# Indonesian Samples Vs. Controls

katalinabobowik

2020-09-13

Last updated: 2020-10-13

## Introduction

This part of the study will analyse differences between our Indonesian samples and compare them with healthy controls. From my previous analysis of healthy Indonesians, I found that Plasmodium, Flavivirus, and Bacteria make up the majority of the taxa found within unmapped reads. However, I want to see how different this is to populations in more sterile environments.

The aim of this analysis is therefore to test whether the pathogen signature identified in our Indonesian samples is unique. We will test this by looking at the sample grouping by PCA, hierarchical clustering, relative abundance of taxa, alpha and beta diversity estimates, and differential abundance testing.

The Indonesian data in this study were generated from the previously-published study by Natri et al. Briefly, 101-bp, paired-end data were generated on an Illumina HiSeq 2500 to an average depth of 30 million read pairs per individualon.

it was run through KMA, CCMetage, etc… (look at scripts, x, y, z, for sample processing)

The control samples were taken from Loohuis et al.

These two other studies are…

## Loading packages and colour setup

The code below will install the packages needed to run the analyses. We're also setting up our directories to run locally, but I've provided all of the paths so they can be easily modified if they need to be run on the server.

```r
require(ggplot2)
require(RColorBrewer)
library(dplyr)
library(plyr)
library(reshape2)
library(ggpubr)
library(metacoder)
library(tidyverse)
library(phyloseq)
library(DESeq2)
library(microbiome)
library(vegan)
library(picante)
library(ALDEx2)
library(metagenomeSeq)
library(HMP)
library(dendextend)
library(selbal)
library(rms)
library(breakaway)
library(microbiomeutilities)
library(mixOmics)
library(SRS)
library(ggrepel)

# set up directories
refdir = "/Users/katalinabobowik/Documents/UniMelb_PhD/Analysis/UniMelb_Sumba/ReferenceFiles/EpiStudy/"
filteringDir = "/Users/katalinabobowik/Documents/UniMelb_PhD/Analysis/UniMelb_Sumba/Output/Epi_Study/Co
```

We'll also set up our colour schemes.

```
# set ggplot colour theme to white
theme_set(theme_bw())

# Set up colour scheme
IndonesiaCol="#4477AA"
MaliCol="#228833"
UKCol="#AA3377"
```

## Reading in the Indonesian data

Many microbiome studies use the package phyloseq to analyse data due to its comprehensive packages. The data structures in Phyloseq (taxa data, otu data, and sample data) are also contained in a single object, which makes it easy to keep everything together.

First, we'll read in our Indonesian single-ended count data and separate taxa information from read abundance information. We'll then assign sample information to the data (namely, population and sample IDs) so that we can compare it to the control datasets.

```
AllREadsSE_Indo_Counts <- read.csv(paste0(refdir,"Counts_Indo_75BP_SE.csv"),check.names=FALSE)

# Separate species' abundances and taxonomy columns
taxa_raw <- as.matrix(AllREadsSE_Indo_Counts[,c("Superkingdom","Kingdom","Phylum", "Class", "Order","Fa
abund_raw <- as.matrix(AllREadsSE_Indo_Counts[,-which(colnames(AllREadsSE_Indo_Counts) %in% c("Superkin

# convert to Phyloseq object
tax = tax_table(taxa_raw)
taxa = otu_table(abund_raw, taxa_are_rows = TRUE)
AllREadsSE_Indo_Counts_physeq = phyloseq(taxa, tax)

# add in sample information, starting with Island
samplenames <- colnames(otu_table(AllREadsSE_Indo_Counts_physeq))
pop <- rep("Indonesia",ncol(otu_table(AllREadsSE_Indo_Counts_physeq)))

# make this into a df and add to the Phloseq object
samples_df=data.frame(SampleName=colnames(otu_table(AllREadsSE_Indo_Counts_physeq)), SamplePop=pop)
```

```
samples = sample_data(samples_df)
rownames(samples)=samples$SampleName
sample_data(AllREadsSE_Indo_Counts_physeq) <- samples

# get information on phyloseq objects
AllREadsSE_Indo_Counts_physeq
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 2458 taxa and 123 samples ]
## sample_data() Sample Data:       [ 123 samples by 2 sample variables ]
## tax_table()   Taxonomy Table:    [ 2458 taxa by 8 taxonomic ranks ]
```

For the Indonesian samples, we can see that we have a phyloseq object consisting of 123 samples, 6 of which are replicates. We'll take the replicates with the highest library depth and then remove the rest.

```
# get replicates
trimmed_samplenames = gsub("Batch1",'',samplenames) %>% gsub("Batch2",'', .) %>% gsub("Batch3",'', .)
trimmed_samplenames = sub("([A-Z]{3})([0-9]{3})", "\\1-\\2", trimmed_samplenames)
replicate_index = which(duplicated(trimmed_samplenames) | duplicated(trimmed_samplenames, fromLast = TRU
replicates = samplenames[replicate_index]

# add sequencing depth information to the Physeq object in order to filter replicates by seqDepth
SeqDepth = colSums(otu_table(AllREadsSE_Indo_Counts_physeq))
sample_data(AllREadsSE_Indo_Counts_physeq)$SeqDepth = SeqDepth

# find out which replicates have the highest sequencing depth
sample_data(AllREadsSE_Indo_Counts_physeq)[replicates,]
```

```
##                           SampleName SamplePop SeqDepth
## MPI-381Batch1         MPI-381Batch1 Indonesia    31097
## MPI-381Batch3         MPI-381Batch3 Indonesia    35468
## MTW-TLL-013Batch2 MTW-TLL-013Batch2 Indonesia    42231
## MTW-TLL013Batch3   MTW-TLL013Batch3 Indonesia    37580
## SMB-ANK-016Batch2 SMB-ANK-016Batch2 Indonesia   111419
## SMB-ANK-027Batch1 SMB-ANK-027Batch1 Indonesia    39086
## SMB-ANK-027Batch2 SMB-ANK-027Batch2 Indonesia    46355
## SMB-ANK016Batch3   SMB-ANK016Batch3 Indonesia    27387
## SMB-ANK027Batch3   SMB-ANK027Batch3 Indonesia    48014
## SMB-WNG-021Batch1 SMB-WNG-021Batch1 Indonesia    53616
## SMB-WNG021Batch3   SMB-WNG021Batch3 Indonesia    35954
```

```
replicateDF=as.data.frame(sample_data(AllREadsSE_Indo_Counts_physeq)[replicates,])
replicateDF$SampleName = sub("([A-Z]{3})([0-9]{3})", "\\1-\\2", replicateDF$SampleName)
replicateDF$SampleName = gsub("Batch1",'',replicateDF$SampleName) %>% gsub("Batch2",'', .) %>% gsub("Ba
replicateDF=replicateDF[with(replicateDF, order(-SeqDepth)), ]
removeReplicates=rownames(replicateDF[which(duplicated(replicateDF$SampleName)),])
keepReplicates=rownames(sample_data(AllREadsSE_Indo_Counts_physeq))[-which(rownames(sample_data(AllREads

# prune these out
AllREadsSE_Indo_Counts_physeq=prune_samples(keepReplicates,AllREadsSE_Indo_Counts_physeq)

# remove taxa with only 0's in the phyloseq object
any(taxa_sums(AllREadsSE_Indo_Counts_physeq) == 0)
```

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

```
AllREadsSE_Indo_Counts_physeq=prune_taxa(taxa_sums(AllREadsSE_Indo_Counts_physeq) > 0, AllREadsSE_Indo_

# add sequencing depth information before filtering
SeqDepth_Prefilter = colSums(otu_table(AllREadsSE_Indo_Counts_physeq))
sample_data(AllREadsSE_Indo_Counts_physeq)$SeqDepth_Prefilter = SeqDepth_Prefilter

AllREadsSE_Indo_Counts_physeq
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 2391 taxa and 117 samples ]
## sample_data() Sample Data:       [ 117 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 2391 taxa by 8 taxonomic ranks ]
```

We now have a phyloseq object of 117 samples and 2,391 taxa.

## Reading in the control datasets

The first control dataset we'll read in is from a study conducted by Tran et al, conducted in 2016. This study looked at transcriptomic differences between individuals pre and post (natural) infection with P. falciparum. We'll only be using the healthy samples pre-infection, of which there are 54.

One of the reasons why this dataset is interesting is becase it comes from individuals living in areas that have a high pathogen load, similar to our study. Therefore, we have a positive control to compare to our samples.

The samples in the Malian study are 100-bp, paired end reads sequenced on an Illumina HiSeq 2000 and depleted of rRNA and globin RNA before amplification using the ScriptSeq Complete Gold Kit (Illumina).

Let's read in the data, make it into a PhyloSeq object, and add in our sample information.

### Malian Samples

```
Mali_Counts <- read.csv(paste0(refdir,"MaliControls_Counts_NoFilter_75BP.csv"),check.names=FALSE)

# Separate species' abundances and taxonomy columns
taxa_raw <- as.matrix(Mali_Counts[,c("Superkingdom","Kingdom","Phylum", "Class", "Order","Family","Genu
abund_raw <- as.matrix(Mali_Counts[,-which(colnames(Mali_Counts) %in% c("Superkingdom","Kingdom","Phylu
# convert to Phyloseq object
tax = tax_table(taxa_raw)
taxa = otu_table(abund_raw, taxa_are_rows = TRUE)
Mali_Counts_physeq = phyloseq(taxa, tax)

# add in sample information, i.e., the sample names and population they're from
samplenames <- colnames(otu_table(Mali_Counts_physeq))
pop <- rep("Mali",ncol(otu_table(Mali_Counts_physeq)))

# make this into a df and add to the Phloseq object
samples_df=data.frame(SampleName=colnames(otu_table(Mali_Counts_physeq)), SamplePop=pop)
samples = sample_data(samples_df)
rownames(samples)=samples$SampleName
sample_data(Mali_Counts_physeq) <- samples

# add sequencing depth information before filtering
SeqDepth_Prefilter = colSums(otu_table(Mali_Counts_physeq))
```

```r
sample_data(Mali_Counts_physeq)$SeqDepth_Prefilter = SeqDepth_Prefilter

# summarise data
Mali_Counts_physeq
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 3355 taxa and 54 samples ]
## sample_data() Sample Data:       [ 54 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 3355 taxa by 8 taxonomic ranks ]
```

We now have a phyloseq object of 54 samples by 3355 taxa.

## European samples

The next set of control samples is from a study conducted by Singhania et al in 2018. The data is 75-bp, paired end data sequenced on an Illumina HiSeq 4000. Whole blood was globin-depleted using the human GLOBINclear kit (Thermo Fisher Scientific).

This study was conducted on whole blood of tubercolosis (TB) and latent-(TB) patients to distinguish between active and asymptomatic TB infection, however the samples used for this analysis are all healthy controls.

As with the other studies, we'll first convert the data to a phyloseq object.

```r
European_Counts <- read.csv(paste0(refdir,"TBControls_NoFiltering_Counts.csv"),check.names=FALSE)

# Separate species' abundances and taxonomy columns
taxa_raw <- as.matrix(European_Counts[,c("Superkingdom","Kingdom","Phylum", "Class", "Order","Family","
abund_raw <- as.matrix(European_Counts[,-which(colnames(European_Counts) %in% c("Superkingdom","Kingdom
# convert to Phyloseq object
tax = tax_table(taxa_raw)
taxa = otu_table(abund_raw, taxa_are_rows = TRUE)
European_Counts_physeq = phyloseq(taxa, tax)

# add in sample information, i.e., the sample names and population they're from
samplenames <- colnames(otu_table(European_Counts_physeq))
pop <- rep("UK",ncol(otu_table(European_Counts_physeq)))

# make this into a df and add to the Phloseq object
samples_df=data.frame(SampleName=colnames(otu_table(European_Counts_physeq)), SamplePop=pop)
samples = sample_data(samples_df)
rownames(samples)=samples$SampleName
sample_data(European_Counts_physeq) <- samples

# add sequencing depth information before filtering
SeqDepth_Prefilter = colSums(otu_table(European_Counts_physeq))
sample_data(European_Counts_physeq)$SeqDepth_Prefilter = SeqDepth_Prefilter

# get phyloseq summary information
European_Counts_physeq
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 1026 taxa and 10 samples ]
## sample_data() Sample Data:       [ 10 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 1026 taxa by 8 taxonomic ranks ]
```

We now have a phyloseq object of 10 samples with 1026 taxa.

## Merging the data

The next step is to merge both the control and Indonesian data together. Phyloseq makes this relatively easy with its function merge_phyloseq().

