Pipeline

Jacob Westaway

Last updated on 2021-03-26

About.

Pipeline from raw reads to creation of a phyloseq object and premlinary data manipulation. This pipeline is based on a workflow from the paper Characterising the bacterial gut microbiome of probiotic-supplmented very-preterm infants and the DADA2 workflow developed by *Callahan, et al.*. In addition is removal of contamination with MicroDecon, and creation of a phyloseq object, which is subsequently filtered, agglomerated and normalised in preperation for analysis.

Bioinformatics Pipeline.

About.

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", "tidyverse"),
        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 <-"GKB-HK-200130_Giana/RawData/Data_Combined"

fnFs <- sort(list.files(path, pattern="_1.fq.gz", full.names = TRUE))

fnRs <- sort(list.files(path, pattern="_2.fq.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])
```

```
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` is not needed as primers/barcodes already removed.
- `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.
- **225 produces no reads out... Are the reads only 200bp long? - ask Giana**

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen = c(220, 220),
                     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
```

Sequence Inference.

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

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

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)))
```

Remove chimeras.

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

rm(seqtab)
```

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)
```

Contamination removal with *MicroDecon*.

```
library(microDecon)
```

Reformat data for *MicroDecon*.

- Transpose sequencing table (post chimera removal) and convert to a dataframe.
- Reorder sequencing table by a prior grouping (days). - **might need to put this before the previous chunk** - can combine.
- Move blank sample columns to the start of the sequencing table.
- Turn row names into their own column as *MicroDecon* requires that the OTUs have a unique ID in column 1.

```
wrangle_microdecon <- function(seqtab.nochim){

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

# a prior grouping
```

```
  Metadata_microdecon <- Metadata %>%
  arrange(Date) %>%
  select(Sample, Date) %>% # select key columns
  mutate(Sample = paste0("AM", Sample)) # add AM into cell values to be the same as the count table

microdecon.df <- microdecon.df %>%
  relocate(any_of(Metadata_microdecon$Sample)) # rearrange the columns by the ordered metadata

# blanks to first columns
Metadata_microdecon <- Metadata %>%
  filter(Type == "negative control") %>%  # filter for negative controls
  select(Sample, Type, Date) %>%
  mutate(Sample = paste0("AM", Sample))

microdecon.df <- microdecon.df %>%
  relocate(any_of(Metadata_microdecon$Sample)) %>%
  tibble::rownames_to_column(var = "ID") # turn the rownames into the first column
}

microdecon.df <- wrangle_microdecon(seqtab.nochim)

rm(seqtab.nochim)
```

Decontaminate data using `decon()`.

- get the counts for each of the a priori groups (date) for `numd.ind`, not including the blanks.
- **Do we need to make sure the metadata and microdecon align as well?** Looks good.

```
numb_ind_vector <- Metadata %>%
  filter(Type == "Microbiome") %>% # remove blanks
  group_by(Date) %>%
  summarise(n()) %>% # get the counts for each date.
  select("n()") %>%
  add_row("n()" = 35) %>% # 26 is the number of samples there are no metadata for.
  rename("n" = "n()")
```

- `(ncol(microdecon.df) - 26) - (nrow(Metadata))` should be equal to 1 (ID column).
- `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, numb.blanks = 7,
                  numb.ind = numb_ind_vector$n, taxa = F)

rm(microdecon.df)
rm(numb_ind_vector)
```

Check *MicroDecon* Outputs.

```
decontaminated$decon.table
decontaminated$reads.removed
decontaminated$OTUs.removed
```

```
decontaminated$mean.per.group
decontaminated$sum.per.group
```

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 = "ID") %>%
  select(-1) %>% # remove mean blank
  as.matrix() %>%
  t()

rm(decontaminated)
```

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/silva_nr_v132_train_set.fa.gz")

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

Need to run species assignment on hpc

```
write.csv(taxa, "taxa.csv")

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

Calculate percentage of NA taxa

```
sum(is.na(taxa))/prod(dim(taxa)) * 100
```

```
## [1] 24.78462
```

```
apply(taxa, 2, function(col)sum(is.na(col))/length(col)) * 100
```

```
##   Kingdom    Phylum     Class     Order    Family     Genus
##  3.616763  8.971837 10.518798 19.558011 37.905943 68.136370
```

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.

Packages for reading in and wrangling metadata

