Correlationnetwork with 3 AMFs

Sri Sai Nandini Ravi

2025-05-24

```
library(tidyverse)  # for data manipulation and plotting
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readxl)     # for reading Excel files
library(reshape2)   # for reshaping data
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(ggplot2)    # for plotting
library(phyloseq)
library(dplyr)
library(vegan)
```

```
## Loading required package: permute
## Loading required package: lattice
```

```
library(ALDEx2)
```

```
## Loading required package: zCompositions
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
```

```
##
##     select
##
## Loading required package: NADA
## Loading required package: survival
##
## Attaching package: 'NADA'
##
## The following object is masked from 'package:stats':
##
##     cor
##
## Loading required package: truncnorm
## Loading required package: latticeExtra
##
## Attaching package: 'latticeExtra'
##
## The following object is masked from 'package:ggplot2':
##
##     layer
```

```
library(indicspecies)
library(Hmisc)     # For Spearman's correlation
```

```
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
##
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(igraph)    # For network analysis
```

```
##
## Attaching package: 'igraph'
##
## The following object is masked from 'package:vegan':
##
##     diversity
##
## The following object is masked from 'package:permute':
##
##     permute
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
```

```
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
library(ggraph)    # For network visualization
library(tidygraph) # For handling network objects
```

```
##
## Attaching package: 'tidygraph'
##
## The following object is masked from 'package:igraph':
##
##     groups
##
## The following object is masked from 'package:MASS':
##
##     select
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(dplyr)     # For data wrangling
```

```
ASV_16s <- read.csv ("Greenhouse_16S_ASV_table_sort.csv")
ASV_ITS <- read.csv("Greenhouse_ITS_ASV_table_sort.csv")
MD <- read_xlsx("GreenhouseMetadata.xlsx")
MD <- MD %>%
  mutate(Treatment = recode(Treatment,
                            "_+M+E" = "ME",
                            "_-M+E" = "E",
                            "_+M-E" = "M",
                            "_-M-E" = "control"))
```

```r
# Identify sample columns (assumes samples start with "FFAR")
sample_columns_16s <- grep("^FFAR", colnames(ASV_16s), value = TRUE)
sample_columns_ITS <- grep("^FFAR", colnames(ASV_ITS), value = TRUE)

# --- Filter out low-abundance ASVs from 16S ---
ASV_16s_filtered <- ASV_16s %>%
  mutate(TotalAbundance = rowSums(across(all_of(sample_columns_16s)))) %>%
  filter(TotalAbundance >= 10)  # Keep only ASVs with total abundance  10

# we are not filtering ASV cause we don't want to miss out the rare abundant taxa
ASV_ITS_filtered <- ASV_ITS

# Reshape 16S ASV table from wide to long format (filtered)
asv_16s_long <- ASV_16s_filtered %>%
  pivot_longer(cols = all_of(sample_columns_16s),
               names_to = "sample_name",
               values_to = "Abundance") %>%
  mutate(Abundance = as.numeric(Abundance))  # Ensure numeric values

# Reshape ITS ASV table from wide to long format (filtered)
asv_ITS_long <- ASV_ITS_filtered %>%
  pivot_longer(cols = all_of(sample_columns_ITS),
               names_to = "sample_name",
               values_to = "Abundance") %>%
  mutate(Abundance = as.numeric(Abundance))  # Ensure numeric values


# Merge ASV data with metadata
m_16s <- asv_16s_long %>%
  inner_join(MD, by = "sample_name")  # Ensure sample names match

m_ITS <- asv_ITS_long %>%
  inner_join(MD, by = "sample_name")  # Ensure sample names match
```

```r
#  Filter to only M-treatment samples
m_ITS <- m_ITS %>% filter(Treatment == "M")
m_16s <- m_16s %>% filter(Treatment == "M")
```

```r
