Wk3 Microbiome Analysis - Data Management and Transformations

Christine V. Hawkes

1/26/2022

Contents

Overview

Learning objectives:

- how to build a phyloseq object
- subsetting and filtering data in phyloseq
- centered log ratio transformation
- variance stabilizing transformation
- practice for all of the above

References:

- McMurdie and Holmes (2014) Waste not, want not: why rarefying microbiome data is inadmissable. PLoS Comput Biol 10: e1003531. doi:10.1371/journal.pcbi.1003531
- Gloor et al. (2017) Microbiome datasets are compositional: and this is not optional. Frontiers Microbiology 8:2224. doi:10.3389/fmicb.2017.02224

- phyloseq manual. https://bioconductor.org/packages/devel/bioc/manuals/phyloseq/man/phyloseq.pdf
- phyloseq website. https://joey711.github.io/phyloseq/index.html
- phyloseq vignette. https://bioconductor.org/packages/devel/bioc/vignettes/phyloseq/inst/doc/phyloseq-basics.html

---

Install new R packages and load libraries

```
# install the following packages if you don't have them
# packages may ask to update all/some/none dependencies
BiocManager::install("microbiome")
BiocManager::install("DESeq2")
BiocManager::install("phyloseq")
install.packages("compositions")
remotes::install_github("cpauvert/psadd")

# load libraries
library(phyloseq)
library(microbiome)
library(DESeq2)
library(ggplot2)
library(tools)
library(compositions)
library(psadd)
```

Load Data

```
#specify row and col names when opening the files

ASV_data <- read.csv("wk3_seqtab_nochim.csv", header=TRUE, row.names=1) #add raw url
str(ASV_data[,1:5]) # check for matrix
```

```
# 'data.frame': 20 obs. of  5 variables:
#  $ ASV1: int  579 405 444 289 228 421 645 325 1495 863 ...
#  $ ASV2: int  345 353 362 304 176 277 489 230 1215 729 ...
#  $ ASV3: int  449 231 345 158 204 302 522 254 913 581 ...
#  $ ASV4: int  430 69 502 164 231 357 583 388 1089 853 ...
#  $ ASV5: int  154 140 189 180 130 104 307 179 453 443 ...
```

```
anyNA(ASV_data) # check for missing values
```

```
# [1] FALSE
```

```
colnames(ASV_data[,1:5]) # check if rows or cols are ASVs
```

```
# [1] "ASV1" "ASV2" "ASV3" "ASV4" "ASV5"
```

```
SAM_data <- read.csv("Wk3_sampledata.csv", header=TRUE, row.names=1, stringsAsFactors = TRUE) #add raw
str(SAM_data) # check for data frame and that treatment columns are factors rather than integers for de
```

```
# 'data.frame': 20 obs. of  2 variables:
#  $ Trt1: Factor w/ 3 levels "a","b","c": 1 1 2 2 2 2 2 2 2 2 ...
#  $ Trt2: Factor w/ 3 levels "x","y","z": 1 2 1 2 1 2 1 2 1 2 ...
```

Create Phyloseq object

Phyloseq objects group the following data:

- ASV table (phyloseq::otu_table), numeric matrix, specify if taxa are rows or cols
- sample data (phyloseq::sample_data), data.frame, rownames must match sample names in otu_table
- taxonomy table (phyloseq::tax_table), character matrix, rownames must match taxa_names in otu_table
- reference sequences (phyloseq::refseq), ASV-sequence key, DNAStringSet class
- phylogenetic tree (phyloseq::phy_tree), phylo class

We will create a simple phyloseq object with ASV and sample data

```
ASV=phyloseq::otu_table(as.matrix(ASV_data), taxa_are_rows = FALSE)
SAM=phyloseq::sample_data(SAM_data)

ps <- phyloseq::phyloseq(ASV, SAM)
ps
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 232 taxa and 20 samples ]
# sample_data() Sample Data:       [ 20 samples by 2 sample variables ]
```

```
# you can also add other files, such as reference sequences and taxa names
# Tax_data <- read.csv("...", row.names=1, header=TRUE, sep=",")
# TAX <- tax_table(as.matrix(Tax_data)) #must be a matrix
# Ref_data <- Biostrings::readDNAStringSet("B:/.../refseqs.fasta")
# REF <- refseq(Ref_data)
# ps <- phyloseq(OTU, SAM, TAX, REF)
```

Subset, filter, and summarize a Phyloseq object

One of the major advantages of using phyloseq is that you can filter, subset, and merge samples or taxa simultaneously across all files in the phyloseq object. These rely on quantitative or logical functions. We'll explore the following today:

- `phyloseq::subset_samples`
- `phyloseq::subset_taxa`
- `phyloseq::prune_samples`
- `phyloseq::prune_taxa`
- `phyloseq::merge_samples`
- `phyloseq::merge_taxa`

Subset samples to remove "mock" community using sample names

```
ps <- phyloseq::subset_samples(ps, sample_names(ps) != "Mock")
ps #check that sample number is reduced from 20 to 19
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 232 taxa and 19 samples ]
# sample_data() Sample Data:       [ 19 samples by 2 sample variables ]
```

Subset samples by treatments into a new ps object

```
ps_Trt1a <- phyloseq::subset_samples(ps, Trt1%in%c("a"))
ps_Trt1a
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 232 taxa and 9 samples ]
# sample_data() Sample Data:       [ 9 samples by 2 sample variables ]
```

Retrieve individual elements from the ps object

```
myASV <- phyloseq::otu_table(ps)
myEnv <- phyloseq::sample_data(ps)
# these can be exported with write.csv
```

