Practical Approaches to Bioanalysis in R

Day 4 – Bioconductor packages

Extending R through packages:
There's a package for everything

Bio-specific R packages are available on Bioconductor

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home | Install | Help | Developers | About

About *Bioconductor*

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. *Bioconductor* uses the R statistical programming language, and is open source and open development. It has two releases each year, and an active user community. *Bioconductor* is also available as an AMI (Amazon Machine Image) and Docker images.

News

- See our google calendar for events, conferences, meetings, forums, etc. Add your event with email to events at bioconductor.org.
- *Bioconductor* 3.11 is available.
- Nominate an outstanding community member for a *Bioconductor* Award! See the support site for more information.
- Registration open for BioC2020.
- Core team **job opportunities** available, contact Martin.Morgan at RoswellPark.org
- *Bioconductor* F1000 Research Channel is

BioC 2020

Get the latest updates on the BioC 2020 Conference!

- BioC 2020 is going virtual July 27 - July 31. Please see the Registration Page for more information.
- Nominate an outstanding *Bioconductor* community member for a *Bioconductor* Award! See posting for more information.
- Call for birds-of-feather, hack-a-thon, and how-to sections. Please see posting for more information.
- Registration is now open. Register today.

Install »

- Discover 1903 software packages available in *Bioconductor* release 3.11.

Get started with *Bioconductor*

- Install *Bioconductor*
- Get support
- Latest newsletter
- Follow us on twitter
- Install R

Learn »

Master *Bioconductor* tools

- Courses
- Support site
- Package vignettes
- Literature citations
- Common work flows
- FAQ
- Community resources
- Videos

Common Bioconductor Methods/Classes

Class – Blueprint for an object. If I say "data.frame", what comes to mind?
Method – Function that "reacts" to class of input. as.data.frame( ) is a simple example.

Importing

- GTF, GFF, BED, BigWig, etc., – rtracklayer::import()
- VCF – VariantAnnotation::readVcf()
- SAM / BAM – Rsamtools::scanBam(), GenomicAlignments::readGAlignment*()
- FASTA – Biostrings::readDNAStringSet()
- FASTQ – ShortRead::readFastq()
- MS data (XML-based and mgf formats) – MSnbase::readMSData(), MSnbase::readMgfData()

Common Classes

- Rectangular feature x sample data – SummarizedExperiment::SummarizedExperiment() (RNAseq count matrix, microarray, …)
- Genomic coordinates – GenomicRanges::GRanges() (1-based, closed interval)
- DNA / RNA / AA sequences – Biostrings::*StringSet()
- Gene sets – GSEABase::GeneSet() GSEABase::GeneSetCollection()
- Multi-omics data – MultiAssayExperiment::MultiAssayExperiment()
- Single cell data – SingleCellExperiment::SingleCellExperiment()
- Mass spec data – MSnbase::MSnExp()

https://bioconductor.org/developers/how-to/commonMethodsAndClasses/

You can install Bioconductor packages using `BiocManager::install()` in RStudio

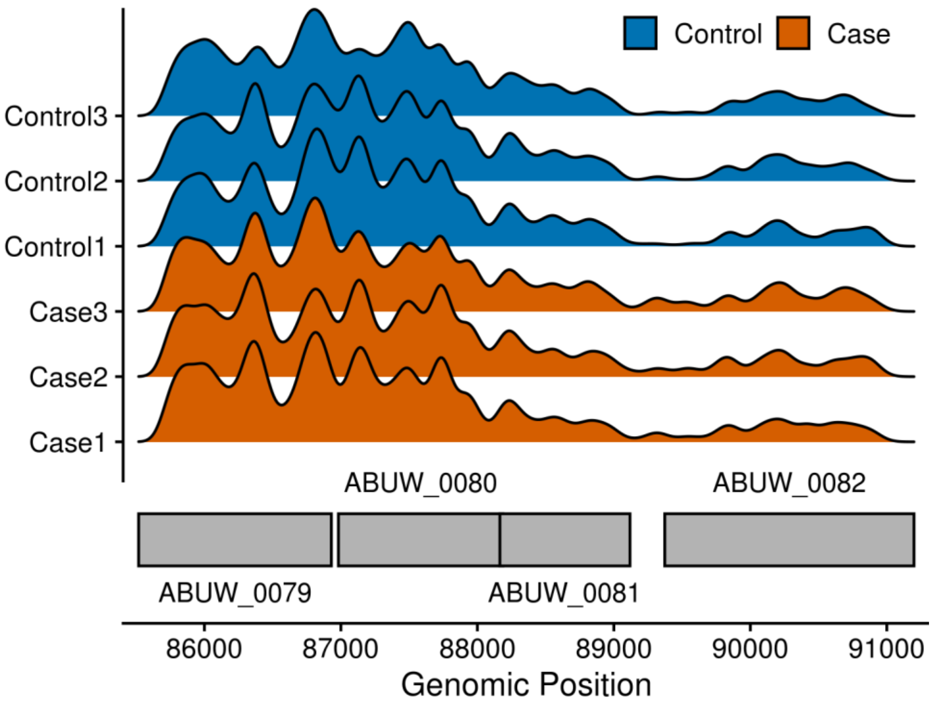
Run this once

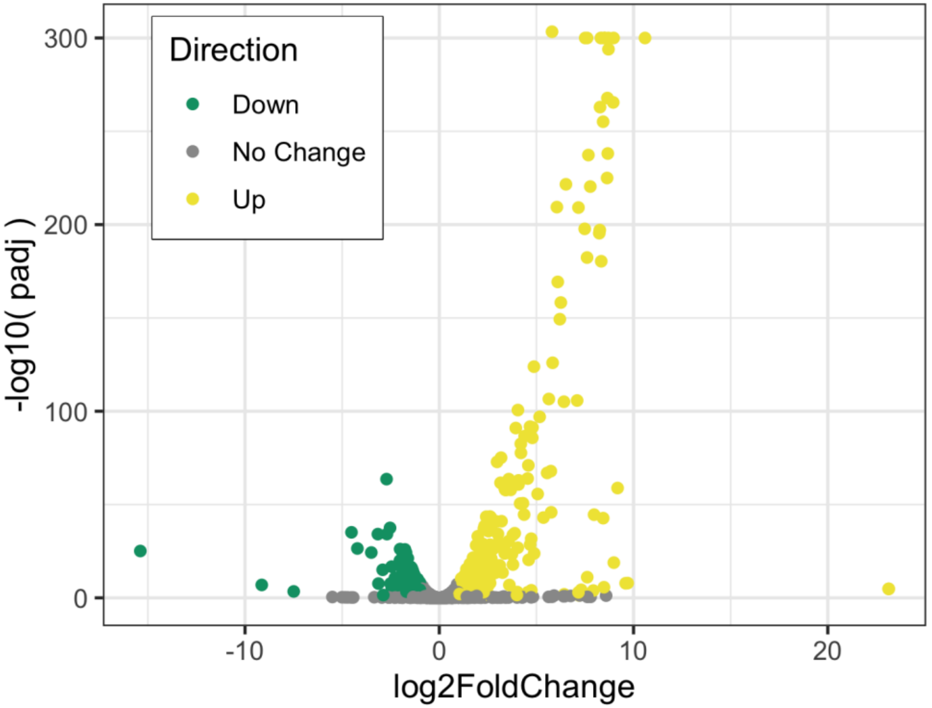
```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("DESeq2")
```

DESeq2: Differential Expression of Sequencing data

DESeq2: Differential Expression of Sequencing data

Go from transcript abundances to normalized log2 fold changes (with p-values).