```
sapply(c( "tidyverse","plyr", "dplyr", "janitor", "stringi"),
    require, character.only = TRUE)
```

Import metadata and construct dataframe.

Read in 16S Data (ID and dates)

```
ID_Dates <- read.csv("Metadata/16S_Data.csv") %>%
  filter(grepl("AM", Sample)) %>%
  select(1:9, 11:12) %>%
  mutate(New_Date = format(as.Date(Collection.Date),"%m/%y/%d")) %>%
  mutate(Date = str_remove(New_Date, "0")) %>%
  select(2,7,8,11,13) %>%
  mutate_if(is.character, as.factor) %>%
  arrange(Date)

ID_Dates$Date <- revalue(ID_Dates$Date, c("3/03/20" = "3/3/20",
                                          "4/07/20" = "4/7/20",
                                          "5/05/20" = "5/5/20",
                                          "6/02/20" = "6/2/20",
                                          "6/09/20" = "6/9/20",
                                          "7/07/20" = "7/7/20"))
```

Read in Metadata, filter for corresponding dates (with *ID-Dates*), and get the averages for each day. - *NB.* The original metadata file was in html format, and the conversion to excel/csv reformats the dates, and not all in the same way. Thus the dates below have to be revalued to the same format as the ID. This doesn't pose any issues because the sampling finished in september so there is no chance of using the wrong dates.

```
Env_Metadata <- read.csv("Metadata/Metadata_updated.csv" ) %>% # Read in data
  slice(8:n()) %>% # Delete empty rows
  row_to_names(1) %>% # Convert first row to column names
  separate(1, into = c("Date", "Time"), sep = "\\s") # Seperate the date and time into seperatre columns

Env_Metadata$Date <- revalue(Env_Metadata$Date, c("11/2/20" = "2/11/20", # Revalue these levels to the
                                          "10/3/20" = "3/10/20",
                                          "12/5/20" = "5/12/20",
                                          "9/6/20" = "6/9/20"))

Env_Metadata <- Env_Metadata %>% # filter for only the dates that align the 16S collection dates
            filter(Date == "2/11/20"| Date == "2/18/20" | Date == "2/25/20" |
                   Date == "3/3/20" | Date == "3/10/20" | Date == "3/17/20" |
```

```
                  Date == "3/24/20" | Date == "3/31/20" | Date == "4/7/20" |
                  Date == "4/14/20"| Date == "4/21/20" | Date == "4/28/20" |
                  Date == "5/5/20" | Date == "5/12/20" | Date == "5/19/20" |
                  Date == "5/26/20" | Date == "6/2/20" | Date == "6/9/20" |
                  Date == "6/16/20" | Date == "6/23/20" | Date == "6/30/20" |
                  Date == "7/7/20" | Date == "7/14/20"|Date == "7/21/20" | Date == "7/28/20") %>%
          mutate(Time = as.numeric(gsub(":", "", Time))) %>% # remove colon and convert time to numer
          filter(Time %in% (40000:100000)) %>% # filter to only include times between 4am and 10am -
          select(-3) %>%
          mutate_if(is.factor, as.character) %>% # convert to characters so that numeric conversion r
          mutate_at(c("Temperature_C", "pH", "Turbidity_NTU",
                      "RDO_Conc_mgL", "ORP_mV", "RainGauge_mm",
                      "Salinity_PSU"), as.numeric) %>%
          mutate_if(is.character, as.factor) %>%
          group_by(Date) %>% # get the averages for the variables on the dates of 16S sample collecti
          dplyr::summarise(Temperature_C = mean(Temperature_C),
                           pH = mean(pH, na.rm=T),
                           Turbidity_NTU = mean(Turbidity_NTU),
                           RDO_Conc_mgL = mean(RDO_Conc_mgL),
                           ORP_mV = mean(ORP_mV),
                           RainGauge_mm = mean(RainGauge_mm),
                           Salinity_PSU = mean(Salinity_PSU, na.rm=T))
```

Read in ddPCR data

```
dd_PCR <- read_csv("Metadata/ddPCR_data.csv") %>%
          select(-c(1,4:6)) %>%
          dplyr::rename(Sample = "Sample No", "ddPCR" = Conc)
```

Merge 16S, environmental and ddPCR metadata