Summarize contents of ps object to examine the data

```
phyloseq::ntaxa(ps) # number of ASVs
```

```
# [1] 232
```

```
phyloseq::sample_names(ps) # what would you do to save the data that result from this command?
```

```
#  [1] "F3D0"   "F3D1"   "F3D141" "F3D142" "F3D143" "F3D144" "F3D145" "F3D146"
#  [9] "F3D147" "F3D148" "F3D149" "F3D150" "F3D2"   "F3D3"   "F3D5"   "F3D6"
# [17] "F3D7"   "F3D8"   "F3D9"
```

```
phyloseq::sample_sums(ps) # sum of ASV counts for each sample
```

```
#   F3D0   F3D1 F3D141 F3D142 F3D143 F3D144 F3D145 F3D146 F3D147 F3D148 F3D149
#   6528   5017   4863   2521   2518   3488   5820   3879  13006   9935  10653
# F3D150   F3D2   F3D3   F3D5   F3D6   F3D7   F3D8   F3D9
#   4240  16835   5491   3716   6679   4217   4547   6015
```

```
median(phyloseq::sample_sums(ps))
```

```
# [1] 5017
```

```
mean(phyloseq::sample_sums(ps))
```

```
# [1] 6314.105
```

```
# other useful summary functions in the microbiome package
microbiome::summarize_phyloseq(ps)
```

```
# [[1]]
# [1] "1] Min. number of reads = 2518"
#
# [[2]]
# [1] "2] Max. number of reads = 16835"
#
# [[3]]
# [1] "3] Total number of reads = 119968"
#
# [[4]]
# [1] "4] Average number of reads = 6314.1052631579"
#
# [[5]]
# [1] "5] Median number of reads = 5017"
#
# [[6]]
# [1] "7] Sparsity = 0.632259528130672"
#
# [[7]]
# [1] "6] Any OTU sum to 1 or less? YES"
#
# [[8]]
# [1] "8] Number of singletons = 14"
#
# [[9]]
# [1] "9] Percent of OTUs that are singletons \n        (i.e. exactly one read detected across all samp
```

```
#
# [[10]]
# [1] "10] Number of sample variables are: 2"
#
# [[11]]
# [1] "Trt1" "Trt2"
```

```
microbiome::top_taxa(ps, n=10) #ASVs by rank abundance, # of ranks defined by n=x
```

```
#  [1] "ASV1"  "ASV2"  "ASV3"  "ASV4"  "ASV5"  "ASV6"  "ASV7"  "ASV8"  "ASV9"
# [10] "ASV10"
```

Data transformations

Compositional data: centered log ratio (CLR)

Run transformation on ASV matrix

```
ASV_for_clr <- phyloseq::otu_table(ps) # get ASV table from ps object
ASV_clr <- compositions::clr(ASV_for_clr) # transform

# if R assigns this rmult; change to dataframe
ASV_clr <- as.data.frame(ASV_clr)
```

Make a copy of original ps object and replace otu_table with transformed table

```
ps_clr <- ps
phyloseq::otu_table(ps_clr) <- phyloseq::otu_table(ASV_clr, taxa_are_rows = FALSE)
ps_clr
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 232 taxa and 19 samples ]
# sample_data() Sample Data:       [ 19 samples by 2 sample variables ]
```

```
# good practice to check that taxa names match
# phyloseq::taxa_names(ps_clr)
# phyloseq::taxa_names(ps)

# can also save transformed ASV files as .csv
# write.csv(ASV_clr, "/.../ASV_clr_transformed.csv")
```

Variance stabilizing transformation in DESeq

Examine sample data and experimental design

```
phyloseq::sample_data(ps)
```

```
#        Trt1 Trt2
# F3D0      a    x
# F3D1      a    y
# F3D141    b    x
# F3D142    b    y
# F3D143    b    x
# F3D144    b    y
# F3D145    b    x
# F3D146    b    y
# F3D147    b    x
# F3D148    b    y
# F3D149    b    x
# F3D150    b    y
# F3D2      a    x
```

```
# F3D3      a    y
# F3D5      a    x
# F3D6      a    y
# F3D7      a    x
# F3D8      a    y
# F3D9      a    x
```

Convert phyloseq to DeSeq object

- after ~, define experimental design using columns in sample data file

- alt - use ~1 as the experimental design if you don't want the actual design to influence transformation
- can also code an interaction term if needed "+ Factor1:Factor2"

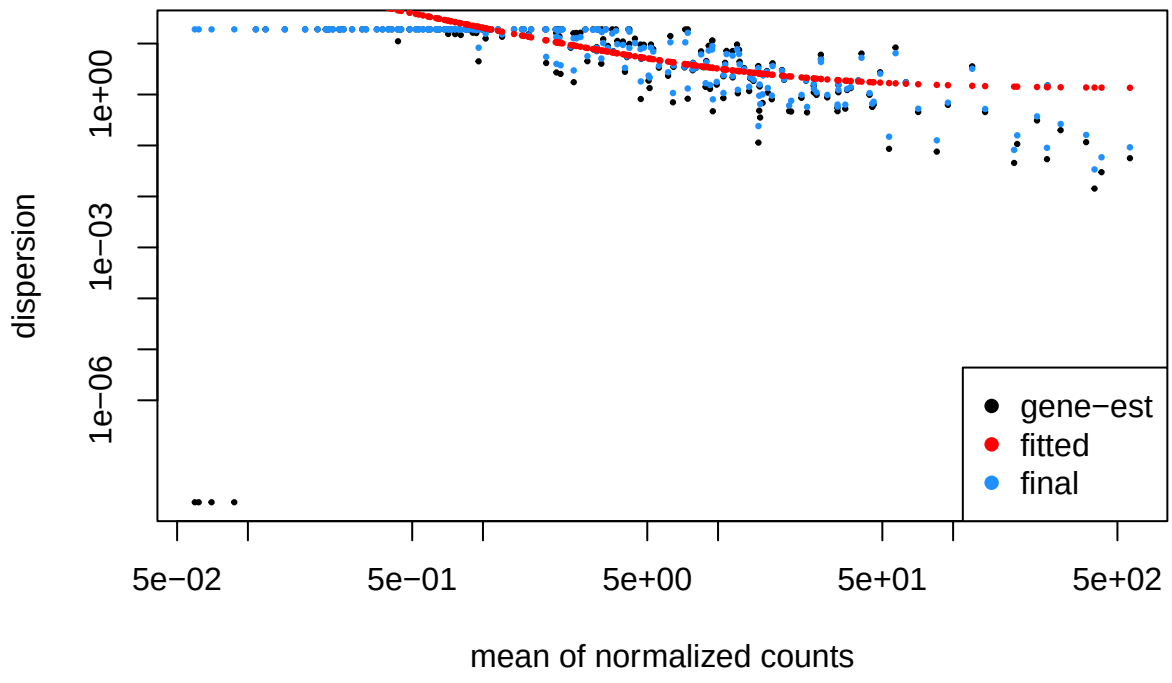
```
ps_ds <- phyloseq::phyloseq_to_deseq2(ps, ~Trt1+Trt2)
```

Estimate size factors and dispersion

```
ps_ds = DESeq2::estimateSizeFactors(ps_ds)
ps_ds = DESeq2::estimateDispersions(ps_ds, fitType = "parametric")
# other fit types are "local" and "mean"
# local is substituted if parametric doesn't fit observed relationship
```

