Workflow for *Characterising the bacterial gut microbiome of probiotic-supplemented very-preterm infants*

Jacob Westaway

Last updated on 2020-12-28

Contents

1 About.

This document contains the workflow for the manuscript *Characterising the bacterial gut microbiome of probiotic-supplemented very-preterm infants*, and includes the bioinformatics pipeline to go from raw reads to interpretable abundances, based largely around this DADA2 workflow developed by *Callahan, et al.*) worfklow, removal of contamination with MicroDecon, and the analysis using a combination of the packages phloseq, DESeq2, lme4, amongst many others.

2 Bioinformatics Pipeline.

2.1 About.

Creating an ASV table from raw reads, using DADA2.

2.2 Load required packages.

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

2.3 Read quality.

2.3.1 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))
```

2.3.2 Extract sample names.

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

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

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

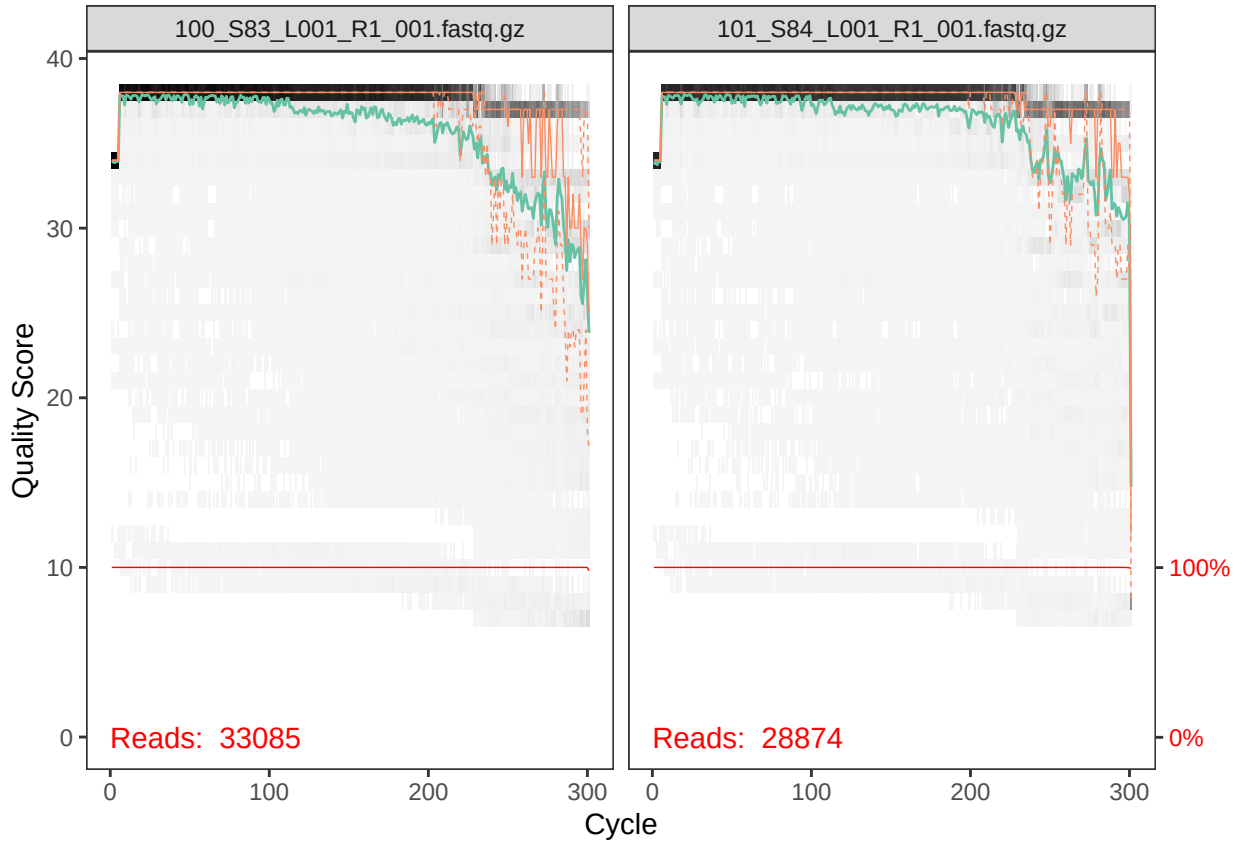


Figure 1: Quality of forward reads.

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

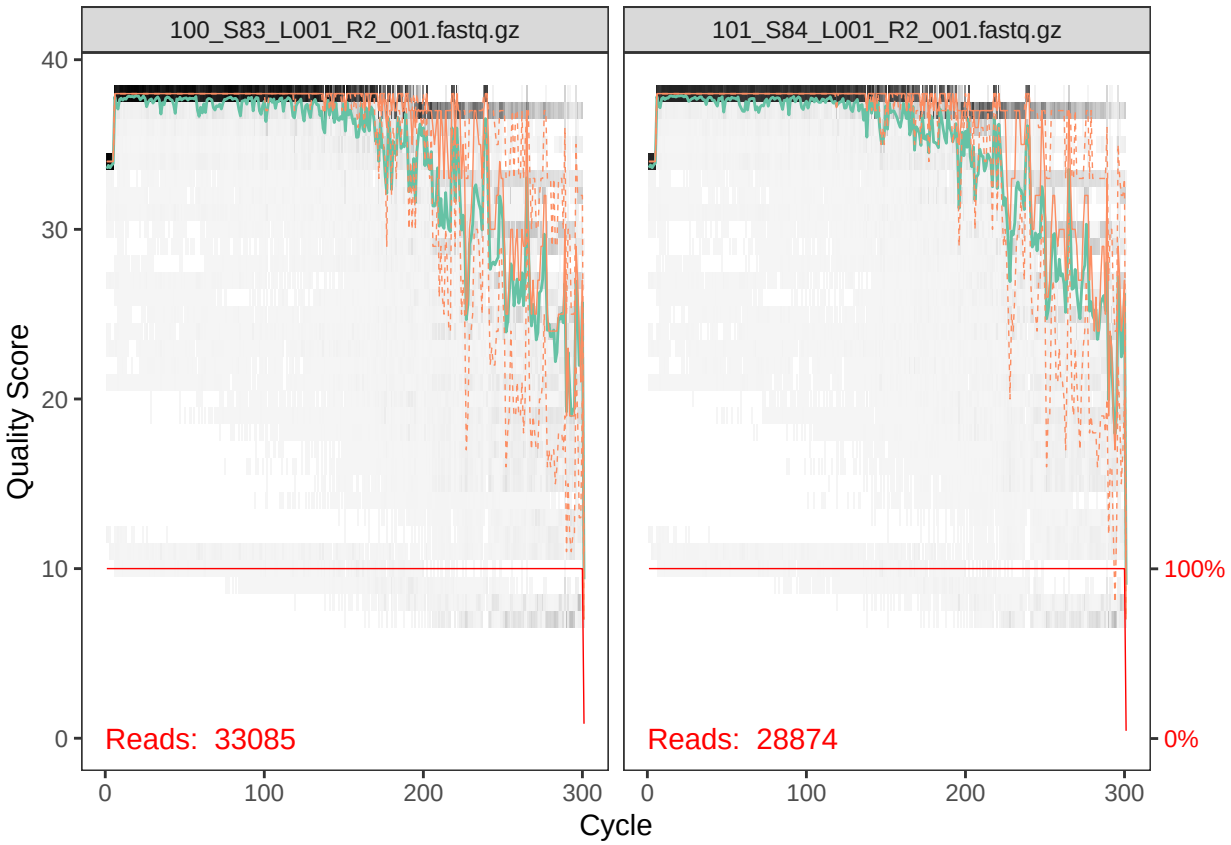


Figure 2: Quality of reverse reads.

2.3.4 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"))
```

2.3.5 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).
- `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)
```

```
##                                reads.in reads.out
## 100_S83_L001_R1_001.fastq.gz      33085     29743
## 101_S84_L001_R1_001.fastq.gz      28874     26864
## 102_S98_L001_R1_001.fastq.gz      26505     24330
## 103_S99_L001_R1_001.fastq.gz      28695     26113
## 104_S100_L001_R1_001.fastq.gz     23909     21872
## 105_S101_L001_R1_001.fastq.gz     25749     23289
```

2.4 Infer sequence variants.

2.4.1 Calculate Error Rates.

- Error rates are used for sample ineference downstream.

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

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

2.4.2 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)
```

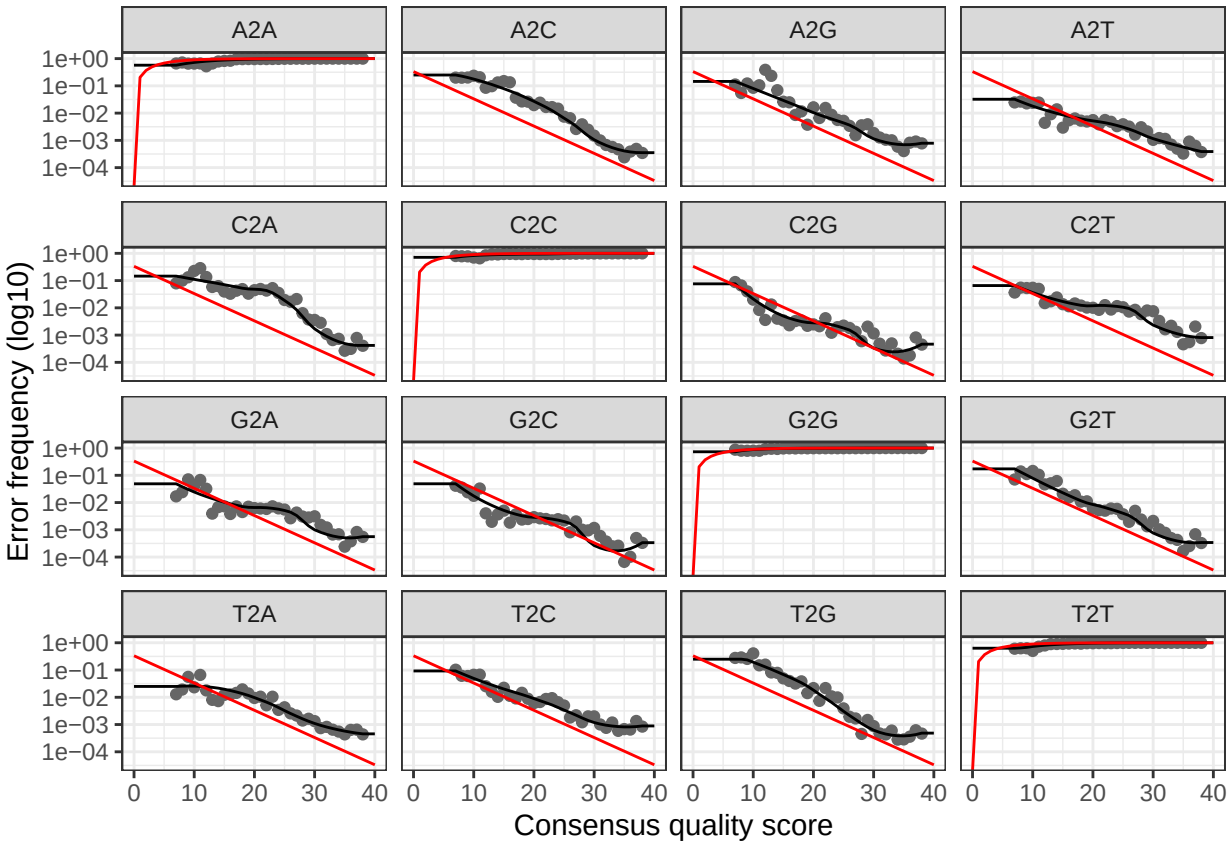


Figure 3: Error rates for forward reads

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

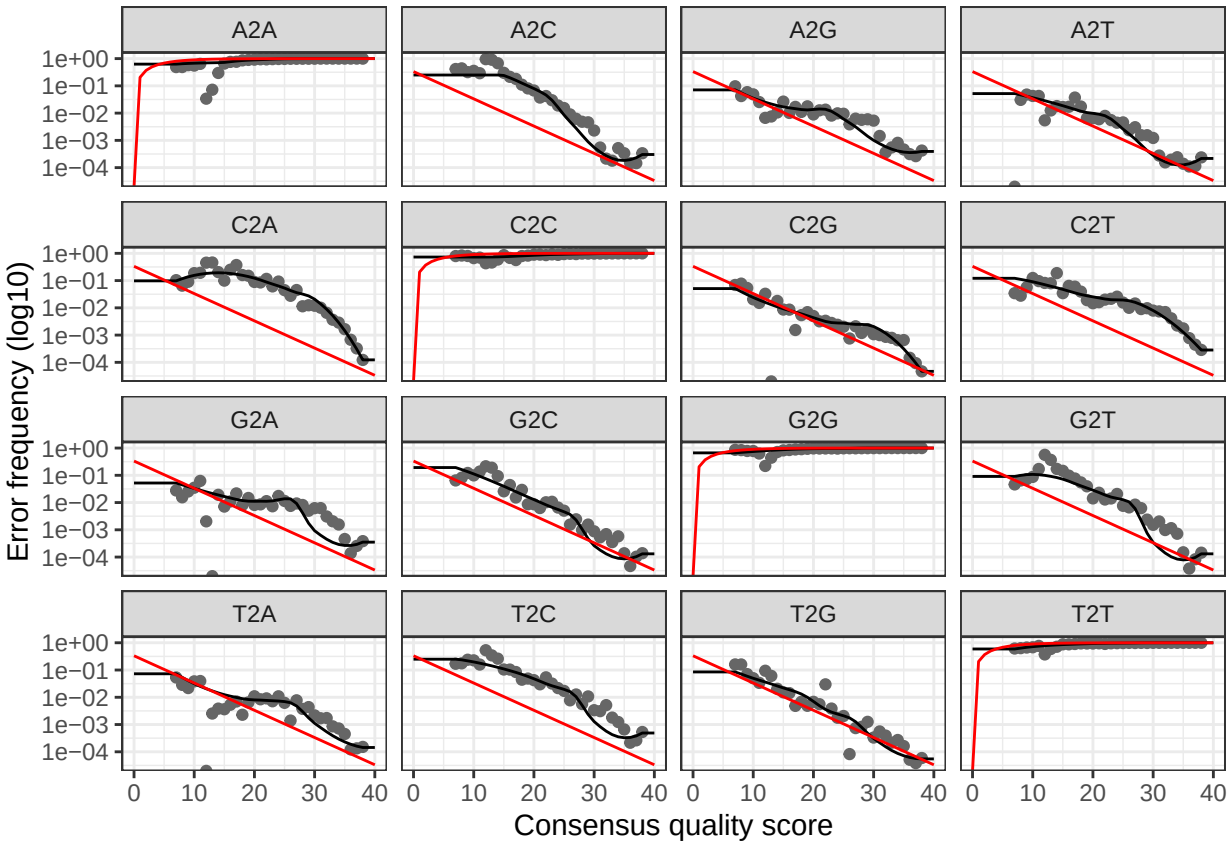


Figure 4: Error rates for reverse reads.