```
Metadata <- left_join(ID_Dates, Env_Metadata, by = "Date") %>%
            mutate(Sample = as.numeric(gsub("AM", "", Sample))) %>% # to make compatible with dd_PCR
            left_join(dd_PCR, by = "Sample") %>%
            mutate("ID" = Sample) %>% # create a second ID column so we keep one as a column
            mutate(ID = as.factor(ID)) %>%
            relocate(ID) %>%
            mutate(Sample = as.factor(paste0("AM", Sample))) %>% # to make comparible with ps obeject
            column_to_rownames("Sample") %>%
               mutate("Week" = as.numeric(dplyr::recode(Date, "2/11/20" = "1", "2/18/20" = "2", "2/25/2
                      "3/3/20" = "4", "3/10/20" = "5", "3/17/20" = "6", "3/24/20" = "7", "3/31/20" = "8
                      "4/7/20" = "9", "4/14/20" = "10", "4/21/20" = "11", "4/28/20" = "12", "5/5/20" =
                      "5/12/20" = "14", "5/19/20" = "15", "5/26/20" = "16", "6/2/20" = "17", "6/9/20" =
                      "6/16/20" = "19", "6/23/20" = "20", "6/30/20" = "21", "7/7/20" = "22", "7/14/20" =
                      "7/21/20" = "24", "7/28/20" = "25"))) %>%
            filter(Type == "Microbiome") %>%
            mutate(parasite_burden = ifelse( ddPCR > mean(ddPCR), "High", "Low"))
```

Load required packages.

```
sapply(c("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)
```

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(Metadata),
               tax_table(taxa))
```

Wrangling the metadata.

- And do some additional wrangling.
- Convert chraracters to factors.

```
sample_data(ps) <- sample_data(ps) %>%
  unclass() %>%
  as.data.frame() %>%
  mutate_if(is.character, as.factor) %>%
  mutate("Sample" = ID) %>% # need to redo the rownames to save it back into the original ps object
  mutate(Sample = paste0("AM", Sample)) %>%
  column_to_rownames("Sample")
```

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.

```
ps1 <- subset_taxa(ps, !is.na(Phylum) & !Phylum %in% c("", "uncharacterized"))
```

Check percentages of NA values left

```
sum(is.na(tax_table(ps1)))/prod(dim(tax_table(ps1))) * 100

apply(tax_table(ps1), 2, function(col)sum(is.na(col))/length(col)) * 100
```

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(ps1),
               MARGIN = ifelse(taxa_are_rows(ps1), yes = 1, no = 2),
               FUN = function(x){sum(x > 0)})

prevdf = data.frame(Prevalence = prevdf,
                    TotalAbundance = taxa_sums(ps1),
                    tax_table(ps1))
```

- 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(ps1, "Phylum")) %>%
  ggplot(aes(TotalAbundance, Prevalence / nsamples(ps1),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")
```