Plot the dispersion estimates

```
DESeq2::plotDispEsts(ps_ds)
```



```
# if you get the error "Error in estimateSizeFactorsForMatrix(counts(object), locfunc = locfunc, : every
# then use the following code instead
# note that for estimateSizeFactors, need to use
# gm_mean = function(x, na.rm=TRUE){ exp(sum(log(x[x > 0]), na.rm=na.rm) / length(x))}
# geoMeans = apply(counts(ps_ds), 1, gm_mean)
# ps_ds = estimateSizeFactors(ps_ds, type="ratio", geoMeans = geoMeans) #alt types are "poscounts" and
# ps_ds = estimateDispersions(ps_ds, fitType = "local")
# ps_ds = DESeq(ps_ds, test="Wald", fitType="parametric")
```

Copy the ps object and replace ASVs with transformed data

```
# make a copy of original ps object, run the variance stabilization, and replace otu_table with the tra
# then save transformed files as .csv
ps_vst <- ps
vst<-DESeq2::getVarianceStabilizedData(ps_ds)
phyloseq::otu_table(ps_vst) <- phyloseq::otu_table(vst, taxa_are_rows = TRUE)
ps_vst
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 232 taxa and 19 samples ]
# sample_data() Sample Data:       [ 19 samples by 2 sample variables ]
```

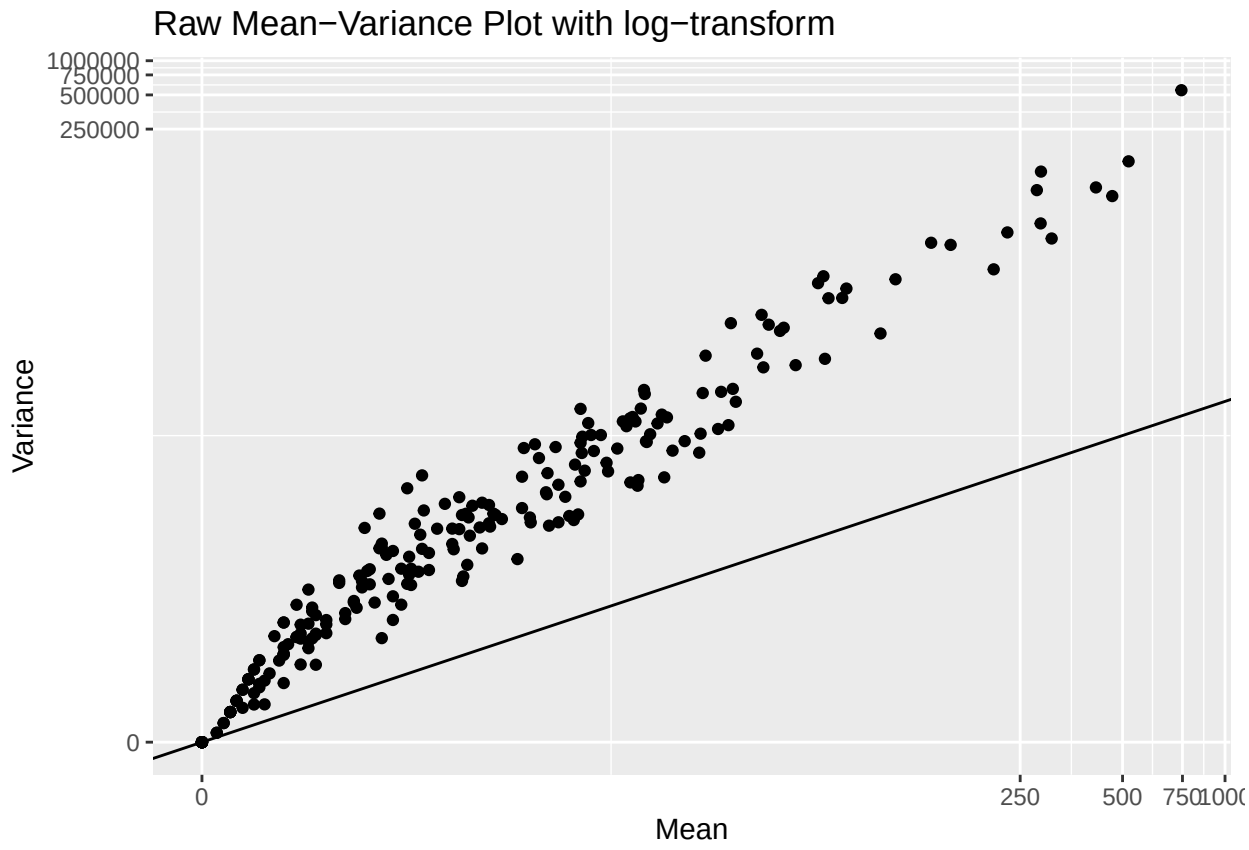
```
# good practice to check that taxa names match
# taxa_names(ps_vst)
# taxa_names(ps)

# can also save your vst-transformed data as a csv file
# ASV_vst <-otu_table(ps_vst)
# ASV_vst_t <- t(ASV_vst) #matrix has taxa as rows, transpose to flip
# write.csv(ASV_vst_t, "/.../ASVs_vst_transformed.csv")
```

