# Sulawesi Project Microbiome Analysis

**Author: Damilola R Oresegun** This is a developing script to carry out microbiome analysis for the Sulawesi Macaque fecal metagenome project. The faecal sample were extracted from healthy wild macaques and sequenced using the Oxford Nnaopore MinION. After this the reads were put through a pipeline developed by the author to carry out assembly of the metagenomic sequences using metaFlye.

### Pre-processing

Here the data will be read in, adjusted for easier downstream manipulation and prepared for full analysis. The data is read in using mia specifically to generate a phylogenetic tree using the addTaxonomy function. After this, the data object is converted into a phyloseq object and read into phyloseq to begin analysis.

```r
# install and set libraries
# if (!requireNamespace("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
# BiocManager::install(c("microbiome/mia","dendextend", "rms", "devtools","picante","tidyr",
#     "remotes","devtools","tidyverse","dplyr","cowplot","scater","knitr","phyloseq","HMP",
#     "vegan","ape"))

library(mia)
library(dplyr)
library(knitr)
library(phyloseq)
library(vegan)
library(tidyr)
library(magrittr)
library(ape)
library(tidyverse)
library(data.table)
library(formatR)
```

```
## Warning: package 'formatR' was built under R version 4.2.1
```

```r
# download the test biom file url <-
# 'https://github.com/damioresegun/Sulawesi_microbiome/blob/dev-branch/MicrobiomeAnalysis/MFMRCFS0822_C
# inpBiom <- 'C:/Users/dro/Desktop/BiomFile.biom' # choose
# where you want to put it download.file(url, inpBiom)
args = c("C:/Users/dro/Dropbox/Work/MacLaptop/JCS_Project/Pipeline/Sulawesi_microbiome/MicrobiomeAnalysi
    "MFMRCFS0822_Combined.biom")
workD = args[1]  # the working directory
inpBiom = args[2]  # the input biom file
setwd(workD)
# import the biom data
JCSData <- loadFromBiom(inpBiom)
# quickview of the data
head(rowData(JCSData))
```

### Load data

```
## DataFrame with 6 rows and 7 columns
##           taxonomy1     taxonomy2     taxonomy3        taxonomy4
##         <character>   <character>   <character>      <character>
## 853     k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
## 2714355 k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
## 2714353 k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
## 2830675 k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
## 2093857 k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
## 292800  k__Bacteria p__Firmicutes c__Clostridia o__Eubacteriales
##                 taxonomy5           taxonomy6              taxonomy7
##               <character>         <character>            <character>
## 853     f__Oscillospiraceae g__Faecalibacterium         s__prausnitzii
## 2714355 f__Oscillospiraceae       g__Vescimonas           s__coprocola
## 2714353 f__Oscillospiraceae       g__Vescimonas          s__fastidiosa
## 2830675 f__Oscillospiraceae    g__Dysosmobacter s__sp. Marseille-Q4140
## 2093857 f__Oscillospiraceae    g__Dysosmobacter           s__welbionis
## 292800  f__Oscillospiraceae   g__Flavonifractor             s__plautii
```

The command `head(rowData(JCSData))` allows a quick view of the information stored in the biom file. Here, the taxonomic information is being shown, with the taxid in the first column followed by the taxonomic levels going from Kingdom ($k$) to Species ($s$). However, this does not give us enough information and requires some manipulation for ease of reading.

```
# Change the column data and remove the k___
names(rowData(JCSData)) <- c("Kingdom", "Phylum", "Class", "Order",
    "Family", "Genus", "Species")
rowData_mod <- BiocParallel::bplapply(rowData(JCSData), FUN = stringr::str_remove,
    pattern = ".*[kpcofgs]__")
# remove any \ in the data
rowData_mod <- BiocParallel::bplapply(rowData_mod, FUN = stringr::str_remove,
    pattern = "\"")
# make the list into a dataframe
rowData_mod <- DataFrame(rowData_mod)
# see the output
head(rowData_mod)
```

```
## DataFrame with 6 rows and 7 columns
##       Kingdom      Phylum       Class         Order           Family
##   <character> <character> <character>   <character>      <character>
## 1    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 2    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 3    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 4    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 5    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 6    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
##              Genus             Species
##        <character>         <character>
## 1 Faecalibacterium         prausnitzii
## 2       Vescimonas           coprocola
## 3       Vescimonas          fastidiosa
## 4    Dysosmobacter sp. Marseille-Q4140
## 5    Dysosmobacter           welbionis
## 6   Flavonifractor             plautii
```