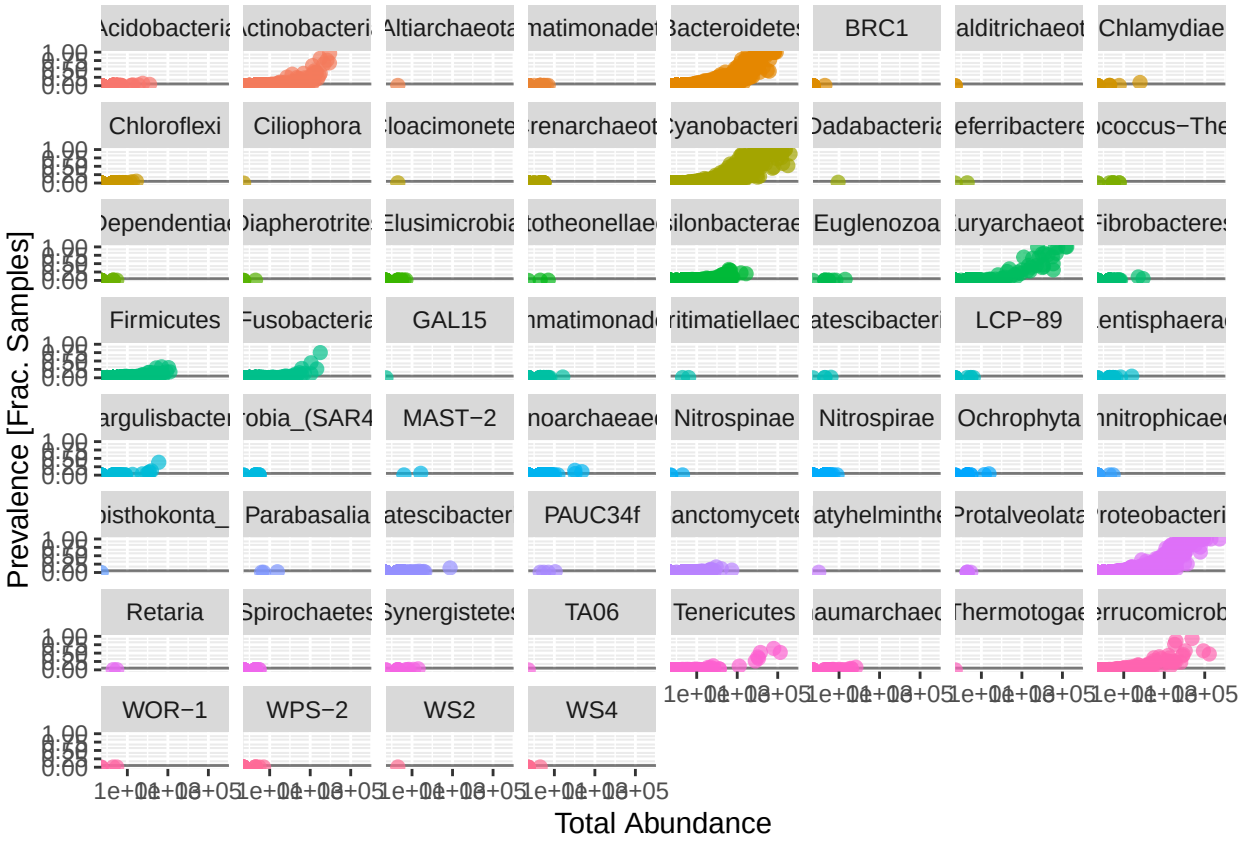


Figure 1: Scatterplot exploring the relationship between prevelance and abundance of phyla.

- 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(ps1)

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

ps2 = prune_taxa(keepTaxa, ps1)
```

- 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")
```