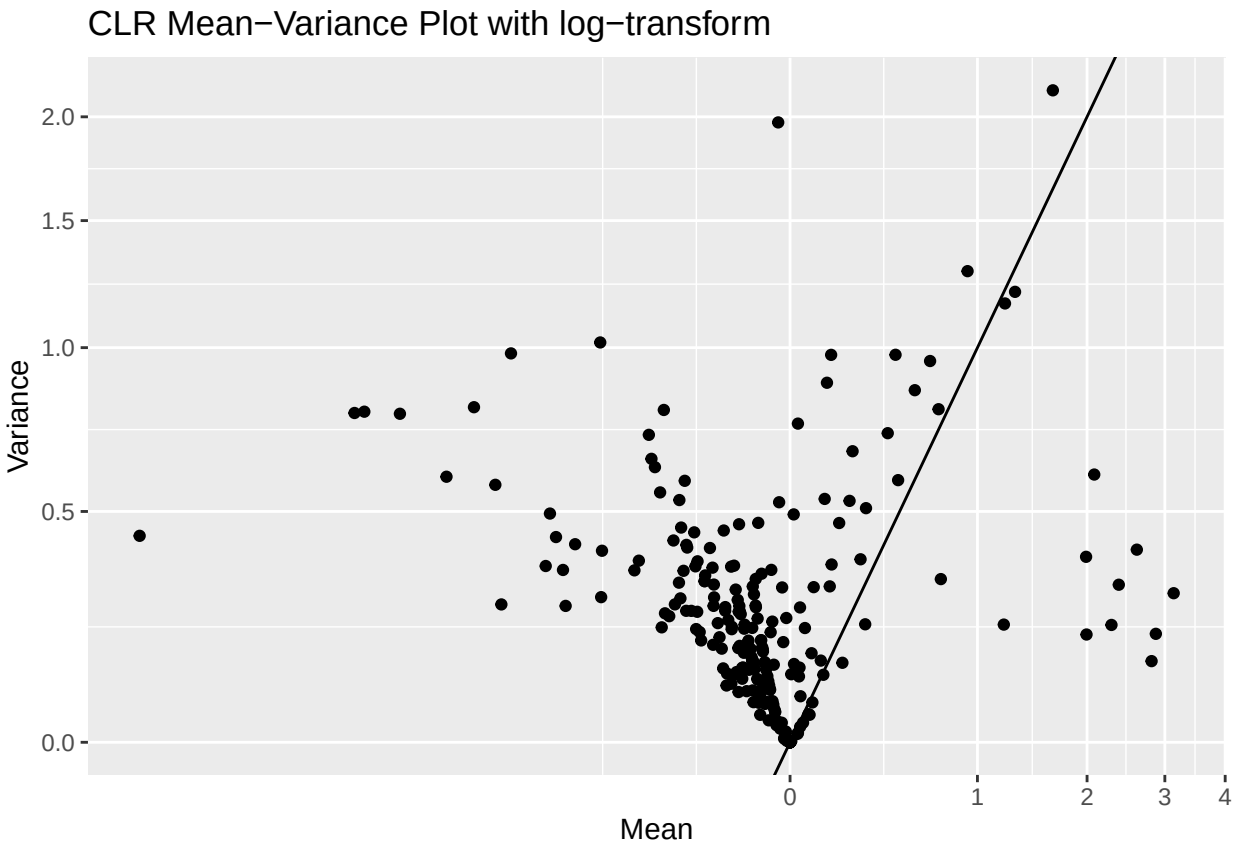
Compare abundance distributions of raw, clr, and vst transforms

Check for overdispersion

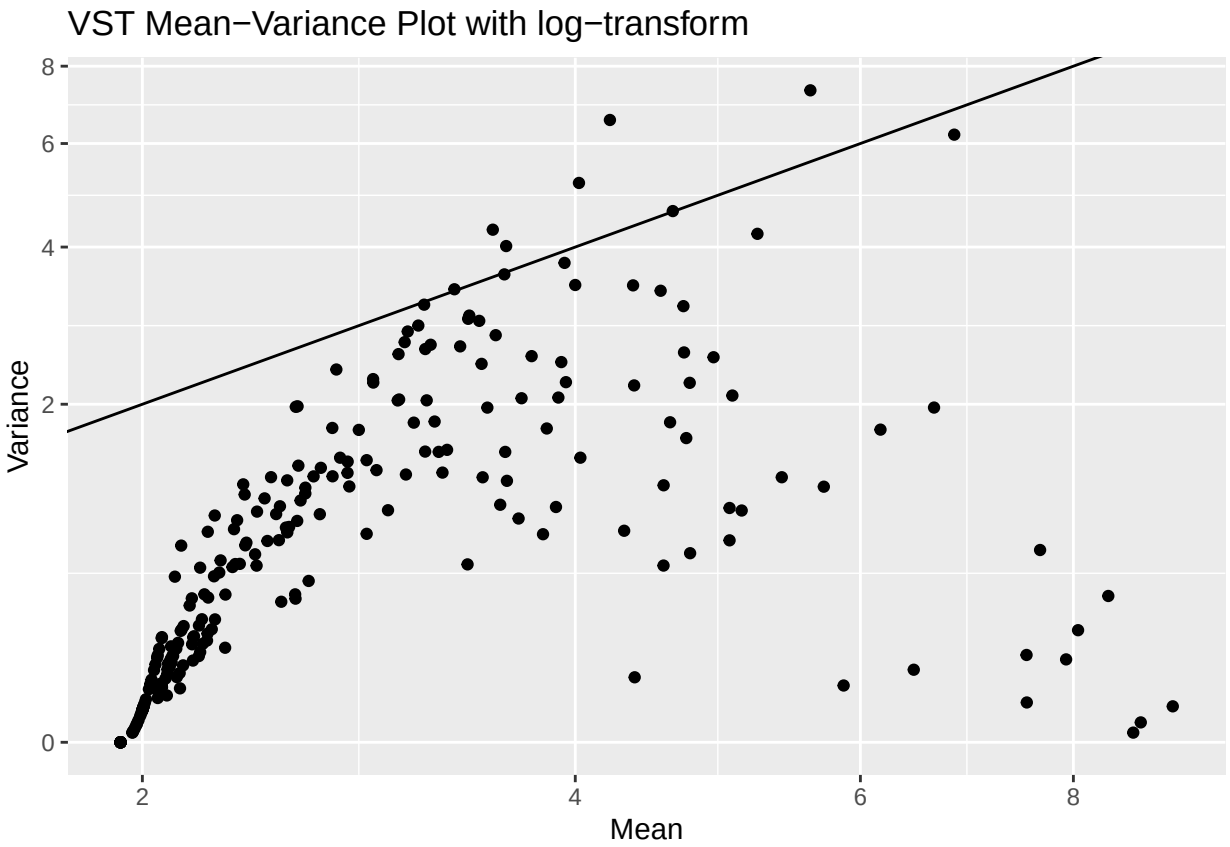
```
(mv_raw<-psadd::plot_mv(ps, title = "Raw Mean-Variance Plot with log-transform"))
```




```
(mv_clr<-psadd::plot_mv(ps_clr, title = "CLR Mean-Variance Plot with log-transform"))
```

