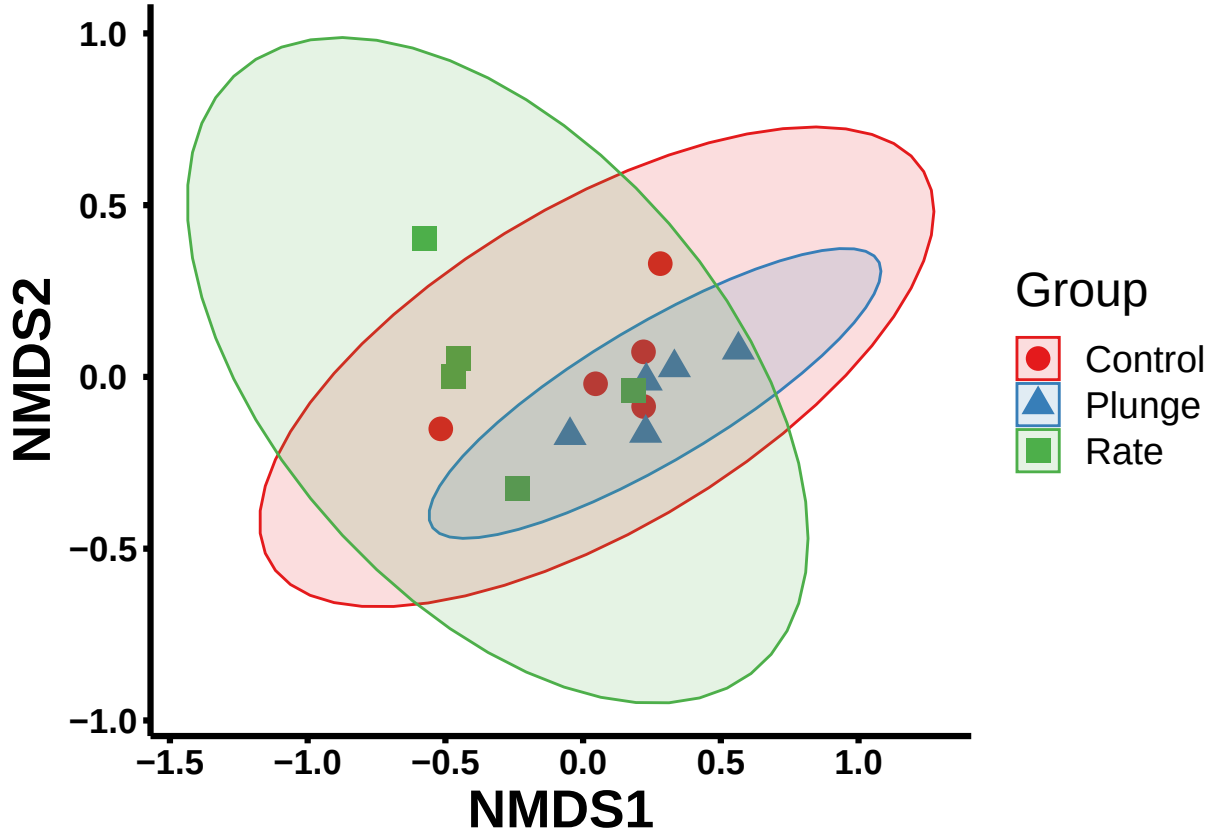
```r
plot_ordination(
  physeq = physeq.norm.ori,
  ordination = nmds.ori,
  # title = "NMDS",
  color = "Group",
  shape = "Group") +
  # scale_x_discrete(name ="NMDS1 ()") +
  # scale_y_discrete(name ="NMDS2 ()") +
  theme_classic() +
  geom_point(aes(color = Group), alpha = 1, size = 4) +
  theme(text = 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")) +
  stat_ellipse(geom = "polygon", type="norm", alpha=0.25, aes(fill = Group)) + # polygon, path, point
  scale_color_brewer(palette="Set2")+
  scale_fill_brewer(palette="Set2")
```




```
plot_ordination(
  physeq = physeq.norm.rich,
  ordination = nmds.rich,
  # title = "PCoA",
  color = "Group",
  shape = "Group") +
  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")) +
  stat_ellipse(geom = "polygon", type="norm", alpha=0.15, aes(fill=Group))+
```

```
  scale_color_brewer(palette="Set1")+
  scale_fill_brewer(palette="Set1")
```