# Step 1: Define 3 AMF species to keep
target_amf_species <- tribble(
  ~Genus,               ~Species,
  "g__Claroideoglomus", "s__Claroideoglomus_etunicatum",
  "g__Rhizophagus",     "s__Rhizophagus_irregularis",
  "g__Funneliformis",   "s__Funneliformis_mosseae"
)

# Filter m_ITS for just these 3 species
filtered_amf <- m_ITS %>%
  semi_join(target_amf_species, by = c("Genus", "Species"))

# Create cleaner label column
filtered_amf <- filtered_amf %>%
  mutate(AMF_label = case_when(
```

```
    Species == "s__Rhizophagus_irregularis" ~ "R_irregularis",
    Species == "s__Claroideoglomus_etunicatum" ~ "C_etunicatum",
    Species == "s__Funneliformis_mosseae" ~ "F_mosseae"
  ))

# Summarize abundance per sample per AMF
amf_wide <- filtered_amf %>%
  group_by(sample_name, AMF_label) %>%
  summarise(Abundance = sum(Abundance), .groups = "drop") %>%
  pivot_wider(names_from = AMF_label, values_from = Abundance, values_fill = 0)
```

```
# Define target AMF species
target_amf_species <- tribble(
  ~Genus,               ~Species,
  "g__Claroideoglomus", "s__Claroideoglomus_etunicatum",
  "g__Rhizophagus",     "s__Rhizophagus_irregularis",
  "g__Funneliformis",   "s__Funneliformis_mosseae"
)

# Filter ITS for just those 3 AMF species
filtered_amf <- m_ITS %>%
  semi_join(target_amf_species, by = c("Genus", "Species")) %>%
  mutate(AMF_label = case_when(
    Species == "s__Rhizophagus_irregularis" ~ "R_irregularis",
    Species == "s__Claroideoglomus_etunicatum" ~ "C_etunicatum",
    Species == "s__Funneliformis_mosseae" ~ "F_mosseae"
  ))

# Summarize abundance per sample per AMF species
amf_long <- filtered_amf %>%
  group_by(sample_name, AMF_label) %>%
  summarise(Abundance = sum(Abundance), .groups = "drop")

# Pivot to wide format: sample_name × each AMF species
amf_matrix <- amf_long %>%
  pivot_wider(names_from = AMF_label, values_from = Abundance, values_fill = 0)
```

```
# Get samples that overlap with AMF matrix
samples_to_keep <- amf_matrix$sample_name

# Filter and summarize genus-level abundances
genus_matrix <- m_16s %>%
  filter(sample_name %in% samples_to_keep) %>%
  filter(!is.na(Genus) & Genus != "") %>%
  filter(!str_detect(Genus, "uncultured|metagenome|environmental")) %>%
  group_by(sample_name, Genus) %>%
  summarise(Total_Abundance = sum(Abundance), .groups = "drop") %>%
  pivot_wider(names_from = Genus, values_from = Total_Abundance, values_fill = 0)
```

```
# Merge AMF and bacterial genus tables by sample_name
combined_matrix <- inner_join(amf_matrix, genus_matrix, by = "sample_name")
```

```r
library(Hmisc)

# Remove sample_name and convert to matrix
cor_input <- combined_matrix %>%
  select(-sample_name) %>%
  as.matrix()

# Run Spearman correlation
cor_results <- rcorr(cor_input, type = "spearman")

# Extract r values and p values
cor_r <- cor_results$r
cor_p <- cor_results$P
```

```r
# Define the AMF columns
amf_cols <- c("R_irregularis", "C_etunicatum", "F_mosseae")

# Get all other columns as bacterial genera
bacteria_cols <- setdiff(colnames(cor_r), amf_cols)

# Extract all AMF  Bacteria combinations into a table
cor_table <- expand.grid(AMF = amf_cols, Bacteria = bacteria_cols) %>%
  mutate(
    Spearman_r = mapply(function(a, b) cor_r[a, b], AMF, Bacteria),
    P_value = mapply(function(a, b) cor_p[a, b], AMF, Bacteria)
  ) %>%
  mutate(
    FDR = p.adjust(P_value, method = "BH"),
    direction = ifelse(Spearman_r > 0, "Positive", "Negative"),
    weight = abs(Spearman_r)
  ) %>%
  filter(P_value < 0.05) %>%
  arrange(desc(weight))
```

```r
# --- Edge Table ---
edges <- cor_table %>%
  rename(Source = AMF, Target = Bacteria) %>%
  select(Source, Target, weight, direction)

# --- Node Table ---
node_ids <- unique(c(edges$Source, edges$Target))

nodes <- data.frame(
  Id = node_ids,
  Label = node_ids,
  type = ifelse(node_ids %in% c("R_irregularis", "C_etunicatum", "F_mosseae"),
