Microbiome data analysis with microViz

Working with barcharts

David Barnett

2023-10-26

Table of Contents

# Setup

getwd()

## [1] "/xdisk/bhurwitz/bh\_class/bonnie/exercises/13\_microviz"

library(rmarkdown)  
library(seriation)  
library(dplyr)  
library(purrr)  
library(ggplot2)  
library(phyloseq)  
library(microViz)  
library(shiny)

mice <- readRDS('/xdisk/bhurwitz/bh\_class/data/microviz/mice.rds')

Now that we are getting into microbiome analytics, we are going to transition to using more complete datasets with taxonomic data and metadata. We’ll start by using a gut microbiome from a study of antibiotic administration in mice.

* mice is 16S rRNA gene amplicon sequencing data, from the mouse antibiotics study
  + They used Illumina MiSeq and processed the data into ASVs using DADA2

The data has already been processed from fastq files into counts per taxon, and are available to you on the UA cluster. This is the exciting bit, you get to explore and visualize the data, and do statistics (yay!).

# Intro to phyloseq

This is a phyloseq S4 object, containing processed microbiota data from the mouse study.

**Click here for more details about the mice dataset:**

**The mice data**

The data originate from a study on the effects of oral antibiotic administration on flavivirus infection (<https://www.ncbi.nlm.nih.gov/pubmed/29590614>). Sequence data was generated from extracted nucleic acid from stool samples collected from individually caged mice and amplified using primers specific for the V4 region using primers 515F/806R.

The study followed flavivirus infection after the following treatments:

1. Koolaid: Antibiotics are provided to the mice via their drinking water. As many of the antibiotics taste bad, koolaid is added as a sweetener. Therefore, the appropriate control is water spiked and labelled *koolaid*.
2. Ampicillin (Amp): <https://en.wikipedia.org/wiki/Ampicillin>
3. Metronidazole (Met): <https://en.wikipedia.org/wiki/Metronidazole>
4. Ampicillin + Metronidazole (Amp+Metro)

Treatments were supplied ad libitum for 2 weeks prior to viral infection and maintained for 2 weeks post-infection. Primary outcome was mouse survival. Each treatment group had two subgroups of mice that were either a) left uninfected as controls or b) infected with West Nile Virus via a subcutaneous foot pad injection.

Get a little familiar with the object. What does it have in it? Can you look at each part?

The printed object shows you functions you can use to access the data inside.

You can also use the @ symbol.

mice

## phyloseq-class experiment-level object  
## otu\_table() OTU Table: [ 3229 taxa and 520 samples ]  
## sample\_data() Sample Data: [ 520 samples by 11 sample variables ]  
## tax\_table() Taxonomy Table: [ 3229 taxa by 7 taxonomic ranks ]

# View(mice)

tax\_table(mice) %>% head()

## Taxonomy Table: [6 taxa by 7 taxonomic ranks]:  
## Kingdom Phylum Class Order   
## ASV0001 "Bacteria" "Bacteroidetes" "Bacteroidia" "Bacteroidales"   
## ASV0002 "Bacteria" "Proteobacteria" "Gammaproteobacteria" "Xanthomonadales"  
## ASV0003 "Bacteria" "Bacteroidetes" "Bacteroidia" "Bacteroidales"   
## ASV0004 "Bacteria" "Bacteroidetes" "Bacteroidia" "Bacteroidales"   
## ASV0005 "Bacteria" "Bacteroidetes" "Bacteroidia" "Bacteroidales"   
## ASV0006 "Bacteria" "Bacteroidetes" "Bacteroidia" "Bacteroidales"   
## Family Genus Species   
## ASV0001 "Porphyromonadaceae" NA NA   
## ASV0002 "Xanthomonadaceae" "Stenotrophomonas" "maltophilia"  
## ASV0003 "Porphyromonadaceae" NA NA   
## ASV0004 "Porphyromonadaceae" NA NA   
## ASV0005 "Porphyromonadaceae" NA NA   
## ASV0006 "Porphyromonadaceae" NA NA

rank\_names(mice)

