zymo3

Library

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

```
## [1] '1.18.0'
```

File path

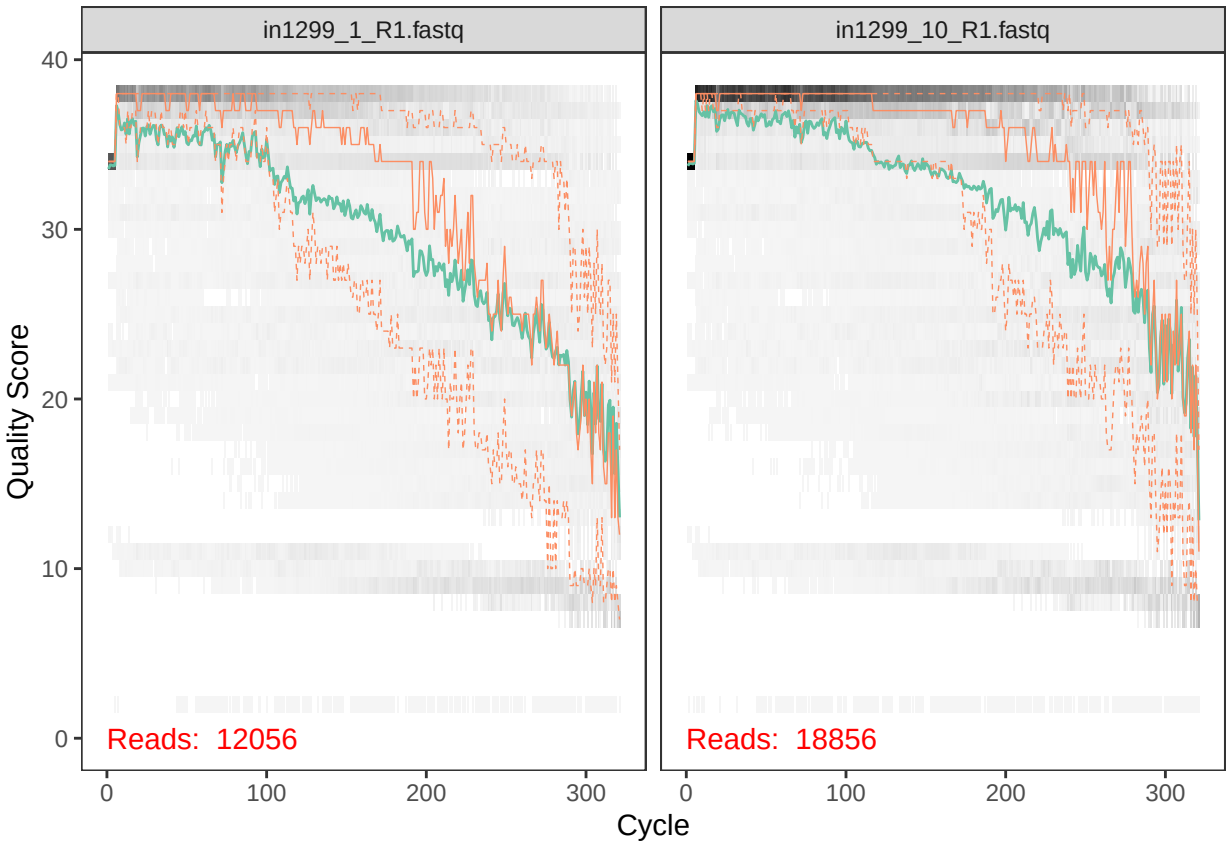
```
path <- file.path("RawSeq")
list.files(path)
```

```
##  [1] "filtered"           "in1299_1_R1.fastq"  "in1299_1_R2.fastq"
##  [4] "in1299_10_R1.fastq" "in1299_10_R2.fastq" "in1299_11_R1.fastq"
##  [7] "in1299_11_R2.fastq" "in1299_12_R1.fastq" "in1299_12_R2.fastq"
## [10] "in1299_13_R1.fastq" "in1299_13_R2.fastq" "in1299_14_R1.fastq"
## [13] "in1299_14_R2.fastq" "in1299_15_R1.fastq" "in1299_15_R2.fastq"
## [16] "in1299_16_R1.fastq" "in1299_16_R2.fastq" "in1299_17_R1.fastq"
## [19] "in1299_17_R2.fastq" "in1299_18_R1.fastq" "in1299_18_R2.fastq"
## [22] "in1299_2_R1.fastq"  "in1299_2_R2.fastq"  "in1299_3_R1.fastq"
## [25] "in1299_3_R2.fastq"  "in1299_4_R1.fastq"  "in1299_4_R2.fastq"
## [28] "in1299_5_R1.fastq"  "in1299_5_R2.fastq"  "in1299_6_R1.fastq"
## [31] "in1299_6_R2.fastq"  "in1299_7_R1.fastq"  "in1299_7_R2.fastq"
## [34] "in1299_8_R1.fastq"  "in1299_8_R2.fastq"  "in1299_9_R1.fastq"
## [37] "in1299_9_R2.fastq"
```

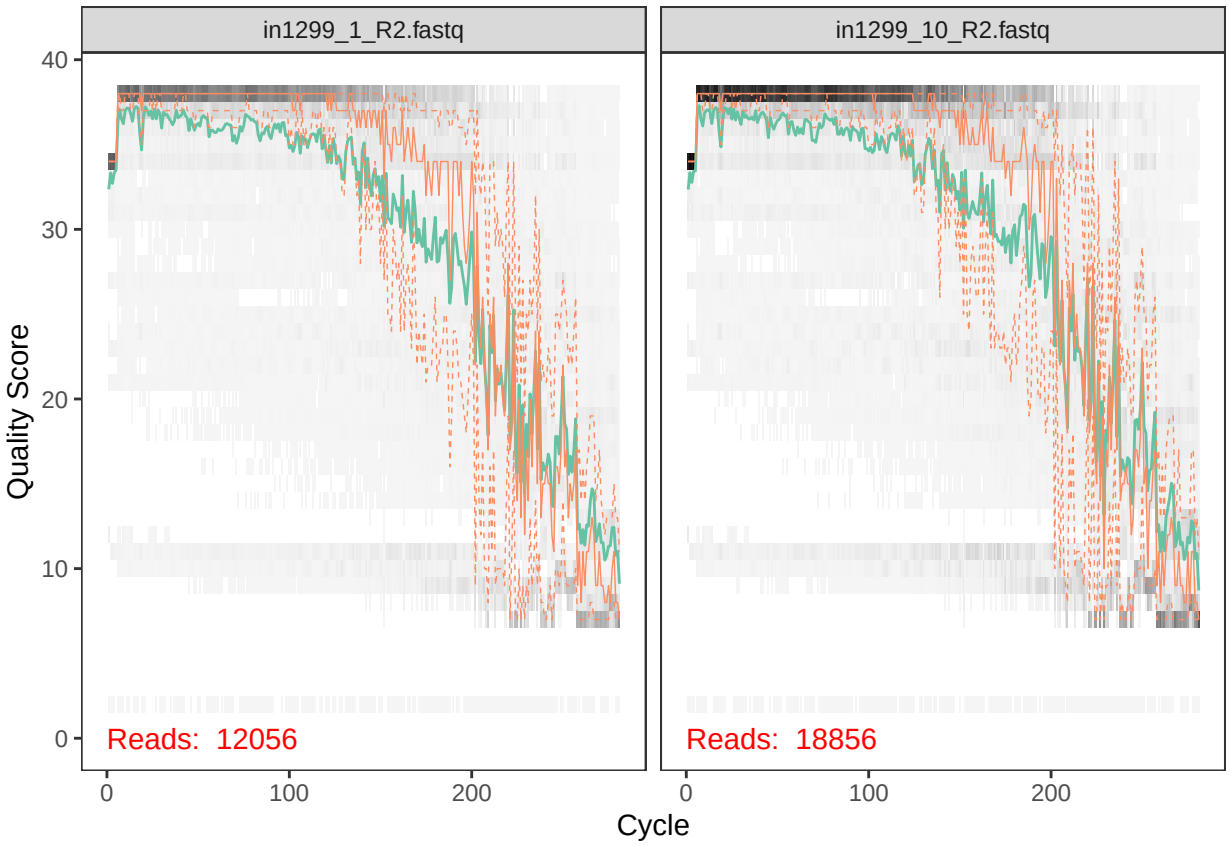
Create object for forward fastq and reverse fastq

```
fns <- sort(list.files(path, full.names = TRUE))
fnFs <- fns[grepl("R1", fns)]
fnRs <- fns[grepl("R2", fns)]

#proper strisplit to get the full name of the files
sample.names <- sapply(strsplit(basename(fnFs), "_R"), `[`, 1)
```