Following slides adapted from DESeq2 vignette:
http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

DESeq2: Minimum example

RNA-Seq Analysis Identifies New Genes Regulated by the Histone-Like Nucleoid Structuring Protein (H-NS) Affecting *Vibrio cholerae* Virulence, Stress Response and Chemotaxis

Hongxia Wang, Julio C. Ayala, Jorge A. Benitez, Anisia J. Silva

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118295

```
dds <- DESeqDataSetFromMatrix( countData = cts,
                                colData = coldata,
                                design= ~ batch + condition)
dds <- DESeq(dds)
resultsNames(dds) # lists the coefficients
res <- results(dds, name="condition_trt_vs_untrt")
```

DESeq2: Minimum example

Matrix with tx count data

GLM defining variables of interest.
NOTE: For more experiments,
ask a statistician for help here

```
dds <- DESeqDataSetFromMatrix( countData = cts,
                               colData = coldata,
                               design= ~ batch + condition)
dds <- DESeq(dds)
resultsNames(dds) # lists the coefficients
res <- results(dds, name="condition_trt_vs_untrt")
```

data.frame with sample info

Object containing data.frame with log2FC and stats

colData	4 obs. of 3 variables
cts	num [1:4427, 1:4] 1441 3004 9734 380 3492 ...
dds	Large DESeqDataSet (4427 elements, 3.6 Mb)
res	Large DESeqResults (6 elements, 587.8 Kb)

An aside on r data types

```
dds                 Large DESeqDataSet (4427 elements, 3.6 Mb)
```

```
dds                 Large DESeqDataSet (4427 elements, 3.6 Mb)
  ..@ design :Class 'formula' language ~phenotype
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  ..@ dispersionFunction:function (q)
  .. ..- attr(*, "coefficients")= Named num [1:2] 0.0245 10.8751
  .. .. ..- attr(*, "names")= chr [1:2] "asymptDisp" "extraPois"
  .. ..- attr(*, "fitType")= chr "parametric"
  .. ..- attr(*, "varLogDispEsts")= num 1.83
  .. ..- attr(*, "dispPriorVar")= num 0.777
  ..@ rowRanges :Formal class 'CompressedGRangesList' [package "GenomicRanges"] with 5 slots
  .. .. ..@ unlistData :Formal class 'GRanges' [package "GenomicRanges"] with 7 slots
  .. .. .. .. ..@ seqnames :Formal class 'Rle' [package "S4Vectors"] with 4 slots
  .. .. .. .. .. .. ..@ values : Factor w/ 0 levels:
  .. .. .. .. .. .. ..@ lengths : int(0)
  .. .. .. .. .. .. ..@ elementMetadata: NULL
  .. .. .. .. .. .. ..@ metadata : list()
  .. .. .. .. ..@ ranges :Formal class 'IRanges' [package "IRanges"] with 6 slots
```

"It's in the data object!"

dds	S4 [4427 x 4] (DESeq2::DESeq	S4 object of class DESeqDataSet
design	formula	~phenotype
dispersionFunction	function	function(q) { ... }
rowRanges	S4 (GenomicRanges::Compres	S4 object of class CompressedGRangesList
unlistData	S4 (GenomicRanges::GRanges	S4 object of class GRanges
elementMetadata	S4 [4427 x 22] (S4Vectors::DF	S4 object of class DFrame
rownames	NULL	Pairlist of length 0
nrows	integer [1]	4427
listData	list [22]	List of length 22
baseMean	double [4427]	1665 3673 11815 410 3211 11135 ...
baseVar	double [4427]	13138 439407 9864711 13807 543332 5011277 ...
allZero	logical [4427]	FALSE FALSE FALSE FALSE FALSE FALSE ...
dispGeneEst	double [4427]	1.84e-03 1.00e-08 1.42e-02 6.48e-02 7.65e-02 6.41e-03 ...
dispGeneIter	double [4427]	9 2 2 2 3 2 ...

```
dds@rowRanges@elementMetadata$baseMean
```

DESeq2: Loading in real data

```
library(tidyverse)
library(tximport) # Bioconductor
library(rhdf5) # Bioconductor
library(DESeq2) # Bioconductor
library(pheatmap)
```

```
kallisto_dir <- "kallisto_results"
kallisto_df <- read_csv("kallisto_results/SraRunTable.csv")
row.names(kallisto_df) <- kallisto_df$Run

# Filter to include only one condition
coldata <- kallisto_df %>%
            filter(od600_nm == 2) %>%
            select(Run, od600_nm, phenotype)

# Put abundance file locations into a list
kallisto_files <- file.path(kallisto_dir, coldata$Run, "abundance.tsv")
kf2 <- file.path(kallisto_dir, kallisto_df$Run, "abundance.tsv")
txi <- tximport(kallisto_files, type="kallisto", txOut = TRUE)
txi2 <- tximport(kf2, type="kallisto", txOut = TRUE)

cts <- txi$counts # This is a count matrix like the one in the example
```

DESeq2: Read normalization & contrasts

```
# Import tx object to DESeq format
dds <- DESeqDataSetFromTximport(txi,
                                colData = coldata,
                                design = ~ phenotype)
dds <- DESeq(dds) # This runs the standard DESeq normalization pipeline
resultsNames(dds) # Show contrast levels
res <- results(dds, contrast = c("phenotype", "HNS.KO", "WT")) # Get log2FC
results_df <- as.data.frame(res)
```