## [1] "Kingdom" "Phylum" "Class" "Order" "Family" "Genus" "Species"

otu\_table(mice)[1:15, 1:8]

## OTU Table: [8 taxa and 15 samples]  
## taxa are columns  
## ASV0001 ASV0002 ASV0003 ASV0004 ASV0005 ASV0006 ASV0007 ASV0008  
## D14.A1 3343 0 4205 3470 3607 1210 1159 1852  
## D14.B5 4332 0 5412 2494 3083 1451 1663 1745  
## D0.D5 5344 0 3906 1439 2396 1402 1217 2078  
## D0.E1 2994 0 4005 2188 2882 1267 1821 1788  
## D0.E2 2315 0 3987 955 1665 1025 899 1342  
## D0.E3 1972 0 4336 1876 3422 1208 551 1900  
## D0.E4 2352 0 2561 1960 2060 1350 1095 1200  
## D0.E5 1386 0 1752 1471 1363 724 1050 871  
## D0.F1 3795 0 4052 2226 3130 539 162 247  
## D0.F2 2867 0 2063 435 1412 410 0 150  
## D0.F3 3850 0 4203 508 2839 533 0 230  
## D0.F4 6740 0 5045 1356 5755 517 85 210  
## D14.C1 2664 3 2892 1530 2176 735 1508 1577  
## D0.F5 5303 3 4203 3313 4471 1241 344 441  
## D0.G1 3052 0 3393 3209 2867 873 372 1512

# mice@otu\_table[1:15, 1:10] # the same result

sample\_variables(mice)

## [1] "sample\_id" "barcode" "run" "plate"   
## [5] "sample" "sex" "cage" "treatment\_days"   
## [9] "treatment" "virus" "survival\_status"

sample\_data(mice)[1:15, 1:5]

## sample\_id barcode run plate sample  
## D14.A1 1.Thackray.D14.A1 ACGAGACTGATT 368 1 1  
## D14.B5 10.Thackray.D14.B5 ACCGGTATGTAC 368 1 10  
## D0.D5 100.Thackray.D0.D5 ATCTACCGAAGC 368 2 100  
## D0.E1 101.Thackray.D0.E1 ACTTGGTGTAAG 368 2 101  
## D0.E2 102.Thackray.D0.E2 TCTTGGAGGTCA 368 2 102  
## D0.E3 103.Thackray.D0.E3 TCACCTCCTTGT 368 2 103  
## D0.E4 104.Thackray.D0.E4 GCACACCTGATA 368 2 104  
## D0.E5 105.Thackray.D0.E5 GCGACAATTACA 368 2 105  
## D0.F1 106.Thackray.D0.F1 TCATGCTCCATT 368 2 106  
## D0.F2 107.Thackray.D0.F2 AGCTGTCAAGCT 368 2 107  
## D0.F3 108.Thackray.D0.F3 GAGAGCAACAGA 368 2 108  
## D0.F4 109.Thackray.D0.F4 TACTCGGGAACT 368 2 109  
## D14.C1 11.Thackray.D14.C1 AATTGTGTCGGA 368 1 11  
## D0.F5 110.Thackray.D0.F5 CGTGCTTAGGCT 368 2 110  
## D0.G1 111.Thackray.D0.G1 TACCGAAGGTAT 368 2 111

sample\_names(mice) %>% head(10)

## [1] "D14.A1" "D14.B5" "D0.D5" "D0.E1" "D0.E2" "D0.E3" "D0.E4" "D0.E5"   
## [9] "D0.F1" "D0.F2"

# Looking at microbiome data

Okay, so how do we look at the microbiota abundance data? To do this, we’re going to use the R package microViz

## Barcharts: bad to better

Lets take a very small subset of this data to get started. Just the control group (vehicle treatment) at day 13.

