DspikeIn with TSE

Mitra Ghotbi

2025-06-20

Required Packages

```
#                          INSTALL CRAN PACKAGES

# Install missing CRAN packages
install.packages(setdiff(
  c(
    "stats", "dplyr", "ggplot2", "flextable", "ggpubr",
    "randomForest", "ggridges", "ggalluvial", "tibble",
    "matrixStats", "RColorBrewer", "ape", "rlang",
    "scales", "magrittr", "phangorn", "igraph", "tidyr",
    "xml2", "data.table", "reshape2", "vegan", "patchwork", "officer"
  ),
  installed.packages()[, "Package"]
))

# Load CRAN packages
lapply(c(
  "stats", "dplyr", "ggplot2", "flextable", "ggpubr", "randomForest",
  "ggridges", "ggalluvial", "tibble", "matrixStats", "RColorBrewer",
  "ape", "rlang", "scales", "magrittr", "phangorn", "igraph", "tidyr",
  "xml2", "data.table", "reshape2", "vegan", "patchwork", "officer"
), library, character.only = TRUE)


#                          INSTALL BIOCONDUCTOR PACKAGES


# Install BiocManager if not installed
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

# Install missing Bioconductor packages
BiocManager::install(setdiff(
  c(
```

```
    "phyloseq", "msa", "DESeq2", "ggtree", "edgeR",
    "Biostrings", "DECIPHER", "microbiome", "limma",
    "S4Vectors", "SummarizedExperiment", "TreeSummarizedExperiment"
  ),
  installed.packages()[, "Package"]
))

# Load Bioconductor packages
lapply(
  c(
    "phyloseq", "msa", "DESeq2", "edgeR", "Biostrings", "ggtree", "DECIPHER",
    "microbiome", "limma", "S4Vectors", "SummarizedExperiment", "TreeSummarizedExperiment"
  ),
  library,
  character.only = TRUE
)


#                          INSTALL DspikeIn FROM GITHUB


# To access the DspikeIn vignette for a detailed tutorial, use vignette("DspikeIn"), or browse
devtools::install_github("mghotbi/DspikeIn", build_vignettes = TRUE, dependencies = TRUE)
library(DspikeIn)
browseVignettes("DspikeIn")
vignette("DspikeIn")

## or

if (!requireNamespace("devtools", quietly = TRUE)) install.packages("devtools")
devtools::install_github("mghotbi/DspikeIn")

# Load DspikeIn only if installed
if ("DspikeIn" %in% installed.packages()[, "Package"]) {
  library(DspikeIn)
} else {
  stop("DspikeIn installation failed. Check errors above.")
}
```

Acknowledgments

The development of the DspikeIn package was made possible through the generous and pioneering efforts of the R and Bioconductor communities. We gratefully acknowledge the developers and maintainers of the following open-source packages, whose tools and infrastructure underpin our work: **Core infrastructure & data manipulation:** methods, stats, utils, graphics, grDevices,

data.table, dplyr, tibble, tidyr, reshape2, matrixStats, rlang, S4Vectors, grid, officer, xml2 **Statistical analysis & modeling:** DESeq2, edgeR, limma, randomForest, microbiome **Phylogenetics & microbiome structure:** phyloseq, TreeSummarizedExperiment, SummarizedExperiment, phangorn, ape, DECIPHER, msa, Biostrings **Network and graph analysis:** igraph, ggraph **Visualization & layout design:** ggplot2, ggrepel, ggpubr, ggnewscale, ggalluvial, ggtree, ggtreeExtra, ggstar, ggridges, patchwork, scales, RColorBrewer, flextable

These tools collectively empowered us to build a reproducible, modular, and extensible platform for robust absolute abundance quantification in microbial community analysis. We further acknowledge the broader scientific community working on absolute microbial quantification, spike-in calibration, and compositional data analysis, whose foundational insights directly informed the design and conceptual framework of DspikeIn.

DspikeIn

The DspikeIn package supports both phyloseq and TreeSummarizedExperiment formats to streamline microbial quantification across diverse experimental setups. It accommodates either a single spike-in taxon or synthetic community taxa with variable or equal spike-in volumes and copy numbers. The package offers a comprehensive suite of tools for AA quantification, addressing challenges through ten core functions: 1) validation of spiked species, 2) data preprocessing, 3) system-specific spiked species retrieval, 4) scaling factor calculation, 5) conversion to absolute abundance, 6) bias correction and normalization, 7) performance assessment, and 8) taxa exploration and filtering 9) network topology assessment 10) further analyses and visualization.

DspikeIn requirements

DspikeIn works with 7 taxonomic ranks

```
#             To remove strain from the taxonomic ranks

# DspikeIn works with 7 taxonomic ranks
# colnames(taxmat) <- c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species")

# Function to remove strain or bracketed info
remove_strain_info <- function(df) {
  pattern <- "Strain.*|strain.*|\\s*\\[.*?\\]|\\s*\\(.*?\\)"
  for (col in colnames(df)) {
    df[[col]] <- gsub(pattern, "", df[[col]])
    df[[col]] <- trimws(df[[col]])}
  return(df)}

# Apply to rowData of TSE
taxonomy <- as.data.frame(rowData(tse_16SOTU))

# Clean strain
```

```r
cleaned_taxonomy <- remove_strain_info(taxonomy)

#  drop "Strain" if present
if ("Strain" %in% colnames(cleaned_taxonomy)) {
  cleaned_taxonomy <- cleaned_taxonomy[, colnames(cleaned_taxonomy) != "Strain"]}

# Re-assign to TSE
rowData(tse_16SOTU) <- cleaned_taxonomy


#             To add species rank to the taxonomic ranks

# Make sure 'Genus' exists
if (!"Genus" %in% colnames(rowData(tse_16SOTU))) {
  stop("The 'Genus' column is missing in rowData.")}

# prepare taxonomy
taxonomy <- as.data.frame(rowData(tse_16SOTU))

# Handle missing genus labels
taxonomy$Genus[is.na(taxonomy$Genus) | taxonomy$Genus == ""] <- "Unknown"
taxonomy$Species <- paste0(taxonomy$Genus, "_OTU", seq_len(nrow(taxonomy)))

# Assign back to TSE
rowData(tse_16SOTU) <- taxonomy
```

Build TreeSummarizedExperiment file

for more information please refer to https://github.com/markrobinsonuzh/TreeSummarizedExperiment

```r
#                       Build TreeSummarizedExperiment (TSE)


otu <- read.csv("otu.csv", header = TRUE, sep = ",", row.names = 1)
otu_mat <- as.matrix(otu) # Convert to matrix
tax <- read.csv("tax.csv", header = TRUE, sep = ",", row.names = 1)
colnames(tax) <- c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species")
tax_mat <- as.matrix(tax) # Convert to matrix
meta <- read.csv("metadata.csv", header = TRUE, sep = ",", row.names = 1)
reference_seqs <- readDNAStringSet("dna-sequences.fasta", format = "fasta")
tse <- TreeSummarizedExperiment(
  assays = list(counts = otu_mat), # OTU table
  rowData = tax_mat, # Taxonomy information
  colData = meta, # Sample metadata
  rowTree = MyTree, # Phylogenetic tree
```

```
  rowSeqs = reference_seqs # Reference sequences)
```

Do all detected sample spike-in sequences cluster with the reference, and are their branch lengths statistically similar, supporting a common ancestor?

spike-in validation

All sample-derived sequences are forming a clade with the reference. We look for a monophyletic grouping of spike-in OTUs The clade is strongly supported (bootstrap around 100 percentage). The branch lengths and distances are in a biologically plausible range.

```
# Use the Neighbor-Joining method  based on a Jukes-Cantor distance matrix and plot the tree w
# we compare the Sanger read of Tetragenococcus halophilus with the FASTA sequence of Tetragen

library(Biostrings)
library(TreeSummarizedExperiment)
library(SummarizedExperiment)
library(DspikeIn)

# Get path to external data folder
extdata_path <- system.file("extdata", package = "DspikeIn")
list.files(extdata_path)
```

```
## [1] "Complete.graphml" "NoBasid.graphml"  "NoHubs.graphml"   "Ref.fasta"        "Sample.fast
```

```
data("physeq_16SOTU", package = "DspikeIn")
tse_16SOTU<-convert_phyloseq_to_tse(physeq_16SOTU)
tse_16SOTU <- tidy_phyloseq_tse(tse_16SOTU)

# Filter TSE object to keep only Bacteria and Archaea

tse_16SOTU <- tse_16SOTU[
  rowData(tse_16SOTU)$Kingdom %in% c("Bacteria", "Archaea"),]


library(Biostrings)
#
# # Subset the TSE object to include only Tetragenococcus
# tetragenococcus_tse <- tse_16SOTU[
#   rowData(tse_16SOTU)$Genus == "Tetragenococcus" &
#     !is.na(rownames(tse_16SOTU)) &
#     rownames(tse_16SOTU) != "", ]
#
# ref_sequences <- referenceSeq(tetragenococcus_tse)
```

```
#
# # Convert to DNAStringSet if needed
# ref_sequences <- Biostrings::DNAStringSet(ref_sequences)
# Biostrings::writeXStringSet(ref_sequences, filepath = "Sample.fasta")
```

```
ref_fasta <- system.file("extdata", "Ref.fasta", package = "DspikeIn")
sample_fasta <- system.file("extdata", "Sample.fasta", package = "DspikeIn")
```

```
result <- validate_spikein_clade(
  reference_fasta = ref_fasta,
  sample_fasta = sample_fasta,
  bootstrap = 200,
  output_prefix = NULL)
```

```
## use default substitution matrix
```

```
# result$tree_plot
```

Did spike-ins behave as expected across all samples?

Tip labels= OTU/ASV names Branch length numbers= Actual evolutionary distances (small = very similar) Prevalence stars How frequently the OTU occurs across samples Blue bar ring= Log10 mean abundance Outer colored tiles= The metadata variable you choose (e.g., Animal.type)

```
#data("physeq_16SOTU", package = "DspikeIn")
tse_16SOTU
```

```
## class: TreeSummarizedExperiment
## dim: 9242 312
## metadata(0):
## assays(1): counts
## rownames(9242): 020e00d90ba97c5898944ab6f7b1b7c9 b00466354053c9065c8aa3d6fbb33eaa ... 17f00
##   5a28da69cc65fb06e686e6260ecbfe0e
## rowData names(7): Kingdom Phylum ... Genus Species
## colnames(312): blank.20818_S96 NTCswab.20626_S96 ... UHM998.20618_S95 UHM999.20617_S83
## colData names(34): sample.id X16S.biosample ... swab.presence MK.spike
```

```
## reducedDimNames(0):
## mainExpName: NULL
```

```
## altExpNames(0):
## rowLinks: a LinkDataFrame (9242 rows)
## rowTree: 1 phylo tree(s) (9255 leaves)
## colLinks: NULL
## colTree: NULL
## referenceSeq: a DNAStringSet (9242 sequences)
```

```r
library(ggstar)
library(ggplot2)

# filter your object to only include spike-in taxa beforehand:
# change the OTU IDs for easy detection
# Big stars = detected in many samples
# Small stars = rarely detected
# log10(Mean Abundance) Bars= Color intensity reflects mean abundance.
# The log-transformed average abundance of each OTU across all samples where it appears.
# Extreme blue may signal unintended over-representation.
# Metadata Ring = factor of your interest e.g. Animal.type
# Each OTU is colored by where it was observed.
# Branch length numbers=    Actual evolutionary distances (small = very similar)


library(DspikeIn)
library(TreeSummarizedExperiment)

# ---- 1. Subset taxa where Genus is Tetragenococcus ----
spikein_tse <- tse_16SOTU[
  rowData(tse_16SOTU)$Genus == "Tetragenococcus", ]


# ---- 2. Diagnostic plot (tree-based) ----
ps <- plot_spikein_tree_diagnostic(
  obj = spikein_tse,
  metadata_var = "Animal.type",
  save_plot = FALSE )
```

Pre_processing

merges monophyletic ASVs/OTUs

Spiked species and related parameters for 16S

```r
#                        PREREQUISITE FOR 16S & CALCULATE SPIKED %


# Define spiked species and related parameters**

library(flextable)
library(DspikeIn)
library(TreeSummarizedExperiment)
library(SummarizedExperiment)
library(dplyr)

# Define spike-in parameters
spiked_cells <- 1847
species_name <- spiked_species <- c("Tetragenococcus_halophilus", "Tetragenococcus_sp.")

merged_spiked_species <- "Tetragenococcus"

# If you prefer Genus-level matching
# species_name <- spiked_species <- c("Tetragenococcus")
# merged_spiked_species <- "Tetragenococcus"



# ------------------------------------------
# Subset taxa for the spike-in species
tse_16SOTU_spiked_taxa <- tse_16SOTU[rowData(tse_16SOTU)$Species %in% species_name, ]

# (OTU) IDs
hashcodes <- rownames(tse_16SOTU_spiked_taxa)

# ------------------------------------------
# Subset samples based on spike-in volume (e.g., "1" or "2")
tse_16SOTU_spiked_samples <- tse_16SOTU[, colData(tse_16SOTU)$spiked.volume %in% c("2", "1") ]

# ------------------------------------------
# Merge spiked OTUs using the DspikeIn function
# merges monophyletic ASVs/OTUs

# The function Pre_processing_species() merges ASVs/OTUs
# of a species using "sum" or "max" methods, preserving taxonomic,
# phylogenetic, and sequencing data.



library(DspikeIn)
library(TreeSummarizedExperiment)
library(SummarizedExperiment)
```

```r
output_rds <- file.path(tempdir(), "merged_tse_sum.rds")

Spiked_16S_sum_scaled <- Pre_processing_species(
  tse_16SOTU_spiked_samples,
  species_name,
  merge_method = "sum",
  output_file = output_rds)

# ------------------------------------------
# Calculate spike-in recovery percentage
Perc <- calculate_spike_percentage(
  Spiked_16S_sum_scaled,
  merged_spiked_species,
  passed_range = c(0.1, 20))
```

Spiked species and related parameters for ITS

```r
#                       PREREQUISITE FOR ITS & CALCULATE SPIKED %


# Define spiked species and related parameters**


# Define the spiked species
# spiked_cells <- 733
# species_name <- spiked_species <- merged_spiked_species <- "Dekkera_bruxellensis"

# Subset taxa for spiked species
# Dekkera <- phyloseq::subset_taxa(
# physeq_ITSOTU,
# Species %in% species_name)

# hashcodes <- row.names(phyloseq::tax_table(Dekkera))

# Subset samples based on spiked volume
# physeq_ITSOTU_spiked <- phyloseq::subset_samples(physeq_ITSOTU, spiked.volume %in% c("2", "1

# if TSE format
# tse_ITSOTU <- convert_phyloseq_to_tse(physeq_ITSOTU)
# physeq_ITSOTU_spiked <- tse_ITSOTU[, tse_ITSOTU$spiked.volume %in% c("2", "1")]
```