HOWEVER, there is a caveat to this: although the merge_phyloseq function says that it merges by 'first separating higher-order objects into a list of their component objects; then, merging any component objects of the same class into one object', this is not the case (for my data, anyhow)!! I'm not the only one who had this problem, so in the end, the solution is to give each taxa name a unique ID, which in this case, is the species name (the most specific name). We'll then make sure that our phyloseq objects from above (before merging) are the same as after we merge.

```
# assign unique taxa names to all phyloseq objects
taxa_names(AllREadsSE_Indo_Counts_physeq) <- paste(tax_table(AllREadsSE_Indo_Counts_physeq)[,"Superking
taxa_names(Mali_Counts_physeq) <- make.unique(paste(tax_table(Mali_Counts_physeq)[,"Superkingdom"], tax_
taxa_names(European_Counts_physeq) <- make.unique(paste(tax_table(European_Counts_physeq)[,"Superkingdom
merged_phylo_counts=merge_phyloseq(AllREadsSE_Indo_Counts_physeq, Mali_Counts_physeq, European_Counts_ph

# subset populations
Indonesian_subset <- phyloseq::subset_samples(merged_phylo_counts, SamplePop == "Indonesia")
Mali_subset <- phyloseq::subset_samples(merged_phylo_counts, SamplePop == "Mali")
UK_subset <- phyloseq::subset_samples(merged_phylo_counts, SamplePop == "UK")

is.identical <- function(pop_to_subset, original_phyloseq_obj){
    pruned_pop_subset <- prune_taxa(taxa_sums(pop_to_subset) > 0, pop_to_subset)
    merged_df=merge(as.data.frame(otu_table(pruned_pop_subset)), as.data.frame(otu_table(original_phylos
    nsamples=ncol(otu_table(pruned_pop_subset))
    merged_df[,"Row.names"] <- NULL
    colnames(merged_df)=rep("Rep",ncol(merged_df))
    is.identical <- identical(merged_df[,1:nsamples],merged_df[,(nsamples+1):ncol(merged_df)])
    return(is.identical)
}

is.identical(pop_to_subset=Indonesian_subset, original_phyloseq_obj=AllREadsSE_Indo_Counts_physeq)
```

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

```
# TRUE
is.identical(pop_to_subset=Mali_subset, original_phyloseq_obj=Mali_Counts_physeq)
```

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

```
# TRUE
is.identical(pop_to_subset=UK_subset, original_phyloseq_obj=European_Counts_physeq)
```

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

```
# TRUE
```

They're all the same, so we can move forward.

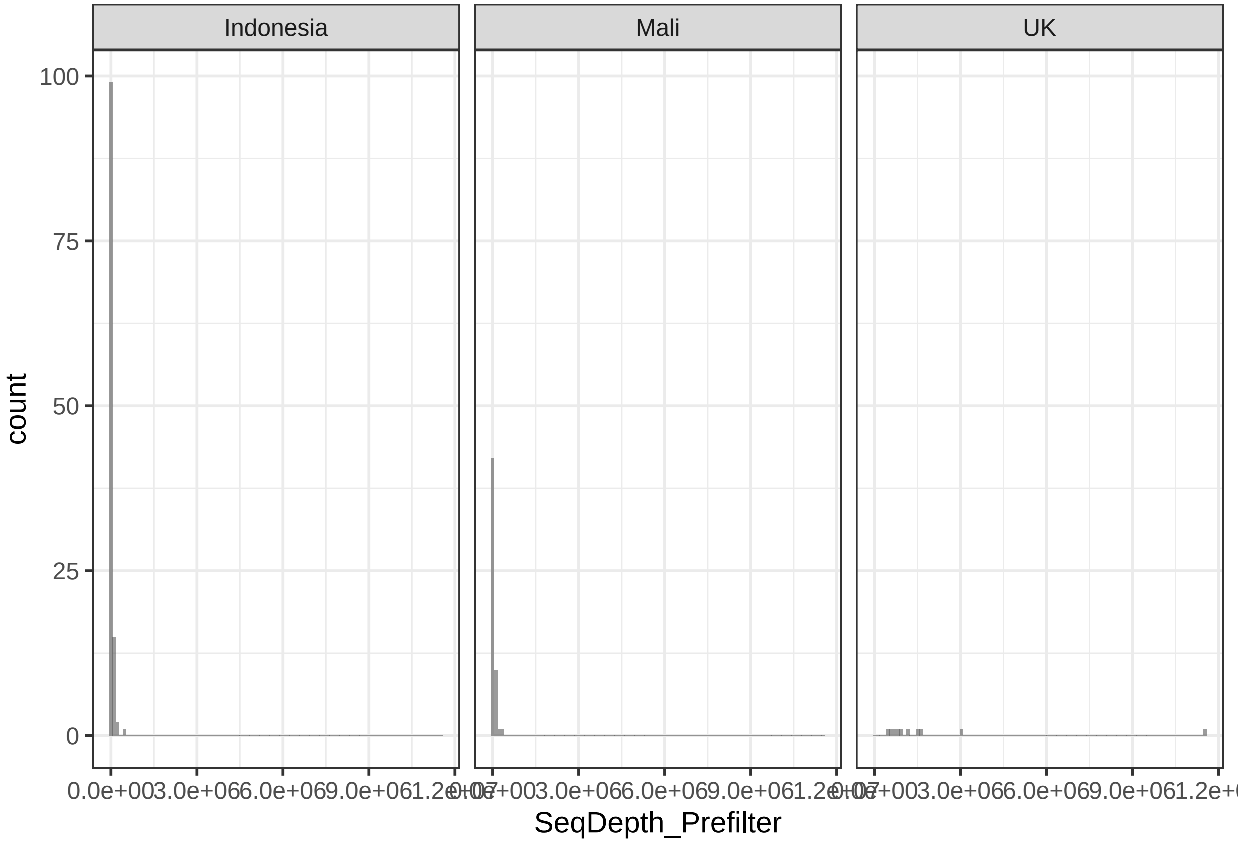
# Data processing

## Removing singletons from the data

The easiest way to get rid of some error in your data is to throw out any count information below some threshold. Oddly, in microbiomics, there's no set thresholding for this. In the end, it's really a compromise between accuracy and keeping rare taxa. What *is* decided is that filtering out at least singletons is standard, since these are regarded as sources of error or contamination. Some resources I found that were helpful on this can be found here and here.

In most of the datasets, we can see that there's a high number of low counts within the data.

```
# histogram of data
ggplot(meta(merged_phylo_counts)) + geom_histogram(aes(x = SeqDepth_Prefilter), alpha= 0.6, bins=100) +
```



With a higher sequencing depth, you can afford to play around with thresholding, however for the Indonesian dataset, the sequencing depth is variable and quite low in some samples. Therefore, pushing this threshold up too high will eliminate rare taxa, especially given that we didn't have a high library size to begin with.

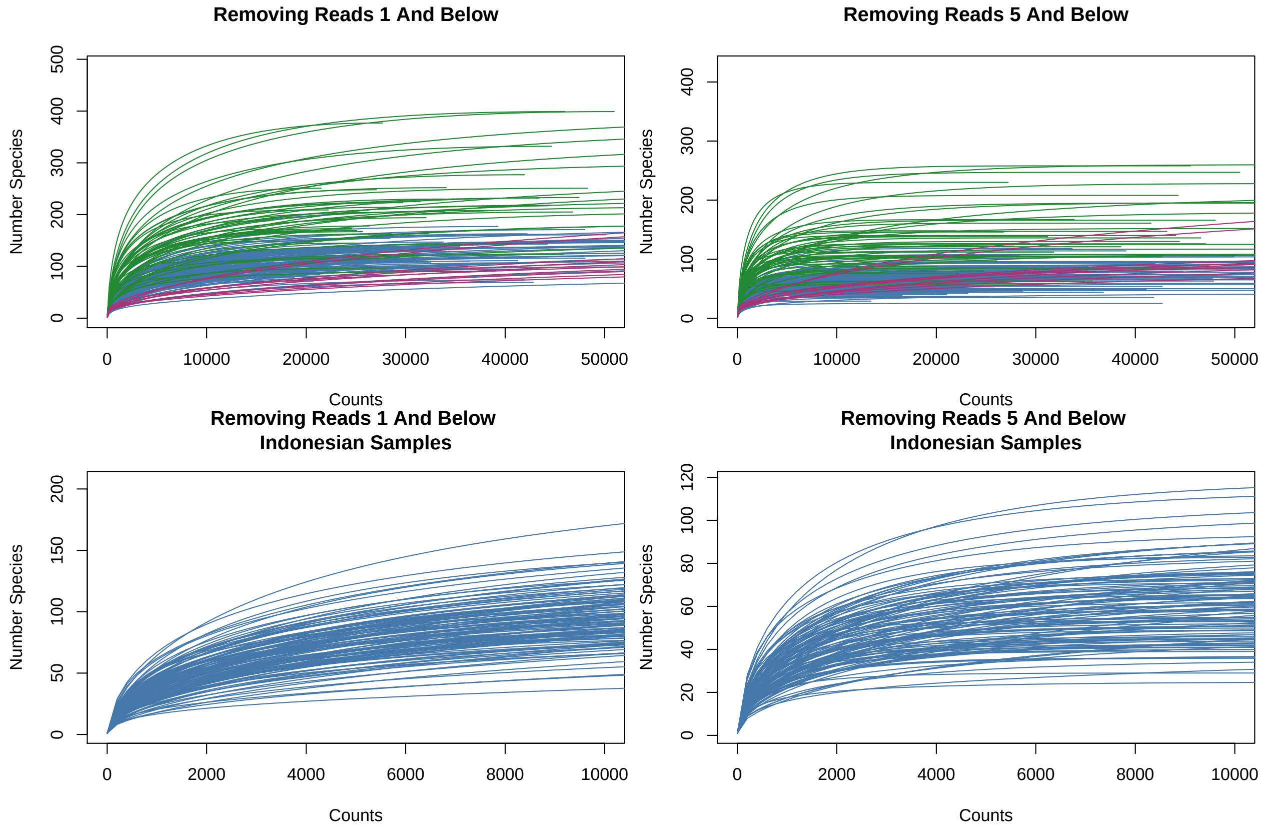
Although our starting library size is small, let's explore the data a bit by looking at the effect of removing singletons.

A great tool to do this is raraefaction curves. Rarefaction curves are commonly used in microbiomics to estimate 1) species richness and 2) determine how extensively a library was sampled. For the first point, it's nearly impossible to capture all species within a community, and therefore this allows for a way to estimate the total species we would expect to find by extrapolating the curve within the rarefaction plot. A rarefaction curve will (if sampled to a high enough depth) asymptote, and this point is regarded as the estimate of the total number of species within that community.

For the second point, we can also use the asymptote of the curve to see how extensively we sampled. If the curve has not started to flatten off, we have not captured everything.

Lets see how rarefaction looks like when we remove singletons and when we remove 5 counts.

```
# Separate species' abundances and taxonomy columns
rarecurve_counts <- otu_table(merged_phylo_counts)
col <- c(rep(IndonesiaCol,sum(sample_data(merged_phylo_counts)[,"SamplePop"]=="Indonesia")),rep(MaliCol
# Try with different filtering thresholds:
for (i in c(1,5)){
    rarecurve_counts[rarecurve_counts<=i]<-0
    rarecurve(t(otu_table(rarecurve_counts, taxa_are_rows = TRUE)), step=200, col=col,label=F, xlab="Co
}

Indonesian_subset <- prune_taxa(taxa_sums(Indonesian_subset) > 0, Indonesian_subset)
rarecurve_counts <- otu_table(Indonesian_subset)
col <- IndonesiaCol
for (i in c(1,5)){
    rarecurve_counts[rarecurve_counts<=i]<-0
    rarecurve(t(otu_table(rarecurve_counts, taxa_are_rows = TRUE)), step=200, col=col,label=F, xlab="Co
}
```



From the rarefaction curves, we can notice a few things. For one, the library size needed to capture all of the taxa is quite variable between studies and individuals. For example, some samples in the Indonesian study are not very diverse and don't need that many reads to capture all of the taxa (<5,000). In contrast to this, the Malian study (in green) is the most diverse, with over 20,000 reads necessary to capture the full diversity in some samples.

We might also notice that when removing we remove reads 5 and below (shown on the right hand side), the curve asymptotes much sooner. This is because you need more reads to detect rare species, and therefore removing reads 5 and below eliminates some of these rare species.