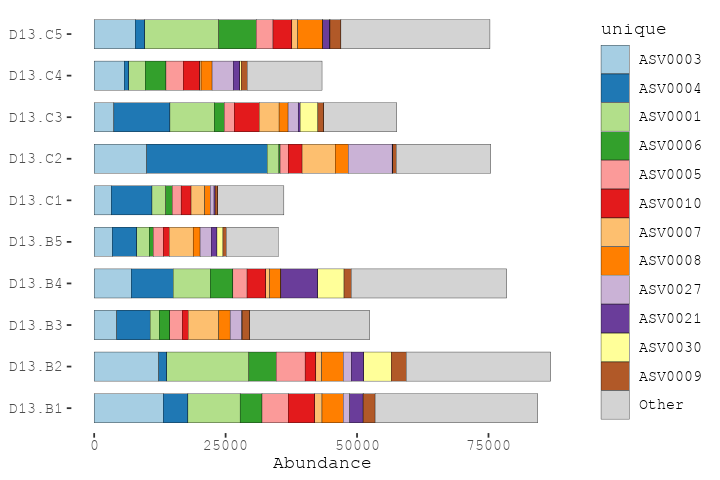
# We can filter the samples like this, using the sample\_data information  
mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle')

## phyloseq-class experiment-level object  
## otu\_table() OTU Table: [ 563 taxa and 10 samples ]  
## sample\_data() Sample Data: [ 10 samples by 11 sample variables ]  
## tax\_table() Taxonomy Table: [ 563 taxa by 7 taxonomic ranks ]

### Bad bars

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
 comp\_barplot(  
 tax\_level = 'unique', n\_taxa = 12, bar\_width = 0.7,  
 sample\_order = 'asis', tax\_transform\_for\_plot = 'identity'  
 ) +  
 coord\_flip()

## NAs detected in phyloseq tax\_table:  
## Consider using tax\_fix() to make taxa uniquely identifiable  
## NAs detected in phyloseq tax\_table:  
## Consider using tax\_fix() to make taxa uniquely identifiable



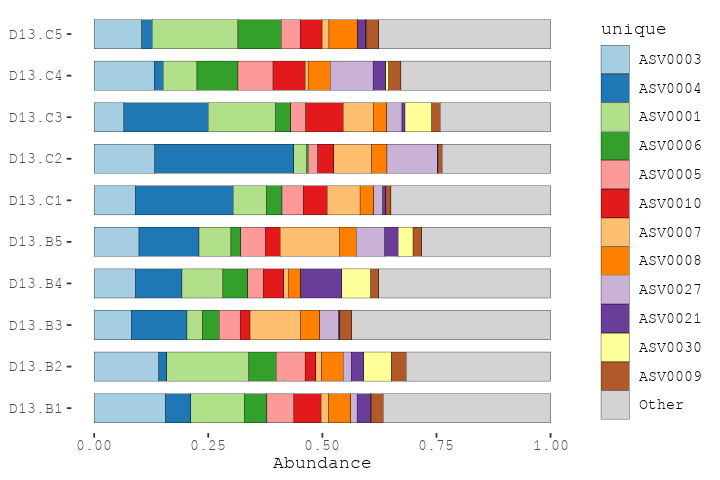
What is going on here?? The unique taxa have uninformative IDs, and we also got a message about problems with the taxonomy table.

The total number of reads also varies a lot between samples! The total number of reads for each sample is NOT a reliable indicator of the biomass or bacterial load of each sample. So for now we will just consider the relative abundance of each taxon, as proportions of the total counts for that sample.