Now the data is easier to read, however the manipulated taxonomic data needs to be placed back into the main data object holding the rest of the information. **Note: The main data object with the full input information is JCSData.**

```r
# add back into the entire data object
rowData(JCSData) <- rowData_mod
head(rowData(JCSData))
```

```
## DataFrame with 6 rows and 7 columns
##             Kingdom      Phylum       Class         Order           Family
##         <character> <character> <character>   <character>      <character>
## 853        Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 2714355    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 2714353    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 2830675    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 2093857    Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
## 292800     Bacteria  Firmicutes  Clostridia Eubacteriales Oscillospiraceae
##                     Genus             Species
##               <character>         <character>
## 853     Faecalibacterium         prausnitzii
## 2714355       Vescimonas           coprocola
## 2714353       Vescimonas          fastidiosa
## 2830675    Dysosmobacter sp. Marseille-Q4140
## 2093857    Dysosmobacter           welbionis
## 292800    Flavonifractor             plautii
```

The biom file that was inputted into the script (*now saved in the data object JCSData*) only contains the taxid and taxonomic classification of all the samples in the experiment. However, it does not contain other sample metadata which we will need for different downstream statistical processing. Ideally, this information would be already be made in a csv file to just import. However, we will simply state them here.

```r
# add sample data metadata (This changes depending on
# project)
sample_meta = DataFrame(read.table("sampleData.csv", sep = ",",
    header = TRUE))
# add the sample names as the row names
rownames(sample_meta) <- sample_meta[, 3]
# add to the object
colData(JCSData) <- sample_meta
```

Another thing that will be needed downstream is a phylogenetic tree. However, this was not previously generated for this dataset. Unfortunately, it is currently not possible to generate a phylogenetic tree for this dataset. There are different reasons for this; ranging from the computational resources needed as well as the practical means of generating this data. Fortunately the mia package is able to generate a taxonomic tree from the input taxonomic data. Importantly, this is not an ideal way to generate the tree as it does not contain a true outgroup to act as a root for the tree. However it can aid in some visualisation downstream

```r
# add/make a phylogenetic tree to the data object
JCSData <- addTaxonomyTree(JCSData)  # this produces a warning message: In toTree(td) : The root is add
# Convert this data object to phyloseq datatype for use in
# phyloseq
JCSData <- makePhyloseqFromTreeSummarizedExperiment(JCSData)
```

```r
# define a function to select an outgroup for rooting phylo
# tree
pick_new_outgroup <- function(tree.unrooted) {
    # tablify parts of tree that is needed
    treeDT <- cbind(data.table(tree.unrooted$edge), data.table(length = tree.unrooted$edge.length))[1:N
        cbind(data.table(id = tree.unrooted$tip.label))
    # Take the longest terminal branch as outgroup
    new.outgroup <- treeDT[which.max(length)]$id
    return(new.outgroup)
}
# root the phylo tree
OutGroup <- pick_new_outgroup(phy_tree(JCSData))
# show the chosen outgroup
OutGroup
```

```
## [1] "Species:venezuelae"
```

```r
# root the tree
phy_tree(JCSData) <- ape::root(phy_tree(JCSData), outgroup = OutGroup,
    resolve.root = TRUE)
# check that it is rooted
phy_tree(JCSData)
```

```
##
## Phylogenetic tree with 177 tips and 131 internal nodes.
##
## Tip labels:
##   Species:venezuelae, Species:timonensis_2, Species:uli, Species:umbonata, Species:catena, Species:ma
## Node labels:
##   Root, Kingdom:Bacteria, Phylum:Actinobacteria, Class:Actinomycetia, Order:Streptomycetales, Family
##
## Rooted; includes branch lengths.
```