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




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



Assign filenames for filtered fastq.gz files

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

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen=c(320,180),
              maxN=0, maxEE=c(10,10), truncQ=6, trimLeft = c(10,15), rm.phix=TRUE,
              compress=TRUE, multithread=FALSE)

save(out, file="out.RData")
```

saving the output as file so I don't need to run this code chunk everytime ideally to remove the first little part with trimLeft

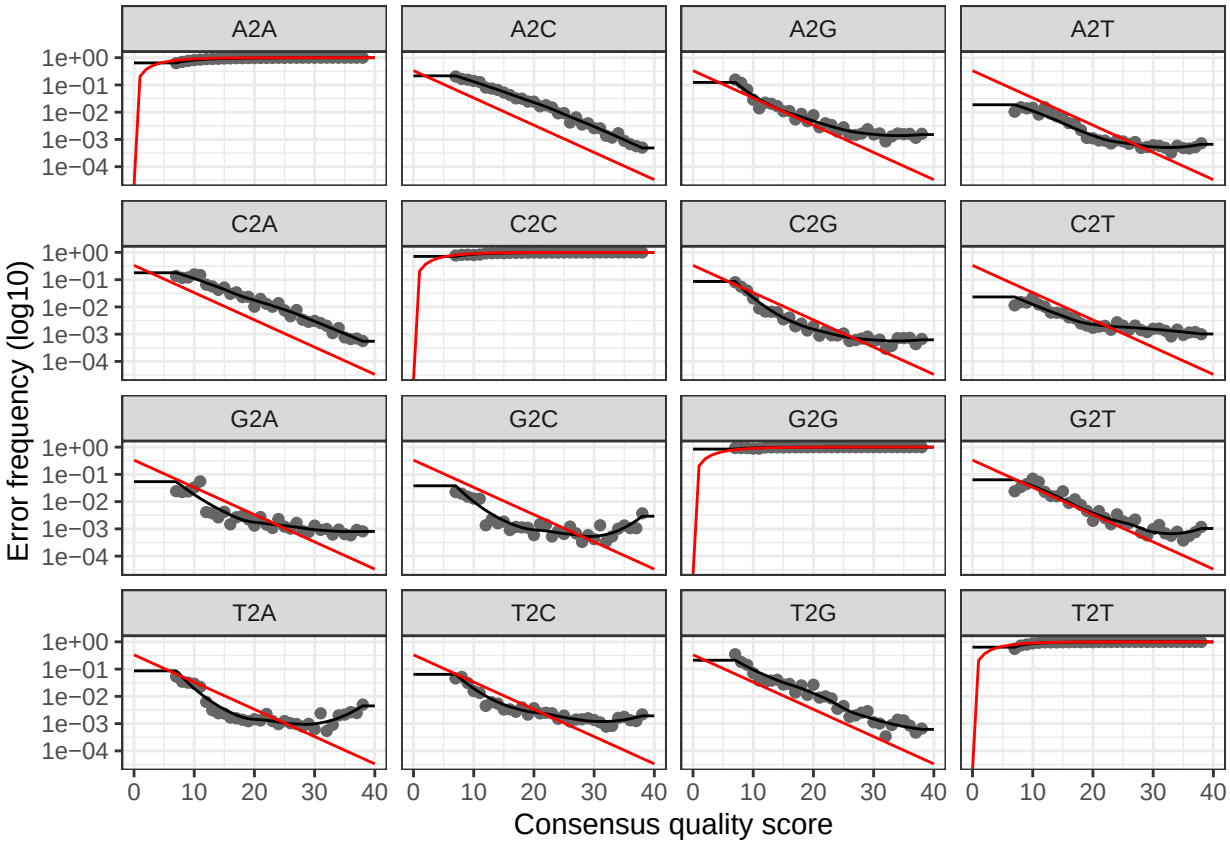
```
load("out.RData")
head(out)
```

```
##                     reads.in reads.out
## in1299_1_R1.fastq      12056     10488
## in1299_10_R1.fastq     18856     17411
```

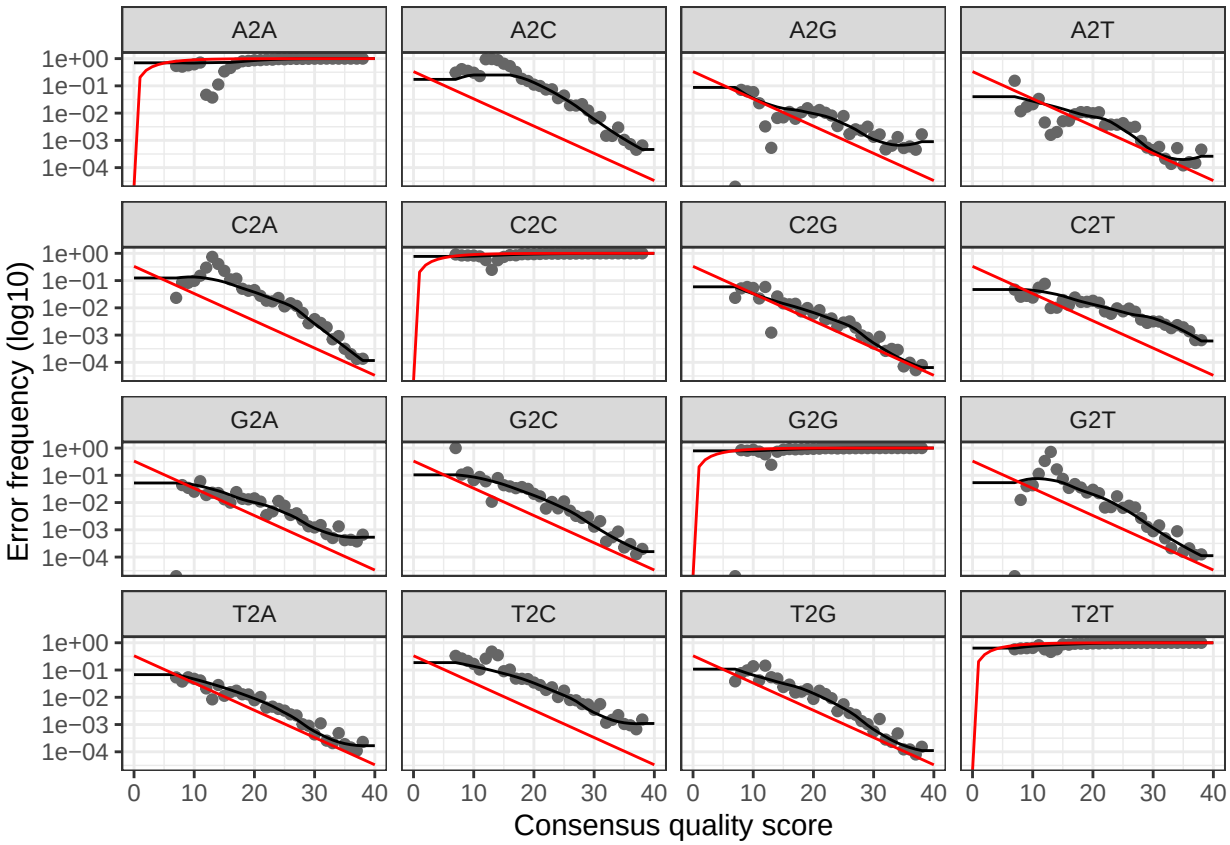
```
## in1299_11_R1.fastq    15372      14497
## in1299_12_R1.fastq    24292      23030
## in1299_13_R1.fastq    16309      14947
## in1299_14_R1.fastq    22322      21099
```

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

save(errF, file="errF.RData")
save(errR, file="errR.RData")
```

```
load("errF.RData")
load("errR.RData")

plotErrors(errF, nominalQ=TRUE)
```




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



Transformation introduced infinite values in continuous y-axis

Apply the core sample inference alogrithm to both the filtered and trimmed sequence data

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

```
## Sample 1 - 10488 reads in 9904 unique sequences.
## Sample 2 - 17411 reads in 14101 unique sequences.
## Sample 3 - 14497 reads in 12151 unique sequences.
## Sample 4 - 23030 reads in 17424 unique sequences.
## Sample 5 - 14947 reads in 13394 unique sequences.
## Sample 6 - 21099 reads in 16362 unique sequences.
## Sample 7 - 18615 reads in 13699 unique sequences.
## Sample 8 - 18426 reads in 15409 unique sequences.
## Sample 9 - 5461 reads in 4666 unique sequences.
## Sample 10 - 8862 reads in 7222 unique sequences.
## Sample 11 - 16641 reads in 13543 unique sequences.
## Sample 12 - 17065 reads in 14948 unique sequences.
## Sample 13 - 19002 reads in 16449 unique sequences.
## Sample 14 - 13409 reads in 12073 unique sequences.
## Sample 15 - 23643 reads in 20485 unique sequences.
## Sample 16 - 20788 reads in 16532 unique sequences.
## Sample 17 - 18945 reads in 16672 unique sequences.
## Sample 18 - 16872 reads in 14156 unique sequences.
```

