16S Microbiome Data Visualization

Long Tian

2/15/2019

Introduction to this script

This script is for the Vinatzer lab's need of 16S microbiome data analysis. Taking the output BIOM file and a couple of intermediate files from QIIME as well as the mapping file, this script aims to analyze and visualize the data with the R package of `phyloseq` from `BioConductor` and `ggplot2`, analyses will include but not limitted to *abundance analysis*, *alpha diversity*, *beta diversity*, and *hierarchical clustering* of the samples.

Set working directory to where the input files are/ will be

```
setwd("/Users/longtian/Desktop/VinatzerLab/Dianthus")
```

Library/package needed

```
source('http://bioconductor.org/biocLite.R')
```

```
## Warning: 'BiocInstaller' and 'biocLite()' are deprecated, use the 'BiocManager'
##   CRAN package instead.
```

```
## Bioconductor version 3.8 (BiocInstaller 1.32.1), ?biocLite for help
```

```
biocLite('phyloseq')
```

```
## Warning: 'biocLite' is deprecated.
## Use 'BiocManager::install' instead.
## See help("Deprecated")
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.8 (BiocInstaller 1.32.1), R 3.5.2 (2018-12-20).
```

```
## Installing package(s) 'phyloseq'
```

```
##
## The downloaded binary packages are in
##  /var/folders/wf/clzm6h595g1f40xhj4x9b2vm0000gn/T//Rtmpo5cGmq/downloaded_packages
```

```
## Old packages: 'dplyr', 'igraph', 'readxl'
```

```
library("phyloseq")
library("scales")
library("ggplot2")
library("Rcpp")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Import files

The main files needed are the BIOM file and the the mapping file. Only to make sure there should be 8 columns in the mapping file, not sure why is that

```
map <- import_qiime_sample_data("Mapping_all_Dianthus_112718.txt")
otu <- import_biom(BIOMfilename = "from_txt.biom",'rep_set.tre')
# Merge map file and OTU table and let it be a phyloseq object
run <- merge_phyloseq(otu,map)
# Use the official taxonomic ranks names
colnames(tax_table(run)) <- c("Kingdom", "Phylum", "Class",
                                    "Order", "Family", "Genus","Species","Rank8","Rank9",
                                    "Rank10","Rank11","Rank12","Rank13","Rank14","Rank15")
```

Abundance analysis

Filtering the data

There would be a lot of taxon whose percentage is very low, and it's not really necessary to show them all. So the first step is to keep only those taxa which each contributes to at least 2% or 5% of the whole microbial community of the sample.

```
run_genus <- run %>%
  tax_glom(taxrank = "Genus") %>%
  transform_sample_counts(function(x){x/sum(x)}) %>%
  psmelt() %>%
  filter(Abundance > 0.02) %>%
  arrange(Genus)
```

Prepare palette

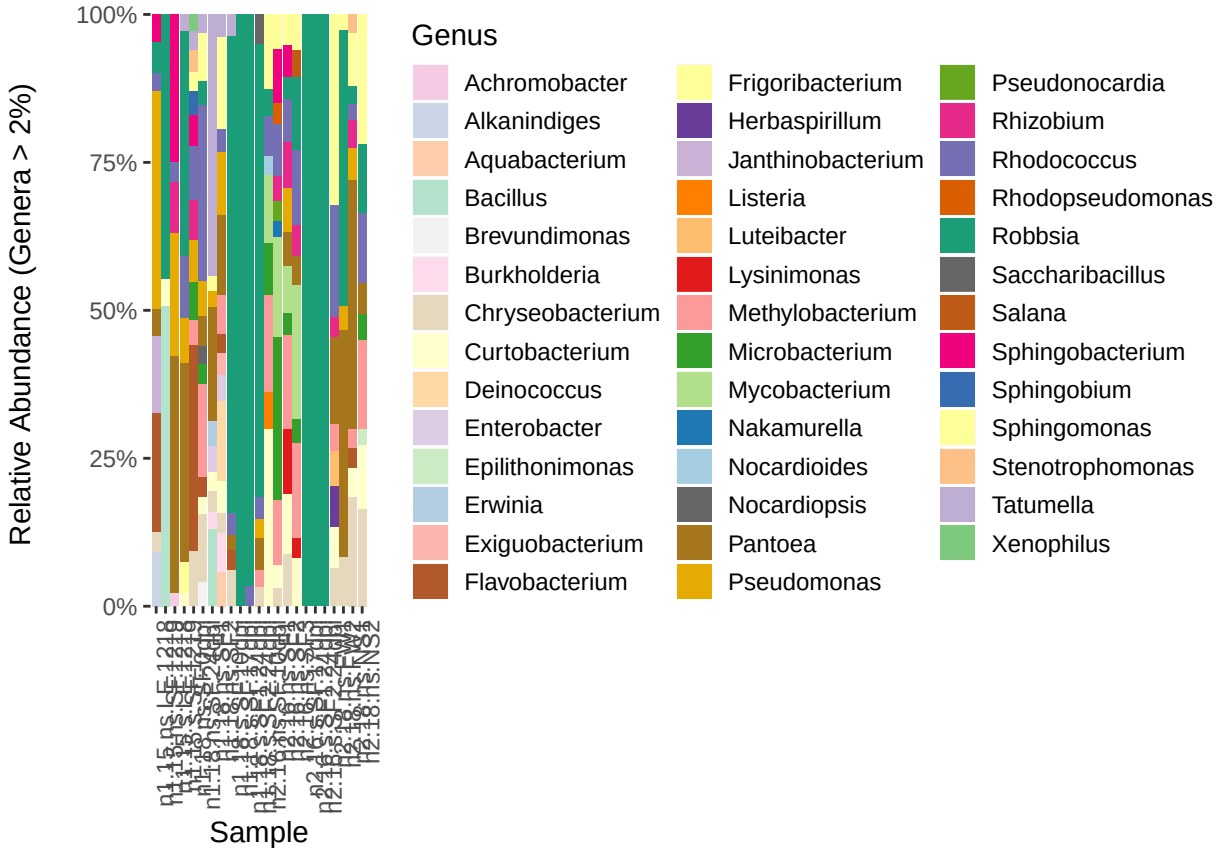
```
library(RColorBrewer)
n <- dim(run_genus)[1]
qual_col_pals = brewer.pal.info[brewer.pal.info$category == 'qual',]
col_vector = unlist(mapply(brewer.pal, qual_col_pals$maxcolors, rownames(qual_col_pals)))
```

Barplot

A normal abundance barplot could be easily generated in this way