As mentioned before, because our starting read depth is small, we will stick with removing singletons. We will also make a copy of the unfiltered data for downstream use.

```
# Filter out singletons
otu_table(merged_phylo_counts)[otu_table(merged_phylo_counts)<=1]<-0
merged_phylo_counts <- prune_taxa(taxa_sums(merged_phylo_counts) > 0, merged_phylo_counts)
# add sequencing depth information after filtering out singletons
SeqDepth_noSingletons = colSums(otu_table(merged_phylo_counts))
sample_data(merged_phylo_counts)$SeqDepth_noSingletons = SeqDepth_noSingletons
```

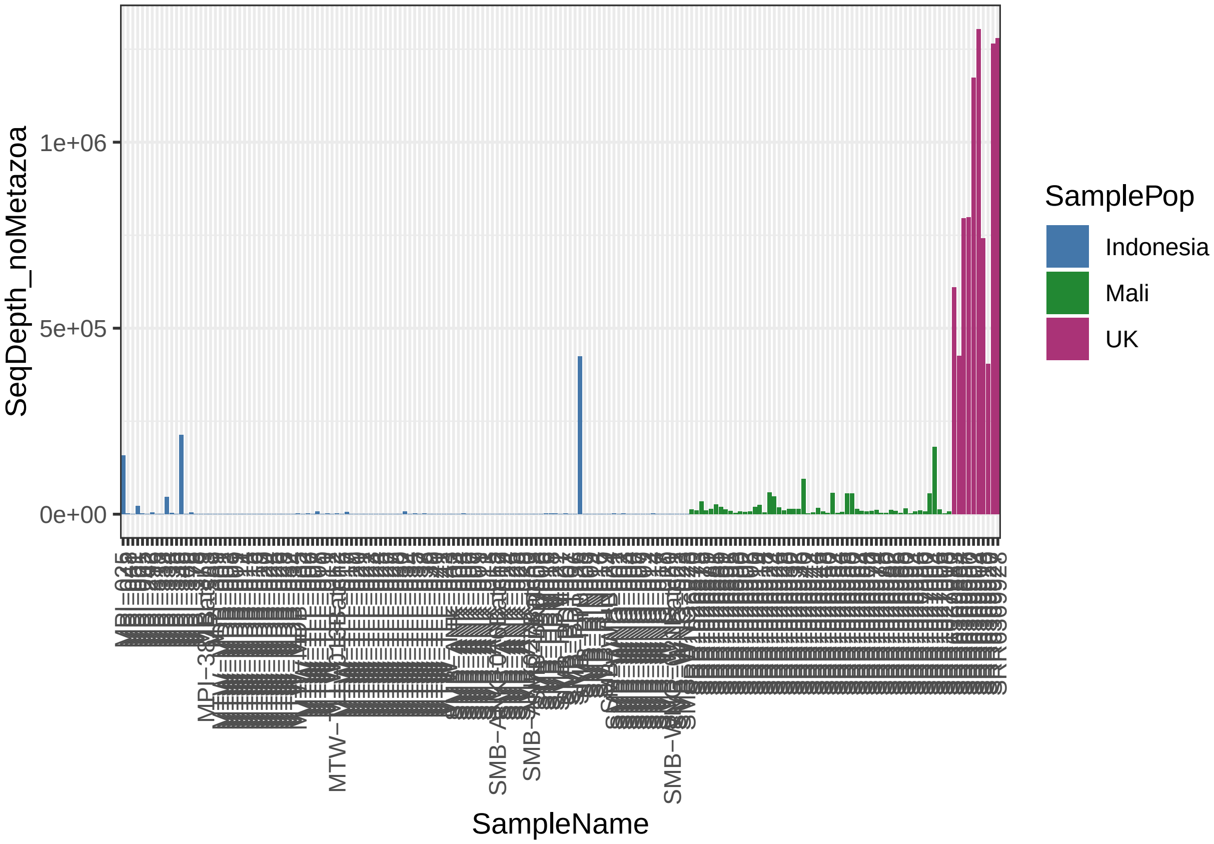
## Removing humans and plants

From the script X, we saw that human reads and viridiplantae are not of interest because of X. We'll filter these out.

```
# Filter out Viridiplantae
merged_phylo_counts <- subset_taxa(merged_phylo_counts, (Kingdom!="Viridiplantae"))
merged_phylo_counts <- prune_taxa(taxa_sums(merged_phylo_counts) > 0, merged_phylo_counts)
# add sequencing depth information after filtering out plants
SeqDepth_noViridiplantae = colSums(otu_table(merged_phylo_counts))
sample_data(merged_phylo_counts)$SeqDepth_noViridiplantae = SeqDepth_noViridiplantae

# Filter out Chordata
merged_phylo_counts <- subset_taxa(merged_phylo_counts, (Phylum!="Chordata"))
merged_phylo_counts <- prune_taxa(taxa_sums(merged_phylo_counts) > 0, merged_phylo_counts)
# add sequencing depth information after filtering out Metazoa
SeqDepth_noChordata = colSums(otu_table(merged_phylo_counts))
sample_data(merged_phylo_counts)$SeqDepth_noChordata = SeqDepth_noChordata

# Filter out Metazoa
merged_phylo_counts <- subset_taxa(merged_phylo_counts, (Kingdom!="Metazoa"))
merged_phylo_counts <- prune_taxa(taxa_sums(merged_phylo_counts) > 0, merged_phylo_counts)
# add sequencing depth information after filtering out Metazoa
SeqDepth_noMetazoa = colSums(otu_table(merged_phylo_counts))
sample_data(merged_phylo_counts)$SeqDepth_noMetazoa = SeqDepth_noMetazoa
```

Now that we've done all of the filtering, we can plot the final library sizes.

```
# barplot of library sizes
ggplot(meta(merged_phylo_counts), aes(SampleName, SeqDepth_noMetazoa)) + geom_bar(stat = "identity", aes
scale_fill_manual(values = c(IndonesiaCol,MaliCol,UKCol)) + rotate_x_text()
```

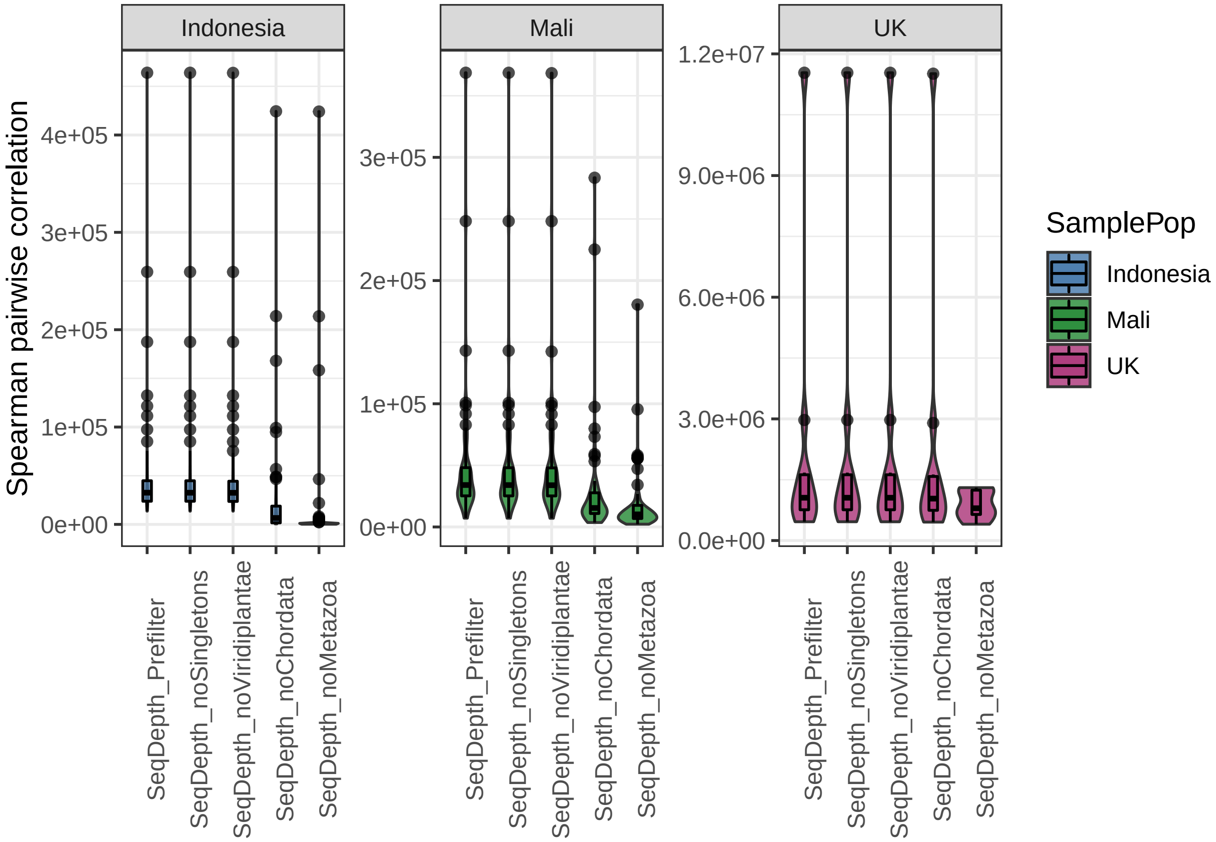


We can see that the library sizes are highly uneven, with the Indonesian data having the lowest sampling depth (with the exception of a few samples) and the UK dataset hacing the highest library depth.

### Summarising the data

The final step us is to summarise the data and see how many reads we lost at each filtering step.

```
FilteringSummary = sample_data(merged_phylo_counts)[,c("SamplePop","SeqDepth_Prefilter","SeqDepth_noSin

# melt df and plot
melted_FilteringSummary = melt(FilteringSummary)
ggplot(melted_FilteringSummary, aes(x=variable, y=value, fill=SamplePop)) +
  geom_violin(alpha=0.8) + theme_bw() + ylab("Spearman pairwise correlation") +
  theme(axis.title.x=element_blank(), axis.text.x = element_text(angle = 90)) + scale_fill_manual(value
  geom_boxplot(color="black",width=0.2, alpha = 0.7) + facet_wrap(~ SamplePop, scales = "free")
```



We can see that in the Indonesian and Malian dataset, most of the reads are removed when removing Chordates, however for the UK dataset, most reads are removed when removing Metazoa. For the UK datset, this is due to a high number of reads mapping to molluscs, as seen in script X.

## Data normalisation

The library sizes between samples and groups is highly variable, and therefore comparing the data to each other will result in biased results.

There are two ways of handling this: 1. Performing a transformation of the data 2. Rarefying the data

### Centered log-ration transformation

Taxa can be viewed by their relative abundance, however changes in the abundance of one taxon will result in changing the abundance of other taxa.

One of the ways to handle this is to transform the data using Centered Log Ratio (CLR)transformation. CLR data shows how OTUs behave relative to the per-sample average and is a commonly-used data transformation method in microbiomics.

Another cool thing about using CLR-transformed data is that it is not affected by sequencing depth. This excerpt from a paper by Gloor et al explains this really well:

“The clr-transformed values are scale-invariant; that is the same ratio is expected to be obtained in a sample with few read counts or an identical sample with many read counts, only the precision of the clr estimate is affected. This is elaborated in the “Probability” and “Log-ratio transformations” section in the Supplement, but the consequence is that count normalization is unnecessary and indeed, undesirable since information on precision is lost."

Unfortunately, one of the disadvantages to using CLR-transformed data is that it can't be used in diversity estimates, and it's also hard to visualise.

Becasue CLR data is an informative measure of our data, I'll first explore sample grouping using this method.

The first step to performing a CLR transformation on the data is to add an offset of 1 to the counts. This is necessary, since performing a log on 0 values is undefined. We'll then perform a log ratio transformation of the data using the mixOmics package.

```
offset_otu=otu_table(merged_phylo_counts)+1
transform_counts=t(otu_table(offset_otu))
data_clr <- logratio.transfo(as.matrix(transform_counts), logratio = 'CLR', offset = 0)
```

### Sample grouping

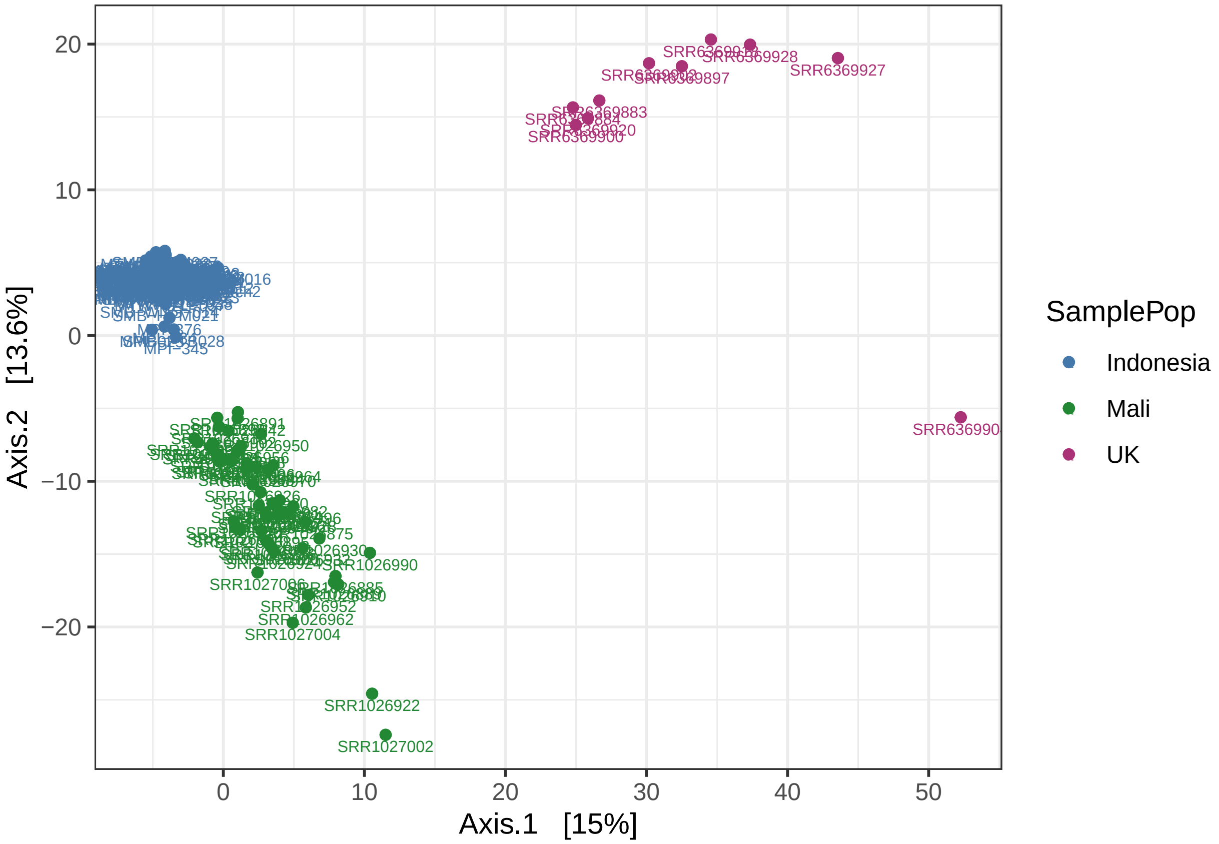
Now that we've transformed our data, we can make a PCA plot to see how each sample clusters. The current obect we have is a CLR-class object. You can plot this type of data object easily with miOmics, however I prefer the visualisation that phyloseq offers (you can't alter the PCA plots that much in mixOmics). So, we'll turn the clr object back into a phyloseq object and make an ordination plot of the data.