                "Fungus", "Bacteria")
)
```

```r
# === Final Edge Table ===
edges_gephi <- cor_table %>%
  rename(Source = AMF, Target = Bacteria) %>%
  select(Source, Target, weight, direction)
```

```r
# === Final Node Table ===
all_nodes <- unique(c(edges_gephi$Source, edges_gephi$Target))

nodes_gephi <- data.frame(
  Id = all_nodes,
  Label = all_nodes,
  type = ifelse(all_nodes %in% c("R_irregularis", "C_etunicatum", "F_mosseae"),
                "Fungus", "Bacteria")
)

# === Save to CSV for Gephi ===
write.csv(edges_gephi, "AMF_bacteria_edges_gephi.csv", row.names = FALSE)
write.csv(nodes_gephi, "AMF_bacteria_nodes_gephi.csv", row.names = FALSE)
```

Too much overlap in the above csvs so we are trimming the bacterial taxa

```r
# Top N bacteria per AMF species (sorted by absolute correlation)
top_cor_trimmed <- cor_table %>%
  group_by(AMF) %>%
  arrange(desc(weight)) %>%
  slice_head(n = 15) %>%
  ungroup()

# Make trimmed edge table
edges_trimmed <- top_cor_trimmed %>%
  rename(Source = AMF, Target = Bacteria) %>%
  select(Source, Target, weight, direction)

# Make trimmed node table
node_ids_trimmed <- unique(c(edges_trimmed$Source, edges_trimmed$Target))

nodes_trimmed <- data.frame(
  Id = node_ids_trimmed,
  Label = node_ids_trimmed,
  type = ifelse(node_ids_trimmed %in% c("R_irregularis", "C_etunicatum", "F_mosseae"),
                "Fungus", "Bacteria")
)

# Save to CSV
write.csv(edges_trimmed, "AMF_bacteria_edges_trimmed.csv", row.names = FALSE)
write.csv(nodes_trimmed, "AMF_bacteria_nodes_trimmed.csv", row.names = FALSE)
```

Fine tune for gehpi

```r
# --- Trim to Top 15 bacteria per AMF based on absolute correlation ---
top_cor_trimmed <- cor_table %>%
  group_by(AMF) %>%
  arrange(desc(abs(Spearman_r))) %>%
  slice_head(n = 10) %>%
  ungroup()

# --- Make Edge Table ---
```

```r
edges_gephi <- top_cor_trimmed %>%
  rename(Source = AMF, Target = Bacteria) %>%
  mutate(
    AMF_group = Source,
    Correlation = ifelse(Spearman_r > 0, "Positive", "Negative"),
    weight = abs(Spearman_r)
  ) %>%
  select(Source, Target, weight, direction = Correlation, AMF_group)
```

```r
# --- Calculate node strength (total weight of connections) ---
node_strength <- edges_gephi %>%
  pivot_longer(cols = c(Source, Target), values_to = "Id") %>%
  group_by(Id) %>%
  summarise(size = sum(weight))

# --- Prepare node table ---
all_nodes <- unique(c(edges_gephi$Source, edges_gephi$Target))

nodes_gephi <- data.frame(
  Id = all_nodes,
  Label = all_nodes,
  type = ifelse(all_nodes %in% c("R_irregularis", "C_etunicatum", "F_mosseae"),
                "Fungus", "Bacteria")
) %>%
  left_join(node_strength, by = "Id") %>%
  mutate(size = replace_na(size, 1)) %>%
  select(Id, Label, type, size)
```

```r
write.csv(edges_gephi, "M_bacteria_edges_colored(2).csv", row.names = FALSE)
write.csv(nodes_gephi, "M_bacteria_nodes_sized(2).csv", row.names = FALSE)
```

###ME Treatment###

```r
# Get the sample columns again
sample_columns_16s <- grep("^FFAR", colnames(ASV_16s), value = TRUE)

# Reshape to long format
asv_16s_long <- ASV_16s %>%
  pivot_longer(cols = all_of(sample_columns_16s),
               names_to = "sample_name",
               values_to = "Abundance") %>%
  mutate(Abundance = as.numeric(Abundance))

# Merge with metadata (important step)
m_16s <- asv_16s_long %>%
  inner_join(MD, by = "sample_name")

# Filter for ME treatment
m_16s_ME <- m_16s %>% filter(Treatment == "ME")
```

```r
# Define AMF species list
target_amf_species <- c(
```

```
  "s__Rhizophagus_irregularis",
  "s__Claroideoglomus_etunicatum",
  "s__Funneliformis_mosseae"
)

# Filter for ME + these species
# Create ME-filtered ITS table BEFORE filtering it
m_ITS_ME <- asv_ITS_long %>%
  inner_join(MD, by = "sample_name")

# Filter for ME + these species
m_ITS_ME <- m_ITS_ME %>%
  filter(Treatment == "ME") %>%