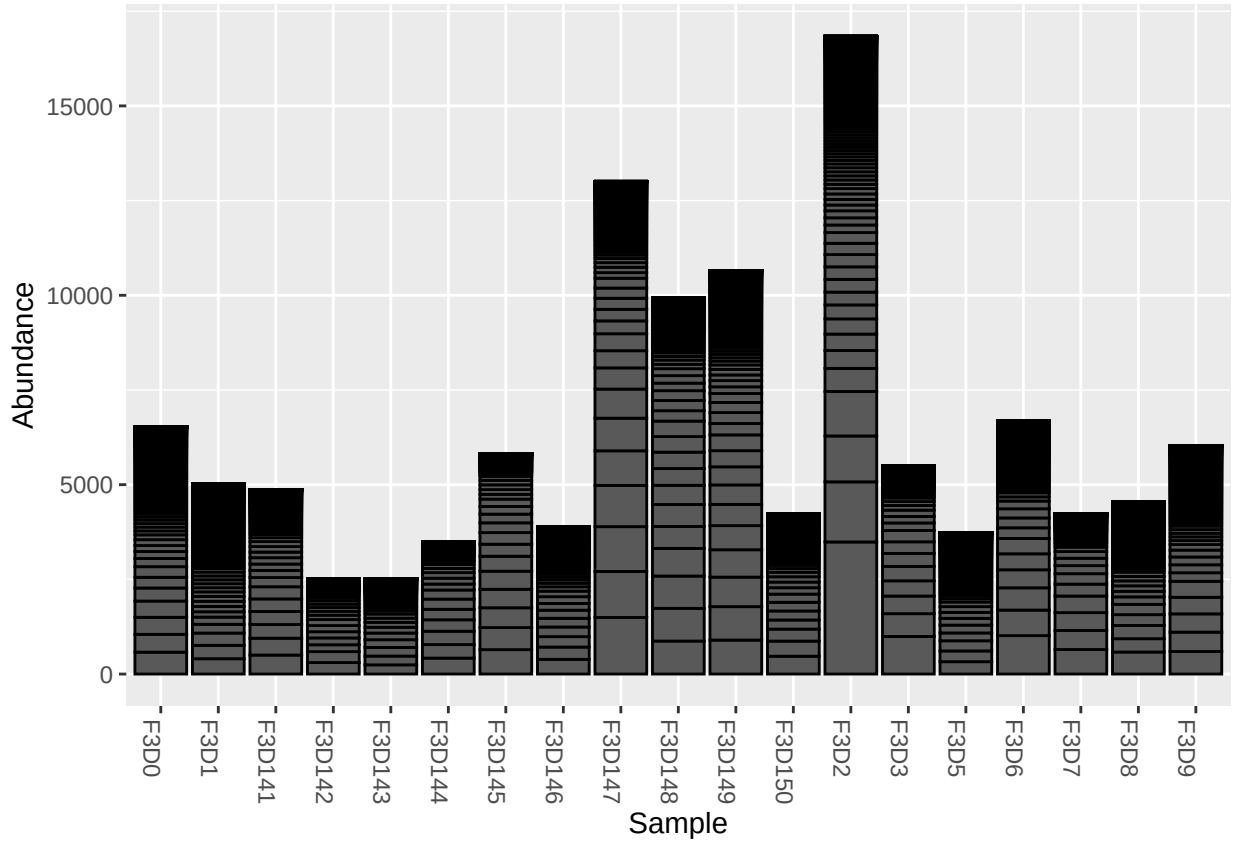



```
(mv_vst<-psadd::plot_mv(ps_vst, title = "VST Mean-Variance Plot with log-transform"))
```