```
run_genus$Genus <- factor(run_genus$Genus, levels = rev(levels(run_genus$Genus)))
ggplot(run_genus,aes(x=Sample,y=Abundance,fill=Genus)) +
  geom_bar(position="fill",stat="identity") +
  scale_fill_manual(values = col_vector) +
  guides(fill=guide_legend(reverse=T,keywidth = 1,keyheight = 1)) +
  ylab("Relative Abundance (Genera > 2%) \n") +
```

```
  xlab("Sample") +
  # ggtitle("Genus Composition")+
  scale_y_continuous(labels=percent_format(),expand=c(0,0))+
  theme(axis.text.x = element_text(angle=90,hjust=1), panel.background = element_blank())
```

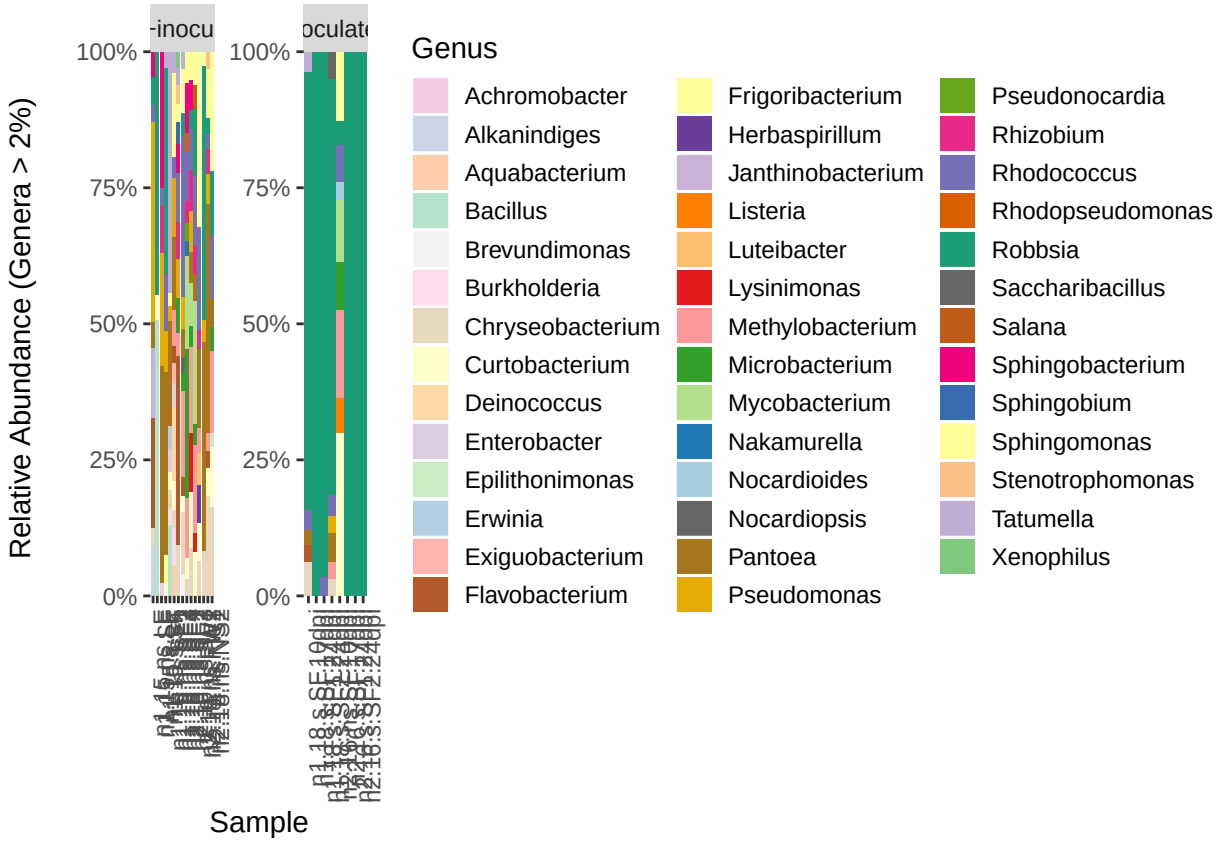


If there the experiment consists of different groups/treatments, the barplot can be divided into groups of choice

```
# run_genus$Genus <- factor(run_genus$Genus, levels = rev(levels(run_genus$Genus)))
# Add grouping info to the table
run_genus$Condition_rev <- factor(run_genus$Condition,level=rev(levels(map$Condition)))
levels(run_genus$Condition_rev) <- c('Non-inoculated','Inoculated')
library(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
# run_genus$Condition_rev <- factor(run_genus$Condition,level=rev(levels(run_genus$Condition)))
# run_genus$RealSampleID <- factor(run_genus$RealSampleID,levels = map$RealSampleID)
ggplot(run_genus,aes(x=RealSampleID,y=Abundance,fill=Genus)) +
  geom_bar(position="fill",stat="identity") +
  scale_fill_manual(values = col_vector) +
  guides(fill=guide_legend(reverse=T,keywidth = 1,keyheight = 1)) +
  ylab("Relative Abundance (Genera > 2%) \n") +
  xlab("Sample") +
  # ggtitle("Genus Composition")+
  scale_y_continuous(labels=percent_format(),expand=c(0,0))+
  theme(axis.text.x = element_text(angle=90,hjust=1), panel.background = element_blank()) +
  # scale_x_discrete(limit=map_total$RealSampleID) +
  # scale_x_discrete(labels=c(map_total[map_total$Condition=='Non-inoculated',]$RealSampleID,map_total[
  facet_wrap(~Condition_rev,scales="free")
```



Alpha diversity

Calculate rarefaction

First we create a function that can iterate the BIOM file, i.e. rarefaction