2.4.3 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
```

2.4.4 Sample Inference.

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

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

2.4.5 Inspect denoised data.

```
dadaFs[[1]]

dadaRs[[1]]
```

2.4.6 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)
```

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

2.5.1 Construct ASV table.

- Check dimentions and inspect distribution of sequence lengths.

```
seqtab <- makeSequenceTable(mergers)

dim(seqtab)

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

2.5.2 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)
```

2.5.3 Remove chimeras.

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

2.5.4 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    29743     29381     29618  28771   14682
## 101 28874    26864     26744     26801  26510   20950
## 102 26505    24330     24167     24289  24040   15006
## 103 28695    26113     26041     26080  25958   18064
## 104 23909    21872     21833     21859  21795   21785
## 105 25749    23289     22546     22953  20782   11160
```

2.6 Contamination removal with *MicroDecon*.

```
library(microDecon)
```

2.6.1 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)`.

2.6.2 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))]
```

2.6.2.1 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)
```

2.6.3 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)
```

2.6.3.1 Check *MicroDecon* Outputs.

```
decontaminated$decon.table
decontaminated$reads.removed
decontaminated$OTUs.removed
decontaminated$mean.per.group
decontaminated$sum.per.group
```

2.6.4 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)
```

2.7 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
```

3 Preprocessing: Creating a Phyloseq Object.

3.1 About.

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

3.2 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"),
    require, character.only = TRUE)
```

3.3 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
```

3.4 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")
```

3.5 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))
```

3.6 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)
```

3.7 Filtering and normalisation.

3.7.1 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"))
```

3.7.2 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")
```

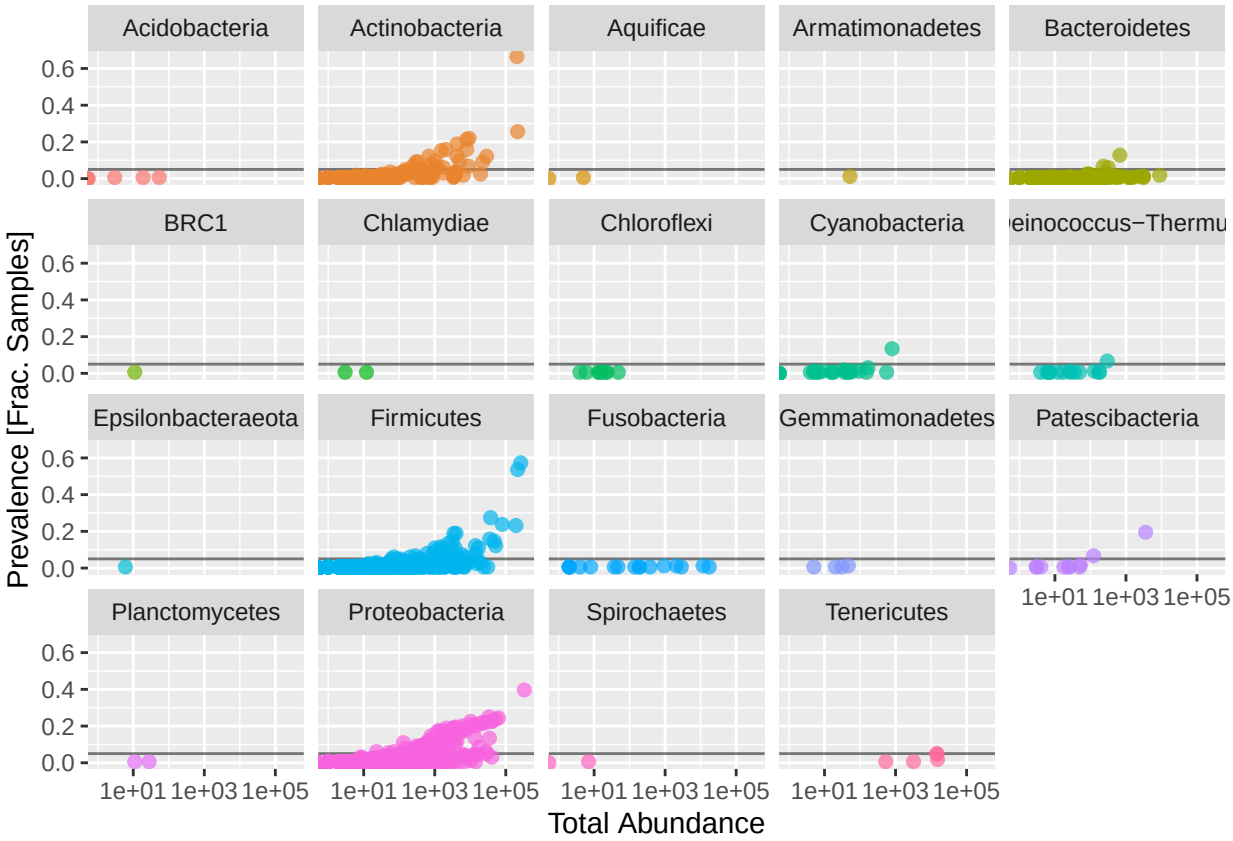


Figure 5: Scatterplot exploring the relationship between prevelance and abundance of phyla.

- 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")
```

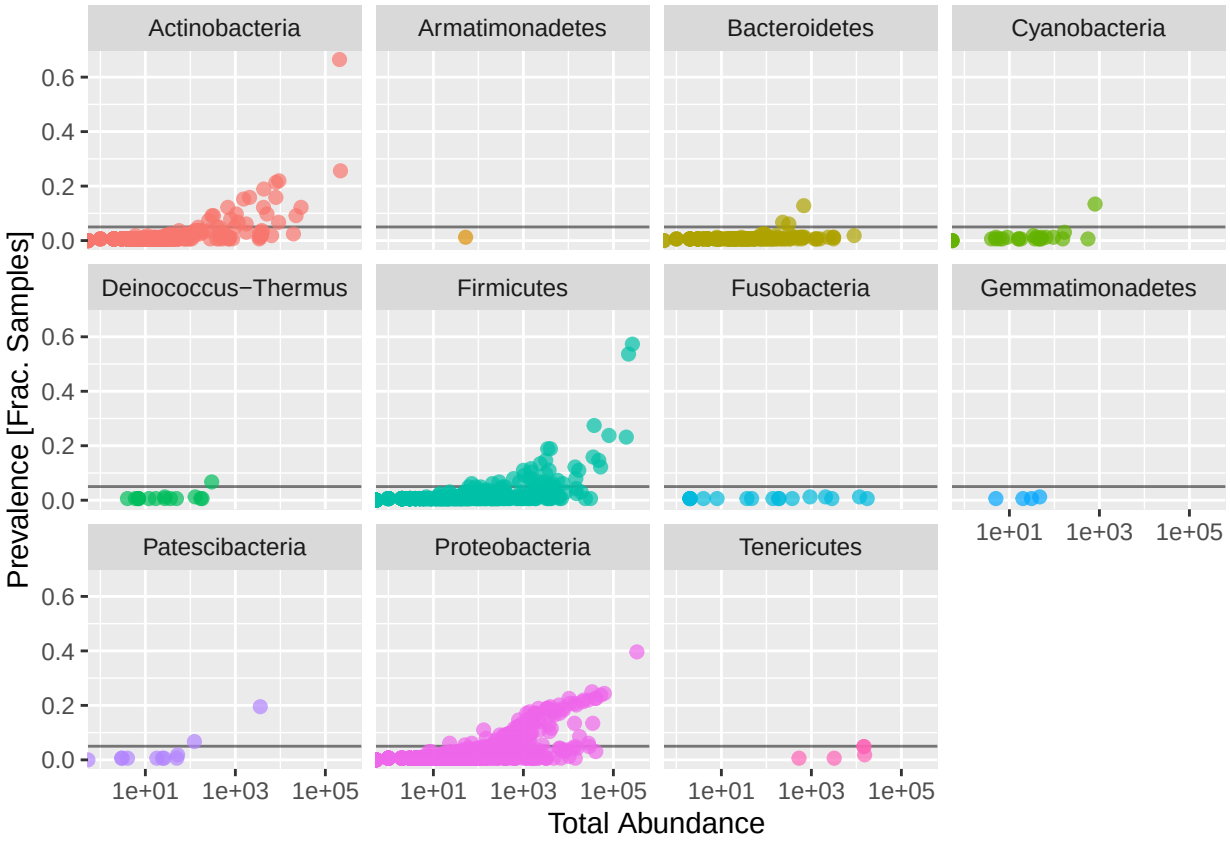


Figure 6: Scatterplot exploring the relationship between prevelance and abundance of phyla on data passed through a prevalence threshold.

3.7.3 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))))
```

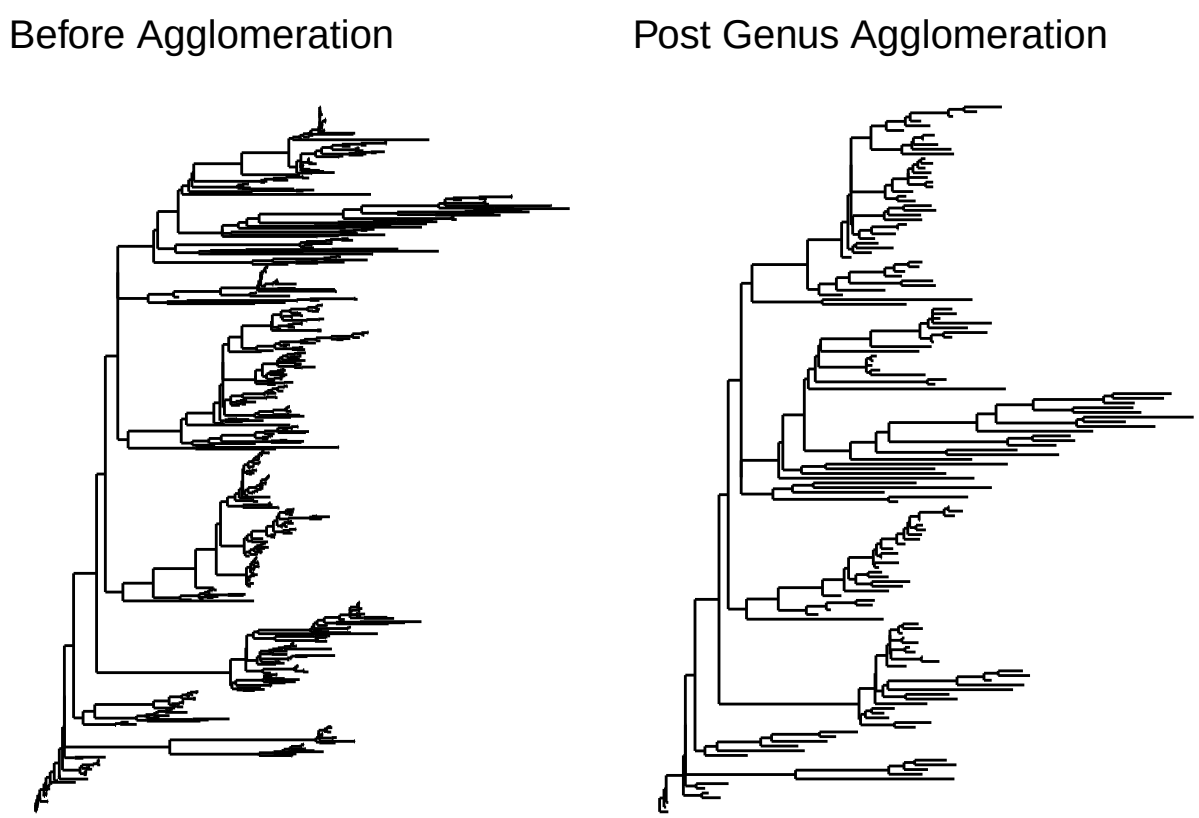
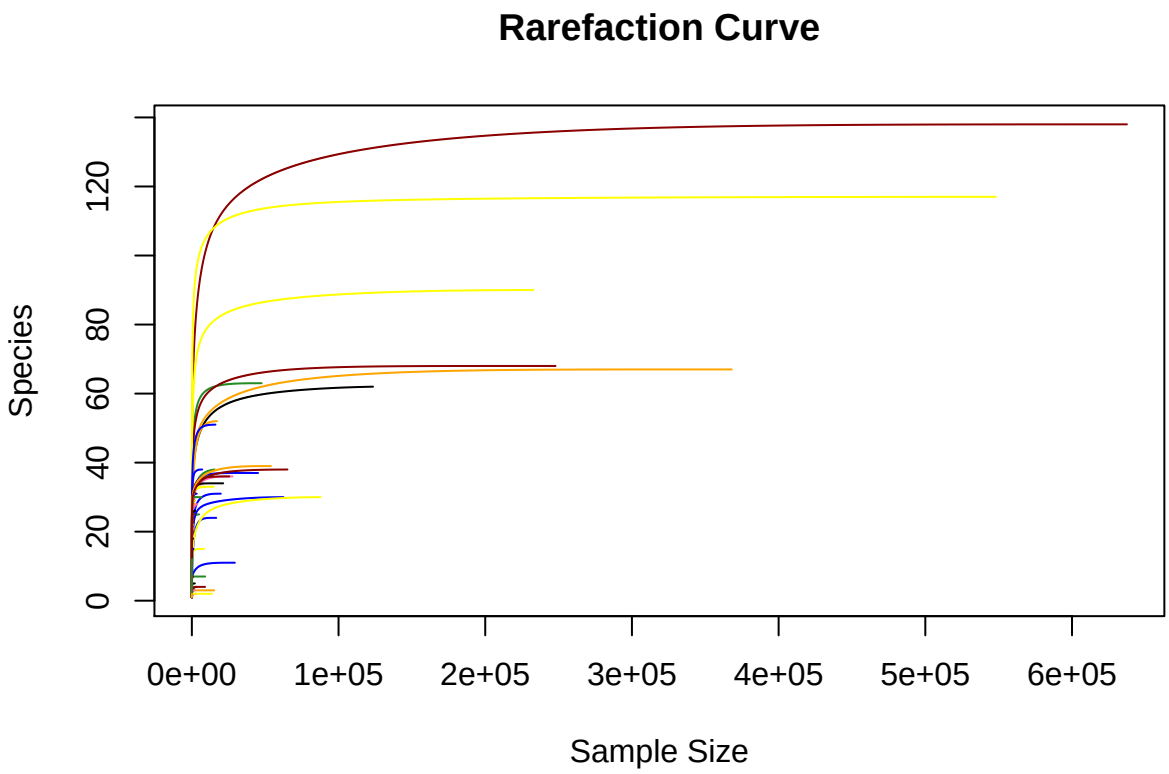


Figure 7: Tree plots exploring the agglomeration of taxa at the genus level.

3.7.4 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))
```