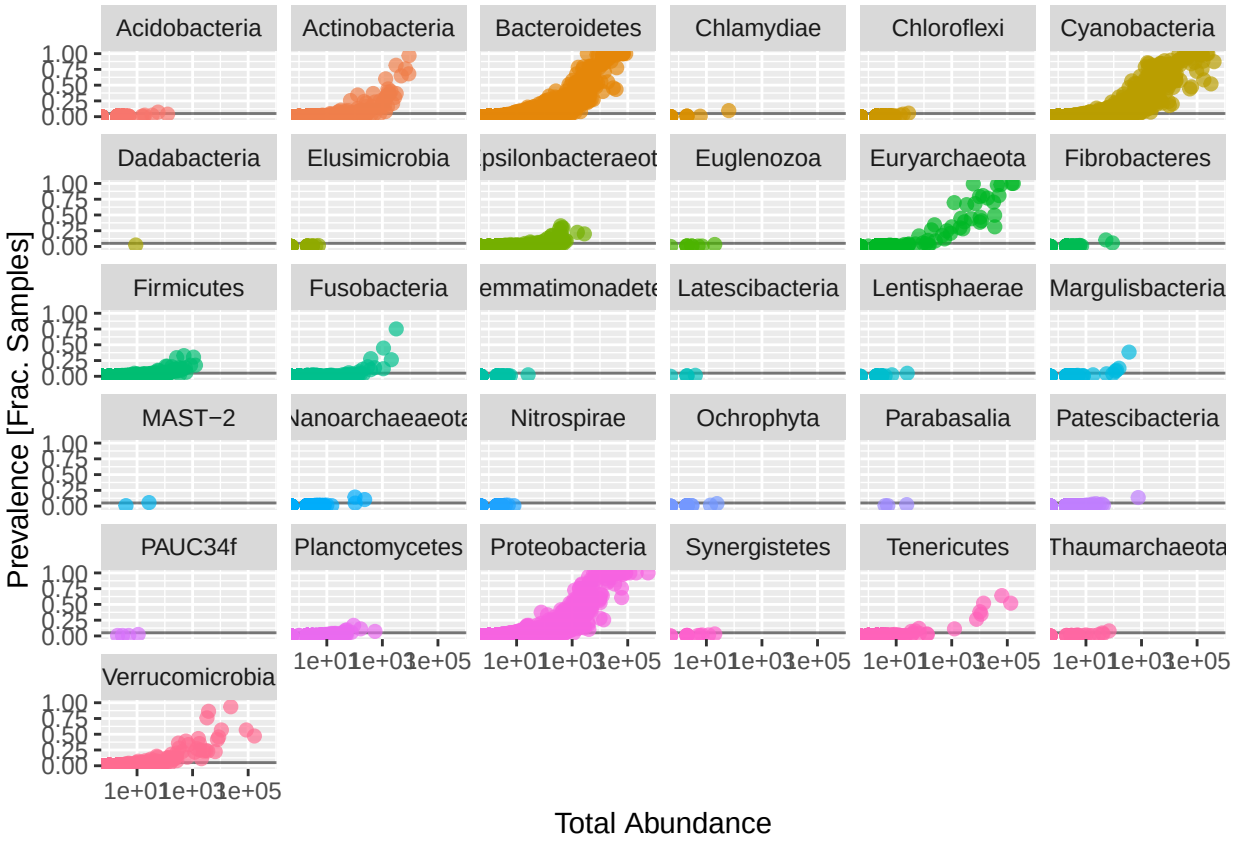


Figure 2: Scatterplot exploring the relationship between prevelance and abundance of phyla on data passed through a prevalence threshold.

Aggolmerate taxa.

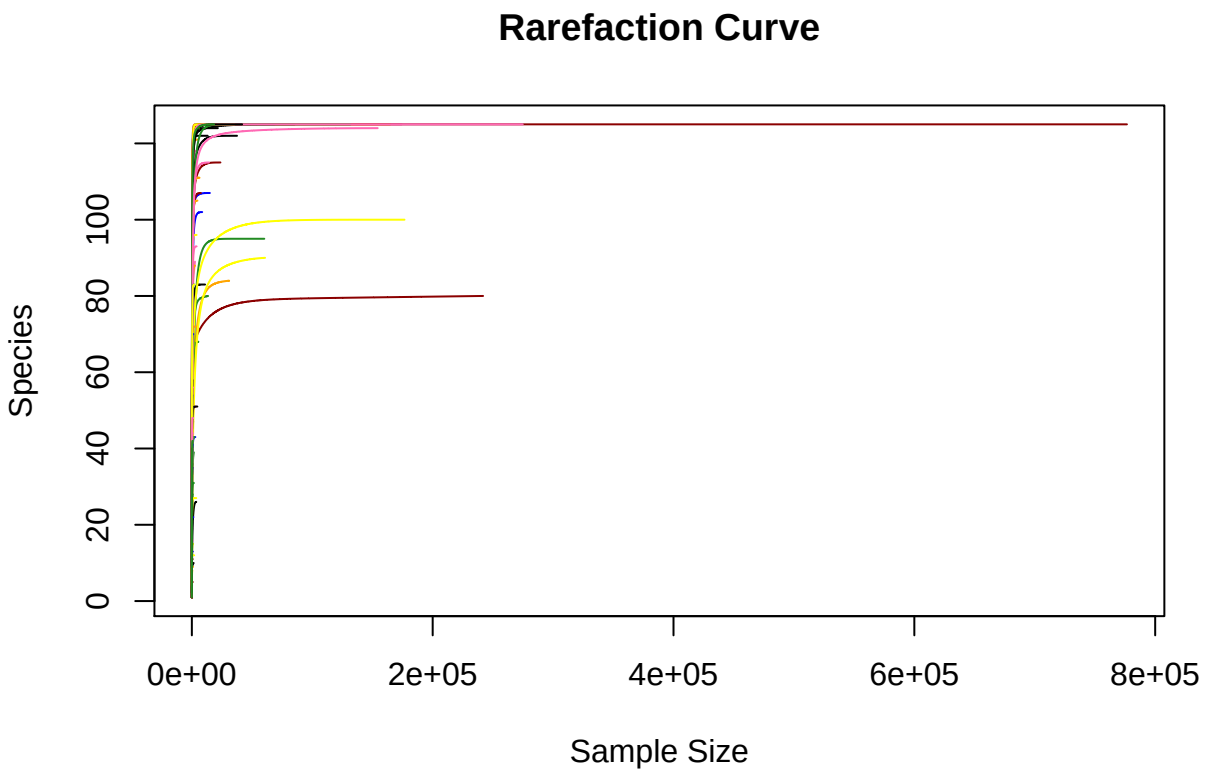
- Combine features that descend from the same genus as most species have not been identified due to the poor sequencing depth in 16S.
- Can check how many genera would be present after filtering by running `length(get_taxa_unique(ps2, taxonomic.rank = "Genus"))`, and `ntaxa(ps3)` will give the number of post agglomeration taxa.