!Note: When Euclidean distances are used in PCoA plots, it is equivalent to a PCA plot.

```
# Make a duplicated phyloseq object to use for plotting
class(data_clr)="matrix"
```

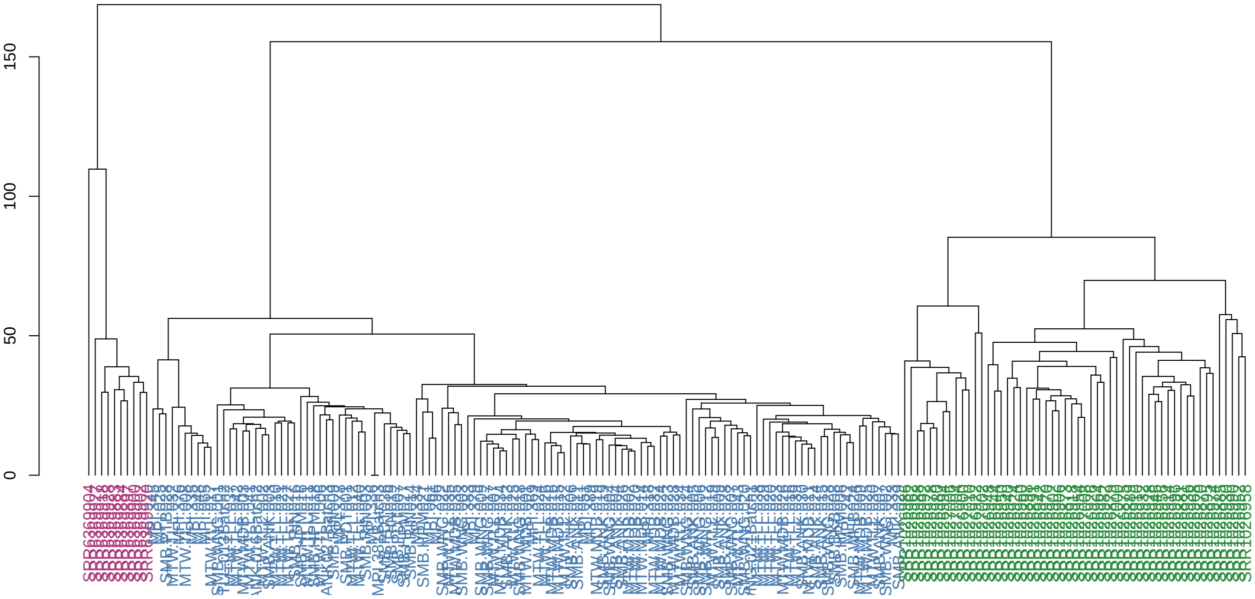
```
## Warning in class(data_clr) = "matrix": Setting class(x) to "matrix" sets
## attribute to NULL; result will no longer be an S4 object
```

```
taxa = otu_table(t(data_clr), taxa_are_rows = TRUE)
merged_phylo_counts_clr=merged_phylo_counts
otu_table(merged_phylo_counts_clr)=taxa

out.wuf.log <- ordinate(merged_phylo_counts_clr, method = "PCoA", distance = "euclidean")
plot_ordination(merged_phylo_counts_clr, out.wuf.log, color="SamplePop", axes = 1:2, label="SampleName"
```



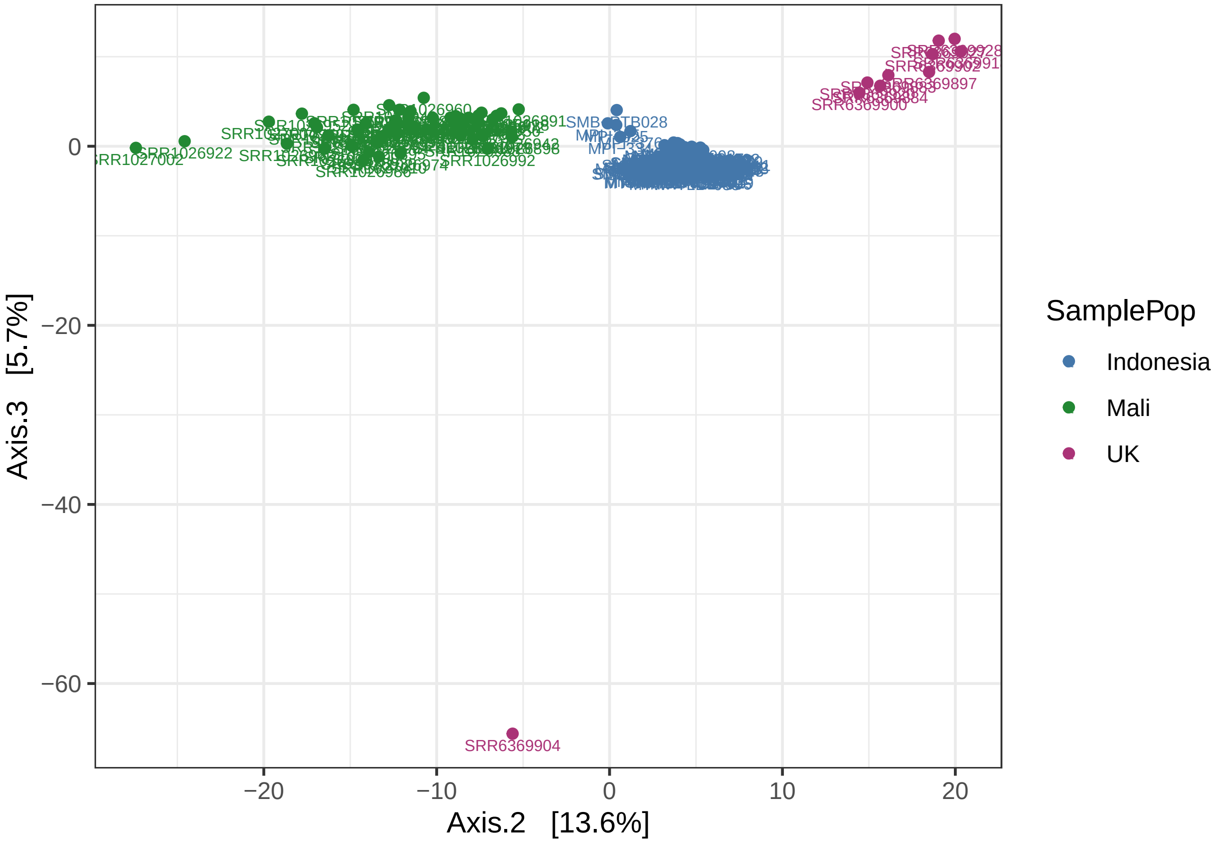
We can see that the first principal component is separating the three studies apart. This is also supported by hierarchical clustering of the clr-tranformed data by euclidean distance.

```
ps_otu <- data.frame(phyloseq::otu_table(merged_phylo_counts_clr))
ps_otu <- t(ps_otu)
bc_dist <- vegan::vegdist(ps_otu, method = "euclidean")
ward <- as.dendrogram(hclust(bc_dist, method = "ward.D2"))
#Provide color codes
meta <- data.frame(phyloseq::sample_data(merged_phylo_counts_clr))
colorCode <- c(Indonesia = IndonesiaCol, `Mali` = MaliCol, `UK` = UKCol)
labels_colors(ward) <- colorCode[meta$SamplePop][order.dendrogram(ward)]
#Plot
plot(ward)
```

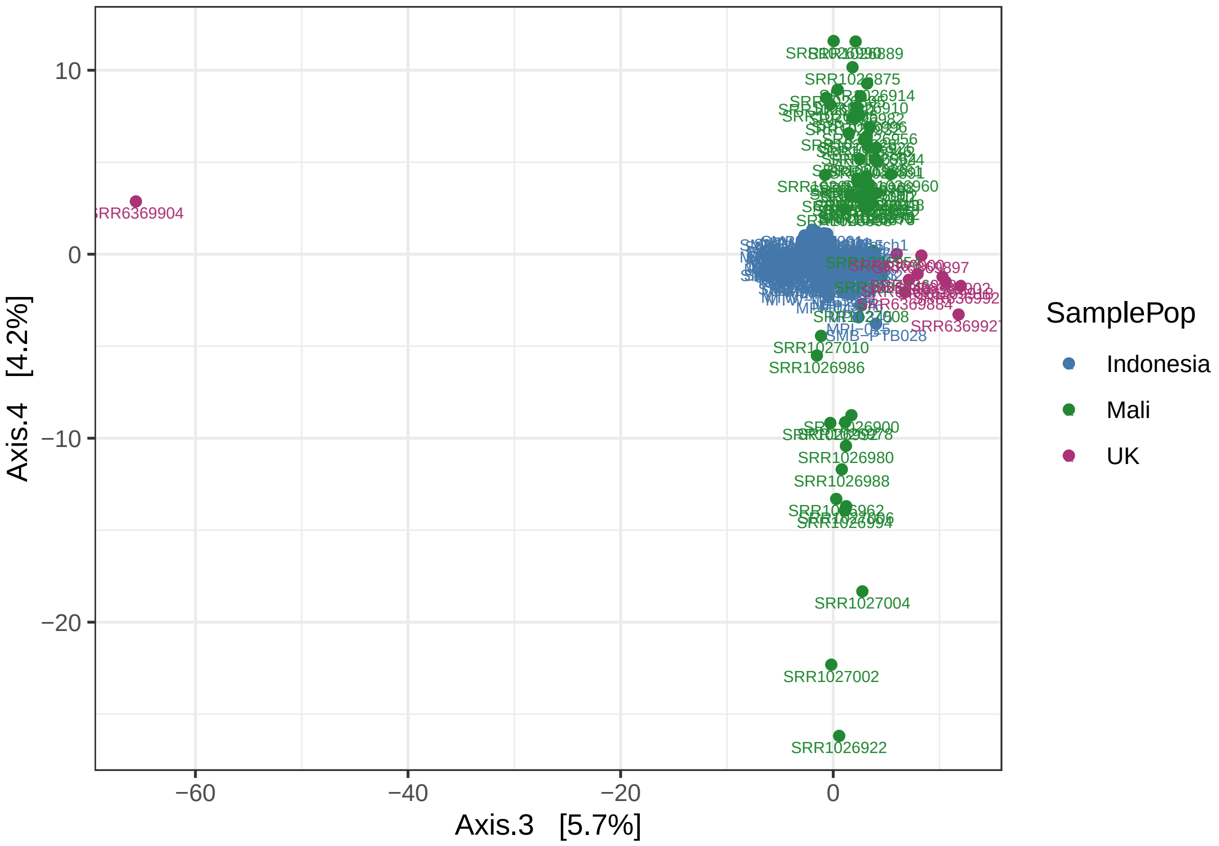


If we look at the second principal component, we can see that it splits the UK samples apart from the Malian and Indonesian samples, while the Indonesian and Malian samples come together. Finally, axis three separates the three studies from an outlier sample in the UK study - SRR6369904. We'll investigate why this samples is an outlier (a bit later).

```
plot_ordination(merged_phylo_counts_clr, out.wuf.log, color="SamplePop", axes = 2:3, label="SampleName"
```



```
plot_ordination(merged_phylo_counts_clr, out.wuf.log, color="SamplePop", axes = 3:4, label="SampleName")
```



## Relative frequency of taxa

One of the questions we're most interested in when investigating these samples is: what is in the data? One of the ways to do this is by visualising the data itself. A common way to do this is by looking at a stacked barplot, one for each sample, composed of the relative frequency of taxa in that sample.

Why do we look at relative frequency? As we've seen before, our library sizes are uneven, and therefore we want to see what the proportion of each taxa is in each sample. However, compositional data does not account for the fact that as one species goes up, it will force another species to go down (i.e., it is bounded).

I'll try to solve this proble in a few ways. The first way is by plotting the relative taxa, then seeing how this compares to the rarefied data. Finally, I'll plot the CLR data (which is unconstrained) to see how this looks.

To visualise relative frequency, I want to show my taxa at the family level, however since there are over 450 unique taxa at the family level, this would be difficult to visualise with so many colours. Instead, we'll make a new taxa variable which combines Superkingdom information with Family-level information. This way, we can highlight colours by superkingdom (which isn't so visually overwhelming), and still preserve Family-level information.