3.7.5 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))
```

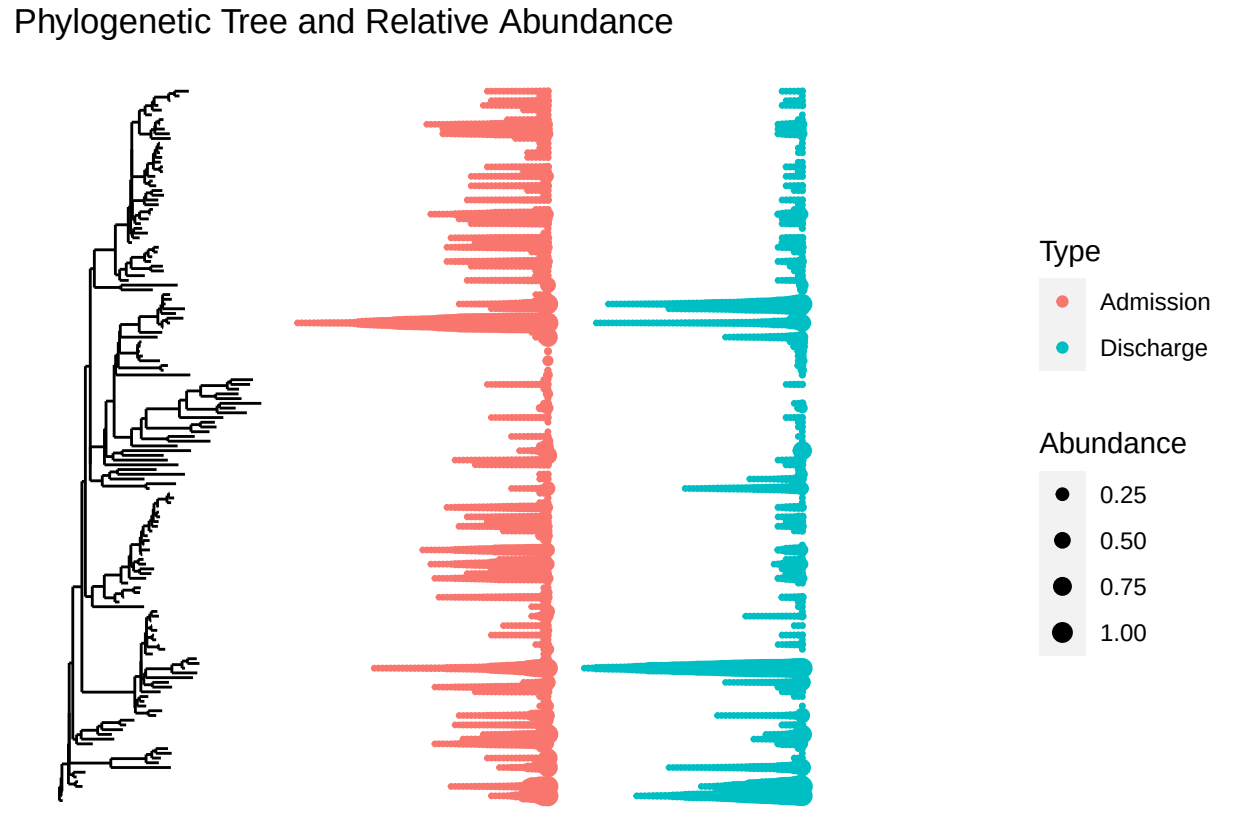


Figure 8: Tree plot exploring the normalised distribution of taxa between admission and discharge samples.

- Explore normalisation with violin plots.
- 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"){
    subset_taxa(physeq, Phylum %in% c("Firmicutes")) %>%
    psmelt() %>%
    subset(Abundance > 0) %>%
    ggplot(mapping = aes_string(x = "Type", 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")),
             plot_abundance(ps4, Title = "Relative Abundance", Color = "Type"))
```

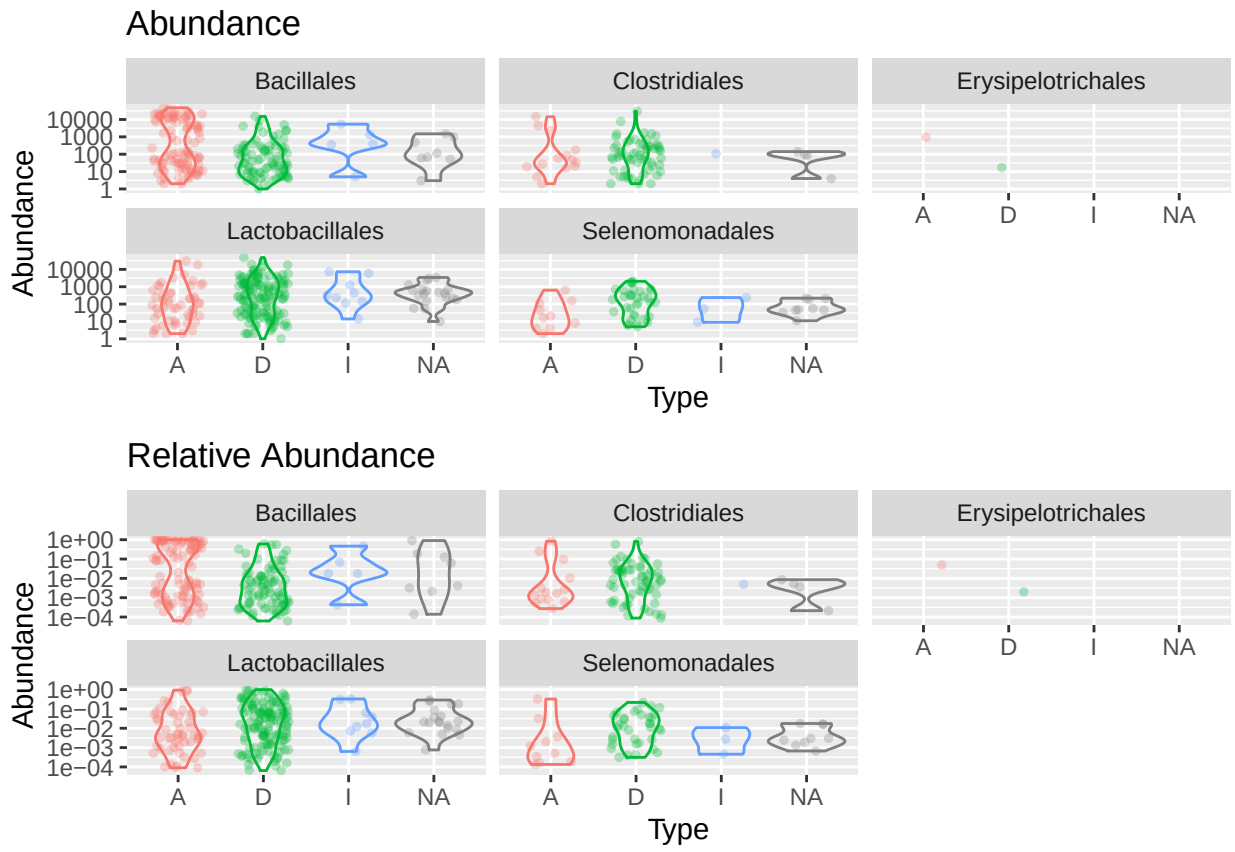


Figure 9: Violin plots exploring of distribution of abundance in Firmicutes before and after normalisation of data. Annotation for x axis; A: Admission, D: Discharge & I: Intermediate.

4 Data Exploration and Univariate Analysis.

4.1 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.

4.2 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)
```

4.3 Taxanomic distribution.

4.3.1 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")
```

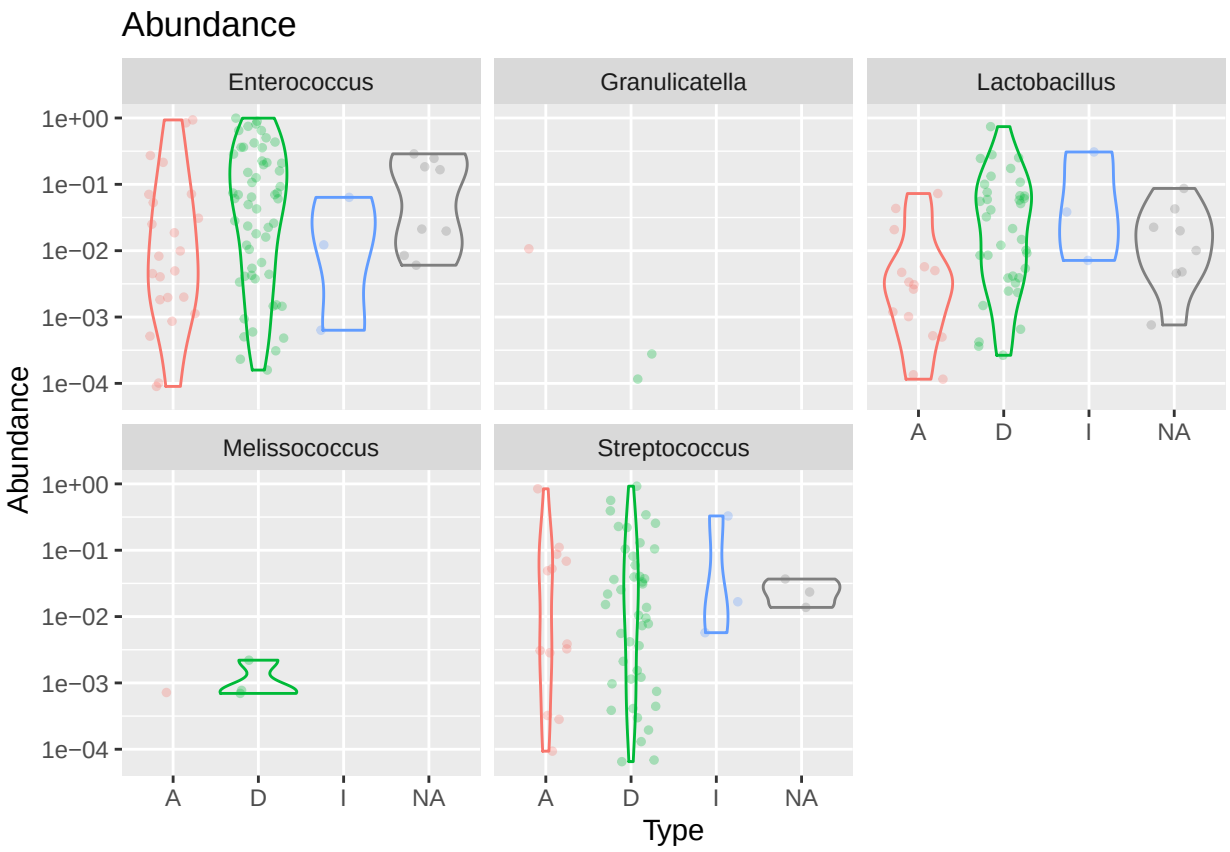


Figure 10: Violin plots exploring of distribution of abundances within Lactoacillales. Annotation for x axis; A: Admission, D: Discharge & I: Intermediate.

