Pipeline

Jacob Westaway

Last updated on 2021-08-15

About.

This document contains the pipepline and preliminary analyses for the manuscript *The bacterial gut microbiome of probiotic-treated very-preterm infants – Changes from admission to discharge*. The first part of this workflow goes raw reads to interpretable abundances, and is based largely around this DADA2 workflow developed by *Callahan, et al.*), in combination with removal of contamination with MicroDecon. The subsequent analyses uses a combination of packages, most notably phloseq, DESeq2, lme4, amongst many others.

Creating an ASV table from raw reads, using DADA2.

Load required packages.

```
sapply(c("dada2", "phyloseq", "DECIPHER", "phangorn", "BiocManager", "BiocStyle",
        "Biostrings", "ShortRead", "ggplot2", "gridExtra", "knitr","tibble"),
        require, character.only = TRUE)
```

Read quality.

Organise forward and reverse fastq filenames into own lists (check file format).

- First define the file path to the directory containing the fastq files (we will use this several times).

```
path <-"Data/"

fnFs <- sort(list.files(path, pattern="_R1_001.fastq.gz", full.names = TRUE))

fnRs <- sort(list.files(path, pattern="_R2_001.fastq.gz", full.names = TRUE))
```

Extract sample names.

```
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

Check quality of Forward and Reverse Reads (used to define truncLen in filtering).

```
plotQualityProfile(fnFs[1:2])
```

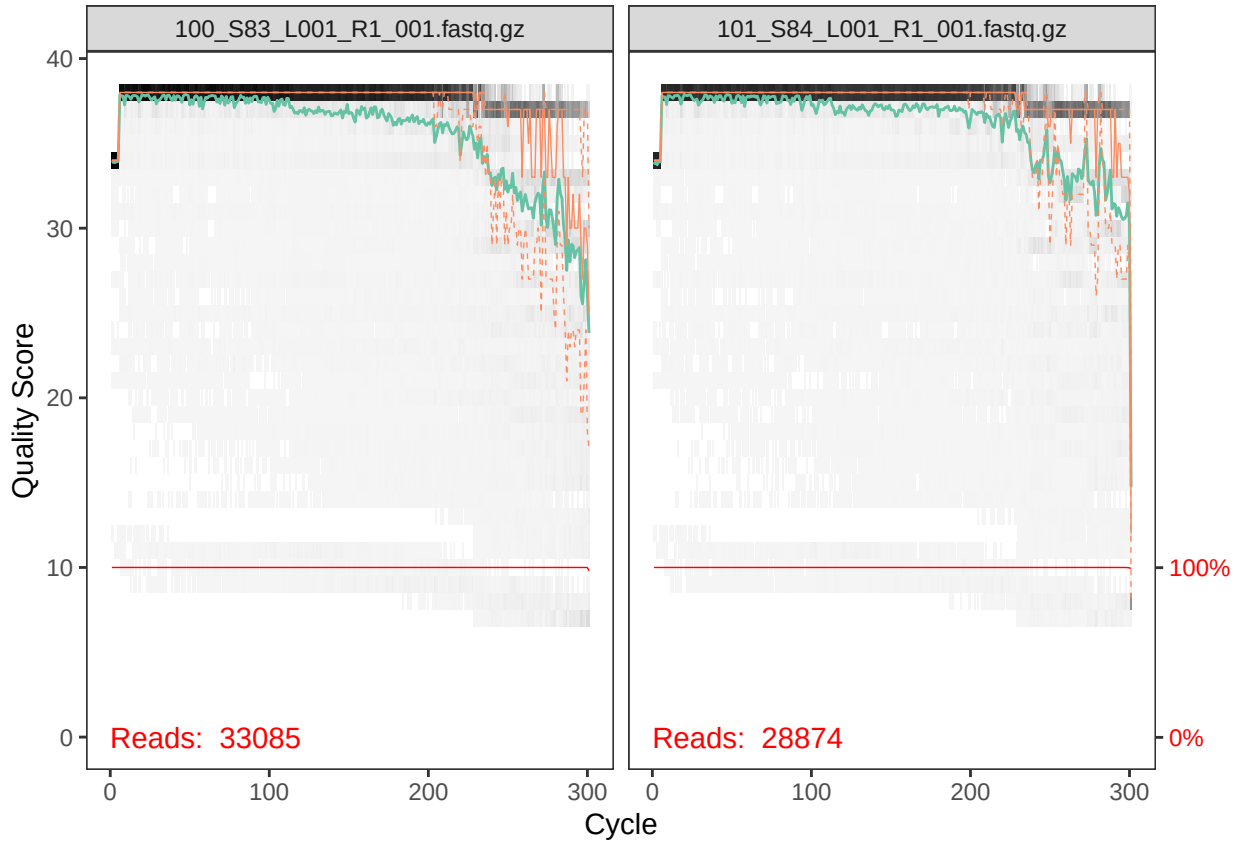


Figure 1: Quality of forward reads.

```
plotQualityProfile(fnRs[1:2])
```

Assign names for filtered reads.

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

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

Filter and trim the reads.

- Paremeters based on data and quality plots.
- `truncLean` defined by when quality plots begin to drop off, but ensuring it is large enough to maintain read overlap (=>20bp) downstream.
- `trimLeft = c(16,21)` is used to remove primers (16 and 21 are F and R primer length).

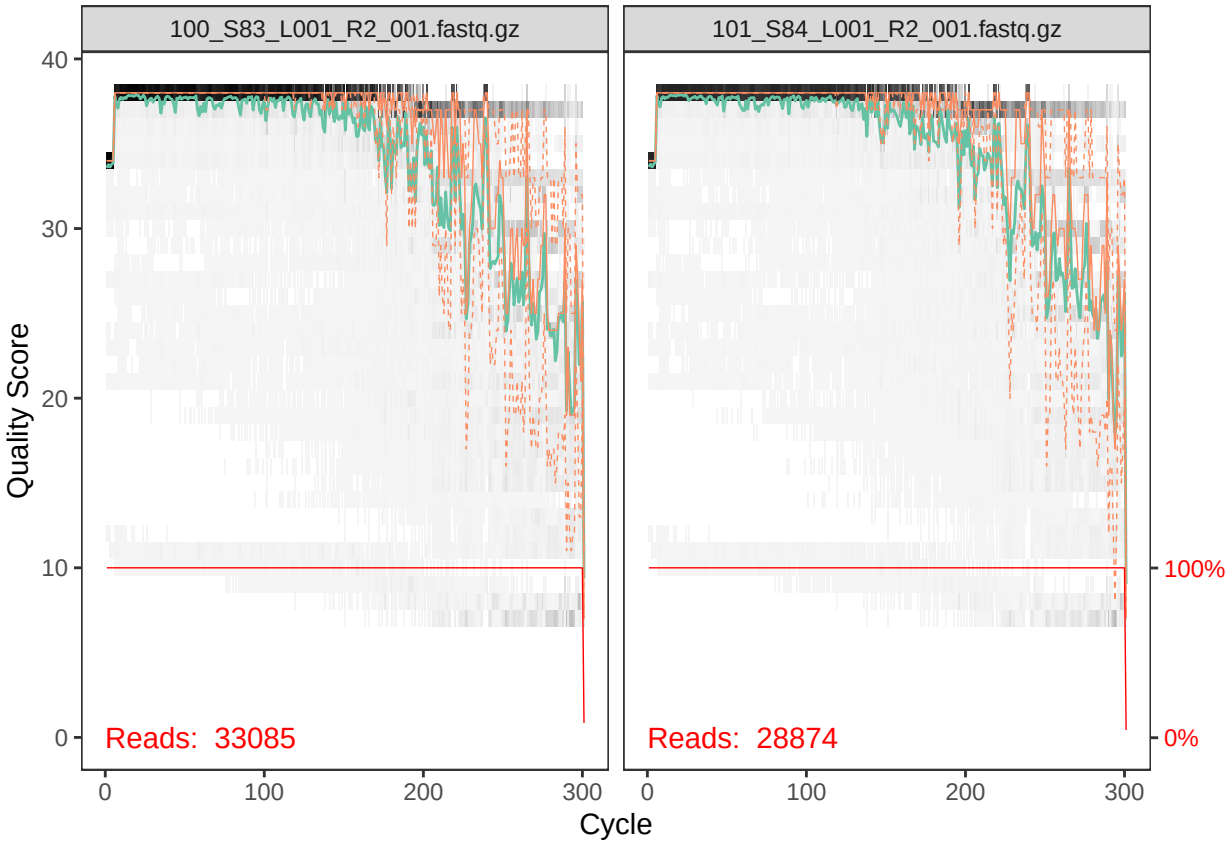


Figure 2: Quality of reverse reads.

- `maxEE = c(2,2)` is for filtering, where the higher the value the more relaxed filtering,allowing more reads to get through.
- Good quality data should allow for more stringent parameters (2 is stringent).
- The number of reads filtered is checked. If reads are too low, can alter parameters.

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen = c(280,200),
                     trimLeft = c(16,21),
                     maxN = 0,
                     maxEE = c(2,2),
                     truncQ = 2,
                     rm.phix = TRUE,
                     compress = TRUE,
                     multithread = FALSE) # windows can't support multithread
head(out)
```

Infer sequence variants.

Calculate Error Rates.

- Error rates are used for sample ineference downstream.

```
errF <- learnErrors(filtFs, multithread=TRUE)

errR <- learnErrors(filtRs, multithread=TRUE)
```

Plot error rates.

- Estimated error rates (black line) should be a good fit to observed rates (points) and error should decrease.

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

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

Dereplication.

- Combine indentical sequences into unique sequence bins.
- Name the derep-class objects by the sample name.

```
derepFs <- derepFastq(filtFs, verbose=TRUE)

derepRs <- derepFastq(filtRs, verbose=TRUE)

names(derepFs) <- sample.names

names(derepRs) <- sample.names
```