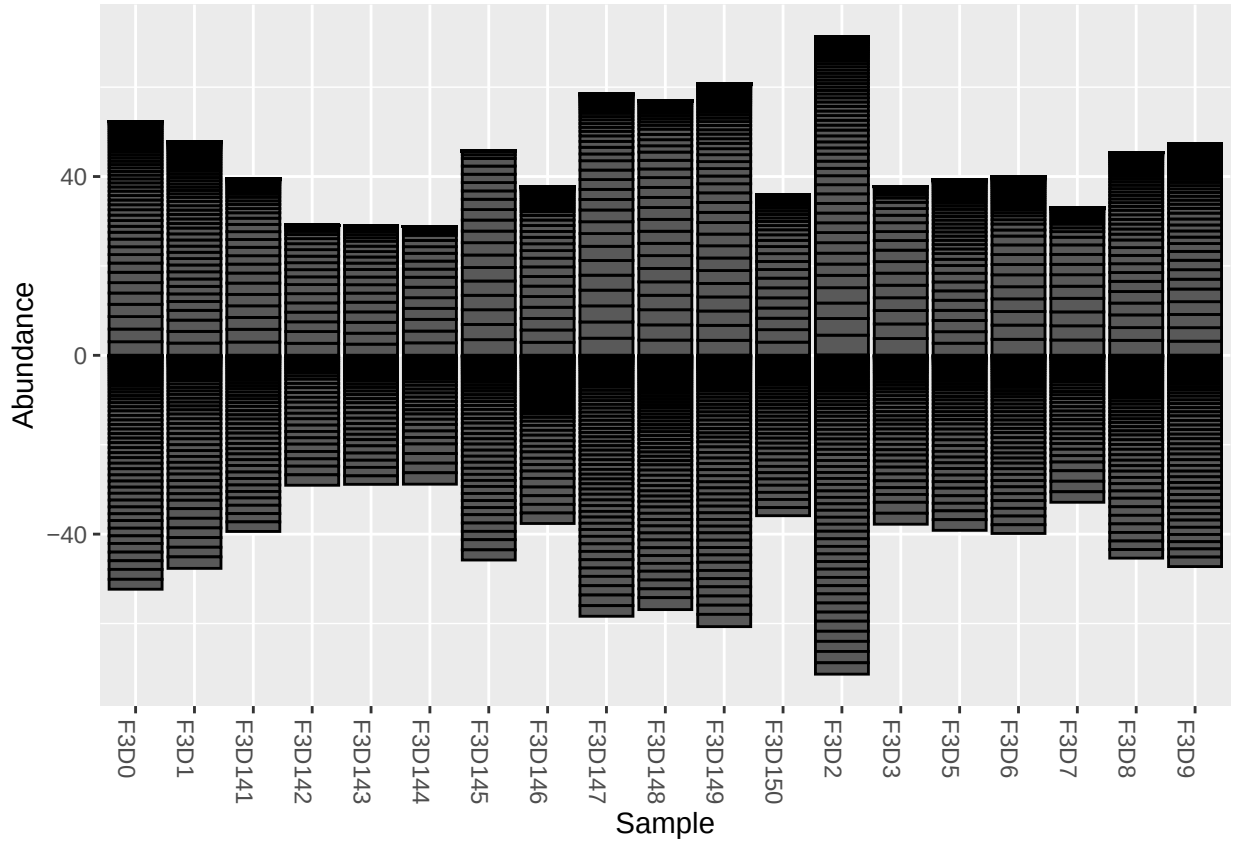

VST Mean−Variance Plot with log−transform
Variance
Mean
0
2
4
6
8
2
4
6
8

Plot ASV abundances by sample

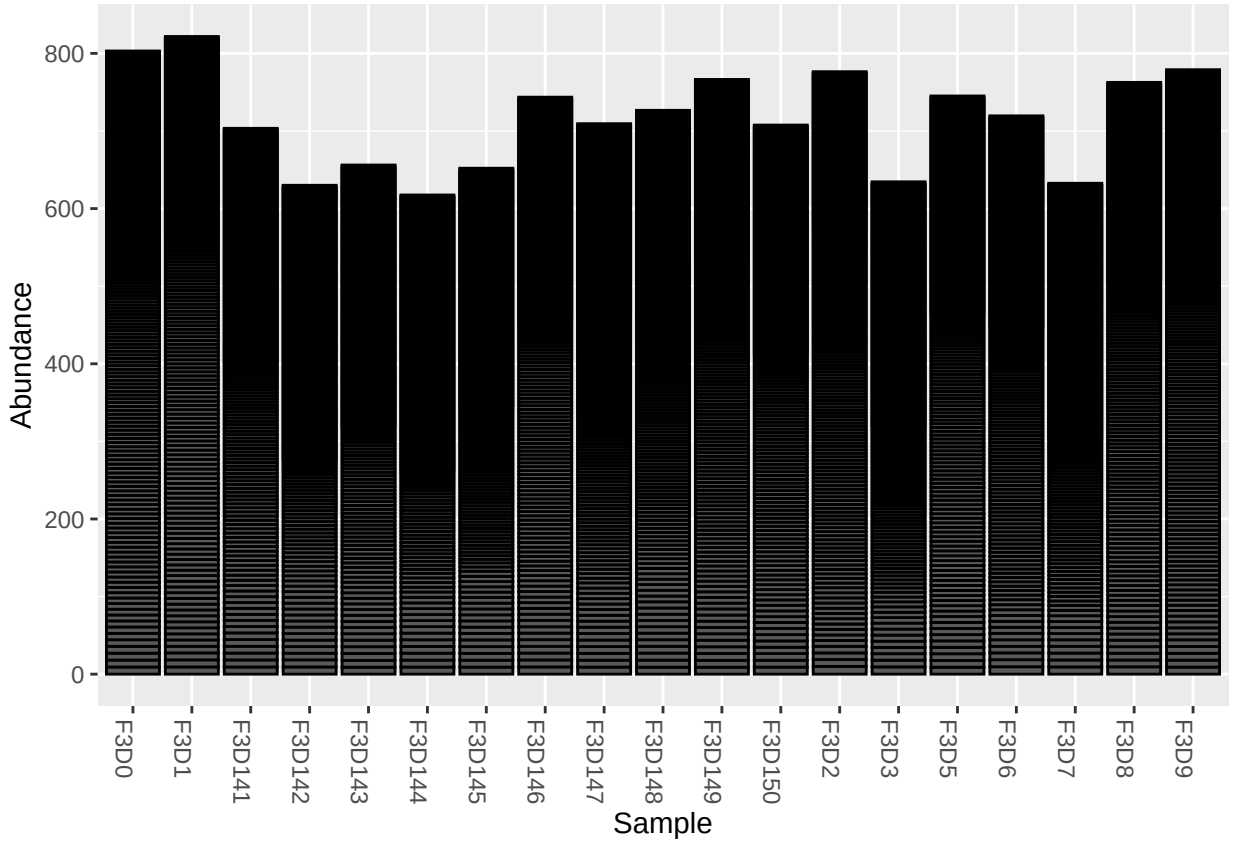
```
phyloseq::plot_bar(ps)
```




```
phyloseq::plot_bar(ps_clr)
```