4.3.2 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))
```

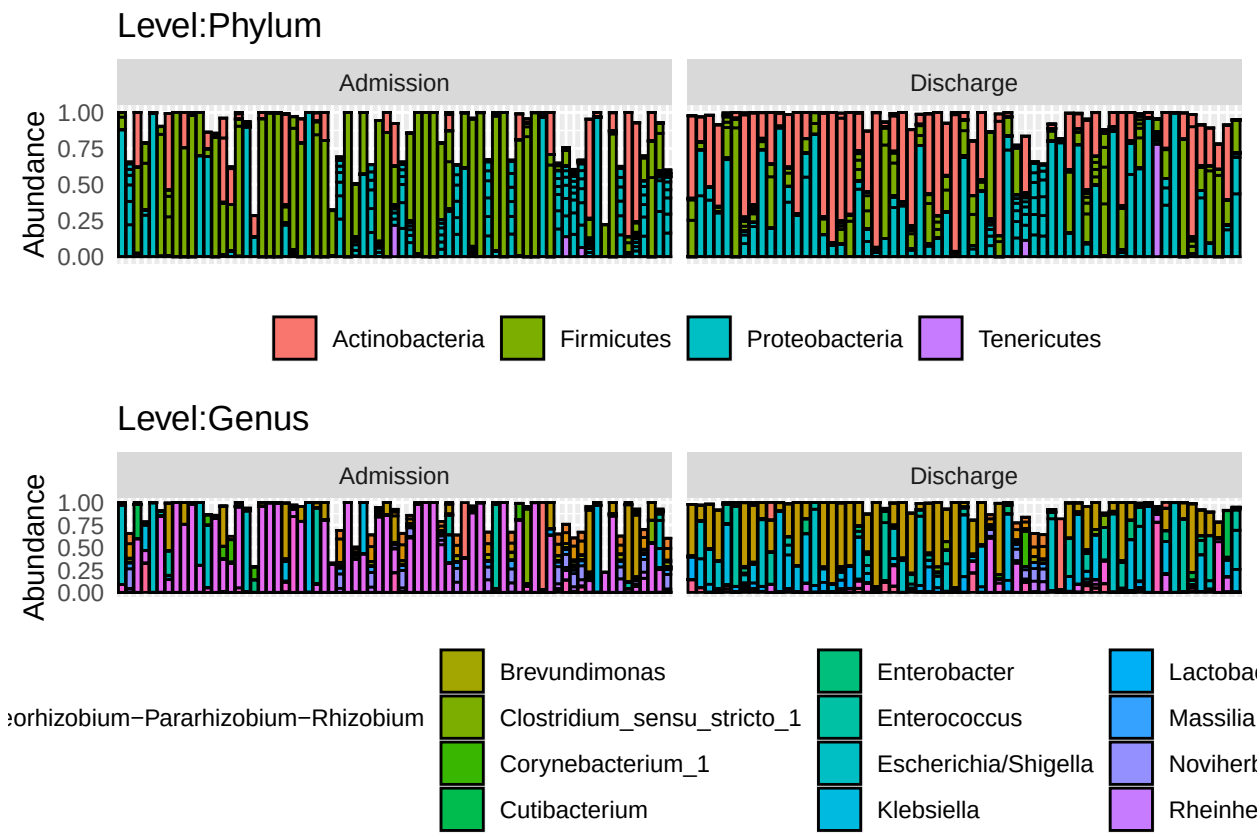


Figure 11: Bar plots of the taxonomic distribution (relative abundance) at phylum and genus levels.

- 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)
```

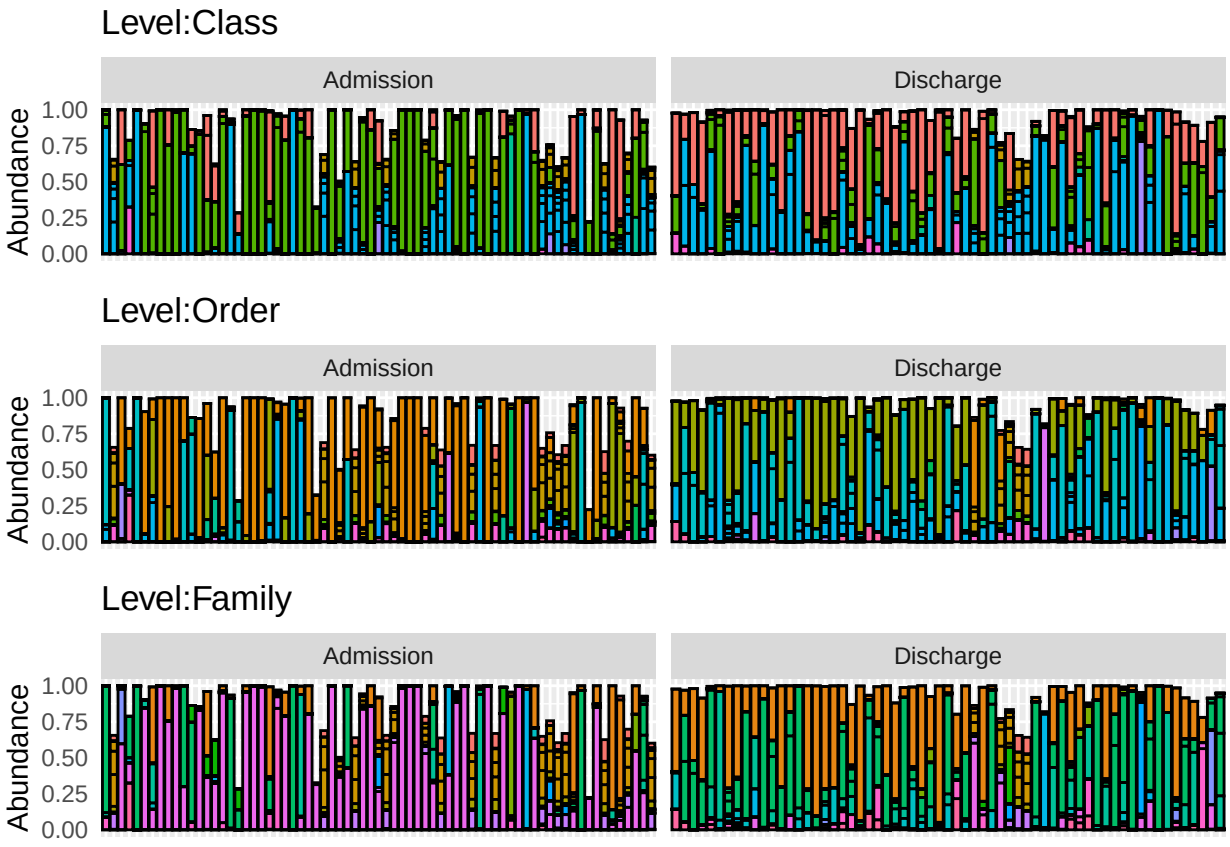


Figure 12: Bar plots of the taxonomic distribution (relative abundance) at class, order and family levels.

4.3.3 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")
```

```
## [1] 99
```

4.4 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*.

4.4.1 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)
```

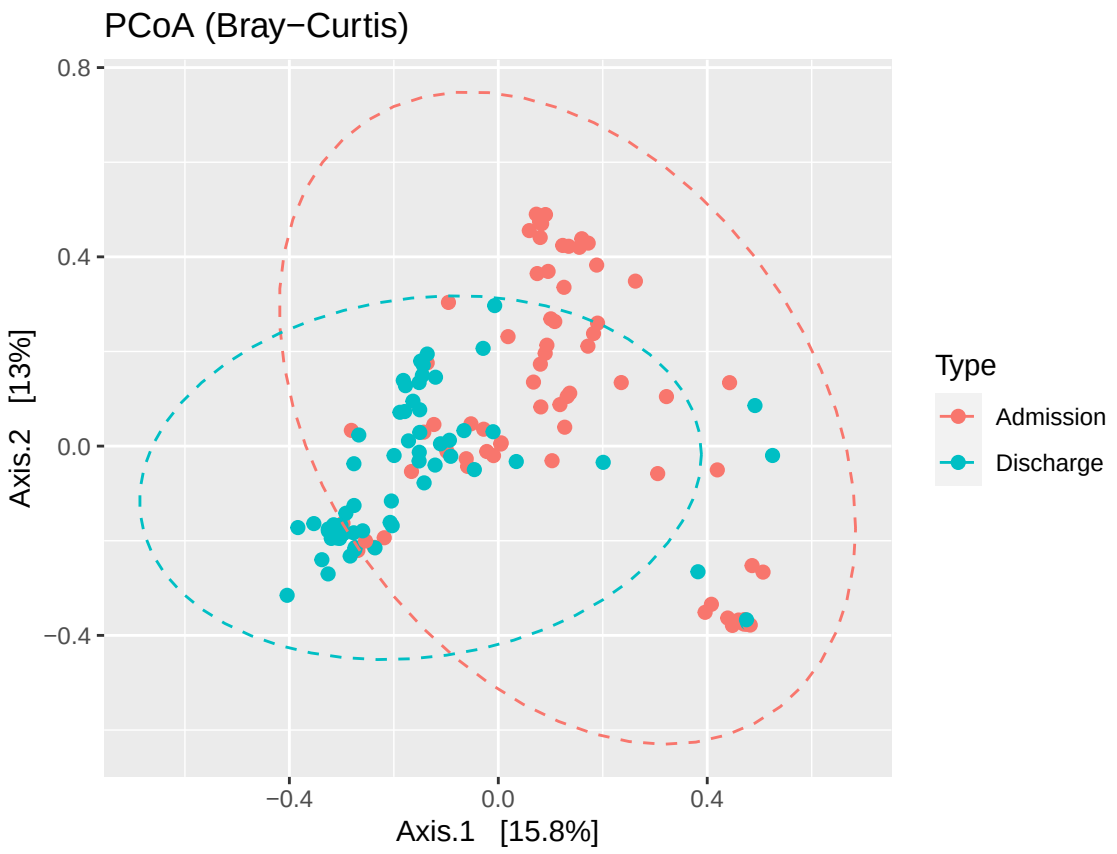


Figure 13: PCoA plot of Bray-Curtis matrix

4.4.2 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)
```

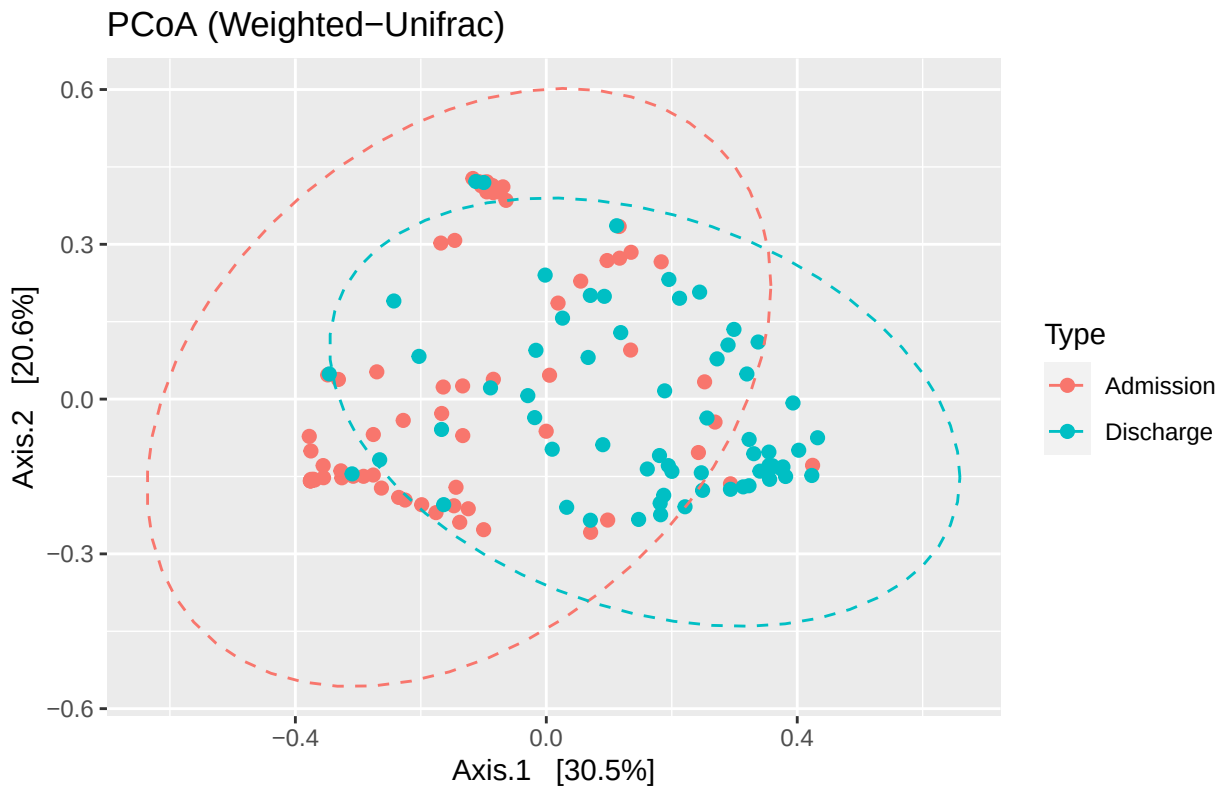


Figure 14: PCoA plot of Weighted-Unifrac matrix

- 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)
```

4.4.3 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  3.652507 3.6525066 9.143081 0.06477881  0.001
## Residuals 132 52.731770 0.3994831       NA 0.93522119     NA
## Total     133 56.384277        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 0.0002303847 0.0002303847 0.03988535 0.8420122
## Residuals 132 0.7624550219 0.0057761744         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")
```

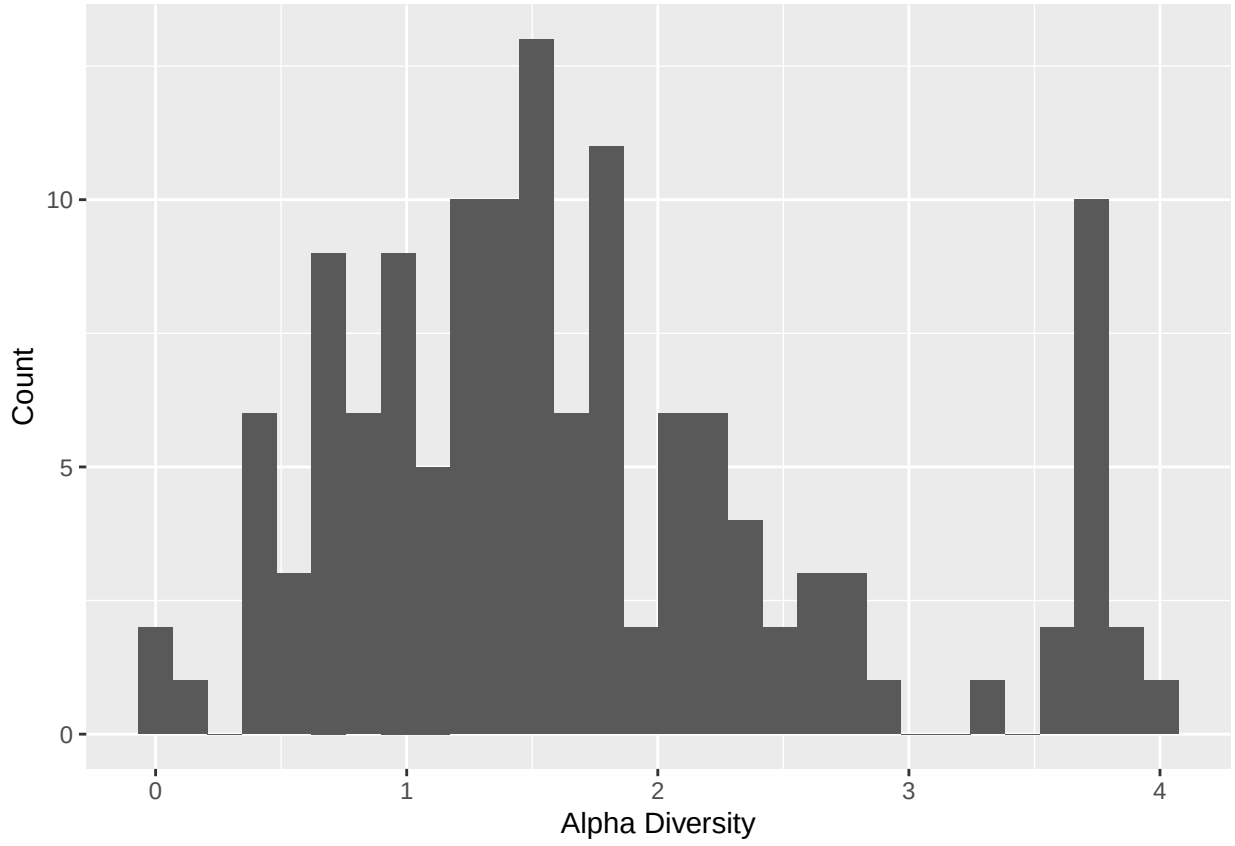
4.5 Alpha diversity.

- 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)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ps_samp$Shannon
## W = 0.92725, p-value = 2.148e-06
```