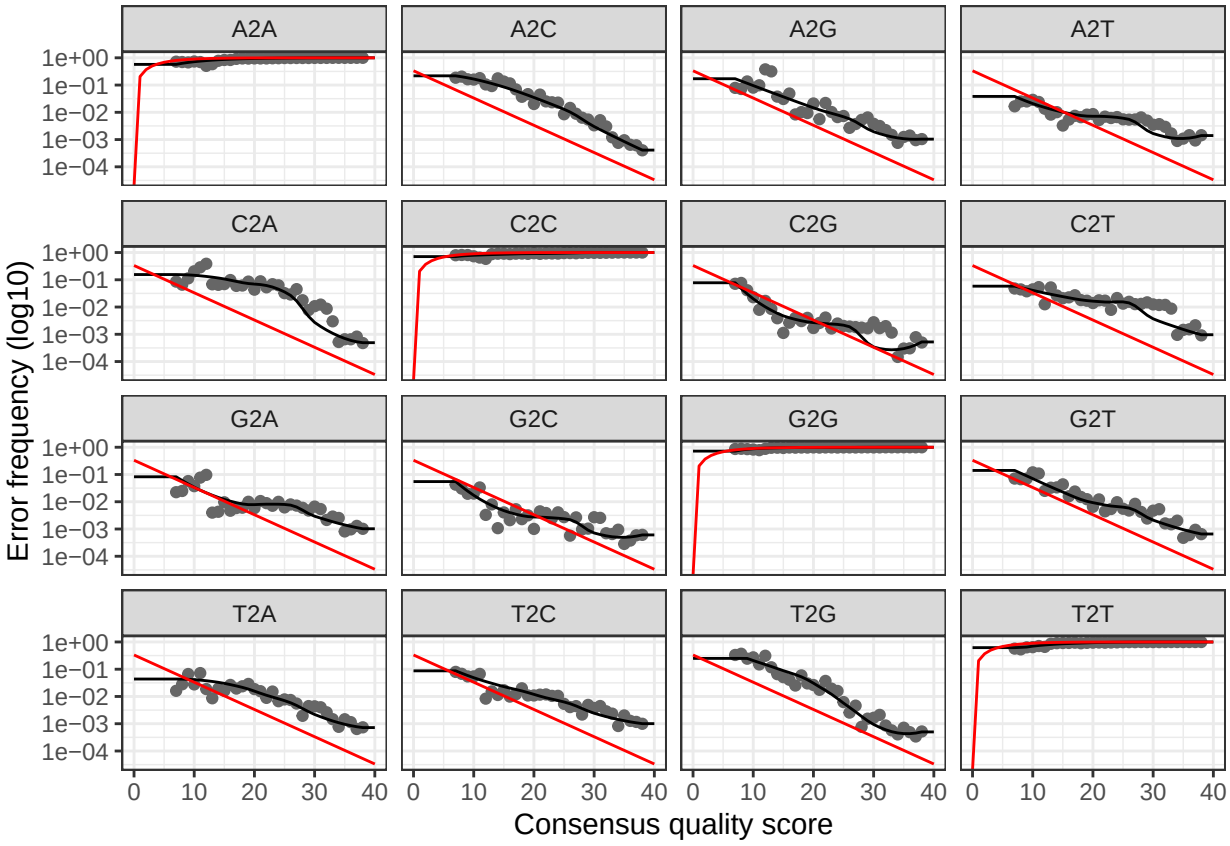


Figure 3: Error rates for forward reads

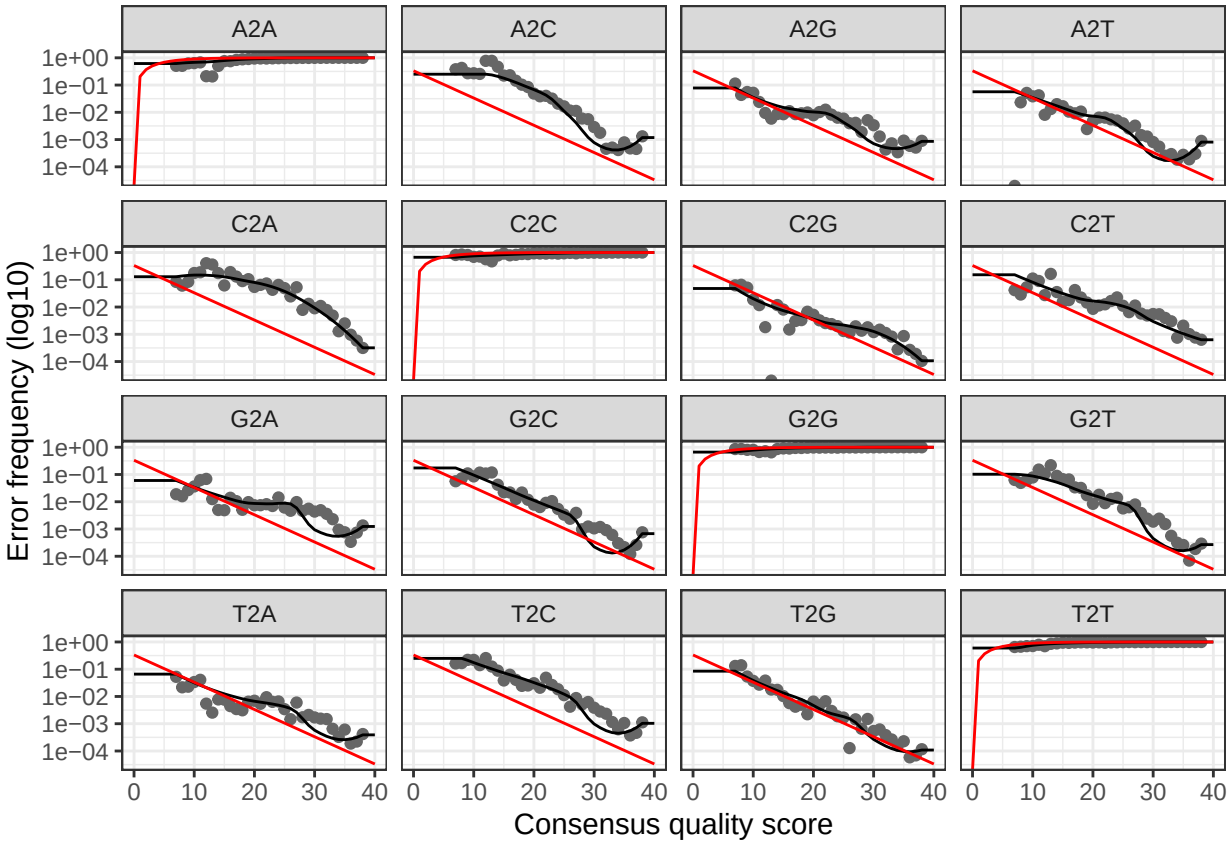


Figure 4: Error rates for reverse reads.

Sample Inference.

```
dadaFs <- dada(derepFs, err=errF, multithread=TRUE)

dadaRs <- dada(derepRs, err=errR, multithread=TRUE)
```

Inspect denoised data.

```
dadaFs[[1]]

dadaRs[[1]]
```

Merge Paired Reads and inspect merged data.

- Removes paired reads that do not perfectly overlap.
- Arguments represent infered samples AND denoised reads.

```
mergers <- mergePairs(dadaFs, derepFs, dadaRs, derepRs, verbose=TRUE)
```

Construct amplicon sequence variance (ASV) table and remove chimeras.

Construct ASV table.

- Check dimentions and inspect distribution of sequence lengths.

```
seqtab <- makeSequenceTable(mergers)

dim(seqtab)

table(nchar(getSequences(seqtab)))
```

Merging multiple sequence runs.

- Merging must be done after filtering.
- *seqtab.pilot* is a pilot dataset that was included in the overall analysis.

```
seqtab.merged <- mergeSequenceTables(seqtab, seqtab.pilot)
```

Remove chimeras.

```
seqtab.nochim <- removeBimeraDenovo(seqtab.merged, method="consensus",
                                    multithread=TRUE, verbose=TRUE)
```

Track reads through pipeline.

```
getN <- function(x) sum(getUniques(x))

track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN),
               sapply(mergers, getN), rowSums(seqtab.nochim))

colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")

rownames(track) <- sample.names

head(track)
```

```
##     input filtered denoisedF denoisedR merged nonchim
## 100 33085    31345     30959     31130  29735   15405
## 101 28874    27997     27874     27887  27348   21895
## 102 26505    25456     25265     25313  24389   15598
## 103 28695    27483     27382     27413  27014   19072
## 104 23909    22937     22892     22870  22780   22774
## 105 25749    24571     23731     24112  21241   11709
```

Contamination removal with *MicroDecon*.

```
library(microDecon)
```

Reformat data for *MicroDecon*.

- Transpose sequencing table (post chimera removal) and convert to a dataframe.

```
microdecon.df <- t(seqtab.nochim) %>%
  as.data.frame()
```

- Determine which columns the blank samples belong to (for the second half of micrdecon(), otherwise an error occurs).

```
which(colnames(microdecon.df)=="183" | colnames(microdecon.df)=="182" |
        colnames(microdecon.df)=="181" | colnames(microdecon.df)=="179" |
        colnames(microdecon.df)=="145" | colnames(microdecon.df)=="99")
```

- Make blank samples the first 6 columns.
- Remove blanks 99 (column 6) and 145 (column 5) (blanks 99 and 145 are both distinct from the other blanks, and appear to be contaminated by adjacent samples sometime during library prepeartion, so we will remove these two blanks prior to running MicroDecon).

```
microdecon.df.blanks <- cbind.data.frame(microdecon.df[,c("183", "182", "181",
                                                          "179", "145", "99")],
                                         microdecon.df[, -c(84, 83, 82, 80, 46, 165)])

microdecon.df.blanks.2 <- microdecon.df.blanks[,-c(6, 5)]
```

- Can check how many columns were removed with: `ncol(microdecon.df.blanks) - ncol(microdecon.df.blanks.2)`.

Restructure dataframe to a priori grouping.

- Read in "ColNames.csv", which has the column names from the metadata restructured (as rows in column A of an excel spreadsheet) so that they are in the below priori groupings:
  - NICU Admission
  - NICU Discharge
  - NICU Unknown
  - SCN
  - Other
- When read in, the data should look like this:

```
V1
<fctr>
179
181
182
183
2
3
6
3b
20
21
```

- Then reorder the microdecon.df.blanks.2 by the imported csv file.
- **NB.** this can be done using the *tidyverse* but it is also convoluted.

```
col.names <- read.csv("ColNames.csv", header = FALSE)

col.order <- col.names[,1]

microdecon.df.3 <- microdecon.df.blanks.2[,match(col.order,
                   colnames(microdecon.df.blanks.2))]
```

Make column 1 the ASV/OTU names.

- *MicroDecon* requires that the OTUs have a unique ID in column 1.
- Take the rownames from microdecon.df.3 and create a column with these names.
- Can check with `length(unique(microdecon.df.3.names[,1])) == nrow(microdecon.df.3)`.

```
microdecon.df.3.names <- cbind.data.frame(rownames(microdecon.df.3), microdecon.df.3)
```

Decontaminate data using `decon()`.

- `numb.ind` is the number of columns for each priori grouping.
- `taxa = F` as there is no taxonomy in the dataframe.

```
decontaminated <- decon(data = microdecon.df.3.names, numb.blanks = 4,
                  numb.ind = c(68, 66, 9, 14, 8), taxa = F)
```

```
decontaminated$decon.table
decontaminated$reads.removed
decontaminated$OTUs.removed %>% View()
decontaminated$mean.per.group
decontaminated$sum.per.group
```

Check *MicroDecon* Outputs.

Reformat decon.table.

- Convert column 1 to row names.
- Remove blank average column (1).
- Save rownames as seperate vector to be added back, as row names are removed during apply().
- Convert numeric values to integers (for downstream analysis).
- Transpose data.

```
seqtab.microdecon <- decontaminated$decon.table %>%
  remove_rownames() %>%
  column_to_rownames(var = "rownames(microdecon.df.3)")

seqtab.microdecon <- seqtab.microdecon[,-1]

save.rows <- rownames(seqtab.microdecon)

seqtab.microdecon <- apply(seqtab.microdecon,2,as.integer)

rownames(seqtab.microdecon) <- save.rows

seqtab.microdecon <- t(seqtab.microdecon)
```

Assign taxonomy.

- With optional species addition (there is an agglomeration step downstream, so you can add species now for curiosities sake, and remove later for analysis).

```
taxa <- assignTaxonomy(seqtab.microdecon, "silva_nr_v132_train_set.fa.gz")

taxa <- addSpecies(taxa, "silva_species_assignment_v132.fa.gz")

taxa.print <- taxa # Removes sequence rownames for display only
rownames(taxa.print) <- NULL
```

Preprocessing: Creating a Phyloseq Object.

About.

Creating a phyloseq object to be used for analysis, and create different objects to be used for different types of analysis downstream.

Load required packages.

```
sapply(c( "shiny","miniUI", "caret", "pls", "e1071", "ggplot2",
    "randomForest", "dplyr", "ggrepel", "nlme", "devtools",
    "reshape2", "PMA", "structSSI", "ade4","ggnetwork",
    "intergraph", "scales", "readxl", "genefilter", "impute",
    "phyloseq", "phangorn", "dada2", "DECIPHER", "gridExtra", "tidyverse"),
    require, character.only = TRUE)
```

Constuct a phylogenetic tree (for Phyloseq object downstream, required for distance measures).

- Peform multiple-allignment.
- `pml` calculates the likelihood of a given tree, and then `optim.pml()` optimizes the tree topology and branch length for the selected model (GTR+G+I max tree).

```
seqs <- getSequences(seqtab.microdecon)

names(seqs) <- seqs

alignment <- AlignSeqs(DNAStringSet(seqs), anchor=NA, verbose=FALSE)

phangAlign <- phyDat(as(alignment, "matrix"), type = "DNA")

fitGTR <- phangAlign %>%
  dist.ml() %>%
  NJ() %>%
  pml(data = phangAlign) %>%
  update(k = 4, inv = 0.2) %>%
  optim.pml(model = "GTR", optInv = TRUE, optGamma = TRUE,
  rearrangement = "NNI", control = pml.control(trace = 0))

detach("package:phangorn", unload = TRUE) # conflicts downstream
```

Import metadata and construct dataframe.

- Use ID column for row names.

```
samdf <- read_excel("SAMPLE_INFORMATION.xlsx", sheet = 1,
         col_names = TRUE, col_types = NULL, skip = 0) %>%
         data.frame(row.names = "ID") %>%
         mutate(DOB = as.Date(DOB)) %>%
         mutate(Date_of_Birth = as.Date(Date_of_Birth)) %>%
  left_join(
    read_excel("Metadata/Additional_Clinical_Data.xlsx") %>% # reviewer requested data
      select(ID, Days_on_antibiotics, Sex) %>%
      rename("Label" = ID)
    )  %>%
  mutate(Days_since_birth = as.numeric(difftime(.$Date_Collected, .$DOB, units = "days"))) %>%
  mutate(Gest_at_collection = Days_since_birth + Gestational.Age.at.Birth) %>%
```

```
  mutate(ID = Label) %>%
  column_to_rownames("ID")
```

Constrcut the Phyloseq object.

- Includes: metadata, ASV table, taxonomy table and phylogenetic tree.

```
ps <- phyloseq(otu_table(seqtab.microdecon, taxa_are_rows=FALSE),
               sample_data(samdf),
               tax_table(taxa),
               phy_tree(fitGTR$tree))
```

Wrangling the metadata.

- And do some additional wrangling.
- Convert chraracters to factors.
- Duplicate the label column, and then convert to newly created duplicate into rownames (need the original column downstream).

```
sample_data(ps) <- sample_data(ps) %>%
  unclass() %>%
  as.data.frame() %>%
  mutate_if(is.character, as.factor) %>%
  mutate("Label2" = Label) %>%
  column_to_rownames("Label2") %>%
  dplyr::rename(Diabetes = Diabetetes)
```

Filtering and normalisation.

Taxonomy filtering.

- Can check the number of phyla before and after transformation with `table(tax_table(ps)[, "Phylum"], exclude = NULL)`.
- Remove features with ambiguous and NA phylum annotation.

```
ps <- subset_taxa(ps, !is.na(Phylum) & !Phylum %in% c("", "uncharacterized"))
```

Prevelance filtering.

- Using an unsupervised method (relying on the data in this experiment) explore the prevelance of features in the dataset.
- Calculate the prevalence of each feature and store as a dataframe.
- Add taxonomy and total read counts.

```
prevdf = apply(X = otu_table(ps),
               MARGIN = ifelse(taxa_are_rows(ps), yes = 1, no = 2),
               FUN = function(x){sum(x > 0)})
```

```
prevdf = data.frame(Prevalence = prevdf,
                    TotalAbundance = taxa_sums(ps),
                    tax_table(ps))
```

- Plot the relationship between prevelance and total read count for each feature. This provides information on outliers and ranges of features.

```
prevdf %>%
  subset(Phylum %in% get_taxa_unique(ps, "Phylum")) %>%
  ggplot(aes(TotalAbundance, Prevalence / nsamples(ps),color=Phylum)) +
  geom_hline(yintercept = 0.05, alpha = 0.5, linetype = 1) +
  geom_point(size = 2, alpha = 0.7) +
  scale_x_log10() +
  xlab("Total Abundance") + ylab("Prevalence [Frac. Samples]") +
  facet_wrap(~Phylum) + theme(legend.position="none")
```