Calculating Scaling Factors

```
#                       CALCULATE SCALING FACTORS

result <- calculate_spikeIn_factors(
  Spiked_16S_sum_scaled,
  spiked_cells = spiked_cells,
  merged_spiked_species = species_name)

# View extracted outputs
result$spiked_species_reads # Merged spiked species name
```

```
##                                      Sample Spiked_Reads
## spiked.blank.20433_S84 spiked.blank.20433_S84            8
## spiked.blank.20817_S84 spiked.blank.20817_S84        47066
## Std2uL.20625_S84             Std2uL.20625_S84        62433
## StdSwab1uL.20624_S72     StdSwab1uL.20624_S72        17639
## STP1719.20422_S47           STP1719.20422_S47        14554
## STP213.20423_S59             STP213.20423_S59           83
## STP268.20424_S71             STP268.20424_S71           17
## STP544.20419_S11             STP544.20419_S11         2259
## STP570.20420_S23             STP570.20420_S23          822
## STP579.20421_S35             STP579.20421_S35         1759
## STP614.20418_S94             STP614.20418_S94            0
## UHM1000.20604_S22           UHM1000.20604_S22          118
## UHM1001.20609_S82           UHM1001.20609_S82           93
## UHM1007.20622_S48           UHM1007.20622_S48          118
## UHM1009.20614_S47           UHM1009.20614_S47          116
## UHM1010.20621_S36           UHM1010.20621_S36          979
## UHM1011.20606_S46           UHM1011.20606_S46          130
## UHM1024.20620_S24           UHM1024.20620_S24          994
## UHM1026.20607_S58           UHM1026.20607_S58          396
## UHM1028.20613_S35           UHM1028.20613_S35          898
## UHM1032.20605_S34           UHM1032.20605_S34         1616
## UHM1033.20619_S12           UHM1033.20619_S12        37416
## UHM1034.20616_S71           UHM1034.20616_S71            4
## UHM1035.20611_S11           UHM1035.20611_S11          210
## UHM1036.20612_S23           UHM1036.20612_S23          249
## UHM1052.20615_S59           UHM1052.20615_S59          150
## UHM1060.20723_S1             UHM1060.20723_S1        24617
## UHM1065.20724_S13           UHM1065.20724_S13         8179
## UHM1068.20732_S14           UHM1068.20732_S14          243
## UHM1069.20742_S39           UHM1069.20742_S39          343
## UHM1070.20725_S25           UHM1070.20725_S25         1741
## UHM1071.20733_S26           UHM1071.20733_S26           56
## UHM1072.20734_S38           UHM1072.20734_S38         1486
```

```
## UHM1073.20735_S50          UHM1073.20735_S50          39
## UHM1075.20726_S37          UHM1075.20726_S37        1179
## UHM1077.20736_S62          UHM1077.20736_S62          73
## UHM1078.20727_S49          UHM1078.20727_S49         176
## UHM1080.20737_S74          UHM1080.20737_S74          69
## UHM1081.20728_S61          UHM1081.20728_S61        1772
## UHM1088.20738_S86          UHM1088.20738_S86         240
## UHM1090.20739_S3            UHM1090.20739_S3        5303
## UHM1093.20729_S73          UHM1093.20729_S73         336
## UHM1095.20730_S85          UHM1095.20730_S85        6219
## UHM1097.20623_S60          UHM1097.20623_S60        6060
## UHM1099.20608_S70          UHM1099.20608_S70          24
## UHM1100.20788_S21          UHM1100.20788_S21          75
## UHM1102.20789_S33          UHM1102.20789_S33         108
## UHM1104.20790_S45          UHM1104.20790_S45          86
## UHM1105.20791_S57          UHM1105.20791_S57         148
## UHM1109.20531_S1            UHM1109.20531_S1         116
## UHM1110.20568_S65          UHM1110.20568_S65          50
## UHM1113.20792_S69          UHM1113.20792_S69          79
## UHM1114.20793_S81          UHM1114.20793_S81         370
## UHM1115.20794_S93          UHM1115.20794_S93         181
## UHM1117.20795_S10          UHM1117.20795_S10          61
## UHM1118.20796_S22          UHM1118.20796_S22         248
## UHM1120.20797_S34          UHM1120.20797_S34         718
## UHM1124.20798_S46          UHM1124.20798_S46         587
## UHM1126.20799_S58          UHM1126.20799_S58         196
## UHM1128.20800_S70          UHM1128.20800_S70         183
## UHM1140.20555_S4            UHM1140.20555_S4         130
## UHM1145.20801_S82          UHM1145.20801_S82          45
## UHM1163.20405_S33          UHM1163.20405_S33         150
## UHM1164.20402_S92          UHM1164.20402_S92           3
## UHM1169.20552_S63          UHM1169.20552_S63        6591
## UHM1171.20579_S7            UHM1171.20579_S7         97
## UHM1176.20404_S21          UHM1176.20404_S21          67
## UHM1177.20546_S86          UHM1177.20546_S86        1558
## UHM1182.20576_S66          UHM1182.20576_S66         233
## UHM1210.20802_S94          UHM1210.20802_S94          98
## UHM1212.20803_S11          UHM1212.20803_S11         312
## UHM1217.20804_S23          UHM1217.20804_S23         198
## UHM1218.20805_S35          UHM1218.20805_S35         151
## UHM1219.20806_S47          UHM1219.20806_S47          57
## UHM1220.20807_S59          UHM1220.20807_S59         104
## UHM1221.20808_S71          UHM1221.20808_S71         105
## UHM1222.20809_S83          UHM1222.20809_S83        3039
## UHM1223.20810_S95          UHM1223.20810_S95          69
## UHM1225.20811_S12          UHM1225.20811_S12         229
## UHM1227.20812_S24          UHM1227.20812_S24         878
## UHM1228.20813_S36          UHM1228.20813_S36          69
```

```
## UHM1237.20814_S48          UHM1237.20814_S48         413
## UHM1240.20566_S41          UHM1240.20566_S41         683
## UHM1246.20815_S60          UHM1246.20815_S60         573
## UHM1247.20816_S72          UHM1247.20816_S72         235
## UHM1248.20575_S54          UHM1248.20575_S54        3287
## UHM1256.20570_S89          UHM1256.20570_S89         901
## UHM1260.20596_S21          UHM1260.20596_S21         614
## UHM1270.20577_S78          UHM1270.20577_S78         217
## UHM1271.20397_S32          UHM1271.20397_S32         896
## UHM1272.20398_S44          UHM1272.20398_S44        1529
## UHM1274.20554_S87          UHM1274.20554_S87         467
## UHM1275.20597_S33          UHM1275.20597_S33        1203
## UHM1282.20599_S57          UHM1282.20599_S57        3820
## UHM1287.20543_S50          UHM1287.20543_S50        1979
## UHM1291.20416_S70          UHM1291.20416_S70         261
## UHM1296.20550_S39          UHM1296.20550_S39          76
## UHM1319.20561_S76          UHM1319.20561_S76        1567
## UHM1324.20413_S34          UHM1324.20413_S34         433
## UHM1327.20545_S74          UHM1327.20545_S74          73
## UHM1328.20572_S18          UHM1328.20572_S18        3350
## UHM1334.20417_S82          UHM1334.20417_S82         155
## UHM1338.20399_S56          UHM1338.20399_S56         760
## UHM1341.20602_S93          UHM1341.20602_S93        8050
## UHM1356.20541_S26          UHM1356.20541_S26         810
## UHM1380.20580_S19          UHM1380.20580_S19         308
## UHM1383.20594_S92          UHM1383.20594_S92         421
## UHM1385.20563_S5            UHM1385.20563_S5       20431
## UHM1399.20756_S17          UHM1399.20756_S17         918
## UHM1400.20757_S29          UHM1400.20757_S29         316
## UHM1401.20758_S41          UHM1401.20758_S41        2414
## UHM1402.20759_S53          UHM1402.20759_S53         166
## UHM1403.20760_S65          UHM1403.20760_S65         309
## UHM1405.20761_S77          UHM1405.20761_S77         219
## UHM1406.20762_S89          UHM1406.20762_S89         223
## UHM1414.20763_S6            UHM1414.20763_S6         107
## UHM1419.20764_S18          UHM1419.20764_S18         488
## UHM1427.20389_S31          UHM1427.20389_S31         103
## UHM1428.20390_S43          UHM1428.20390_S43           0
## UHM1429.20391_S55          UHM1429.20391_S55          29
## UHM1430.20392_S67          UHM1430.20392_S67          11
## UHM1432.20393_S79          UHM1432.20393_S79          18
## UHM1435.20388_S19          UHM1435.20388_S19           0
## UHM162.20560_S64            UHM162.20560_S64         704
## UHM198.20585_S79            UHM198.20585_S79        3649
## UHM20.3314_S52                UHM20.3314_S52         43
## UHM20.3315_S64                UHM20.3315_S64        349
## UHM204.20409_S81            UHM204.20409_S81         66
## UHM206.20410_S93            UHM206.20410_S93          0
```

```
## UHM207.20593_S80          UHM207.20593_S80         200
## UHM208.20411_S10          UHM208.20411_S10         391
## UHM211.20406_S45          UHM211.20406_S45         164
## UHM215.20408_S69          UHM215.20408_S69           9
## UHM216.20429_S36          UHM216.20429_S36          63
## UHM219.20430_S48          UHM219.20430_S48       21041
## UHM236.20431_S60          UHM236.20431_S60          41
## UHM238.20407_S57          UHM238.20407_S57         120
## UHM245.20538_S85          UHM245.20538_S85         108
## UHM252.20558_S40          UHM252.20558_S40        1090
## UHM267.20400_S68          UHM267.20400_S68         131
## UHM274.20581_S31          UHM274.20581_S31        2025
## UHM276.20586_S91          UHM276.20586_S91        1169
## UHM280.20401_S80          UHM280.20401_S80         389
## UHM286.20425_S83          UHM286.20425_S83          30
## UHM289.20426_S95          UHM289.20426_S95           4
## UHM294.20427_S12          UHM294.20427_S12         610
## UHM298.20600_S69          UHM298.20600_S69         404
## UHM325.20548_S15          UHM325.20548_S15         584
## UHM337.20412_S22          UHM337.20412_S22         261
## UHM354.20535_S49          UHM354.20535_S49         355
## UHM356.20415_S58          UHM356.20415_S58         411
## UHM369.20773_S31          UHM369.20773_S31         233
## UHM370.20774_S43          UHM370.20774_S43          88
## UHM372.20775_S55          UHM372.20775_S55          30
## UHM373.20776_S67          UHM373.20776_S67          84
## UHM374.20777_S79          UHM374.20777_S79          69
## UHM375.20778_S91          UHM375.20778_S91         716
## UHM377.20779_S8            UHM377.20779_S8         477
## UHM38.3376_S36              UHM38.3376_S36           0
## UHM386.20781_S32          UHM386.20781_S32         234
## UHM387.20782_S44          UHM387.20782_S44          58
## UHM414.20583_S55          UHM414.20583_S55          85
## UHM418.20765_S30          UHM418.20765_S30         270
## UHM422.20766_S42          UHM422.20766_S42        1147
## UHM425.20767_S54          UHM425.20767_S54         118
## UHM426.20534_S37          UHM426.20534_S37        4598
## UHM428.20544_S62          UHM428.20544_S62         454
## UHM429.20559_S52          UHM429.20559_S52        1283
## UHM435.20547_S3            UHM435.20547_S3        1718
## UHM437.20768_S66          UHM437.20768_S66          49
## UHM439.20564_S17          UHM439.20564_S17          71
## UHM44.3526_S31              UHM44.3526_S31        1281
## UHM445.20569_S77          UHM445.20569_S77          26
## UHM447.20783_S56          UHM447.20783_S56          64
## UHM448.20769_S78          UHM448.20769_S78         424
## UHM45.3539_S92              UHM45.3539_S92           0
## UHM454.20770_S90          UHM454.20770_S90         278
```

```
## UHM455.20785_S80            UHM455.20785_S80          792
## UHM458.20786_S92            UHM458.20786_S92           69
## UHM459.20787_S9              UHM459.20787_S9          120
## UHM461.20771_S7              UHM461.20771_S7          242
## UHM467.20772_S19            UHM467.20772_S19           47
## UHM470.20533_S25            UHM470.20533_S25         1220
## UHM476.20414_S46            UHM476.20414_S46         1138
## UHM478.20549_S27            UHM478.20549_S27          429
## UHM479.20551_S51            UHM479.20551_S51        14439
## UHM481.20403_S9              UHM481.20403_S9          227
## UHM482.20590_S44            UHM482.20590_S44          233
## UHM483.20603_S10            UHM483.20603_S10           91
## UHM519.20582_S43            UHM519.20582_S43          136
## UHM520.20573_S30            UHM520.20573_S30          253
## UHM746.21478_S117          UHM746.21478_S117        85168
## UHM747.21477_S106          UHM747.21477_S106        82287
## UHM748.21467_S170          UHM748.21467_S170        55291
## UHM748.21487_S129          UHM748.21487_S129        36462
## UHM749.21479_S128          UHM749.21479_S128        62780
## UHM759.21466_S159          UHM759.21466_S159        76705
## UHM759.21486_S118          UHM759.21486_S118        16500
## UHM775.21485_S107          UHM775.21485_S107        43046
## UHM776.21482_S161          UHM776.21482_S161        58165
## UHM777.21484_S183          UHM777.21484_S183        23485
## UHM779.21468_S181          UHM779.21468_S181        80976
## UHM779.21488_S140          UHM779.21488_S140            9
## UHM782.21480_S139          UHM782.21480_S139        38354
## UHM810.21472_S138          UHM810.21472_S138        94712
## UHM811.21471_S127          UHM811.21471_S127        30104
## UHM813.21481_S150          UHM813.21481_S150        64962
## UHM818.21469_S105          UHM818.21469_S105        50417
## UHM818.21489_S151          UHM818.21489_S151           19
## UHM819.21473_S149          UHM819.21473_S149        83424
## UHM820.21470_S116          UHM820.21470_S116        66075
## UHM820.21490_S162          UHM820.21490_S162          515
## UHM827.21474_S160          UHM827.21474_S160        83467
## UHM829.21476_S182          UHM829.21476_S182        69357
## UHM832.21483_S172          UHM832.21483_S172        68494
## UHM836.20385_S78            UHM836.20385_S78            0
## UHM837.20386_S90            UHM837.20386_S90            0
## UHM838.20387_S7              UHM838.20387_S7           11
## UHM891.20384_S66            UHM891.20384_S66           85
## UHM892.20532_S13            UHM892.20532_S13         1164
## UHM893.20595_S9              UHM893.20595_S9           16
## UHM894.20540_S14            UHM894.20540_S14           36
## UHM895.20536_S61            UHM895.20536_S61         2116
## UHM896.20601_S81            UHM896.20601_S81            0
## UHM897.20591_S56            UHM897.20591_S56            0
```

```
## UHM898.20394_S91             UHM898.20394_S91            0
## UHM899.20588_S20             UHM899.20588_S20           24
## UHM900.20395_S8               UHM900.20395_S8            0
## UHM901.20542_S38             UHM901.20542_S38           95
## UHM902.20584_S67             UHM902.20584_S67         1217
## UHM903.20587_S8               UHM903.20587_S8          470
## UHM904.20567_S53             UHM904.20567_S53           45
## UHM905.20598_S45             UHM905.20598_S45           89
## UHM906.20565_S29             UHM906.20565_S29            0
## UHM907.20592_S68             UHM907.20592_S68           30
## UHM908.20396_S20             UHM908.20396_S20            0
## UHM909.20557_S28             UHM909.20557_S28           87
## UHM910.20562_S88             UHM910.20562_S88           34
## UHM965.20537_S73             UHM965.20537_S73          464
## UHM966.20743_S51             UHM966.20743_S51          461
## UHM967.20744_S63             UHM967.20744_S63            0
## UHM968.20571_S6               UHM968.20571_S6        33012
## UHM969.20745_S75             UHM969.20745_S75           30
## UHM971.20746_S87             UHM971.20746_S87           10
## UHM973.20578_S90             UHM973.20578_S90           46
## UHM974.20432_S72             UHM974.20432_S72            0
## UHM975.20747_S4               UHM975.20747_S4           33
## UHM977.20748_S16             UHM977.20748_S16          100
## UHM978.20749_S28             UHM978.20749_S28           91
## UHM979.20750_S40             UHM979.20750_S40           35
## UHM980.20731_S2               UHM980.20731_S2          248
## UHM981.20539_S2               UHM981.20539_S2          766
## UHM982.20740_S15             UHM982.20740_S15          161
## UHM983.20556_S16             UHM983.20556_S16        12547
## UHM984.20751_S52             UHM984.20751_S52           26
## UHM985.20752_S64             UHM985.20752_S64          801
## UHM988.20753_S76             UHM988.20753_S76          603
## UHM989.20754_S88             UHM989.20754_S88           17
## UHM991.20755_S5               UHM991.20755_S5         1868
## UHM993.20741_S27             UHM993.20741_S27           52
## UHM996.20610_S94             UHM996.20610_S94            0
## UHM997.20553_S75             UHM997.20553_S75            0
## UHM998.20618_S95             UHM998.20618_S95         2610
## UHM999.20617_S83             UHM999.20617_S83           72
```

```
result$total_reads # Total reads detected for the spike
```

```
##                                      Sample Total_Reads
## spiked.blank.20433_S84 spiked.blank.20433_S84           8
## spiked.blank.20817_S84 spiked.blank.20817_S84       47103
## Std2uL.20625_S84             Std2uL.20625_S84       62444
## StdSwab1uL.20624_S72     StdSwab1uL.20624_S72       24897
```

```
## STP1719.20422_S47          STP1719.20422_S47       19142
## STP213.20423_S59            STP213.20423_S59        8462
## STP268.20424_S71            STP268.20424_S71        6968
## STP544.20419_S11            STP544.20419_S11        2340
## STP570.20420_S23            STP570.20420_S23        2647
## STP579.20421_S35            STP579.20421_S35        5193
## STP614.20418_S94            STP614.20418_S94           5
## UHM1000.20604_S22          UHM1000.20604_S22        5538
## UHM1001.20609_S82          UHM1001.20609_S82        3957
## UHM1007.20622_S48          UHM1007.20622_S48       11042
## UHM1009.20614_S47          UHM1009.20614_S47       22828
## UHM1010.20621_S36          UHM1010.20621_S36        6453
## UHM1011.20606_S46          UHM1011.20606_S46       11406
## UHM1024.20620_S24          UHM1024.20620_S24        2889
## UHM1026.20607_S58          UHM1026.20607_S58        3207
## UHM1028.20613_S35          UHM1028.20613_S35        2555
## UHM1032.20605_S34          UHM1032.20605_S34        2472
## UHM1033.20619_S12          UHM1033.20619_S12       39314
## UHM1034.20616_S71          UHM1034.20616_S71        3389
## UHM1035.20611_S11          UHM1035.20611_S11        1837
## UHM1036.20612_S23          UHM1036.20612_S23        2327
## UHM1052.20615_S59          UHM1052.20615_S59       16010
## UHM1060.20723_S1            UHM1060.20723_S1       27549
## UHM1065.20724_S13          UHM1065.20724_S13       10557
## UHM1068.20732_S14          UHM1068.20732_S14        2526
## UHM1069.20742_S39          UHM1069.20742_S39       16428
## UHM1070.20725_S25          UHM1070.20725_S25       10491
## UHM1071.20733_S26          UHM1071.20733_S26        2028
## UHM1072.20734_S38          UHM1072.20734_S38        3425
## UHM1073.20735_S50          UHM1073.20735_S50        2731
## UHM1075.20726_S37          UHM1075.20726_S37        7732
## UHM1077.20736_S62          UHM1077.20736_S62        9489
## UHM1078.20727_S49          UHM1078.20727_S49        4532
## UHM1080.20737_S74          UHM1080.20737_S74        8886
## UHM1081.20728_S61          UHM1081.20728_S61        2033
## UHM1088.20738_S86          UHM1088.20738_S86        6274
## UHM1090.20739_S3            UHM1090.20739_S3       12258
## UHM1093.20729_S73          UHM1093.20729_S73        2009
## UHM1095.20730_S85          UHM1095.20730_S85       11680
## UHM1097.20623_S60          UHM1097.20623_S60       10958
## UHM1099.20608_S70          UHM1099.20608_S70        3670
## UHM1100.20788_S21          UHM1100.20788_S21        4879
## UHM1102.20789_S33          UHM1102.20789_S33        2410
## UHM1104.20790_S45          UHM1104.20790_S45        3410
## UHM1105.20791_S57          UHM1105.20791_S57        2360
## UHM1109.20531_S1            UHM1109.20531_S1        3122
## UHM1110.20568_S65          UHM1110.20568_S65        6965
## UHM1113.20792_S69          UHM1113.20792_S69       16025
```

```
## UHM1114.20793_S81          UHM1114.20793_S81        5041
## UHM1115.20794_S93          UHM1115.20794_S93        5256
## UHM1117.20795_S10          UHM1117.20795_S10        3976
## UHM1118.20796_S22          UHM1118.20796_S22        5348
## UHM1120.20797_S34          UHM1120.20797_S34        2506
## UHM1124.20798_S46          UHM1124.20798_S46        6294
## UHM1126.20799_S58          UHM1126.20799_S58        5941
## UHM1128.20800_S70          UHM1128.20800_S70        2595
## UHM1140.20555_S4            UHM1140.20555_S4       10197
## UHM1145.20801_S82          UHM1145.20801_S82        3319
## UHM1163.20405_S33          UHM1163.20405_S33        5402
## UHM1164.20402_S92          UHM1164.20402_S92         911
## UHM1169.20552_S63          UHM1169.20552_S63        7438
## UHM1171.20579_S7            UHM1171.20579_S7        4952
## UHM1176.20404_S21          UHM1176.20404_S21        5583
## UHM1177.20546_S86          UHM1177.20546_S86        5901
## UHM1182.20576_S66          UHM1182.20576_S66        2437
## UHM1210.20802_S94          UHM1210.20802_S94        5710
## UHM1212.20803_S11          UHM1212.20803_S11        5886
## UHM1217.20804_S23          UHM1217.20804_S23        6804
## UHM1218.20805_S35          UHM1218.20805_S35        3493
## UHM1219.20806_S47          UHM1219.20806_S47        2276
## UHM1220.20807_S59          UHM1220.20807_S59        3906
## UHM1221.20808_S71          UHM1221.20808_S71        2137
## UHM1222.20809_S83          UHM1222.20809_S83        8953
## UHM1223.20810_S95          UHM1223.20810_S95        4269
## UHM1225.20811_S12          UHM1225.20811_S12        5986
## UHM1227.20812_S24          UHM1227.20812_S24        9891
## UHM1228.20813_S36          UHM1228.20813_S36        5503
## UHM1237.20814_S48          UHM1237.20814_S48        5591
## UHM1240.20566_S41          UHM1240.20566_S41       12987
## UHM1246.20815_S60          UHM1246.20815_S60        5004
## UHM1247.20816_S72          UHM1247.20816_S72        3378
## UHM1248.20575_S54          UHM1248.20575_S54       13609
## UHM1256.20570_S89          UHM1256.20570_S89       17303
## UHM1260.20596_S21          UHM1260.20596_S21       11405
## UHM1270.20577_S78          UHM1270.20577_S78        1427
## UHM1271.20397_S32          UHM1271.20397_S32        9272
## UHM1272.20398_S44          UHM1272.20398_S44        8652
## UHM1274.20554_S87          UHM1274.20554_S87        8200
## UHM1275.20597_S33          UHM1275.20597_S33        9805
## UHM1282.20599_S57          UHM1282.20599_S57       13538
## UHM1287.20543_S50          UHM1287.20543_S50       11499
## UHM1291.20416_S70          UHM1291.20416_S70       10486
## UHM1296.20550_S39          UHM1296.20550_S39        8543
## UHM1319.20561_S76          UHM1319.20561_S76        9848
## UHM1324.20413_S34          UHM1324.20413_S34       10595
## UHM1327.20545_S74          UHM1327.20545_S74       10782
```

```
## UHM1328.20572_S18          UHM1328.20572_S18       16676
## UHM1334.20417_S82          UHM1334.20417_S82        6421
## UHM1338.20399_S56          UHM1338.20399_S56       10825
## UHM1341.20602_S93          UHM1341.20602_S93       21036
## UHM1356.20541_S26          UHM1356.20541_S26        9245
## UHM1380.20580_S19          UHM1380.20580_S19       11947
## UHM1383.20594_S92          UHM1383.20594_S92        6453
## UHM1385.20563_S5            UHM1385.20563_S5       24006
## UHM1399.20756_S17          UHM1399.20756_S17        5659
## UHM1400.20757_S29          UHM1400.20757_S29        8433
## UHM1401.20758_S41          UHM1401.20758_S41        6518
## UHM1402.20759_S53          UHM1402.20759_S53        7091
## UHM1403.20760_S65          UHM1403.20760_S65        7923
## UHM1405.20761_S77          UHM1405.20761_S77        5923
## UHM1406.20762_S89          UHM1406.20762_S89       10431
## UHM1414.20763_S6            UHM1414.20763_S6        6573
## UHM1419.20764_S18          UHM1419.20764_S18        5863
## UHM1427.20389_S31          UHM1427.20389_S31        5228
## UHM1428.20390_S43          UHM1428.20390_S43        4532
## UHM1429.20391_S55          UHM1429.20391_S55        3859
## UHM1430.20392_S67          UHM1430.20392_S67        5040
## UHM1432.20393_S79          UHM1432.20393_S79        4047
## UHM1435.20388_S19          UHM1435.20388_S19        9292
## UHM162.20560_S64            UHM162.20560_S64       14141
## UHM198.20585_S79            UHM198.20585_S79       11058
## UHM20.3314_S52                UHM20.3314_S52       31095
## UHM20.3315_S64                UHM20.3315_S64       66533
## UHM204.20409_S81            UHM204.20409_S81        5723
## UHM206.20410_S93            UHM206.20410_S93         523
## UHM207.20593_S80            UHM207.20593_S80        8319
## UHM208.20411_S10            UHM208.20411_S10        7217
## UHM211.20406_S45            UHM211.20406_S45        7727
## UHM215.20408_S69            UHM215.20408_S69       19236
## UHM216.20429_S36            UHM216.20429_S36        7081
## UHM219.20430_S48            UHM219.20430_S48       21661
## UHM236.20431_S60            UHM236.20431_S60        4428
## UHM238.20407_S57            UHM238.20407_S57        4161
## UHM245.20538_S85            UHM245.20538_S85       32445
## UHM252.20558_S40            UHM252.20558_S40        8412
## UHM267.20400_S68            UHM267.20400_S68       12606
## UHM274.20581_S31            UHM274.20581_S31       15008
## UHM276.20586_S91            UHM276.20586_S91       13645
## UHM280.20401_S80            UHM280.20401_S80        7442
## UHM286.20425_S83            UHM286.20425_S83        2127
## UHM289.20426_S95            UHM289.20426_S95          13
## UHM294.20427_S12            UHM294.20427_S12        5674
## UHM298.20600_S69            UHM298.20600_S69       22131
## UHM325.20548_S15            UHM325.20548_S15        7768
```

```
## UHM337.20412_S22           UHM337.20412_S22        4612
## UHM354.20535_S49           UHM354.20535_S49       11303
## UHM356.20415_S58           UHM356.20415_S58        9725
## UHM369.20773_S31           UHM369.20773_S31       10033
## UHM370.20774_S43           UHM370.20774_S43        4214
## UHM372.20775_S55           UHM372.20775_S55         343
## UHM373.20776_S67           UHM373.20776_S67        5140
## UHM374.20777_S79           UHM374.20777_S79        3588
## UHM375.20778_S91           UHM375.20778_S91        7304
## UHM377.20779_S8             UHM377.20779_S8        7549
## UHM38.3376_S36               UHM38.3376_S36       29112
## UHM386.20781_S32           UHM386.20781_S32        6867
## UHM387.20782_S44           UHM387.20782_S44        2149
## UHM414.20583_S55           UHM414.20583_S55       12802
## UHM418.20765_S30           UHM418.20765_S30        6818
## UHM422.20766_S42           UHM422.20766_S42        6452
## UHM425.20767_S54           UHM425.20767_S54        4230
## UHM426.20534_S37           UHM426.20534_S37       13508
## UHM428.20544_S62           UHM428.20544_S62        3325
## UHM429.20559_S52           UHM429.20559_S52       14735
## UHM435.20547_S3             UHM435.20547_S3       12442
## UHM437.20768_S66           UHM437.20768_S66        3931
## UHM439.20564_S17           UHM439.20564_S17        3706
## UHM44.3526_S31               UHM44.3526_S31        3822
## UHM445.20569_S77           UHM445.20569_S77        1267
## UHM447.20783_S56           UHM447.20783_S56        5833
## UHM448.20769_S78           UHM448.20769_S78        6456
## UHM45.3539_S92               UHM45.3539_S92       35221
## UHM454.20770_S90           UHM454.20770_S90        6567
## UHM455.20785_S80           UHM455.20785_S80        6400
## UHM458.20786_S92           UHM458.20786_S92        3554
## UHM459.20787_S9             UHM459.20787_S9        6949
## UHM461.20771_S7             UHM461.20771_S7       10174
## UHM467.20772_S19           UHM467.20772_S19        4970
## UHM470.20533_S25           UHM470.20533_S25        6080
## UHM476.20414_S46           UHM476.20414_S46       12376
## UHM478.20549_S27           UHM478.20549_S27        6455
## UHM479.20551_S51           UHM479.20551_S51       16542
## UHM481.20403_S9             UHM481.20403_S9        1938
## UHM482.20590_S44           UHM482.20590_S44         254
## UHM483.20603_S10           UHM483.20603_S10        4453
## UHM519.20582_S43           UHM519.20582_S43       16240
## UHM520.20573_S30           UHM520.20573_S30        9366
## UHM746.21478_S117         UHM746.21478_S117       85545
## UHM747.21477_S106         UHM747.21477_S106       82434
## UHM748.21467_S170         UHM748.21467_S170       55338
## UHM748.21487_S129         UHM748.21487_S129       36665
## UHM749.21479_S128         UHM749.21479_S128       62924
```

```
## UHM759.21466_S159          UHM759.21466_S159       76912
## UHM759.21486_S118          UHM759.21486_S118       16768
## UHM775.21485_S107          UHM775.21485_S107       43137
## UHM776.21482_S161          UHM776.21482_S161       58259
## UHM777.21484_S183          UHM777.21484_S183       23552
## UHM779.21468_S181          UHM779.21468_S181       81397
## UHM779.21488_S140          UHM779.21488_S140           9
## UHM782.21480_S139          UHM782.21480_S139       38391
## UHM810.21472_S138          UHM810.21472_S138       94869
## UHM811.21471_S127          UHM811.21471_S127       30255
## UHM813.21481_S150          UHM813.21481_S150       65073
## UHM818.21469_S105          UHM818.21469_S105       50515
## UHM818.21489_S151          UHM818.21489_S151          30
## UHM819.21473_S149          UHM819.21473_S149       83507
## UHM820.21470_S116          UHM820.21470_S116       66154
## UHM820.21490_S162          UHM820.21490_S162         518
## UHM827.21474_S160          UHM827.21474_S160       84834
## UHM829.21476_S182          UHM829.21476_S182       69806
## UHM832.21483_S172          UHM832.21483_S172       76349
## UHM836.20385_S78            UHM836.20385_S78        3026
## UHM837.20386_S90            UHM837.20386_S90        2442
## UHM838.20387_S7              UHM838.20387_S7        4330
## UHM891.20384_S66            UHM891.20384_S66        3238
## UHM892.20532_S13            UHM892.20532_S13        7052
## UHM893.20595_S9              UHM893.20595_S9        9991
## UHM894.20540_S14            UHM894.20540_S14        5678
## UHM895.20536_S61            UHM895.20536_S61        6765
## UHM896.20601_S81            UHM896.20601_S81       10057
## UHM897.20591_S56            UHM897.20591_S56       13724
## UHM898.20394_S91            UHM898.20394_S91         603
## UHM899.20588_S20            UHM899.20588_S20       10593
## UHM900.20395_S8              UHM900.20395_S8        6596
## UHM901.20542_S38            UHM901.20542_S38       12693
## UHM902.20584_S67            UHM902.20584_S67        4178
## UHM903.20587_S8              UHM903.20587_S8        4707
## UHM904.20567_S53            UHM904.20567_S53       16165
## UHM905.20598_S45            UHM905.20598_S45        3207
## UHM906.20565_S29            UHM906.20565_S29       11360
## UHM907.20592_S68            UHM907.20592_S68        6155
## UHM908.20396_S20            UHM908.20396_S20        8212
## UHM909.20557_S28            UHM909.20557_S28        2759
## UHM910.20562_S88            UHM910.20562_S88        6385
## UHM965.20537_S73            UHM965.20537_S73        1777
## UHM966.20743_S51            UHM966.20743_S51        7100
## UHM967.20744_S63            UHM967.20744_S63        4222
## UHM968.20571_S6              UHM968.20571_S6       33909
## UHM969.20745_S75            UHM969.20745_S75        6115
## UHM971.20746_S87            UHM971.20746_S87        6911
```

```
## UHM973.20578_S90            UHM973.20578_S90        2680
## UHM974.20432_S72            UHM974.20432_S72        1133
## UHM975.20747_S4              UHM975.20747_S4        6185
## UHM977.20748_S16            UHM977.20748_S16        7885
## UHM978.20749_S28            UHM978.20749_S28        6336
## UHM979.20750_S40            UHM979.20750_S40        5449
## UHM980.20731_S2              UHM980.20731_S2        3823
## UHM981.20539_S2              UHM981.20539_S2       16669
## UHM982.20740_S15            UHM982.20740_S15        5740
## UHM983.20556_S16            UHM983.20556_S16       12838
## UHM984.20751_S52            UHM984.20751_S52        4646
## UHM985.20752_S64            UHM985.20752_S64        2573
## UHM988.20753_S76            UHM988.20753_S76        7752
## UHM989.20754_S88            UHM989.20754_S88        5843
## UHM991.20755_S5              UHM991.20755_S5        8388
## UHM993.20741_S27            UHM993.20741_S27        6685
## UHM996.20610_S94            UHM996.20610_S94       14720
## UHM997.20553_S75            UHM997.20553_S75       12274
## UHM998.20618_S95            UHM998.20618_S95       16875
## UHM999.20617_S83            UHM999.20617_S83        7227
```

```r
scaling_factors <- result$scaling_factors
head(scaling_factors) # Vector of scaling factors per sample
```

```
## spiked.blank.20433_S84 spiked.blank.20817_S84       Std2uL.20625_S84   StdSwab1uL.20624_S72
##           230.87500000             0.03924277             0.02958371             0.05235558
```