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 0.232  0.23  0.23     ns       Wilcox~
## 2 Observed            Admission Discharge 0.0692 0.069 0.069    ns       Wilcox~
## 3 Chao1               Admission Discharge 0.0699 0.07  0.07     ns       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")
```

4.5.1 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))
```

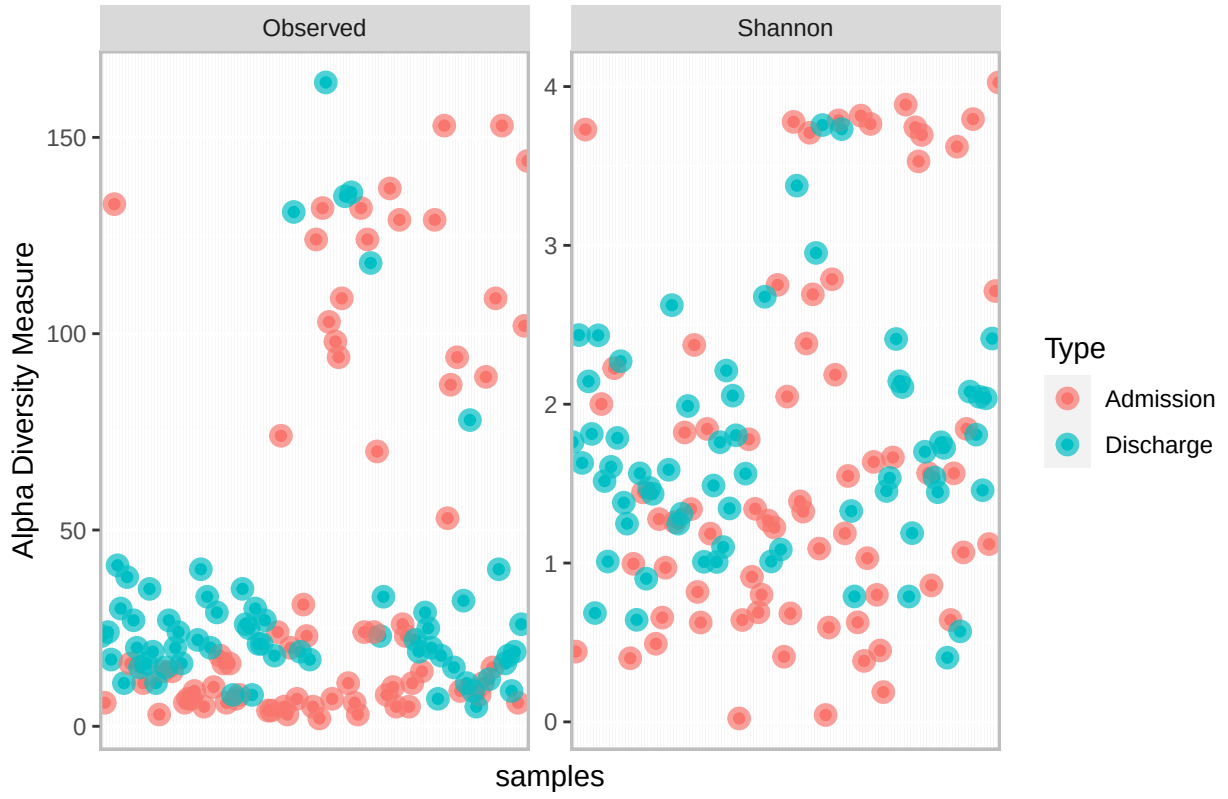


Figure 15: Scatterplot of richness and shannon diversity metrics coloured by the type of sample.

- 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))
```

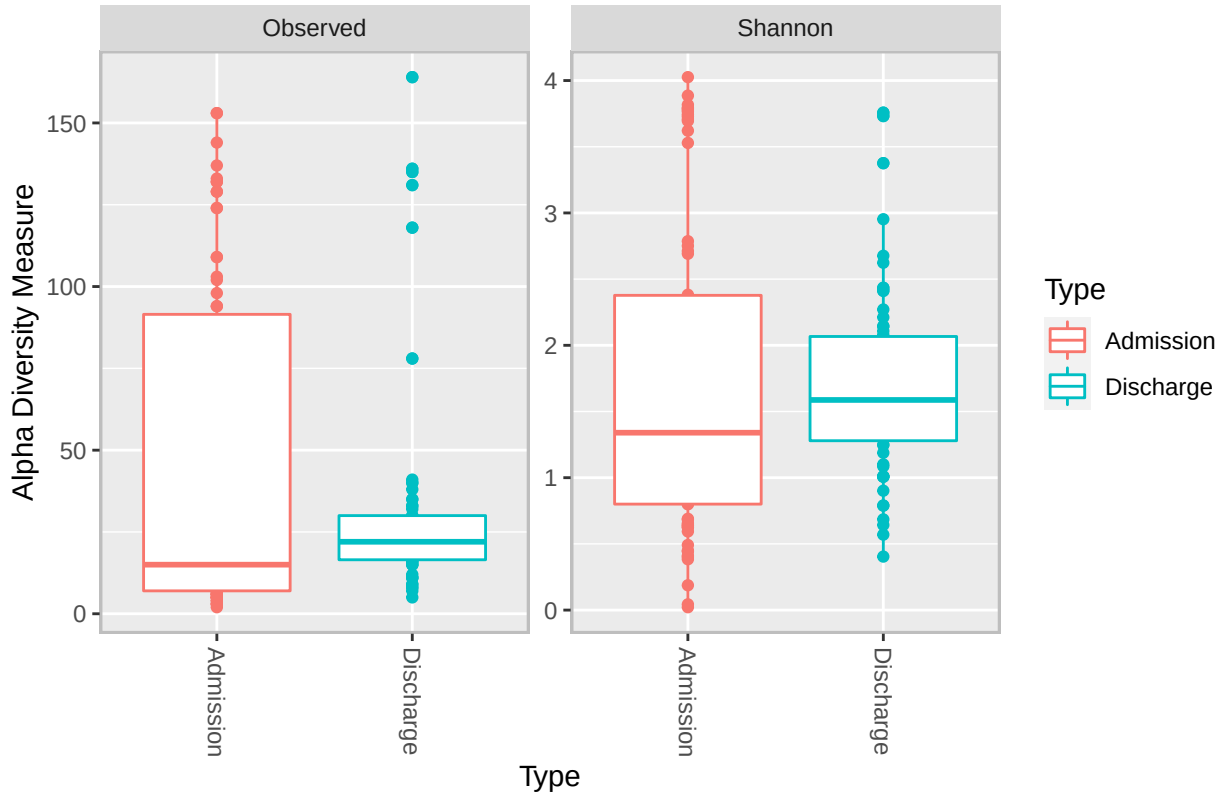


Figure 16: Boxplots of richness and shannon diversity metrics coloured by the type of sample.

- 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.

```
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))
```

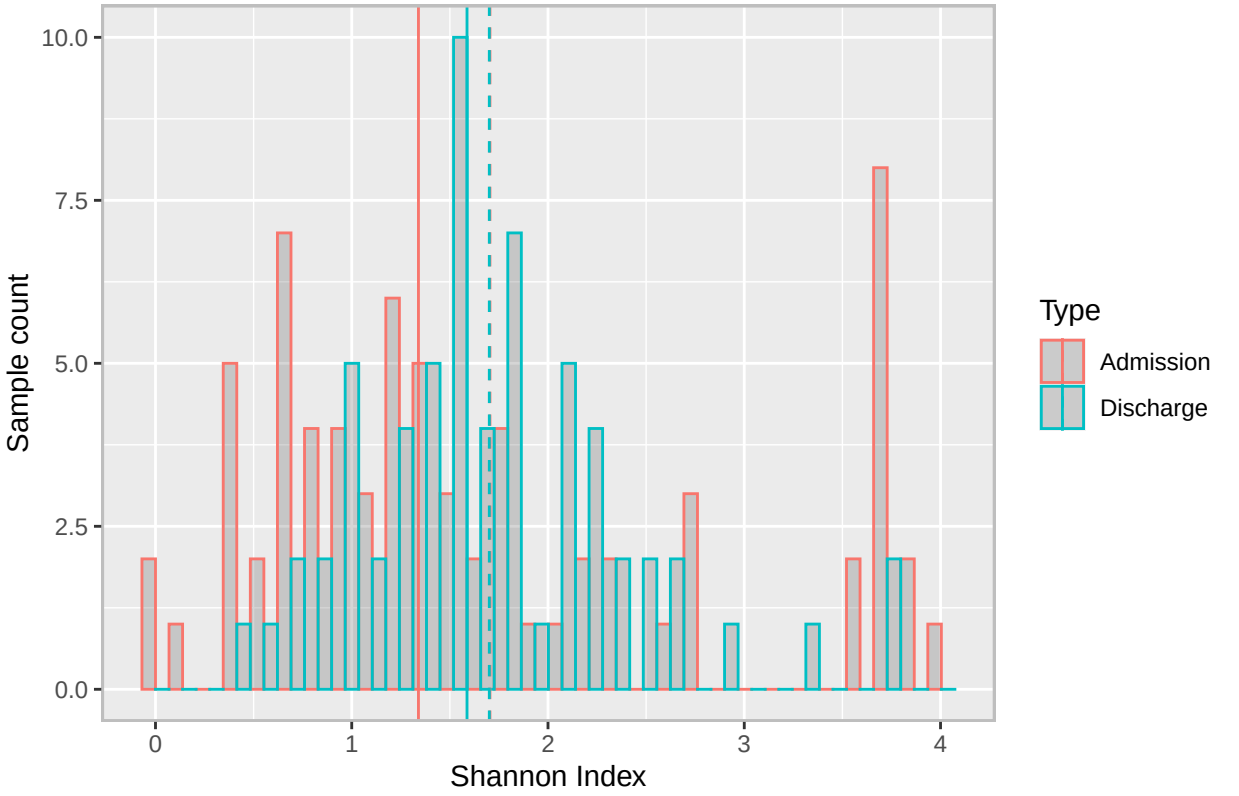


Figure 17: Histogram showing the distribution of the shannon index scores across samples, coloured by sample type and with lines representing the mean (dashed) and median (solid).

- 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)
```

4.6 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)
```

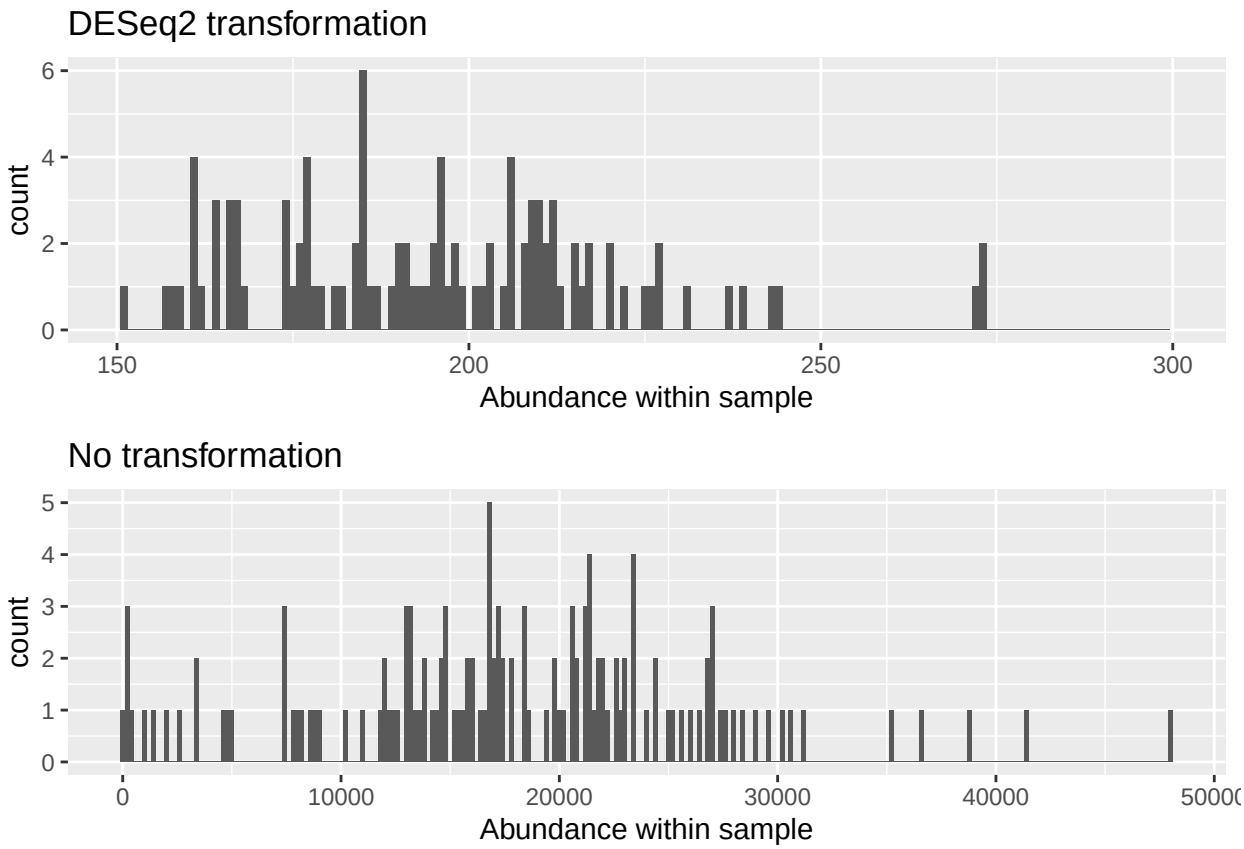


Figure 18: Pre and post transformation of taxonomic counts with DESeq2

4.6.1 Statisical test: calculate differential abundances with *DESeq2*.

- Use `DESeq()` to perform differential expression analysis based on the negative binomial distribution.
- 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$padj, na.last = NA), ]
alpha = 0.05
sigtab = res[(res$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()
```

```
##      baseMean log2FoldChange     lfcSE         padj         Phylum
## 1 22039.55426      -6.346785 0.5932271 1.464963e-24     Firmicutes
## 2     8.76185       6.188136 1.3287488 2.276764e-04     Firmicutes
## 3   240.21824       4.506096 1.0066658 3.595560e-04     Firmicutes
## 4    23.99158       5.307828 1.2984109 1.545022e-03 Proteobacteria
## 5    74.29650       3.996580 1.0701427 5.339093e-03     Firmicutes
##                 Class             Order             Family
## 1             Bacilli        Bacillales  Staphylococcaceae
## 2          Clostridia     Clostridiales   Clostridiaceae_1
## 3             Bacilli   Lactobacillales   Lactobacillaceae
## 4 Gammaproteobacteria Enterobacteriales Enterobacteriaceae
## 5       Negativicutes   Selenomonadales    Veillonellaceae
##                         Genus
## 1              Staphylococcus
## 2 Clostridium_sensu_stricto_1
## 3               Lactobacillus
## 4                Enterobacter
## 5                  Veillonella
```

- Export results.