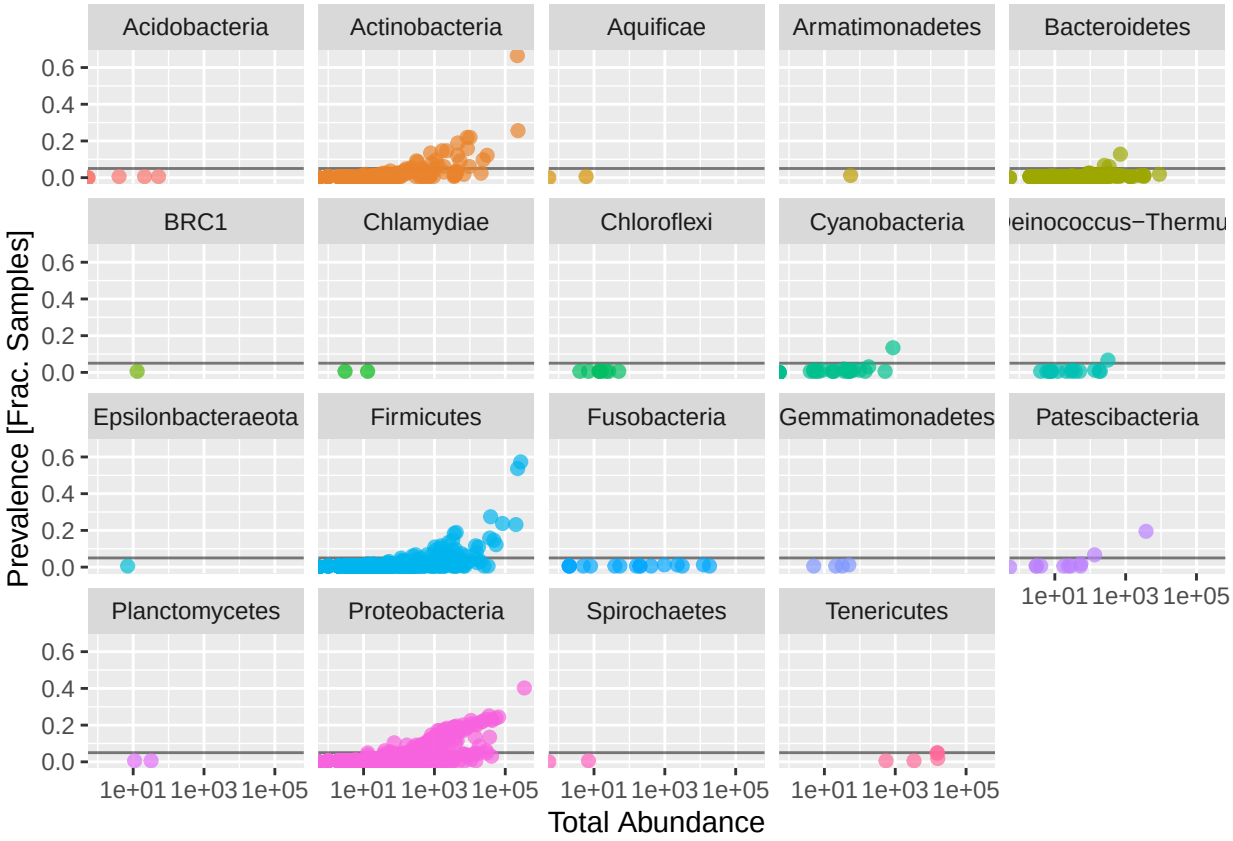


Figure 5: Scatterplot exploring the relationship between prevelance and abundance of phyla.

- Define prevalence threshold based on the plot (~1% is standard) and apply to ps object (if prevelance is too low don't designate a threshold).

```
prevalenceThreshold = 0.01 * nsamples(ps)

keepTaxa = rownames(prevdf)[(prevdf$Prevalence >= prevalenceThreshold)]

ps2 = prune_taxa(keepTaxa, ps)
```

- Explore the relationship on the filtered data set.

```
prevdf %>%
  subset(Phylum %in% get_taxa_unique(ps2, "Phylum")) %>%
  ggplot(aes(TotalAbundance, Prevalence / nsamples(ps2),color=Phylum)) +
  geom_hline(yintercept = 0.05, alpha = 0.5, linetype = 1) +
  geom_point(size = 2, alpha = 0.7) +
  scale_x_log10() +
  xlab("Total Abundance") + ylab("Prevalence [Frac. Samples]") +
  facet_wrap(~Phylum) + theme(legend.position="none")
```

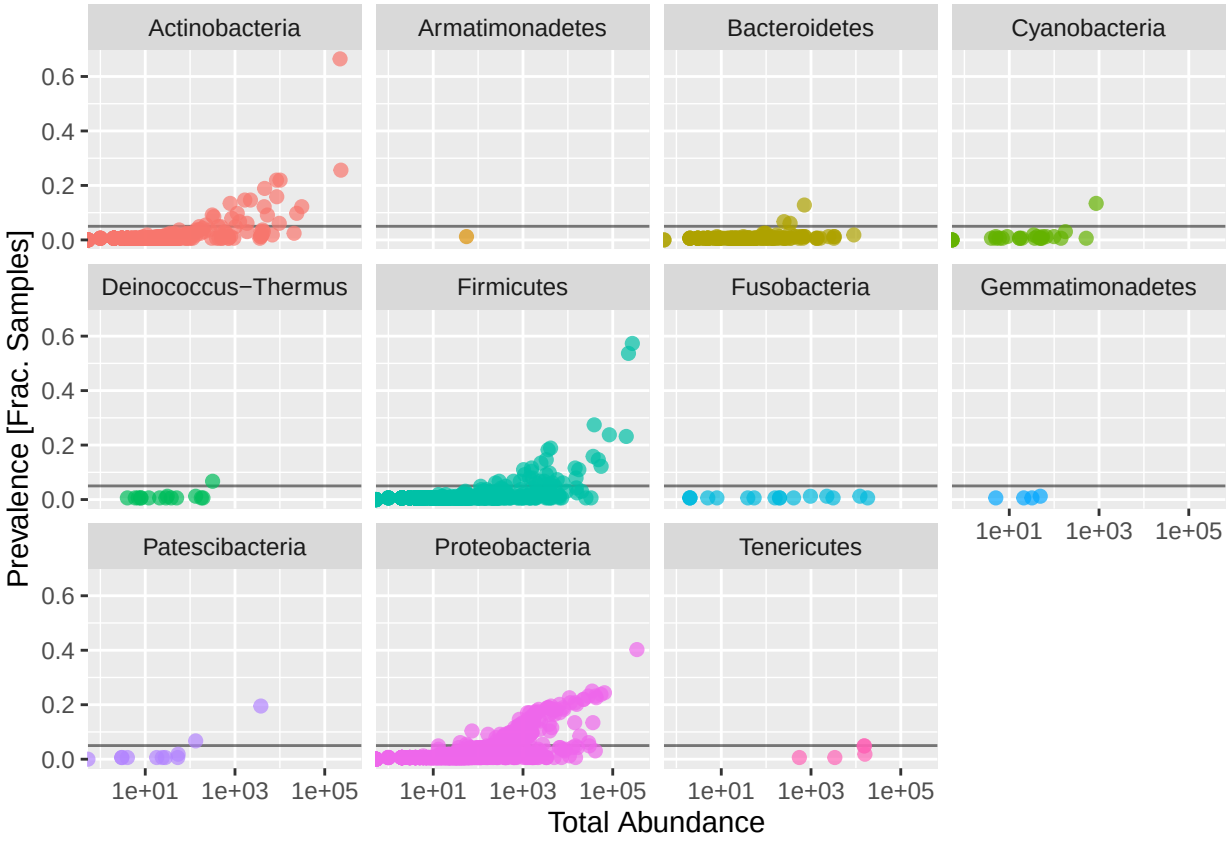


Figure 6: Scatterplot exploring the relationship between prevelance and abundance of phyla on data passed through a prevalence threshold.

Aggolmerate taxa.

- Combine features that descend from the same genus as most species have not been identified due to the poor sequencing depth in 16S.

- Can check how many genera would be present after filtering by running `length(get_taxa_unique(ps2, taxonomic.rank = "Genus"))`, and `ntaxa(ps3)` will give the number of post agglomeration taxa.

```
ps3 = tax_glom(ps2, "Genus", NArm = TRUE)
```

- Create tree plots to observe pre and post agglomeration.

```
grid.arrange(nrow = 1,
    (plot_tree(ps2, method = "treeonly",
      ladderize = "left", title = "Before Agglomeration") +
      theme(plot.title = element_text(size = 15))),
    (plot_tree(ps3, method = "treeonly",
      ladderize = "left", title = "Post Genus Agglomeration") +
      theme(plot.title = element_text(size = 15))))
```

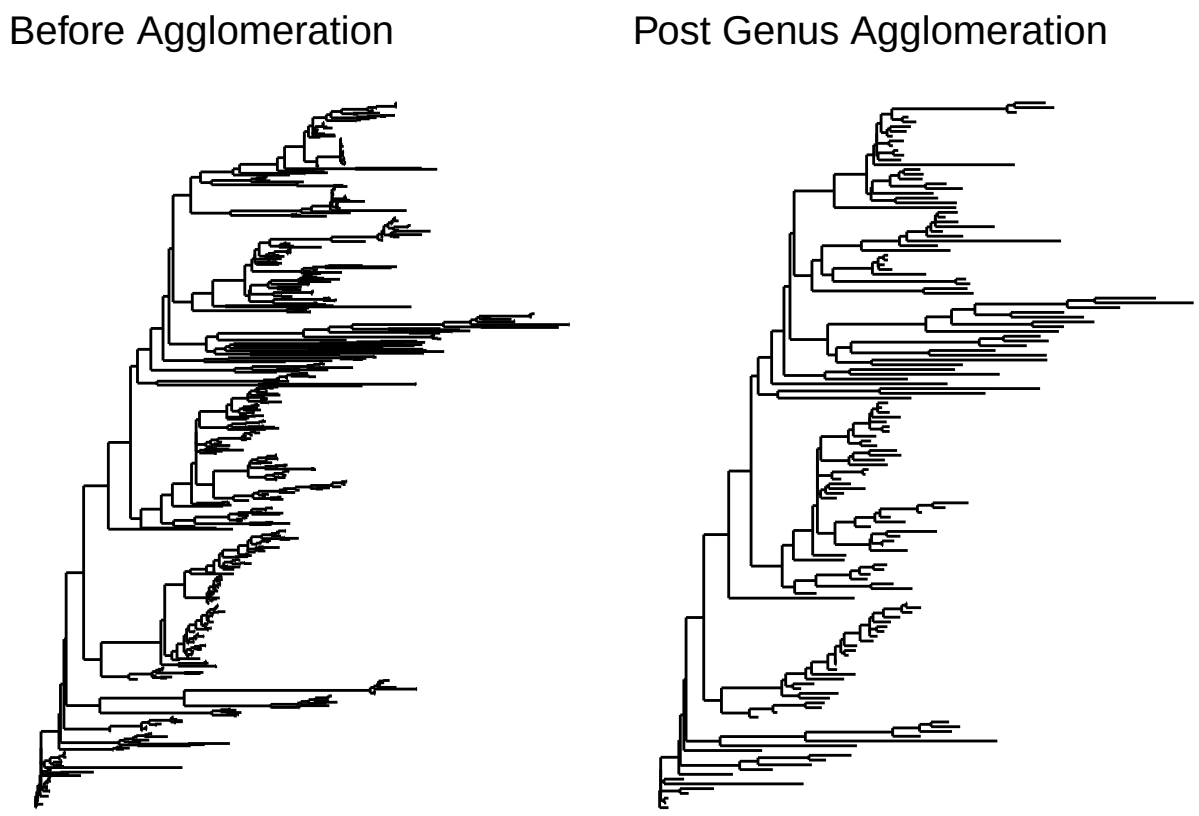
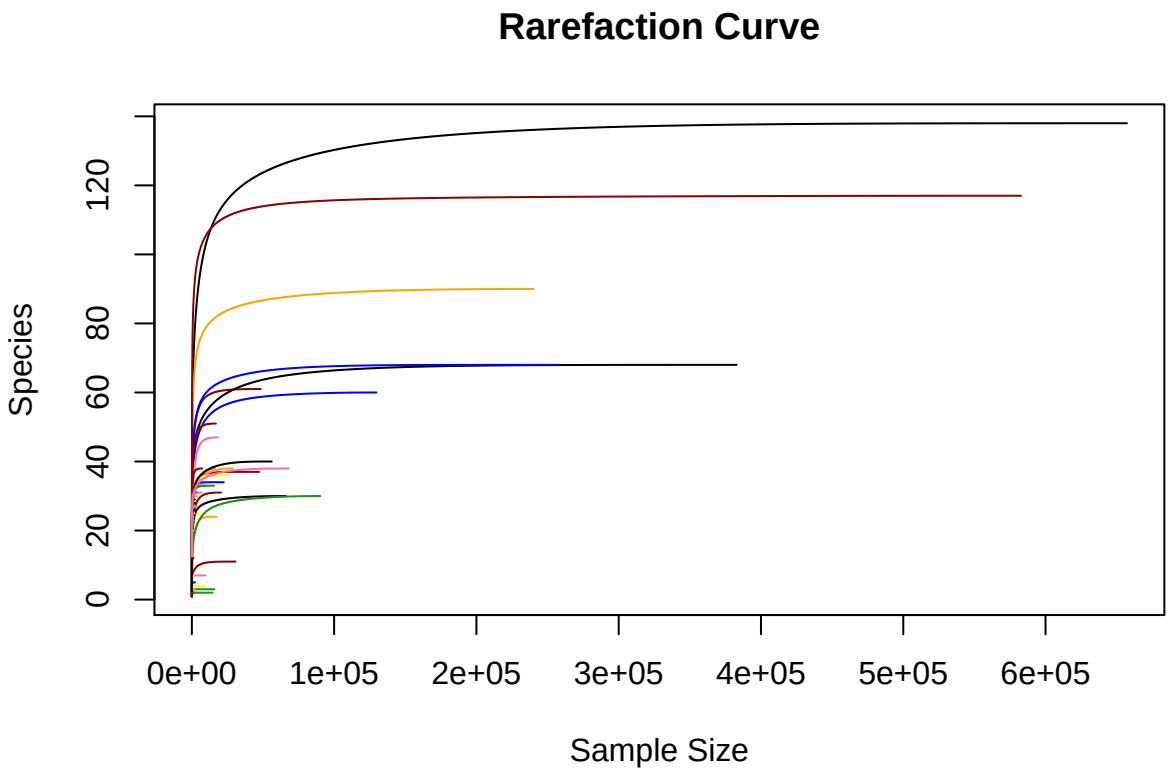


Figure 7: Tree plots exploring the agglomeration of taxa at the genus level.

Normalisation.