### Unsubsampled compositional data

```
# add a new column containing family names and superkingdom
tax_table(merged_phylo_counts)[,"Superkingdom"] = paste(tax_table(merged_phylo_counts)[,"Superkingdom"]
tax_table(merged_phylo_counts)[,"Superkingdom"] <- gsub("Bacteria_$", "Bacteria_unclassified", tax_tabl
tax_table(merged_phylo_counts)[,"Superkingdom"] <- gsub("Eukaryota_$", "Eukaryota_unclassified", tax_tab
tax_table(merged_phylo_counts)[,"Superkingdom"] <- gsub("Viruses_$", "Viruses_unclassified", tax_table(m
```

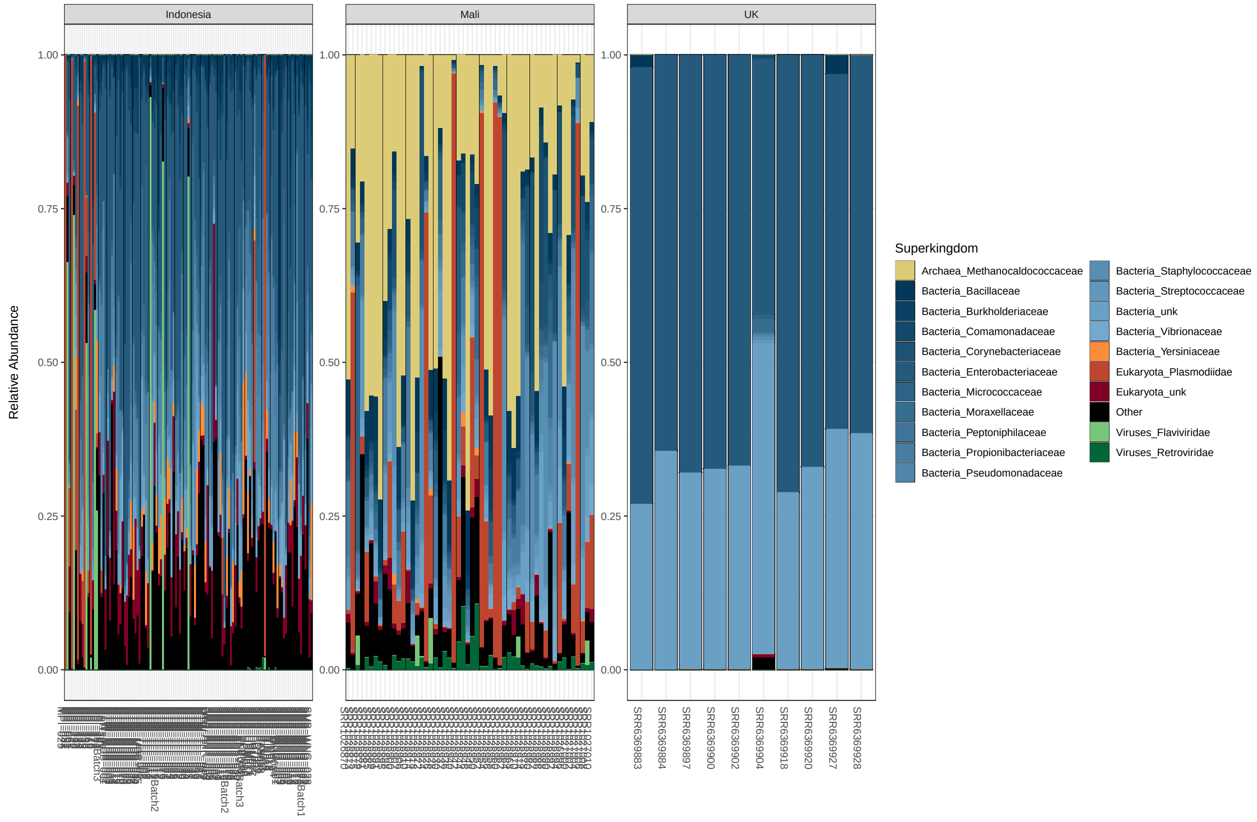
As pointed out, we have a lot of taxa at the family level, and it would be hard to look over everything at once. Instead, we can focus on the most prevalent taxa and highlight everything else in another colour.

Here, I chose to highlight the top 20 taxa, since that's still representative while not being too visually exhausting.

```
aggregated_phyloCounts <- aggregate_top_taxa(merged_phylo_counts, "Superkingdom", top = 20)
# transform to relative counts
relative_phyloCounts <- microbiome::transform(aggregated_phyloCounts, "compositional")
# Remove weird extra family names added at the end of Superkingdom names
tax_table(relative_phyloCounts)[,"Superkingdom"] <- paste(sapply(strsplit(taxa_names(relative_phyloCount
# Change "Other_NA" to just "Other"
tax_table(relative_phyloCounts)[,"Superkingdom"][grep("Other", taxa_names(relative_phyloCounts))] = "Oth

# Plot
p=plot_bar(relative_phyloCounts, fill = "Superkingdom")

# set colour palette
families=levels(p$data$Superkingdom)
# get number of families in each kingdom
table(sapply(strsplit(families, "[_.]"), `[`, 1))
```

```
##
##   Archaea  Bacteria Eukaryota     Other   Viruses
##         1        15         2         1         2
```

```
PaletteArchaea = colorRampPalette(c("#DDCC77"))(1)
PaletteBacteria = colorRampPalette(c("#023858","#74a9cf"))(14)
PaletteEukaryote = colorRampPalette(c("#fd8d3c","#800026"))(3)
PaletteOther = colorRampPalette(c("black"))(1)
PaletteVirus = colorRampPalette(c("#78c679","#006837"))(2)

Merged_Palette <- c(PaletteArchaea,PaletteBacteria,PaletteEukaryote,PaletteOther,PaletteVirus)

phyloseq::plot_bar(relative_phyloCounts, fill = "Superkingdom") +
  geom_bar(aes(fill = Superkingdom), stat = "identity", position = "stack") +
  labs(x = "", y = "Relative Abundance\n") +
  facet_wrap(~ SamplePop, scales = "free") + scale_fill_manual(values=Merged_Palette) +
  theme(panel.background = element_blank(), axis.ticks.x=element_blank())
```



We can see that all populations have a high bacterial load, however in the Malian and Indonesian dataset, there'a also a high abundance of Plasmodium and flavivirus. We can also see that the Malian dataset has a high proportion of reads mapping to archaea.

Now let's check to see how this compares to the rarefied data.

## Rarefied compositional data

One of the most common ways to normalise data is through rarefying, or subsampling, the entire library to the lowest read depth in the dataset. Rarefaction is performed by drawing reads without replacement from each sample so that all samples have the same number of total counts. This process standardizes the library size across samples and is especially important for calulating diversity metrics, where read depth influences microbe diversity.

From what I've found, many people in the microbiomics community have very strong opinions about rarefying data. Some camps, such as QIIME, think that it's just fine, whereas others, such as the creators of Phyloseq, strongly advise against it. A seminal, and of the most well-referenced papers about the disadvantages of rarefying data, was published in 2014 by McMurdie & Holmes. The arguments they have echo the main concerns seen within the 'anti-rarefy' camps: namely that rarefying data 1) decreases the ability to detect rare taxa and 2) can lead to unequally-rarefied data due to rare taxa being over or underrepresented in libraries normalised to a small size.

Although some authors argue that rarefaction is, in fact, a perfectly suitable option (particularly for small library sizes), I do think this is a valid point.

Recently, a package called SRS (standing for Scaling with Ranked Subsampling) came out in R and it preserves OTU frequencies by 1) scaling counts by a constant factor where the sum of the scaled counts equals the minimum library size chosen by the user and 2) performing a ranked subsampling on the data.

For this study, I chose Cmin (i.e., the number of counts to which all samples will be normalized) to 2,000 since a seminal paper in the microbiome field tested this and found that 2,000 single-end reads are sufficient for detecting most communities.

SRS won't work if we have samples with a library size under this threshold, so let's remove all samples under 2000, then perform SRS.

```
minControl=2000
keep=names(which(sample_sums(merged_phylo_counts)>=minControl))
pruned_data=prune_samples(keep, merged_phylo_counts)
any(taxa_sums(pruned_data) == 0)
```

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

```
# TRUE
pruned_data <- prune_taxa(taxa_sums(pruned_data) > 0, pruned_data)
any(taxa_sums(pruned_data) == 0)
```

```
## [1] FALSE
```

```
# FALSE
pruned_data_df=as.data.frame(otu_table(pruned_data))
SRS=SRS(pruned_data_df,minControl)
rownames(SRS)=rownames(pruned_data_df)

# transform back into phyloseq object
taxa = otu_table(SRS, taxa_are_rows = TRUE)
otu_table(pruned_data)=taxa
any(taxa_sums(pruned_data) == 0)
```

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

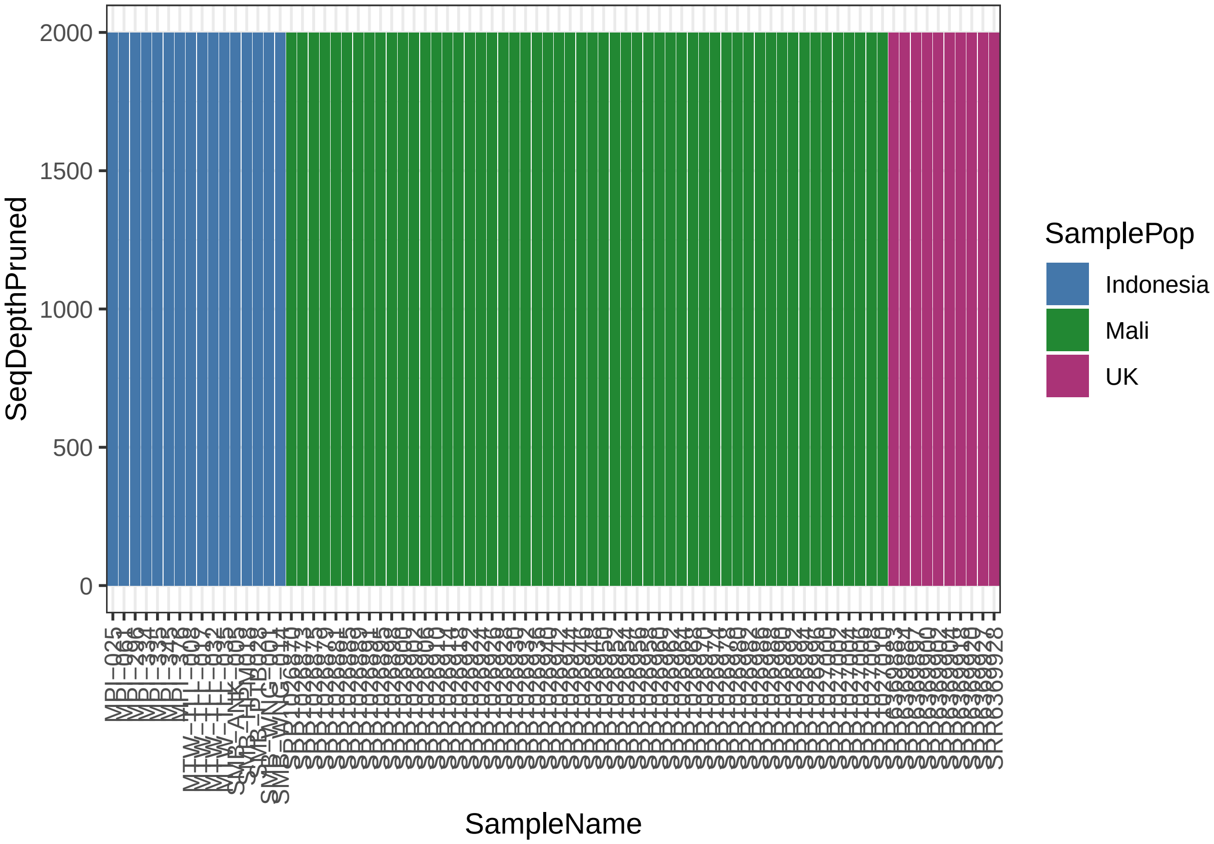
```
# TRUE
pruned_data <- prune_taxa(taxa_sums(pruned_data) > 0, pruned_data)
any(taxa_sums(pruned_data) == 0)
```

```
## [1] FALSE
```