DESeq2: Differential expression

```
view( results_df )
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
VC1130|ID:1735915|hns|	1.024319e+04	-15.390456	1.4180686	-10.8531110	1.927509e-27	7.838538e-26
VCSEN_bncRNA561	4.964988e+01	-9.139446	1.5808870	-5.7812140	7.416347e-09	1.164257e-07
VIBCH10482|ID:1733969|	1.570148e+01	-7.494045	1.8264345	-4.1031008	4.076496e-05	3.635394e-04
VC1854|ID:1736307|ompT|	1.335629e+05	-5.533453	0.1802948	-30.6911446	7.471610e-207	1.329324e-204
VCSEN_ancRNA266	4.001725e+00	-5.502976	3.6938425	-1.4897700	1.362847e-01	3.452647e-01
VCSEN_ancRNA274	2.807449e+00	-5.010165	4.0839363	-1.2267980	2.198985e-01	4.623174e-01
VCSEN_ancRNA71	2.547773e+00	-4.854970	4.2120729	-1.1526319	2.490615e-01	4.992923e-01
VCSEN_ancRNA324	2.296628e+00	-4.714862	4.3528026	-1.0831785	2.787292e-01	5.358729e-01
VCSEN_ancRNA301	2.039622e+00	-4.539884	4.5269179	-1.0028642	3.159264e-01	5.774853e-01
VC1333|ID:1736020|	1.241901e+03	-4.519809	0.3528569	-12.8091853	1.456557e-37	7.584755e-36
VCSEN_bncRNA580	1.891270e+00	-4.431406	4.6511926	-0.9527463	3.407186e-01	6.041813e-01
VC1334|ID:1736021|	3.406184e+03	-4.216178	0.3784948	-11.1393297	8.072463e-29	3.481759e-27

DESeq2: Checking low count data

```
counts_df <- counts(dds) # Make a df with counts for each read
colnames(counts_df) <- coldata$Run # Fix column names