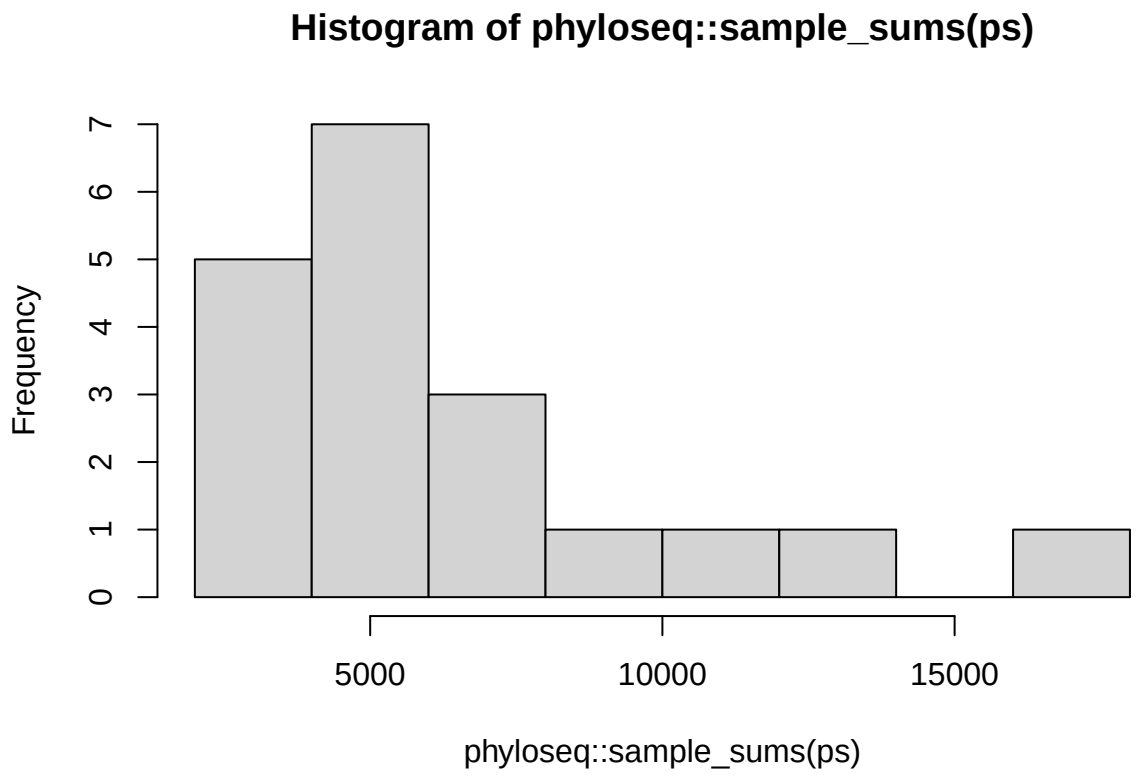



```
phyloseq::plot_bar(ps_vst)
```

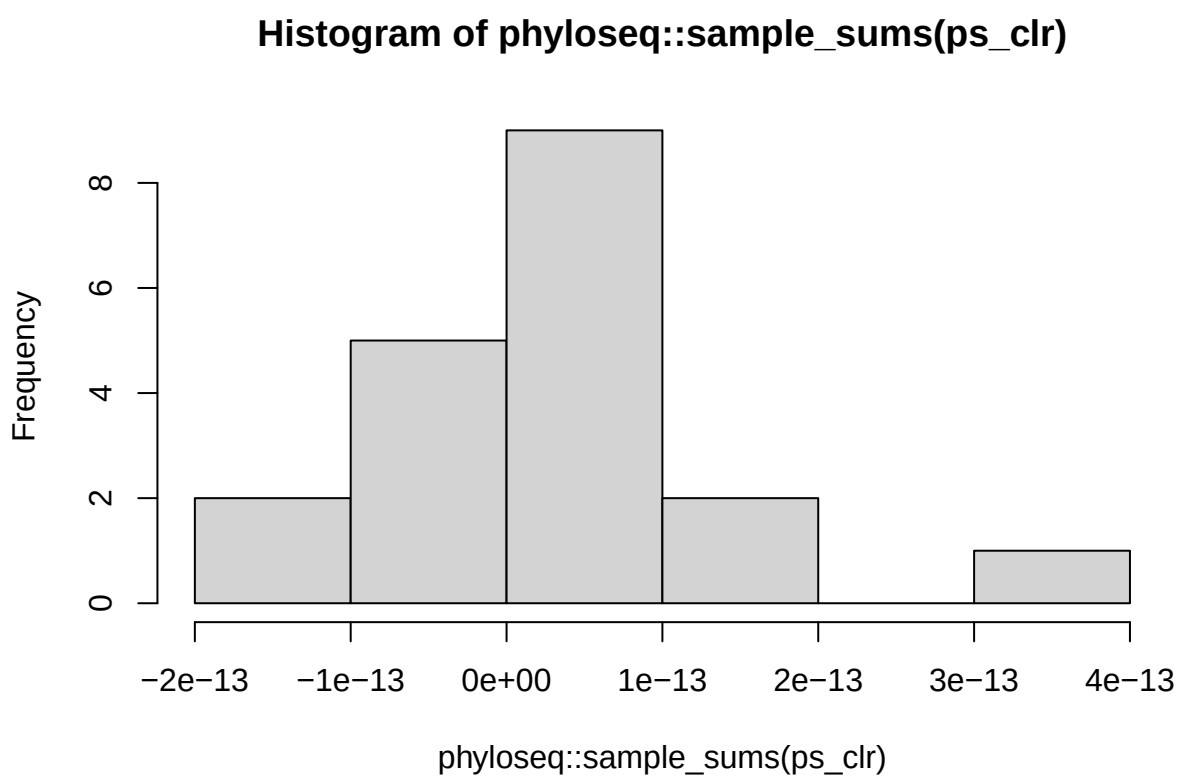

Abundance
800
600
400
200
0
F3D0
F3D1
F3D141
F3D142
F3D143
F3D144
F3D145
F3D146
F3D147
F3D148
F3D149
F3D150
F3D2
F3D3
F3D5
F3D6
F3D7
F3D8
F3D9
Sample

Plot histograms of sample sums

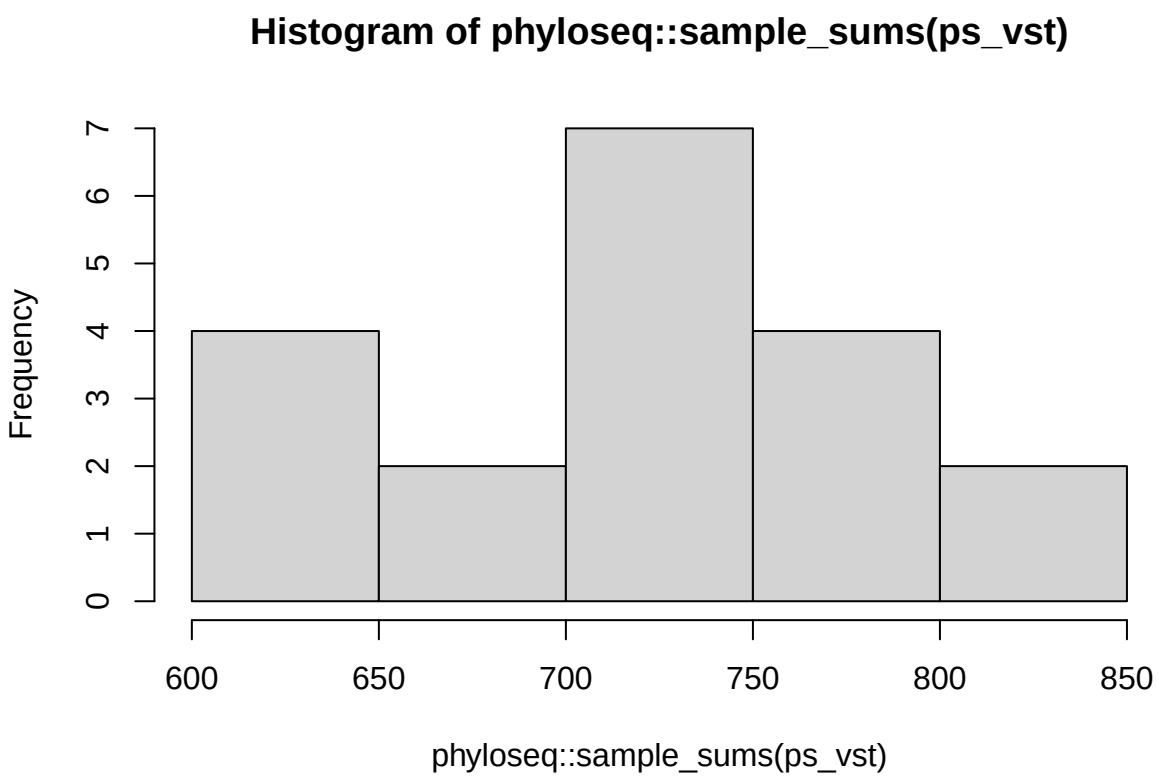
```
hist(phyloseq::sample_sums(ps))
```




```
hist(phyloseq::sample_sums(ps_clr))
```




```
hist(phyloseq::sample_sums(ps_vst))
```


Histogram of phyloseq::sample_sums(ps_vst)
Frequency
0
1
2
3
4
5
6
7
600
650
700
750
800
850
phyloseq::sample_sums(ps_vst)

Coding exercises

Please work in R markdown and knit to an html or pdf. Exercises 1-3 use the simple ps object we created in class. Exercises 4-5 use data modified from Wagner et al. Include comments to address questions below as needed.