```
calculate_rarefaction_curves <- function(psdata, measures, depths) {
  require('plyr') # ldply
  require('reshape2') # melt
  estimate_rarified_richness <- function(psdata, measures, depth) {
    if(max(sample_sums(psdata)) < depth) return()
    psdata <- prune_samples(sample_sums(psdata) >= depth, psdata)
```

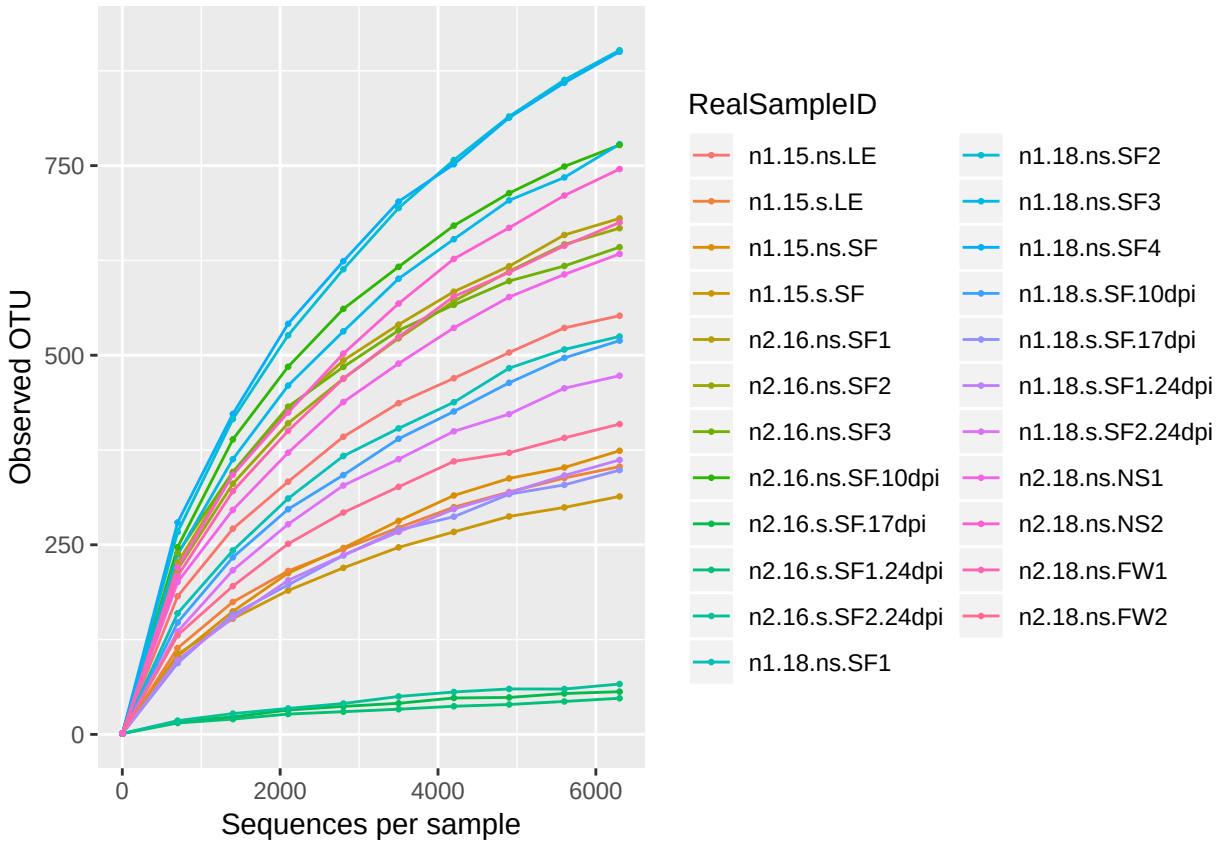
```
    rarified_psdata <- rarefy_even_depth(psdata, depth, verbose = FALSE)
    alpha_diversity <- estimate_richness(rarified_psdata, measures = measures) # as.matrix forces the u
    molten_alpha_diversity <- melt(as.matrix(alpha_diversity), varnames = c('Sample', 'Measure'), value
    molten_alpha_diversity
  }
  names(depths) <- depths # this enables automatic addition of the Depth to the output by ldply
  rarefaction_curve_data <- ldply(depths, estimate_rarified_richness, psdata = psdata, measures = measu
  rarefaction_curve_data$Depth <- as.numeric(levels(rarefaction_curve_data$Depth))[rarefaction_curve_da
  rarefaction_curve_data
}
```

One thing worth to note is the choice of depth. For the most of the time, rarefaction curve is used to compare the richness of each sample, so the depth is the minimal number of OTU of the samples. However, if you want to see the raw sample size, use the maximum.