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

```
## Sample 1 - 10488 reads in 4087 unique sequences.
## Sample 2 - 17411 reads in 7012 unique sequences.
## Sample 3 - 14497 reads in 7310 unique sequences.
## Sample 4 - 23030 reads in 6710 unique sequences.
## Sample 5 - 14947 reads in 4767 unique sequences.
## Sample 6 - 21099 reads in 6236 unique sequences.
## Sample 7 - 18615 reads in 5973 unique sequences.
## Sample 8 - 18426 reads in 5658 unique sequences.
## Sample 9 - 5461 reads in 2124 unique sequences.
## Sample 10 - 8862 reads in 3678 unique sequences.
## Sample 11 - 16641 reads in 6570 unique sequences.
## Sample 12 - 17065 reads in 6728 unique sequences.
## Sample 13 - 19002 reads in 7115 unique sequences.
## Sample 14 - 13409 reads in 4458 unique sequences.
## Sample 15 - 23643 reads in 8331 unique sequences.
## Sample 16 - 20788 reads in 7639 unique sequences.
## Sample 17 - 18945 reads in 6765 unique sequences.
## Sample 18 - 16872 reads in 5975 unique sequences.
```

```
dadaFs[[1]]
```

```
## dada-class: object describing DADA2 denoising results
## 17 sequence variants were inferred from 9904 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, OMEGA_C = 1e-40, BAND_SIZE = 16
```

Obtain the full denoised sequence

is it possible to create 3 objects in a function to try out different parameters

```
mergers <- mergePairs(dadaFs, filtFs, dadaRs, filtRs, verbose=TRUE, minOverlap = 12)
```

```
head(mergers[[1]])
```

```
##
## 1                              GGCTGCAGTTAGGAATCTTCGTCAATGGGCGAAAGCCTGAACGAGCGACGCCGCTTGAGGGACGAAGCCCT
## 2 GGCTGCAGTGGGGAATTTTGGACAATGGGCGAAAGCCTGATCCAGCAATGCCGCGTGTGTGAAGAAGGCCTTCGGGTTGTAAAGCACTTTTGTCCGGA
## 3  GGCAGCAGCCAGGAATCTTGCGCAATGGGCGAAAGCCTGACGCAGCAACGCCGCGTGGGCGATGAAGGCCTTCGGGTCGTAAAGCCCTGTTGTCGGG
## 4                      GGCTGCAGTGGGGAATATTGCACAATGGGCGAAAGCCTGATGCAGCGACGCCGCGTGAGGGATGACGGCCTTCGGGTT
## 5                       GGCTGCAGTGGGGAATTTTCCGCAATGGGCGAAAGCCTGACGGAGCAAGACCGCGTGAGGGAGGAAGGCTCTTGGGT
## 6      GGCTGCAGTAAGGAATATTGGTCAATGGAGGCAACTCTGAACCAGCCATGCCGCGTGCAGGAAGACAGCCCTCTGGGTCGTAAACTGCTTTTA
##   abundance forward reverse nmatch nmismatch nindel prefer accept
## 1      1254       1       1     60         0      0      2   TRUE
## 2      1200       2       2     33         0      0      2   TRUE
## 3      1089       3       3     34         0      0      2   TRUE
## 4      1003       6       4     53         0      0      2   TRUE
## 5       738       9       7     54         0      0      2   TRUE
## 6       698       4       6     38         0      0      2   TRUE
```

Construct sequence table

```
seqtab <- makeSequenceTable(mergers)
dim(seqtab)
```

```
## [1]  18 376
```

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

Distribution of sequence lengths

```
##
## 310 323 399 402 415 417 418 419 420 421 422 423 424 427 429 430 434 436 437 439
##   1   1   1   1   6  62   6   2   1   1  30   3   2   1   1   2   2   1  21   1
## 441 442 443
##  10 166  54
```

Remove non-target-length sequence Want to make sure if this could work

```
seqtab <- seqtab[,nchar(colnames(seqtab)) %in% seq(399,443)]

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

```
##
## 399 402 415 417 418 419 420 421 422 423 424 427 429 430 434 436 437 439 441 442
##   1   1   6  62   6   2   1   1  30   3   2   1   1   2   2   1  21   1  10 166
## 443
##  54
```

Remove chimeras

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

```
## [1]  18 247
```

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

```
## [1] 0.8967427
```

Track reads through the pipeline

```
getN <- function(x) sum(getUniques(x))
track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN), sapply(mergers, getN), rowSums(seqtab.n
# If processing a single sample, remove the sapply calls: e.g. replace sapply(dadaFs, getN) with getN(d
colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")
rownames(track) <- sample.names
head(track)
```

```
##           input filtered denoisedF denoisedR merged nonchim
```

```
## in1299_1  12056    10488     10133     10361   9948    9948
## in1299_10 18856    17411     17318     17231  16993   15533
## in1299_11 15372    14497     13798     14265  12705   11824
## in1299_12 24292    23030     22660     22749  21838   17771
## in1299_13 16309    14947     14860     14800  14643   14460
## in1299_14 22322    21099     20923     20852  20567   17158
```

Most reads drops in the filter step, which is a good sign.

Not too many reads are removed in the chimeras steps, which is a good sign.

Assign taxonomy

Silva reference database use in classifying prokaryotic 16S sequencing data

```
taxa <- assignTaxonomy(seqtab.nochim, "./tax/silva_nr99_v138.1_train_set.fa.gz", multithread=TRUE)
```

```
taxa <- addSpecies(taxa, "./tax/silva_species_assignment_v138.1.fa.gz")
```

exact matching ASVs and sequenced reference strains to assign species

```
taxa.print <- taxa
rownames(taxa.print) <- NULL
head(taxa.print)
```

taxonomy table

```
##      Kingdom    Phylum           Class                 Order
## [1,] "Bacteria" "Firmicutes"     "Bacilli"             "Bacillales"
## [2,] "Bacteria" "Proteobacteria" "Gammaproteobacteria" "Burkholderiales"
## [3,] "Bacteria" "Firmicutes"     "Bacilli"             "Lactobacillales"
## [4,] "Bacteria" "Firmicutes"     "Bacilli"             "Staphylococcales"
## [5,] "Bacteria" "Firmicutes"     "Bacilli"             "Lactobacillales"
## [6,] "Bacteria" "Firmicutes"     "Bacilli"             "Lactobacillales"
##      Family              Genus                 Species
## [1,] "Bacillaceae"       "Bacillus"            NA
## [2,] "Burkholderiaceae"  "Ralstonia"           NA
## [3,] "Enterococcaceae"   "Enterococcus"        NA
## [4,] "Staphylococcaceae" "Staphylococcus"      NA
## [5,] "Listeriaceae"      "Listeria"            NA
## [6,] "Lactobacillaceae"  "Limosilactobacillus" NA
```

Phyloseq

Library

```
library(phyloseq); packageVersion("phyloseq")
```

```
## [1] '1.34.0'
```

```
library(Biostrings); packageVersion("Biostrings")
```

```
## [1] '2.58.0'
```

```
library(ggplot2); packageVersion("ggplot2")
```

```
## [1] '3.3.4'
```

```
library(stringr)
library(readxl)
theme_set(theme_bw())
```

Should have a file for sample data

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

```
samples.out <- rownames(seqtab.nochim)
subject <- sapply(strsplit(samples.out, "_"), `[`, 2)
sampleTest<-ifelse(subject<=10, "Negative", "Standard")