### Compositions (%)

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
 comp\_barplot(  
 tax\_level = 'unique', n\_taxa = 12, sample\_order = 'asis', bar\_width = 0.7  
 ) +  
 coord\_flip()

## NAs detected in phyloseq tax\_table:  
## Consider using tax\_fix() to make taxa uniquely identifiable  
## NAs detected in phyloseq tax\_table:  
## Consider using tax\_fix() to make taxa uniquely identifiable



### Fixing tax\_table

Let’s look at the taxonomy table interactively

# tax\_fix\_interactive(mice) # run this in the R Console for an interactive look

Looks like we just need to fill in some blank cells when a sequence was not classified at genus or family. tax\_fix can do this, it just copies down info from a higher rank classification. Let’s update our mice phyloseq object with this fix.

mice <- tax\_fix(mice, verbose = FALSE)

We can also rename the unique taxa with a more informative name, according to their classification at the rank of Family (and how common they are).

mice %>% taxa\_names() %>% head

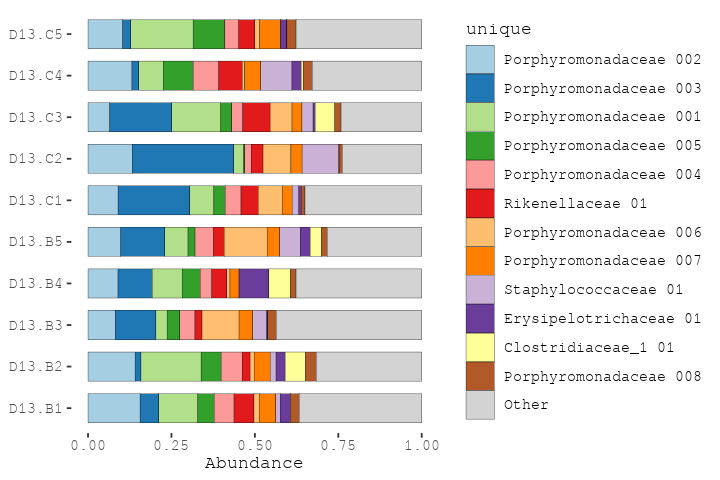
## [1] "ASV0001" "ASV0002" "ASV0003" "ASV0004" "ASV0005" "ASV0006"

mice <- tax\_rename(mice, rank = 'Family')  
mice %>% taxa\_names() %>% head

## [1] "Porphyromonadaceae 001" "Xanthomonadaceae 01" "Porphyromonadaceae 002"  
## [4] "Porphyromonadaceae 003" "Porphyromonadaceae 004" "Porphyromonadaceae 005"

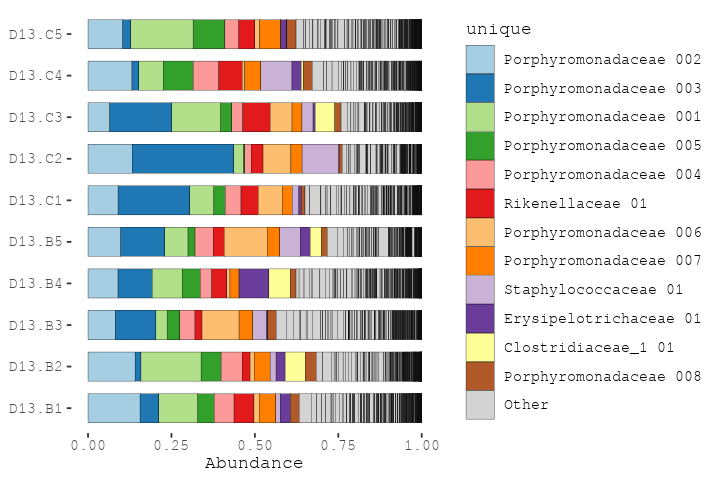
Let’s try again with the better names.

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
 comp\_barplot(  
 tax\_level = 'unique', n\_taxa = 12, sample\_order = 'asis', bar\_width = 0.7  
 ) +  
 coord\_flip()



Sadly we don’t have enough distinct colours to show all the unique taxa.