```
depth = 7000
step = 700
rarefaction_curve_data <- calculate_rarefaction_curves(run, c('Observed'), rep(seq(1,depth,by=step), ea
```

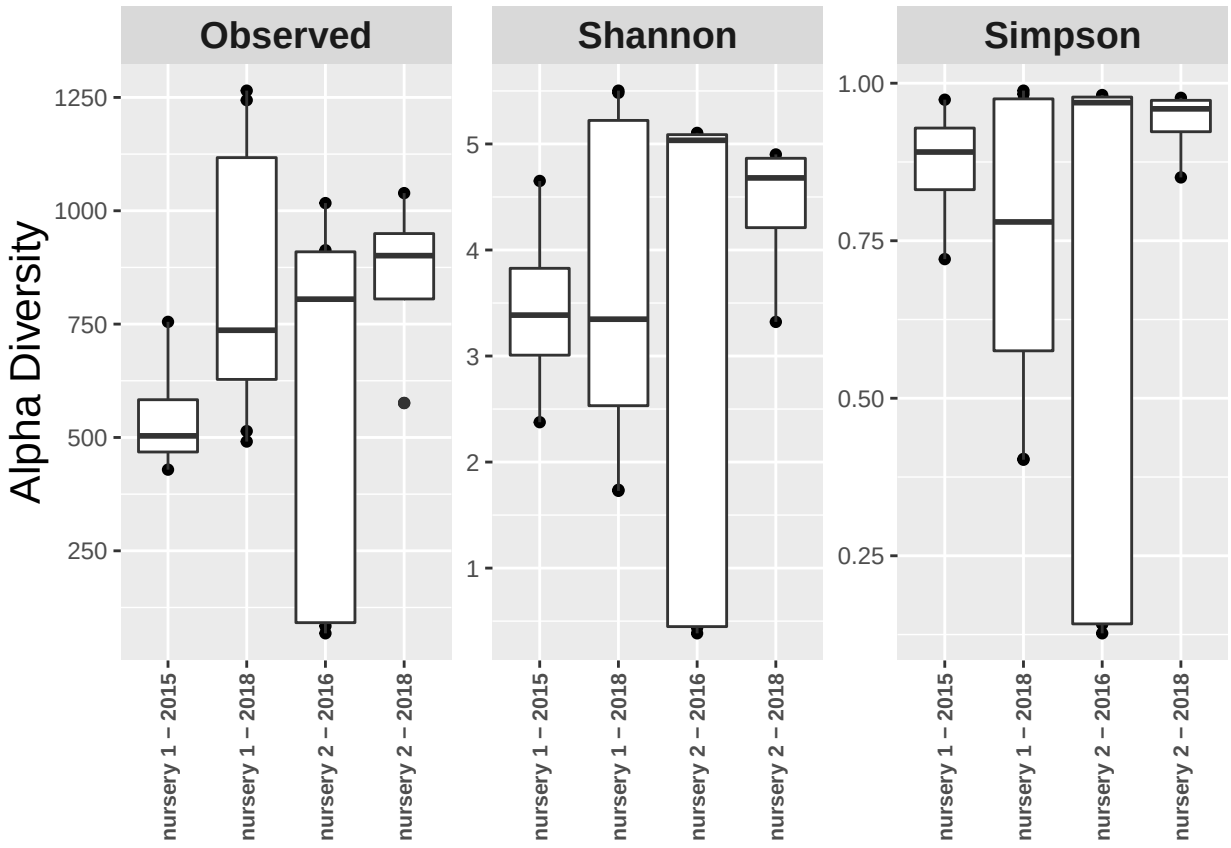
```
## Loading required package: reshape2
```

```
rarefaction_curve_data_summary <- ddply(rarefaction_curve_data, c('Depth', 'Sample', 'Measure'), summari
rarefaction_curve_data_summary_verbose <- merge(rarefaction_curve_data_summary, data.frame(sample_data(
rarefaction_curve_data_summary_verbose$RealSampleID <- factor(rarefaction_curve_data_summary_verbose$Rea
n <- dim(rarefaction_curve_data_summary_verbose)[1]
qual_col_pals = brewer.pal.info[brewer.pal.info$category == 'qual',]
col_vector = unlist(mapply(brewer.pal, qual_col_pals$maxcolors, rownames(qual_col_pals)))
ggplot( data = rarefaction_curve_data_summary_verbose,
        mapping = aes( x = Depth, y = Alpha_diversity_mean,
                       ymin = Alpha_diversity_mean - Alpha_diversity_sd,
                       ymax = Alpha_diversity_mean + Alpha_diversity_sd,
                       colour = RealSampleID,
                       group = RealSampleID)
        ) +
  scale_fill_manual(values=col_vector) +
  geom_line( ) +
  geom_point(size=0.5 ) +
  guides(fill=guide_legend(title="Sample")) +
  xlab("Sequences per sample") +
  ylab("Observed OTU")
```



Also, boxplot with other alpha diversity indices can be generated. If there is such a column in the mapping file representing the desired grouping condition, say `FigureTwo` here.

```
plot_richness(run,x="FigureTwo",measures=c("Observed","Shannon","Simpson")) +
  geom_boxplot() +
  ylab("Alpha Diversity") +
  theme(axis.text.x=element_text(angle=90,hjust=0.5,size=8,face='bold'),
        axis.title = element_text(size=16),
        axis.title.x = element_blank(),
        strip.text.x = element_text(size=14,face="bold"))
```



Beta diversity

Beta diviersity indicates the relatedness between samples.
First, pairwise distances between samples are calculated. The two measurement used are weighted-Unifrac and unweighted-Unifrac here.