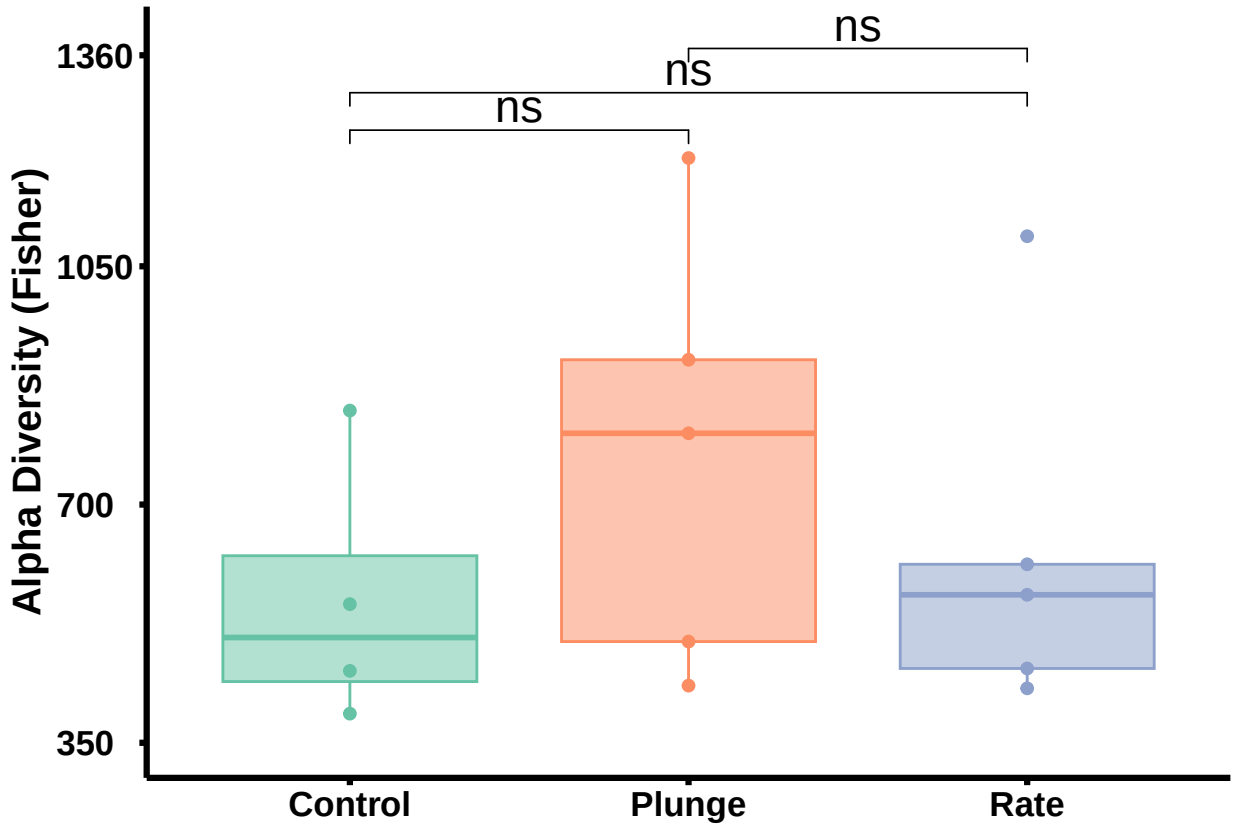


Alpha diversity

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.

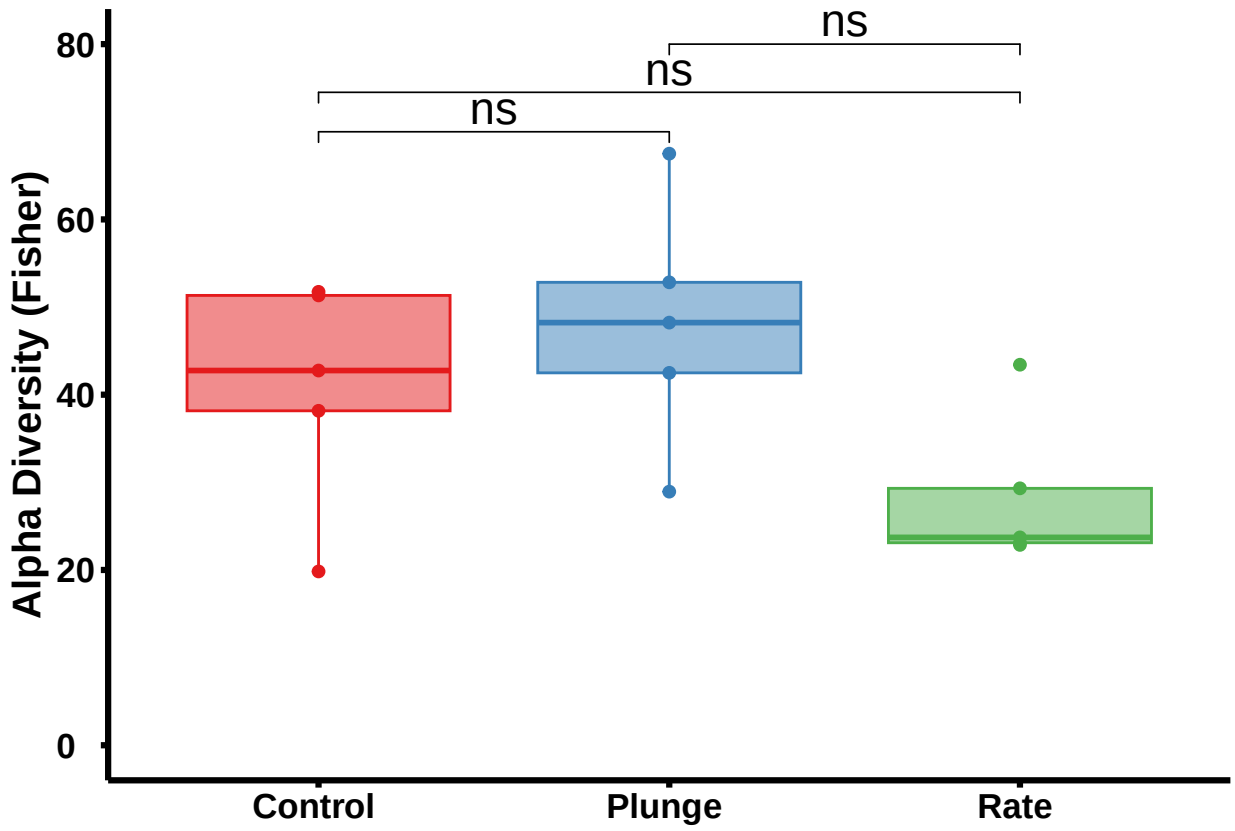
```
par(mfrow=c(1,2))
# original
tab = cbind(x = sample_data(physeq.norm.ori),
            y = estimate_richness(physeq.norm.ori, measures = 'Fisher'))

stat.test <- tab %>%
  # group_by(Neutrophils, GROUP1) %>%
  t_test(Fisher ~ x.Group) %>%
```

```r
  adjust_pvalue(method = "bonferroni") %>%
  add_significance()

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(1250, 1305, 1370),
                     label = "p.adj.signif",
                     face="bold",
                     size = 6,
                     linetype = 1,
                     tip.length = 0.02,
                     inherit.aes = FALSE) +
  scale_y_continuous(limits=c(350, 1380), breaks = c(350, 700, 1050, 1360)) +
  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") +
  scale_color_brewer(palette="Set2")+
  scale_fill_brewer(palette="Set2")
```




```
# Enriched
tab = cbind(x = sample_data(physeq.norm.rich),
            y = estimate_richness(physeq.norm.rich, measures = 'Fisher'))

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

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(70, 74.5, 80),
                     label = "p.adj.signif",
                     face="bold",
                     size = 6,
                     linetype = 1,
                     tip.length = 0.02,
                     inherit.aes = FALSE) +
  scale_y_continuous(limits=c(0, 80), breaks = c(0, 20, 40, 60, 80)) +
```

```
  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") +
  scale_color_brewer(palette="Set1")+
  scale_fill_brewer(palette="Set1")
```