Convert relative counts to absolute counts

```r
#             Convert relative counts to absolute counts


# absolute counts=relative counts×sample-specific scaling factor

# Convert to absolute counts
library(DspikeIn)
absolute <- convert_to_absolute_counts(Spiked_16S_sum_scaled, scaling_factors)

# Extract processed data
absolute_counts <- absolute$absolute_counts
tse_absolute <- absolute$obj_adj

tse_absolute <- tidy_phyloseq_tse(tse_absolute)
```

```r
# View absolute count data
head(tse_absolute)
```

```
## class: TreeSummarizedExperiment
## dim: 6 264
## metadata(0):
## assays(1): counts
## rownames(6): 020e00d90ba97c5898944ab6f7b1b7c9 b00466354053c9065c8aa3d6fbb33eaa ... ed285eb1a
##   63f5509575600a9e7afb6847d6296976
## rowData names(7): Kingdom Phylum ... Genus Species
## colnames(264): spiked.blank.20433_S84 spiked.blank.20817_S84 ... UHM998.20618_S95 UHM999.206
## colData names(34): sample.id X16S.biosample ... swab.presence MK.spike

## reducedDimNames(0):
## mainExpName: NULL

## altExpNames(0):
## rowLinks: a LinkDataFrame (6 rows)
## rowTree: 1 phylo tree(s) (9227 leaves)
## colLinks: NULL
## colTree: NULL
```