- Plot a refraction curve to see if total sum scaling will surfice.
- Define colours and lines.
- Step = step size for sample sizes in rarefaction curve.

```
vegan::rarecurve(t(otu_table(ps3)), step = 20, label = FALSE, main = "Rarefaction Curve",
        col = c("black", "darkred", "forestgreen", "orange", "blue", "yellow", "hotpink"))
```



- Perform total sum scaling on agglomerated dataset.

```
ps4 <- transform_sample_counts(ps3, function(x) x / sum(x))
```

Subset phyloseq object for data to be analyzed.

```
ps4.NICU_no_na <- subset_samples(ps4,
              Primary_Group == "NICU" &
              (Type == "Admission" | Type == "Discharge"))
```

- Explore normalisation with tree plots.

```
plot_tree(ps4.NICU_no_na, size = "Abundance", color = "Type",
    justify = "yes please", ladderize = "left") +
    labs(title = "Phylogenetic Tree and Relative Abundance") +
    scale_size_continuous(range = c(.5, 3))
```

- Explore normalisation with violin plots.

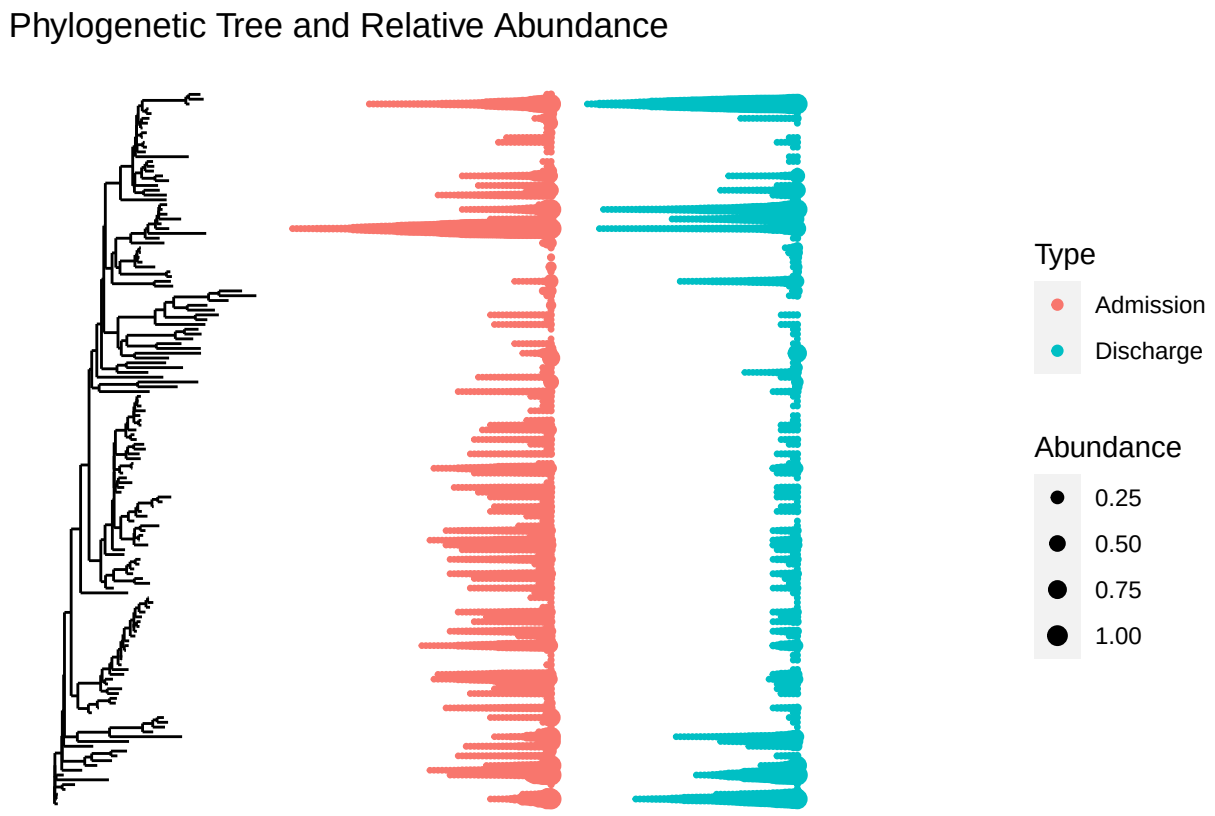


Figure 8: Tree plot exploring the normalised distribution of taxa between admission and discharge samples.

- Compares differences in scale and distribution of the abundance values before and after transformation.
- Using arbitrary subset, based on Phylum = Firmicutes, for plotting (ie. can explore any taxa to observe transformation).

```
plot_abundance = function(physeq, Title = "Abundance", Facet = "Order", Color = "Phylum", variable = "Ty
    subset_taxa(physeq, Phylum %in% c("Firmicutes")) %>%
    psmelt() %>%
    subset(Abundance > 0) %>%
    ggplot(mapping = aes_string(x = variable, y = "Abundance", color = Color, fill = Color)) +
      geom_violin(fill = NA) +
      geom_point(size = 1, alpha = 0.3, position = position_jitter(width = 0.3)) +
      facet_wrap(facets = Facet) +
      scale_y_log10()+
      scale_x_discrete(labels = c("A", "D", "I", "NA")) +
      theme(legend.position="none") +
      labs(title = Title)
}

grid.arrange(nrow = 2, (plot_abundance(ps3, Title = "Abundance", Color = "Type", variable = "Type")),
             plot_abundance(ps4, Title = "Relative Abundance", Color = "Type", variable = "Type"))
```

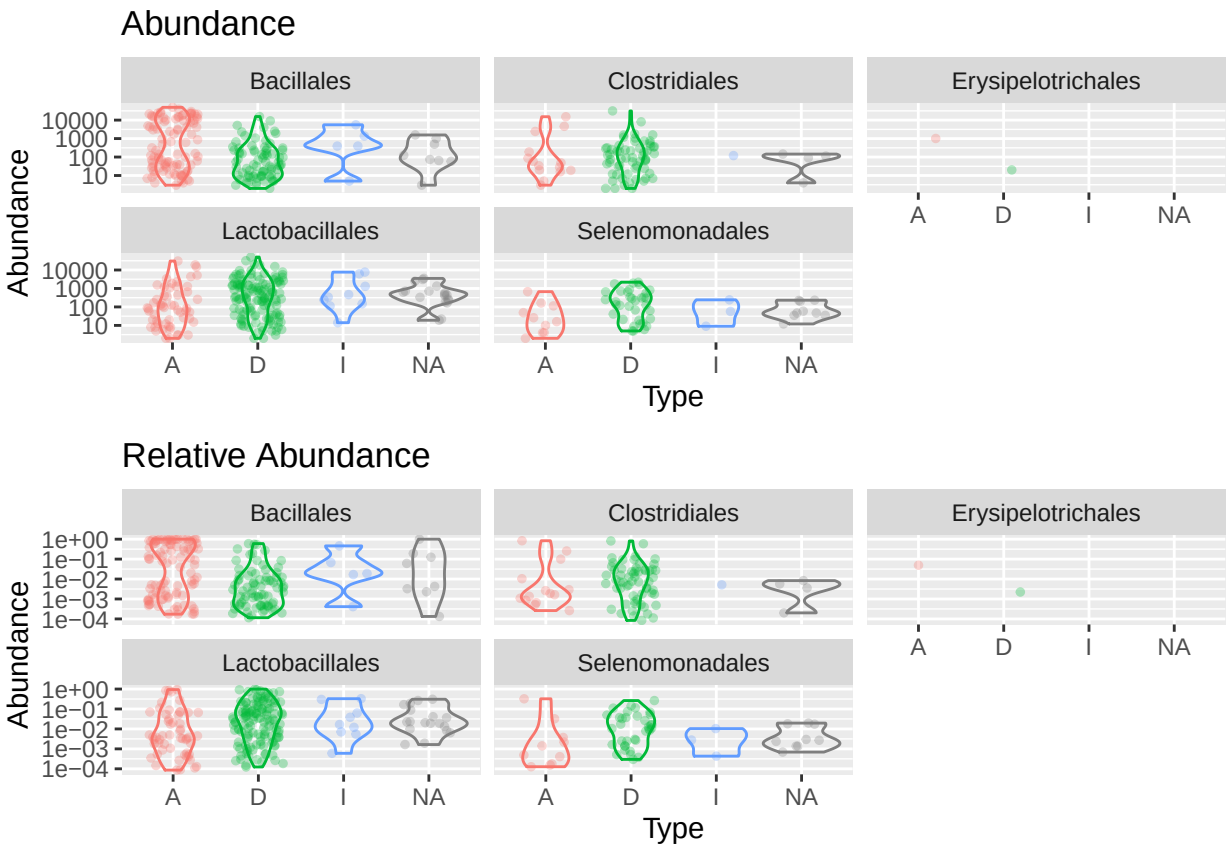


Figure 9: Violin plots exploring of distribution of abundance in Firmicutes before and after normalisation of data. Annotation for x axis; A: Admission, D: Discharge & I: Intermediate.

Data Exploration and Univariate Analysis.

About.

This section again uses the phyloseq package (along with several others) to explore the data using bar, violin and ordination plot. This then leads into a collection of univariate analyses, including; alpha and beta diversity, and also taxonomic differential abundance.

Load required packages.

```
sapply(c("BiocManager", "ggplot2", "ggforce", "vegan", "knitr", "dplyr",
         "phyloseq", "phyloseqGraphTest", "igraph", "ggnetwork", "nlme",
         "reshape2", "tidyverse", "plyr", "DESeq2", "sjPlot", "ggpubr",
         "gridExtra", "grid", "gtable", "lazyeval"), require, character.only = TRUE)
```

Taxanomic distribution.

Violin plots.

- Use previously defined violin plots to explore distributions of taxa.
- If a bimodal distribution is observed we can subset the data to determine if there is a taxonomic explination.
- Considerations: arguments can be altered for exploration.

```
subset_taxa(ps4, Order == "Lactobacillales") %>%
plot_abundance(Facet = "Genus", Color = "Type")
```

Bar charts

- Use `plot_bar_auto()` function wrapped around phyloseq's `plot_bar()` to explore the distribution of taxa at the genus and phylum levels.
- Subset transformed data (relative abundance) to only the top20 taxa.

```
top20 <- names(sort(taxa_sums(ps4.NICU_no_na), decreasing=TRUE))[1:20]
ps.top20 <- prune_taxa(top20, ps4.NICU_no_na)

plot_bar_auto <- function(ps, taxonomy){
plot_bar(ps, fill = taxonomy) +
    facet_wrap(~Type, scales = "free_x") +
    labs(title = paste0("Level:", taxonomy), y = "Abundance") +
    theme(legend.position = "bottom", legend.title = element_blank(),
    axis.title.x = element_blank(), axis.text.x = element_blank(),
    axis.ticks = element_blank())
}

grid.arrange(plot_bar_auto(ps.top20, "Phylum"),
             plot_bar_auto(ps.top20, "Genus"),
             nrow = 2, heights = c (1,1.2))
```

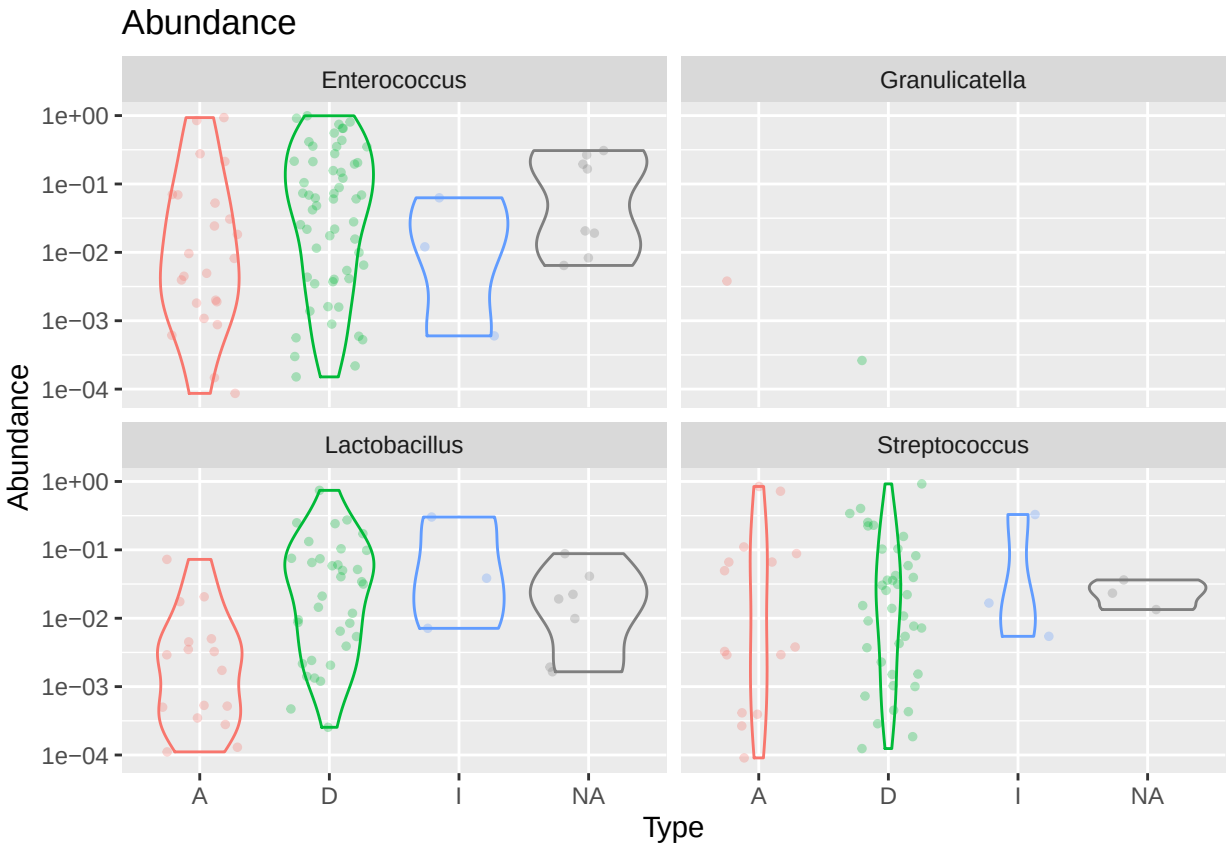


Figure 10: Violin plots exploring of distribution of abundances within Lactoacillales. Annotation for x axis; A: Admission, D: Discharge & I: Intermediate.