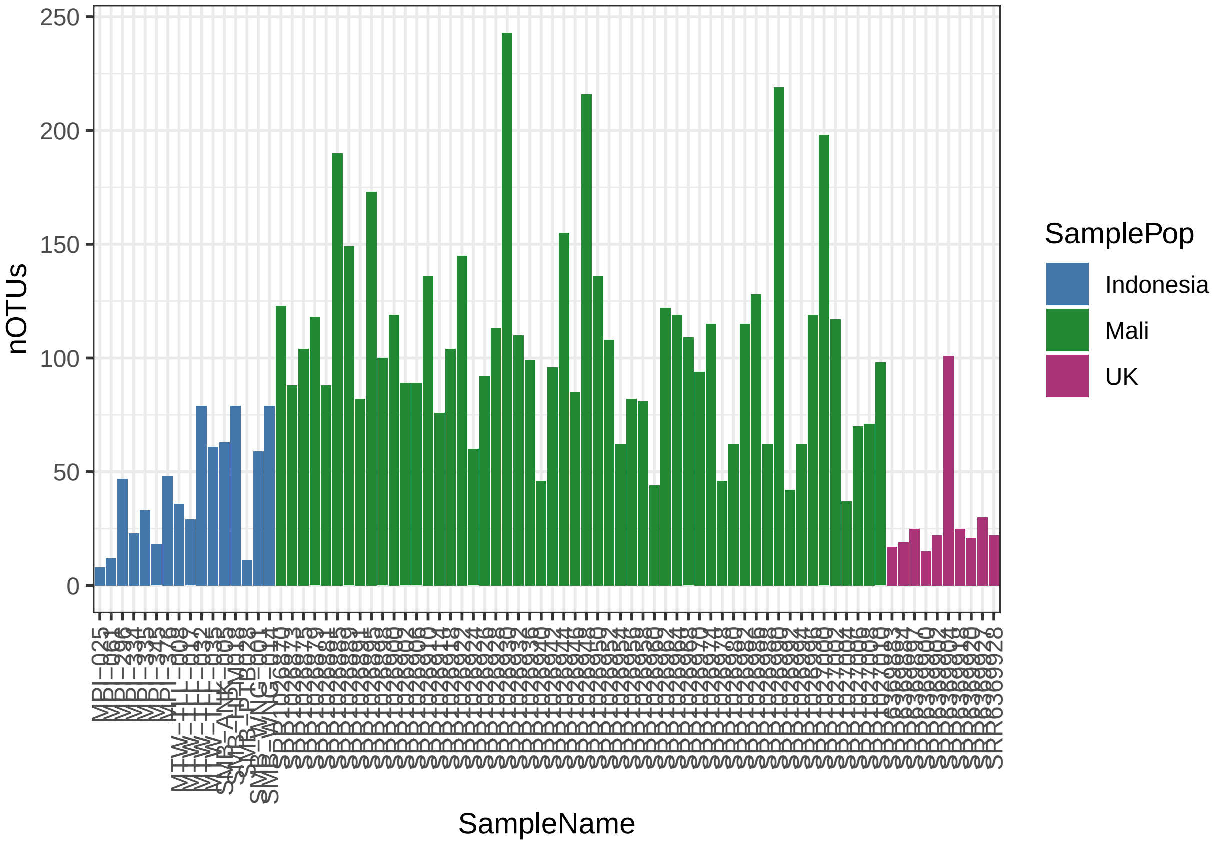
```
# FALSE
```

Now let's make sure the library sizes are the same and see how the OTU numbers look like between datasets.

```
SeqDepthPruned = sample_sums(pruned_data)
sample_data(pruned_data)$SeqDepthPruned = SeqDepthPruned

# barplot of library sizes
ggplot(meta(pruned_data), aes(SampleName, SeqDepthPruned)) + geom_bar(stat = "identity", aes(fill = Samp
scale_fill_manual(values = c(IndonesiaCol,MaliCol,UKCol)) + rotate_x_text()
```



```
# get barplot of total counts per individual
nOTUs = colSums(otu_table(pruned_data)!=0)
sample_data(pruned_data)$nOTUs = nOTUs

# barplot of OTUs
ggplot(meta(pruned_data), aes(SampleName, nOTUs)) + geom_bar(stat = "identity", aes(fill = SamplePop)) +
scale_fill_manual(values = c(IndonesiaCol,MaliCol,UKCol)) + rotate_x_text()
```

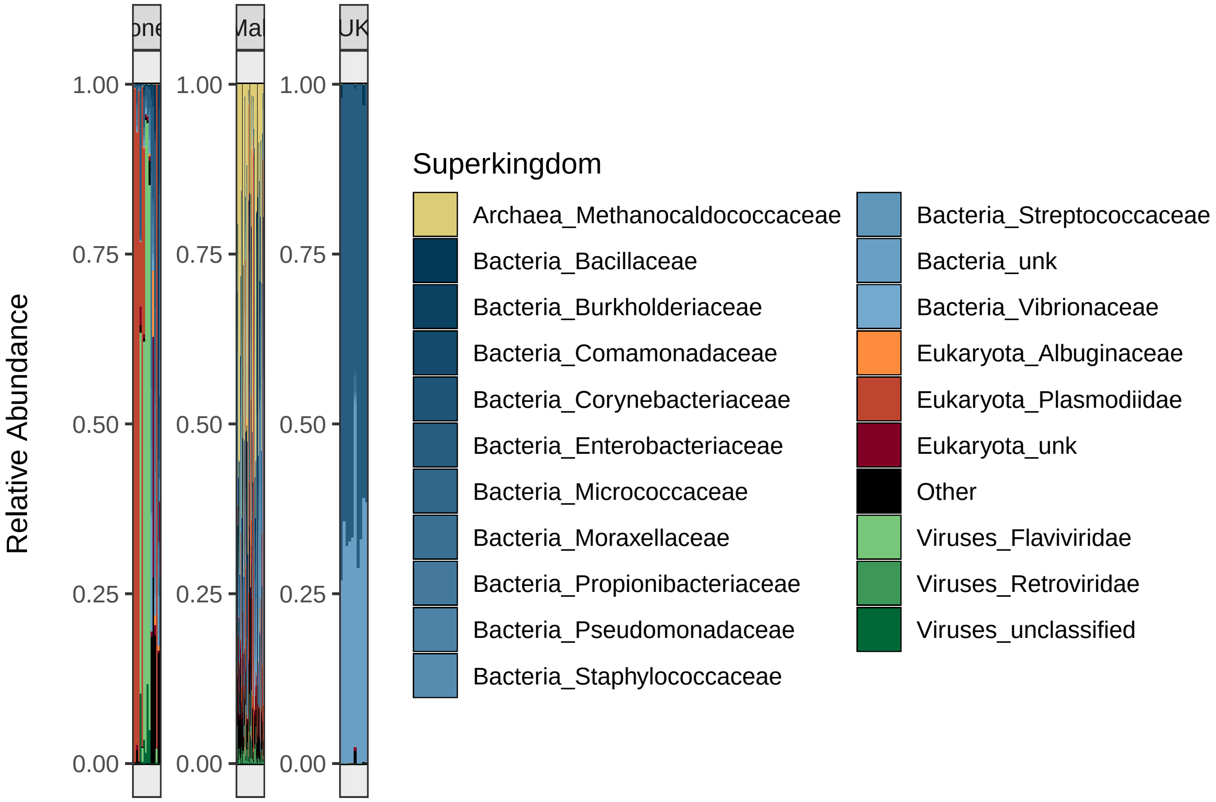


Now lets compare the rarefied data to the compositional data.

```
# add a new column containing family names and superkingdom
tax_table(pruned_data)[,"Superkingdom"] = paste(tax_table(pruned_data)[,"Superkingdom"], tax_table(prune
tax_table(pruned_data)[,"Superkingdom"] <- gsub("Bacteria_$", "Bacteria_unclassified", tax_table(pruned_
tax_table(pruned_data)[,"Superkingdom"] <- gsub("Eukaryota_$", "Eukaryota_unclassified", tax_table(prune
tax_table(pruned_data)[,"Superkingdom"] <- gsub("Viruses_$", "Viruses_unclassified", tax_table(pruned_da

# For some reason, top is actually top + 1, so here it would be 20
aggregated_phyloCounts <- aggregate_top_taxa(pruned_data, "Superkingdom", top = 20)
# transform to relative counts
relative_phyloCounts <- microbiome::transform(aggregated_phyloCounts, "compositional")
# Remove weird extra family names added at the end of Superkingdom names
tax_table(relative_phyloCounts)[,"Superkingdom"] <- paste(sapply(strsplit(taxa_names(relative_phyloCount
# Change "Other_NA" to just "Other"
tax_table(relative_phyloCounts)[,"Superkingdom"][grep("Other", taxa_names(relative_phyloCounts))] = "Ot

# Plot
p=plot_bar(relative_phyloCounts, fill = "Superkingdom")

# set colour palette
families=levels(p$data$Superkingdom)
# get number of families in each kingdom
table(sapply(strsplit(families, "[_.]"), `[`, 1))
```

```
##
##   Archaea  Bacteria Eukaryota     Other   Viruses
##         1        13         3         1         3
```

```
PaletteArchaea = colorRampPalette(c("#DDCC77"))(1)
PaletteBacteria = colorRampPalette(c("#023858","#74a9cf"))(13)
PaletteEukaryote = colorRampPalette(c("#fd8d3c","#800026"))(3)
PaletteOther = colorRampPalette(c("black"))(1)
PaletteVirus = colorRampPalette(c("#78c679","#006837"))(3)

Merged_Palette <- c(PaletteArchaea,PaletteBacteria,PaletteEukaryote,PaletteOther,PaletteVirus)

phyloseq::plot_bar(relative_phyloCounts, fill = "Superkingdom") +
  geom_bar(aes(fill = Superkingdom), stat = "identity", position = "stack") +
  labs(x = "", y = "Relative Abundance\n") +
  facet_wrap(~ SamplePop, scales = "free") + scale_fill_manual(values=Merged_Palette) +
  theme(panel.background = element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank())
```



Although a lot of the samples are missing, the same trend still holds true- bacteria dominating the UK samples, while apicomplexa, flavivirus, and archae being dominant in the Indonesian and Malian samples.

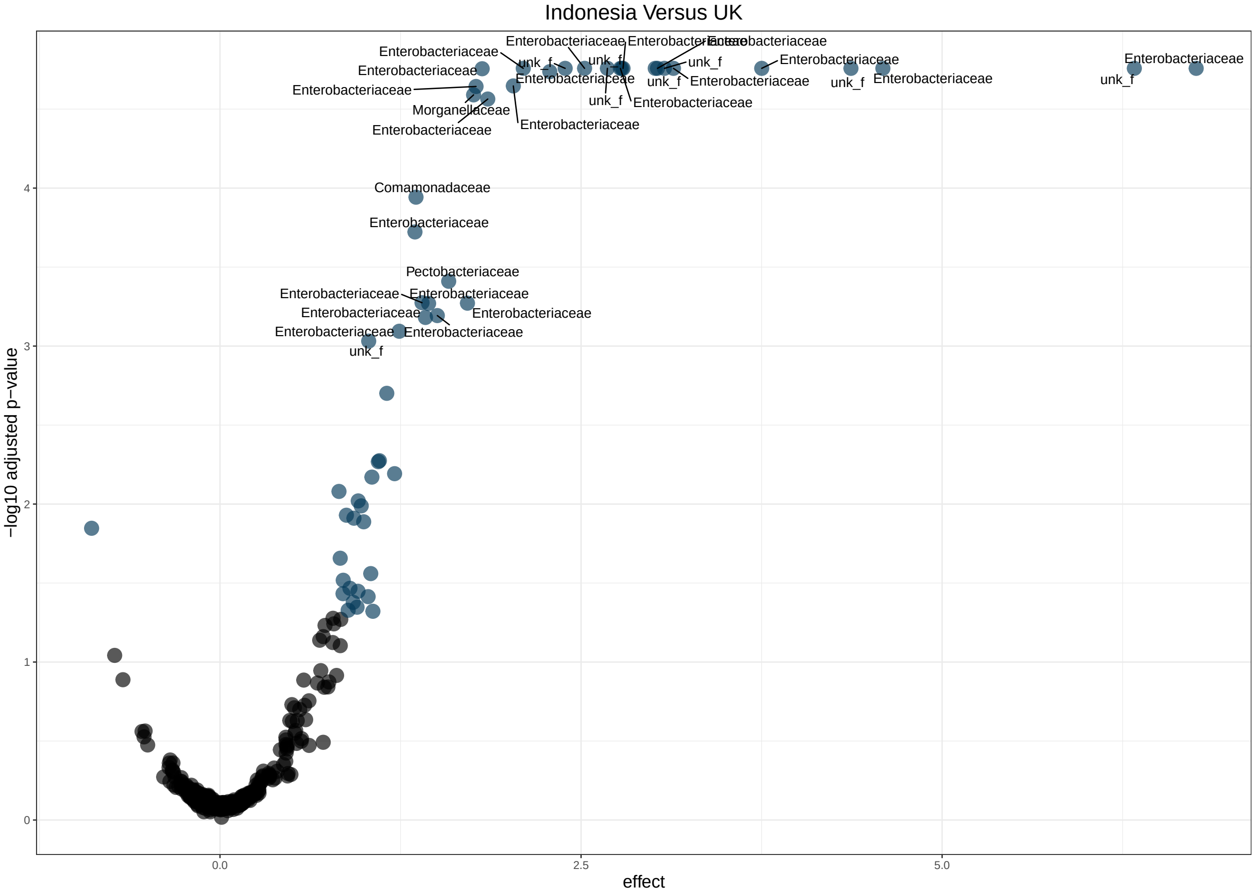
## Differential abundance testing

We're intersted in testing whether the species composition between populations is significantly different. Visually, we saw that the populations look different, but we can't say that for sure. One way to test this is through differnetial abundance testing.

```
# Differnetial abundance testing
IndoVsUK=subset_samples(merged_phylo_counts, SamplePop != "Mali")
any(taxa_sums(IndoVsUK) == 0)
```