sampleData<- read_excel("./tax/mappingTable.xlsx", col_names=FALSE) %>% data.frame
colnames(sampleData) <- c("Name", "Value")
sampleData$Test <- ifelse(str_detect(sampleData[[2]], "Negative"), "Negative", "Standard" )
rownames(sampleData) <- sampleData[[1]]
```

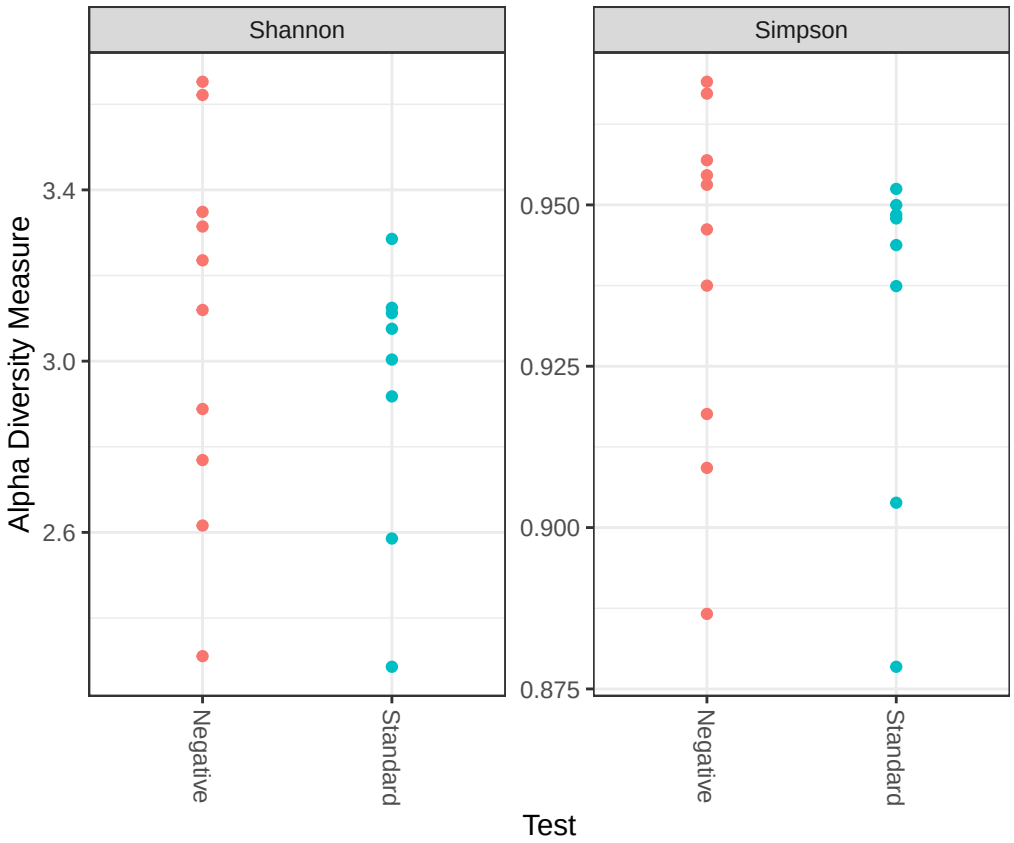
Construct phyloseq object

```
ps <- phyloseq(otu_table(seqtab.nochim, taxa_are_rows=FALSE),
               tax_table(taxa),
               sample_data(sampleData))
ps
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 247 taxa and 18 samples ]
## sample_data() Sample Data:       [ 18 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 247 taxa by 7 taxonomic ranks ]
```

```
saveRDS(ps, "./tax/ps_zymo.rds")
```

```
plot_richness(ps, x="Test", measures=c("Shannon", "Simpson"), color="Test")
```



Alpha-diversity

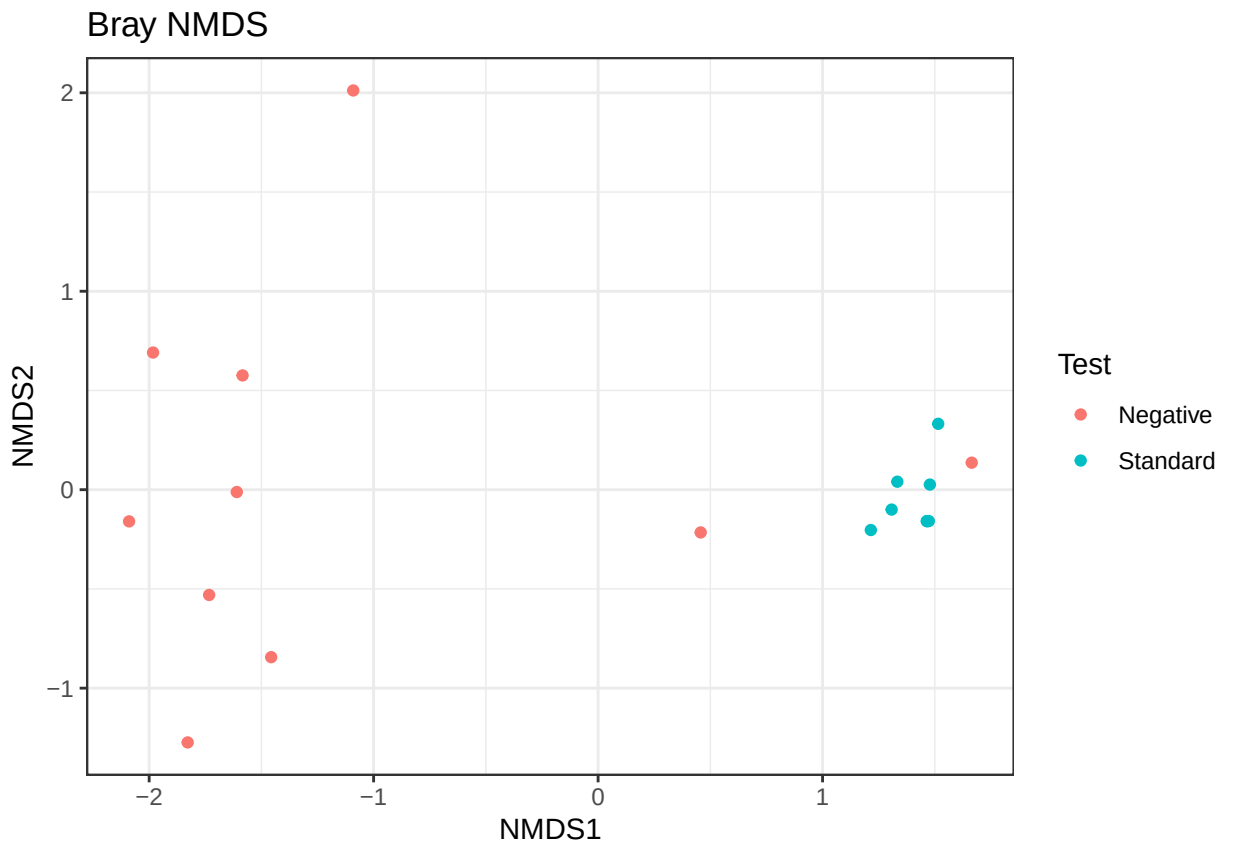
```
ps.prop <- transform_sample_counts(ps, function(otu) otu/sum(otu))
ord.nmds.bray <- ordinate(ps.prop, method="NMDS", distance="bray")
```

Ordination Plot

```
## Run 0 stress 0.06576883
## Run 1 stress 0.06674413
## Run 2 stress 0.0657679
## ... New best solution
## ... Procrustes: rmse 0.001500862  max resid 0.00483192
## ... Similar to previous best
## Run 3 stress 0.08400457
## Run 4 stress 0.06576864
## ... Procrustes: rmse 0.0004771873  max resid 0.001663181
## ... Similar to previous best
## Run 5 stress 0.08602642
## Run 6 stress 0.06576863
## ... Procrustes: rmse 0.0004644849  max resid 0.001616796
## ... Similar to previous best
## Run 7 stress 0.08548887
## Run 8 stress 0.08100562
## Run 9 stress 0.08100578
## Run 10 stress 0.06576884
## ... Procrustes: rmse 0.001506934  max resid 0.004852508
```

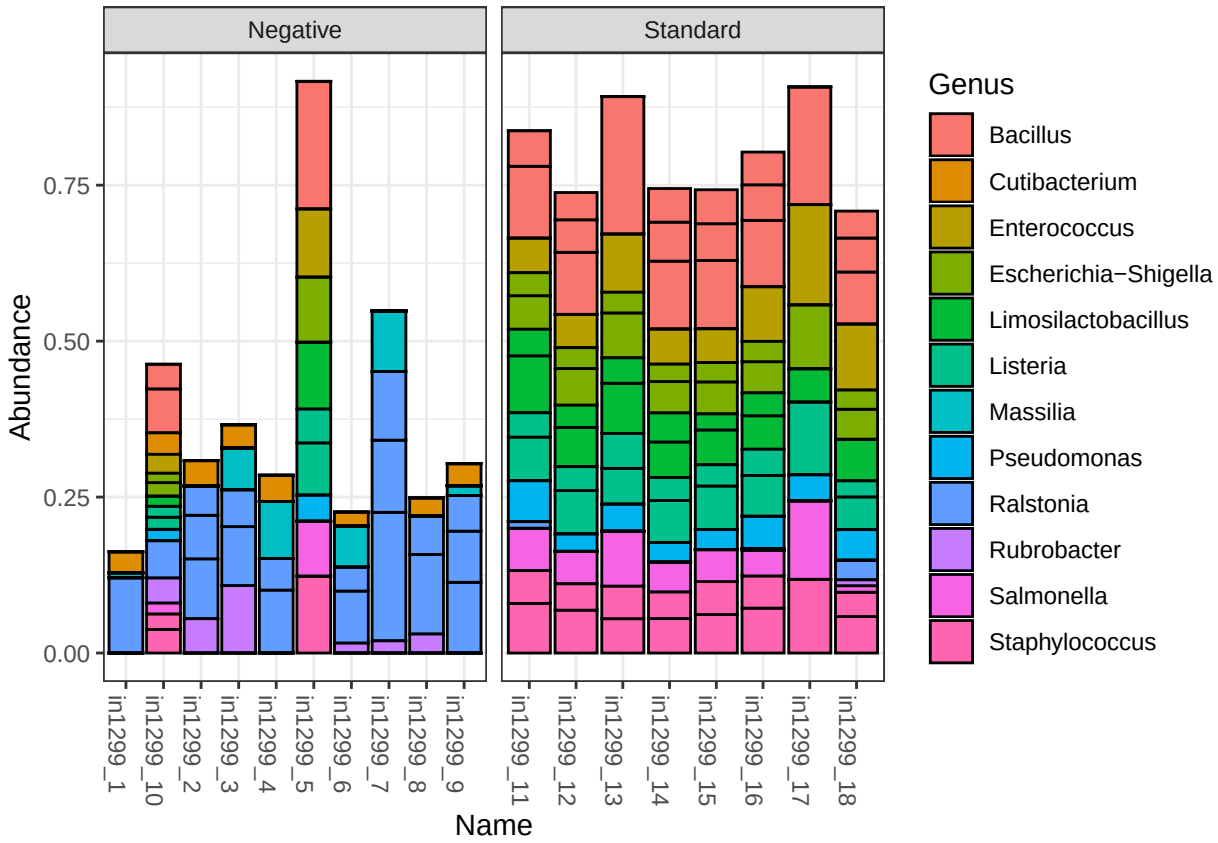
```
## ... Similar to previous best
## Run 11 stress 0.06576807
## ... Procrustes: rmse 0.0009406154  max resid 0.003101439
## ... Similar to previous best
## Run 12 stress 0.06576796
## ... Procrustes: rmse 5.468295e-05  max resid 0.0001904673
## ... Similar to previous best
## Run 13 stress 0.06576801
## ... Procrustes: rmse 9.701586e-05  max resid 0.0003369798
## ... Similar to previous best
## Run 14 stress 0.08602634
## Run 15 stress 0.09139267
## Run 16 stress 0.09026029
## Run 17 stress 0.08762669
## Run 18 stress 0.08400527
## Run 19 stress 0.06576846
## ... Procrustes: rmse 0.0003856261  max resid 0.001318996
## ... Similar to previous best
## Run 20 stress 0.08602635
## *** Solution reached
```