results_df <- cbind(counts_df, results_df)# Bind counts and results dfs
```

DESeq2: Checking low count data

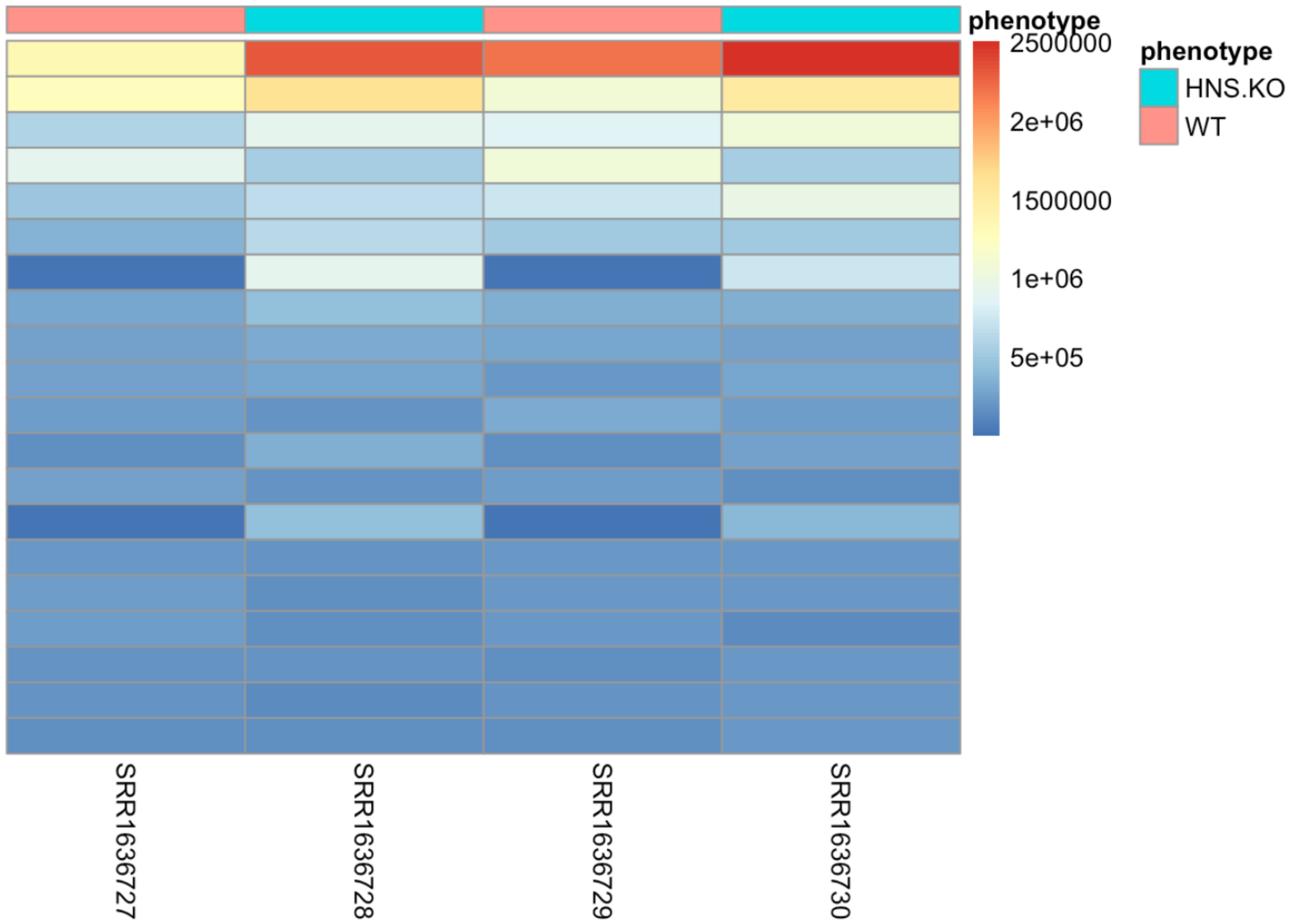
```
counts_df <- counts(dds) # Make a df with counts for each read
colnames(counts_df) <- coldata$Run # Fix column names
results_df <- cbind(counts_df, results_df)# Bind counts and results dfs
```

	SRR1636727	SRR1636728	SRR1636729	SRR1636730	baseMean	log2FoldChange
VC1130|ID:1735915|hns|	20960	0	18141	1	1.024319e+04	-15.390456
VCSEN_bncRNA561	77	0	114	0	4.964988e+01	-9.139446
VIBCH10482|ID:1733969|	22	0	38	0	1.570148e+01	-7.494045
VC1854|ID:1736307|ompT|	265941	6399	237758	5352	1.335629e+05	-5.533453
VCSEN_ancRNA266	15	0	0	0	4.001725e+00	-5.502976
VCSEN_ancRNA274	0	0	11	0	2.807449e+00	-5.010165
VCSEN_ancRNA71	0	0	10	0	2.547773e+00	-4.854970
VCSEN_ancRNA324	0	0	9	0	2.296628e+00	-4.714862
VCSEN_ancRNA301	0	0	8	0	2.039622e+00	-4.539884
VC1333|ID:1736020|	2550	86	2007	125	1.241901e+03	-4.519809
VCSEN_bncRNA580	7	0	0	0	1.891270e+00	-4.431406

DESeq2: Making a heatmap

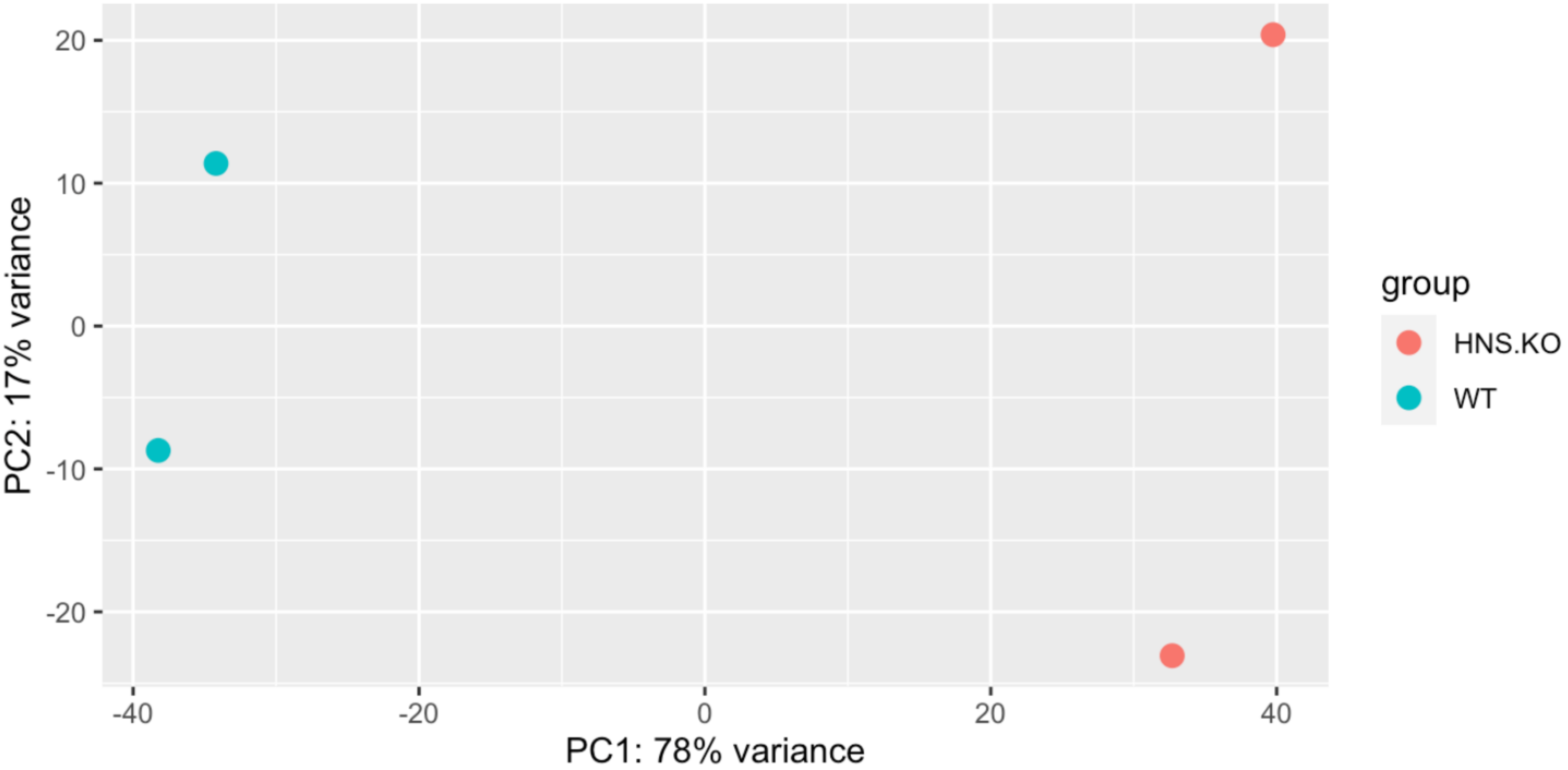
```
select <- order(rowMeans(counts_df), decreasing = TRUE)[1:20] # Select top 20 genes
df <- as.data.frame( colData(dds)[,c("phenotype")] ) # Get coldata for phenotype
rownames(df) <- coldata$Run # Fix rownames (must matach counts_df columns)
colnames(df) <- "phenotype" # Fix column name