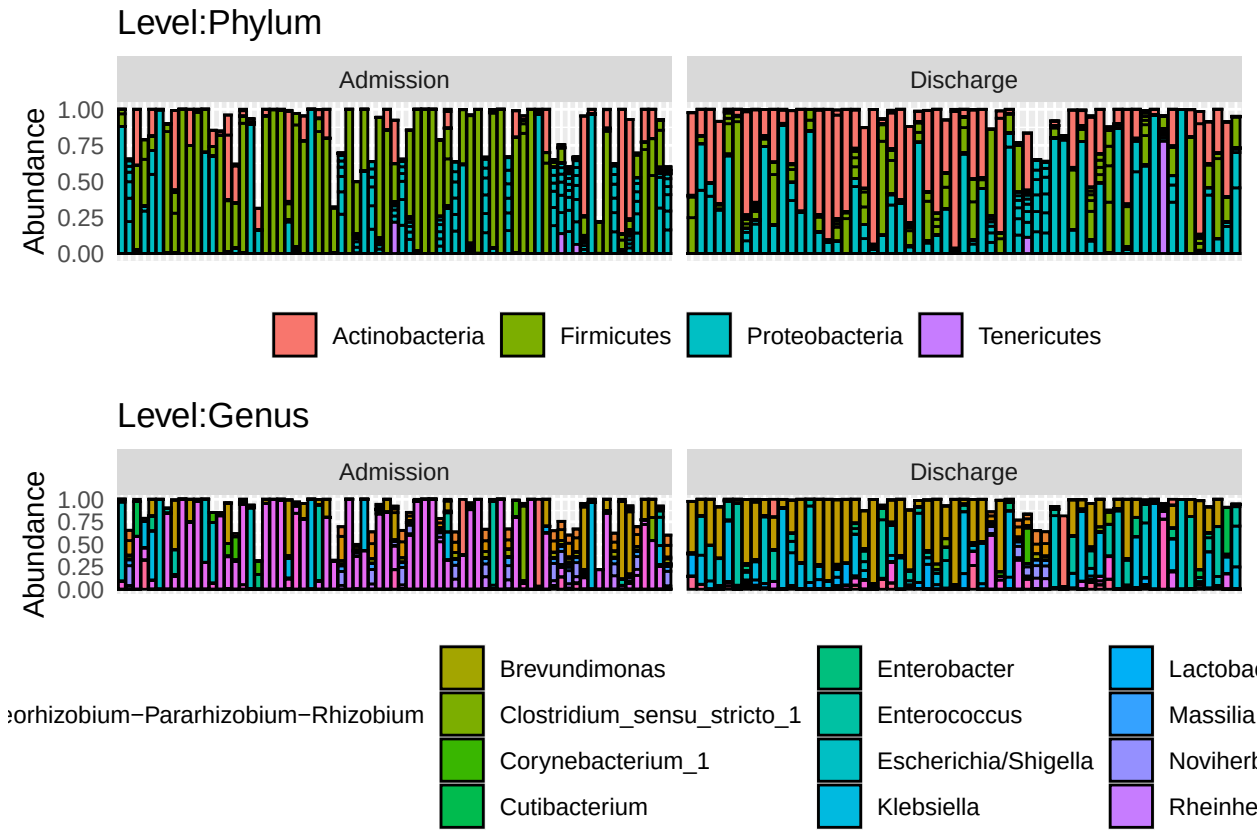


Figure 11: Bar plots of the taxonomic distribution (relative abundance) at phylum and genus levels.

- For other levels of taxonomy, with the legend hiden using `legend.position = "none"`.

```
plot_bar_auto_no_legend <- function(ps, taxonomy){
plot_bar(ps, fill = taxonomy) +
    facet_wrap(~Type, scales = "free_x") +
    labs(title = paste0("Level:", taxonomy), y = "Abundance") +
    theme(legend.position = "none", legend.title = element_blank(),
    axis.title.x = element_blank(), axis.text.x = element_blank(),
    axis.ticks = element_blank())
}

grid.arrange(plot_bar_auto_no_legend(ps.top20, "Class"),
             plot_bar_auto_no_legend(ps.top20, "Order"),
             plot_bar_auto_no_legend(ps.top20, "Family"),
             nrow = 3)
```

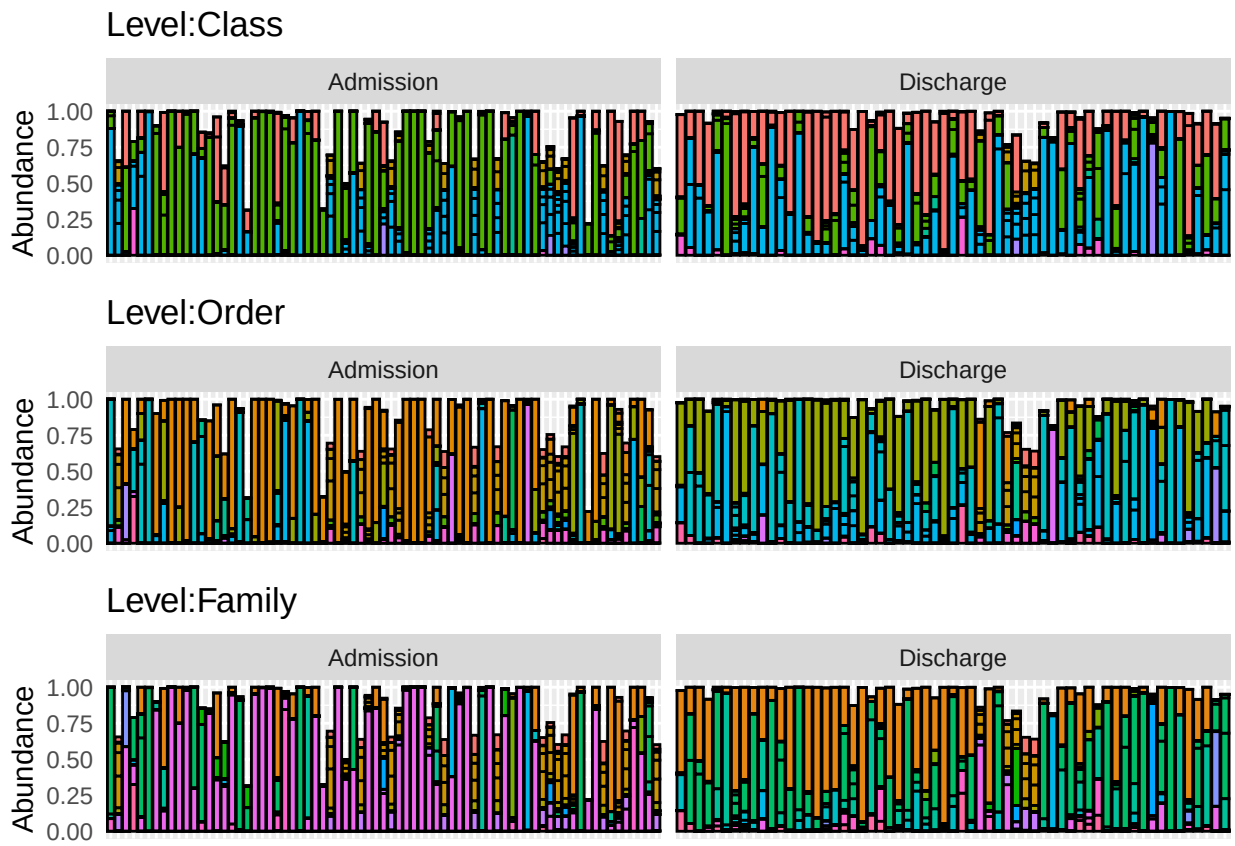


Figure 12: Bar plots of the taxonomic distribution (relative abundance) at class, order and family levels.

Calculate the number samples containing a given taxa `samples_with_taxa()` function.

- The function takes the phyloseq object, taxonomy level and taxanomic name (with the later two as strings).
- It then gets the ASV name from the *phyloseq* `tax_table()` by filtering with *dply* and *lazyeval*. (*lazyeval* is needed because of two concepts;non-standard evaluation and lazy evaluation.

- `paste()` is then used to concatenate the ASVs and `collapse` to insert the 'or' symbol.
- The function then matches the ASV names to the `otu_table()` of the *phyloseq* object to select the desired column(s) that represent the taxa of interest, and then counts the number of rows that have any of the selected taxa with counts greater than 0 to get the number of samples with that taxa present.

```
samples_with_taxa <- function(ps_object, taxonomy_level, taxa){
 ASV <- tax_table(ps_object) %>%
        unclass() %>%
        as.data.frame() %>%
        filter_(interp(~y == x, .values=list(y = as.name(taxonomy_level), x = taxa))) %>%
        row.names() %>%
        paste(collapse = " | ")

   otu_table(ps_object) %>%
        as.data.frame() %>%
        select(matches(ASV)) %>%
        filter_all(any_vars( . > 0)) %>%
        nrow()
}

samples_with_taxa(ps4.NICU_no_na, "Genus", "Bifidobacterium")
```

Beta diversity

- Use distance and ordination methods to explore the relationship between metadata.
- We calculate the distances using pruned, transformed and non-agglomerated data.

```
ps2.NICU_no_na <- subset_samples(ps2, Primary_Group == "NICU" &
        (Type == "Admission" | Type == "Discharge")) %>%
        transform_sample_counts(function(x) x / sum(x))
```

- We can then create distance matrices and plots for this data subset using several methods:
- bray-curtis or weighted unifrac distances with principle coordinate analysis (PCoA).
  - weighted-unifrac: phylogeny.
  - bray-curtis: abundance and phylogeny.
- Ordinate using PCoA and Weighted-Unifrac/Bray-Curtis.
- Extract eigenvalues from ordination.
- Plot ordination using eigenvalues and colour by variable *Type*.

PCoA and Bray-Curtis.

```
ps_ordination <- ordinate(ps2.NICU_no_na, method = "PCoA", distance = "bray")

evals <- ps_ordination$values$Eigenvalues

plot_ordination(ps2.NICU_no_na , ps_ordination, color = "Type",
  title = "PCoA (Bray-Curtis)") +
  labs(col = "Type") +
  coord_fixed(sqrt(evals[2] / evals[1])) +
```

```
  geom_point(size = 2)+
  stat_ellipse(type = "norm", linetype = 2)
```

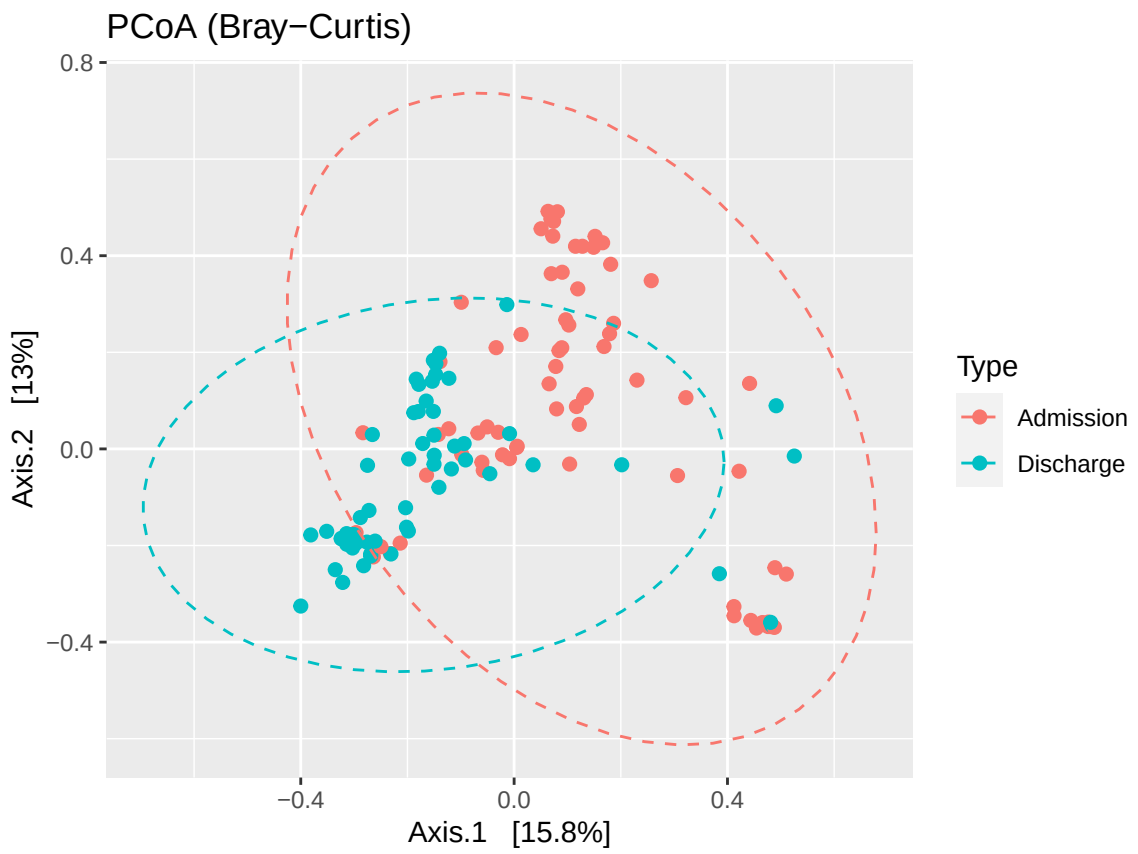


Figure 13: PCoA plot of Bray-Curtis matrix

PCoA and Weighted-Unifrac.

```
ps_ordination <- ordinate(ps2.NICU_no_na, method = "PCoA", distance = "wunifrac")

evals <- ps_ordination$values$Eigenvalues

plot_ordination(ps2.NICU_no_na , ps_ordination, color = "Type",
  title = "PCoA (Weighted-Unifrac)") +
  labs(col = "Type") +
  coord_fixed(sqrt(evals[2] / evals[1])) +
  geom_point(size = 2)+
  stat_ellipse(type = "norm", linetype = 2)
```

Is the overlap the result of collection date? i.e. late admission and early discharge samples overlap?

- Create a new column in the metadata table that is the collection date minus the date of birth.