```
plot_ordination(ps, ord.nmds.bray, color="Test", title="Bray NMDS")
```




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

```
ps.top20 <- transform_sample_counts(ps, function(OTU) OTU/sum(OTU))
ps.top20 <- prune_taxa(top20, ps.top20)
plot_bar(ps.top20, x="Name", fill="Genus") + facet_wrap(~Test, scales="free_x")
```



Bar plot

Library

```
setup_example<- (c("igraph", "phyloseq", "phyloseqGraphTest", "ggnetwork", "intergraph","gridExtra"))

lapply(setup_example,require, character.only=TRUE)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
```

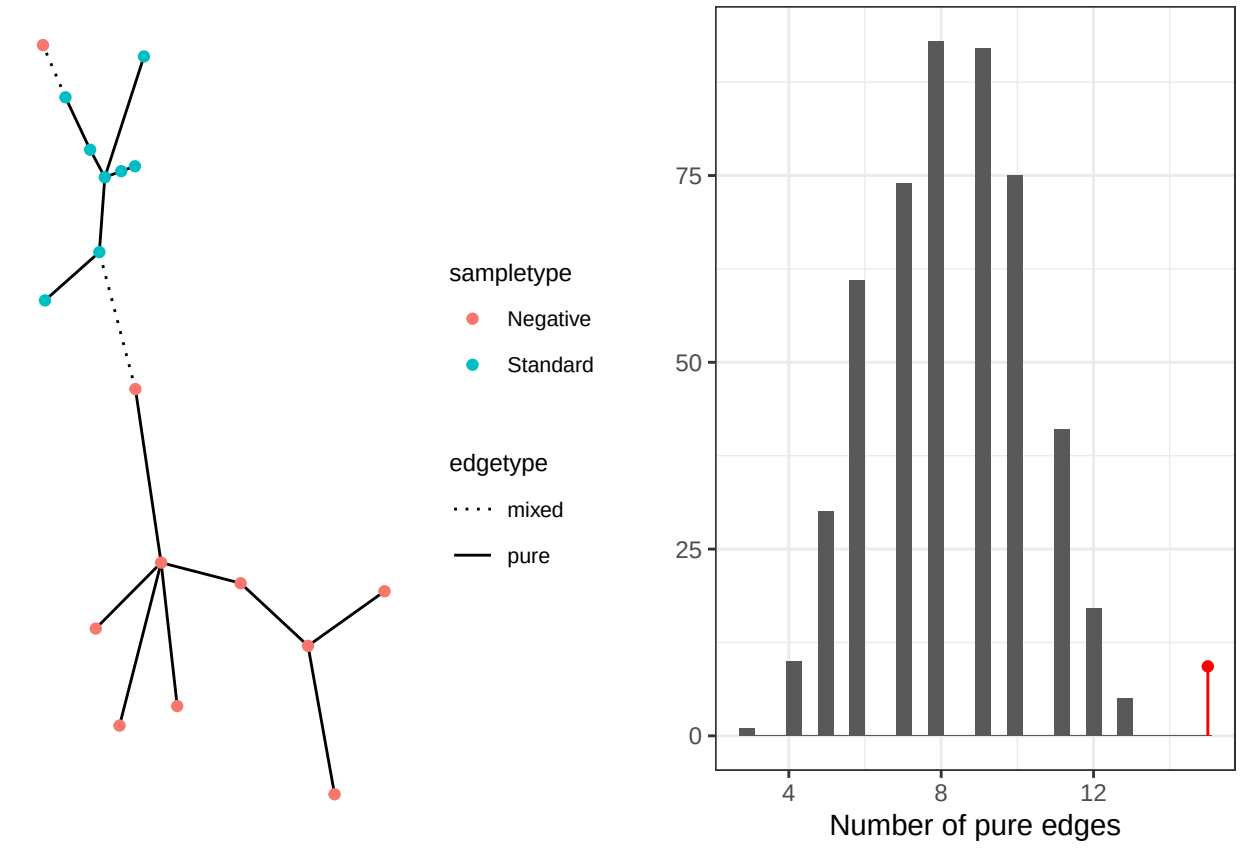
```
##
## [[6]]
## [1] TRUE
```

```
gt <- graph_perm_test(ps, "Test", grouping = "Name",
             distance = "jaccard", type = "mst")
gt$pval
```

Minimum Spanning Tree

```
## [1] 0.002
```

```
plotNet1=plot_test_network(gt) + theme(legend.text = element_text(size = 8),
         legend.title = element_text(size = 9))
plotPerm1=plot_permutations(gt)
grid.arrange(ncol = 2, plotNet1, plotPerm1)
```



Can we claim the Negative control sample in the top left corner as a possible contaminant?

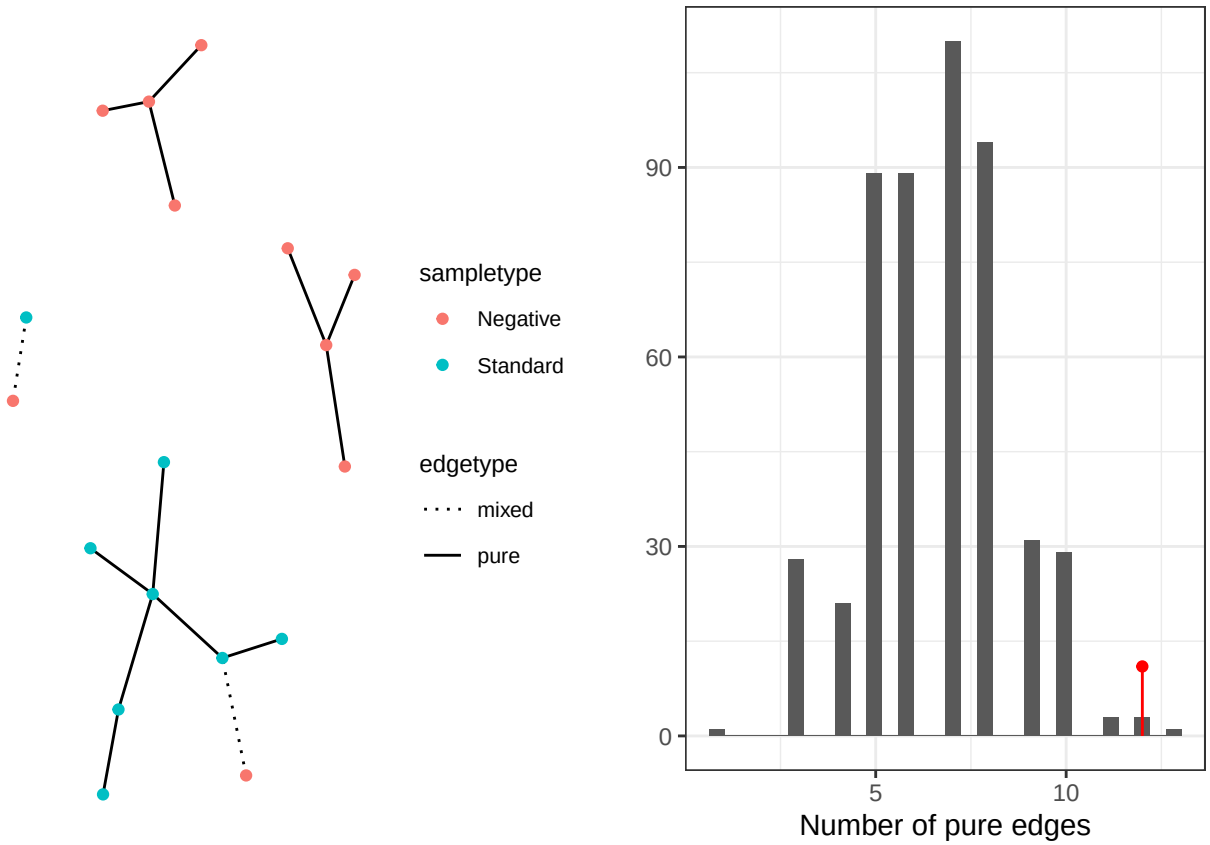
```
gt <- graph_perm_test(ps, "Test", grouping = "Name",
             distance = "jaccard", type = "knn", knn = 1)
```