```
dm_weighted_unifrac <- distance(run, method = "wUniFrac")
```

```
## Warning in UniFrac(physeq, weighted = TRUE, ...): Randomly assigning root
## as -- JF703554.1.1369 -- in the phylogenetic tree in the data you provided.
```

```
## Warning in matrix(tree$edge[order(tree$edge[, 1]), ][, 2], byrow = TRUE, :
## data length [11107] is not a sub-multiple or multiple of the number of rows
## [5554]
```

```
dm_unweighted_unifrac <- distance(run, method='Unifrac')
```

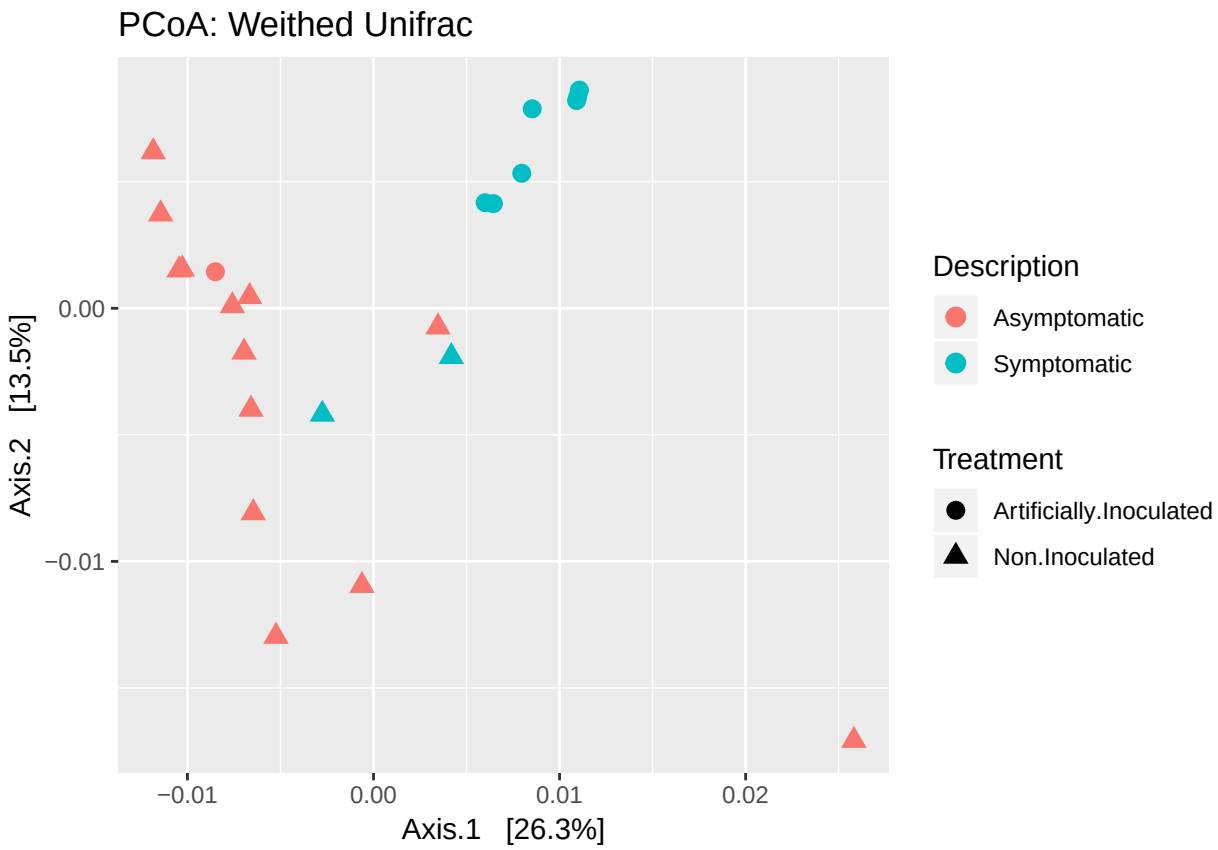
```
## Warning in UniFrac(physeq, ...): Randomly assigning root as --
## FN421553.1.1379 -- in the phylogenetic tree in the data you provided.

## Warning in UniFrac(physeq, ...): data length [11107] is not a sub-multiple
## or multiple of the number of rows [5554]
```

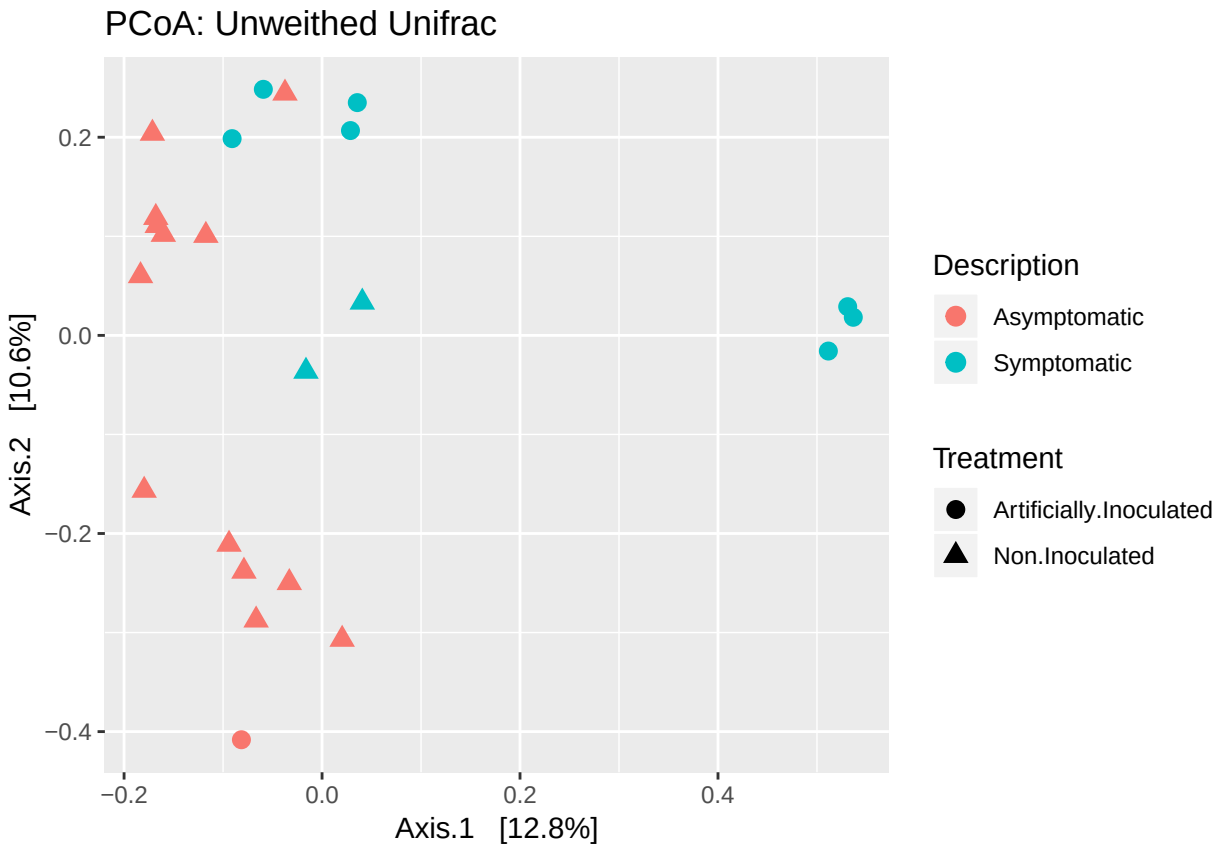
```
Treatment <- map$Treatment
Symptom <- map$Description
```