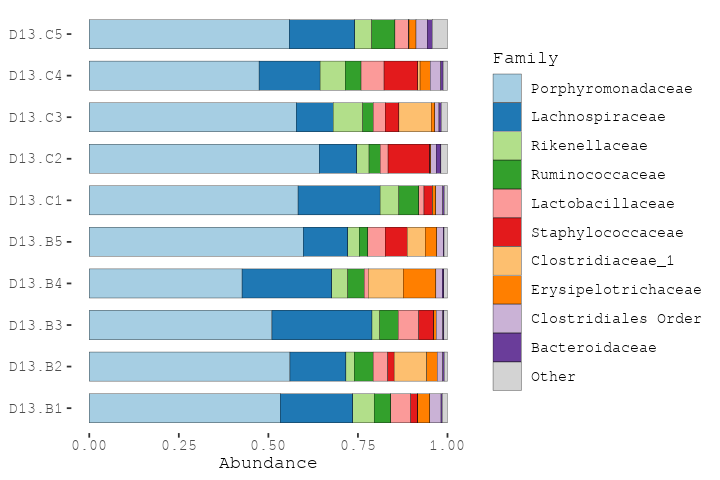
mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
 comp\_barplot(  
 tax\_level = 'unique', n\_taxa = 12, sample\_order = 'asis', bar\_width = 0.7,  
 merge\_other = FALSE  
 ) +  
 coord\_flip()



### Aggregating taxa

So let’s “aggregate” all the counts into family-level groups. For each family, this adds together all the counts from every ASV that belongs to that family. We can do that by changing the tax\_level argument to “Family”.

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
 comp\_barplot(  
 tax\_level = "Family", n\_taxa = 10, bar\_width = 0.7, sample\_order = 'asis'  
 ) +  
 coord\_flip()

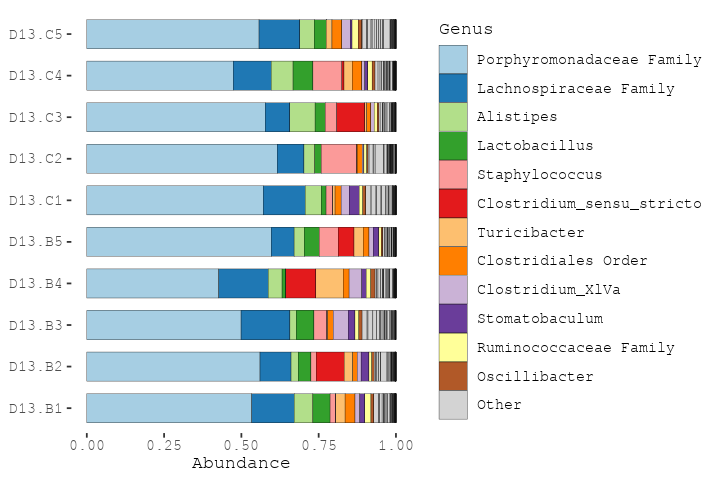


By aggregating at family level, we have sacrificed taxonomic resolution, compared to using ASVs. But this way we can get an idea of which families are the most abundant, and how variable the communities are.

Try making some similar plots aggregated at different taxonomic ranks.

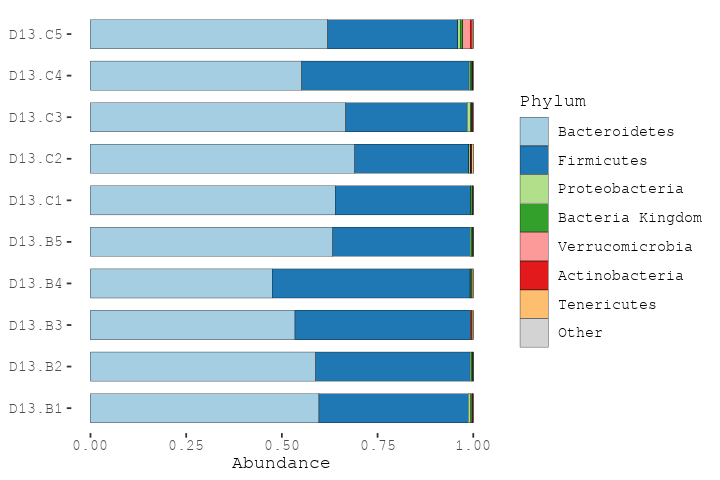
# rank\_names(mice)  
# mice %>%  
# ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%  
# comp\_barplot(tax\_level = , n\_taxa = 10, sample\_order = 'asis', merge\_other = FALSE)

