The UK Crop Microbiome Cryobank (ITS - Wheat)

2023-10-03

ITS of Wheat

The relationship between fungi and wheat encompasses significant importance in agricultural ecosystems, transcending beyond the realm of infections. The coexistence of these two entities profoundly impacts agricultural productivity, with the specter of wheat root rot disease, wrought by soil-borne fungal pathogens, casting a formidable shadow over global crop yields, resulting in staggering annual economic losses. In this case study, we investigated the nexus between rhizosphere soil fungal diversity and wheat roots in "normal" condition to find out the abundance of soil-borne fungal pathogens.

Qiime2 to Phyloseq

Converting QIIME2 outcomes to the phyloseq object structure in R enables comprehensive analysis and visualisation of microbial community profiles. The phyloseq package provides functions for organising and manipulating data within the phyloseq object, allowing for diverse analyses such as diversity assessments, differential abundance testing, and taxonomic profile visualisation. By leveraging R's capabilities, researchers can perform advanced statistical analysis, integrate with other omics data, and gain deeper insights into microbiome datasets.

```r
# Download qiime2R from Github
# if (!requireNamespace("devtools", quietly = TRUE))
#   install.packages("devtools")
# devtools::install_github("jbisanz/qiime2R")
library("qiime2R")

# Download phyloseq from CRAN
# if (!require("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
# BiocManager::install("phyloseq")
library("phyloseq")
# install.packages("tidyverse")
library("tidyverse")
# install.packages("ggpubr")
library("ggpubr")
```

```r
# Convert qiime2 to phyloseq format
physeq.a <- qza_to_phyloseq(
  features = "ITS/table-its-fungi-with-phyla-no-mitochondria-no-chloroplast.qza", # table.qza
  tree = "ITS/phylogeny-align-to-tree-mafft-fasttree/rooted_tree.qza",
  taxonomy = "ITS/taxonomy-fungi.qza",
  metadata = "ITS/meta-table.txt"
)
physeq.a ## confirm the object
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 3904 taxa and 37 samples ]
## sample_data() Sample Data:       [ 37 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 3904 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 3904 tips and 3878 internal nodes ]
```

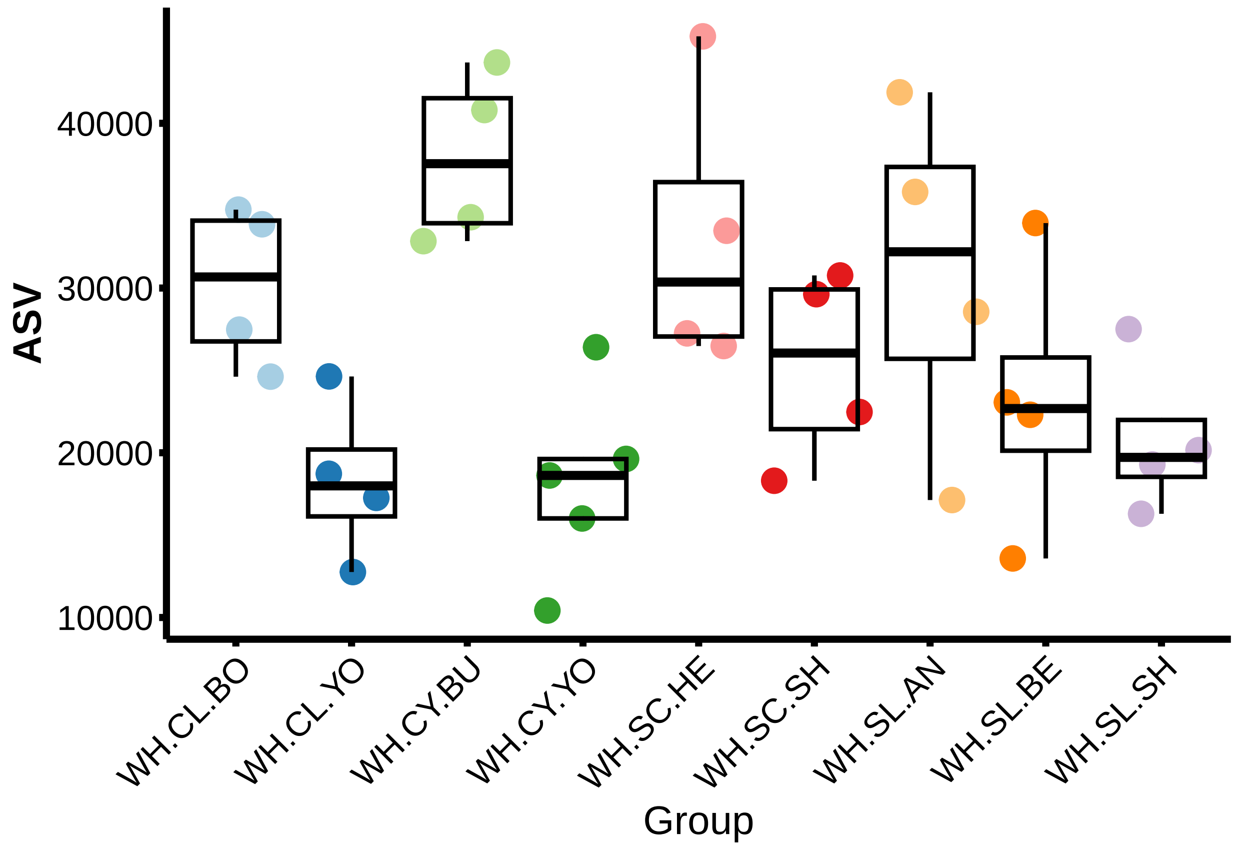
```
## (3A) Merge the replicate samples for each Group
physeq.a.group = merge_samples(physeq.a, "Group") # Sum between replicate samples

# (3B) repair factors in the sample metadata
sample_data(physeq.a.group)$Group <- levels(sample_data(physeq.a)$Group)[get_variable(physeq.a.group, "G
sample_data(physeq.a.group)$Soil <- levels(sample_data(physeq.a)$Soil)[get_variable(physeq.a.group, "So
```

Assessing ASV Distribution (merged and filtered)

We evaluated the number of Amplicon Sequence Variants (ASVs) after merging and filtering, as well as visualised the raw read counts per sample within each group. This provided insights into the distribution of ASVs and the sequencing depth of each sample, allowing for a comprehensive assessment of data quality and quantity.

```
# install.packages("ggplot2")
library("ggplot2")
# install.packages("RColorBrewer")
library("RColorBrewer")
```

```
ASV <- sample_sums(physeq.a)
ASV <- as.data.frame(ASV)
ASV$Group <- physeq.a@sam_data$Group

#### pdf
# pdf(file = "Suppl_ASV_ITS.pdf", width = 8,height = 5)
ggplot(ASV, aes(x = Group, y = ASV,  colour = Group))+
  scale_color_brewer(palette = "Paired") +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "none")
```