After making the taxonomic tree and adding it to the `JCSData` data object, the data type is converted to a phyloseq data object. This is because phyloseq is a much better supported, documented and widely used microbiome analysis package. As such, moving forward, the data manipulation will mainly occur via the phyloseq package.

In order to root the tree in `JCSData`, an outgroup will be chosen based on the taxid with the longest branch. In this case, the outgroup is the `Species:venezulae`.

Moving forward, empty characters and `N.A` characters are to be removed

```r
JCSData <- subset_taxa(JCSData, !is.na(Kingdom) & !Kingdom %in%
    c("", "uncharacterized"))
JCSData <- subset_taxa(JCSData, !is.na(Phylum) & !Phylum %in%
    c("", "uncharacterized"))
JCSData <- subset_taxa(JCSData, !is.na(Class) & !Class %in% c("",
    "uncharacterized"))
JCSData <- subset_taxa(JCSData, !is.na(Family) & !Family %in%
    c("", "uncharacterized"))
JCSData <- subset_taxa(JCSData, !is.na(Order) & !Order %in% c("",
    "uncharacterized"))
```

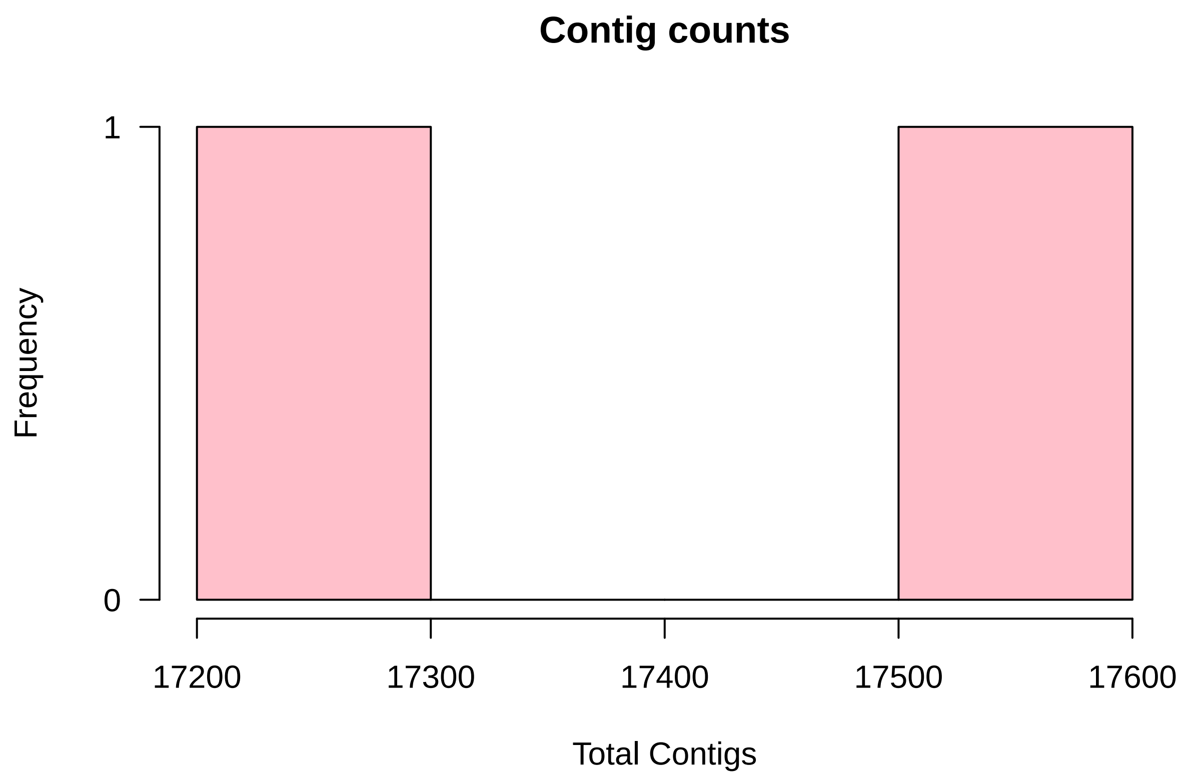
```
JCSData <- subset_taxa(JCSData, !is.na(Species) & !Species %in%
    c("", "uncharacterized"))
```