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.

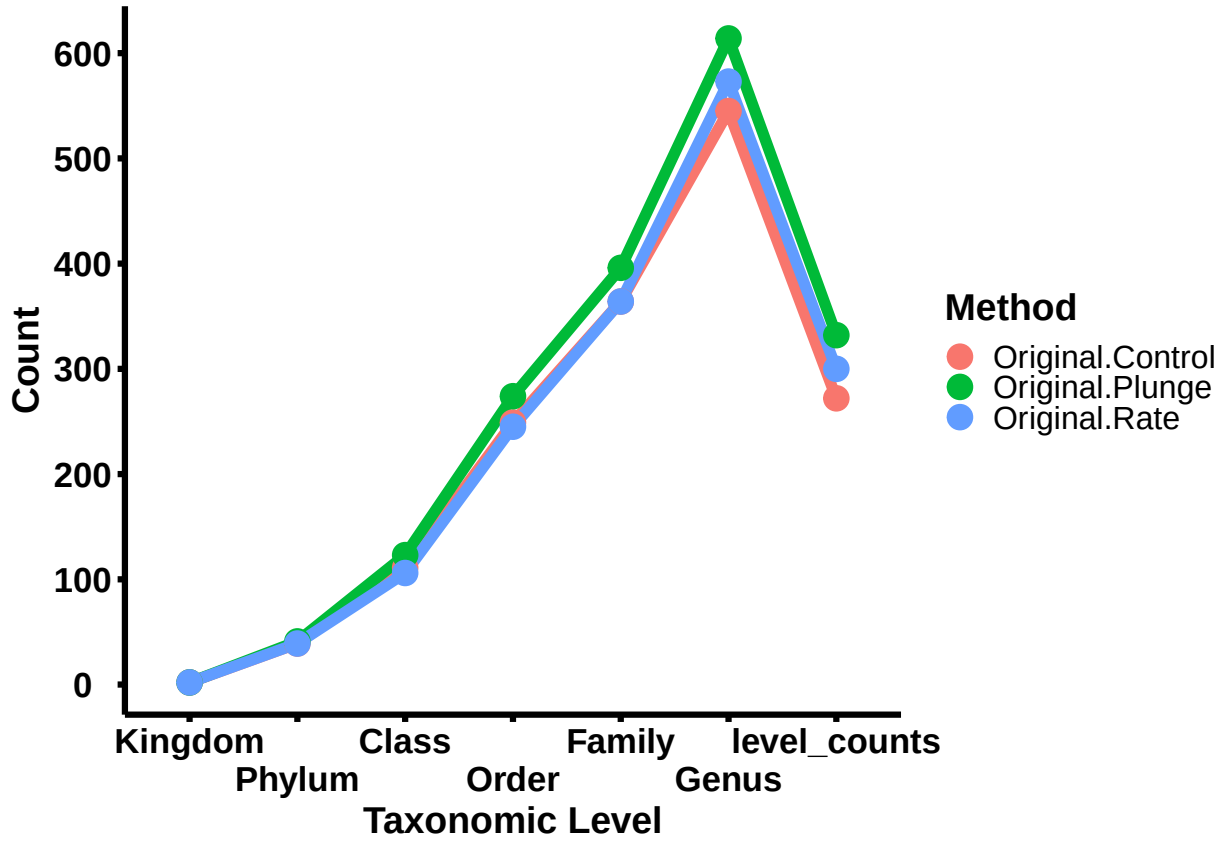
```
# Create an empty list to store genus-level abundance data for each taxonomic level
gentab_levels <- list()

# Set observation threshold
observationThreshold <- 1

# Define the taxonomic levels
```

```r
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.ori.group)[, level])

  # calculate the abundance of each genus within each sample
  gentab <- apply(otu_table(physeq.norm.ori.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)[1:6] <- genus_levels
B2$name <- rownames(B2)

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

```
##                  Kingdom Phylum Class Order Family Genus level_counts
## Original.Control       2     39   111   249    364   545          272
## Original.Plunge        2     41   123   274    396   614          332
## Original.Rate          2     39   106   245    364   573          300
##                           name
## Original.Control Original.Control
## Original.Plunge   Original.Plunge
## Original.Rate       Original.Rate
```

```r
# Clean up by removing unnecessary objects
rm(gentab_levels, BB)