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

```r
# TRUE
IndoVsUK <- prune_taxa(taxa_sums(IndoVsUK) > 0, IndoVsUK)
taxa_names(IndoVsUK)=make.unique(tax_table(IndoVsUK)[,"Family"])
aldex2_IndoVsUK <- ALDEx2::aldex(data.frame(phyloseq::otu_table(IndoVsUK)), phyloseq::sample_data(IndoVs
sig_aldex2_IndoVsUK <- aldex2_IndoVsUK %>%
  rownames_to_column(var = "OTU") %>%
  filter(wi.eBH < 0.05) %>%
  arrange(effect, wi.eBH) %>%
  dplyr::select(OTU, diff.btw, diff.win, effect, wi.ep, wi.eBH)
# set significance colours
aldex2_IndoVsUK$threshold <- aldex2_IndoVsUK$we.eBH <= 0.05
aldex2_IndoVsUK$threshold = as.numeric(aldex2_IndoVsUK$threshold) + 1

# adjust label names
labels = sapply(strsplit(rownames(aldex2_IndoVsUK), "[..]"), `[`, 1) %>% gsub("unk_p","Fungi",.)
taxa_superkingdom = sapply(strsplit(tax_table(IndoVsUK)[,"Superkingdom"], "[_.]"), `[`, 1) %>% gsub("unk

# plot
ggplot(aldex2_IndoVsUK) +
  geom_point(aes(x = effect, y = -log10(wi.eBH)), color = ifelse(aldex2_IndoVsUK$wi.eBH <= 0.05, c("grey
  #geom_text_repel(aes(x = effect, y = -log10(wi.eBH), label = rownames(aldex2_IndoVsDutch))) +
  geom_text_repel(aes(x = effect, y = -log10(wi.eBH), label = ifelse(wi.eBH <= 0.001, labels,""))) +
  ggtitle("Indonesia Versus UK") +
  xlab("effect") +
  ylab("-log10 adjusted p-value") +
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

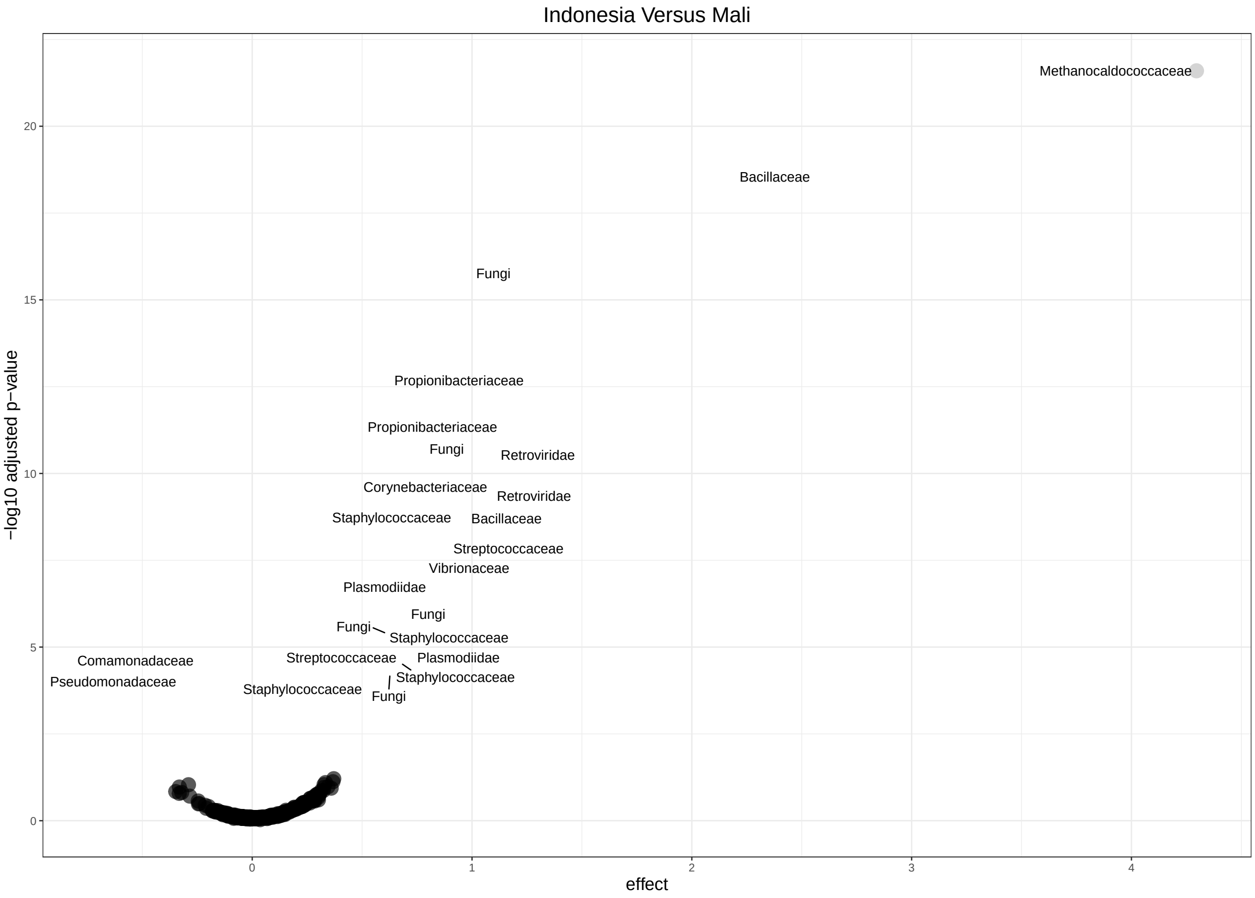


```r
IndoVsMali=subset_samples(merged_phylo_counts, SamplePop != "UK")
any(taxa_sums(IndoVsMali) == 0)
```

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

```r
# TRUE
IndoVsMali <- prune_taxa(taxa_sums(IndoVsMali) > 0, IndoVsMali)
taxa_names(IndoVsMali)=make.unique(tax_table(IndoVsMali)[,"Family"])
aldex2_IndoVsMali <- ALDEx2::aldex(data.frame(phyloseq::otu_table(IndoVsMali)), phyloseq::sample_data(In
sig_aldex2_IndoVsMali <- aldex2_IndoVsMali %>%
  rownames_to_column(var = "OTU") %>%
  filter(wi.eBH < 0.05) %>%
  arrange(effect, wi.eBH) %>%
  dplyr::select(OTU, diff.btw, diff.win, effect, wi.ep, wi.eBH)

# set significance colours
aldex2_IndoVsMali$threshold <- aldex2_IndoVsMali$we.eBH <= 0.05
aldex2_IndoVsMali$threshold = as.numeric(aldex2_IndoVsMali$threshold) + 1

# adjust label names
labels = sapply(strsplit(rownames(aldex2_IndoVsMali), "[..]"), `[`, 1) %>% gsub("unk_f","Fungi",.)
# plot
ggplot(aldex2_IndoVsMali) +
  geom_point(aes(x = effect, y = -log10(wi.eBH)), color = ifelse(aldex2_IndoVsMali$wi.eBH <= 0.05, c("g
  geom_text_repel(aes(x = effect, y = -log10(wi.eBH), label = ifelse(wi.eBH <= 0.001, labels,""))) +
  #geom_text_repel(aes(x = effect, y = -log10(wi.eBH), label = ifelse(wi.eBH <= 0.001, labels,""))) +
  ggtitle("Indonesia Versus Mali") +
  xlab("effect") +
  ylab("-log10 adjusted p-value") +
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

```
## Warning: Removed 53 rows containing missing values (geom_point).
```



# Alpha diversity

Alpha diversity measures within-sample diverity and looks at how many taxa are observed, as well as how evenly they are distributed.

There is a lot of controversy around how best to analyse alpha diversity. Because a higher sequencing depth will lead to a greater likelihood of diversity, many people rarefy their data beforehand. However, rarefying data (as pointed out above), not only discards data, but leads to biases when rarefied.

Current tools to estimate alpha diversity either underestimate richness or underestimate uncertainty, however DivNet is a package that adresses these problems. Divnet is a method for estimating within- and between-community diversity in ecosystems where taxa interact via an ecological network. It accounts for differences in sequencing depth and estimates the number of missing species based on the sequence depth and number of rare taxa in the data.

To use DivNet, you need unsubsampled data without removing singletons.

```
# Get phyloseq object without singletons by merging original phyloseq objects
merged_phylo_counts_withSingletons <- merge_phyloseq(AllREadsSE_Indo_Counts_physeq, Mali_Counts_physeq,
# Remove Viridiplantae and Metazoa
merged_phylo_counts_withSingletons <- subset_taxa(merged_phylo_counts_withSingletons, (Kingdom!="Viridip
merged_phylo_counts_withSingletons <- subset_taxa(merged_phylo_counts_withSingletons, (Kingdom!="Metazoa
# remove any empty rows
merged_phylo_counts_withSingletons <- prune_taxa(taxa_sums(merged_phylo_counts_withSingletons) > 0, merg
```

Now that we have data without singletons, we now need to merge our data at a specified taxonomic level. DivNet is computationally expensive, and therefore a higher level is much, much faster. We'll therefore test how our groups look like at the Phylum level. Then, we'll run DivNet without specifying any hypothesis testing.

```
# comparing diversity at the phylum level
pop_comparison <- merged_phylo_counts_withSingletons %>%
  tax_glom("Phylum")

# If we don't change the sample names here from hyphens to periods, we'll get an error later
sample_names(pop_comparison) <- gsub("\\-", ".", sample_names(pop_comparison))

# Run divnet without specifying any hypothesis testing
dv_pop_comparison <- divnet(pop_comparison, ncores = 4)
```

```
##   |                                                                               |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing
##   |                                                                               |==================
##   |                                                                               |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================
##   |                                                                               |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================
## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
```

```r
# DivNet outputs a list of the estimates shannon, simpson (alpha diversity)
# bray-curtis, euclidean (beta diversity)
dv_pop_comparison %>% names
```

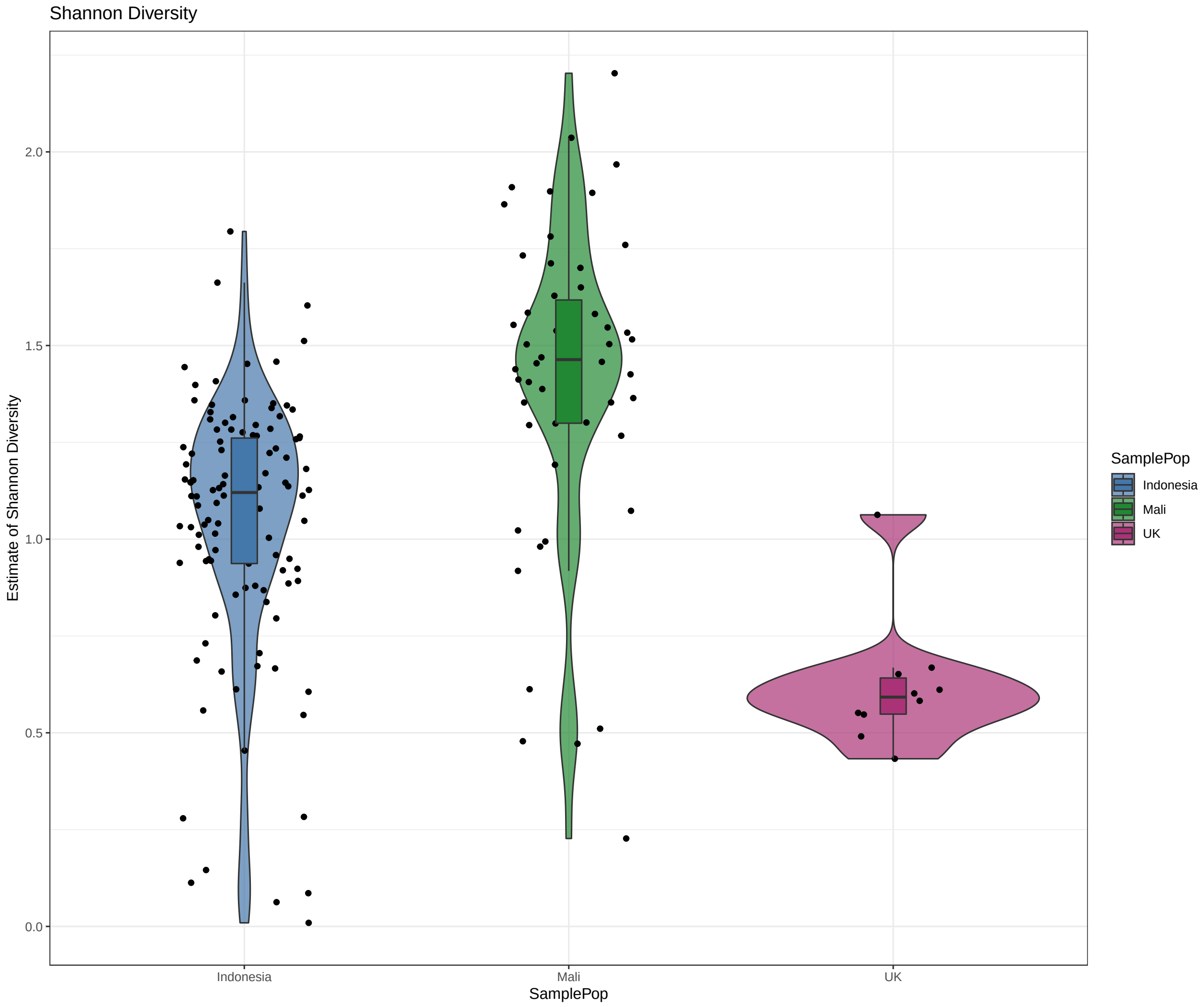
```
##  [1] "shannon"               "simpson"               "bray-curtis"
##  [4] "euclidean"             "shannon-variance"      "simpson-variance"
##  [7] "bray-curtis-variance" "euclidean-variance"    "X"
## [10] "fitted_z"
```

DivNet will output an object with estimates for multiple different alpha (and beta) diversity measures (we'll get to the beta diversity estimates later). The Shannon and Simpson index are two popular alpha diversity indices to measure species richness. For Shannon diversity, the importance of rare taxa are downeighted, since they do not play a large role in the community or they could potentially be due to error. For this reason, the Shannon index is one of the most popular alpha diversity metrics.

To interpret the index, a higher Shannon index means higher diversity, whereas a lowed index number means lower diversity.