**Simple metrics** Here, simple metrics will be derived. The number of samples, taxa, number of contigs for each sample/sample type will be calculated and plotted where appropriate.

```
# get the number of contigs for each samples
sample_sums(JCSData)
```

```
##    Am_M_001_FS_DNA Am_M_001_FS_dscDNA
##              17505              17272
```

```
# plot histogram of contig counts
hist(sample_sums(JCSData), main = "Contig counts", xlab = "Total Contigs",
    col = "pink", las = 1)
```



```
# add the number of contigs back to the sample data
sample_data(JCSData)$ClassifiedContigs <- sample_sums(JCSData)
# get the number of taxa
ntaxa(JCSData)
```

```
## [1] 177
```

```r
# get the number of total contigs of each taxa over all
# samples
head(taxa_sums(JCSData))
```

```
##     Species:venezuelae   Species:timonensis_2            Species:uli
##                    278                    173                     27
##       Species:umbonata         Species:catena Species:massiliensis_3
##                     37                    129                    244
```

```r
# get the number of contigs of each taxa per sample type
data.frame(otu_table(JCSData)[1:10])  # shows the top 10
```

```
##                         Am_M_001_FS_DNA Am_M_001_FS_dscDNA
## Species:venezuelae                  278                  0
## Species:timonensis_2                 87                 86
## Species:uli                          14                 13
## Species:umbonata                     17                 20
## Species:catena                       63                 66
## Species:massiliensis_3              123                121
## Species:immobilis                    48                 53
## Species:aerofaciens                 131                143
## Species:equolifaciens                22                 24
## Species:massiliensis_1               10                  0
```