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

rm(ASV)
```

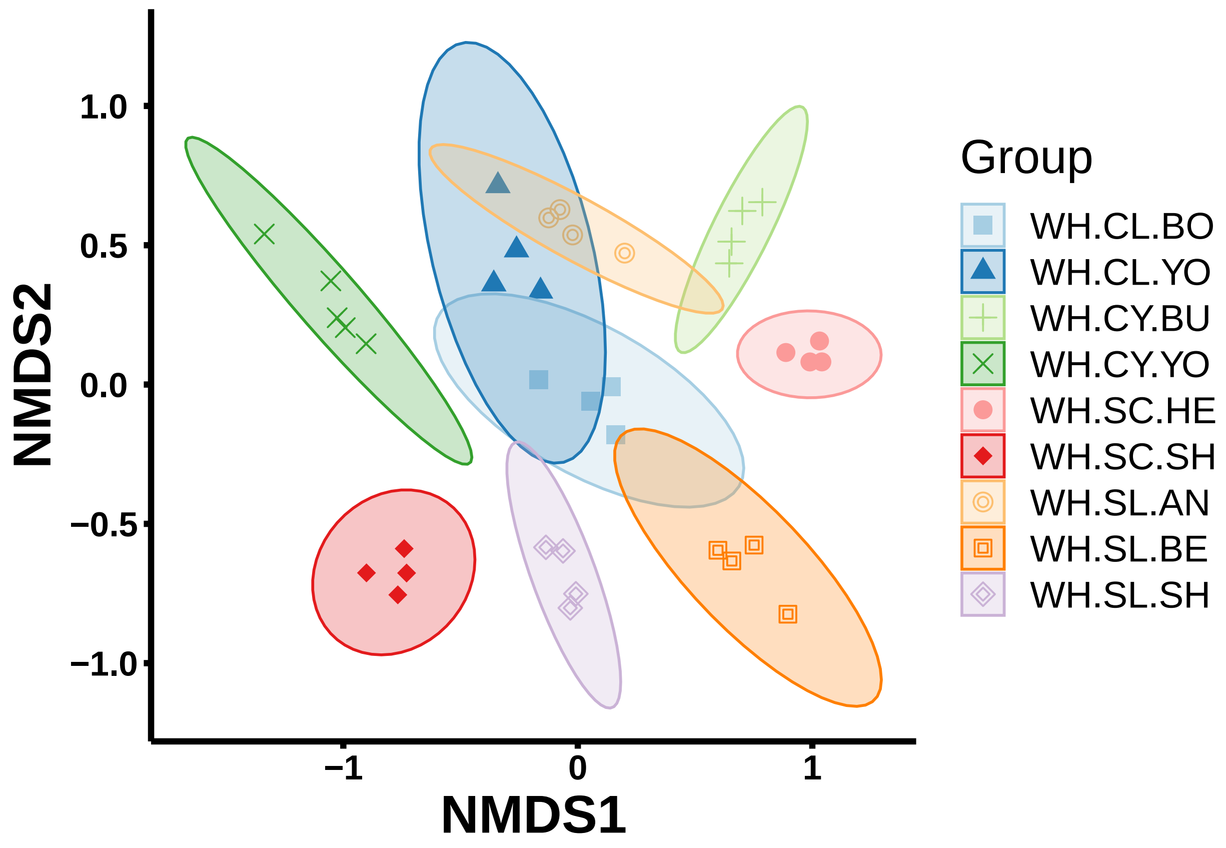
Beta diversity

Beta diversity is a metric used in microbial community studies to assess species composition dissimilarity between samples, providing insights into microbial community dynamics and their ecological significance. By quantifying variation in community structure and employing visualisation techniques, beta diversity analysis allows comparisons across habitats or treatments, revealing key drivers of community variation and functional implications. It aids our understanding of microbial assemblages and their responses to environmental changes in ecology, environmental science, and human health research.

```
# method options: NMDS / PCoA
NMDS <- ordinate(physeq = physeq.a, method = "NMDS", distance = "bray")
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.196196
## Run 1 stress 0.230723
## Run 2 stress 0.1929835
## ... New best solution
## ... Procrustes: rmse 0.1085367  max resid 0.2861642
## Run 3 stress 0.196196
## Run 4 stress 0.2133068
## Run 5 stress 0.2096686
## Run 6 stress 0.2290658
## Run 7 stress 0.2742863
## Run 8 stress 0.1929835
## ... New best solution
## ... Procrustes: rmse 5.317177e-06  max resid 1.62905e-05
## ... Similar to previous best
## Run 9 stress 0.2096688
## Run 10 stress 0.196196
## Run 11 stress 0.2217112
## Run 12 stress 0.1929835
## ... New best solution
## ... Procrustes: rmse 2.179965e-06  max resid 5.596278e-06
## ... Similar to previous best
## Run 13 stress 0.1929835
## ... New best solution
## ... Procrustes: rmse 1.912817e-06  max resid 5.861761e-06
## ... Similar to previous best
## Run 14 stress 0.2096686
## Run 15 stress 0.1929835
## ... Procrustes: rmse 1.467555e-06  max resid 3.300139e-06
## ... Similar to previous best
## Run 16 stress 0.2315529
## Run 17 stress 0.1929835
## ... Procrustes: rmse 5.387347e-06  max resid 1.731008e-05
## ... Similar to previous best
## Run 18 stress 0.1929835
## ... Procrustes: rmse 3.917501e-06  max resid 1.255278e-05
## ... Similar to previous best
## Run 19 stress 0.1929835
## ... Procrustes: rmse 1.02338e-06  max resid 2.853153e-06
## ... Similar to previous best
## Run 20 stress 0.1929835
## ... Procrustes: rmse 1.845219e-06  max resid 4.285748e-06
## ... Similar to previous best
## *** Best solution repeated 6 times
```

```r
# pdf
# pdf(file = "Fig7A_bata_ITS.pdf", width = 8,height = 5)
# Plot ordination
plot_ordination(physeq = physeq.a,
                      ordination = NMDS,
                      color = "Group",
                      shape = "Group"
) +
  theme_classic() +
    scale_color_brewer(palette = "Paired") +
  scale_fill_brewer(palette = "Paired") +
  geom_point(aes(color = Group), alpha = 1, size = 3) +
  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)) +
  scale_shape_manual(values = c(15, 17, 3, 4, 16, 18, 21, 22, 23)) # Set custom shapes
```

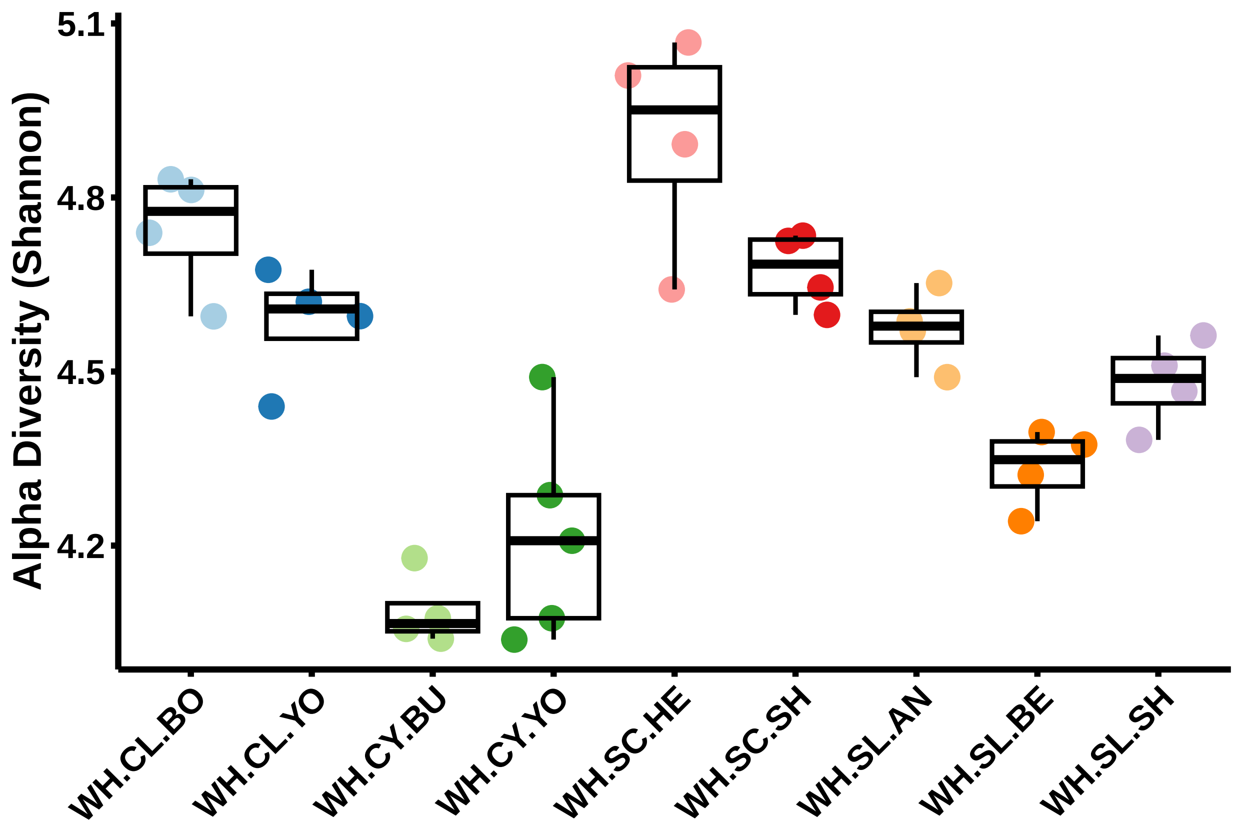