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

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.microbiome <- subset_samples(ps4, Type == "Microbiome")
```

- 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"))
```

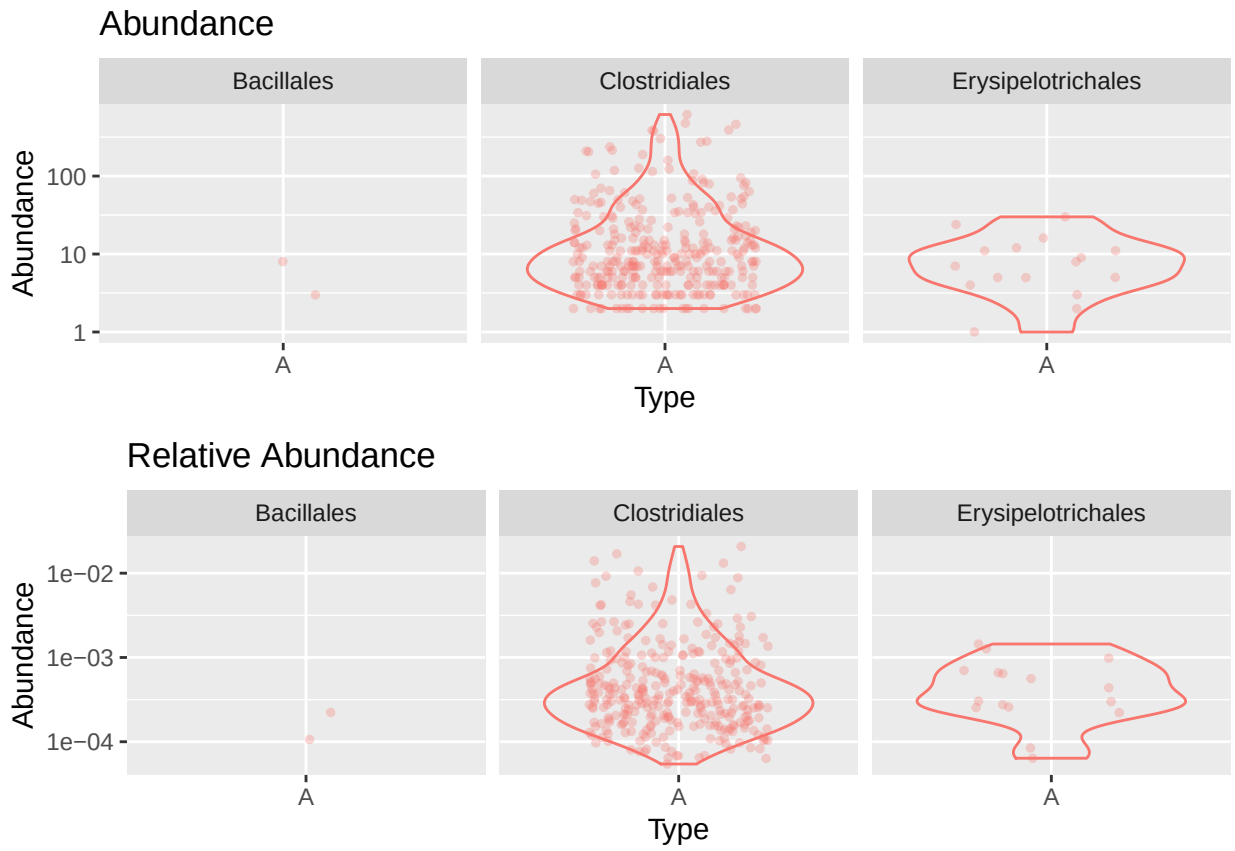


Figure 3: Violin plots exploring of distribution of abundance in Firmicutes before and after normalisation of data. Annotation for x axis; A: Admission, D: Discharge & I: Intermediate.