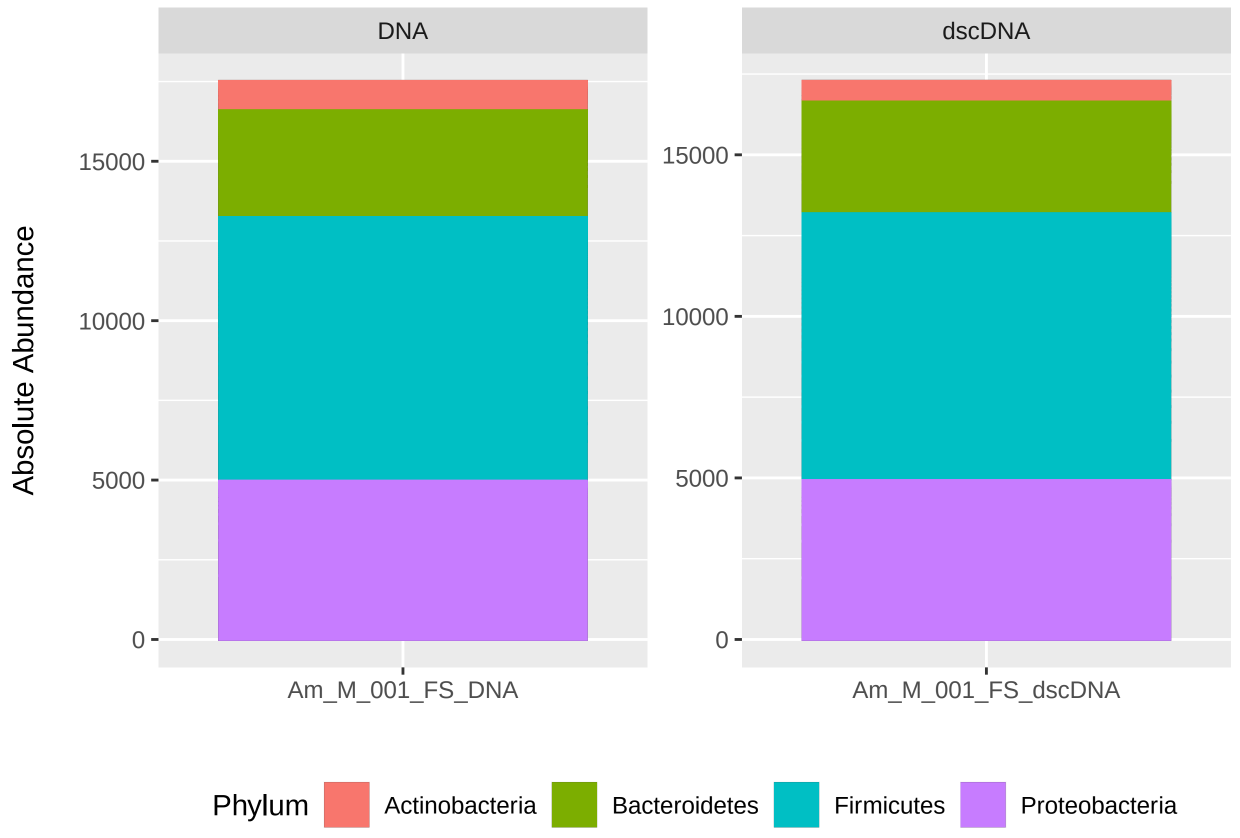
```r
# save the otu table
DaOtuTab <- data.frame(otu_table(JCSData))
# save to file
write.csv(DaOtuTab, "Contigs_per_taxa.csv", row.names = TRUE)
# create a table for the number of identified features
table(tax_table(JCSData)[, "Kingdom"])  # number of features per kingdom
```

```
##
## Bacteria
##      177
```

```r
table(tax_table(JCSData)[, "Phylum"])  # number of features per phylum
```

```
##
## Actinobacteria  Bacteroidetes     Firmicutes Proteobacteria
##             17             56             83             21
```

```r
# make this into a dataframe save total number of features
# per phylum to file
write.csv(table(data.frame(tax_table(JCSData)[, 2])), "PhylumFeatureCount.csv",
    row.names = FALSE)
# plot the dataset using phyla to colour the stacked bar
# chart
(p1 <- plot_bar(JCSData, fill = "Phylum") + geom_bar(aes(color = Phylum,
    fill = Phylum), stat = "identity", position = "stack") +
    labs(x = "", y = "Absolute Abundance\n") + facet_wrap(~Sample_Type,
    scales = "free", nrow = 1) + theme(legend.position = "bottom") +
    theme(axis.text.x = element_text(angle = 0, hjust = 0.5)))
```



```
# save the plot to file
tiff("Taxonomy_Abundance.tif", width = 5000, height = 5000, units = "px",
    res = 300)
p1
dev.off()
```