```
gt$pval
```

Two-nearest neighbors with the Bray-Curtis dissimilarity

```
## [1] 0.01
```

```
plotNet1=plot_test_network(gt) + theme(legend.text = element_text(size = 8),
         legend.title = element_text(size = 9))
plotPerm1=plot_permutations(gt)
grid.arrange(ncol = 2, plotNet1, plotPerm1)
```



Both graph-based visualization shows that number of pure edges is more than the test statistics, so we reject the null hypotehsis of two samples come from the same distribution.

BARBI

Reference: https://pratheepaj.github.io/BARBI/articles/BARBI.html

Library

```
library(BARBI)
library(phyloseq)
library(dplyr)
library(HDInterval)
library(grid)
library(gtable)
```

```
library(gridExtra)
library(magrittr)
library(ggplot2)
library(DESeq2)
library(reshape2)
library(ggwordcloud)
```

```
if(dim(otu_table(ps))[1]!=ntaxa(ps)){
  otu_table(ps) <- t(otu_table(ps))}
```

```
blocks <- rep("Set1", nsamples(ps))

sample_data(ps)$block <- blocks
```

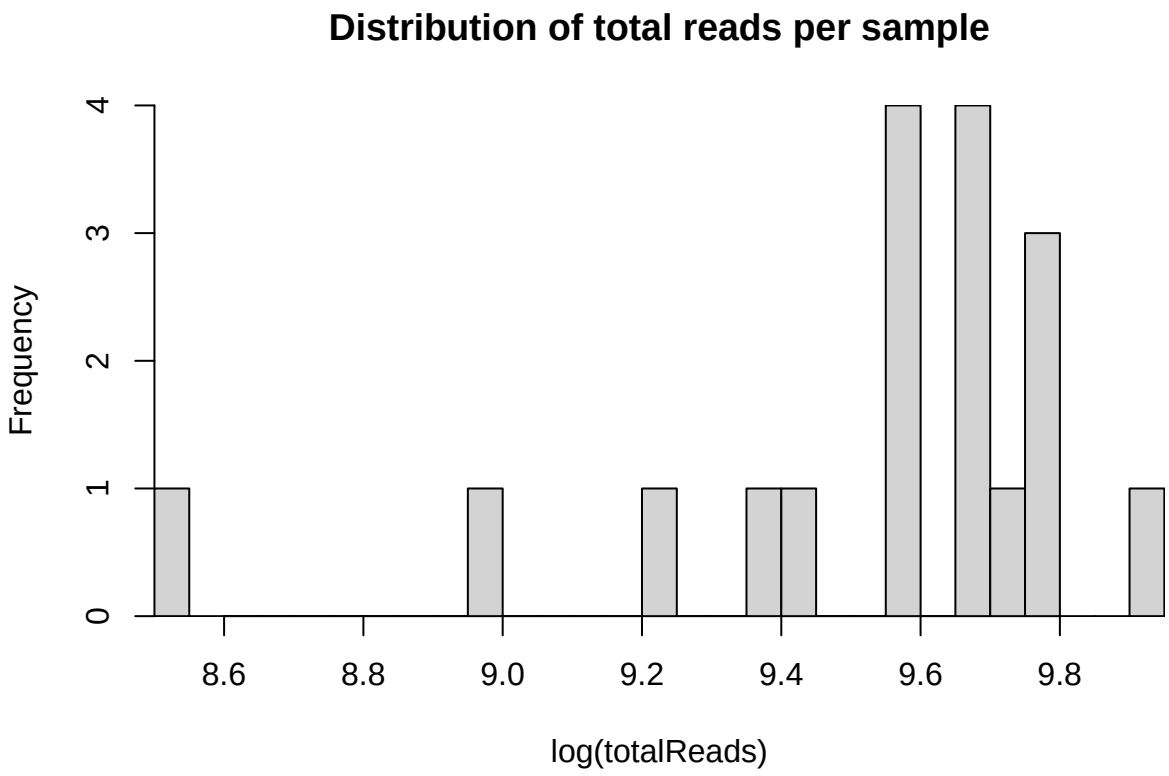
```
ps2 <- prune_taxa(taxa_sums(ps) > 0, ps)
ps_specimen <-  subset_samples(ps2,
                               Test %in% c("Standard"))
prevTaxaP <- apply(otu_table(ps_specimen), 1,
                   function(x){sum(x>0)})

Contaminants1 <- names(prevTaxaP)[prevTaxaP == 0]
ps2 <- prune_taxa(prevTaxaP > 0, ps2)
ps2
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 71 taxa and 18 samples ]
## sample_data() Sample Data:       [ 18 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 71 taxa by 7 taxonomic ranks ]
```

Had 247 taxa, but 71 remained

Library Depth

```
totalReads <- colSums(otu_table(ps))
hist(log(totalReads),
     yaxs="i",
     xaxs="i",
     main="Distribution of total reads per sample",
     breaks=50)
```



Phyloseq for BARBI method

I should consider changing Test column to sampleType and Name to sampleID

```
psBlockResult <- psBlockResults(ps2,
                                sampleTypeVar = "Test",
                                caselevels = c("Standard"),
                                controllevel = c("Negative"),
                                sampleName = "Name",
                                blockVar = "block")

psByBlock <- psBlockResult[[1]] # The original phyloseq object
psNCbyBlock <- psBlockResult[[2]] # Negative Control Samples
psallzeroInNC <- psBlockResult[[3]] # prevalence of zero
psPlByBlock <- psBlockResult[[4]] # specimen samples
```

Estimate parameters for contaminant intensities in negative control samples

partial information about contamination intensities available in negative controls

```
con_int_neg_ctrl <- alphaBetaNegControl(psNCbyBlock = psNCbyBlock)
```

Density parameters for contaminant intensities should this be true intensities

```
num_blks <- length(con_int_neg_ctrl)
blks <- seq(1, num_blks) %>% as.list

con_int_specimen <- lapply(blks, function(x){
    con_int_specimen_each_blk <- alphaBetaContInPlasma(psPlByBlock = psPlByBlock,
                                                       psallzeroInNC = psallzeroInNC,
                                                       blk = x,
                                                       alphaBetaNegControl = con_int_neg_ctrl)
        return(con_int_specimen_each_blk)
})
```

Sample from marginal posterior for true intensities

```
itera = 100
t1 <- proc.time()

mar_post_true_intensities <- lapply(blks,function(x){
    mar_post_true_intensities_each_blk <- samplingPosterior(psPlByBlock = psPlByBlock,
                                                            blk = x,
                                                            gammaPrior_Cont = con_int_specimen[[x]],
                                                            itera = itera)
    return(mar_post_true_intensities_each_blk)
})

proc.time()-t1
```

```
##    user  system elapsed
##    8.30    0.03    8.33
```

```
con_int_specimen_mar_post_true_intensities <- list(con_int_specimen, mar_post_true_intensities)
```

Tables for each sample

```
ASV <- as.character(paste0("ASV_",seq(1,ntaxa(ps))))
ASV.Genus <- paste0("ASV_",seq(1,ntaxa(ps)),"_",as.character(tax_table(ps)[,6]))
ASV.Genus.Species <- paste0(ASV,"_",as.character(tax_table(ps)[,6]),"_", as.character(tax_table(ps)[,7]

df.ASV <- data.frame(seq.variant = taxa_names(ps), ASV = ASV, ASV.Genus = ASV.Genus, ASV.Genus.Species =
```