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

4.7 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) %>%
            mutate(across(where(is.numeric), round, 2)) %>%
            remove_rownames() %>%
            add_column("p-adj" = "<0.01") %>%
            select("Genus","p-adj", "log2FoldChange","lfcSE") %>%
            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 = 15)

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

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

5 Multivariant Analysis.

5.1 About.

This section is for exploring the impact of several variables on alpha diversity and taxonomic abundances.

5.2 Mixed effects modelling with *DESeq2* for differential abundance testing.

- This analysis is creating two seperate models to explore potential associations between clinical variables and microbiome taxonomy.
- The variables were selected for the model using a combination of exploratory analysis (not detailed here) and the literature.
- The first part of the workflow details the steps for the analysis on the *Admission* model, and there is a seperate chunk with the entire analysis for the *Discharge* model.
- The significant results from the two models are combined at the end.

```
sapply(c("DESeq2", "phyloseq", "dplyr", "ggplot2", "grid",
         "gridExtra", "ggpubr", "sjPlot", "pheatmap", "tidyverse"),
          require, character.only = TRUE)
```

5.2.1 Subset data.

- Subset to filtered/agglomerated *Admission* data and scale continuous variables.

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

sample_data(ps.NICU)$Gestation_Days_scaled <- scale(sample_data(ps.NICU)$Gestational.Age.at.Birth,
                    center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Gestational.Age.at.Birth))

sample_data(ps.NICU)$Birth.Weight_scaled <- scale(sample_data(ps.NICU)$Birth.Weight,
                    center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Birth.Weight))
```

5.2.1.1 Testing for multicollinearity.

- Define the `corvif()`function that takes metadata and creates a linear model to see if any collinearity exists between variables.
- Then use this function on a defined a vector with all the variables to be included in the model.
- If GVIF < 3 = no collinearity.

```
corvif <- function(data) {
  data <- as.data.frame(data)

  form    <- formula(paste("fooy ~ ",paste(strsplit(names(data)," "),collapse = " + ")))
  data  <- data.frame(fooy = 1 + rnorm(nrow(data)) ,data)
  lm_mod  <- lm(form,data) # runs linear model with above formula and metadata

  cat("\n\nVariance inflation factors\n\n")
  print(myvif(lm_mod))
}

model_data <- sample_data(ps.NICU) %>%
  unclass() %>%
  as.data.frame()

model_data <- cbind(model_data$Mode.of.Delivery, model_data$Feeding.Type,
                    model_data$Gestation_Days_scaled, model_data$NEC,
                    model_data$Sepsis, model_data$Chorioamnionitis,
                    model_data$Preeclampsia, model_data$ROP)

corvif(model_data)
```

- Convert from *phyloseq* to *deseq* object.

```
multi.deseq = phyloseq_to_deseq2(ps.NICU, ~Sepsis + Feeding.Type +
                                   Chorioamnionitis + Mode.of.Delivery +
                                   Gestation_Days_scaled  + NEC +
                                   Preeclampsia + ROP)
```

- Define function for calculating geometric means.
- Calculate geometric means, and subsetuently estimate size factors.
- Subset out taxa with small counts and low occurance (at least 10 in 20 or more samples).

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

geoMeans <- apply(counts(multi.deseq), 1, gm_mean)
multi.deseq <- estimateSizeFactors(multi.deseq, geoMeans = geoMeans)

nc <- counts(multi.deseq, normalized = TRUE)
filtered <- rowSums(nc >= 10) >= 20
multi.deseq <- multi.deseq[filtered,]
```

- 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.

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

abund_sums_trans <- data.frame(sum = colSums(getVarianceStabilizedData(multi.deseq) ),
                      sample = colnames(getVarianceStabilizedData(multi.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, 200) +
  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)
```

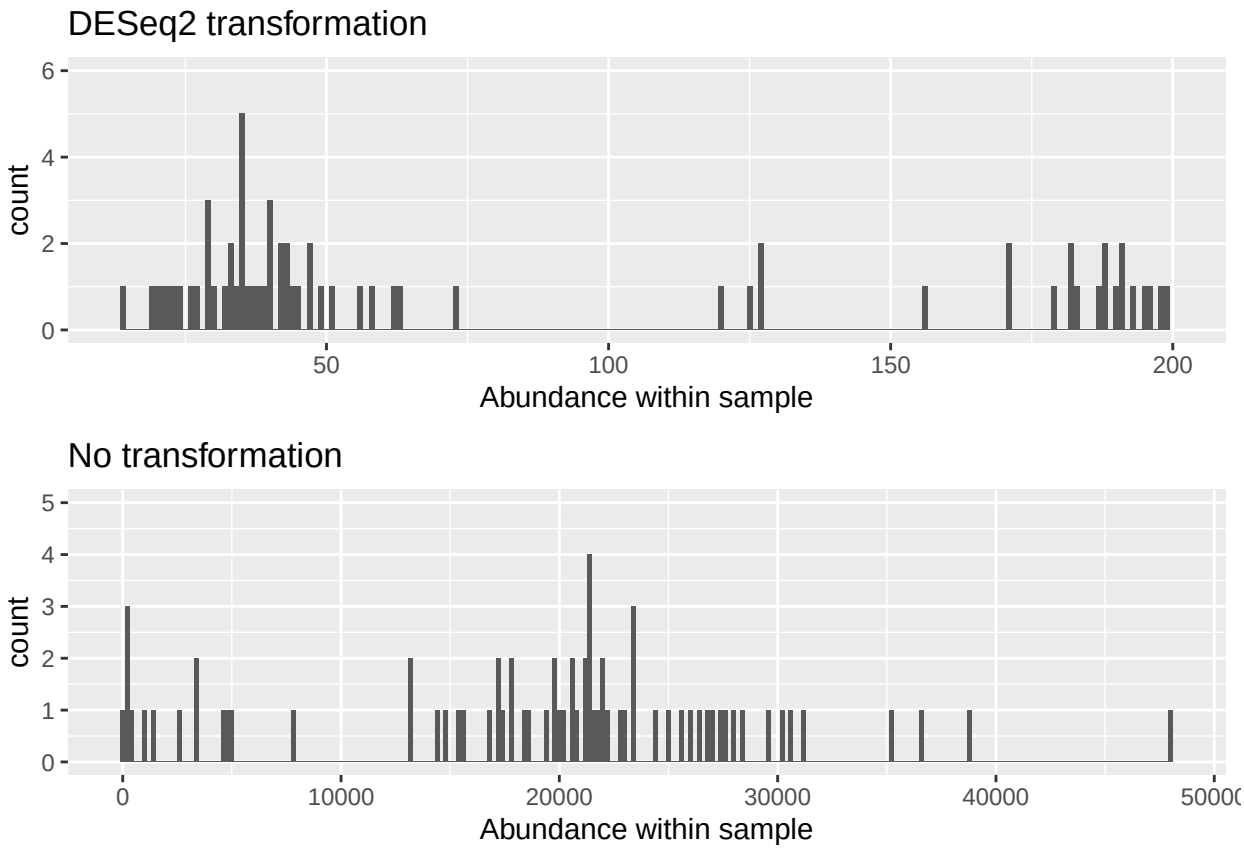


Figure 19: Pre and post transformation of taxonomic counts with DESeq2

5.2.2 Statisical test: calculate differential abundances with *DESeq2*.

- Use `Deseq()` to perform the normalisation and analysis.
- Omit non-converging rows.

```
multi.deseq = DESeq(multi.deseq, fitType = "local", test = "Wald")

multi.deseq.clean <- multi.deseq[which(mcols(multi.deseq)$betaConv),]
```

- Define a function to extract the results.
- Extract the results, order by p value, selects significant (<0.05) results, binds this data to the *tax_table* from the *phyloseq* object to get the taxonomic information, and then select and order the desired columns.

```
get_deseq_res <- function(desq_object, contrast_variable, level1, level2){
  res = results(desq_object, contrast = c(contrast_variable, level1, level2))
  res = res[order(res$padj, na.last = NA), ]
  alpha = 0.05
  sigtab = res[(res$padj < alpha), ]
  sigtab = cbind(as(sigtab, "data.frame"),
          as(tax_table(ps.NICU)[rownames(sigtab), ], "matrix"))
```

```
  sigtab %>%
  arrange(log2FoldChange) %>%
  select("log2FoldChange", "lfcSE", "padj", "Genus") %>%
  add_column(Variable = paste0(contrast_variable, ":Yes"))
}
```

- Use the `get_deseq_res()` to create a table of each of the significant variables.

```
sigtab_admission <- bind_rows(get_deseq_res(multi.deseq.clean,
                              "Chorioamnionitis", "Yes", "No"),
                              get_deseq_res(multi.deseq.clean,
                              "Sepsis", "Yes", "No"),
                              get_deseq_res(multi.deseq.clean,
                              "NEC", "Yes", "No"),
                              get_deseq_res(multi.deseq.clean,
                              "ROP", "Yes", "No")) %>%
                     add_column(Sample = "Admission")
```

- Create a *Discharge* model.

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

sample_data(ps.NICU)$Gestation_Days_scaled <- scale(sample_data(ps.NICU)$Gestational.Age.at.Birth,
                     center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Gestational.Age.at.Birth))

sample_data(ps.NICU)$Birth.Weight_scaled <- scale(sample_data(ps.NICU)$Birth.Weight,
                     center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Birth.Weight))

multi.deseq = phyloseq_to_deseq2(ps.NICU, ~Sepsis + Feeding.Type + Chorioamnionitis +
                                   Mode.of.Delivery + Gestation_Days_scaled  + NEC +
                                   Preeclampsia + ROP)

geoMeans <- apply(counts(multi.deseq), 1, gm_mean)
multi.deseq <- estimateSizeFactors(multi.deseq, geoMeans = geoMeans)

nc <- counts(multi.deseq, normalized = TRUE)
filtered <- rowSums(nc >= 10) >= 20
multi.deseq <- multi.deseq[filtered,]

multi.deseq = DESeq(multi.deseq, fitType = "local", test = "Wald")
multi.deseq.clean <- multi.deseq[which(mcols(multi.deseq)$betaConv),]

sigtab_discharge <- bind_rows(get_deseq_res(multi.deseq.clean,
                              "Chorioamnionitis", "Yes", "No"),
                              get_deseq_res(multi.deseq.clean,
                              "Preeclampsia", "Yes", "No"),
                              get_deseq_res(multi.deseq.clean,
                              "Feeding.Type", "Formula", "Breastmilk")) %>%
                     add_column(Sample = "Discharge")
```

- Merge *Admission* and *Discharge* outputs.

```
DESeq2_Summary_Table <- bind_rows(sigtab_admission, sigtab_discharge)
```

- Export combined table.

```
tab_df(DESeq2_Summary_Table, alternate.rows = TRUE,
       title = "Differentially Abundant Taxa",
       file = "Differentially_Abundant_Taxa_Summary.doc")
```

5.2.3 Visualisations for *deseq* modelling.

- Visualising deseq-transformed abundances with heat maps.

```
multi.deseq.clean %>%
  varianceStabilizingTransformation() %>%
  assay() %>%
  cor() %>%
  pheatmap()
```

- Visualising deseq-transformed abundances with PCA (substitute in any variable).

```
multi.deseq.clean %>%
  varianceStabilizingTransformation() %>%
  plotPCA(intgroup = "Mode.of.Delivery") +
   xlim(-20,20)+
   ylim(-20,20)
```

- Plot counts for comparisons (takes ASV).

```
ggplot((plotCounts(multi.deseq.clean,  "GTGGGGAATATTGCACAATGGGCGCAAGCCTGATGCAGCCATGCCGCGTGTATGAAGAAGGCC
      intgroup = "Preeclampsia", returnData = TRUE)),
      aes(x = Preeclampsia, y = count)) +
  geom_point()+
  scale_y_log10() +
  labs(title = "SEscherichia/Shigella", x = "", y = "")
```

5.3 Mixed effects modelling with lme4 for Shannon Diversity.

- This analysis is exploring the impact of several clinical variables on alpha diversity using a backwards selection method that allows determination of the least complex adequate model.

```
sapply(c("lme4", "nlme", "dplyr", "plyr", "lmerTest", "aods3",
         "tidyr", "ggplot2", "MuMIn", "sjPlot", "gridExtra",
         "grid", "car", "emmeans", "ggpubr"),
         require, character.only = TRUE)
```

5.3.1 Subset data.

- As done previously, subset ps to exclude SCN and NA values.
- Scale continuous variables.
- Estimate richness and save as object.
- 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 <- subset_samples(ps,
                 Primary_Group == "NICU" &
                 (Type == "Admission" | Type == "Discharge"))

sample_data(ps.NICU)$Gestation_Days_scaled <- scale(sample_data(ps.NICU)$Gestational.Age.at.Birth,
            center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Gestational.Age.at.Birth))

sample_data(ps.NICU)$Birth.Weight_scaled <- scale(sample_data(ps.NICU)$Birth.Weight,
            center = TRUE, scale = 2*sd(sample_data(ps.NICU)$Birth.Weight))

ps_alpha_div <- ps.NICU %>%
                estimate_richness(measures = c("Shannon")) %>%
                add_column(Label = row.names(sample_data(ps.NICU)))

ps_samp <- sample_data(ps.NICU) %>%
  unclass() %>%
  data.frame() %>%
  left_join(ps_alpha_div, by = "Label")
```

5.3.2 Testing for multicollinearity.

- Use previously defined `corvif()` function that takes metadata and creates a linear model to see if any collinearity exists between variables.
- Use this function on a defined a vector with all the variables to be included in the model.
- If GVIF < 3 = no collinearity.