# Make a heatmap
pheatmap(counts_df[select,],
         cluster_rows = FALSE,
         show_rownames = FALSE,
         cluster_cols = FALSE,
         annotation_col = df)
```

DESeq2: Making a heatmap



DESeq2: Making a PCA plot

```
ntd <- normTransform(dds) # Normalizes to gives log2(n + 1)
colnames(ntd) <- coldata$Run # Fix column names
plotPCA(ntd, intgroup = "phenotype") # Plot PCA
```



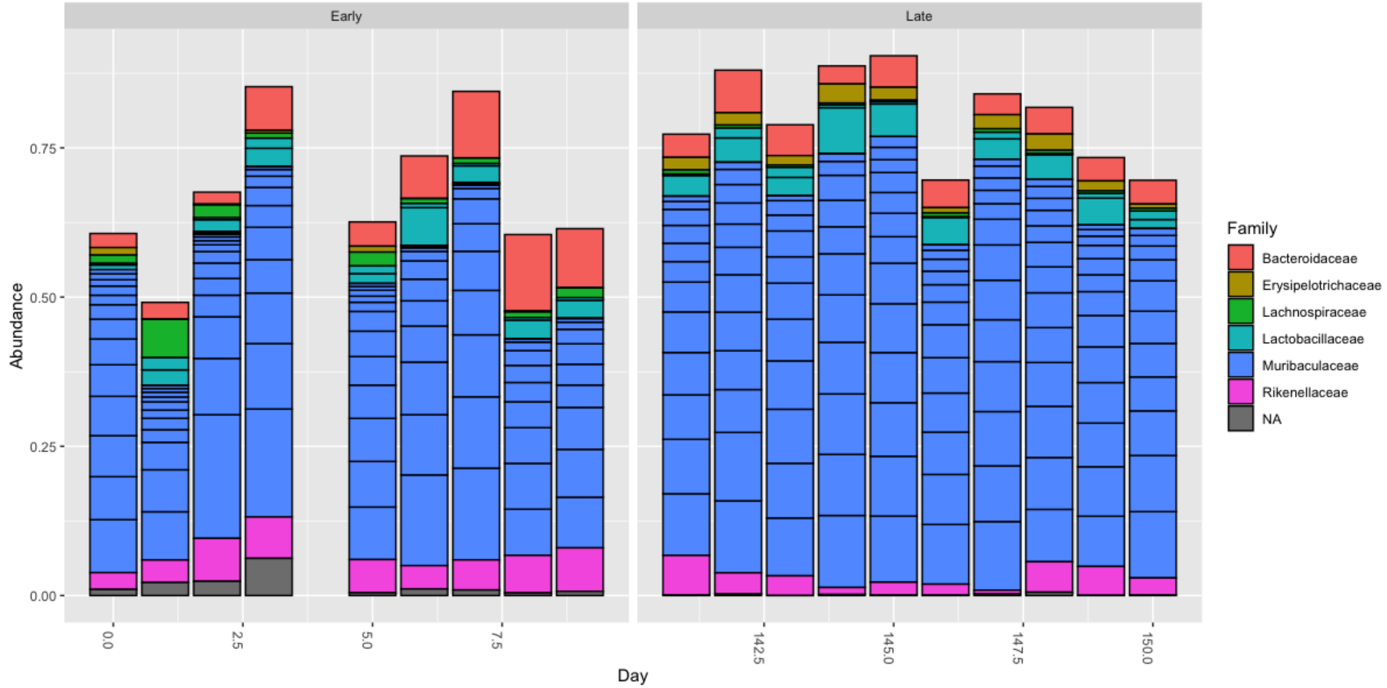
DADA2: Analyzing 16S data

DADA2: Analyzing 16S data

From raw fastqs…

@READ_1
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65

to OTUs (Operational Taxonomic Units)!



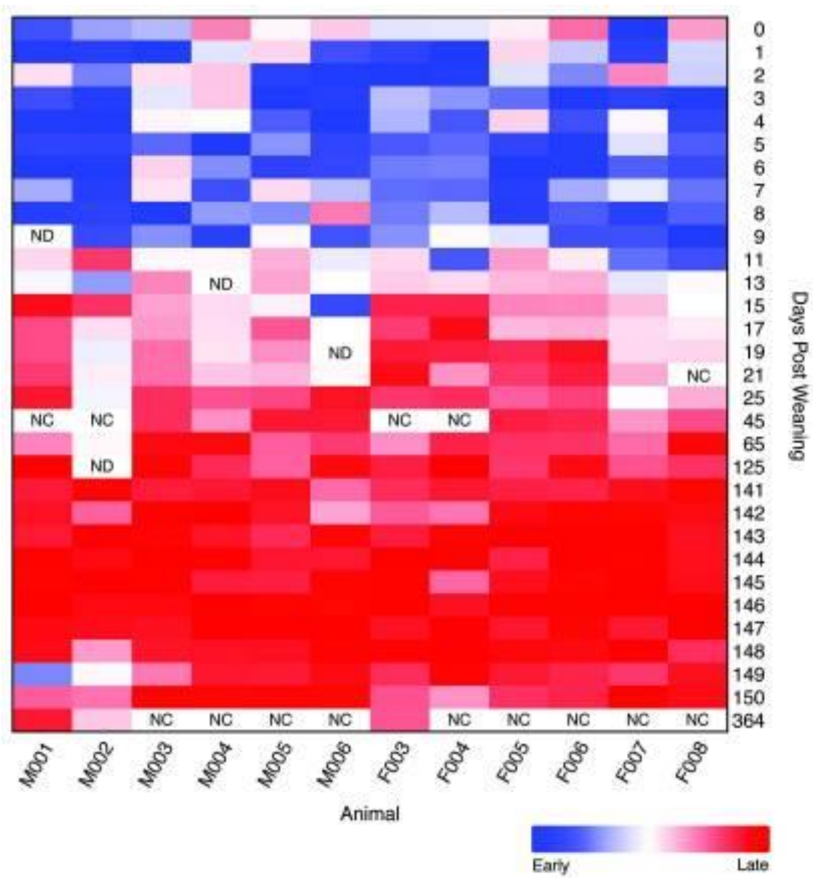
Following slides adapted from DADA2 vignette: https://benjjneb.github.io/dada2/tutorial.html

DADA2 – The dataset

Stabilization of the Murine Gut Microbiome Following Weaning

Patrick D Schloss, Alyxandria M Schubert, Joseph P Zackular, Kathryn D Iverson, Vincent B Young, Joseph F Petrosino

https://pubmed.ncbi.nlm.nih.gov/22688727/



DADA2 – Getting the data onto RStudio

Getting ready

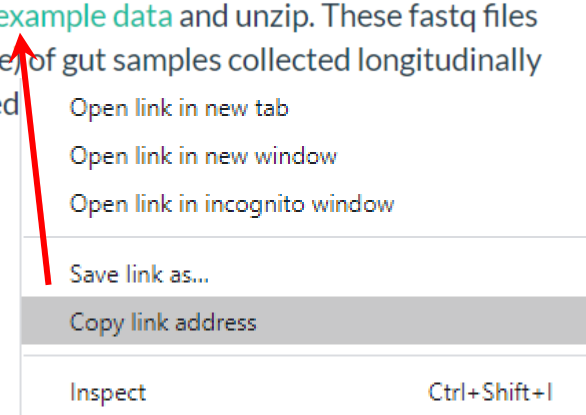
First we load the `dada2` package If you don't already it, see the dada2 installation instructions:

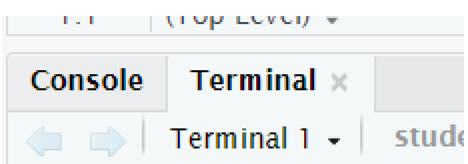
```
library(dada2); packageVersion("dada2")
```

```
## [1] '1.6.0'
```

Older versions of this workflow associated with previous release versions or the dada2 R package are also available: version 1.2, version 1.4.

The data we will work with are the same as those used in the Mothur Miseq SOP. Download the example data and unzip. These fastq files were generated by amplicon sequencing (Illumina MiSeq, 2x250, V4 region of the 16S rRNA gene) of gut samples collected longitudinally from a mouse post-weaning, and one mock community control. For now just consider them paired following path variable so that it points to the extracted directory on **your** machine:





wget https://mothur.s3.us-east-2.amazonaws.com/wiki/miseqsopdata.zip

unzip miseqsopdata.zip

DADA2: Import data from fastqs