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

```r
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"),
    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")
  ) +
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  scale_y_continuous(breaks=seq(0,600,by=100))
```



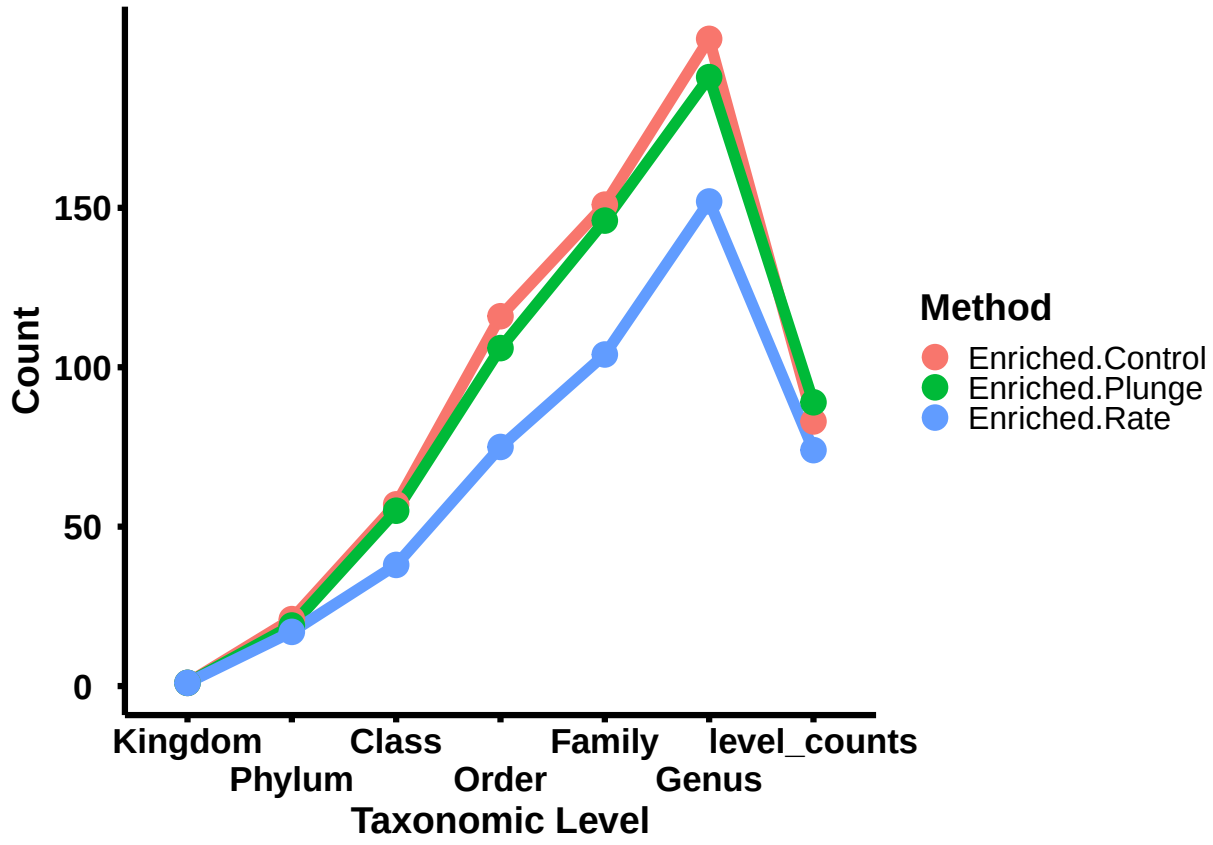

```r
# Create an empty list to store genus-level abundance data for each taxonomic level
gentab_levels <- list()

# Set observation threshold
observationThreshold <- 1
```

```r
# 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.rich.group)[, level])

  # calculate the abundance of each genus within each sample
  gentab <- apply(otu_table(physeq.norm.rich.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)[1:6] <- genus_levels
B2$name <- rownames(B2)

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

```
##                   Kingdom Phylum Class Order Family Genus level_counts
## Enriched.Control        1     21    57   116    151   203           83
## Enriched.Plunge         1     19    55   106    146   191           89
## Enriched.Rate           1     17    38    75    104   152           74
##                              name
## Enriched.Control  Enriched.Control
## Enriched.Plunge    Enriched.Plunge
## Enriched.Rate        Enriched.Rate
```

```r
# Clean up by removing unnecessary objects
rm(gentab_levels, BB)

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

```
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"),
    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")
  ) +
  scale_x_discrete(guide = guide_axis(n.dodge=2)) +
  scale_y_continuous(breaks=seq(0,150,by=50))
```



Pairwise comparison using PERMANOVA

Pairwise comparison using PERMANOVA (Permutational Multivariate Analysis of Variance) is a statistical method commonly used in ecological and microbial community studies to examine differences between multiple groups or treatments. It assesses the dissimilarity or distance matrix between samples, allowing for the comparison of multivariate data, such as microbial community composition. By testing the null hypothesis

of no difference among groups, PERMANOVA provides a p-value that indicates the significance of observed differences.