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%   
 comp\_barplot(  
 tax\_level = "Genus", n\_taxa = 12, bar\_width = 0.7,  
 sample\_order = 'asis', merge\_other = FALSE  
 ) +  
 coord\_flip()



Many of the ASVs in this mice data, the *Porphyromonadaceae*, could not be classified at genus level.

mice %>%   
 ps\_filter(treatment\_days == 'D13', virus == 'WNV2000', treatment == 'Vehicle') %>%   
 comp\_barplot(  
 tax\_level = "Phylum", n\_taxa = 7, bar\_width = 0.7, sample\_order = 'asis'  
 ) +  
 coord\_flip()



### Fickle Phyla

A note on phylum names! There have been major changes this year and some of these are now old names. Most published research is of course with the old names (and still probably will be for a year or so).

# Fun with barcharts

More examples/tutorial of visualizing microbiome data using stacked barcharts can be found here: <https://david-barnett.github.io/microViz/articles/web-only/compositions.html>

Try it out for yourself a bit!

Bar charts often look better when you sort the samples by similarity. The webpage mentions using Bray-Curtis distances and hierarchical clustering to sort samples. We haven’t discussed dissimilarity or distances yet, but we will in the next session today!

For now, just appreciate that it can make the bar charts easier to read!

# Session info

Records your package versions etc. Useful for debugging / reproducing analysis.

devtools::session\_info()

## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'  
## had status 1

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value  
## version R version 4.2.2 (2022-10-31)  
## os CentOS Linux 7 (Core)  
## system x86\_64, linux-gnu  
## ui X11  
## language (EN)  
## collate C  
## ctype en\_US.UTF-8  
## tz UTC  
## date 2023-10-26  
## pandoc 2.18 @ /opt/ohpc/pub/apps/rstudio-server/2022.07.1-554/bin/pandoc/ (via rmarkdown)  
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date (UTC) lib source  
## ade4 1.7-22 2023-02-06 [1] CRAN (R 4.2.2)  
## ape 5.7-1 2023-03-13 [1] CRAN (R 4.2.2)  
## Biobase 2.58.0 2022-11-01 [1] Bioconductor  
## BiocGenerics 0.44.0 2022-11-01 [1] Bioconductor  
## biomformat 1.26.0 2022-11-01 [1] Bioconductor  
## Biostrings 2.66.0 2022-11-01 [1] Bioconductor  
## bitops 1.0-7 2021-04-24 [1] CRAN (R 4.2.2)  
## ca 0.71.1 2020-01-24 [1] CRAN (R 4.2.2)  
## cachem 1.0.8 2023-05-01 [1] CRAN (R 4.2.2)  
## callr 3.7.3 2022-11-02 [1] CRAN (R 4.2.2)  
## cli 3.6.1 2023-03-23 [1] CRAN (R 4.2.2)  
## cluster 2.1.4 2022-08-22 [3] CRAN (R 4.2.2)  
## codetools 0.2-18 2020-11-04 [3] CRAN (R 4.2.2)  
## colorspace 2.1-0 2023-01-23 [1] CRAN (R 4.2.2)  
## crayon 1.5.2 2022-09-29 [1] CRAN (R 4.2.2)  
## data.table 1.14.8 2023-02-17 [1] CRAN (R 4.2.2)  
## devtools 2.4.5 2022-10-11 [1] CRAN (R 4.2.2)  
## digest 0.6.33 2023-07-07 [1] CRAN (R 4.2.2)  
## dplyr \* 1.1.3 2023-09-03 [1] CRAN (R 4.2.2)  
## ellipsis 0.3.2 2021-04-29 [1] CRAN (R 4.2.2)  
## evaluate 0.22 2023-09-29 [1] CRAN (R 4.2.2)  
## fansi 1.0.5 2023-10-08 [1] CRAN (R 4.2.2)  
## farver 2.1.1 2022-07-06 [1] CRAN (R 4.2.2)  
## fastmap 1.1.1 2023-02-24 [1] CRAN (R 4.2.2)  
## foreach 1.5.2 2022-02-02 [1] CRAN (R 4.2.2)  
## fs 1.6.3 2023-07-20 [1] CRAN (R 4.2.2)  
## generics 0.1.3 2022-07-05 [1] CRAN (R 4.2.2)  
## GenomeInfoDb 1.34.9 2023-02-02 [1] Bioconductor  
## GenomeInfoDbData 1.2.9 2023-09-12 [1] Bioconductor  
## ggplot2 \* 3.4.4 2023-10-12 [1] CRAN (R 4.2.2)  
## glue 1.6.2 2022-02-24 [1] CRAN (R 4.2.2)  
## gtable 0.3.4 2023-08-21 [1] CRAN (R 4.2.2)  
## htmltools 0.5.6.1 2023-10-06 [1] CRAN (R 4.2.2)  
## htmlwidgets 1.6.2 2023-03-17 [1] CRAN (R 4.2.2)  
## httpuv 1.6.11 2023-05-11 [1] CRAN (R 4.2.2)  
## igraph 1.5.1 2023-08-10 [1] CRAN (R 4.2.2)  
## IRanges 2.32.0 2022-11-01 [1] Bioconductor  
## iterators 1.0.14 2022-02-05 [1] CRAN (R 4.2.2)  
## jsonlite 1.8.7 2023-06-29 [1] CRAN (R 4.2.2)  
## knitr 1.44 2023-09-11 [1] CRAN (R 4.2.2)  
## labeling 0.4.3 2023-08-29 [1] CRAN (R 4.2.2)  
## later 1.3.1 2023-05-02 [1] CRAN (R 4.2.2)  
## lattice 0.20-45 2021-09-22 [3] CRAN (R 4.2.2)  
## lifecycle 1.0.3 2022-10-07 [1] CRAN (R 4.2.2)  
## magrittr 2.0.3 2022-03-30 [1] CRAN (R 4.2.2)  
## MASS 7.3-58.1 2022-08-03 [3] CRAN (R 4.2.2)  
## Matrix 1.5-1 2022-09-13 [3] CRAN (R 4.2.2)  
## memoise 2.0.1 2021-11-26 [1] CRAN (R 4.2.2)  
## mgcv 1.8-41 2022-10-21 [3] CRAN (R 4.2.2)  
## microbiome 1.20.0 2022-11-01 [1] Bioconductor  