```
variables <- cbind(ps_samp$Mode.of.Delivery, ps_samp$Feeding.Type,
                   ps_samp$Gestation_Days_scaled, ps_samp$Birth.Weight_scaled,
                   ps_samp$Antenatal.Antibiotics, ps_samp$Antenatal..Infections,
                   ps_samp$NEC, ps_samp$Sepsis, ps_samp$Chorioamnionitis,
                   ps_samp$Neonatal.Antibiotics, ps_samp$Died,
                   ps_samp$Prolonged..Membrane..Rupture, ps_samp$Preeclampsia,
                   ps_samp$Diabetes, ps_samp$ROP, ps_samp$Type)

corvif(variables)
```

5.3.3 Fit Model.

- Fit the model using *lme4* to obtain AIC values allowing downstream backwards selection (this assumes gaussian distribution).
- *URN* (individual infant) is a random effect that we want to account for but not measure.
- *Type* (of sample) is an interaction variable.

```
global <- lme4::lmer(Shannon ~  (Mode.of.Delivery + Feeding.Type +
                     Gestation_Days_scaled  + Antenatal.Antibiotics +
                     Antenatal..Infections + NEC + Sepsis +
                     Chorioamnionitis + Neonatal.Antibiotics + Died +
                     Prolonged..Membrane..Rupture + Preeclampsia +
                     Diabetes + ROP) * Type + (1|URN), data = ps_samp)
```

- Calculate the goodness of fit (how the sample data fits the distribution) and the Pearsons Chi Square coefficient (how likely observed differences arose by chance).
- Calculate these again post bakwards selection.

```
gof(global)
sum(residuals(global,"pearson")^2)
```

5.3.4 Backwards Selection.

- Define a function that determines what variable is contributing least to the model, as determined by AIC score.
- Then apply that function to the model, and subsequent models, removing variables from the model that are not contributing (first from the interaction and then from the model entirely).

```
dfun <- function(x) {
  x$AIC <- x$AIC-min(x$AIC)
  names(x)[2] <- "dAIC"
  x
}

dfun(drop1(global))
```

```
## Single term deletions
##
## Model:
## Shannon ~ (Mode.of.Delivery + Feeding.Type + Gestation_Days_scaled +
##     Antenatal.Antibiotics + Antenatal..Infections + NEC + Sepsis +
##     Chorioamnionitis + Neonatal.Antibiotics + Died + Prolonged..Membrane..Rupture +
##     Preeclampsia + Diabetes + ROP) * Type + (1 | URN)
##                                    npar   dAIC
## <none>                                  1.9846
## Mode.of.Delivery:Type                 1 3.5147
## Feeding.Type:Type                     2 0.6056
## Gestation_Days_scaled:Type            1 0.7615
## Antenatal.Antibiotics:Type            1 0.3818
## Antenatal..Infections:Type            1 0.0669
## NEC:Type                              1 4.2573
## Sepsis:Type                           1 0.0000
## Chorioamnionitis:Type                 1 0.7315
## Neonatal.Antibiotics:Type             1 0.0020
## Died:Type                             1 0.0804
## Prolonged..Membrane..Rupture:Type     1 0.1535
## Preeclampsia:Type                     1 1.8793
## Diabetes:Type                         1 0.1711
## ROP:Type                              1 7.8182
```

```
global2 <- lme4::lmer(Shannon ~ Neonatal.Antibiotics + (Mode.of.Delivery + Feeding.Type +
                      Gestation_Days_scaled  + Antenatal.Antibiotics +
                      Antenatal..Infections + NEC + Sepsis + Chorioamnionitis + Died +
                      Prolonged..Membrane..Rupture + Preeclampsia + Diabetes + ROP) *
                      Type + (1|URN), data = ps_samp)
dfun(drop1(global2))
```

```
## Single term deletions
##
## Model:
## Shannon ~ Neonatal.Antibiotics + (Mode.of.Delivery + Feeding.Type +
##     Gestation_Days_scaled + Antenatal.Antibiotics + Antenatal..Infections +
##     NEC + Sepsis + Chorioamnionitis + Died + Prolonged..Membrane..Rupture +
##     Preeclampsia + Diabetes + ROP) * Type + (1 | URN)
##                                    npar   dAIC
## <none>                                  1.9824
## Neonatal.Antibiotics                  1 0.9412
## Mode.of.Delivery:Type                 1 3.6701
## Feeding.Type:Type                     2 0.5951
## Gestation_Days_scaled:Type            1 0.7668
## Antenatal.Antibiotics:Type            1 0.4175
## Antenatal..Infections:Type            1 0.0560
## NEC:Type                              1 4.2486
## Sepsis:Type                           1 0.0000
## Chorioamnionitis:Type                 1 0.7939
## Died:Type                             1 0.0829
## Prolonged..Membrane..Rupture:Type     1 0.1991
## Preeclampsia:Type                     1 1.8938
## Diabetes:Type                         1 0.1601
## ROP:Type                              1 7.8566
```

- Continue backwards selection until least complex adequate model is found.

```
global16 <- lme4::lmer(Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis + (Mode.of.Delivery + Gestati
dfun(drop1(global16))
```

```
## Single term deletions
##
## Model:
## Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis + (Mode.of.Delivery +
##     Gestation_Days_scaled + NEC + Preeclampsia + ROP) * Type +
##     (1 | URN)
##                              npar    dAIC
## <none>                             0.0000
## Sepsis                          1  3.3048
## Feeding.Type                    2  4.2259
## Chorioamnionitis                1  0.5295
## Mode.of.Delivery:Type           1  2.6798
## Gestation_Days_scaled:Type      1  0.2766
## NEC:Type                        1  0.5284
## Preeclampsia:Type               1  3.0899
## ROP:Type                        1 10.2922
```

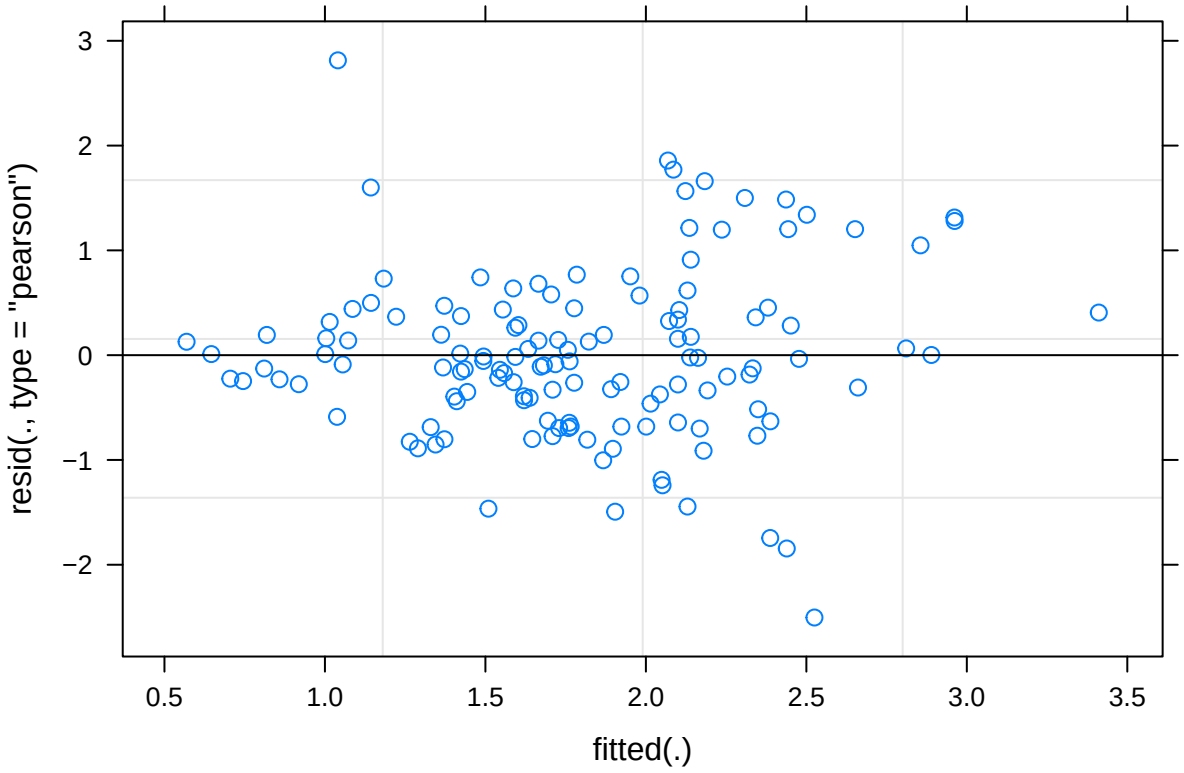
- Get a summary of the final model.

```
summary(global16)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis + (Mode.of.Delivery +
##     Gestation_Days_scaled + NEC + Preeclampsia + ROP) * Type +      (1 | URN)
##    Data: ps_samp
##
## REML criterion at convergence: 350.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8764 -0.5021 -0.0657  0.4575  3.2323
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  URN      (Intercept) 0.04394  0.2096
##  Residual             0.75763  0.8704
## Number of obs: 134, groups:  URN, 85
##
## Fixed effects:
##                                      Estimate Std. Error t value
## (Intercept)                          1.806741   0.211054   8.561
## SepsisYes                           -0.921296   0.424435  -2.171
## Feeding.TypeBreastmilk and Formula   0.216477   0.218806   0.989
## Feeding.TypeFormula                  0.536168   0.198288   2.704
## ChorioamnionitisYes                 -0.285618   0.192975  -1.480
## Mode.of.DeliveryVaginal              0.379515   0.238634   1.590
## Gestation_Days_scaled                0.352624   0.268963   1.311
## NECYes                               0.531529   0.421797   1.260
## PreeclampsiaYes                      0.596584   0.368453   1.619
## ROPYes                              -0.900978   0.252821  -3.564
## TypeDischarge                       -0.003014   0.242147  -0.012
## Mode.of.DeliveryVaginal:TypeDischarge -0.721701   0.352981  -2.045
## Gestation_Days_scaled:TypeDischarge   -0.526170   0.367406  -1.432
## NECYes:TypeDischarge                 -0.995615   0.657695  -1.514
## PreeclampsiaYes:TypeDischarge        -1.063910   0.478268  -2.225
## ROPYes:TypeDischarge                  1.233819   0.362389   3.405

##
## Correlation matrix not shown by default, as p = 16 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

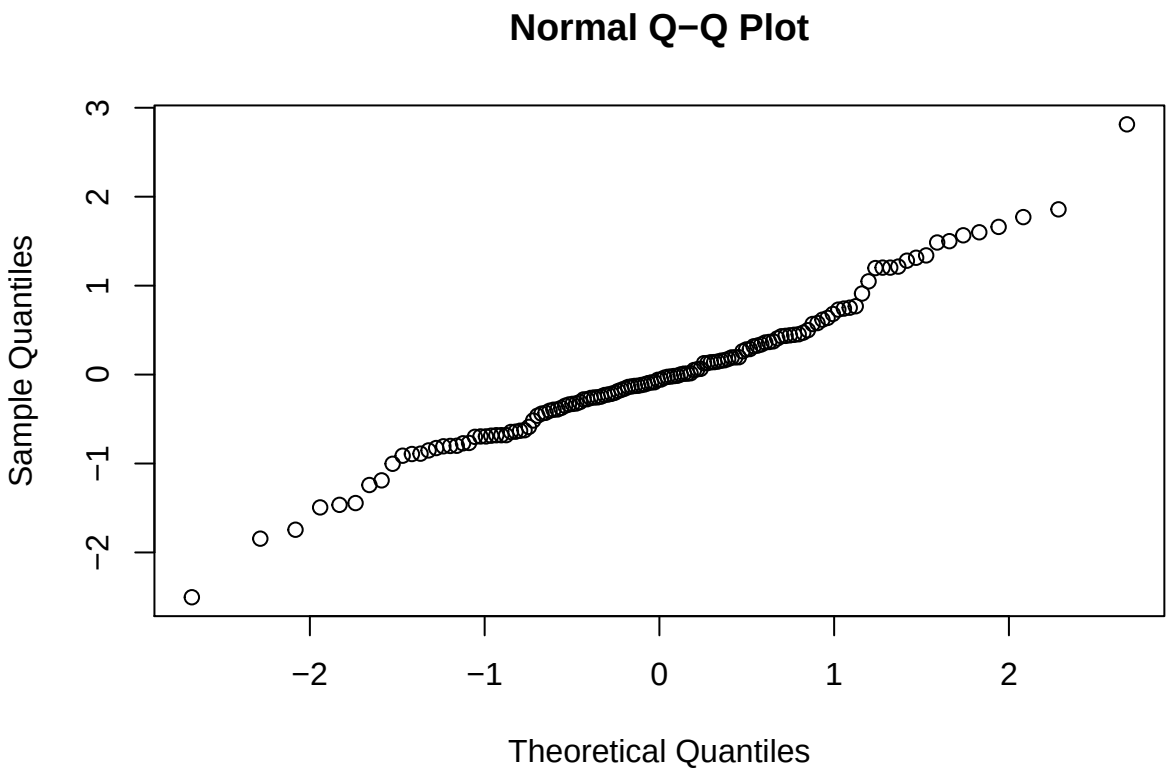
- Explore model distribution (linearity).

```
plot(global16)
```



- Explore model distribution (normality).

```
qqnorm(resid(global16))
```



- Calculate the goodness of fit (how the sample data fits the distribution).

```
gof(global16)
```

```
## D  = 350.4732, df = 116, P(>D) = 2.206633e-25
## X2 = 84.8627, df = 116, P(>X2) = 0.9867121
```