Summary Stat

```r
#                       CALCULATE SPIKE PERCENTAGE & summary stat

# ** Calculate spike percentage & Generate summary statistics for absolute counts**

# Generate summary statistics for absolute counts
post_eval_summary <- calculate_summary_stats_table(absolute_counts)

# You may want to Back normal to check calculation accuracy
# the scaling factor was computed based on spiked species reads and fixed cell count.
# Multiplying the spiked species read count by this scaling factor restores the exact spiked c
# lets check it
# BackNormal <- calculate_spike_percentage(
#  tse_absolute,
#  merged_spiked_species,
#  passed_range = c(0.1, 20)
# )
```

spiked species retrieval is system-dependent

The goal is to identify the range where, for example, the evenness of your community remains independent of spiked species retrieval—meaning the p-value should not be significant, and the R^2 value should be low, indicating minimal influence. Hill number is interpretable as "effective diversity" (number of abundant species).

```
# The acceptable range of spiked species retrieval is system-dependent
# Spiked species become centroid of the community (Distance to Centroid)
# Spiked species become dominant and imbalance the community (Evenness)

# What range of spiked species retrieval is appropriate for your system?
# Calculate Pielou's Evenness using Shannon index and species richness (Observed)
# Hill number q = 1 = exp(Shannon index), representing the effective number of equally abundan
# Unlike Pielou's evenness, this metric is not normalized by richness and it shows Effective n

library(TreeSummarizedExperiment)
library(SummarizedExperiment)
library(dplyr)
library(tibble)
library(microbiome)
library(mia)
library(vegan)
library(S4Vectors)
library(ggplot2)


# --- 1. Extract current metadata from TSE ---
metadata <- colData(tse_absolute) %>%
  as.data.frame() %>%
  rownames_to_column("Sample")

# --- 2. Add spike-in reads (Perc) ---
# Ensure Perc has 'Sample' column and matching format
metadata <- dplyr::left_join(metadata, Perc, by = "Sample")

# --- 3. Estimate alpha diversity indices and extract ---
tse_absolute <- mia::addAlpha(
  tse_absolute,
  index = c("observed", "shannon", "pielou", "hill")
)

alpha_df <- colData(tse_absolute) %>%
  as.data.frame() %>%
  rownames_to_column("Sample") %>%
  select(Sample, observed, shannon, pielou, hill)
```

```r
metadata <- dplyr::left_join(metadata, alpha_df, by = "Sample")

# --- 4. Compute Bray-Curtis distance to centroid ---
otu_mat <- assay(tse_absolute, "counts")
otu_mat <- t(otu_mat)  # samples as rows
otu_mat_rel <- vegan::decostand(otu_mat, method = "total")

centroid_profile <- colMeans(otu_mat_rel)

dist_to_centroid <- apply(otu_mat_rel, 1, function(x) {
  vegan::vegdist(rbind(x, centroid_profile), method = "bray")[1]
})

# Match and assign distances
metadata$Dist_to_Centroid <- dist_to_centroid[metadata$Sample]

# --- 5. Assign updated metadata to TSE ---
metadata <- column_to_rownames(metadata, var = "Sample")
metadata <- metadata[colnames(tse_absolute), , drop = FALSE]  # ensure correct order and size

colData(tse_absolute) <- S4Vectors::DataFrame(metadata)


# 4. Regression Plots: Diversity vs. Spike-in Reads
# ==============================================================================

# Pielou's Evenness
plot_object_pielou <- regression_plot(
  data = metadata,
  x_var = "pielou",
  y_var = "Spiked_Reads",
  custom_range = c(0.1, 20, 30, 40, 50, 60, 100),
  plot_title = "Pielou's Evenness vs. Spike-in Reads"
)

# Hill Number (q = 1)
plot_object_hill <- regression_plot(
  data = metadata,
  x_var = "hill",
  y_var = "Spiked_Reads",
  custom_range = c(0.1, 10, 20, 30, 100),
  plot_title = "Hill Number (q = 1) vs. Spike-in Reads"
)

# Distance to Centroid
plot_object_centroid <- regression_plot(
  data = metadata,
```

```
  x_var = "Dist_to_Centroid",
  y_var = "Spiked_Reads",
  custom_range = c(0.1, 20, 30, 40, 50, 60, 100),
  plot_title = "Distance to Global Centroid vs. Spike-in Reads"
)


# Interpretation
# ================================================================================
# - Pielou's evenness is normalized by richness; useful for detecting imbalance.
# - Hill number q = 1 gives effective number of common species; sensitive to dominance.
# - Distance to centroid in full Bray-Curtis space shows deviation from the average community.
```