```
# Close the PDF device and save the plot to a file
# dev.off()

# Clean up by removing objects that are no longer needed
rm(NMDS)
```

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")]
alpha.object <- cbind(
  x = sample_data(physeq.a),
  y = estimate_richness(physeq.a, measures = 'Shannon')
)

#### pdf
# pdf(file = "Fig7B_Alpha_ITS.pdf", width = 8,height = 5)
# Create a plot for alpha diversity
ggplot(data = alpha.object, aes(x = x.Group, y = Shannon, color = x.Group)) +
  scale_color_brewer(palette = "Paired") +
  theme_classic() +
  labs(
    x = element_blank(),                    # No x-axis label
    y = "Alpha Diversity (Shannon)"         # y-axis label
  ) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  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 = 45, hjust = 1, 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"                # Hide legend
  )
```



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


# Clean up by removing the alpha.object
rm(alpha.object)
```

Pairwise comparison using PERMANOVA

Pairwise PERMANOVA is a statistical method used to compare multiple groups or treatments in ecological and microbial community studies. It assesses dissimilarity between samples and provides a p-value to determine the significance of observed differences. This approach is valuable for targeted group comparisons, allowing researchers to investigate the effects of specific factors on microbial communities and uncover significant variations in community composition. By considering within- and between-group variation, pairwise PERMANOVA provides robust statistical analysis and insights into microbial community dynamics and functioning.

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

```r
# p.adjust.methods
# c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")

metdat = as.data.frame(as.matrix(physeq.a@sam_data))
```

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

# Keep only the necessary columns
df <- perma[, c("pairs", "p.adjusted")]

# Split the "pairs" column into two separate columns
df_split <- strsplit(df$pairs, " vs ")
df$A <- sapply(df_split, "[[", 1)
df$B <- sapply(df_split, "[[", 2)

# Create a matrix
cor_matrix <- matrix(NA, nrow = length(unique(df$A)), ncol = length(unique(df$B)))
rownames(cor_matrix) <- unique(df$A)
colnames(cor_matrix) <- unique(df$B)

# Fill in the correlation matrix with p.adjusted sig values
for (i in 1:nrow(cor_matrix)) {
  for (j in 1:ncol(cor_matrix)) {
    a <- rownames(cor_matrix)[i]
    b <- colnames(cor_matrix)[j]
    value <- df[df$A == a & df$B == b, "p.adjusted"]
    if (length(value) > 0) {
      cor_matrix[i, j] <- value
    }
  }
}

# Clean up by removing unnecessary objects generated from the loop
# rm(a, b, i, j, value)

print(cor_matrix)
```

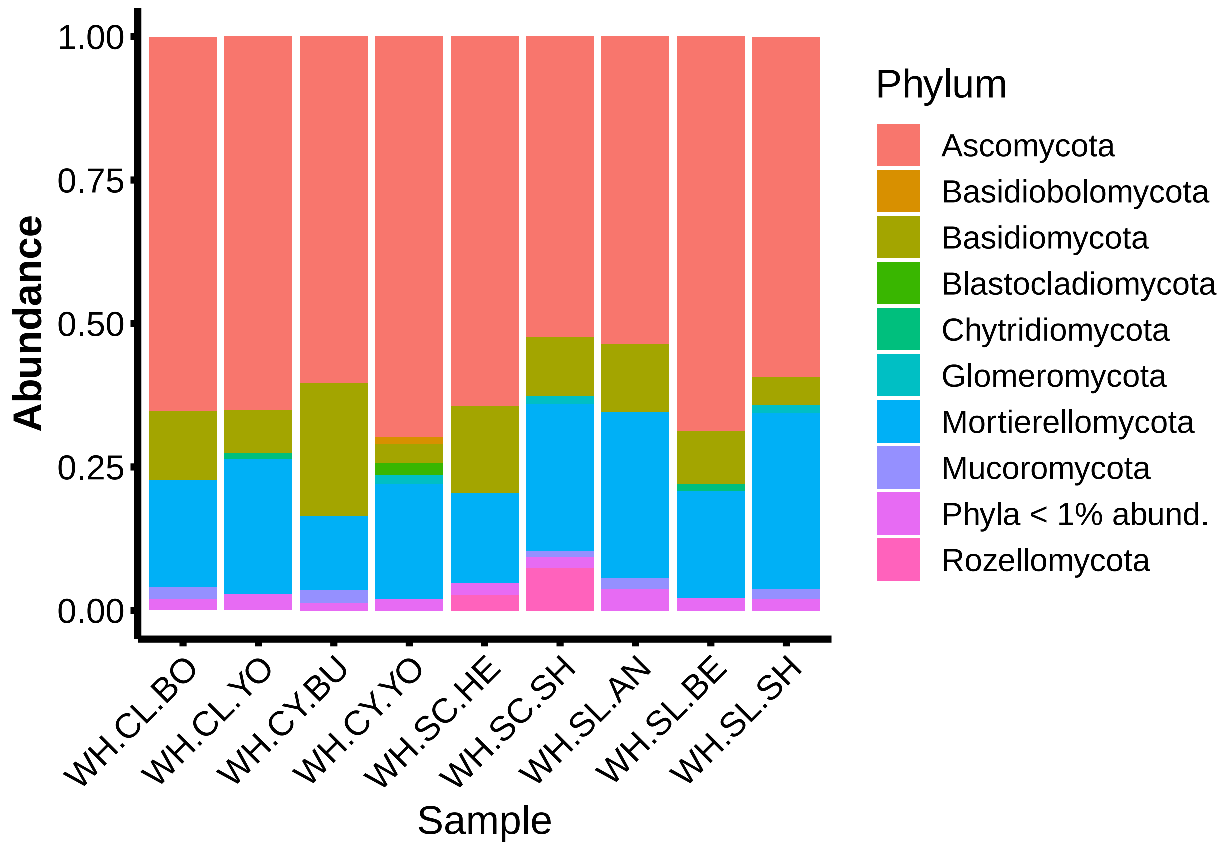
```
##             WH.CL.YO   WH.CY.BU   WH.CY.YO   WH.SC.HE   WH.SC.SH   WH.SL.AN
## WH.CL.BO 0.02857143 0.02857143 0.02857143 0.02857143 0.02857143 0.02857143
## WH.CL.YO         NA 0.02857143 0.02857143 0.02857143 0.02857143 0.02857143
## WH.CY.BU         NA         NA 0.02857143 0.02857143 0.02857143 0.02857143
## WH.CY.YO         NA         NA         NA 0.02857143 0.02857143 0.02857143
## WH.SC.HE         NA         NA         NA         NA 0.02857143 0.02857143
## WH.SC.SH         NA         NA         NA         NA         NA 0.02857143
## WH.SL.AN         NA         NA         NA         NA         NA         NA
## WH.SL.BE         NA         NA         NA         NA         NA         NA
##             WH.SL.BE   WH.SL.SH
## WH.CL.BO 0.02857143 0.02857143
## WH.CL.YO 0.02857143 0.02857143
## WH.CY.BU 0.02857143 0.02857143
## WH.CY.YO 0.02857143 0.02857143
## WH.SC.HE 0.02857143 0.02857143
## WH.SC.SH 0.02857143 0.02857143
## WH.SL.AN 0.02857143 0.02857143
## WH.SL.BE         NA 0.02857143
```