- Calculate the Pearsons Chi Square coefficient (how likely observed differences arose by chance).

```
sum(residuals(global16,"pearson")^2)
```

```
## [1] 84.86274
```

- Check R2 (good fit is between 0.2 - 0.4).

```
MuMIn::r.squaredGLMM(global16)
```

```
##            R2m       R2c
## [1,] 0.2586314 0.2992686
```

5.3.5 Statistical tests for generalised linear model.

- Have the option to run `lmer()` from the *lmertest* package or use *car* to compute an analysis of variance.
- With *lmerTest*.

```
summary(lmer(Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis + (Mode.of.Delivery +
             Gestation_Days_scaled  + NEC + Preeclampsia + ROP) * Type + (1|URN),
             data = ps_samp))
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis + (Mode.of.Delivery +
##     Gestation_Days_scaled + NEC + Preeclampsia + ROP) * Type +      (1 | URN)
##    Data: ps_samp
##
## REML criterion at convergence: 350.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8764 -0.5021 -0.0657  0.4575  3.2323
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  URN      (Intercept) 0.04394  0.2096
##  Residual             0.75763  0.8704
## Number of obs: 134, groups:  URN, 85
##
## Fixed effects:
##                                      Estimate Std. Error         df t value
## (Intercept)                          1.806741   0.211054  96.769296   8.561
## SepsisYes                           -0.921296   0.424435  96.742775  -2.171
## Feeding.TypeBreastmilk and Formula   0.216477   0.218806  68.231583   0.989
## Feeding.TypeFormula                  0.536168   0.198288  60.476101   2.704
## ChorioamnionitisYes                 -0.285618   0.192975  66.995552  -1.480
## Mode.of.DeliveryVaginal              0.379515   0.238634 116.733073   1.590
## Gestation_Days_scaled                0.352624   0.268963 115.447932   1.311
## NECYes                               0.531529   0.421797 116.762666   1.260
## PreeclampsiaYes                      0.596584   0.368453 117.431766   1.619
## ROPYes                              -0.900978   0.252821 117.395909  -3.564
## TypeDischarge                       -0.003014   0.242147  85.441589  -0.012
## Mode.of.DeliveryVaginal:TypeDischarge -0.721701   0.352981  86.518967  -2.045
## Gestation_Days_scaled:TypeDischarge  -0.526170   0.367406  91.911464  -1.432
## NECYes:TypeDischarge                 -0.995615   0.657695 117.953773  -1.514
## PreeclampsiaYes:TypeDischarge        -1.063910   0.478268  81.873985  -2.225
## ROPYes:TypeDischarge                  1.233819   0.362389  73.429895   3.405
##                                     Pr(>|t|)
## (Intercept)                         1.73e-13 ***
## SepsisYes                            0.03241 *
## Feeding.TypeBreastmilk and Formula   0.32599
## Feeding.TypeFormula                  0.00888 **
## ChorioamnionitisYes                  0.14354
## Mode.of.DeliveryVaginal              0.11446
```

```
## Gestation_Days_scaled                   0.19244
## NECYes                                  0.21013
## PreeclampsiaYes                         0.10810
## ROPYes                                  0.00053 ***
## TypeDischarge                           0.99010
## Mode.of.DeliveryVaginal:TypeDischarge   0.04393 *
## Gestation_Days_scaled:TypeDischarge     0.15550
## NECYes:TypeDischarge                    0.13275
## PreeclampsiaYes:TypeDischarge           0.02887 *
## ROPYes:TypeDischarge                    0.00108 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Export *lmerTest* results (either save previous command as an object or drop code into the function).

```
tab_model(global17, show.se = TRUE, string.se = "Standard Error", show.ci = FALSE,
          show.re.var = FALSE, show.ngroups = FALSE, show.icc = FALSE,
          title = "Linear Mixed Effects Model: Shannon Diversity Index",
          file = "lme4_global17.doc")
```

- With *car*.

```
car::Anova(global16) %>%
  as.data.frame(row.names = NULL)
```

```
##                                 Chisq Df   Pr(>Chisq)
## Sepsis                     4.71168402  1 0.0299582912
## Feeding.Type               7.34363033  2 0.0254302679
## Chorioamnionitis           2.19064366  1 0.1388512806
## Mode.of.Delivery           0.16073680  1 0.6884790628
## Gestation_Days_scaled      0.23791770  1 0.6257141229
## NEC                        0.13118995  1 0.7172014327
## Preeclampsia               0.01003993  1 0.9201859715
## ROP                        2.92511428  1 0.0872110500
## Type                       0.13685141  1 0.7114314187
## Mode.of.Delivery:Type      4.18034784  1 0.0408953065
## Gestation_Days_scaled:Type 2.05097148  1 0.1521091035
## NEC:Type                   2.29157755  1 0.1300776528
## Preeclampsia:Type          4.94843148  1 0.0261143507
## ROP:Type                  11.59185811  1 0.0006624119
```

- Export *car* results (either save previous command as an object or drop code into the function).

```
tab_df(global_anova, alternate.rows = TRUE, show.rownames = TRUE,
       title = "ANOVA: Shannon Diversity lmer Model", file = "Shannon_Anova.doc")
```

- Perform post-hoc Tukeys analysis to find pairwise differences.

```
as.data.frame(emmeans(global16, list(pairwise ~ Sepsis), adjust = "tukey"))
```

```
##   Sepsis contrast    emmean        SE        df   lower.CL upper.CL
## 1     No        . 1.8297804 0.1880356  55.64220  1.3656120 2.293949
## 2    Yes        . 0.9084842 0.4209482 100.45395 -0.1164116 1.933380
## 3      . No - Yes 0.9212963 0.4266243  97.46146 -0.1179618 1.960554
```

```
as.data.frame(emmeans(global16, list(pairwise ~ Feeding.Type), adjust = "tukey"))
```

```
as.data.frame(emmeans(global16, list(pairwise ~ Mode.of.Delivery:Type), adjust = "tukey"))
```

```
as.data.frame(emmeans(global16, list(pairwise ~ Preeclampsia:Type), adjust = "tukey"))
```

```
as.data.frame(emmeans(global16, list(pairwise ~ ROP:Type), adjust = "tukey"))
```

- Export post-hoc results (either save previous commands as an object or drop code into the function).

```
tab_df(Sepsis_tukeys, alternate.rows = TRUE, show.rownames = TRUE,
       title = "Tukey's: Sepsis and Shannon Diversity",
       file = "Sepsis_tukeys.doc")
```

- To get a summary for significant pairwise comparisons.

```
ddply(subset(ps_samp, Type == "Admission"), "Preeclampsia", summarise,
              N    = length(Shannon),
              mean = mean(Shannon),
              sd   = sd(Shannon),
              se   = sd / sqrt(N))
```

5.3.6 Reintegration.

- Add variables from the first model back into the final model one at a time to calculate estimates, SE and p values for each.

```
summary(lmer(Shannon ~ Sepsis + Feeding.Type + Chorioamnionitis +
            (Antenatal.Antibiotics + Mode.of.Delivery + Gestation_Days_scaled  +
             NEC + Preeclampsia + ROP) * Type + (1|URN), data = ps_samp))
```

- Export reintigration results (either save previous commands as an object or drop code into the function).

```
tab_model(global18, terms = c("Antenatal.AntibioticsNo", "Antenatal.AntibioticsNo:TypeDischarge" ),
          show.se = TRUE, string.se = "Standard Error", show.ci = FALSE, show.re.var = FALSE,
          show.ngroups = FALSE, show.icc = FALSE,
          title = "Linear Mixed Effects Model: Shannon Diversity",
          file = "lme4_antibiotics_reintigration.doc")
```

5.3.7 Visualisation: plot significant variables as box plots.

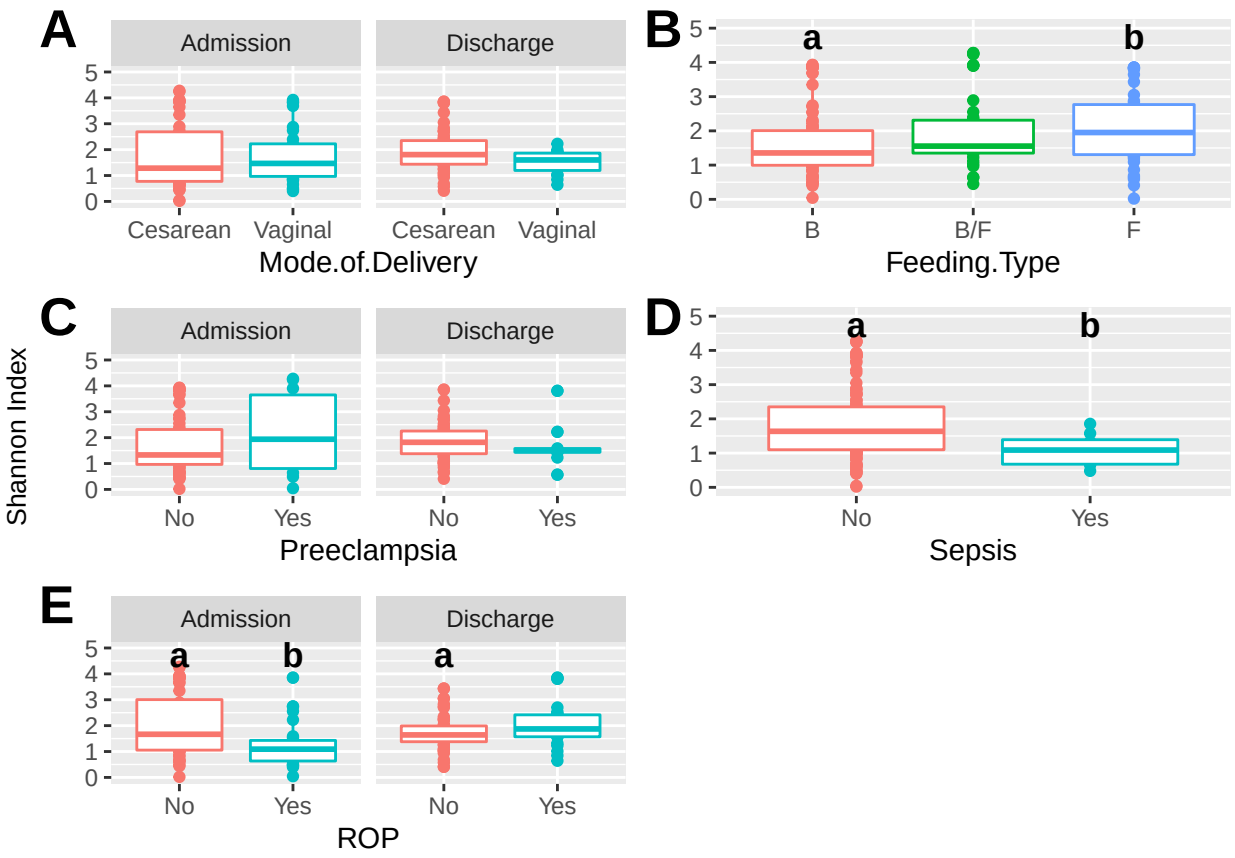
- Create a function for the boxplots (as it gets repetitive) that takes the data, the variable and any added annotation as arguments.

- Any other variable-specifc specifications for the plots can be added after the function. **eg.** Variables that interact with *Type* are faceted by the variable with `+ facet_wrap(~Type)`.

```
shannon_box_plot <- function(data, variable, anno){
  ggplot(data, aes(x = data[[variable]], y = data$Shannon)) +
  geom_point(aes(colour = data[[variable]])) +
  geom_boxplot(aes(colour = data[[variable]])) +
  labs(x = variable, y = "") +
  theme(legend.position = "none") +
  geom_text(data = anno, aes(x = xstar, y = ystar,
  label = lab, size = 12, fontface = "bold")) +
  ylim(0, 5)
}
```

- Wrap all the plots into a function so intermediate objects don't need to be created in environment.

```
box_grid <- function(ps_samp){
# Delivery
anno <- data.frame(xstar = c(1, 2), ystar = c(4.75, 4.75), lab = "")
ggplot_Mode.of.Delivery <- (shannon_box_plot(ps_samp, "Mode.of.Delivery", anno) +
  facet_wrap(~Type))   %>%
  annotate_figure(fig.lab = "A", fig.lab.face = "bold", fig.lab.size = 20)

# Diet
anno <- data.frame(xstar = c(1, 3), ystar = c(4.75, 4.75),lab = c("a", "b"))
ggplot_Feeding.Type <- (shannon_box_plot(ps_samp, "Feeding.Type", anno)  +
  scale_x_discrete(labels = c("B", "B/F", "F"))) %>%
  annotate_figure(fig.lab = "B", fig.lab.face = "bold", fig.lab.size = 20)

# Preeclampsia
anno <- data.frame(xstar = c(1, 2), ystar = c(4.75, 4.75), lab = "")
ggplot_Preeclampsia <- (shannon_box_plot(ps_samp, "Preeclampsia", anno)  +
  facet_wrap(~Type)) %>%
  annotate_figure(fig.lab = "C", fig.lab.face = "bold", fig.lab.size = 20)

# Sepsis
anno <- data.frame(xstar = c(1, 2), ystar = c(4.75, 4.75), lab = c("a", "b"))
ggplot_Sepsis <- shannon_box_plot(ps_samp, "Sepsis", anno) %>%
  annotate_figure(fig.lab = "D", fig.lab.face = "bold", fig.lab.size = 20)

# ROP
anno <- data.frame(xstar = c(1, 2, 1), ystar = c(4.75, 4.75, 4.75),
        lab = c("a", "b", "a"), Type = c("Admission", "Admission", "Discharge"))
ggplot_ROP <- (shannon_box_plot(ps_samp, "ROP", anno)  +
  facet_wrap(~Type)) %>%
  annotate_figure(fig.lab = "E", fig.lab.face = "bold", fig.lab.size = 20)

# Create the grid
grid.arrange(ggplot_Mode.of.Delivery, ggplot_Feeding.Type, ggplot_Preeclampsia, ggplot_Sepsis, ggplot_R
}

box_grid(ps_samp)
```



- Export box plot grid.

```
ggsave("Linear_model_boxplot.png", plot = (box_grid(ps_samp)), dpi = 600, height = 10, width = 10)
```

FINISIHED

Link to github repo.