Calculate the percentage of spiked species retrieval

```
# * Calculate the percentage of spiked species retrieval per sample*

library(mia)
library(dplyr)
library(SummarizedExperiment)

# Subset TSE to remove blanks
tse_absolute_filtered <- tse_absolute[, colData(tse_absolute)$sample.or.blank != "blank"]

# Calculate spike-in retrieval percentage
result_perc <- calculate_spike_percentage(
  tse_absolute_filtered,
  merged_spiked_species,
  passed_range = c(0.1, 20)
)

# Generate conclusion report
conc <- conclusion(
  tse_absolute_filtered,
  merged_spiked_species,
  max_passed_range = 20,
  output_path = output_path)

conc$full_report
```

```
##                Sample Total_Reads Spiked_Reads   Percentage Result
## 1   STP1719.20422_S47        2427         1847  76.10218377 failed
## 2    STP213.20423_S59      188314         1847   0.98080865 passed
## 3    STP268.20424_S71      757053         1847   0.24397235 passed
```

```
## 4    STP544.20419_S11        1913         1847  96.54992159 failed
## 5    STP570.20420_S23        5948         1847  31.05245461 failed
## 6    STP579.20421_S35        5452         1847  33.87747616 failed
## 7    STP614.20418_S94           5            0   0.00000000 failed
## 8   UHM1000.20604_S22       43347          924   2.13163541 passed
## 9   UHM1001.20609_S82       39309          924   2.35060673 passed
## 10  UHM1007.20622_S48       86418          924   1.06922169 passed
## 11  UHM1009.20614_S47      181742          924   0.50841303 passed
## 12  UHM1010.20621_S36        6123          924  15.09064184 passed
## 13  UHM1011.20606_S46       81017          924   1.14050138 passed
## 14  UHM1024.20620_S24        2690          924  34.34944238 failed
## 15  UHM1026.20607_S58        7494          924  12.32986389 passed
## 16  UHM1028.20613_S35        2620          924  35.26717557 failed
## 17  UHM1032.20605_S34        1410          924  65.53191489 failed
## 18  UHM1033.20619_S12         966          924  95.65217391 failed
## 19  UHM1034.20616_S71      782439          924   0.11809227 passed
## 20  UHM1035.20611_S11        8082          924  11.43281366 passed
## 21  UHM1036.20612_S23        8625          924  10.71304348 passed
## 22  UHM1052.20615_S59       98557          924   0.93752854 passed
## 23   UHM1060.20723_S1        2062         1847  89.57322987 failed
## 24  UHM1065.20724_S13        2384         1847  77.47483221 failed
## 25  UHM1068.20732_S14       19198         1847   9.62079383 passed
## 26  UHM1069.20742_S39       88462         1847   2.08790215 passed
## 27  UHM1070.20725_S25       11119         1847  16.61120604 passed
## 28  UHM1071.20733_S26       66892         1847   2.76116725 passed
## 29  UHM1072.20734_S38        4254         1847  43.41795957 failed
## 30  UHM1073.20735_S50      129333         1847   1.42809646 passed
## 31  UHM1075.20726_S37       12116         1847  15.24430505 passed
## 32  UHM1077.20736_S62      240090         1847   0.76929485 passed
## 33  UHM1078.20727_S49       47549         1847   3.88441397 passed
## 34  UHM1080.20737_S74      237852         1847   0.77653331 passed
## 35  UHM1081.20728_S61        2118         1847  87.20491029 failed
## 36  UHM1088.20738_S86       48261         1847   3.82710677 passed
## 37   UHM1090.20739_S3        4260         1847  43.35680751 failed
## 38  UHM1093.20729_S73       11040         1847  16.73007246 passed
## 39  UHM1095.20730_S85        3475         1847  53.15107914 failed
## 40  UHM1097.20623_S60        1639          924  56.37583893 failed
## 41  UHM1099.20608_S70      141219          924   0.65430289 passed
## 42  UHM1100.20788_S21      120156         1847   1.53716835 passed
## 43  UHM1102.20789_S33       41207         1847   4.48224816 passed
## 44  UHM1104.20790_S45       73225         1847   2.52236258 passed
## 45  UHM1105.20791_S57       29445         1847   6.27271184 passed
## 46   UHM1109.20531_S1       49710         1847   3.71555019 passed
## 47  UHM1110.20568_S65      257312         1847   0.71780562 passed
## 48  UHM1113.20792_S69      374668         1847   0.49296978 passed
## 49  UHM1114.20793_S81       25164         1847   7.33985058 passed
## 50  UHM1115.20794_S93       53637         1847   3.44351847 passed
## 51  UHM1117.20795_S10      120412         1847   1.53390028 passed
```

```
## 52  UHM1118.20796_S22       39826         1847   4.63767388 passed
## 53  UHM1120.20797_S34        6447         1847  28.64898402 failed
## 54  UHM1124.20798_S46       19768         1847   9.34338325 passed
## 55  UHM1126.20799_S58       55979         1847   3.29945158 passed
## 56  UHM1128.20800_S70       26179         1847   7.05527331 passed
## 57   UHM1140.20555_S4      144876         1847   1.27488335 passed
## 58  UHM1145.20801_S82      136223         1847   1.35586502 passed
## 59  UHM1163.20405_S33       66525         1847   2.77639985 passed
## 60  UHM1164.20402_S92      560873         1847   0.32930806 passed
## 61  UHM1169.20552_S63        2087         1847  88.50023958 failed
## 62   UHM1171.20579_S7       94275         1847   1.95916203 passed
## 63  UHM1176.20404_S21      153911         1847   1.20004418 passed
## 64  UHM1177.20546_S86        7001         1847  26.38194544 failed
## 65  UHM1182.20576_S66       19322         1847   9.55905186 passed
## 66  UHM1210.20802_S94      107631         1847   1.71604835 passed
## 67  UHM1212.20803_S11       34861         1847   5.29818422 passed
## 68  UHM1217.20804_S23       63472         1847   2.90994454 passed
## 69  UHM1218.20805_S35       42728         1847   4.32269238 passed
## 70  UHM1219.20806_S47       73751         1847   2.50437282 passed
## 71  UHM1220.20807_S59       69371         1847   2.66249586 passed
## 72  UHM1221.20808_S71       37595         1847   4.91288735 passed
## 73  UHM1222.20809_S83        5442         1847  33.93972804 failed
## 74  UHM1223.20810_S95      114277         1847   1.61624824 passed
## 75  UHM1225.20811_S12       48275         1847   3.82599689 passed
## 76  UHM1227.20812_S24       20802         1847   8.87895395 passed
## 77  UHM1228.20813_S36      147297         1847   1.25392914 passed
## 78  UHM1237.20814_S48       24999         1847   7.38829553 passed
## 79  UHM1240.20566_S41       35116         1847   5.25971067 passed
## 80  UHM1246.20815_S60       16128         1847  11.45213294 passed
## 81  UHM1247.20816_S72       26551         1847   6.95642349 passed
## 82  UHM1248.20575_S54        7646         1847  24.15642166 failed
## 83  UHM1256.20570_S89       35459         1847   5.20883274 passed
## 84  UHM1260.20596_S21       34301         1847   5.38468266 passed
## 85  UHM1270.20577_S78       12149         1847  15.20289736 passed
## 86  UHM1271.20397_S32       19107         1847   9.66661433 passed
## 87  UHM1272.20398_S44       10447         1847  17.67971667 passed
## 88  UHM1274.20554_S87       32434         1847   5.69464143 passed
## 89  UHM1275.20597_S33       15061         1847  12.26346192 passed
## 90  UHM1282.20599_S57        6536         1847  28.25887393 failed
## 91  UHM1287.20543_S50       10735         1847  17.20540289 passed
## 92  UHM1291.20416_S70       74189         1847   2.48958741 passed
## 93  UHM1296.20550_S39      207620         1847   0.88960601 passed
## 94  UHM1319.20561_S76       11611         1847  15.90732926 passed
## 95  UHM1324.20413_S34       45204         1847   4.08592160 passed
## 96  UHM1327.20545_S74      272806         1847   0.67703790 passed
## 97  UHM1328.20572_S18        9198         1847  20.08045227 failed
## 98  UHM1334.20417_S82       76515         1847   2.41390577 passed
## 99  UHM1338.20399_S56       26305         1847   7.02147881 passed
```

```
## 100 UHM1341.20602_S93        4829        1847  38.24808449 failed
## 101 UHM1356.20541_S26       21082        1847   8.76102837 passed
## 102 UHM1380.20580_S19       71651        1847   2.57777282 passed
## 103 UHM1383.20594_S92       28316        1847   6.52281396 passed
## 104  UHM1385.20563_S5        2166        1847  85.27239151 failed
## 105 UHM1399.20756_S17       11379        1847  16.23165480 passed
## 106 UHM1400.20757_S29       49300        1847   3.74645030 passed
## 107 UHM1401.20758_S41        4984        1847  37.05858748 failed
## 108 UHM1402.20759_S53       78898        1847   2.34099724 passed
## 109 UHM1403.20760_S65       47369        1847   3.89917457 passed
## 110 UHM1405.20761_S77       49948        1847   3.69784576 passed
## 111 UHM1406.20762_S89       86410        1847   2.13748409 passed
## 112  UHM1414.20763_S6      113476        1847   1.62765695 passed
## 113 UHM1419.20764_S18       22182        1847   8.32657109 passed
## 114 UHM1427.20389_S31       93753        1847   1.97007029 passed
## 115 UHM1428.20390_S43        4532           0   0.00000000 failed
## 116 UHM1429.20391_S55      245778        1847   0.75149118 passed
## 117 UHM1430.20392_S67      846266        1847   0.21825289 passed
## 118 UHM1432.20393_S79      415264        1847   0.44477730 passed
## 119 UHM1435.20388_S19        9292           0   0.00000000 failed
## 120  UHM162.20560_S64       37088        1847   4.98004745 passed
## 121  UHM198.20585_S79        5610        1847  32.92335116 failed
## 122    UHM20.3314_S52     1335639        1847   0.13828587 passed
## 123    UHM20.3315_S64      352109        1847   0.52455348 passed
## 124  UHM204.20409_S81      160159        1847   1.15322898 passed
## 125  UHM206.20410_S93         523           0   0.00000000 failed
## 126  UHM207.20593_S80       76828        1847   2.40407143 passed
## 127  UHM208.20411_S10       34089        1847   5.41817008 passed
## 128  UHM211.20406_S45       87029        1847   2.12228108 passed
## 129  UHM215.20408_S69     3947656        1847   0.04678726 failed
## 130  UHM216.20429_S36      207601        1847   0.88968743 passed
## 131  UHM219.20430_S48        1897        1847  97.36425936 failed
## 132  UHM236.20431_S60      199471        1847   0.92594914 passed
## 133  UHM238.20407_S57       64050        1847   2.88368462 passed
## 134  UHM245.20538_S85      554868        1847   0.33287196 passed
## 135  UHM252.20558_S40       14250        1847  12.96140351 passed
## 136  UHM267.20400_S68      177722        1847   1.03926357 passed
## 137  UHM274.20581_S31       13695        1847  13.48667397 passed
## 138  UHM276.20586_S91       21560        1847   8.56679035 passed
## 139  UHM280.20401_S80       35333        1847   5.22740781 passed
## 140  UHM286.20425_S83      130953        1847   1.41042970 passed
## 141  UHM289.20426_S95        6003        1847  30.76794936 failed
## 142  UHM294.20427_S12       17177        1847  10.75275077 passed
## 143  UHM298.20600_S69      101173        1847   1.82558588 passed
## 144  UHM325.20548_S15       24562        1847   7.51974595 passed
## 145  UHM337.20412_S22       32632        1847   5.66008826 passed
## 146  UHM354.20535_S49       58805        1847   3.14088938 passed
## 147  UHM356.20415_S58       43688        1847   4.22770555 passed
```

```
## 148  UHM369.20773_S31       79551         1847   2.32178100 passed
## 149  UHM370.20774_S43       88455         1847   2.08806738 passed
## 150  UHM372.20775_S55       21119         1847   8.74567925 passed
## 151  UHM373.20776_S67      113026         1847   1.63413728 passed
## 152  UHM374.20777_S79       96042         1847   1.92311697 passed
## 153  UHM375.20778_S91       18842         1847   9.80256873 passed
## 154   UHM377.20779_S8       29234         1847   6.31798591 passed
## 155    UHM38.3376_S36       29112            0   0.00000000 failed
## 156  UHM386.20781_S32       54216         1847   3.40674340 passed
## 157  UHM387.20782_S44       68442         1847   2.69863534 passed
## 158  UHM414.20583_S55      278181         1847   0.66395620 passed
## 159  UHM418.20765_S30       46656         1847   3.95876200 passed
## 160  UHM422.20766_S42       10385         1847  17.78526721 passed
## 161  UHM425.20767_S54       66208         1847   2.78969309 passed
## 162  UHM426.20534_S37        5428         1847  34.02726603 failed
## 163  UHM428.20544_S62       13527         1847  13.65417314 passed
## 164  UHM429.20559_S52       21213         1847   8.70692500 passed
## 165   UHM435.20547_S3       13365         1847  13.81967826 passed
## 166  UHM437.20768_S66      148167         1847   1.24656637 passed
## 167  UHM439.20564_S17       96401         1847   1.91595523 passed
## 168    UHM44.3526_S31        5511         1847  33.51478860 failed
## 169  UHM445.20569_S77       90006         1847   2.05208542 passed
## 170  UHM447.20783_S56      168350         1847   1.09711910 passed
## 171  UHM448.20769_S78       28120         1847   6.56827881 passed
## 172    UHM45.3539_S92       35221            0   0.00000000 failed
## 173  UHM454.20770_S90       43632         1847   4.23313165 passed
## 174  UHM455.20785_S80       14926         1847  12.37438028 passed
## 175  UHM458.20786_S92       95135         1847   1.94145162 passed
## 176   UHM459.20787_S9      106958         1847   1.72684605 passed
## 177   UHM461.20771_S7       77653         1847   2.37853013 passed
## 178  UHM467.20772_S19      195315         1847   0.94565190 passed
## 179  UHM470.20533_S25        9207         1847  20.06082329 failed
## 180  UHM476.20414_S46       20085         1847   9.19591735 passed
## 181  UHM478.20549_S27       27792         1847   6.64579735 passed
## 182  UHM479.20551_S51        2116         1847  87.28733459 failed
## 183   UHM481.20403_S9       15762         1847  11.71805608 passed
## 184  UHM482.20590_S44        2014         1847  91.70804369 failed
## 185  UHM483.20603_S10       90381         1847   2.04357110 passed
## 186  UHM519.20582_S43      220555         1847   0.83743284 passed
## 187  UHM520.20573_S30       68381         1847   2.70104269 passed
## 188 UHM746.21478_S117         927          924  99.67637540 failed
## 189 UHM747.21477_S106         924          924 100.00000000 failed
## 190 UHM748.21467_S170         924          924 100.00000000 failed
## 191 UHM748.21487_S129         928          924  99.56896552 failed
## 192 UHM749.21479_S128         925          924  99.89189189 failed
## 193 UHM759.21466_S159         923          923 100.00000000 failed
## 194 UHM759.21486_S118         938          924  98.50746269 failed
## 195 UHM775.21485_S107         923          923 100.00000000 failed
```

```
## 196 UHM776.21482_S161         923          923 100.00000000 failed
## 197 UHM777.21484_S183         925          924  99.89189189 failed
## 198 UHM779.21468_S181         928          924  99.56896552 failed
## 199 UHM779.21488_S140         924          924 100.00000000 failed
## 200 UHM782.21480_S139         924          924 100.00000000 failed
##  [ reached 'max' / getOption("max.print") -- omitted 60 rows ]
```

```
# Filter to keep only the samples that passed
passed_samples <- result_perc$Sample[result_perc$Result == "passed"]

# Subset the TSE object to include only passed samples
tse_passed <- tse_absolute_filtered[, colnames(tse_absolute_filtered) %in% passed_samples]
dim(tse_passed)
```

```
## [1] 8742  177
```

Abundance-based Core microbiome

```
tse_absolute <- absolute$obj_adj
pps_Abs <- DspikeIn::get_long_format_data(tse_absolute)

# calculation for relative abundance needs sum of total reads
# total_reads <- sum(pps_Abs$Abundance)

# Generate an alluvial plot

alluvial_plot_abs <- alluvial_plot(
  data = pps_Abs,
  axes = c("Host.genus", "Ecoregion.III","Diet", "Animal.ecomode"),
  abundance_threshold = 10000,
  fill_variable = "Class",
  silent = TRUE,
  abundance_type = "absolute",
  top_taxa = 15,
  text_size = 4,
  legend_ncol = 1,
  custom_colors = DspikeIn::color_palette$light_MG # Use the color palette from DspikeIn
)
```