```r
library(dada2) # Bioconductor
library(phyloseq) # Bioconductor
library(Biostrings) # Bioconductor
library(tidyverse)
```

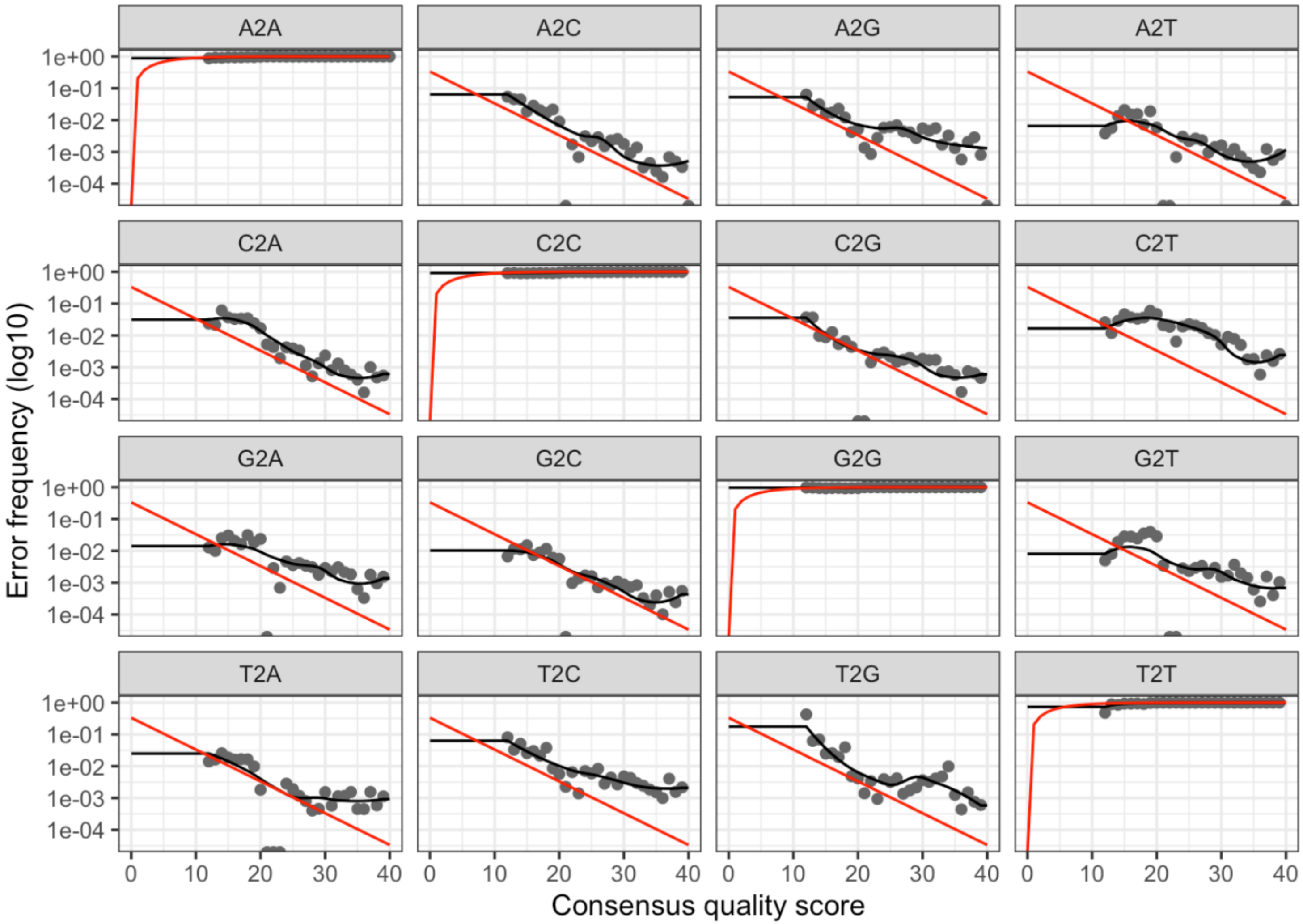
```r
path2dada2 <- "MiSeq_SOP" # Name directory with .fastq files
# Forward and reverse fastq filenames have format: SAMPLENAME_R1_001.fastq
# and SAMPLENAME_R2_001.fastq
fnFs <- sort(list.files(path2dada2, pattern="_R1_001.fastq", full.names = TRUE))
fnRs <- sort(list.files(path2dada2, pattern="_R2_001.fastq", full.names = TRUE))
# Extract sample names, assuming filenames have format: SAMPLENAME_XXX.fastq
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)

filtFs <- file.path(path2dada2, "filtered", paste0(sample.names, "_F_filt.fastq.gz"))
filtRs <- file.path(path2dada2, "filtered", paste0(sample.names, "_R_filt.fastq.gz"))
names(filtFs) <- sample.names
names(filtRs) <- sample.names

# Filter and trim input .fastq files
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen=c(240,160),
                     maxN=0, maxEE=c(2,2), truncQ=2, rm.phix=TRUE,
                     compress=TRUE, multithread=TRUE)
```

DADA2: Learn error rates

```
# These function "learn" errror rates. By using the distribution of nucleotides
# paired with underlying quality scores, dada2 can make a best guess at PCR or sequencing
#errors using machine learning.
errF <- learnErrors(filtFs, multithread=TRUE)
errR <- learnErrors(filtRs, multithread=TRUE)
```

DADA2: Learn error rates

```
plotErrors(errF, nominalQ = TRUE)
```



DADA2: Dereplicate samples

```
# Dereplication cuts down later computation steps by combining identical sequencing
# reads into into "unique sequences" with a corresponding "abundance" to cut down computation
# time. This step is no longer necessary in the newest version of dada2.
derepFs <- derepFastq(filtFs, verbose=TRUE)
derepRs <- derepFastq(filtRs, verbose=TRUE)
# Name the derep-class objects by the sample names
names(derepFs) <- sample.names
names(derepRs) <- sample.names
```

DADA2: Group reads into 16s amplicons

```
# Sample inference (group reads into 16s amplicons)
dadaFs <- dada(derepFs, err=errF, multithread=TRUE)
dadaRs <- dada(derepRs, err=errR, multithread=TRUE)
```

```
dadaFs[["F3D0"]] # Summary for first read pair of first read (F3D0)
dadaFs[["F3D0"]] # Summary for first read pair of first read (F3D0)
```

```
## dada-class: object describing DADA2 denoising results
## 128 sample sequences were inferred from 1979 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, BAND_SIZE = 16, USE_QUALS = TRUE
```

DADA2: Merge reads

```
# Merge paired end sequences
mergers <- mergePairs(dadaFs,
                      derepFs,
                      dadaRs,
                      derepRs,
                      verbose=TRUE)

head(mergers[[1]]) # Inspect the merger date from first sample
```

DADA2: Merge reads