```
ordWU <- ordinate(run,method='PCoA',distance=dm_weighted_unifrac)
plot_ordination(run, ordWU,,color = 'Description',shape='Treatment') +
```

```
  geom_point(size=3) +
  # geom_text(aes(label=map$RealSampleID),hjust=0,vjust=0) +
  # theme(axis.title = element_text(size=20),
  #       axis.text = element_text(size=16),
  #       legend.text = element_text(size=16)) +
  ggtitle("PCoA: Weithed Unifrac")
```




```
ordUU <- ordinate(run,method='PCoA',distance=dm_unweighted_unifrac)
plot_ordination(run, ordUU,,color = 'Description',shape='Treatment') +
  geom_point(size=3) +
  # geom_text(aes(label=map$RealSampleID),hjust=0,vjust=0) +
  # theme(axis.title = element_text(size=20),
  #       axis.text = element_text(size=16),
  #       legend.text = element_text(size=16)) +
  ggtitle("PCoA: Unweithed Unifrac")
```



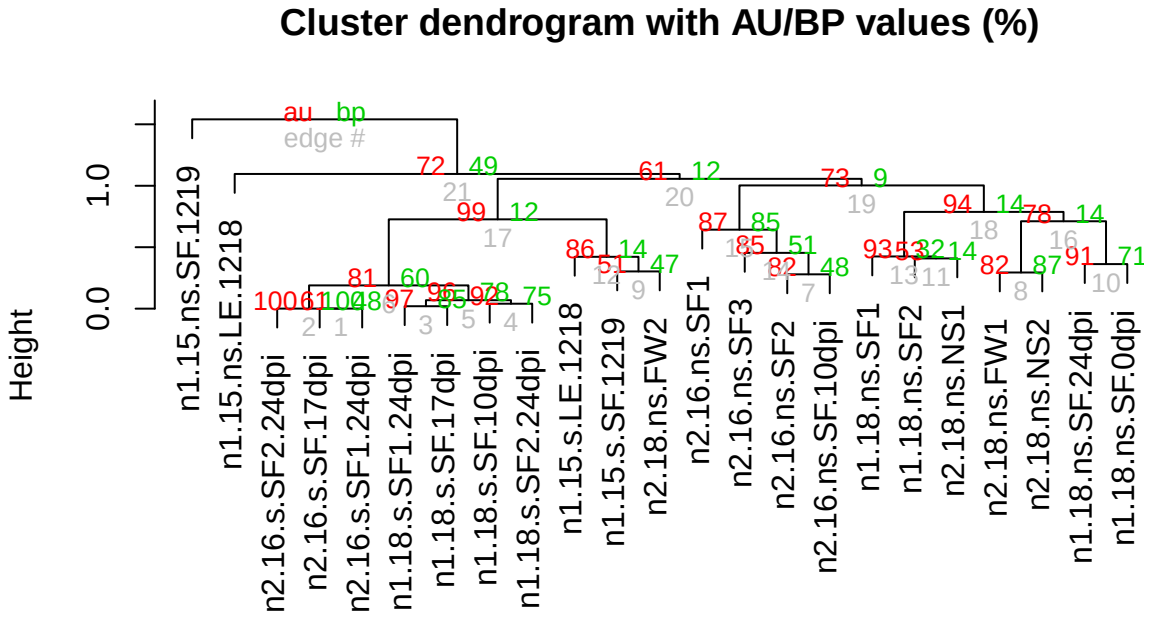
Phylogenetics tree of the samples

Bootstrap values can be generated from `pvclust`, and the topology is the same with what calculated from regular `hclust` function

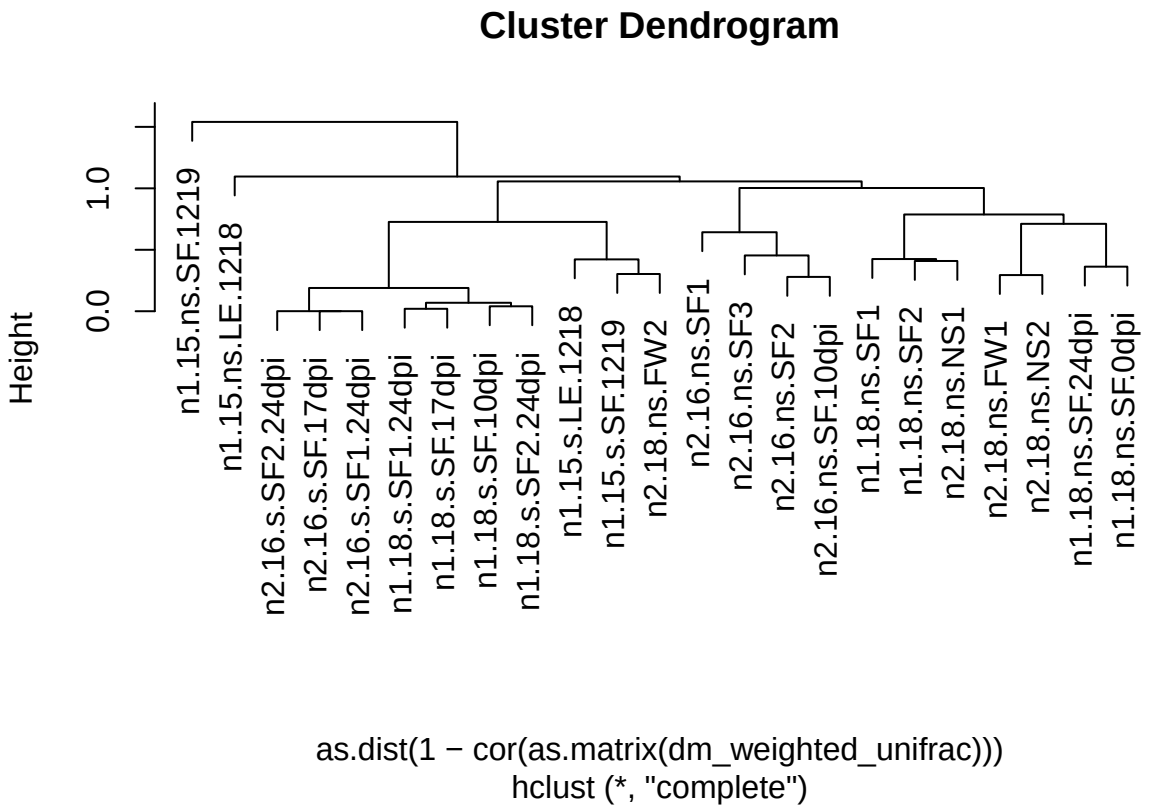
```
library(pvclust)
hc_pv <- pvclust(as.matrix(dm_weighted_unifrac),method.hclust = "complete")
```