Data transform & Normalization

you may select to transform your data befor moving forward with Differential Abundance

```r
# you may need to normalize/transform your data to reduce biases

ps <- tse_16SOTU

# TC Normalization
result_TC <- normalization_set(ps, method = "TC", groups = "Host.species")
normalized_ps_TC <- result_TC$dat.normed
scaling_factors_TC <- result_TC$scaling.factor

# UQ Normalization
result_UQ <- normalization_set(ps, method = "UQ", groups = "Host.species")
normalized_ps_UQ <- result_UQ$dat.normed
scaling_factors_UQ <- result_UQ$scaling.factor

# Median Normalization
result_med <- normalization_set(ps, method = "med", groups = "Host.species")
normalized_ps_med <- result_med$dat.normed
scaling_factors_med <- result_med$scaling.factor

# DESeq Normalization
ps_n <- remove_zero_negative_count_samples(ps)
result_DESeq <- normalization_set(ps_n, method = "DESeq", groups = "Animal.type")
normalized_ps_DESeq <- result_DESeq$dat.normed
scaling_factors_DESeq <- result_DESeq$scaling.factor

# Poisson Normalization
result_Poisson <- normalization_set(ps, method = "Poisson", groups = "Host.genus")
normalized_ps_Poisson <- result_Poisson$dat.normed
scaling_factors_Poisson <- result_Poisson$scaling.factor

# Quantile Normalization
result_QN <- normalization_set(ps, method = "QN")
normalized_ps_QN <- result_QN$dat.normed
scaling_factors_QN <- result_QN$scaling.factor

# TMM Normalization
result_TMM <- normalization_set(ps, method = "TMM", groups = "Animal.type")
normalized_ps_TMM <- result_TMM$dat.normed
scaling_factors_TMM <- result_TMM$scaling.factor

# CLR Normalization
result_clr <- normalization_set(ps, method = "clr")
normalized_ps_clr <- result_clr$dat.normed
scaling_factors_clr <- result_clr$scaling.factor

# Rarefying
result_rar <- normalization_set(ps, method = "rar")
```

```
normalized_ps_rar <- result_rar$dat.normed
scaling_factors_rar <- result_rar$scaling.factor

# CSS Normalization
result_css <- normalization_set(ps, method = "css")
normalized_ps_css <- result_css$dat.normed
scaling_factors_css <- result_css$scaling.factor

# TSS Normalization
result_tss <- normalization_set(ps, method = "tss")
normalized_ps_tss <- result_tss$dat.normed
scaling_factors_tss <- result_tss$scaling.factor

# RLE Normalization
result_rle <- normalization_set(ps, method = "rle")
normalized_ps_rle <- result_rle$dat.normed
scaling_factors_rle <- result_rle$scaling.factor
```

Ridge plot

```
rf_tse <- RandomForest_selected(
  tse_16SOTU,
  response_var = "Host.genus",
  na_vars = c("Habitat", "Ecoregion.III", "Host.genus", "Diet")
)

ridge_tse<- ridge_plot_it(rf_tse, taxrank = "Family", top_n = 10) + scale_fill_manual(values =
# ridge_physeq

result_css <- normalization_set(rf_tse, method = "css")
normalized_ps_css <- result_css$dat.normed

ridge_tse <- ridge_plot_it(normalized_ps_css, taxrank = "Family", top_n = 10) + scale_fill_manu

# ridge_physeq
```

Further analysis

remove the spike-in sp before further analysis

```
library(mia)
library(dplyr)
library(SummarizedExperiment)

# ---Remove unwanted taxa by Genus, Family, or Order ---
```

```r
tse_filtered <- tse_absolute[
  rowData(tse_absolute)$Genus != "Tetragenococcus" &
  rowData(tse_absolute)$Family != "Chloroplast" &
  rowData(tse_absolute)$Order != "Chloroplast", ]

tse_caudate <- tse_filtered[, colData(tse_filtered)$Clade.Order == "Caudate"]
genera_keep <- c("Desmognathus", "Plethodon", "Eurycea")
tse_three_genera <- tse_caudate[, colData(tse_caudate)$Host.genus %in% genera_keep]
tse_blue_ridge <- tse_three_genera[, colData(tse_three_genera)$Ecoregion.III == "Blue Ridge"]
tse_desmog <- tse_blue_ridge[, colData(tse_blue_ridge)$Host.genus == "Desmognathus"]

# --- Differential Abundance with DESeq2 ---
results_DESeq2 <- perform_and_visualize_DA(
  obj = tse_desmog,
  method = "DESeq2",
  group_var = "Host.taxon",
  contrast = c("Desmognathus monticola", "Desmognathus imitator"),
  output_csv_path = "DA_DESeq2.csv",
  target_glom = "Genus",
  significance_level = 0.05)


head(results_DESeq2$results)
```

```
##    baseMean         logFC     lfcSE          stat    pvalue padj FDR    Significance
## 1 10.000000  9.125533e-08 0.2416083  3.776995e-07 0.9999997    1   1 Not Significant  Desmog
## 2  2.000000  1.338986e-07 0.5334178  2.510202e-07 0.9999998    1   1 Not Significant  Desmog
## 3  1.357143 -1.058919e-06 0.7500333 -1.411829e-06 0.9999989    1   1 Not Significant Desmogr
## 4  2.000000  1.338986e-07 0.5334178  2.510202e-07 0.9999998    1   1 Not Significant  Desmog
## 5 12.000000  8.273936e-08 0.2214583  3.736114e-07 0.9999997    1   1 Not Significant  Desmog
## 6 13.809524 -3.680398e-01 0.2448251 -1.503276e+00 0.1327678    1   1 Not Significant Desmogr
##                                OTU  Kingdom          Phylum              Class
## 1 b00466354053c9065c8aa3d6fbb33eaa Bacteria  Armatimonadota          uncultured         uncu
## 2 f872c4bf84bcf44434fa2023788f6517 Bacteria  Proteobacteria Gammaproteobacteria       Chroma
## 3 df13f71584d4a579c81d909eaba11a74 Bacteria      Firmicutes    Syntrophomonadia Syntrophomor
## 4 ed285eb1aac505a1f062b482300b69f7 Bacteria      Firmicutes     Symbiobacteriia  Symbiobacte
## 5 63f5509575600a9e7afb6847d6296976 Bacteria Gemmatimonadota    Gemmatimonadetes   Gemmatimor
## 6 fea92298310d6159915036da73e7a88a Bacteria Gemmatimonadota    Gemmatimonadetes   Gemmatimor
##             Genus Species
## 1      uncultured    <NA>
## 2 Thiophaeococcus    <NA>
## 3  Syntrophomonas    <NA>
## 4      uncultured    <NA>
## 5      uncultured    <NA>
## 6    Gemmatimonas    <NA>
```

```
results_DESeq2$obj_significant
```

```
## class: TreeSummarizedExperiment
## dim: 27 42
## metadata(0):
## assays(1): counts
## rownames(27): ab89127678d37273440428628a6127bb bdade7c571ecaa00a901b7f3cfe29f69 ...
##   32edc6e65f17f593f98160672a4b2dda dc1b9fd38680bd0be4b9f3d1bd95d204
## rowData names(7): Kingdom Phylum ... Genus Species
## colnames(42): UHM1240.20566_S41 UHM1248.20575_S54 ... UHM470.20533_S25 UHM476.20414_S46
## colData names(34): sample.id X16S.biosample ... swab.presence MK.spike

## reducedDimNames(0):
## mainExpName: NULL

## altExpNames(0):
## rowLinks: a LinkDataFrame (27 rows)
## rowTree: 1 phylo tree(s) (27 leaves)
## colLinks: NULL
## colTree: NULL
```

```
# Optional: Visualization
# results_DESeq2$plot
# results_DESeq2$bar_plot
```

Relative abundance

```
# Relative abundance
#  Extract OTU matrices from TSE Relative counts
# ------------------------------------------

# ---Remove unwanted taxa by Genus, Family, or Order ---
tse_filtered <- tse_16SOTU[
  rowData(tse_16SOTU)$Genus != "Tetragenococcus" &
  rowData(tse_16SOTU)$Family != "Chloroplast" &
  rowData(tse_16SOTU)$Order != "Chloroplast", ]

tse_caudate <- tse_filtered[, colData(tse_filtered)$Clade.Order == "Caudate"]
genera_keep <- c("Desmognathus", "Plethodon", "Eurycea")
tse_three_genera <- tse_caudate[, colData(tse_caudate)$Host.genus %in% genera_keep]
tse_blue_ridge <- tse_three_genera[, colData(tse_three_genera)$Ecoregion.III == "Blue Ridge"]
tse_desmog_rel <- tse_blue_ridge[, colData(tse_blue_ridge)$Host.genus == "Desmognathus"]
```

```r
# --- Differential Abundance with DESeq2 ---
results_DESeq2_rel <- perform_and_visualize_DA(
  obj = tse_desmog_rel,
  method = "DESeq2",
  group_var = "Host.taxon",
  contrast = c("Desmognathus monticola", "Desmognathus imitator"),
  output_csv_path = "DA_DESeq2.csv",
  target_glom = "Genus",
  significance_level = 0.05)


head(results_DESeq2_rel$results)
```

```
##    baseMean         logFC     lfcSE          stat    pvalue padj FDR    Significance
## 1 10.000000  8.808868e-08 0.2373794  3.710881e-07 0.9999997    1   1 Not Significant  Desmog
## 2  2.000000  1.284288e-07 0.5224090  2.458396e-07 0.9999998    1   1 Not Significant  Desmog
## 3  1.047619 -4.024325e-07 0.7345416 -5.478689e-07 0.9999996    1   1 Not Significant Desmogr
## 4  2.000000  1.284288e-07 0.5224090  2.458396e-07 0.9999998    1   1 Not Significant  Desmog
## 5 12.000000  7.961841e-08 0.2172416  3.664971e-07 0.9999997    1   1 Not Significant  Desmog
## 6 13.023810 -1.227825e-02 0.2086006 -5.886010e-02 0.9530635    1   1 Not Significant Desmogr
##                                OTU  Kingdom          Phylum               Class
## 1 b00466354053c9065c8aa3d6fbb33eaa Bacteria  Armatimonadota          uncultured         uncu
## 2 f872c4bf84bcf44434fa2023788f6517 Bacteria  Proteobacteria Gammaproteobacteria       Chroma
## 3 df13f71584d4a579c81d909eaba11a74 Bacteria      Firmicutes    Syntrophomonadia Syntrophomor
## 4 ed285eb1aac505a1f062b482300b69f7 Bacteria      Firmicutes     Symbiobacteriia  Symbiobacte
## 5 63f5509575600a9e7afb6847d6296976 Bacteria Gemmatimonadota    Gemmatimonadetes   Gemmatimor
## 6 fea92298310d6159915036da73e7a88a Bacteria Gemmatimonadota    Gemmatimonadetes   Gemmatimor
##             Genus Species
## 1      uncultured    <NA>
## 2 Thiophaeococcus    <NA>
## 3  Syntrophomonas    <NA>
## 4      uncultured    <NA>
## 5      uncultured    <NA>
## 6    Gemmatimonas    <NA>
```

```r
results_DESeq2_rel$obj_significant
```

```
## class: TreeSummarizedExperiment
## dim: 4 42
## metadata(0):
## assays(1): counts
## rownames(4): f74a23038095f558d209d451c03c9b7e 6e105f3f79341d4c6ac024b928786898
##   24ded21a86db222daa97a8f03093350f fb93efa4d1a901a697231635e8e2f683
## rowData names(7): Kingdom Phylum ... Genus Species
## colnames(42): UHM1240.20566_S41 UHM1248.20575_S54 ... UHM470.20533_S25 UHM476.20414_S46
## colData names(34): sample.id X16S.biosample ... swab.presence MK.spike
```

```
## reducedDimNames(0):
## mainExpName: NULL

## altExpNames(0):
## rowLinks: a LinkDataFrame (4 rows)
## rowTree: 1 phylo tree(s) (4 leaves)
## colLinks: NULL
## colTree: NULL
## referenceSeq: a DNAStringSet (4 sequences)
```

```
# Optional: Visualization
# results_DESeq2$plot
# results_DESeq2$bar_plot
```

Turnover (Presence/absence analysis)

taxonomic detection consistency between RA and AA OTU tables Presence/absence analysis to detect concordance between AA and RA profiles

```
# ------------------------------------------
#  Extract OTU matrices from TSE objects
# ------------------------------------------

otu_rel <- assay(tse_desmog_rel, "counts")
otu_abs <- assay(tse_desmog, "counts")

# Make sure samples are rows and taxa are columns
otu_rel <- t(otu_rel)
otu_abs <- t(otu_abs)

# Ensure they are matrices
otu_rel <- as.matrix(otu_rel)
otu_abs <- as.matrix(otu_abs)

# ------------------------------------------
# 2. Convert to Presence/Absence
# ------------------------------------------
otu_rel_pa <- (otu_rel > 0) * 1
otu_abs_pa <- (otu_abs > 0) * 1

# ------------------------------------------
# 3. Identify common samples & taxa
# ------------------------------------------
shared_samples <- intersect(rownames(otu_rel_pa), rownames(otu_abs_pa))
shared_taxa <- intersect(colnames(otu_rel_pa), colnames(otu_abs_pa))
```

```r
# Subset to common set
otu_rel_pa <- otu_rel_pa[shared_samples, shared_taxa]
otu_abs_pa <- otu_abs_pa[shared_samples, shared_taxa]

# ------------------------------------------
# 4. Compare presence/absence profiles
# ------------------------------------------
shared_pa <- (otu_rel_pa + otu_abs_pa) == 2
total_present <- (otu_rel_pa + otu_abs_pa) >= 1

# Calculate percent agreement (global)
percent_shared <- sum(shared_pa) / sum(total_present)

cat("Percent of shared taxa detections (AA vs RA):",
    round(100 * percent_shared, 1), "%\n")
```

```
## Percent of shared taxa detections (AA vs RA): 99.6 %
```