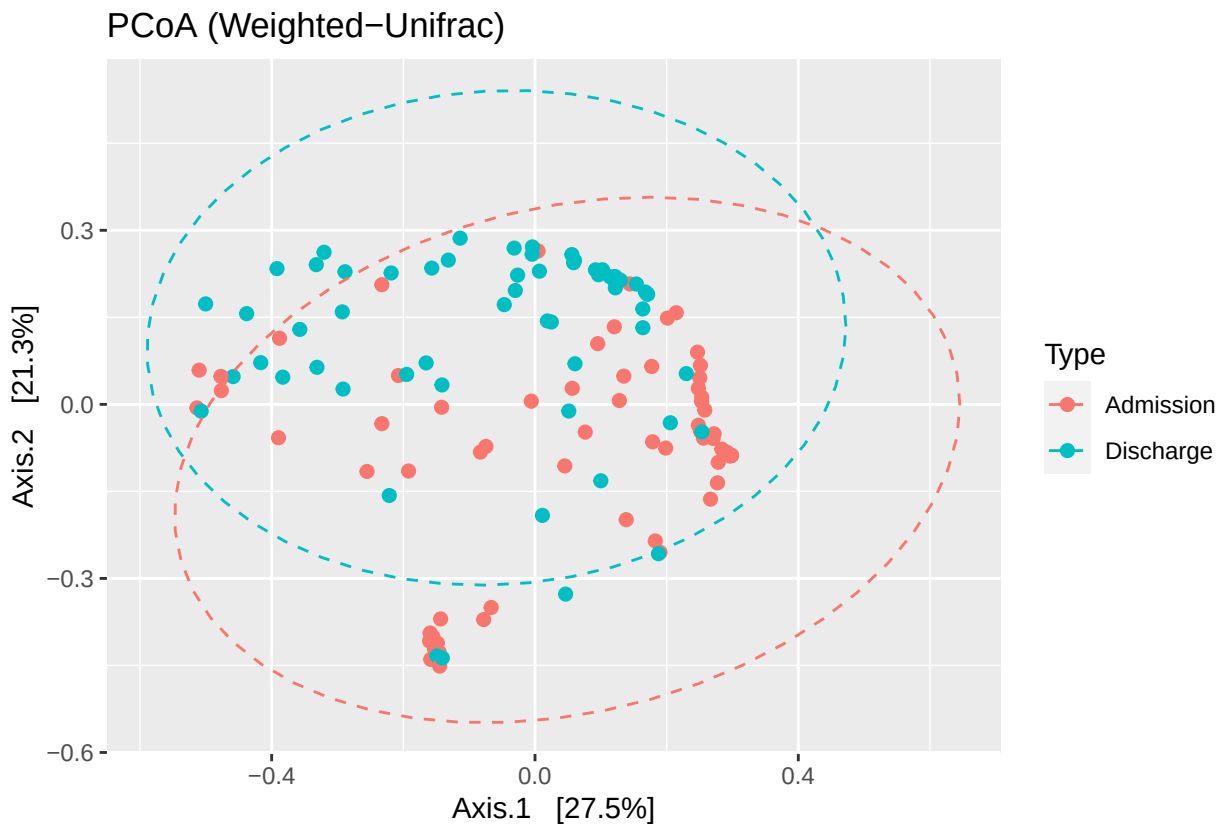


Figure 14: PCoA plot of Weighted-Unifrac matrix

```
sample_data(ps2.NICU_no_na) <- ps2.NICU_no_na %>%
  sample_data() %>%
  unclass() %>%
  as.data.frame() %>%
  mutate_at(7:8, as.character) %>%
  mutate_at(7:8, as.numeric) %>%
  mutate(Days_since_birth = Date_Collected-Date_of_Birth) %>%
  mutate("Label2" = Label) %>%
  column_to_rownames("Label2")

ps_ordination <- ordinate(ps2.NICU_no_na, method = "PCoA", distance = "bray")

evals <- ps_ordination$values$Eigenvalues

plot_ordination(ps2.NICU_no_na , ps_ordination, color = "Days_since_birth", shape = "Type",
  title = "PCoA (Bray-Curtis)") +
  coord_fixed(sqrt(evals[2] / evals[1])) +
  geom_point(size = 3.5) +
  stat_ellipse(type = "norm", linetype = 2)
```

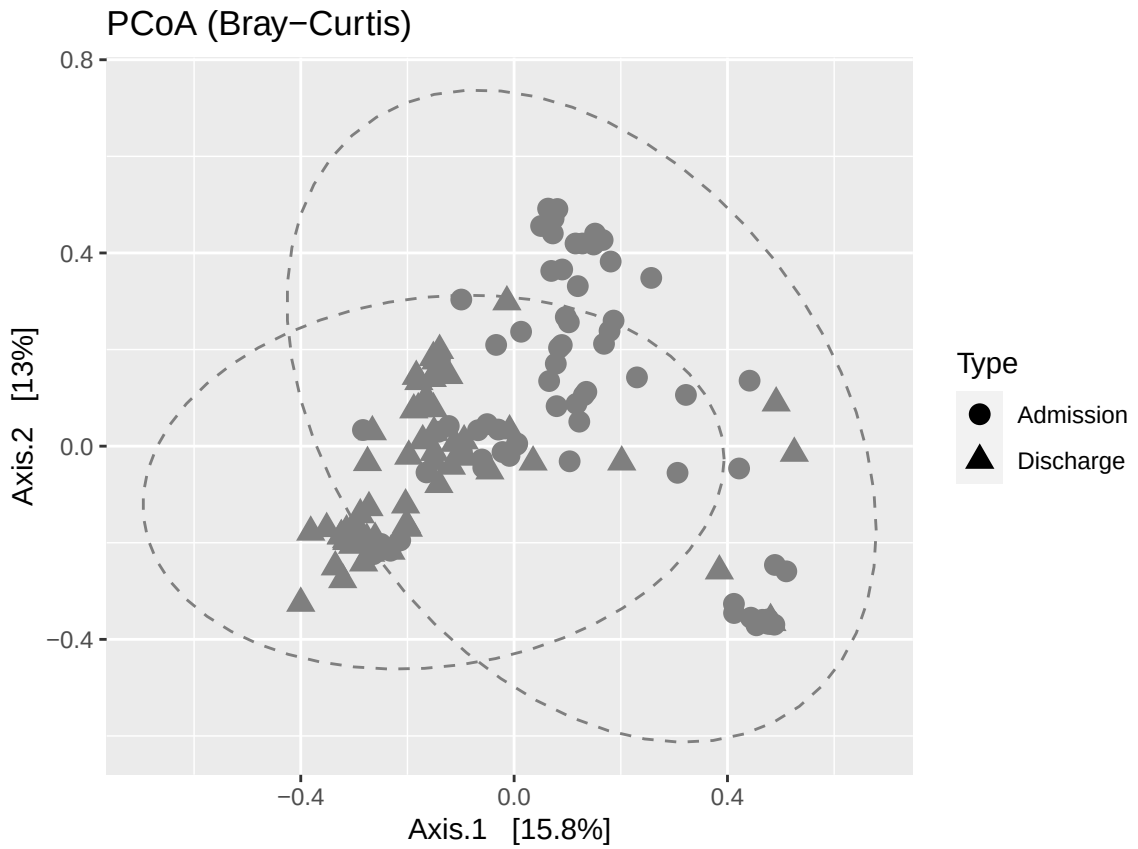


Figure 15: PCoA plot of Bray-Curtis matrix coloured by days since birth the collection occured.

- Export plots.

```
ggsave("PCoA_Weighted-Unifrac.png",
  plot = (plot_ordination(ps2.NICU_no_na , ps_ordination,
  color = "Type", title = "PCoA (Weighted-Unifrac)") +
  labs(col = "Type") +
  coord_fixed(sqrt(evals[2] / evals[1])) +
  geom_point(size = 2)+
  stat_ellipse(type = "norm", linetype = 2)), dpi = 600, height = 5, width = 5)
```

Statistical test: PERMANOVA.

- Preforming permutational anova for group-level (*Type* of sample) differences based on dissimilarity.
- Extract otu table and metadata from phyloseq object.
- Use `adonis()` from the *vegan* package to perform the PERMANOVA.

```
ps_otu <- data.frame(otu_table(ps2.NICU_no_na))
ps_samp <- data.frame(sample_data(ps2.NICU_no_na))

permanova <- adonis(ps_otu ~Type, data = ps_samp, method = "bray")
as.data.frame(permanova$aov.tab)
```

```
##            Df SumsOfSqs   MeanSqs  F.Model         R2 Pr(>F)
## Type        1   3.60625 3.6062497 9.021595 0.06397315  0.001
## Residuals 132  52.76506 0.3997353       NA 0.93602685     NA
## Total     133  56.37131        NA       NA 1.00000000     NA
```

- Significant PERMANOVA means one of three things:
- there is a difference in the location of the samples (i.e. the average community composition).
- there is a difference in the dispersion of the samples (i.e. the variability in the community composition).
- there is a difference in both the location and the dispersion.
- If you get a significant PERMANOVA you'll want to distinguish between the three options by checking the homogeneity condition using `permdisp()`. If you get a non-significant result the first option above is correct.

```
dist <- vegdist(ps_otu)
as.data.frame(anova(betadisper(dist, ps_samp$Type)))
```

```
##            Df      Sum Sq     Mean Sq  F value    Pr(>F)
## Groups      1 0.001784538 0.001784538 0.303317 0.5827413
## Residuals 132 0.776609826 0.005883408       NA        NA
```

- `betadisper()` gives a measure of the dispersion within groups. Thus, if the PERMANOVA test is significant and the permdisp is not, the significant result in your communities is due to a mean shift in community composition and not from increased variance within groups.

- Export results.

```
tab_df((as.data.frame(permanova$aov.tab)),
       alternate.rows = TRUE,
       title = "PERMANOVA: Admission vs Discharge",
       file = "PERMANOVA_ADvsDIS.doc")
```

```
tab_df((as.data.frame(anova(betadisper(dist, ps_samp$Type)))),
       alternate.rows = TRUE,
       title = "Homogeneity (PERMANOVA): Admission vs Discharge",
       file = "Homogeneity_PERMANOVA_ADvsDIS.doc")
```

- Explore the major contributors to the differences.

```
coef <- coefficients(permanova)["Type1",]
top.coef <- coef[rev(order(abs(coef)))[1:20]]

major_contributors <- tax_table(ps2.NICU_no_na) %>%
    unclass() %>%
    as.data.frame() %>%
    select("Genus", "Species") %>%
    rownames_to_column(var = "ASV") %>%
    right_join((as.data.frame(top.coef) %>%
    rownames_to_column(var = "ASV"))) %>%
    select(!"ASV")
```

- Export table.

```
tab_df(major_contributors , alternate.rows = TRUE,
       title = "Major Contriutors to PERMANOVA differences.",
       file = "Major_contributors_beta_diversity.doc")
```

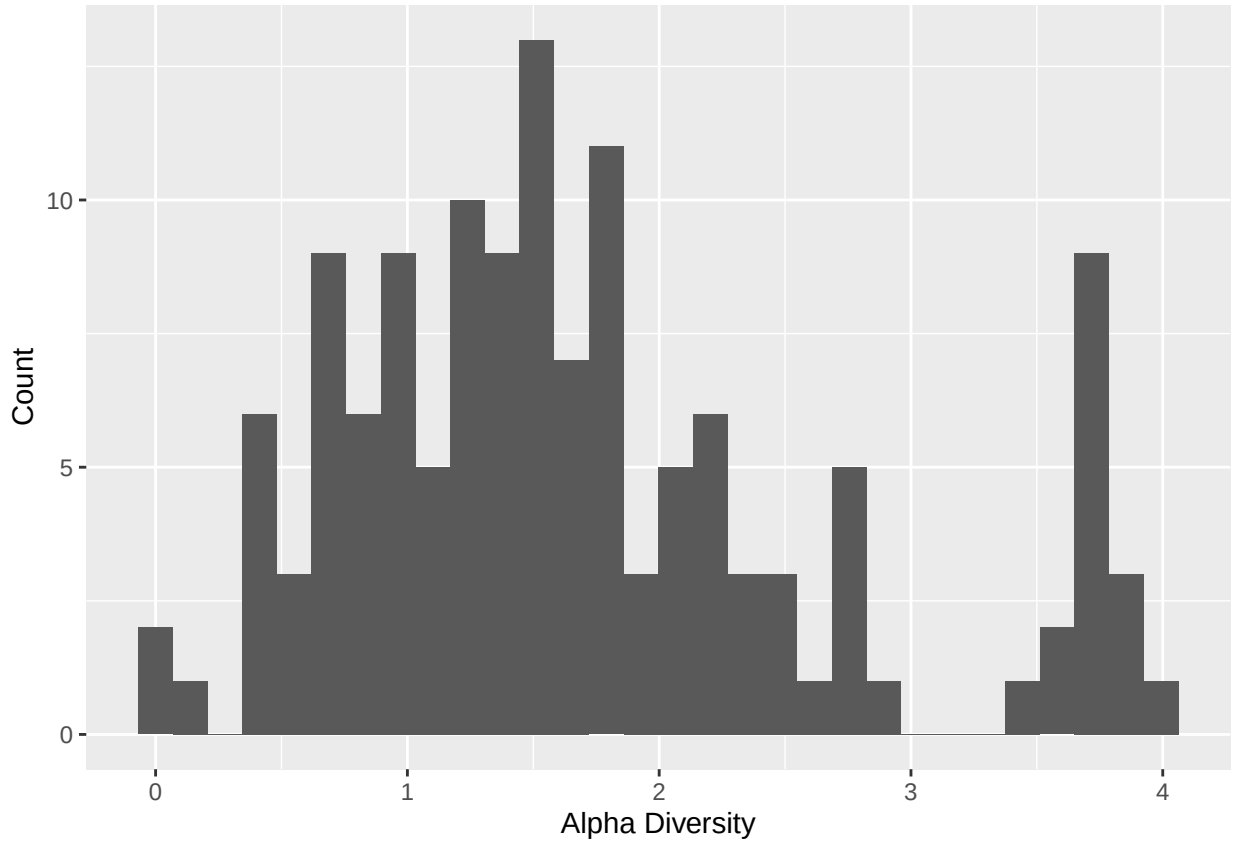
Alpha diversity.