```
## Bootstrap (r = 0.48)... Done.
## Bootstrap (r = 0.57)... Done.
## Bootstrap (r = 0.7)... Done.
## Bootstrap (r = 0.78)... Done.
## Bootstrap (r = 0.87)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.09)... Done.
## Bootstrap (r = 1.17)... Done.
## Bootstrap (r = 1.26)... Done.
## Bootstrap (r = 1.39)... Done.
```

```
plot(hc_pv)
```



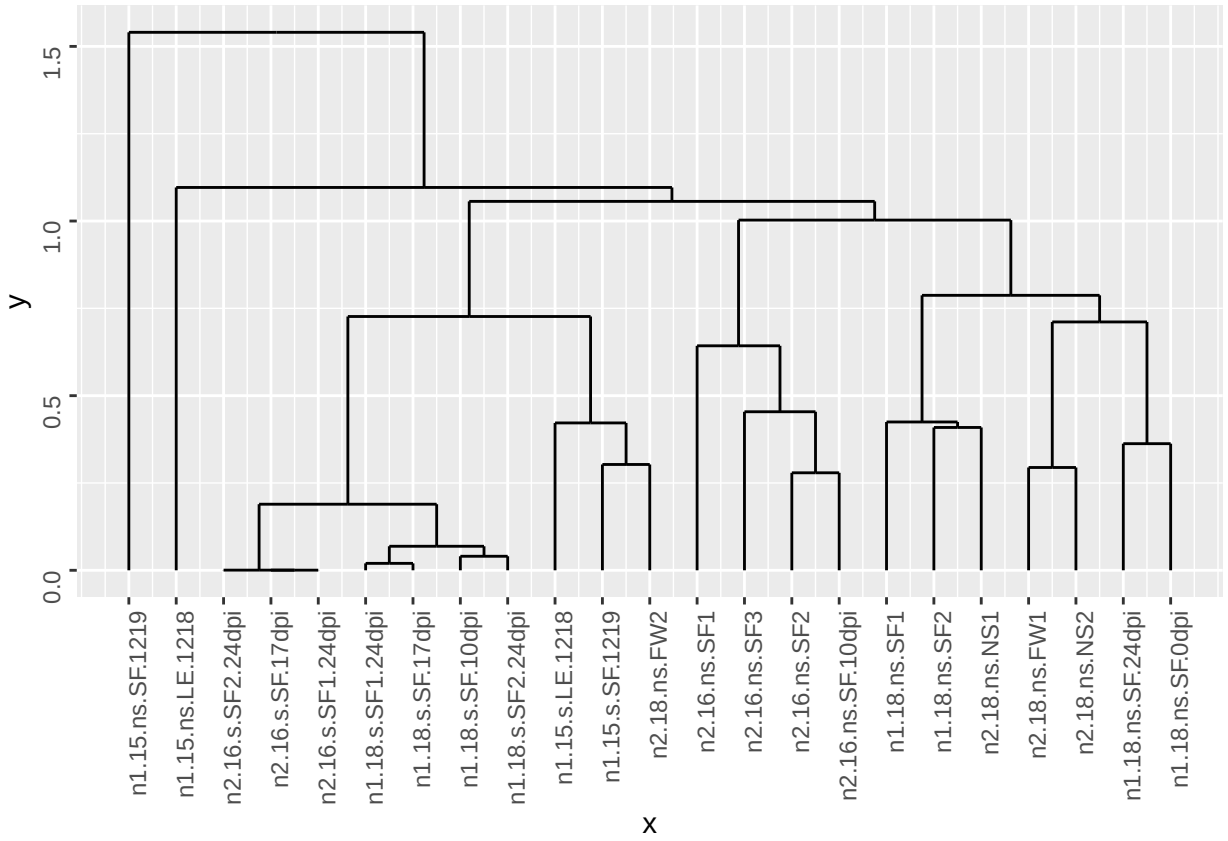

```
hc <- hclust(as.dist(1-cor(as.matrix(dm_weighted_unifrac))),method="complete")
plot(hc)
```



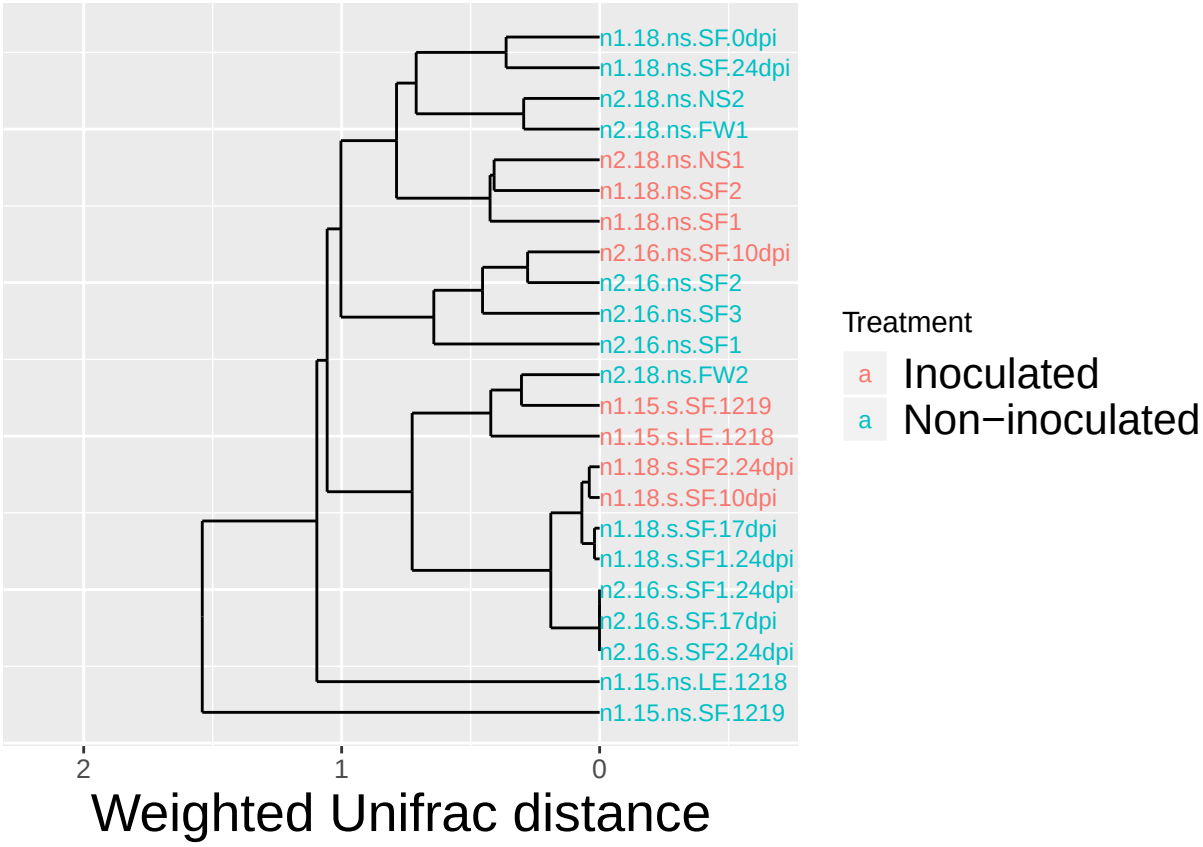
The den-

drogram can be plotted through `ggplot2` with the help from `ggdendro`.

```
library(ggdendro)
ggdendrogram(hc,rotate=F, size=4, theme_dendro = F)
```




```
ddata <- dendro_data(as.dendrogram(hc))
mapping <- map
label <- as.character(ddata$labels$label)
rownames(mapping) <- mapping$RealSampleID
Treatment <- mapping$Condition
labeled_data <- cbind(label(ddata),Treatment)
ggplot(segment(ddata)) +