Bar plot - relative abundance

Bar plots illustrating relative abundance in microbiome enable to visually represent the proportions of different microbial taxa within a given sample. Each bar corresponds to a specific taxonomic group, with its height indicating the relative abundance of that group in the microbiome.

```r
# install.packages("plyr")
library("plyr")
```

```r
# https://github.com/joey711/phyloseq/issues/901

# Filter taxa with mean abundance > 0.1
physeq2 = filter_taxa(physeq.a.group, function(x) mean(x) > 0.1, TRUE)

# Transform sample counts to relative abundances
physeq3 = transform_sample_counts(physeq2, function(x) x / sum(x))

# Aggregate taxa at the Phylum level
glom <- tax_glom(physeq3, taxrank = 'Phylum')

# Convert the aggregated phyloseq object to a dataframe
data <- psmelt(glom)
# Convert Phylum column to character
data$Phylum <- as.character(data$Phylum)

# Rename phyla with abundance < 1% to "< 1% abund."
data$Phylum[data$Abundance < 0.01] <- "< 1% abund."

# Calculate medians of abundance for each Phylum
medians <- ddply(data, ~Phylum, function(x) c(median = median(x$Abundance)))

# Extract phyla with median abundance <= 0.01
remainder <- medians[medians$median <= 0.01, ]$Phylum

# Rename selected phyla to "Phyla < 1% abund."
data[data$Phylum %in% remainder, ]$Phylum <- "Phyla < 1% abund."

# Rename phyla with abundance < 1% to "Phyla < 1% abund."
data$Phylum[data$Abundance < 0.01] <- "Phyla < 1% abund."

# Plot the data with condensed phyla categories
p <- ggplot(data = data, aes(x = Sample, y = Abundance, fill = Phylum))
p + geom_bar(aes(), stat = "identity", position = "stack") +
    scale_color_brewer(palette = "Paired") +
  theme_classic() +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=15,face="bold"), legend.position = "right") # +
```



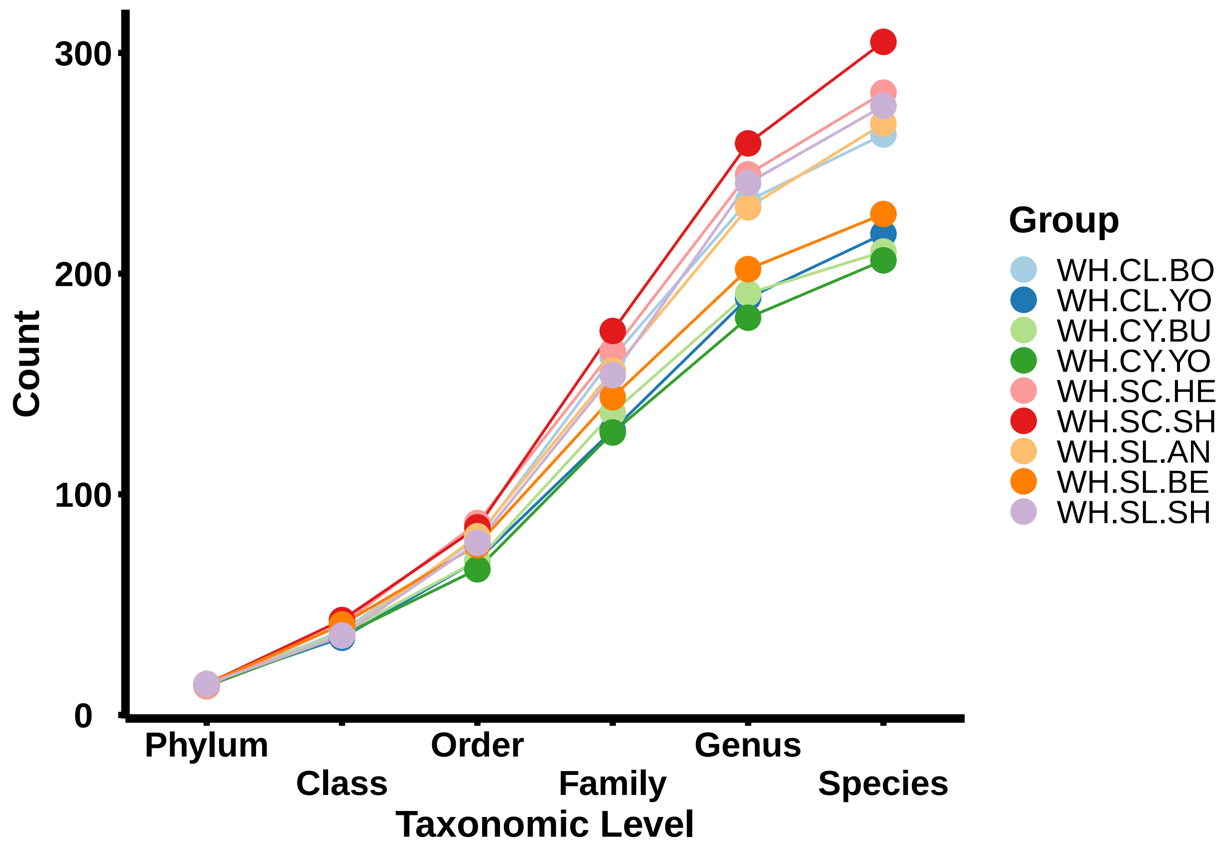
Number of taxa in groups Examining the cumulative count of taxa within a particular taxonomic group for each distinct subgroup involves the summation of the total number of taxa within that group while eliminating any overlapping taxa based on their names within individual subgroups. This process aids in comprehending the dynamics of microbiome populations.

```
library("dplyr")
library("reshape2")
```

```
physeq <- merge_samples(physeq.a, "Group", fun = sum)
gentab_levels <- list()
observationThreshold <- 1

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

# Something wrong with Kingdom Level, removed from the analysis
# loop through all the taxonomic levels
  for (level in genus_levels) {

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

    # calculate the abundance of each genus within each sample
    gentab <- apply(otu_table(physeq), 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 %>% reduce(full_join, by = "name")

# Set row names and column names
row.names(B2) <- B2$name
B2$name <- NULL
colnames(B2)[1:6] <- genus_levels

print(B2)
```

```
##           Phylum Class Order Family Genus Species
## WH.CL.BO      14    38    80    162   233     263
## WH.CL.YO      14    35    70    129   189     218
## WH.CY.BU      14    37    70    137   191     210
## WH.CY.YO      13    36    66    128   180     206
## WH.SC.HE      13    41    87    165   245     282
## WH.SC.SH      14    43    85    174   259     305
## WH.SL.AN      14    36    81    156   230     268
## WH.SL.BE      14    41    77    144   202     227
## WH.SL.SH      14    36    78    154   241     276
```

```
# write.csv(B2, file = "ITS/level_counts_by_group.csv", row.names = TRUE)
B2$name = row.names(B2)
data_long <- melt(B2)
data_long <- melt(B2, id.vars = "name", variable.name = "Level", value.name = "Taxa")


# Plot the data as a line graph using ggplot
# Open a new PDF graphics device
# pdf(file = "Fig7C_line_graph_ITS.pdf", width=8,height=5)
ggplot(data_long, aes(x = Level, y = Taxa, color = name, group = name)) +
  geom_line() +
  geom_point(size = 4) +
  labs(x = "Taxonomic Level", y = "Count", color = "Group") +
  theme_classic() +   scale_color_brewer(palette = "Paired") +
  theme(
    text = element_text(size = 19, colour = "black"),
    axis.ticks = element_line(colour = "black", size = 1.1),
    axis.line = element_line(colour = 'black', size = 1.5),
    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(0,400,by=100))
```