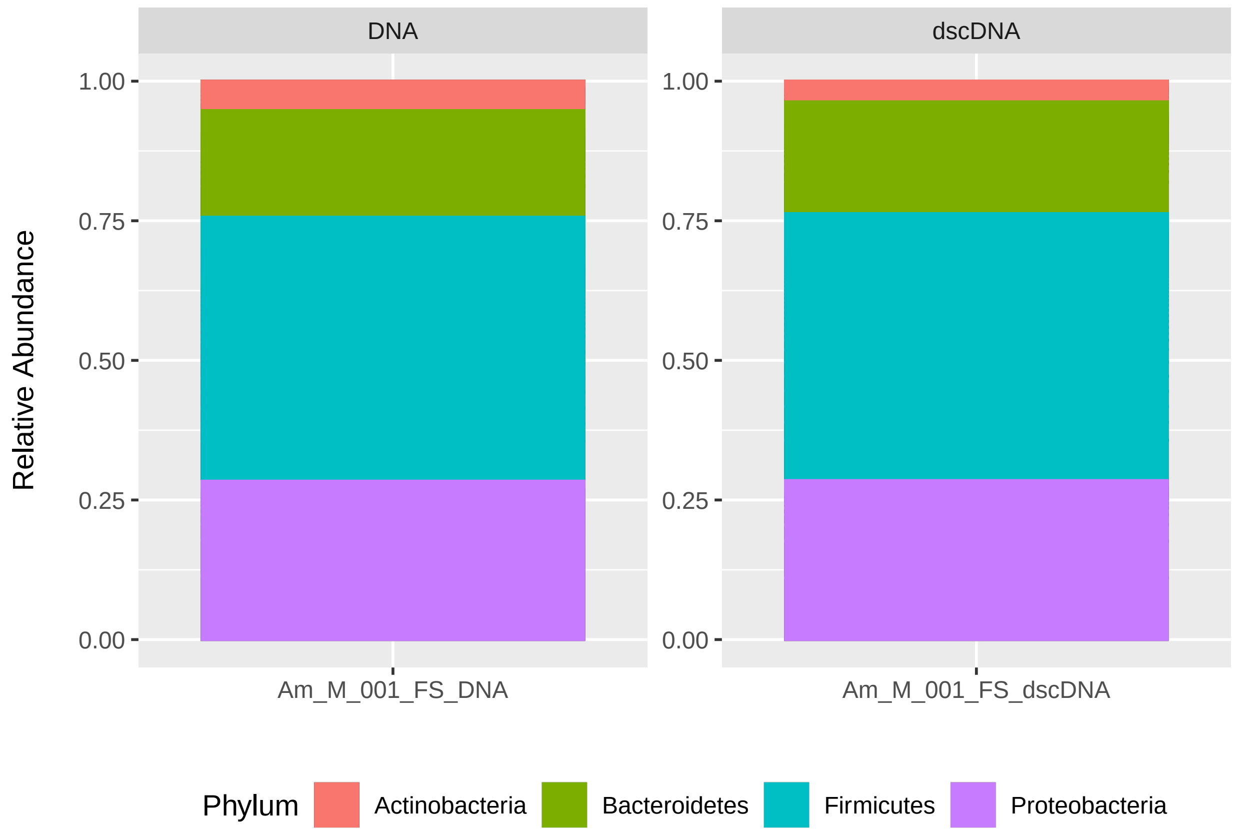
```
## pdf
##   2
```

The large code above does a few things. Using simple commands like `sample_sums`, `ntaxa` and `taxa_sums` give simple metric outputs of the input information. In the above, the DNA sequences report `17505` contigs that has been classified while cDNA sequences have `17272` contigs. This is different from the number of contigs that was initially reported in the sample_data CSV file. This is because the CSV file shows the number of contigs that was assembled, while `sample_sums` gives the number of contigs that successfully got classified. On the other hand, `ntaxa` gives the number of individual taxonomies identified across both the DNA and cDNA sample types. Essentially, this means that 177 species were found due to the Kraken and Bracken classifications. The `otu_table` takes this further and shows the number of contigs present for each sample type for each taxonomy. This is saved to file for later investigation.