- Don't forget to commit and push your work to your GitHub repo
- If you don't finish these in class, complete as homework
- Knitted filename should be Wk3_yourlastname.html or .pdf
- Push to GitHub by next Wed

1. Use the transform_sample_counts() function in phyloseq for custom data transformation
Examples with relative abundance and log2 - choose something different!

```
## relative (proportional) abundance  - example with function definition:
    ps_ra <- phyloseq::transform_sample_counts(ps, function(x){x / sum(x)})

## log2 - no function definition
    ps_log2 <- phyloseq::transform_sample_counts(ps, log2)
```

2. Some transformations result in negative values that make other analyses difficult

- Use the clr-transformed ps object to add a constant to each cell to obtain non-negatives
- Base your constant on the minimum value in the file
- Check that your transform worked

3. Rerun the DeSeq2 dispersion model with local and mean fits and compare to parametric fit dispersion plots - which is better?

- outliers are flagged as larger blue circles around the black dots (default outlier = 2SD from mean fitted value)
- outliers keep original values and are not shrunk by the transform

4. Practice creating a ps object

- We will use data from Wagner et al. 2016 Nature Communications https://www.nature.com/articles/ncomms12151#Sec22
- I reduced these data to a more manageable size by:
  - removing unidentified taxa `phyloseq::subset_taxa`
  - limiting to root samples in 2011 `phyloseq::subset_samples`

  - removing taxa with less than 20 reads `phyloseq::prune_taxa`
- Open the Wk3_Wagner_sm_XXX_data.csv files from the ClassData repo, where XXX = ASV, SAM, and TAX
  - be sure to specify `row.names=1` for each file
- Merge into ps object called ps_wag

5. Practice subsetting and filtering data

Use the ps_wag object to:

- subset by a single genotype or site in the SAM data with `phyloseq::subset_samples`
  - https://rdrr.io/bioc/phyloseq/man/subset_samples-methods.html
- subset by phylum in the TAX data with `phyloseq::subset_taxa` and use `phyloseq::get_taxa_unique` to select
  - https://rdrr.io/bioc/phyloseq/man/subset_taxa-methods.html
  - https://www.rdocumentation.org/packages/phyloseq/versions/1.16.2/topics/get_taxa_unique
- subset to only very abundant taxa (e.g., >1000 reads) with `phyloseq::prune_taxa`
  - https://rdrr.io/bioc/phyloseq/man/prune_taxa-methods.html

Session Info

```
sessionInfo()
```

```
# R version 4.1.2 (2021-11-01)
# Platform: x86_64-w64-mingw32/x64 (64-bit)
# Running under: Windows 10 x64 (build 19042)
#
# Matrix products: default
#
# locale:
# [1] LC_COLLATE=English_United States.1252
# [2] LC_CTYPE=English_United States.1252
# [3] LC_MONETARY=English_United States.1252
# [4] LC_NUMERIC=C
# [5] LC_TIME=English_United States.1252
#
# attached base packages:
# [1] stats     graphics  grDevices utils     datasets  methods   base
#
# loaded via a namespace (and not attached):
#   [1] nlme_3.1-155               bitops_1.0-7
#   [3] matrixStats_0.61.0         phyloseq_1.38.0
#   [5] bit64_4.0.5                psadd_0.1.3
#   [7] RColorBrewer_1.1-2         httr_1.4.2
#   [9] GenomeInfoDb_1.30.0        tensorA_0.36.2
#  [11] tools_4.1.2                utf8_1.2.2
#  [13] R6_2.5.1                   vegan_2.5-7
#  [15] DBI_1.1.2                  BiocGenerics_0.40.0
#  [17] mgcv_1.8-38                colorspace_2.0-2
#  [19] permute_0.9-5              rhdf5filters_1.6.0
#  [21] ade4_1.7-18                tidyselect_1.1.1
#  [23] DESeq2_1.34.0              bit_4.0.4
#  [25] bayesm_3.1-4               compiler_4.1.2
#  [27] compositions_2.0-4         microbiome_1.16.0
#  [29] Biobase_2.54.0             DelayedArray_0.20.0
#  [31] labeling_0.4.2             scales_1.1.1
#  [33] DEoptimR_1.0-10            robustbase_0.93-9
#  [35] genefilter_1.76.0          stringr_1.4.0
#  [37] digest_0.6.29              rmarkdown_2.11
#  [39] XVector_0.34.0             pkgconfig_2.0.3
#  [41] htmltools_0.5.2            plotrix_3.8-2
#  [43] MatrixGenerics_1.6.0       highr_0.9
#  [45] fastmap_1.1.0              rlang_0.4.12
#  [47] RSQLite_2.2.9              farver_2.1.0
#  [49] generics_0.1.1             jsonlite_1.7.3
#  [51] BiocParallel_1.28.3        dplyr_1.0.7
#  [53] RCurl_1.98-1.5             magrittr_2.0.1
#  [55] GenomeInfoDbData_1.2.7     biomformat_1.22.0
#  [57] Matrix_1.4-0               Rcpp_1.0.8
#  [59] munsell_0.5.0              S4Vectors_0.32.3
#  [61] Rhdf5lib_1.16.0            fansi_0.5.0
#  [63] ape_5.6-1                  lifecycle_1.0.1
```

```
#  [65] stringi_1.7.6              yaml_2.2.1
#  [67] MASS_7.3-54                SummarizedExperiment_1.24.0
#  [69] zlibbioc_1.40.0            rhdf5_2.38.0
#  [71] Rtsne_0.15                 plyr_1.8.6
#  [73] blob_1.2.2                 grid_4.1.2
#  [75] parallel_4.1.2             crayon_1.4.2
#  [77] lattice_0.20-45            Biostrings_2.62.0
#  [79] splines_4.1.2              multtest_2.50.0
#  [81] annotate_1.72.0            KEGGREST_1.34.0
#  [83] locfit_1.5-9.4             knitr_1.37
#  [85] pillar_1.6.4               igraph_1.2.11
#  [87] GenomicRanges_1.46.1       geneplotter_1.72.0
#  [89] reshape2_1.4.4             codetools_0.2-18
#  [91] stats4_4.1.2               XML_3.99-0.8
#  [93] glue_1.6.0                 evaluate_0.14
#  [95] data.table_1.14.2          png_0.1-7
#  [97] vctrs_0.3.8                foreach_1.5.1
#  [99] gtable_0.3.0               purrr_0.3.4
# [101] tidyr_1.1.4                assertthat_0.2.1
# [103] cachem_1.0.6               ggplot2_3.3.5
# [105] xfun_0.29                  xtable_1.8-4
# [107] survival_3.2-13            tibble_3.1.6
# [109] iterators_1.0.13           memoise_2.0.1
# [111] AnnotationDbi_1.56.2       IRanges_2.28.0
# [113] cluster_2.1.2              ellipsis_0.3.2
```