Differential abundance (Single and Multilayer Pairwise)

```r
# DspikeIn: Differential Abundance Examples
# Using perform_and_visualize_DA() with multiple contrast scenarios

# 1. Single Contrast

res_single <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  group_var = "Diet",
  contrast = c("Insectivore", "Carnivore"),
  target_glom = "Genus")



# 2. Single Factor - All Pairwise Contrasts
col_df <- as.data.frame(colData(tse_16SOTU))
col_df$Host.taxon <- factor(make.names(as.character(col_df$Host.taxon)))
colData(tse_16SOTU)$Host.taxon <- col_df$Host.taxon

# Get unique DESeq2-compatible factor levels
host_levels <- levels(col_df$Host.taxon)
print(host_levels)

# contrast list
```

```r
contrast_named <- list(
  Host.taxon = combn(host_levels, 2, simplify = FALSE))

# multiple pairwise contrasts
res_multi <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  group_var = "Host.taxon",
  contrast = contrast_named,
  target_glom = "Genus")



# 3. Single Factor - Selected Contrasts

contrast_list <- list(
  c("Insectivore", "Carnivore"),
  c("Omnivore", "Herbivore"))

res_selected <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  group_var = "Diet",
  contrast = contrast_list,
  global_fdr = TRUE)



# 4. Multiple Factors - Selected Contrasts

contrast_named <- list(
  Diet = list(
    c("Insectivore", "Carnivore"),
    c("Omnivore", "Carnivore") ),
  Animal.type = list(
    c("Frog", "Salamander") ))

res_multi_factor <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  contrast = contrast_named,
  target_glom = "Genus",
  significance_level = 0.01,
  global_fdr = TRUE)
```

```r
# 5. Multiple Factors - all pairwise contrasts

colData(tse_16SOTU)$Host.taxon <- droplevels(factor(colData(tse_16SOTU)$Host.taxon))
colData(tse_16SOTU)$Habitat <- droplevels(factor(colData(tse_16SOTU)$Habitat))
host_levels <- levels(colData(tse_16SOTU)$Host.taxon)
habitat_levels <- levels(colData(tse_16SOTU)$Habitat)

# Define pairwise contrasts as a named list
contrast_named <- list(
  Host.taxon = combn(host_levels, 2, simplify = FALSE),
  Habitat = combn(habitat_levels, 2, simplify = FALSE))

# Define ComboGroup variable
colData(tse_16SOTU)$ComboGroup <- factor(interaction(
  colData(tse_16SOTU)$Animal.type,
  colData(tse_16SOTU)$Diet,
  drop = TRUE))



# --- Run multi-factor DESeq2 comparisons ---
results_multi_factor <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  contrast = contrast_named,
  target_glom = "Genus",
  significance_level = 0.01)

# --- Combined Group Interactions ---
colData(tse_16SOTU)$ComboGroup <- factor(interaction(
  colData(tse_16SOTU)$Animal.type,
  colData(tse_16SOTU)$Diet,
  drop = TRUE))

# --- Define custom interaction contrasts ---
contrast_list <- list(
  c("Salamander.Insectivore", "Lizard.Insectivore"),
  c("Salamander.Carnivore", "Snake.Carnivore"),
  c("Salamander.Carnivore", "Frog.Carnivore"))

# --- Run combined group comparisons ---
res_combo <- perform_and_visualize_DA(
  obj = tse_16SOTU,
  method = "DESeq2",
  group_var = "ComboGroup",
  contrast = contrast_list,
  target_glom = "Genus",
```

```
  global_fdr = TRUE)
```

Visualization

```
#   Visualization of community composition

# Subset rows where Genus is not Tetragenococcus and not NA
keep_taxa <- !is.na(rowData(tse_16SOTU)$Genus) & rowData(tse_16SOTU)$Genus != "Tetragenococcus"
# Subset the TSE by keeping only those taxa (rows)
tse_filtered <- tse_16SOTU[keep_taxa, ]
# Exclude blanks
tse_filtered <- tse_filtered[, colData(tse_filtered)$sample.or.blank != "blank"]
# subset Salamander samples
tse_sal <- tse_filtered[, colData(tse_filtered)$Animal.type == "Salamander"]
Prok_OTU_sal <- tidy_phyloseq_tse(tse_sal)

TB<-taxa_barplot(
  Prok_OTU_sal,
  target_glom = "Genus",
  custom_tax_names = NULL,
  normalize = TRUE,
  treatment_variable = "Habitat",
  abundance_type = "relative",
  x_angle = 25,
  fill_variable = "Family",
  facet_variable = "Diet",
  top_n_taxa = 20,
  palette = DspikeIn::color_palette$mix_MG,
  legend_size = 11,
  legend_columns = 1,
  x_scale = "free",
  xlab = NULL)
```

Microbial dynamics & NetWork comparision

```
# 1. Initialization and loading NetWorks for Comparision


# library(SpiecEasi)
# library(ggnet)
# library(igraph)
```

```r
library(DspikeIn)
library(tidyr)
library(dplyr)
library(ggpubr)
library(igraph)

# To create a microbial co-occurrence network, you can refer to the SpiecEasi package availabl
# SpiecEasi GitHub Repository  https://github.com/zdk123/SpiecEasi

# herp.Bas.rel.f is a merged phyloseq object for both bacterial and fungal domains
# spiec.easi(herp.Bas.rel.f, method='mb', lambda.min.ratio=1e-3, nlambda=250,pulsar.select=TRU

Complete <- load_graphml("Complete.graphml")
NoBasid <- load_graphml("NoBasid.graphml")
NoHubs <- load_graphml("NoHubs.graphml")

# result <- weight_Network(graph_path = "Complete.graphml")
# result

result_kk <- degree_network(
  graph_path = load_graphml("Complete.graphml"),
  save_metrics = TRUE,
  layout_type = "stress"
)
# print(result_kk$plot)
```

Network Topological Metrics

```r
# 2. Metrics Calculation


result_Complete <- node_level_metrics(Complete)
result_NoHubs <- node_level_metrics(NoHubs)
result_NoBasid <- node_level_metrics(NoBasid)

Complete_metrics <- result_Complete$metrics
Nohub_metrics <- result_NoHubs$metrics
Nobasid_metrics <- result_NoBasid$metrics

Complete_metrics <- data.frame(lapply(Complete_metrics, as.character), stringsAsFactors = FALSE
Nohub_metrics <- data.frame(lapply(Nohub_metrics, as.character), stringsAsFactors = FALSE)
Nobasid_metrics <- data.frame(lapply(Nobasid_metrics, as.character), stringsAsFactors = FALSE)
```

```r
print(igraph::vcount(Complete)) # Number of nodes
```

```
## [1] 308
```

```r
print(igraph::ecount(Complete)) # Number of edges
```

```
## [1] 1144
```

```r
print(igraph::vcount(NoBasid))
```

```
## [1] 307
```

```r
print(igraph::ecount(NoBasid))
```

```
## [1] 1187
```

```r
print(igraph::vcount(NoHubs))
```

```
## [1] 286
```

```r
print(igraph::ecount(NoHubs))
```

```
## [1] 916
```

```r
metrics_scaled <- bind_rows(
  Complete_metrics %>% mutate(Network = "Complete"),
  Nohub_metrics %>% mutate(Network = "NoHubs"),
  Nobasid_metrics %>% mutate(Network = "NoBasid")
) %>%
  dplyr::mutate(dplyr::across(where(is.numeric), scale))

metrics_long_scaled <- metrics_scaled %>%
  tidyr::pivot_longer(cols = -c(Node, Network), names_to = "Metric", values_to = "Value")
```

bind the metrics to plot them

```r
# 3. Visualization


# Remove missing values
metrics_long_scaled <- na.omit(metrics_long_scaled)
```

```r
# We visualize only six metrics
selected_metrics <- c(
  "Degree", "Closeness", "Betweenness",
  "EigenvectorCentrality", "PageRank", "Transitivity"
)

metrics_long_filtered <- metrics_long_scaled %>%
  filter(Metric %in% selected_metrics) %>%
  mutate(
    Value = as.numeric(as.character(Value)),
    Network = recode(Network,
      "Complete" = "Complete Network",
      "NoHubs" = "Network & Module Hubs Removed",
      "NoBasid" = "Basidiobolus Subnetwork Removed" ) ) %>%
  na.omit() # Remove any NA if any

metrics_long_filtered$Network <- factor(metrics_long_filtered$Network,
  levels = c(
    "Complete Network",
    "Network & Module Hubs Removed",
    "Basidiobolus Subnetwork Removed" ))

# DspikeIn::color_palette$light_MG
network_colors <- c(
  "Complete Network" = "#F1E0C5",
  "Network & Module Hubs Removed" = "#D2A5A1",
  "Basidiobolus Subnetwork Removed" = "#B2C3A8")

# statistical comparisons a vs b
comparisons <- list(
  c("Complete Network", "Network & Module Hubs Removed"),
  c("Complete Network", "Basidiobolus Subnetwork Removed"),
  c("Network & Module Hubs Removed", "Basidiobolus Subnetwork Removed"))

networks_in_data <- unique(metrics_long_filtered$Network)
comparisons <- comparisons[sapply(comparisons, function(pair) all(pair %in% networks_in_data))]

met <- ggplot(metrics_long_filtered, aes(x = Network, y = Value, fill = Network)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(aes(color = Network),
    position = position_jitter(0.2), alpha = 0.2, size = 1.5 ) +
  scale_fill_manual(values = network_colors) +
  scale_color_manual(values = network_colors) +
  facet_wrap(~Metric, scales = "free_y", labeller = label_wrap_gen(width = 20)) +
  ggpubr::stat_compare_means(method = "wilcox.test", label = "p.signif", comparisons = comparis
  theme_minimal() +
  theme(
```

```
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.x = element_text(size = 10, angle = 10, hjust = 0.8),
    strip.text = element_text(size = 12, margin = margin(t = 15, b = 5)),
    legend.position = "top",
    legend.text = element_text(size = 13),
    legend.title = element_text(size = 13, face = "bold"),
    plot.title = element_text(size = 13, face = "bold")
  ) + labs(title = "Selected Node Metrics Across Networks", fill = "Network Type", color = "Ne
```

To find first and second neighbors your target node

```
Complete <- load_graphml("Complete.graphml")
result2 <- extract_neighbors(
  graph = Complete,
  target_node = "OTU69:Basidiobolus_sp", mode = "all")
head(result2$summary)
```

```
##             Type                            Node
## 1 First Neighbor             OTU8:Mortierella_sp
## 2 First Neighbor            OTU13:Mortierella_sp
## 3 First Neighbor            OTU15:Mortierella_sp
## 4 First Neighbor             OTU16:Ascomycota_sp
## 5 First Neighbor             OTU18:Helotiales_sp
## 6 First Neighbor OTU19:Margaritispora_monticola
```

Compute Node-Level Metrics

Metric	Description
'Node'	Node name (character format)
'Degree'	Number of edges connected to the node
'Strength'	Sum of edge weights connected to the node
'Closeness'	Closeness centrality (normalized, based on shortest paths)
'Betweenness'	Betweenness centrality (normalized, measures control over network flow)
'EigenvectorCentrality'	Eigenvector centrality (importance based on connections to influential nodes)
'PageRank'	PageRank score (importance based on incoming links)
'Transitivity'	Local clustering coefficient (tendency of a node to form triangles)
'Coreness'	Node's coreness (from k-core decomposition)

Metric	Description
'Constraint'	Burt's constraint (measures structural holes in a node's ego network)
'EffectiveSize'	Inverse of constraint (larger values = more non-redundant connections)
'Redundancy'	Sum of constraint values of a node's alters
'Community'	Community assignment from Louvain clustering
'Efficiency'	Global efficiency (average inverse shortest path length)
'Local_Efficiency'	Local efficiency (subgraph efficiency for a node's neighbors)
'Within_Module_Connectivity'	Proportion of neighbors in the same community
'Among_Module_Connectivity'	Proportion of neighbors in different communities

```
# Load required libraries
library(igraph)
library(dplyr)
library(tidyr)
library(ggplot2)
library(ggrepel)

# Compute Node-Level Metrics
completeMetrics <- node_level_metrics(Complete)
NoHubsMetrics <- node_level_metrics(NoHubs)
NoBasidMetrics <- node_level_metrics(NoBasid)

# completeMetrics$facet_plot

# Ensure each dataset has a "Network" column before combining
completeMetrics$metrics <- completeMetrics$metrics %>% mutate(Network = "Complete Network")
NoHubsMetrics$metrics <- NoHubsMetrics$metrics %>% mutate(Network = "Network & Module Hubs Remo
NoBasidMetrics$metrics <- NoBasidMetrics$metrics %>% mutate(Network = "Basidiobolus Subnetwork

# Combine All Data
combined_data <- bind_rows(
  completeMetrics$metrics,
  NoHubsMetrics$metrics,
  NoBasidMetrics$metrics
)


# Add Node Identifier if missing
if (!"Node" %in% colnames(combined_data)) {
  combined_data <- combined_data %>% mutate(Node = rownames(.))
}

# Convert `Network` into Factor
combined_data$Network <- factor(combined_data$Network, levels = c(
```

```
  "Complete Network",
  "Network & Module Hubs Removed",
  "Basidiobolus Subnetwork Removed"))

# Convert Data to Long Format
metrics_long <- combined_data %>%
  pivot_longer(
    cols = c("Redundancy", "Efficiency", "Betweenness"),
    names_to = "Metric", values_to = "Value")


# Define Custom Colors and Shapes
  network_colors <- c(
  "Complete Network" = "#F1E0C5",
  "Network & Module Hubs Removed" = "#D2A5A1",
  "Basidiobolus Subnetwork Removed" = "#B2C3A8")

network_shapes <- c(
  "Complete Network" = 21,
  "Network & Module Hubs Removed" = 22,
  "Basidiobolus Subnetwork Removed" = 23)

# Determine Top 30% of Nodes to Label/Optional
metrics_long <- metrics_long %>%
  group_by(Network, Metric) %>%
  mutate(Label = ifelse(rank(-Value, ties.method = "random") / n() <= 0.3, Node, NA))

# ?quadrant_plot() can creat plot for indivisual network
# plot <- quadrant_plot(metrics, x_metric = "Degree", y_metric = "Efficiency")

# Create comparision Plots
create_metric_plot <- function(metric_name, data, title) {
  data_filtered <- data %>% filter(Metric == metric_name)
  median_degree <- median(data_filtered$Degree, na.rm = TRUE)
  median_value <- median(data_filtered$Value, na.rm = TRUE)

  ggplot(data_filtered, aes(x = Degree, y = Value, fill = Network)) +
    geom_point(aes(shape = Network), size = 3, stroke = 1, color = "black") +
    geom_text_repel(aes(label = Label), size = 3, max.overlaps = 50) +
    scale_fill_manual(values = network_colors) +
    scale_shape_manual(values = network_shapes) +
    geom_vline(xintercept = median_degree, linetype = "dashed", color = "black", size = 1) +
    geom_hline(yintercept = median_value, linetype = "dashed", color = "black", size = 1) +
    labs(
      title = title,
      x = "Degree",
      y = metric_name,
```

```
      fill = "Network",
      shape = "Network" ) +
    theme_minimal() +
    theme(
      plot.title = element_text(
        hjust = 0.5, size = 16, face = "bold",
        margin = margin(t = 10, b = 20) # Moves the title downward
      ),
      axis.title = element_text(size = 14, face = "bold"),
      legend.position = "top",
      legend.title = element_text(size = 14, face = "bold"),
      legend.text = element_text(size = 12)  )
}


# Generate Plots
plot_redundancy <- create_metric_plot("Redundancy", metrics_long, "Redundancy vs. Degree Across
plot_efficiency <- create_metric_plot("Efficiency", metrics_long, "Efficiency vs. Degree Across
plot_betweenness <- create_metric_plot("Betweenness", metrics_long, "Betweenness vs. Degree Ac

# Save Plots
# ggsave("plot_redundancy_20percent.png", plot_redundancy, width = 8, height = 6)
# ggsave("plot_efficiency_20percent.png", plot_efficiency, width = 8, height = 6)
# ggsave("plot_betweenness_20percent.png", plot_betweenness, width = 8, height = 6)

# Print Plots
# print(plot_redundancy)
# print(plot_efficiency)
# print(plot_betweenness)
```