```
itera <- 100
burnIn <- 10
cov.pro <- .95
mak_tab <- FALSE # Save tables or print tables

# con_int_specimen_mar_post_true_intensities <- readRDS("./con_int_specimen_mar_post_true_intensities_v

con_int_specimen <- con_int_specimen_mar_post_true_intensities[[1]]
mar_post_true_intensities <- con_int_specimen_mar_post_true_intensities[[2]]

## Keep true
```

```
all_true_taxa_blk <- list()

for(blk in 1:num_blks){

  mar_post_true_intensities_blk <- mar_post_true_intensities[[blk]]
  con_int_specimen_blk <- con_int_specimen[[blk]]

  all_true_taxa <- character()

  for(sam in 1:nsamples(psPlByBlock[[blk]])){
      taxa_post <- mar_post_true_intensities_blk[[sam]]
      acceptance <- list()
      lower.r <- list()
      upper.r <- list()
      lower.c <- list()
      upper.c <- list()
      all.zero.nc <- list()

      for(taxa in 1:length(taxa_post)){
        burnIn  <- burnIn
        acceptance[[taxa]]  <-  1 - mean(duplicated(taxa_post[[taxa]][-(1:burnIn),]))

        HPD.r <- hdi(taxa_post[[taxa]][-(1:burnIn),],
                     credMass = cov.pro)
        lower.r[[taxa]] <- round(HPD.r[1], digits = 0)
        upper.r[[taxa]] <- round(HPD.r[2], digits = 0)
        lamda.c <- rgamma((itera-burnIn+1),
                     shape= con_int_specimen_blk[[sam]][[1]][taxa],
                     rate = con_int_specimen_blk[[sam]][[2]][taxa])

        HDI.c <- hdi(lamda.c, credMass = cov.pro)
        lower.c[[taxa]] <- round(HDI.c[1], digits = 0)
        upper.c[[taxa]] <- round(HDI.c[2], digits = 0)

        all.zero.nc[[taxa]] <-  con_int_specimen_blk[[sam]][[5]][taxa]
      }

    tax_names <- taxa_names(psPlByBlock[[blk]])
    tax_names <- df.ASV$ASV.Genus[which(as.character(df.ASV$seq.variant) %in%  tax_names)]

    df <- data.frame(Species = tax_names,
                     xj = as.numeric(con_int_specimen_blk[[sam]][[3]]),
                     l.r = unlist(lower.r),
                     u.r = unlist(upper.r),
                     l.c = unlist(lower.c),
                     u.c = unlist(upper.c),
                     all.zero.nc = unlist(all.zero.nc))


      # List all true taxa
      df <- arrange(filter(df,(l.r > u.c) & (l.r > 0)),
                    desc(xj))
```

```
    # If there is no true taxa
    if(dim(df)[1]==0){
        df <- data.frame(Species="Negative",
                         xj="Negative",
                         l.r="Negative",
                         u.r="Negative",
                         l.c ="Negative",
                         u.c="Negative",
                         all.zero.nc = "Negative")
    }



    # collect all true taxa in the specimen
    all_true_taxa <- c(all_true_taxa,
                       as.character(df$Species))

    if(mak_tab){
      filname <- paste("./",
                       sample_names(psPlByBlock[[blk]])[sam],
                      ".png",
                      sep = "")

      png(filname, height = 600, width = 750)

      df.p <- tableGrob(df)
      title <- textGrob(sample_names(psPlByBlock[[blk]])[sam],
                        gp = gpar(fontsize = 12))

      padding <- unit(0.5,"line")

      df.p <- gtable_add_rows(df.p,
                              heights = grobHeight(title) + padding,
                              pos = 0)

      df.p <- gtable_add_grob(df.p,
                              list(title),
                              t = 1,
                              l = 1,
                              r = ncol(df.p))

      grid.newpage()
      grid.draw(df.p)
      dev.off()

    }else{
      df.p <- tableGrob(df)
      title <- textGrob(sample_names(psPlByBlock[[blk]])[sam],
                        gp = gpar(fontsize = 12))

      padding <- unit(0.5,"line")

      df.p <- gtable_add_rows(df.p,
```

```
                              heights = grobHeight(title) + padding,
                              pos = 0)

        df.p <- gtable_add_grob(df.p,
                              list(title),
                              t = 1,
                              l = 1,
                              r = ncol(df.p))
        grid.newpage()
        grid.draw(df.p)
      }