This approach is particularly useful when researchers want to focus on specific group comparisons of interest rather than comparing all groups simultaneously. Pairwise PERMANOVA enables the investigation of the effects of specific treatments or factors on microbial communities. It helps determine if there are significant differences in community composition between selected groups, shedding light on the influence of particular variables on microbial diversity. By considering variation within and between groups, pairwise PERMANOVA offers a robust statistical assessment of dissimilarity, accounting for potential confounding factors and providing insights into community structure differences.

```
##                 pairs Df SumsOfSqs  F.Model        R2     p.value p.adjusted
## 1 Control  vs Plunge   1 0.2135437 2.383923 0.2295783 0.024039760 0.07211928
## 2   Control  vs Rate   1 0.1787888 1.287497 0.1386269 0.269677303 0.80903191
## 3    Plunge  vs Rate   1 0.3600888 3.368058 0.2962738 0.008079919 0.02423976
##   sig
## 1
## 2
## 3   .
```

```
##               pairs Df  SumsOfSqs  F.Model        R2    p.value p.adjusted sig
## 1 Control vs Plunge  1 0.05666013 1.131139 0.1391120 0.22195778  0.6658733
## 2   Control vs Rate  1 0.06639183 1.461060 0.1726805 0.06287937  0.1886381
## 3    Plunge vs Rate  1 0.05134752 1.223352 0.1326364 0.12777872  0.3833362
```

Comparisons using DESeq2

DESeq2 is a popular tool for differential abundance analysis in 16S data. It compares different experimental conditions to identify significantly different taxa. By using statistical models and accounting for data variability, DESeq2 accurately detects differentially abundant taxa and controls false positives. This tool provides insights into microbial composition within sample groups and helps understand the microbiome's response to various factors. Thus, DESeq2 is a robust tool for identifying key microbial taxa associated with specific conditions.

```
## Subset the data
# Create an empty data frame to store the results
result_df <- data.frame(Comparison = character(),
                        Group = character(),
                        Type = character(),
                        Lost = integer())

# Define the combinations of groups to remove and their corresponding types
combinations <- list(
  list(Comparison = "Control vs Plunge", Group = "Control", Remove_Group = "Rate"),
  list(Comparison = "Plunge vs Rate", Group = "Plunge", Remove_Group = "Control"),
  list(Comparison = "Control vs Rate", Group = "Control", Remove_Group = "Plunge")
)

# Define the types of data
data_types <- list(
  list(Type = "Enriched", Data = physeq.norm.rich),
  list(Type = "Original", Data = physeq.norm.ori)
)
```

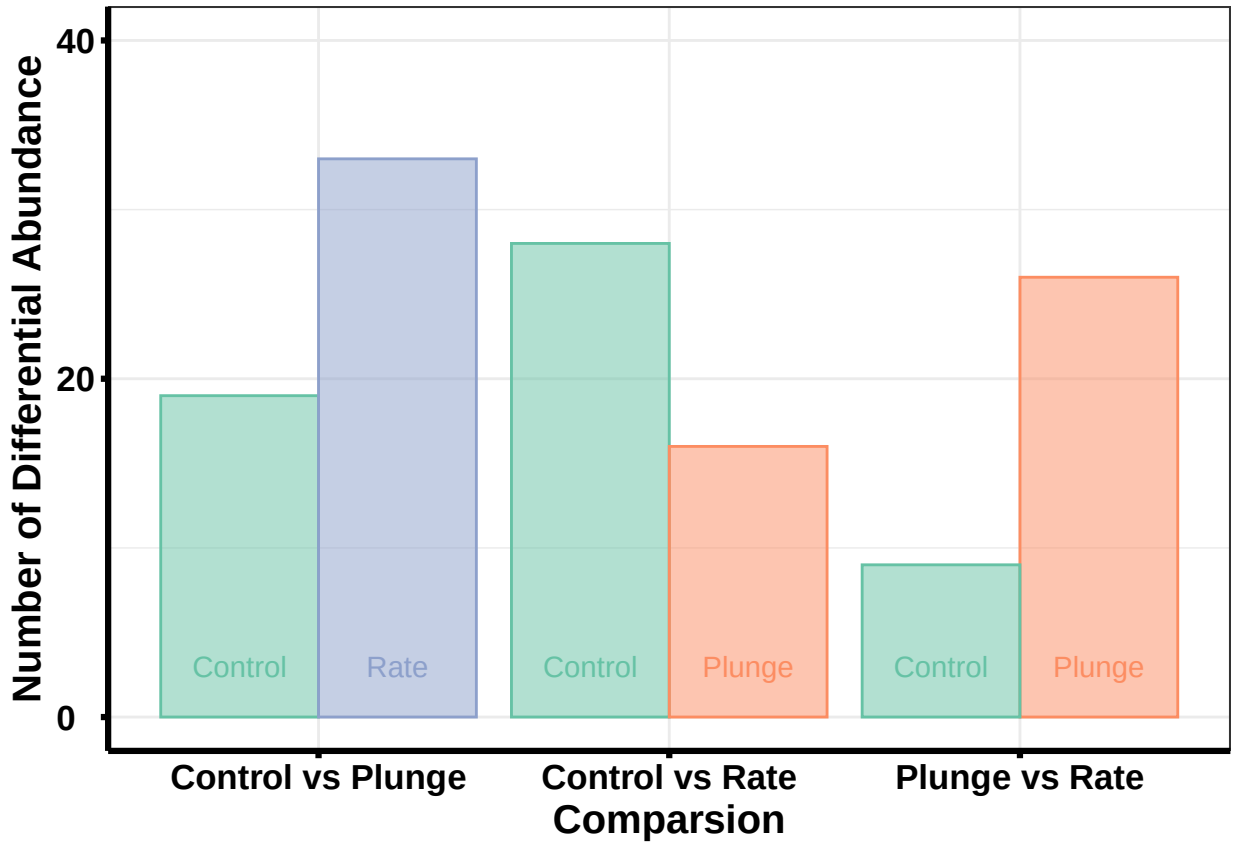
```r
# Iterate over each data type
for (i in 1:length(data_types)) {
  data_type <- data_types[[i]]
  data <- data_type$Data

  # Iterate over each combination
  for (j in 1:length(combinations)) {
    comp <- combinations[[j]]

    # Subset the samples
    physeq.norm.cp <- subset_samples(data, !(Group %in% comp$Remove_Group))

    # Run DESeq2 analysis
    diagdds <- phyloseq_to_deseq2(physeq.norm.cp, ~ Group)
    diagdds <- DESeq(diagdds, test = "Wald", fitType = "parametric")
    res <- results(diagdds, cooksCutoff = FALSE)
    sigtab <- cbind(as(res, "data.frame"), as(tax_table(physeq.norm.cp)[rownames(res), ], "matrix"))

    # Filter for significant features
    List2 <- filter(sigtab, padj < 0.05 & log2FoldChange < -1)
    List2.num <- nrow(List2)
    List3 <- filter(sigtab, padj < 0.05 & log2FoldChange > 1)
    List3.num <- nrow(List3)

    # Add the results to the main data frame
    result_df <- rbind(result_df,
                       data.frame(
                         Comparison = comp$Comparison,
                         Group = comp$Group,
                         Type = data_type$Type,
                         Lost = List2.num
                       ),
                       data.frame(
                         Comparison = comp$Comparison,
                         Group = comp$Remove_Group,
                         Type = data_type$Type,
                         Lost = List3.num
                       )
    )
  }
}

# Print the final result data frame
print(result_df)
```

```
##           Comparison   Group     Type Lost
## 1  Control vs Plunge Control Enriched   22
## 2  Control vs Plunge    Rate Enriched   38
## 3     Plunge vs Rate  Plunge Enriched   22
## 4     Plunge vs Rate Control Enriched   38
## 5    Control vs Rate Control Enriched   22
## 6    Control vs Rate  Plunge Enriched   38
## 7  Control vs Plunge Control Original   19
```