```
                                     sequence
## 1 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGCAGGCGGAAGATCAAGTCAGCGGTAAAATTGAGAGGCTCAACCTCTTCGAGCCGTTGAAA
CTGGTTTTCTTGAGTGAGCGAGAAGTATGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACTCCGATTGCGAAGGCAGCATACCGGCGCTCAACTGACG
CTCATGCACGAAAGTGTGGGTATCGAACAGG
## 2 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGTAGGCGGCCTGCCAAGTCAGCGGTAAAATTGCGGGGCTCAACCCCGTACAGCCGTTGAAA
CTGCCGGGCTCGAGTGGGCGAGAAGTATGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACCCCGATTGCGAAGGCAGCATACCGGCGCCCTACTGACG
CTGAGGCACGAAAGTGCGGGGATCAAACAGG
## 3 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGTAGGCGGGCTGTTAAGTCAGCGGTCAAATGTCGGGGCTCAACCCCGGCCTGCCGTTGAAA
CTGGCGGCCTCGAGTGGGCGAGAAGTATGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACTCCGATTGCGAAGGCAGCATACCGGCGCCCGACTGACG
CTGAGGCACGAAAGCGTGGGTATCGAACAGG
## 4 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGTAGGCGGGCTTTTAAGTCAGCGGTAAAAATTCGGGGCTCAACCCCGTCCGGCCGTTGAAA
CTGGGGGCCTTGAGTGGGCGAGAAGAAGGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACCCCGATTGCGAAGGCAGCCTTCCGGCGCCCTACTGACG
CTGAGGCACGAAAGTGCGGGGATCGAACAGG
## 5 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGCAGGCGGACTCTCAAGTCAGCGGTCAAATCGCGGGGCTCAACCCCGTTCCGCCGTTGAAA
CTGGGAGCCTTGAGTGCGCGAGAAGTAGGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACTCCGATTGCGAAGGCAGCCTACCGGCGCGCAACTGACG
CTCATGCACGAAAGCGTGGGTATCGAACAGG
## 6 TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGTAGGCGGGATGCCAAGTCAGCGGTAAAAAAGCGGTGCTCAACGCCGTCGAGCCGTTGAAA
CTGGCGTTCTTGAGTGGGCGAGAAGTATGCGGAATGCGTGGTGTAGCGGTGAAATGCATAGATATCACGCAGAACTCCGATTGCGAAGGCAGCATACCGGCGCCCTACTGACG
CTGAGGCACGAAAGCGTGGGTATCGAACAGG
##   abundance forward reverse nmatch nmismatch nindel prefer accept
## 1       579       1       1    148         0      0      1   TRUE
## 2       470       2       2    148         0      0      2   TRUE
## 3       449       3       4    148         0      0      1   TRUE
## 4       430       4       3    148         0      0      2   TRUE
## 5       345       5       6    148         0      0      1   TRUE
## 6       282       6       5    148         0      0      2   TRUE
```

DADA2: Look into amplicon distributions

```
seqtab <- makeSequenceTable(mergers) # Makes count table for amplicon sequence variants
dim(seqtab) # How many OTUs (amplicon sequence variants) do we see?
```

```
## [1] 20 288
```

```
table(nchar(getSequences(seqtab))) # Inspect distribution of sequence lengths
```

```
## 251    252    253    254    255
##   1     87    192      6      2
```

DADA2: Remove chimeras

```
# Remove chimeras (i.e. r1 from one sequence, r2 from another)
seqtab.nochim <-
removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbose=TRUE)

sum(seqtab.nochim)/sum(seqtab)
```

[1] 0.9643085

```
dim(seqtab.nochim)
```

[1] 20 229

```
dim(seqtab)
```

[1] 20 288

DADA2: Pipeline summary

```
# Track read counts through pipeline
getN <- function(x) sum(getUniques(x)) # Function to get unique reads
track <-
cbind(out,
        sapply(dadaFs, getN),
        sapply(dadaRs, getN),
        sapply(mergers, getN),
        rowSums(seqtab.nochim)) # Get unique reads for each step

colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")
rownames(track) <- sample.names
head(track)
```

##	input	filtered	denoisedF	denoisedR	merged	nonchim
## F3D0	7793	7113	7113	7113	6600	6588
## F3D1	5869	5299	5299	5299	5078	5067
## F3D141	5958	5463	5463	5463	5047	4928
## F3D142	3183	2914	2914	2914	2663	2600
## F3D143	3178	2941	2941	2941	2575	2550
## F3D144	4827	4312	4312	4312	3668	3527

DADA2: Assign taxonomies

```
# Assign taxonomy
taxa <- assignTaxonomy(seqtab.nochim,
                       refFasta = "taxonomy/silva_nr_v138_train_set.fa.gz",
                       multithread = TRUE)
taxa.print <- taxa # Removing sequence rownames for display only
rownames(taxa.print) <- NULL
head(taxa.print)
```

```
## Kingdom Phylum Class Order
## [1,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## [2,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## [3,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## [4,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## [5,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## [6,] "Bacteria" "Bacteroidota" "Bacteroidia" "Bacteroidales"
## Family Genus
## [1,] "Muribaculaceae" NA
## [2,] "Muribaculaceae" NA
## [3,] "Muribaculaceae" NA
## [4,] "Muribaculaceae" NA
## [5,] "Bacteroidaceae" "Bacteroides"
## [6,] "Muribaculaceae" NA
```

DADA2: Format for phyloseq

```
# Format data for phyloseq
samples.out <- rownames(seqtab.nochim)
subject <- sapply(strsplit(samples.out, "D"), `[`, 1)
gender <- substr(subject,1,1)
subject <- substr(subject,2,999)
day <- as.integer(sapply(strsplit(samples.out, "D"), `[`, 2))
samdf <- data.frame(Subject = subject, Gender = gender, Day = day)
samdf$When <- "Early"
samdf$When[samdf$Day>100] <- "Late"
rownames(samdf) <- samples.out
```

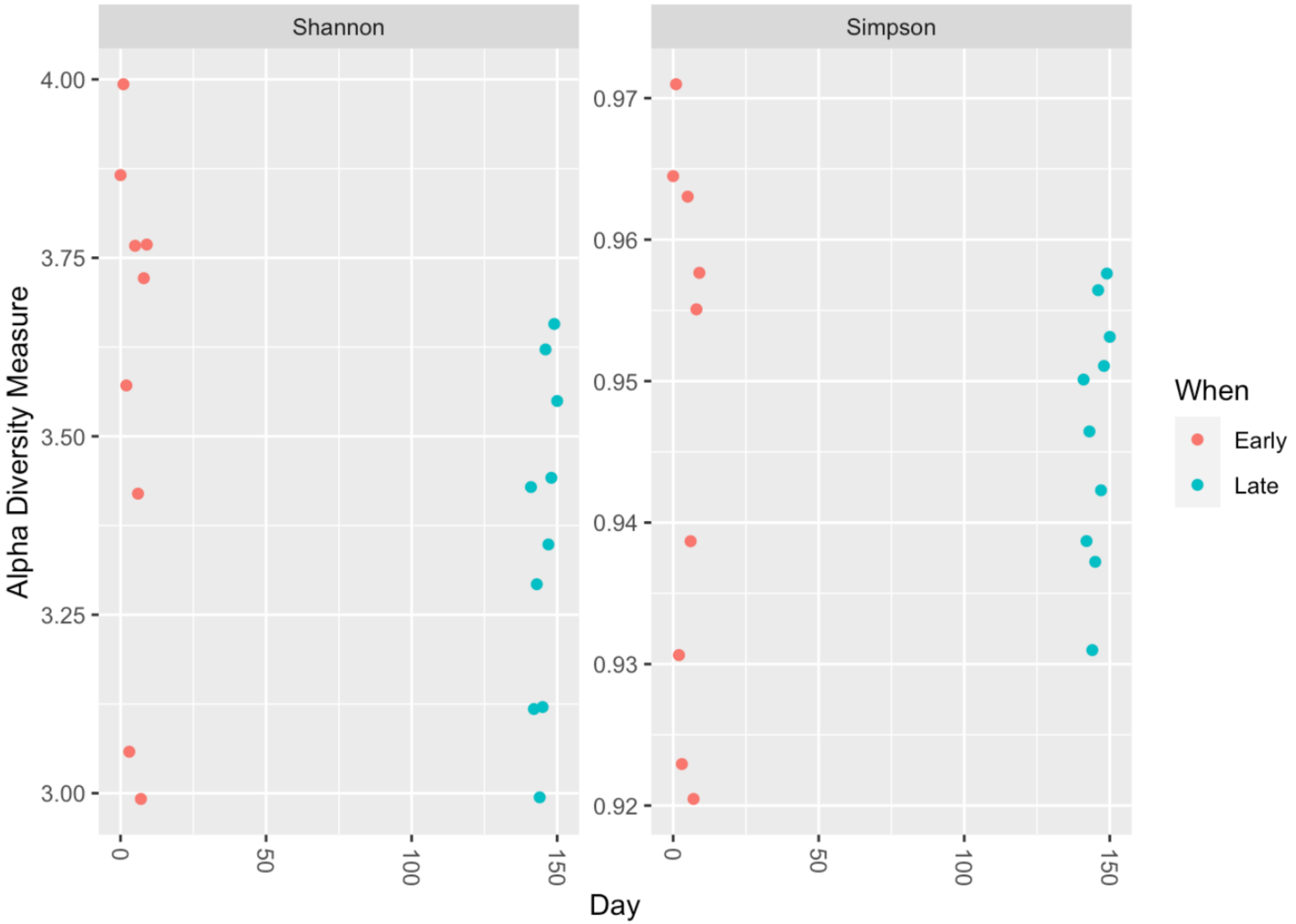
DADA2: Format for phyloseq