      all_true_taxa <- unique(all_true_taxa)
  }

  all_true_taxa_blk[[blk]] <- all_true_taxa
}
```

2	ASV_6_Limosilactobacillus	1080	1026	1125	0	85	No
3	ASV_4_Staphylococcus	941	889	976	0	158	No
4	ASV_5_Listeria	826	774	888	0	79	No
5	ASV_9_Salmonella	801	748	865	0	76	No
6	ASV_13_Pseudomonas	776	715	824	0	62	No
7	ASV_11_Bacillus	674	616	711	0	78	No
8	ASV_3_Enterococcus	656	616	701	0	144	No
9	ASV_8_Escherichia−Shigella	636	593	687	0	86	No
10	ASV_12_Staphylococcus	622	581	664	0	109	No
11	ASV_17_Limosilactobacillus	503	461	542	0	45	No
12	ASV_25_Salmonella	486	447	524	0	29	No
13	ASV_14_Listeria	461	421	486	0	51	No
14	ASV_19_Escherichia−Shigella	435	400	479	0	26	No
15	ASV_28_Enterococcus	433	398	470	0	0	Yes
16	ASV_67_Escherichia−Shigella	192	161	216	0	0	Yes
17	ASV_26_Bacillus	191	169	218	0	40	No
18	ASV_46_Limosilactobacillus	179	154	203	0	0	Yes

7	ASV_9_Salmonella	925	882	988	0	154	No
8	ASV_11_Bacillus	925	873	967	0	122	No
9	ASV_18_Bacillus	777	715	820	0	0	Yes
10	ASV_12_Staphylococcus	750	704	799	0	70	No
11	ASV_14_Listeria	681	642	737	0	121	No
12	ASV_28_Enterococcus	676	623	724	0	0	Yes
13	ASV_17_Limosilactobacillus	633	587	683	0	32	No
14	ASV_19_Escherichia−Shigella	593	548	624	0	35	No
15	ASV_22_Staphylococcus	577	524	610	0	38	No
16	ASV_24_Listeria	577	538	624	0	82	No
17	ASV_13_Pseudomonas	498	452	533	0	64	No
18	ASV_25_Salmonella	492	459	534	0	28	No
19	ASV_26_Bacillus	401	369	433	0	29	No
20	ASV_39_Bacillus	401	359	427	0	0	Yes
21	ASV_40_Bacillus	303	278	330	0	0	Yes
22	ASV_67_Escherichia−Shigella	296	266	331	0	0	Yes

	Species	xj	i.r	d.r	i.c	d.c	all.zero.nc
1	ASV_1_Bacillus	3186	3075	3266	0	364	No
2	ASV_3_Enterococcus	1349	1287	1395	0	171	No
3	ASV_9_Salmonella	1277	1201	1341	0	139	No
4	ASV_6_Limosilactobacillus	1163	1097	1224	0	92	No
5	ASV_8_Escherichia−Shigella	1041	988	1100	0	150	No
6	ASV_14_Listeria	830	784	885	0	66	No
7	ASV_5_Listeria	809	771	858	0	159	No
8	ASV_12_Staphylococcus	796	751	833	0	73	No
9	ASV_4_Staphylococcus	752	710	801	0	271	No
10	ASV_13_Pseudomonas	626	588	679	0	58	No
11	ASV_17_Limosilactobacillus	590	553	640	0	43	No
12	ASV_19_Escherichia−Shigella	479	440	511	0	29	No
13	ASV_22_Staphylococcus	458	429	494	0	43	No
14	ASV_24_Listeria	432	388	466	0	41	No
15	ASV_45_Enterococcus	286	260	321	0	0	Yes
16	ASV_53_Salmonella	223	193	250	0	15	No

5	ASV_3_Enterococcus	966	912	1014	0	250	No
6	ASV_4_Staphylococcus	948	888	993	0	301	No
7	ASV_18_Bacillus	925	879	975	0	0	Yes
8	ASV_8_Escherichia−Shigella	865	820	912	0	180	No
9	ASV_28_Enterococcus	836	778	891	0	0	Yes
10	ASV_9_Salmonella	825	765	862	0	177	No
11	ASV_17_Limosilactobacillus	798	759	851	0	54	No
12	ASV_12_Staphylococcus	729	689	768	0	82	No
13	ASV_14_Listeria	631	573	664	0	105	No
14	ASV_22_Staphylococcus	563	519	594	0	55	No
15	ASV_24_Listeria	542	500	604	0	47	No
16	ASV_13_Pseudomonas	539	495	570	0	97	No
17	ASV_25_Salmonella	517	469	559	0	54	No
18	ASV_19_Escherichia−Shigella	477	447	509	0	60	No
19	ASV_26_Bacillus	464	429	508	0	77	No
20	ASV_39_Bacillus	422	396	459	0	0	Yes
21	ASV_40_Bacillus	337	314	372	0	0	Yes

6	ASV_18_Bacillus	801	767	863	0	0	Yes
7	ASV_3_Enterococcus	798	736	841	0	210	No
8	ASV_12_Staphylococcus	777	740	839	0	68	No
9	ASV_8_Escherichia−Shigella	755	705	796	0	105	No
10	ASV_9_Salmonella	753	702	796	0	140	No
11	ASV_22_Staphylococcus	615	572	647	0	46	No
12	ASV_24_Listeria	540	497	574	0	18	No
13	ASV_14_Listeria	501	469	545	0	138	No
14	ASV_13_Pseudomonas	476	439	505	0	113	No
15	ASV_19_Escherichia−Shigella	455	413	489	0	52	No
16	ASV_28_Enterococcus	392	354	439	0	0	Yes
17	ASV_25_Salmonella	388	351	421	0	71	No
18	ASV_17_Limosilactobacillus	378	333	408	0	50	No
19	ASV_39_Bacillus	373	343	409	0	0	Yes
20	ASV_26_Bacillus	302	264	330	0	28	No
21	ASV_40_Bacillus	266	237	298	0	0	Yes
22	ASV_80_Staphylococcus	195	168	220	0	56	No

5	ASV_11_Bacillus	890	840	956	0	152	No
6	ASV_6_Limosilactobacillus	838	786	880	0	103	No
7	ASV_18_Bacillus	819	767	876	0	0	Yes
8	ASV_13_Pseudomonas	812	762	873	0	62	No
9	ASV_12_Staphylococcus	805	765	867	0	54	No
10	ASV_8_Escherichia−Shigella	779	731	823	0	157	No
11	ASV_14_Listeria	654	607	723	0	126	No
12	ASV_9_Salmonella	637	588	693	0	140	No
13	ASV_17_Limosilactobacillus	573	526	610	0	47	No
14	ASV_22_Staphylococcus	571	542	612	0	72	No
15	ASV_24_Listeria	570	535	610	0	51	No
16	ASV_19_Escherichia−Shigella	508	453	532	0	46	No
17	ASV_25_Salmonella	470	435	503	0	33	No
18	ASV_26_Bacillus	342	309	371	0	29	No
19	ASV_45_Enterococcus	276	238	308	0	0	Yes
20	ASV_40_Bacillus	269	246	300	0	0	Yes

in1299_17

	Species	xj	l.r	u.r	l.c	u.c	all.zero.nc
1	ASV_1_Bacillus	954	885	991	0	48	No
2	ASV_3_Enterococcus	813	763	866	0	91	No
3	ASV_9_Salmonella	637	605	689	0	80	No
4	ASV_4_Staphylococcus	597	549	633	0	142	No
5	ASV_5_Listeria	590	540	626	0	51	No
6	ASV_8_Escherichia−Shigella	518	475	547	0	30	No
7	ASV_17_Limosilactobacillus	269	243	291	0	6	No
8	ASV_13_Pseudomonas	211	190	241	0	16	No
9	ASV_53_Salmonella	163	133	182	0	16	No
10	ASV_217_Corynebacterium	69	54	86	0	0	Yes
11	ASV_220_Bradyrhizobium	50	38	65	0	1	No
12	ASV_238_PMMR1	27	17	36	0	0	Yes
13	ASV_240_Kytococcus	18	10	24	0	0	Yes

5	ASV_11_Bacillus	436	396	464	0	62	No
6	ASV_5_Listeria	420	387	468	0	34	No
7	ASV_13_Pseudomonas	393	365	432	0	38	No
8	ASV_8_Escherichia−Shigella	386	338	416	0	113	No
9	ASV_18_Bacillus	346	322	377	0	0	Yes
10	ASV_12_Staphylococcus	312	279	334	0	46	No
11	ASV_19_Escherichia−Shigella	246	221	278	0	26	No
12	ASV_22_Staphylococcus	229	202	252	0	44	No
13	ASV_14_Listeria	206	178	229	0	72	No
14	ASV_26_Bacillus	158	132	176	0	19	No
15	ASV_45_Enterococcus	143	124	163	0	0	Yes
16	ASV_195_Hydrogenophilus	128	107	152	0	0	Yes
17	ASV_46_Limosilactobacillus	119	101	140	0	0	Yes
18	ASV_201_Sphingomonas	110	86	128	0	0	Yes
19	ASV_205_Planomicrobium	95	77	113	0	0	Yes
20	ASV_213_Effusibacillus	81	67	102	0	0	Yes
21	ASV_215_Pseudonocardia	79	64	93	0	0	Yes

```
all_true_taxa_blk <- unlist(all_true_taxa_blk)
ASV = df.ASV$seq.variant[which(as.character(df.ASV$ASV.Genus) %in% as.character(all_true_taxa_blk))] %>%
ps_decon <- prune_taxa(ASV, ps)
ps_decon
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 53 taxa and 18 samples ]
## sample_data() Sample Data:       [ 18 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 53 taxa by 7 taxonomic ranks ]
```

LDA

```
short.sample.names = c(paste0("NC.", c(1,10)),
                       paste0("Di.", seq(1,8)),
                       paste0("NC.", seq(2,9)))
sample_names(ps) = short.sample.names

x = t(get_taxa(ps))
dimnames(x) = NULL
K = 4
stan.data <- list(K = K,
  V = ncol(x),
  D = nrow(x),
  n = x,
  alpha = rep(1, K),
```

```
  gamma = rep(0.5, ncol(x),
              control=list(max_treedepth=50))
)

stan.fit = LDAtopicmodel(stan_data = stan.data, iter = 100, chains = 1)
```

```
##
## SAMPLING FOR MODEL 'lda' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 7
## Chain 1:            adapt_window = 38
## Chain 1:            term_buffer = 5
## Chain 1:
## Chain 1: Iteration:  1 / 100 [  1%]  (Warmup)
## Chain 1: Iteration: 10 / 100 [ 10%]  (Warmup)
## Chain 1: Iteration: 20 / 100 [ 20%]  (Warmup)
## Chain 1: Iteration: 30 / 100 [ 30%]  (Warmup)
## Chain 1: Iteration: 40 / 100 [ 40%]  (Warmup)
## Chain 1: Iteration: 50 / 100 [ 50%]  (Warmup)
## Chain 1: Iteration: 51 / 100 [ 51%]  (Sampling)
## Chain 1: Iteration: 60 / 100 [ 60%]  (Sampling)
## Chain 1: Iteration: 70 / 100 [ 70%]  (Sampling)
## Chain 1: Iteration: 80 / 100 [ 80%]  (Sampling)
## Chain 1: Iteration: 90 / 100 [ 90%]  (Sampling)
## Chain 1: Iteration: 100 / 100 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 22.312 seconds (Warm-up)
## Chain 1:                58.021 seconds (Sampling)
## Chain 1:                80.333 seconds (Total)
## Chain 1:
```

attempting to adjust max_treedepth as recommended

Extract Posterior samples

```
samples = rstan::extract(stan.fit, permuted = TRUE, inc_warmup = FALSE, include = TRUE)
```

Word Cloud

```
beta = samples$beta
dimnames(beta)[[2]] = c(paste0("Topic ", seq(1,K)))
```

```
tax_tab = tax_table(ps) %>% data.frame()
tax_tab = mutate(tax_tab, seq.variant = rownames(tax_tab))

dimnames(beta)[[3]] =tax_tab[, "seq.variant"]
beta.all = melt(beta)
colnames(beta.all) = c("Chain", "Topic", "ASV", "ASV.distribution")
beta.all$ASV = as.character(beta.all$ASV)
beta.all = left_join(beta.all, tax_tab, by = c("ASV"= "seq.variant"))
beta.all$Topic = factor(beta.all$Topic)
beta.all$ASV = factor(beta.all$ASV)
```

```
max.beta.in.each.asv.all.topics = group_by(beta.all,
                                           Topic,
                                           Family,
                                           Genus) %>% summarise(max_beta = max(ASV.distribution)) %>% to


ggplot(max.beta.in.each.asv.all.topics,
                  aes(label = Genus, size = max_beta, color = Family)) +
  geom_text_wordcloud() +
  theme_minimal() +
  scale_size_area(max_size = 8) +
  facet_wrap(~ Topic) +
  theme(strip.text.x = element_text(size = 12, face = "bold"))
```