- Subset ps2 to exclude SCN and NA values.
- Estimate richness and save as object.
- Remove chao1 standard error column and "X" from row names.
- Create a new variable column with rownames.
- Merge alpha diversity estimates (*ps_alpha_div*) with the metadata (*samdf*) by the *Label* column (orignially row names), for downstream analysis.

```
ps.NICU_no_na <- subset_samples(ps2,
              Primary_Group == "NICU" &
              (Type == "Admission" | Type == "Discharge"))

ps_alpha_div <- ps.NICU_no_na %>%
              estimate_richness(measures = c("Shannon", "Observed", "Chao1")) %>%
              subset(select =  -se.chao1)

rownames(ps_alpha_div) <- sub("X", "", rownames(ps_alpha_div))

ps_alpha_div$Label <- rownames(ps_alpha_div) %>%
              as.factor()

ps_samp <- samdf %>%
              filter(Primary_Group == "NICU" &
              (Type == "Admission" | Type == "Discharge")) %>%
              right_join(ps_alpha_div, by = "Label") %>%
              as.data.frame()
```

- Create histogram to examine distribution.

```
# To determine if diveristy is normally distributed
ggplot(ps_samp, aes(x = Shannon)) + geom_histogram() +
       xlab("Alpha Diversity") + ylab("Count")
```



- Test for normality.

```
shapiro.test(ps_samp$Shannon)
```

Statistical test: compare mean/median bteween admission and discharge samples.

- Combine the outputs.

```
Shannon <- compare_means(Shannon ~ Type, data = ps_samp,
                         method = "wilcox.test", p.adjust.method = "fdr")
Observed <- compare_means(Observed ~ Type, data = ps_samp,
                          method = "wilcox.test", p.adjust.method = "fdr")
Chao1 <- compare_means(Chao1 ~ Type, data = ps_samp,
                       method = "wilcox.test", p.adjust.method = "fdr")

Diversity_Analysis <- bind_rows(Shannon, Observed, Chao1) %>%
  rename(c(".y." = "Diversity Measure"))

Diversity_Analysis
```

```
## # A tibble: 3 x 8
##   `Diversity Measure` group1    group2         p p.adj p.format p.signif method
##   <chr>               <chr>     <chr>      <dbl> <dbl> <chr>    <chr>    <chr>
## 1 Shannon             Admission Discharge 0.191  0.19  0.19     ns       Wilcox~
## 2 Observed            Admission Discharge 0.0422 0.042 0.042    *        Wilcox~
## 3 Chao1               Admission Discharge 0.0422 0.042 0.042    *        Wilcox~
```

- Export *Diversity_Analysis* results table.

```
tab_df(Diversity_Analysis, alternate.rows = TRUE,
       title = "Diversity Analysis: Admission Vs Discharge",
       file = "Alpha_Diversity_Analysis_Type.doc")
```

Plot alpha diversity.

- Use `plot_richness()` from *phyloseq*, which estimates alpha diversity metrics using *vegan* and plots them, taking standard *ggplot2 geoms_* for the plot design.

```
plot_richness(ps.NICU_no_na, measures = c("Shannon", "Observed"),
              color = "Type", title = "") +
    geom_point(size = 3.5, alpha = 0.7) +
    theme(axis.text.x = element_blank(),
          axis.ticks.x = element_blank(),
          panel.border = element_rect(colour = "grey", fill = NA, size = 1))
```

- Export scatterplot.

```
ggsave("Alpha_Point.png", dpi = 600, height = 5, width = 5)
```

- Use `plot_richness()` to create boxplots of alpha diversity.
- To add a layer with p values use `stat_compare_means(comparisons = list(c("Admission", "Discharge")), method = "wilcox.test")`.

```
plot_richness(ps.NICU_no_na, measures = c("Shannon", "Observed"), x = "Type", color = "Type", title = "
    geom_point(size = 1, alpha = 0.7) +
    geom_boxplot() +
    theme(panel.border = element_rect(colour = "grey", fill = NA, size = 1))
```

- Export boxplot.

```
ggsave("Alpha_Box.png", dpi = 600, height = 5, width = 5)
```

- Explore the distribution of alpha diversity across the two groups using a histogram.
- Calculate mean and medians for shannon diversity to be used for dotted lines in histogram.

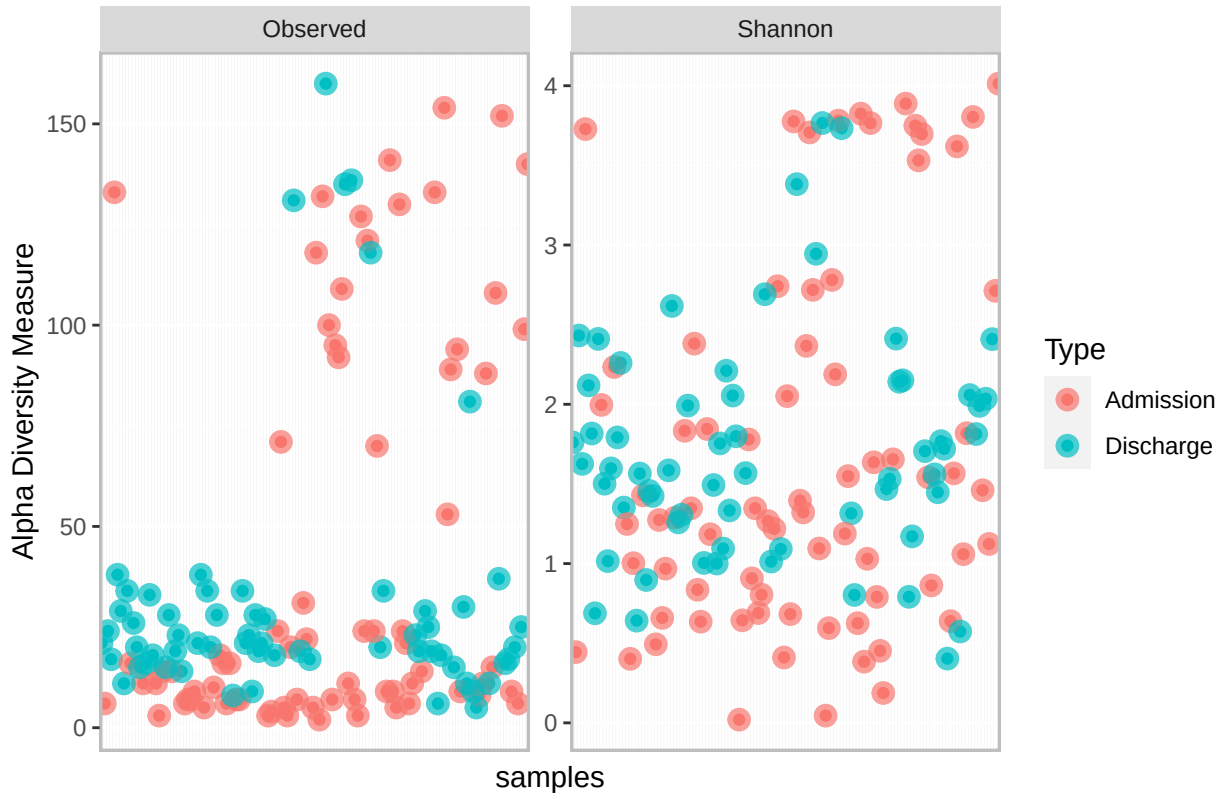


Figure 16: Scatterplot of richness and shannon diversity metrics coloured by the type of sample.

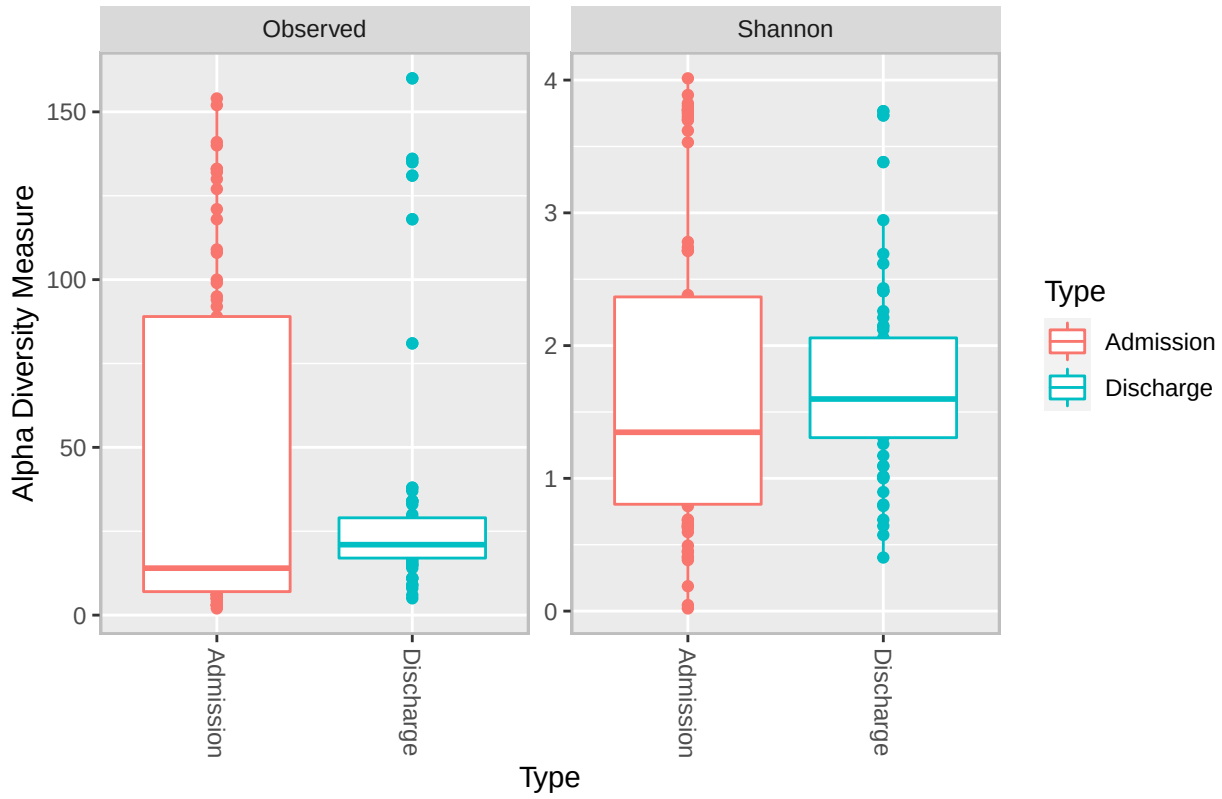


Figure 17: Boxplots of richness and shannon diversity metrics coloured by the type of sample.

```
div_med <- ddply(ps_samp, "Type", summarise, grp.med = median(Shannon))
div_mean <- ddply(ps_samp, "Type", summarise, grp.mean = mean(Shannon))

ggplot(ps_samp, aes(x = Shannon, color = Type)) +
  geom_histogram(alpha = 0.25, position="dodge") +
  labs(title = "", x = "Shannon Index", y = "Sample count") +
  geom_vline(data = div_mean, aes(xintercept = grp.mean, color = Type),
             linetype = "dashed") +
  geom_vline(data = div_med, aes(xintercept = grp.med, color = Type),
             linetype = "solid") +
  theme(panel.border = element_rect(colour = "grey", fill = NA, size = 1))
```

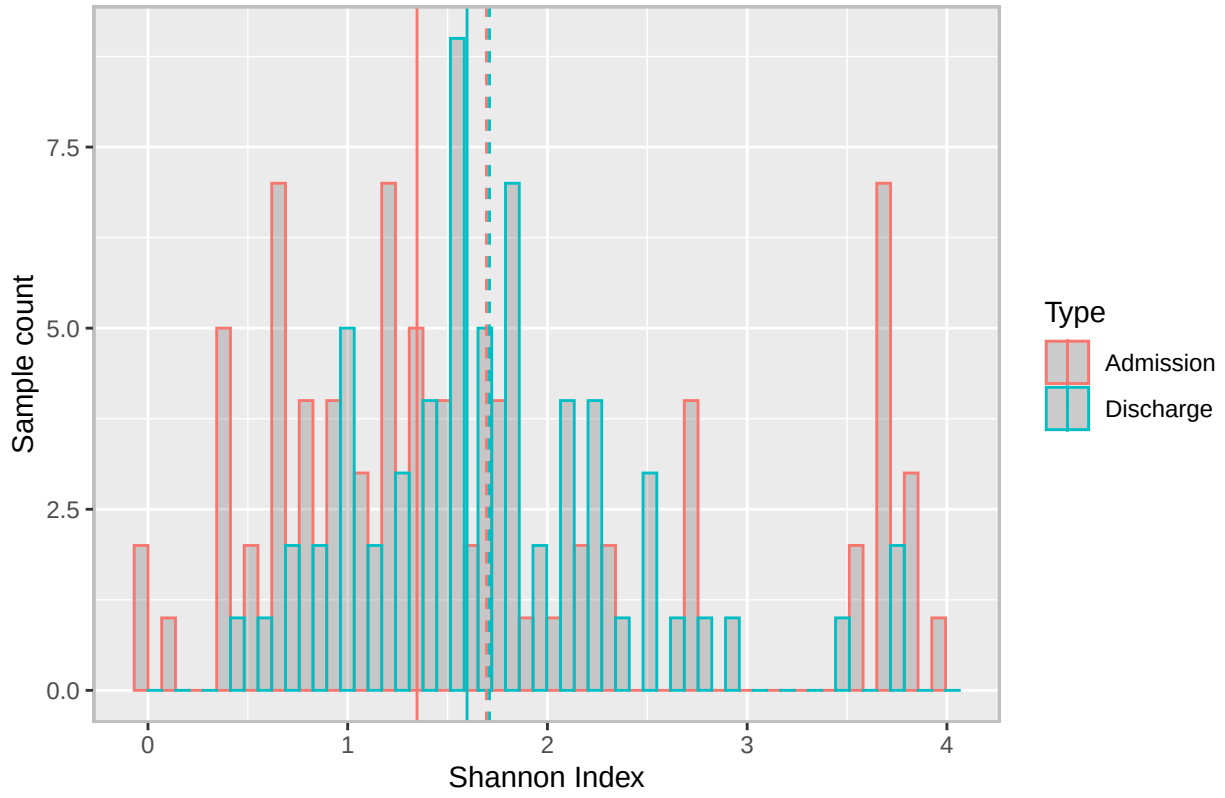


Figure 18: Histogram showing the distribution of the shannon index scores across samples, coloured by sample type and with lines representing the mean (dashed) and median (solid).