## microViz \* 0.11.0 2023-10-16 [1] https://david-barnett.r-universe.dev (R 4.2.2)  
## mime 0.12 2021-09-28 [1] CRAN (R 4.2.2)  
## miniUI 0.1.1.1 2018-05-18 [1] CRAN (R 4.2.2)  
## multtest 2.54.0 2022-11-01 [1] Bioconductor  
## munsell 0.5.0 2018-06-12 [1] CRAN (R 4.2.2)  
## nlme 3.1-160 2022-10-10 [3] CRAN (R 4.2.2)  
## permute 0.9-7 2022-01-27 [1] CRAN (R 4.2.2)  
## phyloseq \* 1.42.0 2022-11-01 [1] Bioconductor  
## pillar 1.9.0 2023-03-22 [1] CRAN (R 4.2.2)  
## pkgbuild 1.4.2 2023-06-26 [1] CRAN (R 4.2.2)  
## pkgconfig 2.0.3 2019-09-22 [1] CRAN (R 4.2.2)  
## pkgload 1.3.3 2023-09-22 [1] CRAN (R 4.2.2)  
## plyr 1.8.9 2023-10-02 [1] CRAN (R 4.2.2)  
## prettyunits 1.2.0 2023-09-24 [1] CRAN (R 4.2.2)  
## processx 3.8.2 2023-06-30 [1] CRAN (R 4.2.2)  
## profvis 0.3.8 2023-05-02 [1] CRAN (R 4.2.2)  
## promises 1.2.1 2023-08-10 [1] CRAN (R 4.2.2)  
## ps 1.7.5 2023-04-18 [1] CRAN (R 4.2.2)  
## purrr \* 1.0.2 2023-08-10 [1] CRAN (R 4.2.2)  
## R6 2.5.1 2021-08-19 [1] CRAN (R 4.2.2)  
## Rcpp 1.0.11 2023-07-06 [1] CRAN (R 4.2.2)  
## RCurl 1.98-1.12 2023-03-27 [1] CRAN (R 4.2.2)  
## registry 0.5-1 2019-03-05 [1] CRAN (R 4.2.2)  
## remotes 2.4.2.1 2023-07-18 [1] CRAN (R 4.2.2)  
## reshape2 1.4.4 2020-04-09 [1] CRAN (R 4.2.2)  
## rhdf5 2.42.1 2023-04-07 [1] Bioconductor  
## rhdf5filters 1.10.1 2023-03-24 [1] Bioconductor  
## Rhdf5lib 1.20.0 2022-11-01 [1] Bioconductor  
## rlang 1.1.1 2023-04-28 [1] CRAN (R 4.2.2)  
## rmarkdown \* 2.25 2023-09-18 [1] CRAN (R 4.2.2)  
## rstudioapi 0.15.0 2023-07-07 [1] CRAN (R 4.2.2)  
## Rtsne 0.16 2022-04-17 [1] CRAN (R 4.2.2)  
## S4Vectors 0.36.2 2023-02-26 [1] Bioconductor  
## scales 1.2.1 2022-08-20 [1] CRAN (R 4.2.2)  
## seriation \* 1.5.1 2023-07-20 [1] CRAN (R 4.2.2)  
## sessioninfo 1.2.2 2021-12-06 [1] CRAN (R 4.2.2)  
## shiny \* 1.7.5.1 2023-10-14 [1] CRAN (R 4.2.2)  
## stringi 1.7.12 2023-01-11 [1] CRAN (R 4.2.2)  
## stringr 1.5.0 2022-12-02 [1] CRAN (R 4.2.2)  
## survival 3.4-0 2022-08-09 [3] CRAN (R 4.2.2)  
## tibble 3.2.1 2023-03-20 [1] CRAN (R 4.2.2)  
## tidyr 1.3.0 2023-01-24 [1] CRAN (R 4.2.2)  
## tidyselect 1.2.0 2022-10-10 [1] CRAN (R 4.2.2)  
## TSP 1.2-4 2023-04-04 [1] CRAN (R 4.2.2)  
## urlchecker 1.0.1 2021-11-30 [1] CRAN (R 4.2.2)  
## usethis 2.2.2 2023-07-06 [1] CRAN (R 4.2.2)  
## utf8 1.2.3 2023-01-31 [1] CRAN (R 4.2.2)  
## vctrs 0.6.4 2023-10-12 [1] CRAN (R 4.2.2)  
## vegan 2.6-4 2022-10-11 [1] CRAN (R 4.2.2)  
## withr 2.5.1 2023-09-26 [1] CRAN (R 4.2.2)  
## xfun 0.40 2023-08-09 [1] CRAN (R 4.2.2)  
## xtable 1.8-4 2019-04-21 [1] CRAN (R 4.2.2)  
## XVector 0.38.0 2022-11-01 [1] Bioconductor  
## yaml 2.3.7 2023-01-23 [1] CRAN (R 4.2.2)  
## zlibbioc 1.44.0 2022-11-01 [1] Bioconductor  
##   
## [1] /groups/bhurwitz/R/library\_R\_v4.2.2  
## [2] /home/u20/bhurwitz/R/x86\_64-pc-linux-gnu-library/4.2  
## [3] /opt/ohpc/pub/apps/R/4.2.2/lib64/R/library  
##   
## ──────────────────────────────────────────────────────────────────────────────