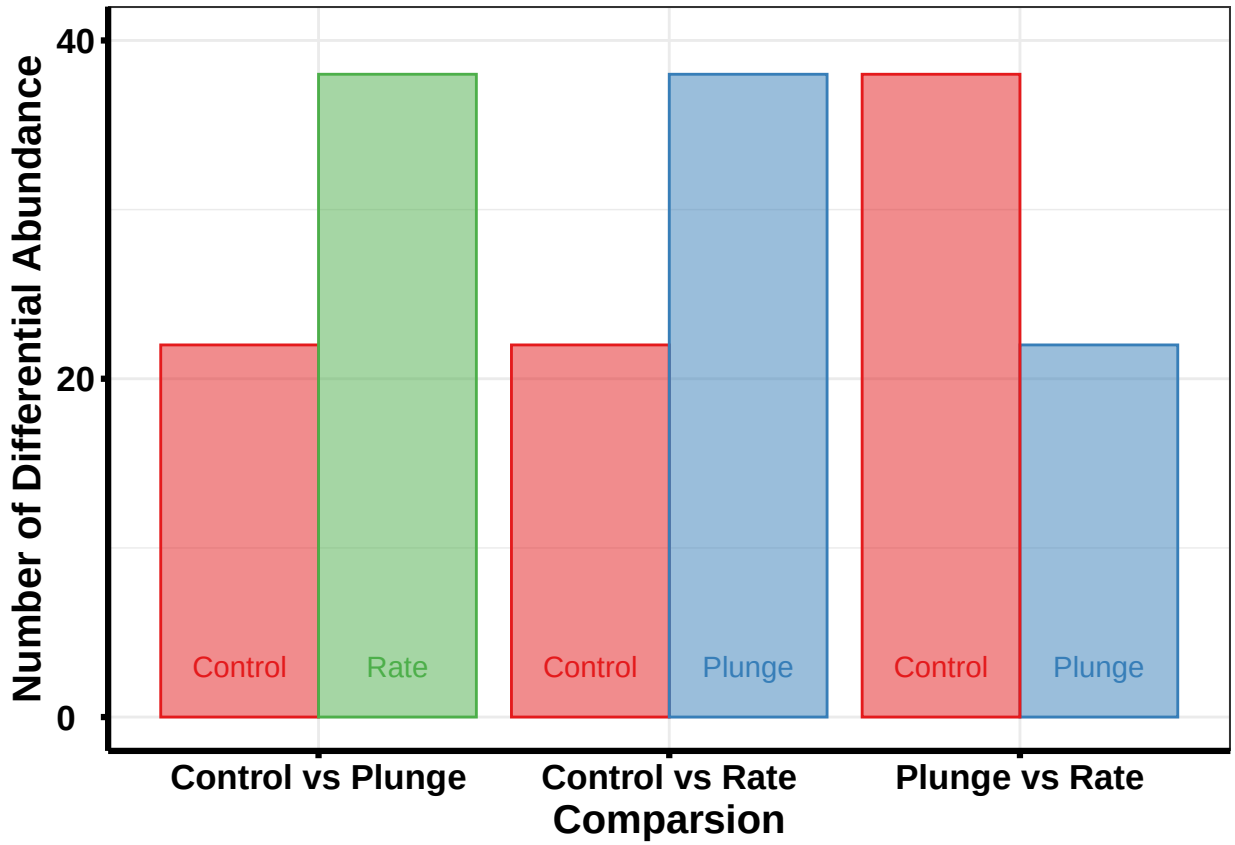
```
## 8  Control vs Plunge    Rate Original   33
## 9     Plunge vs Rate  Plunge Original   26
## 10    Plunge vs Rate Control Original    9
## 11   Control vs Rate Control Original   28
## 12   Control vs Rate  Plunge Original   16
```