- The admission outliers in red may explain why were not seeing the significant differences in diversity.
- Export histogram.

```
ggsave("Alpha_Distribution.png", dpi = 600, height = 5, width = 10)
```

Taxonomic abundance with *DESeq2*.

- Subset to filtered/agglomerated data.

- Convert from *phyloseq* to *deseq* object.
- To perform analysis at other levels of taxonomy use `tax_glom(ps2, "Phylum", NArm = TRUE)` prior to running the chunk below.

```
ps.NICU <- subset_samples(ps3, Primary_Group == "NICU" &
        (Type == "Admission" | Type == "Discharge"))

ps.NICU.deseq = phyloseq_to_deseq2(ps.NICU, ~Type)
```

- Define function for calculating geometric means.
- Calculate geometric means, and subsetuently estimate size factors.

```
gm_mean = function(x, na.rm = TRUE){
  exp(sum(log(x[x > 0]), na.rm = na.rm) / length(x))
}

geoMeans <- apply(counts(ps.NICU.deseq), 1, gm_mean)

ps.NICU.deseq <- estimateSizeFactors(ps.NICU.deseq, geoMeans = geoMeans)
```

- Construct histograms to compare pre and post transformation.
- Call `estimateDispersions()` to calculate abundances with `getVarianceStabilizedData()`.
- **NB.** the samples are in columns in the *deseq* object but in rows for the *phyloseq* object.
- Axis adujsted for what best represents the distribution.

```
ps.NICU.deseq <- estimateDispersions(ps.NICU.deseq, fitType = "local")

abund_sums_trans <- data.frame(sum = colSums(getVarianceStabilizedData(ps.NICU.deseq) ),
                     sample = colnames(getVarianceStabilizedData(ps.NICU.deseq) ),
                     type = "DESeq2")

abund_sums_no_trans <- data.frame(sum = rowSums(otu_table(ps.NICU)),
                       sample = rownames(otu_table(ps.NICU)),
                       type = "None")

grid.arrange((ggplot(abund_sums_trans) +
  geom_histogram(aes(x = sum), binwidth = 1) +
  xlab("Abundance within sample") +
  xlim(NA, 300) +
  ylim(0,6) +
  ggtitle("DESeq2 transformation")),
  (ggplot(abund_sums_no_trans) +
  geom_histogram(aes(x = sum), binwidth = 200) +
  xlab("Abundance within sample") +
  ylim(0,5) +
  ggtitle("No transformation")),
  nrow = 2)
```

Statisical test: calculate differential abundances with *DESeq2*.

- Use `DESeq()` to perform differential expression analysis based on the negative binomial distribution.

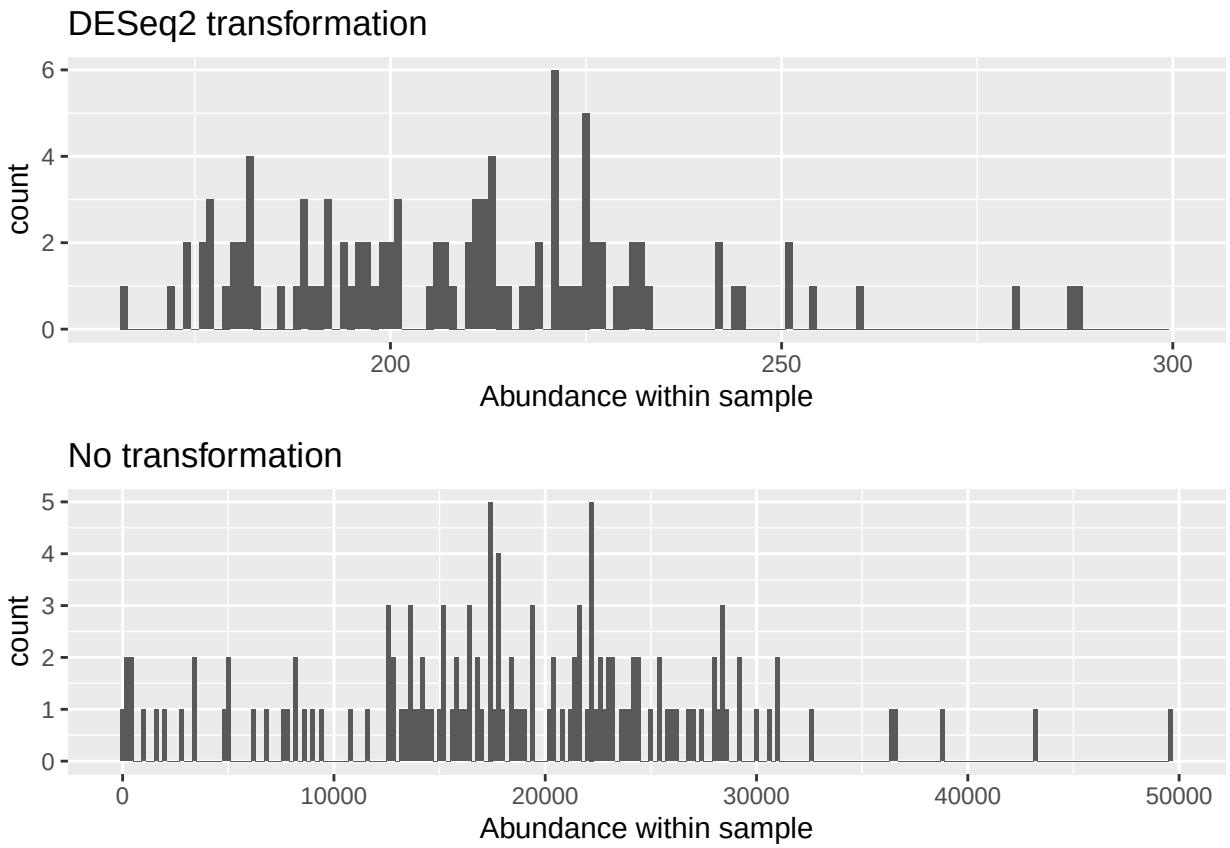


Figure 19: Pre and post transformation of taxonomic counts with DESeq2

- The function estimates size factors, estimates dispersion, fits a negative binomial GLM and performs a Wald test.
- Extract the results, order by p value, select only significant (<0.05) results, bind this data to the *tax_table* from the *phyloseq* object to get the taxonomic information, and then select and order the desired columns.

```
ps.NICU.deseq = DESeq(ps.NICU.deseq, fitType = "local")
res = results(ps.NICU.deseq, contrast = c("Type", "Discharge", "Admission"))
res = res[order(res$padj, na.last = NA), ]
alpha = 0.01
sigtab = res[(res$padj < alpha), ]
sigtab = cbind(as(sigtab, "data.frame"),
         as(tax_table(ps.NICU)[rownames(sigtab), ], "matrix"))

sigtab %>%
  select("baseMean", "log2FoldChange", "lfcSE", "padj",
  "Phylum", "Class", "Order", "Family", "Genus") %>%
  remove_rownames()  %>%
  kable()
```

baseMean	log2FoldChange	lfcSE	padj	Phylum	Class	Order	Family	Genus
25516.904014	-7.182048	0.587010	0.000000	Firmicutes	Bacilli	Bacillales	Staphylococcaceae	Staphylococcus
218.247032	4.658349	1.008420	0.000263	Firmicutes	Bacilli	Lactobacillales	Lactobacillaceae	Lactobacillus
8.236649	5.947260	1.318757	0.000296	Firmicutes	Clostridia	Clostridiales	Clostridiaceae_1	Clostridium_sensu_stricto_1
30.377169	5.398122	1.318815	0.001457	Proteobacteria	Gammaproteobacteria	Enterobacteriales	Enterobacteriaceae	Enterobacter
71.185568	3.949979	1.061284	0.005418	Firmicutes	Negativicutes	Selenomonadales	Veillonellaceae	Veillonella

- Export results.

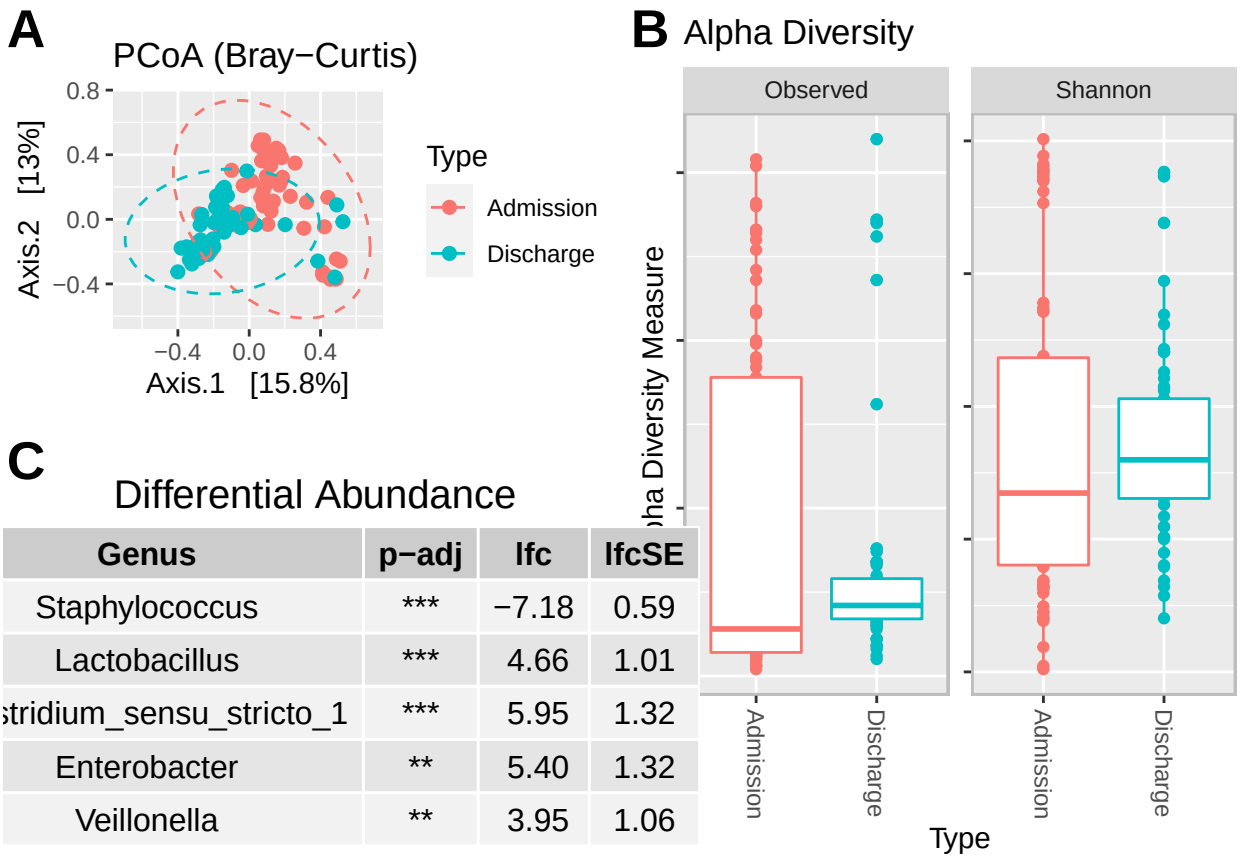
```
tab_df(posigtab, alternate.rows = TRUE,
       title = "Differential Abundance of Genera: Admission Vs Discharge",
       file = "Diff_Abundance_Genera.doc")
```

Summary.

- Create a summary grid including alpha and beta diversity metrics, as well as differential abundance testing results.

```
#PCoA Plot
ps_ordination <- ordinate(ps2.NICU_no_na , method = "PCoA", distance ="bray")

evals <- ps_ordination$values$Eigenvalues

PCoA_plot <- plot_ordination(ps2.NICU_no_na, ps_ordination, color = "Type") +
             coord_fixed(sqrt(evals[2] / evals[1])) +
             labs(col = "Type", title = "PCoA (Bray-Curtis)") +
             geom_point(size = 2) +
             stat_ellipse(type = "norm", linetype = 2)
```

```
PCoA_plot <- annotate_figure(PCoA_plot, fig.lab = "A",
                             fig.lab.face = "bold", fig.lab.size = 20)

# Alpha Diversity Plot
alpha_plot <- plot_richness(ps.NICU_no_na, measures = c("Shannon", "Observed"),
              x = "Type", color = "Type", title = "Alpha Diversity") +
              geom_point(size = 1, alpha = 0.7) +
              geom_boxplot() +
              theme(panel.border = element_rect(colour = "grey", fill = NA, size = 1),
              legend.position = "none", axis.text.y=element_blank())

alpha_plot <- annotate_figure(alpha_plot, fig.lab = "B",
                              fig.lab.face = "bold", fig.lab.size = 20)

# Differenital Abundance
title <- textGrob("Differential Abundance", gp = gpar(fontsize = 15))

padding <- unit(5,"mm")

genus_df <- as.data.frame(sigtab) %>%
            remove_rownames() %>%
            add_column("p-adj" = c("***", "***", "***", "**", "**")) %>%
            select("Genus","p-adj", "log2FoldChange","lfcSE") %>%
            mutate_if(is.numeric, round, 2) %>%
            dplyr::rename("lfc" = log2FoldChange) %>%
            tableGrob(rows = NULL) %>%
            gtable_add_rows(heights = grobHeight(title) + padding, pos = 0) %>%
            gtable_add_grob(title, 1, 1, 1, 4)

genus_df <- annotate_figure(genus_df, fig.lab = "C",
                            fig.lab.face = "bold", fig.lab.size = 20)

# grid layout
lay <- rbind(c(1,2),
             c(3,2))

grid.arrange(PCoA_plot, alpha_plot, genus_df, nrow = 2, layout_matrix = lay)
```



C

Differential Abundance

Genus	p−adj	lfc	lfcSE
Staphylococcus	***	−7.18	0.59
Lactobacillus	***	4.66	1.01
stridium_sensu_stricto_1	***	5.95	1.32
Enterobacter	**	5.40	1.32
Veillonella	**	3.95	1.06

Finished