Compute degree metrics and visualize the network

```
# Compute degree metrics and visualize the network
# Options: `"stress"` (default), `"graphopt"`, `"fr"`

result <- degree_network(graph_path = Complete, save_metrics = TRUE)
# print(result$metrics)
# print(result$plot)


# Compute network weights for different graph structures
NH <- weight_Network(graph_path = "NoHubs.graphml")
NB <- weight_Network(graph_path = "NoBasid.graphml")
C <- weight_Network(graph_path = "Complete.graphml")
```

```r
# Extract metrics from the computed network weights
CompleteM <- C$metrics
NoHubsM <- NH$metrics
NoBasidM <- NB$metrics

# Combine metrics into a single dataframe for comparison
df <- bind_rows(
  CompleteM %>% mutate(Group = "CompleteM"),
  NoHubsM %>% mutate(Group = "NoHubsM"),
  NoBasidM %>% mutate(Group = "NoBasidM")) %>%
  pivot_longer(cols = -Group, names_to = "Metric", values_to = "Value")

# Aggregate the total values by metric and group
df_bar <- df %>%
  group_by(Metric, Group) %>%
  summarise(Total_Value = sum(Value), .groups = "drop")

# Plot the metrics comparison
pg<-ggplot(df_bar, aes(x = Metric, y = log1p(Total_Value), fill = Group)) +
  geom_bar(stat = "identity", position = "dodge", alpha = 0.8) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("#F1E0C5", "#D2A5A1", "#B2C3A8")) +
  labs(
    title = "Total Network Metrics Comparison",
    x = "Metric",
    y = "Total Value (log-scaled)",
    fill = "Group" )
```

Small-World Index Analysis

To determine whether the complete network exhibited small-world topology, we computed the Small-World Index (SWI; σ) following the quantitative framework established by Humphries M, Gurney K, 2008. PLoS ONE 4 (Eq.1) (SWI; σ) = (Global clustering coefficient real/Global clustering coefficient random)/(Avg Path real/Avg Path random))

```r
library(igraph)
library(tidygraph)
library(ggraph)
library(DspikeIn)


# AA abundance
Complete<-load_graphml("Complete.graphml")
# NoHubs<-load_graphml("NoHubs.graphml")
```

```r
# NoBasid<-load_graphml("NoBasid.graphml")

# RA abundance
# Complete_Rel<-load_graphml("~/Downloads/herp.spiecsym.Rel.graphml")

# Degree distribution
# deg <- degree(Complete_Rel)
deg <- degree(Complete)

hist(deg, breaks = 30, main = "Degree Distribution", xlab = "Degree")
fit <- fit_power_law(deg + 1)  # Avoid zero degrees
print(fit)


# ---- empirical network ----
g_empirical = Complete
g_empirical = Complete_Rel



# ---- Calculate real network metrics ----
creal <- transitivity(g_empirical, type = "global")
E(g_empirical)$weight <- abs(E(g_empirical)$weight)
lreal <- mean_distance(g_empirical, directed = FALSE, unconnected = TRUE, weights = E(g_empiri

# ---- Generate 1000 Erdős-Rényi random graphs with same size ----
set.seed(42)  # for reproducibility
n_nodes <- vcount(g_empirical)
n_edges <- ecount(g_empirical)

crand_vals <- numeric(1000)
lrand_vals <- numeric(1000)

for (i in 1:1000) {
g_rand <- sample_gnm(n = n_nodes, m = n_edges, directed = FALSE)
if (!is_connected(g_rand)) next

  crand_vals[i] <- transitivity(g_rand, type = "global")
  lrand_vals[i] <- mean_distance(g_rand, directed = FALSE, unconnected = TRUE)
}

# ---- Calculate mean values across random graphs ----
crand <- mean(crand_vals, na.rm = TRUE)
lrand <- mean(lrand_vals, na.rm = TRUE)

# ---- Compute Small-World Index ( ) ----
sigma <- (creal / crand) / (lreal / lrand)
```

```
cat("Global clustering coefficient (real):", creal, "\n")
cat("Average path length (real):", lreal, "\n")
cat("Mean clustering coefficient (random):", crand, "\n")
cat("Mean path length (random):", lrand, "\n")
cat("Small-World Index ( ):", round(sigma, 2), "\n")
```

```
sessionInfo()
```

```
## R version 4.5.0 (2025-04-11)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sequoia 15.5
##
## Matrix products: default
## BLAS:   /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framew
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] ggrepel_0.9.6                   igraph_2.1.4                    ggpubr_0.6.0
##  [4] tidyr_1.3.1                     DspikeIn_0.99.11                rmarkdown_2.29
##  [7] mia_1.16.0                      MultiAssayExperiment_1.34.0     vegan_2.7-1
## [10] permute_0.9-7                   microbiome_1.30.0               tibble_3.3.0
## [13] dplyr_1.1.4                     flextable_0.9.9                 TreeSummarizedExperimer
## [16] SingleCellExperiment_1.30.1     SummarizedExperiment_1.38.1     Biobase_2.68.0
## [19] GenomicRanges_1.60.0            MatrixGenerics_1.20.0           matrixStats_1.5.0
## [22] ggplot2_3.5.2                   ggstar_1.0.4                    phyloseq_1.52.0
## [25] Biostrings_2.76.0               GenomeInfoDb_1.44.0             XVector_0.48.0
## [28] IRanges_2.42.0                  S4Vectors_0.46.0                BiocGenerics_0.54.0
## [31] generics_0.1.4                  BiocStyle_2.36.0
##
## loaded via a namespace (and not attached):
##   [1] urlchecker_1.0.1           TH.data_1.1-3              vctrs_0.6.5                di
##   [5] BiocBaseUtils_1.10.0       rbiom_2.2.0                parallelly_1.45.0          ms
##   [9] MASS_7.3-65                fontLiberation_0.1.0       reshape2_1.4.4             ht
##  [13] foreach_1.5.2              withr_3.0.2                xfun_0.52                  gg
##  [17] ellipsis_0.3.2             survival_3.8-3             memoise_2.0.1              co
##  [21] ggbeeswarm_0.7.2           emmeans_1.11.1             profvis_0.4.0              sy
```

```
##  [25] ragg_1.4.0                 tidytree_0.4.6             zoo_1.8-14                 gg
##  [29] Formula_1.2-5              sys_3.4.3                  promises_1.3.3             ht
##  [33] rstatix_0.7.2              rhdf5filters_1.20.0        ps_1.9.1                   rl
##  [37] rstudioapi_0.17.1          UCSC.utils_1.4.0           miniUI_0.1.2               gg
##  [41] processx_3.8.6             curl_6.3.0                 ScaledMatrix_1.16.0        gg
##  [45] polyclip_1.10-7            randomForest_4.7-1.2       GenomeInfoDbData_1.2.14    qu
##  [49] SparseArray_1.8.0          RBGL_1.84.0                xtable_1.8-4               st
##  [53] desc_1.4.3                 ade4_1.7-23                evaluate_1.0.4             S4
##  [57] BiocFileCache_2.16.0       hms_1.1.3                  bookdown_0.43              ir
##  [61] filelock_1.0.3             readxl_1.4.5               readr_2.1.5                ma
##  [65] later_1.4.2                viridis_0.6.5              ggtree_3.16.0              la
##  [69] DECIPHER_3.4.0             XML_3.99-0.18              scuttle_1.18.0             pi
##  [73] nlme_3.1-168               iterators_1.0.14           decontam_1.28.0            co
##  [77] beachmat_2.24.0            stringi_1.8.7              biomformat_1.36.0          de
##  [81] plyr_1.8.9                 crayon_1.5.3               abind_1.4-8                sc
##  [85] ggtext_0.1.2               gridGraphics_0.5-1         locfit_1.5-9.12            gr
##  [89] bit_4.6.0                  sandwich_3.1-1             fastmatch_1.1-6            co
##  [93] multcomp_1.4-28            textshaping_1.0.1          BiocSingular_1.24.0        op
##  [97] slam_0.1-55                bslib_0.9.0                multtest_2.64.0            mi
## [101] splines_4.5.0              Rcpp_1.0.14                dbplyr_2.5.0               sp
## [105] BiocCheck_1.44.2           cellranger_1.1.0           gridtext_0.1.5             kn
## [109] blob_1.2.4                 utf8_1.2.6                 fs_1.6.6                   De
## [113] pkgbuild_1.4.8             ggsignif_0.6.4             ggplotify_0.1.2            es
## [117] Matrix_1.7-3               callr_3.7.6                statmod_1.5.0              tz
## [121] tweenr_2.0.3               pkgconfig_2.0.3            tools_4.5.0                ca
## [125] RSQLite_2.4.1              viridisLite_0.4.2          DBI_1.2.3                  fa
## [129] scales_1.4.0               grid_4.5.0                 credentials_2.0.2          us
## [133] fillpattern_1.0.2          broom_1.0.8                sass_0.4.10                or
## [137] patchwork_1.3.0            coda_0.19-4.1              BiocManager_1.30.26        gr
## [141] carData_3.0-5              farver_2.1.2               tidygraph_1.3.1            mg
## [145] biocViews_1.76.0           yaml_2.3.10                roxygen2_7.3.2             cl
## [149] purrr_1.0.4                lifecycle_1.0.4            rsconnect_1.4.2            as
## [153] mvtnorm_1.3-3              bluster_1.18.0             sessioninfo_1.2.3          ba
## [157] BiocParallel_1.42.1        gtable_0.3.6               ggridges_0.5.6             pa
## [161] ape_5.8-1                  testthat_3.2.3             limma_3.64.1               js
## [165] edgeR_4.6.2                bitops_1.0-9               bit64_4.6.0-1              bi
## [169] Rtsne_0.17                 yulab.utils_0.2.0          BiocNeighbors_2.2.0        zi
## [173] jquerylib_0.1.4            lazyeval_0.2.2             shiny_1.10.0               ht
## [177] rappdirs_0.3.3             tinytex_0.57               glue_1.8.0                 ht
## [181] gdtools_0.4.2              RCurl_1.98-1.17            rprojroot_2.0.4            tr
## [185] gridExtra_2.3              R6_2.6.1                   DESeq2_1.48.1              la
## [189] cluster_2.1.8.1            pkgload_1.4.0              Rhdf5lib_1.30.0            st
## [193] aplot_0.2.6                DirichletMultinomial_1.50.0 DelayedArray_0.34.1       ti
## [197] vipor_0.4.7                ggforce_0.5.0              xml2_1.3.8                 fo
## [201] car_3.1-3                  rsvd_1.0.5                 fontquiver_0.2.1           da
## [205] htmlwidgets_1.6.4          RColorBrewer_1.1-3         rlang_1.1.6                ge
## [209] uuid_1.2-1                 remotes_2.5.0              RUnit_0.4.33.1             gg
## [213] phangorn_2.12.1            beeswarm_0.4.0
```

Dspikeln volume protocol

Spike-in volume Protocol;

The species Tetragenococcus halophilus (bacterial spike; ATCC33315) and Dekkera bruxellensis (fungal spike; WLP4642-White Labs) were selected as taxa to spike into gut microbiome samples as they were not found in an extensive collection of wildlife skin (GenBank BioProjects: PRJNA1114724, PRJNA 1114659) or gut microbiome samples. Stock cell suspensions of both microbes were grown in either static tryptic soy broth (T. halophilus) or potato dextrose broth (D. bruxellensis) for 72 hours then serially diluted and optical density (OD600) determined on a ClarioStar plate reader. Cell suspensions with an optical density of 1.0, 0.1, 0.01, 0.001 were DNA extracted using the Qiagen DNeasy Powersoil Pro Kit. These DNA isolations were used as standards to determine the proper spike in volume of cells to represent 0.1-10% of a sample (Rao et al., 2021b) Fecal pellets (3.1 ± 1.6 mg; range = 1 – 5.1 mg) from an ongoing live animal study using wood frogs (Lithobates sylvaticus) were used to standardize the input material for the development of this protocol. A total of (n=9) samples were used to validate the spike in protocol. Each fecal sample was homogenized in 1mL of sterile molecular grade water then 250uL of fecal slurry was DNA extracted as above with and without spiked cells. Two approaches were used to evaluate the target spike-in of 0.1-10%, the range of effective spike-in percentage described in (Rao et al., 2021b), including 1) an expected increase of qPCR cycle threshold (Ct) value that is proportional to the amount of spiked cells and 2) the expected increase in copy number of T. halophilus and D. bruxellensis in spiked vs. unspiked samples. A standard curve was generated using a synthetic fragment of DNA for the 16S-V4 rRNA and ITS1 rDNA regions of T. halophilus and D. bruxellensis, respectively. The standard curve was used to convert Ct values into log copy number for statistical analyses (detailed approach in[2, 3]) using the formula y = -0.2426x + 10.584 for T. halophilus and y = -0.3071x + 10.349 for D. bruxellensis, where x is the average Ct for each unknown sample. Quantitative PCR (qPCR) was used to compare known copy numbers from synthetic DNA sequences of T. halophilus and D. bruxellensis to DNA extractions of T. halophilus and D. bruxellensis independently, and wood frog fecal samples with and without spiked cells. SYBR qPCR assays were run at 20ul total volume including 10ul 2X Quantabio PerfeCTa SYBR Green Fastmix, 1ul of 10uM forward and reverse primers, 1ul of ArcticEnzymes dsDNAse master mix clean up kit, and either 1ul of DNA for D. bruxellensis or 3ul for T. halophilus. Different volumes of DNA were chosen for amplification of bacteria and fungi due to previous optimization of library preparation and sequencing steps [3]. The 515F [4] and 806R [5] primers were chosen to amplify bacteria and ITS1FI2 [6] and ITS2 for fungi, as these are the same primers used during amplicon library preparation and sequencing. Cycling conditions on an Agilent AriaMX consisted of 95 C for 3 mins followed by 40 cycles of 95 C for 15 sec, 60 C for 30 sec and 72 C for 30 sec. Following amplification, a melt curve was generated under the following conditions including 95 C for 30 sec, and a melt from 60 C to 90 C increasing in resolution of 0.5 C in increments of a 5 sec soak time. To validate the spike in protocol we selected two sets of fecal samples including 360 samples from a diverse species pool of frogs, lizards, salamanders and snakes and a more targeted approach of 122 fecal samples from three genera of salamanders from the Plethodontidae. (Supplemental Table #). FFecal samples were not weighed in the field, rather, a complete fecal pellet was diluted in an equal volume of sterile water and standardized volume of fecal slurry (250µL) extracted for independent samples.. A volume of 1ul T. halophilus (1847 copies) and 1ul D. bruxellensis (733 copies) were spiked into each fecal sample then DNA was extracted as above, libraries constructed, and amplicon sequenced on an Illumina MiSeq as in [7] .

https://github.com/mghotbi/DspikeIn