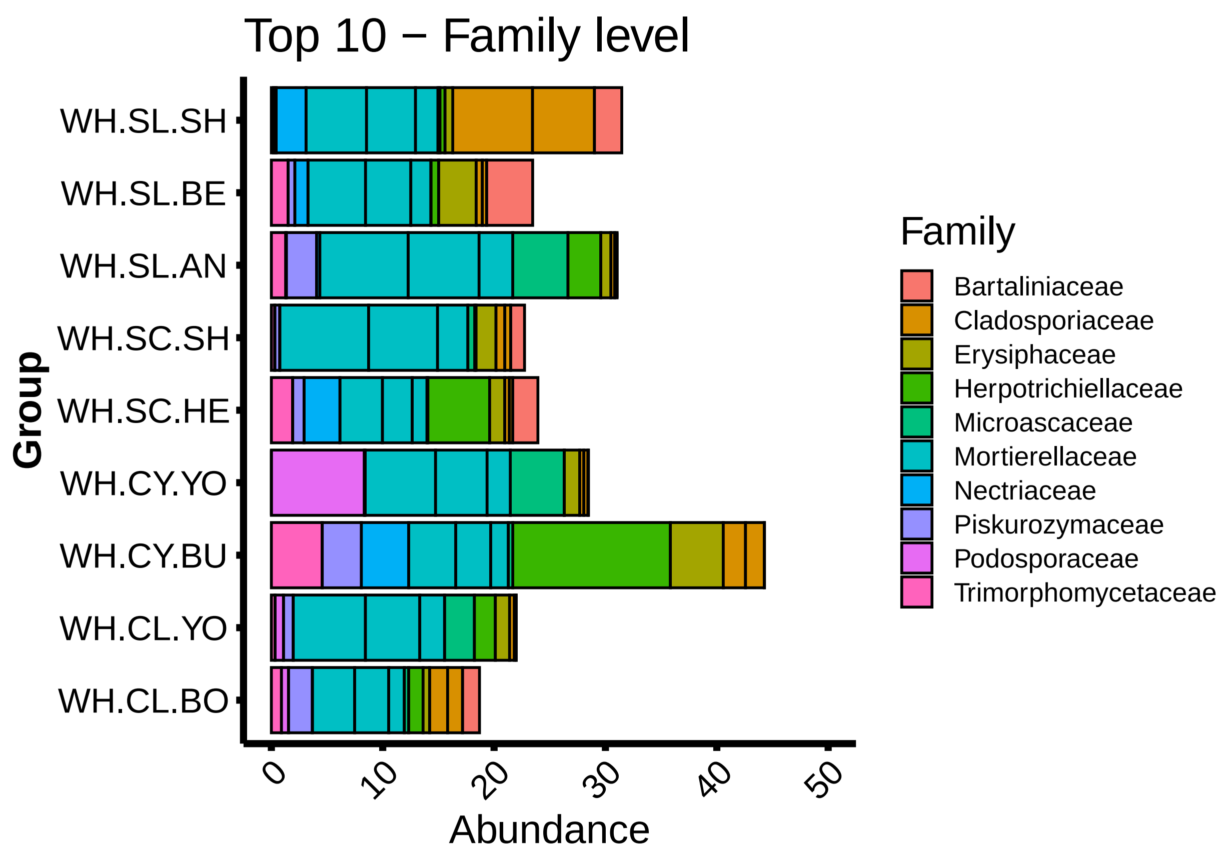
```
# Close the PDF device and save the plot to a file
# dev.off()
```

Plotting the top 10 taxa at family level

Generating a visual representation through the plotting of the top 10 taxa at the family level offers a concise overview of the predominant microbial groups within a dataset. This approach is instrumental in identifying and emphasizing significant contributors to the structure and function of microbial communities across diverse groups or conditions.

```
## Transform normalised ASVs to proportions
proportions = transform_sample_counts(physeq.a.group, function(x) 100 * x/sum(x))
```

```r
##
Top10ASVs = names(sort(taxa_sums(proportions), TRUE)[1:13])
Taxtab10 = cbind(tax_table(proportions), Family30 = NA)
Taxtab10[Top10ASVs, "Family30"] <- as(tax_table(proportions)[Top10ASVs, "Family"], "character")
tax_table(proportions) <- tax_table(Taxtab10)

Rgsm10 = prune_taxa(Top10ASVs, proportions)

# plotting
title = "Top 10 - Family level"
plot_bar(Rgsm10, "Group", fill = "Family", title = title) +
  coord_flip() +
  ylim(0, 50) +
  scale_color_brewer(palette = "Paired") +
  theme_classic() +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=15,face="bold"),
        legend.position = "right",
        legend.text = element_text(size = 9.5),
        legend.key.height= unit(0.45, 'cm'),
        legend.key.width= unit(0.45, 'cm')
        )
```



```r
# Clean up by removing unnecessary objects
rm(proportions, Top10ASVs, Taxtab10, Rgsm10, title)
```

Upset plot using UpsetR

When it comes to representing sets visually, the go-to option is usually a Venn diagram. These diagrams work well when dealing with up to five sets, providing a clear visualisation. However, as the dataset expands, such as when dealing with five sets, deriving the desired insights from the diagram becomes more complex. As a result, considering an UpSet graph for data visualisation becomes an appealing choice. UpSet graphs offer a more streamlined way to display intersections and complements, particularly when dealing with larger datasets or multiple sets. This option ensures a more intuitive and informative representation of the data.