```
# Construct phyloseq object
ps <- phyloseq(otu_table(seqtab.nochim, taxa_are_rows=FALSE),
               sample_data(samdf),
               tax_table(taxa))
ps <- prune_samples(sample_names(ps) != "Mock", ps) # Remove mock sample

# Store full DNA sequences and give ASVs short names to more easily visualize
dna <- Biostrings::DNAStringSet(taxa_names(ps))
names(dna) <- taxa_names(ps)
ps <- merge_phyloseq(ps, dna)
taxa_names(ps) <- paste0("ASV", seq(ntaxa(ps)))
ps
```

```
## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 229 taxa and 19 samples ]
## sample_data() Sample Data: [ 19 samples by 4 sample variables ]
## tax_table() Taxonomy Table: [ 229 taxa by 6 taxonomic ranks ]
## refseq() DNAStringSet: [ 229 reference sequences ]
```

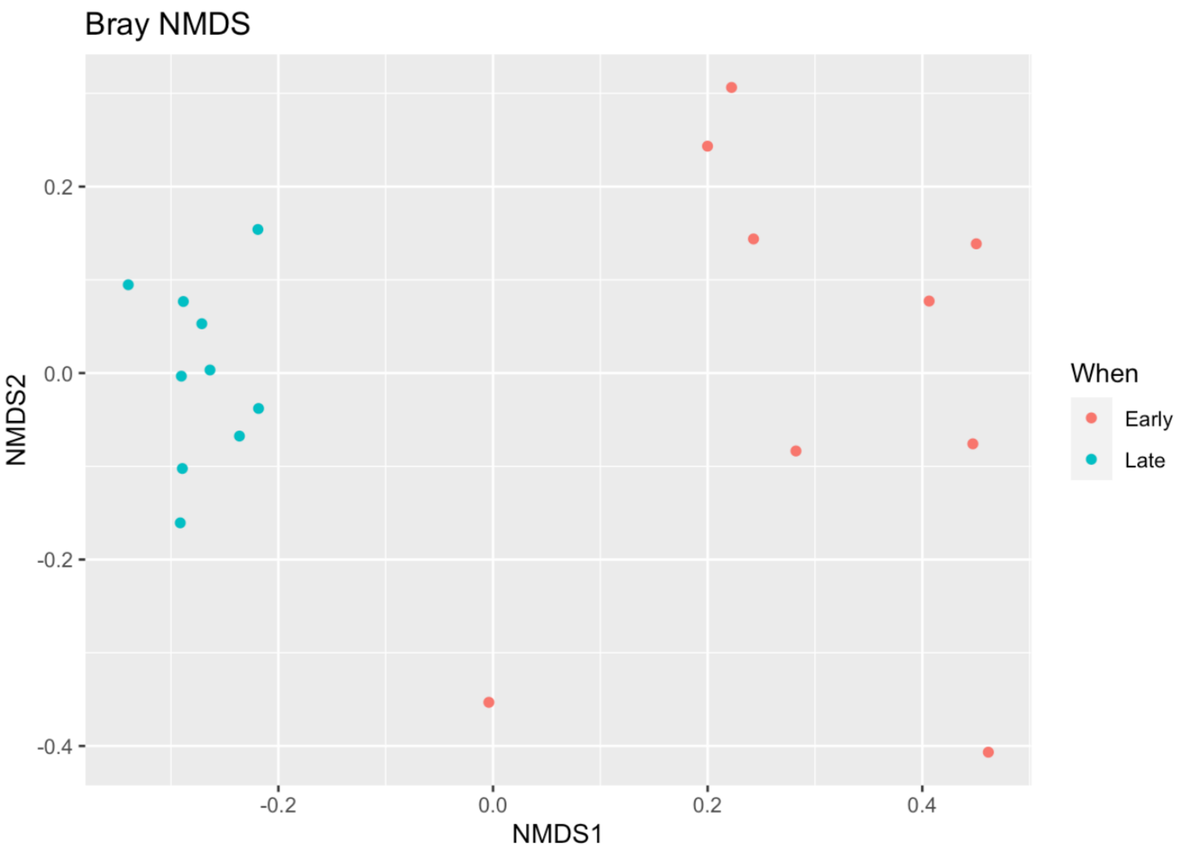
DADA2: Plot alpha diversity

```
# Alpha diversity
plot_richness(ps, x="Day", measures=c("Shannon", "Simpson"), color="When")
```



DADA2: Plot ordination (PCA)

```
# Transform data to proportions as appropriate for Bray-Curtis distances (similar to PCA)
ps.prop <- transform_sample_counts(ps, function(otu) otu/sum(otu))
ord.nmds.bray <- ordinate(ps.prop, method="NMDS", distance="bray")
plot_ordination(ps.prop, ord.nmds.bray, color="When", title="Bray NMDS")
```



DADA2: Plot species abundances

```
top20 <- names(sort(taxa_sums(ps), decreasing=TRUE))[1:20]
ps.top20 <- transform_sample_counts(ps, function(OTU) OTU/sum(OTU))
ps.top20 <- prune_taxa(top20, ps.top20)
plot_bar(ps.top20, x="Day", fill="Family") +
          facet_wrap(~When, scales="free_x")
```