visualise the differential ASVs

```
Ori = subset(result_df, Type =="Original")
Rich = subset(result_df, Type =="Enriched")

### Original ###
ggplot(Ori, aes(x = factor(Comparison), y = Lost, fill = Group, colour = Group)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.5)  +
  # geom_errorbar(aes(ymin=target, ymax=target), position = position_dodge(0.9), width = 0.25,
  #               show.legend = FALSE) +
  labs(x="Comparsion", y="Number of Differential Abundance") +
  theme_bw() +
  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"),
        axis.title.x = element_text(color="black", size=15,face="bold"),
        legend.position = "none") +
  scale_y_continuous(limits=c(0, 40), breaks = c(0, 20, 40)) +
  geom_text(aes(label=Group, y = 3),
            position = position_dodge(0.90),
            show.legend = FALSE) +
  scale_fill_brewer(palette = "Set2") +
  scale_color_brewer(palette = "Set2")
```




```
### Enriched ###
ggplot(Rich, aes(x = factor(Comparison), y = Lost, fill = Group, colour = Group)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.5)  +
  # geom_errorbar(aes(ymin=target, ymax=target), position = position_dodge(0.9), width = 0.25,
  #               show.legend = FALSE) +
  labs(x="Comparsion", y="Number of Differential Abundance") +
  theme_bw() +
  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"),
        axis.title.x = element_text(color="black", size=15,face="bold"),
        legend.position = "none") +
  scale_y_continuous(limits=c(0, 40), breaks = c(0, 20, 40)) +
  geom_text(aes(label=Group, y = 3),
            position = position_dodge(0.90),
            show.legend = FALSE) +
  scale_fill_brewer(palette = "Set1") +
  scale_color_brewer(palette = "Set1")
```

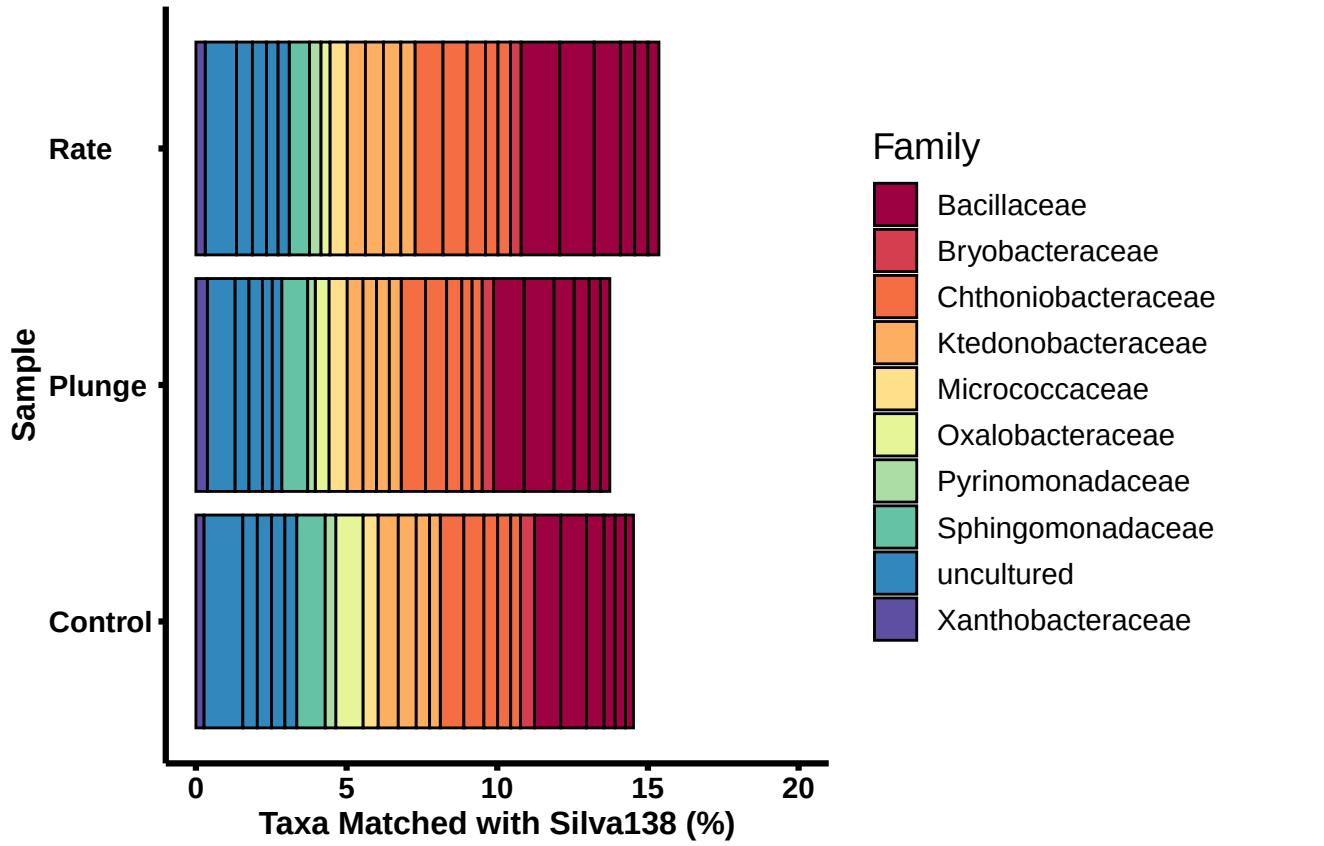


Top 10 samples

```
# Merge reads by groups (Original)
physeq.ori <- subset_samples(physeq, Comparison=="Original")
AyBCode <- merge_samples(physeq.ori, "Group", fun = sum)

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

top10otus = names(sort(taxa_sums(AyBCode.percent), TRUE)[1:26])
taxtab10 = cbind(tax_table(AyBCode.percent), Family = NA)
taxtab10[top10otus, "Family"] <- as(tax_table(AyBCode.percent)[top10otus, "Family"],"character")
tax_table(AyBCode.percent) <- tax_table(taxtab10)

top10plot = prune_taxa(top10otus, AyBCode.percent)
top10plot
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 26 taxa and 3 samples ]
## sample_data() Sample Data:       [ 3 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 26 taxa by 8 taxonomic ranks ]
```

```
title = "The 10 most abundant taxa (16s - Original)"

plot_bar(top10plot, fill = "Family") + coord_flip() +
  ylab("Taxa Matched with Silva138 (%)") + ylim(0, 20) +
  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")
```