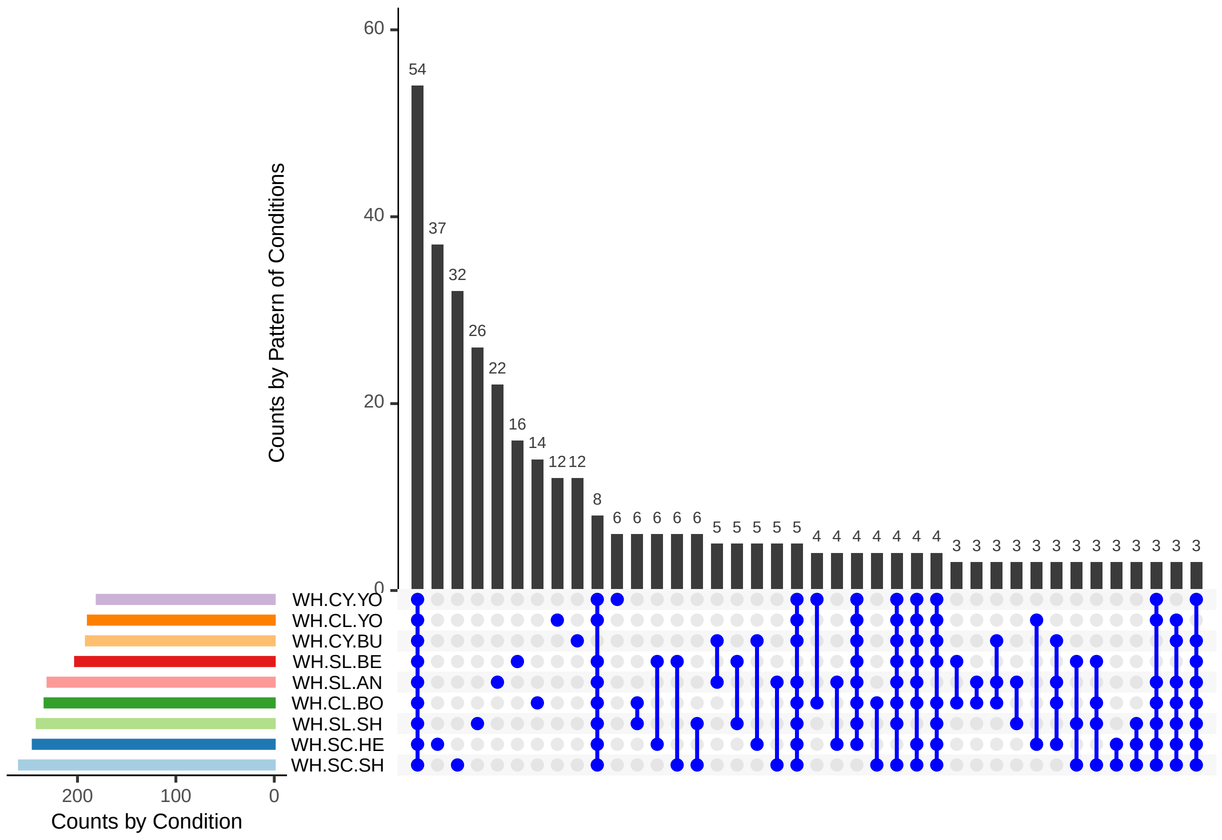
```r
library("UpSetR")
library("reshape2")
library("plyr")
library("dplyr")
# BiocManager::install("microbiome")
library("microbiome")
```

```r
# Aggregate taxa at the genus level
B <- aggregate_taxa(physeq.a.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"
```

```
# 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(desc(Abundance))) %>%
   filter(Abundance > 0) %>%
  ungroup() %>%
  select(Sample, Abundance, ASV)

# Remove the Abundance column
D$Abundance <- NULL

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

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

# Define a binary function
binary_fun <- function(x) {
  x[is.na(x)] <- 0
  ifelse(x > 0, 1, 0)
}

col = brewer.pal(n = 9, name = "Paired")

# Apply the binary function to columns 2 to 10
```

```r
temp_df <- apply(E[2:10], 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",
                    point.size=2
                    )

# Open a new PDF graphics device
# pdf(file = "Fig7D_UpSet_ITS.pdf", width=8,height=5)

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



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

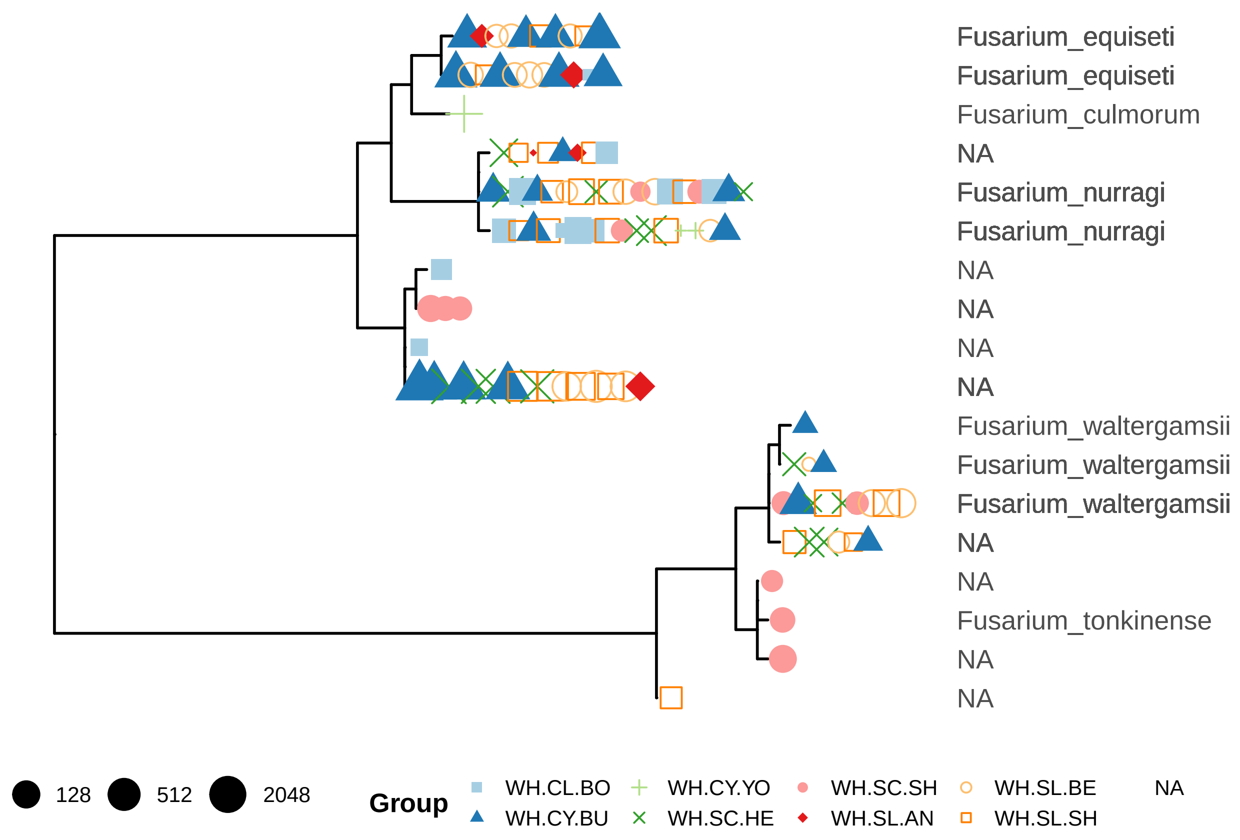
Phylogenetic tree

Phylogenetic trees are branching diagrams that show the evolutionary relationships between different species or other entities. They are constructed by comparing the organisms' DNA, proteins, or other features. The more similar the features, the more closely related the organisms are thought to be. It can be used to answer a variety of questions about evolution, such as how different species are related to each other, when different species diverged from each other, and what are the common ancestors of different species.

Here we focus on Fusarium. Fusarium is a large genus of filamentous fungi, part of a group often referred to as hyphomycetes, widely distributed in soil and associated with plants1. Most species are harmless saprobes, and are relatively abundant members of the soil microbial community1. Some species produce mycotoxins in cereal crops that can affect human and animal health if they enter the food chain1. Despite most species apparently being harmless, some Fusarium species and subspecific groups are among the most important fungal pathogens of plants and animals. (draft)