Tables are then created to get the number of taxonomies (*or features*) for each Kingdom and Phylum. For this dataset, only the Bacteria Kingdom was successfully classified and thus only Bacterial phyla are reported. This classification is likely due to the nature of the data i.e. being assembled. Importantly, the classification was carried out using a Kraken threshold of 3 (*while the reads approach was done with a threshold of 5*) and a bracken threshold of 10 (*while reads approach was done with a threshold of 20*). As such, it is likely that sequences from other Kingdoms were unable to either be assembled altogether (*by metaFlye*) or classified (*by Kraken+bracken*) – in both cases, due to abundance; resulting in their absence in this analysis. A re-think the approach of the pipeline directly after assembly is recommended. As abundance might be a limiting factor, investigating the abundance of the present Phyla can provide some insights

**Relative Abundance** The relative abundance is calculated using the number of contigs for each taxid divided by the total number of contigs. This gives a relative abundance that is normalised across the taxids and samples in the dataset.

```
# calculate and add the relative abundance data to the
# object
JCSData_Abund <- transform_sample_counts(JCSData, function(x) {
    x/sum(x)
})
# plot the dataset using phyla to colour the stacked bar
# chart
(p2 <- plot_bar(JCSData_Abund, fill = "Phylum") + geom_bar(aes(color = Phylum,
    fill = Phylum), stat = "identity", position = "stack") +
    labs(x = "", y = "Relative Abundance\n") + facet_wrap(~Sample_Type,
    scales = "free") + theme(legend.position = "bottom") + theme(axis.text.x = element_text(angle = 0,
    hjust = 0.5)))
```



```
# save the plot to file
tiff("Taxonomy_RelativeAbundance.tif", width = 5000, height = 5000,
    units = "px", res = 300)
p2
dev.off()
```

```
## pdf
##   2
```

For this example, this does not provide much information due to having the same host sample (*but different sequence types*). However, more information will be available with more samples combined into a single data object. As such, at this point, it would be advised to re-trace the pipeline in order to recover more sequences from other Kingdoms and diversity

### Alpha diversity

This is a metric to determine the diversity within a sample. There are multiple different indices within this however the most commonly quoted are the Shannon and Simpson diversity indices.

```
rareBac <- rarefy_even_depth(JCSData, rngseed = 12355, replace = FALSE)
```

```
## ‘set.seed(12355)‘ was used to initialize repeatable random subsampling.

## Please record this for your records so others can reproduce.

## Try ‘set.seed(12355); .Random.seed‘ for the full vector

## ...
```

```
(alpha_div <- estimate_richness(rareBac))
```