  geom_segment(aes(x=x,y=y,xend=xend,yend=yend)) +
  coord_flip() + scale_y_reverse(expand=c(0.5,0)) +
  geom_text(data = labeled_data, aes(x=x,y=y,label=label,hjust=0,color=Treatment),size=3) +
  ylab("Weighted Unifrac distance") +
  theme(axis.line.y=element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        axis.title.y=element_blank(),
        plot.title = element_text(hjust=0.5),
        legend.text = element_text(size=16),
        axis.text.x = element_text(size=10),
        axis.title.x = element_text(size=20))
```


n1.18.ns.SF.0dpi
n1.18.ns.SF.24dpi
n2.18.ns.NS2
n2.18.ns.FW1
n2.18.ns.NS1
n1.18.ns.SF2
n1.18.ns.SF1
n2.16.ns.SF.10dpi
n2.16.ns.SF2
n2.16.ns.SF3
n2.16.ns.SF1
n2.18.ns.FW2
n1.15.s.SF.1219
n1.15.s.LE.1218
n1.18.s.SF2.24dpi
n1.18.s.SF.10dpi
n1.18.s.SF.17dpi
n1.18.s.SF1.24dpi
n2.16.s.SF1.24dpi
n2.16.s.SF.17dpi
n2.16.s.SF2.24dpi
n1.15.ns.LE.1218
n1.15.ns.SF.1219
2
1
0
Weighted Unifrac distance
Treatment
a Inoculated
a Non−inoculated