```r
# Load required libraries
library("phyloseq")   # For handling phylogenetic sequencing data
# BiocManager::install("ggtree")
library("ggtree")     # For tree visualisation
library("scales")     # For scaling transformations
```

```r
# Load the GlobalPatterns dataset and prune taxa
GP <- prune_taxa(taxa_sums(physeq.a) > 0, physeq.a)  # Remove taxa with zero sums
GP.chl <- subset_taxa(GP, Genus == "Fusarium")   # Subset data based on Phylum value


# Create a ggtree plot
p <- ggtree(GP.chl, ladderize = TRUE) +
  # geom_text2(aes(subset = !isTip, label = label), hjust = -1.5, size = 3.5) +
  geom_tiplab(aes(label = Species), as_ylab=TRUE) +
  geom_point(aes(x = x + hjust, color = Group,
                 shape = Group, size = Abundance), na.rm = TRUE) +
  scale_size_continuous(trans = log_trans(2)) +
  scale_shape_manual(values = c(15, 17, 3, 4, 16, 18, 21, 22, 23)) + # Set custom
  scale_color_brewer(palette = "Paired") +
  theme(text = element_text(size=13, colour = "black"),
        axis.ticks = element_line(colour = "black", size = 1.25),
        axis.line = element_line(colour = 'black', size = 1.25) ,
        axis.title.y = element_text(color="black", size=2.5,face="bold"),
        legend.text = element_text(size = 8),
        legend.key.size=unit(0.4,"cm"),
        legend.title = element_text(size = 10, face = "bold"),
        legend.position = "bottom"
  )

# Open a new PDF graphics device
# pdf(file = "Fig8_Tree_ITS.pdf", width=8,height=5)
# Print the ggtree plot
print(p)
```



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

Fusarium equiseti is a fungal species and plant pathogen on a varied range of crops. It is considered to be a weak pathogen on cereals and is occasionally found to be associated with 'Fusarium head blight' infected kernels. It is commonly found in tropical and sub-tropical areas. (draft)

Fusarium nurragi has been isolated from heath species growing on the coasts in the Australian states of Victoria and Western Australia. Although F. nurragi has not been assigned to a Fusarium species complex, it is closely related to the F. heterosporum and F. tricinctum species complexes. (draft)

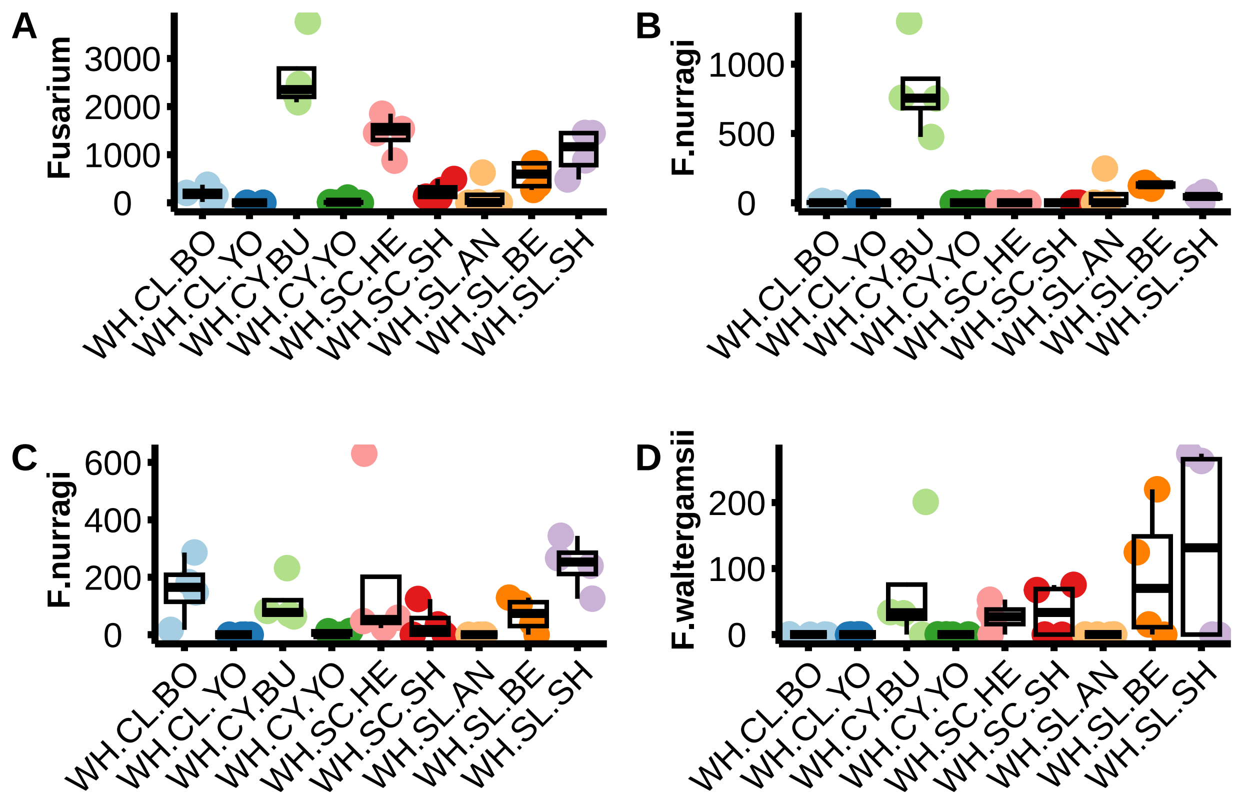
Fusarium waltergamsii . . . .

Fusarium culmorum and Fusarium tonkinense were identified in only one of the samples. This could potentially be attributed to errors in the Polymerase Chain Reaction (PCR) process or limitations inherent to ITS sequencing.

```
# Plotting for the abundance of one specific fungi
# Subset the taxa to Genus from physeq.wheat
physeq.a.genus <- subset_taxa(physeq.a, Genus == "Fusarium")
physeq.a.equiseti <- subset_taxa(physeq.a, Species == "Fusarium_equiseti")
physeq.a.nurragi <- subset_taxa(physeq.a, Species == "Fusarium_nurragi")
physeq.a.waltergamsii <- subset_taxa(physeq.a, Species == "Fusarium_waltergamsii")

# Calculate the total abundance of Fusarium for each sample
meta = physeq.a@sam_data
otudf = as.data.frame(t(as.data.frame(physeq.a.genus@otu_table)))
```

```r
meta$Fusarium = rowSums(otudf)
otudf = as.data.frame(t(as.data.frame(physeq.a.equiseti@otu_table)))
meta$F.equiseti = rowSums(otudf)
otudf = as.data.frame(t(as.data.frame(physeq.a.nurragi@otu_table)))
meta$F.nurragi = rowSums(otudf)
otudf = as.data.frame(t(as.data.frame(physeq.a.waltergamsii@otu_table)))
meta$F.waltergamsii = rowSums(otudf)