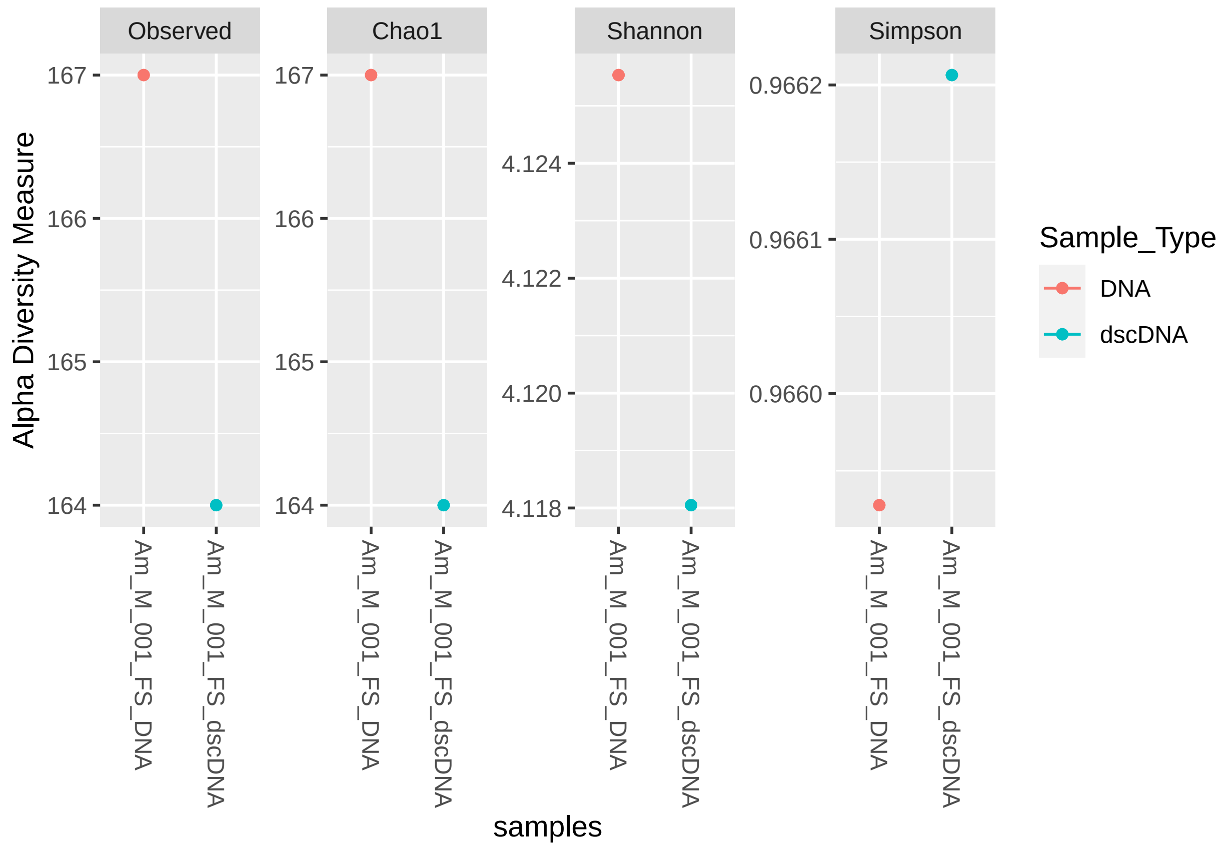
```
## Warning in estimate_richness(rareBac): The data you have provided does not have
## any singletons. This is highly suspicious. Results of richness
## estimates (for example) are probably unreliable, or wrong, if you have already
## trimmed low-abundance taxa from the data.
##
## We recommended that you find the un-trimmed data and retry.

##                      Observed Chao1 se.chao1 ACE   se.ACE  Shannon   Simpson
## Am_M_001_FS_DNA           167   167        0 167 1.975903 4.125535 0.9659277
## Am_M_001_FS_dscDNA        164   164        0 164 1.405564 4.118049 0.9662064
##                      InvSimpson   Fisher
## Am_M_001_FS_DNA        29.34938 25.63576
## Am_M_001_FS_dscDNA     29.59140 25.09292
```

Here, the diversity indices calculated using the Shannon and Simpson indices provide very similar outputs. For the Shannon diversity, the higher the value, the more diverse the sample composition is. The Shannon diversity is influenced by species richness and rare species. On the other hand, Simpson diversity index is between 0 and 1 where 0 represents infinite diversity and 1 representing no diversity. It gives more weight to evenness and common species.

```
# save to file
write.csv(alpha_div, "AlphaDiversity.csv", row.names = TRUE)
# Plot the diversity indices
(p4 <- plot_richness(rareBac, measures = c("Observed", "Shannon",
    "Chao1", "Simpson"), color = "Sample_Type") + geom_point(aes(color = Sample_Type)) +
    labs(color = "Sample_Type"))
```

```
## Warning in estimate_richness(physeq, split = TRUE, measures = measures): The data you have provided d
## any singletons. This is highly suspicious. Results of richness
## estimates (for example) are probably unreliable, or wrong, if you have already
## trimmed low-abundance taxa from the data.
##
## We recommended that you find the un-trimmed data and retry.
```


Observed
Chao1
Shannon
Simpson
Alpha Diversity Measure
Am_M_001_FS_DNA
Am_M_001_FS_dscDNA
samples
Sample_Type
DNA
dscDNA