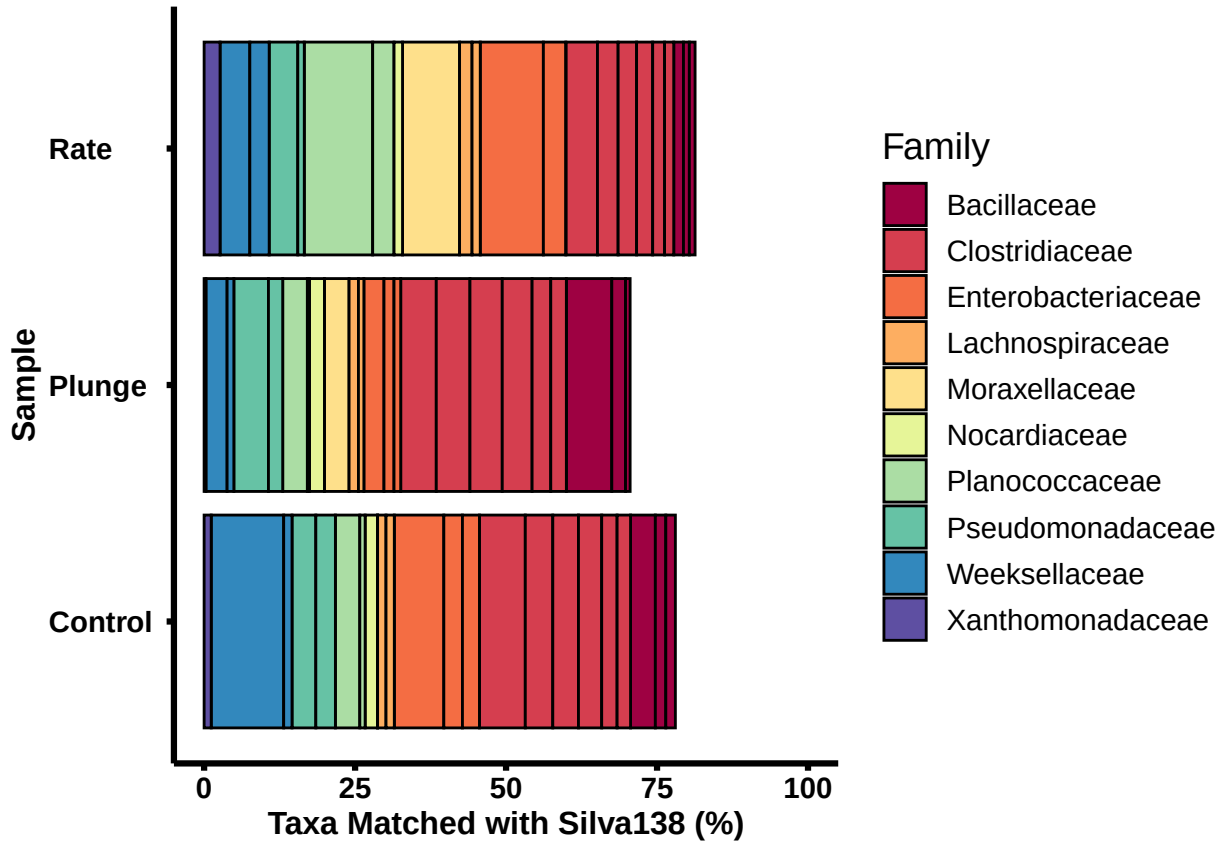



```
# Merge reads by groups (Enriched)
physeq.rich <- subset_samples(physeq, Comparison=="Enriched")
AyBCode <- merge_samples(physeq.rich, "Group", fun = sum)

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

top10otus = names(sort(taxa_sums(AyBCode.percent), TRUE)[1:23])
```

```
taxtab10 = cbind(tax_table(AyBCode.percent), Family = NA)
taxtab10[top10otus, "Family"] <- as(tax_table(AyBCode.percent)[top10otus, "Family"],"character")
tax_table(AyBCode.percent) <- tax_table(taxtab10)

top10plot = prune_taxa(top10otus, AyBCode.percent)
top10plot
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 23 taxa and 3 samples ]
## sample_data() Sample Data:       [ 3 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 23 taxa by 8 taxonomic ranks ]
```

```
title = "The 10 most abundant taxa (16s - Enriched)"

plot_bar(top10plot, fill = "Family") + coord_flip() +
  ylab("Taxa Matched with Silva138 (%)") + ylim(0, 100) +
  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")
```


Rate
Plunge
Control
Sample
0
25
50
75
100
Taxa Matched with Silva138 (%)
Family
Bacillaceae
Clostridiaceae
Enterobacteriaceae
Lachnospiraceae
Moraxellaceae
Nocardiaceae
Planococcaceae
Pseudomonadaceae
Weeksellaceae
Xanthomonadaceae