# Plot a graph of the abundance of Fusarium for each sample grouped by Group:
p1 <- ggplot(subset(meta, Group %in% c("WH.CL.BO","WH.CL.YO",
                                       "WH.CY.BU","WH.CY.YO",
                                       "WH.SC.HE","WH.SC.SH",
                                       "WH.SL.AN","WH.SL.BE",
                                       "WH.SL.SH")),
             aes(x = Group, y = Fusarium,  colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +   scale_color_brewer(palette = "Paired") +
  labs(x = "", y = "\n Fusarium") +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=12,face="bold"), legend.position = "none")

p2 <- ggplot(subset(meta, Group %in% c("WH.CL.BO","WH.CL.YO",
                                       "WH.CY.BU","WH.CY.YO",
                                       "WH.SC.HE","WH.SC.SH",
                                       "WH.SL.AN","WH.SL.BE",
                                       "WH.SL.SH")),
             aes(x = Group, y = F.equiseti,  colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +   scale_color_brewer(palette = "Paired") +
  labs(x = "", y = "\n F.nurragi") +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=12,face="bold"),
        legend.position = "none")

p3 <- ggplot(subset(meta, Group %in% c("WH.CL.BO","WH.CL.YO",
                                       "WH.CY.BU","WH.CY.YO",
                                       "WH.SC.HE","WH.SC.SH",
                                       "WH.SL.AN","WH.SL.BE",
                                       "WH.SL.SH")),
             aes(x = Group, y = F.nurragi,  colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +   scale_color_brewer(palette = "Paired") +
  labs(x = "", y = "\n F.nurragi") +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=12,face="bold"),
        legend.position = "none")

p4 <- ggplot(subset(meta, Group %in% c("WH.CL.BO","WH.CL.YO",
                                       "WH.CY.BU","WH.CY.YO",
                                       "WH.SC.HE","WH.SC.SH",
                                       "WH.SL.AN","WH.SL.BE",
                                       "WH.SL.SH")),
             aes(x = Group, y = F.waltergamsii,  colour = interaction(Group))) +
  geom_point(alpha = 1, position = "jitter", size = 4) +
  geom_boxplot(alpha = 0, colour = "black", size = 0.8)+
  theme_classic() +   scale_color_brewer(palette = "Paired") +
  labs(x = "", y = "\n F.waltergamsii") +
  theme(text = element_text(size=15, 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(angle=45, hjust=1, colour = "black", size = 13),
        axis.text.y = element_text(angle=0, hjust=0.5, colour = "black",size = 13),
        axis.title.y = element_text(color="black", size=12,face="bold"),
        legend.position = "none")

# Combine and Arrange the plots
fig <- ggarrange(p1, p2, p3, p4, labels = c("A", "B", "C", "D"), size = 5,
                 ncol = 2, nrow = 2)

# Add labels
fig <- annotate_figure(fig)

# Print the figure
print(fig)
```

```
## $`1`
```


A
Fusarium
0
1000
2000
3000
B
F.nurragi
0
500
1000
C
F.nurragi
0
200
400
600
D
F.waltergamsii
0
100
200
WH.CL.BO
WH.CL.YO
WH.CY.BU
WH.CY.YO
WH.SC.HE
WH.SC.SH
WH.SL.AN
WH.SL.BE
WH.SL.SH

```
##
## $‘2‘
```

A

```
##
## attr(,"class")
## [1] "list"      "ggarrange"
```

```r
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22621)
##
## 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] stats     graphics  grDevices utils     datasets  methods   base
```

```
##
## other attached packages:
##  [1] scales_1.2.1          ggtree_3.8.2          microbiome_1.22.0
##  [4] UpSetR_1.4.0          reshape2_1.4.4        plyr_1.8.9
##  [7] pairwiseAdonis_0.4.1  cluster_2.1.4         vegan_2.6-4
## [10] lattice_0.21-8        permute_0.9-7         RColorBrewer_1.1-3
## [13] ggpubr_0.6.0          lubridate_1.9.3       forcats_1.0.0
## [16] stringr_1.5.0         dplyr_1.1.3           purrr_1.0.2
## [19] readr_2.1.4           tidyr_1.3.0           tibble_3.2.1
## [22] ggplot2_3.4.3         tidyverse_2.0.0       phyloseq_1.44.0
## [25] qiime2R_0.99.6
##
## loaded via a namespace (and not attached):
##   [1] rstudioapi_0.15.0       jsonlite_1.8.7          magrittr_2.0.3
##   [4] NADA_1.6-1.1            farver_2.1.1            rmarkdown_2.25
##   [7] fs_1.6.3                zlibbioc_1.46.0         vctrs_0.6.3
##  [10] multtest_2.56.0         memoise_2.0.1           RCurl_1.98-1.12
##  [13] base64enc_0.1-3         rstatix_0.7.2           htmltools_0.5.6
##  [16] truncnorm_1.0-9         broom_1.0.5             Rhdf5lib_1.22.1
##  [19] Formula_1.2-5           rhdf5_2.44.0            gridGraphics_0.5-1
##  [22] htmlwidgets_1.6.2       cachem_1.0.8            igraph_1.5.1
##  [25] lifecycle_1.0.3         iterators_1.0.14        pkgconfig_2.0.3
##  [28] Matrix_1.6-1.1          R6_2.5.1                fastmap_1.1.1
##  [31] GenomeInfoDbData_1.2.10 digest_0.6.33           aplot_0.2.2
##  [34] colorspace_2.1-0        patchwork_1.1.3         S4Vectors_0.38.2
##  [37] Hmisc_5.1-1             labeling_0.4.3          fansi_1.0.4
##  [40] timechange_0.2.0        abind_1.4-5             mgcv_1.8-42
##  [43] compiler_4.3.1          withr_2.5.1             htmlTable_2.4.1
##  [46] backports_1.4.1         carData_3.0-5           ggsignif_0.6.4
##  [49] MASS_7.3-60             biomformat_1.28.0       tools_4.3.1
##  [52] foreign_0.8-84          ape_5.7-1               nnet_7.3-19
##  [55] glue_1.6.2              nlme_3.1-162            rhdf5filters_1.12.1
##  [58] grid_4.3.1              checkmate_2.2.0         Rtsne_0.16
##  [61] ade4_1.7-22             generics_0.1.3          gtable_0.3.4
##  [64] tzdb_0.4.0              data.table_1.14.8       hms_1.1.3
##  [67] car_3.1-2               utf8_1.2.3              XVector_0.40.0
##  [70] BiocGenerics_0.46.0     foreach_1.5.2           pillar_1.9.0
##  [73] yulab.utils_0.1.0       splines_4.3.1           treeio_1.24.3
##  [76] survival_3.5-5          tidyselect_1.2.0        Biostrings_2.68.1
##  [79] knitr_1.44              gridExtra_2.3           IRanges_2.34.1
##  [82] zCompositions_1.4.1     stats4_4.3.1            xfun_0.40
##  [85] Biobase_2.60.0          DT_0.30                 stringi_1.7.12
##  [88] lazyeval_0.2.2          ggfun_0.1.3             yaml_2.3.7
##  [91] evaluate_0.22           codetools_0.2-19        ggplotify_0.1.2
##  [94] cli_3.6.1               rpart_4.1.19            munsell_0.5.0
##  [97] Rcpp_1.0.11             GenomeInfoDb_1.36.4     parallel_4.3.1
## [100] bitops_1.0-7            tidytree_0.4.5          crayon_1.5.2
## [103] rlang_1.1.1             cowplot_1.1.1
```