  filter(Species %in% target_amf_species)



# Check result
table(m_ITS_ME$Species)
```

```
##
## s__Claroideoglomus_etunicatum      s__Funneliformis_mosseae
##                           216                            54
##    s__Rhizophagus_irregularis
##                           378
```

```
# Create a readable AMF label

m_ITS_ME <- m_ITS_ME %>%
  mutate(AMF_label = case_when(
    Species == "s__Rhizophagus_irregularis" ~ "R_irregularis",
    Species == "s__Claroideoglomus_etunicatum" ~ "C_etunicatum",
    Species == "s__Funneliformis_mosseae" ~ "F_mosseae"
  ))

# Summarize AMF abundance per sample
amf_long_ME <- m_ITS_ME %>%
  group_by(sample_name, AMF_label) %>%
  summarise(Abundance = sum(Abundance), .groups = "drop")

# Pivot to wide format
amf_matrix_ME <- amf_long_ME %>%
  pivot_wider(names_from = AMF_label, values_from = Abundance, values_fill = 0)
```

```
# Filter and summarize genus-level abundances
genus_matrix_ME <- m_16s_ME %>%
  filter(!is.na(Genus) & Genus != "") %>%
  filter(!str_detect(Genus, "uncultured|metagenome|environmental")) %>%
  group_by(sample_name, Genus) %>%
  summarise(Total_Abundance = sum(Abundance), .groups = "drop") %>%
  pivot_wider(names_from = Genus, values_from = Total_Abundance, values_fill = 0)
```

```r
# Fix sample names before joining
amf_matrix_ME$sample_name <- trimws(amf_matrix_ME$sample_name)
genus_matrix_ME$sample_name <- trimws(genus_matrix_ME$sample_name)

# Merge AMF and bacterial genus data (ME samples only)
combined_matrix_ME <- inner_join(amf_matrix_ME, genus_matrix_ME, by = "sample_name")
```

```r
library(Hmisc)

# Drop sample_name and convert to numeric matrix
cor_input_ME <- combined_matrix_ME %>%
  select(-sample_name) %>%
  as.matrix()

# Run Spearman correlation
cor_results_ME <- rcorr(cor_input_ME, type = "spearman")

# Extract correlation (r) and p-values
cor_r_ME <- cor_results_ME$r
cor_p_ME <- cor_results_ME$P
```

```r
# Define the AMF columns
amf_cols_ME <- c("R_irregularis", "C_etunicatum", "F_mosseae")

# Get all other columns as bacterial genera
bacteria_cols_ME <- setdiff(colnames(cor_r_ME), amf_cols_ME)

# Extract AMF  Bacteria correlation pairs
cor_table_ME <- expand.grid(AMF = amf_cols_ME, Bacteria = bacteria_cols_ME) %>%
  mutate(
    Spearman_r = mapply(function(a, b) cor_r_ME[a, b], AMF, Bacteria),
    P_value = mapply(function(a, b) cor_p_ME[a, b], AMF, Bacteria)
  ) %>%
  mutate(
    FDR = p.adjust(P_value, method = "BH"),
    direction = ifelse(Spearman_r > 0, "Positive", "Negative"),
    weight = abs(Spearman_r)
  ) %>%
  filter(P_value < 0.05) %>%
  arrange(desc(weight))
```

```r
# Keep top 20 bacterial genera per AMF species
top_cor_trimmed_ME <- cor_table_ME %>%
  group_by(AMF) %>%
  arrange(desc(weight)) %>%
  slice_head(n = 20) %>%
  ungroup()
```

```r
# --- Edge Table ---
edges_ME <- top_cor_trimmed_ME %>%
  rename(Source = AMF, Target = Bacteria) %>%
  mutate(
```

```
    AMF_group = Source,
    Correlation = direction,
    weight = abs(Spearman_r)
  ) %>%
  select(Source, Target, weight, direction = Correlation, AMF_group)
```

```
# --- Node Strength Calculation (total edge weight per node) ---
node_strength_ME <- edges_ME %>%
  pivot_longer(cols = c(Source, Target), values_to = "Id") %>%
  group_by(Id) %>%
  summarise(size = sum(weight), .groups = "drop")

# --- Build Node Table ---
node_ids_ME <- unique(c(edges_ME$Source, edges_ME$Target))

nodes_ME <- data.frame(
  Id = node_ids_ME,
  Label = node_ids_ME,
  type = ifelse(node_ids_ME %in% c("R_irregularis", "C_etunicatum", "F_mosseae"),
                "Fungus", "Bacteria")
) %>%
  left_join(node_strength_ME, by = "Id") %>%
  mutate(size = replace_na(size, 1)) %>%
  select(Id, Label, type, size)
```

```
write.csv(edges_ME, "ME_edges_top20.csv", row.names = FALSE)
write.csv(nodes_ME, "ME_nodes_top20.csv", row.names = FALSE)
```