Let's take out the Shannon diversity metric from DivNet and plot it.

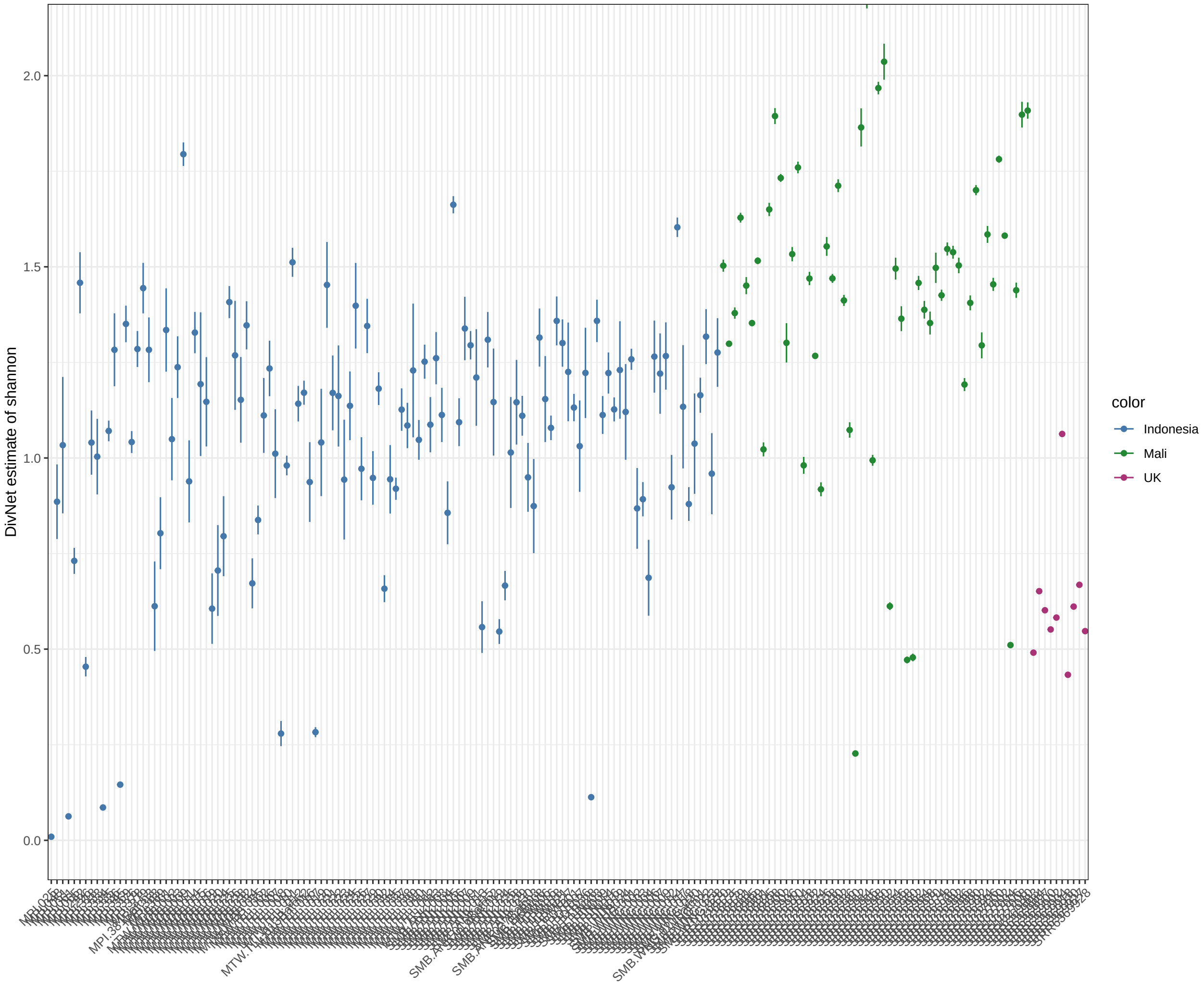
```r
# Now let's plot the results of shannon and Simpson diversity
summary_df_shannon <- as.data.frame(dv_pop_comparison$shannon %>%
  summary %>%
  add_column("SampleNames" = pop_comparison %>% otu_table %>% sample_names) %>%
  add_column("SamplePop" =  pop_comparison %>% sample_data %>% .[,"SamplePop"] %>% as.matrix(.) %>% .[,

ggplot(summary_df_shannon, aes(y = estimate, x = SamplePop, fill = SamplePop)) + geom_violin(alpha=0.7)
  geom_jitter(height = 0, width = .2) + geom_boxplot(width=0.08, outlier.color = NA) +
  scale_fill_manual(values=c(IndonesiaCol,MaliCol,UKCol)) + ggtitle("Shannon Diversity") +
  ylab("Estimate of Shannon Diversity")
```



We can see that the Malian samples, on average, have the highest estimates of Shannon alpha diversity, followed by the Indonesian population, then the UK population.

We can also plot each individual sample, along with their standard deviation (another cool, and imprortant feature that DivNet calculates and uses in their hypothesis testing).

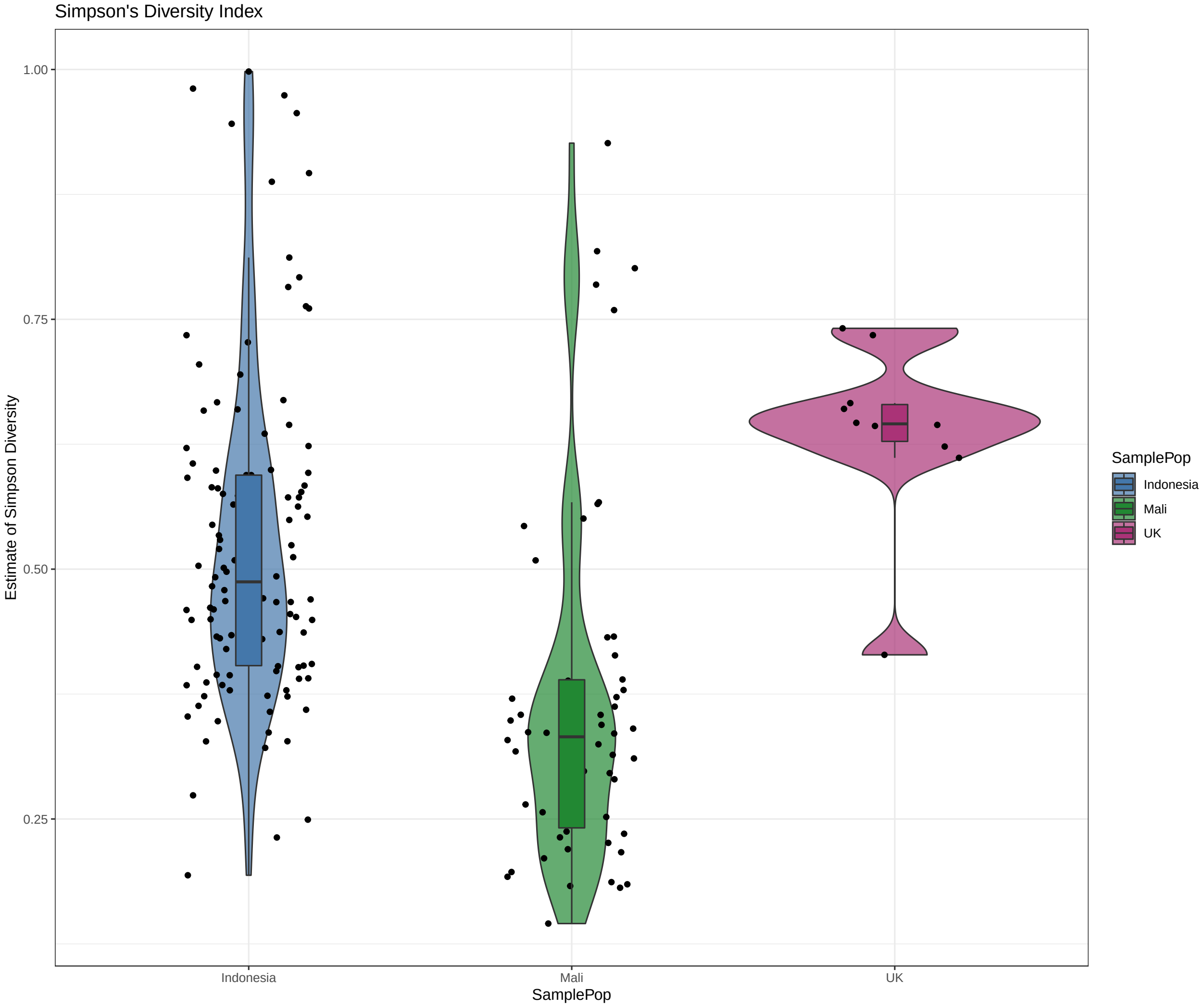
```
plot(dv_pop_comparison$shannon, pop_comparison, col = "SamplePop") + scale_colour_manual(values=c(Indone
```



We can see that the Indonesian samples have the highest amount of SE around their estimates, but this makes sense given that this is the population with the lowest library size.

Now let's use see how the population diversity looks like when we use the Simpson diversity. The Simpson diversity index is a similarity index where the higher the value, the lower the diversity. It measures the probability that two individuals randomly selected from a sample will belong to the same species. With this index, 0 represents infinite diversity and 1, no diversity.

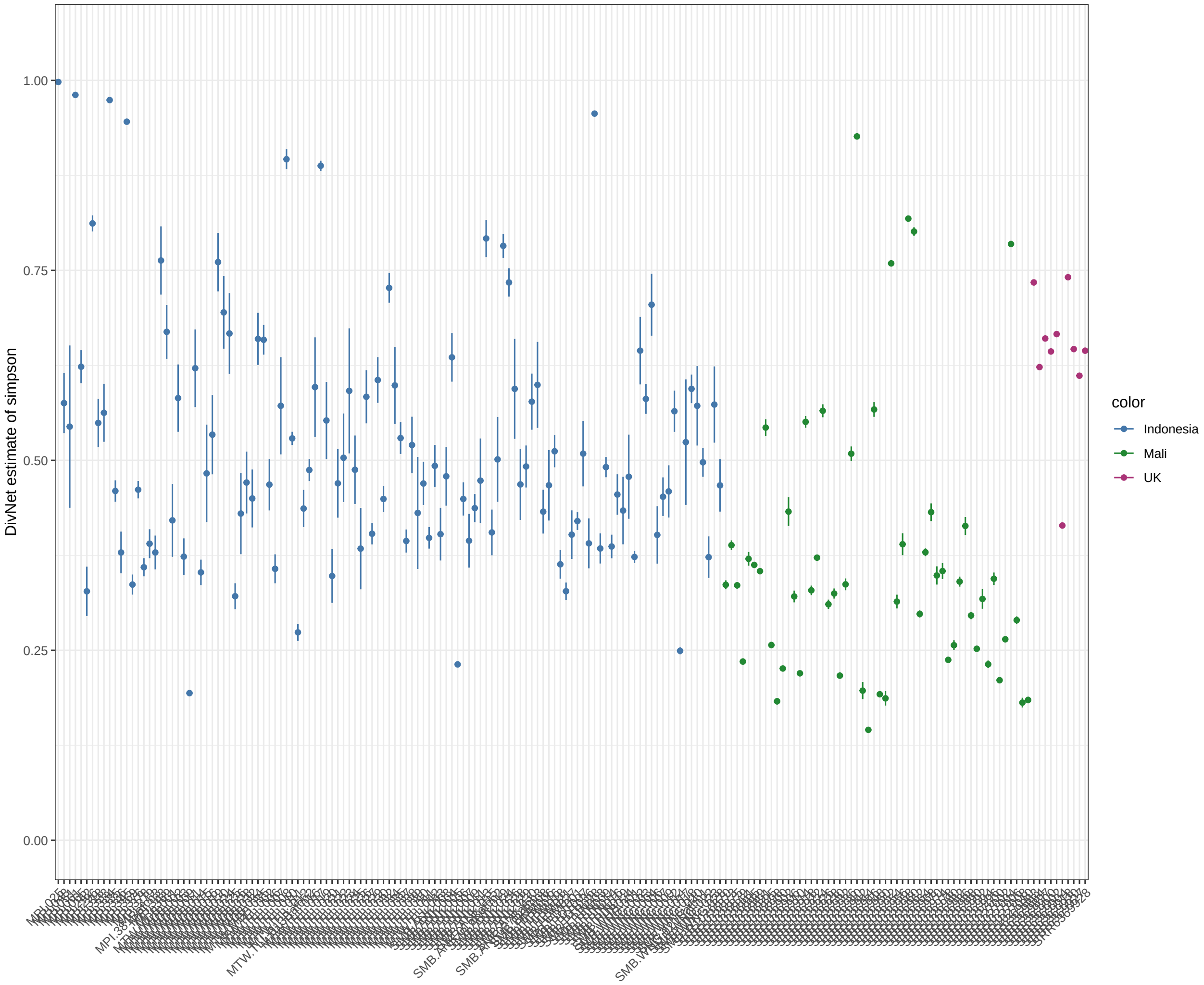
```
summary_df_simpson <- as.data.frame(dv_pop_comparison$simpson %>%
  summary %>%
  add_column("SampleNames" = pop_comparison %>% otu_table %>% sample_names) %>%
  add_column("SamplePop" =  pop_comparison %>% sample_data %>% .[,"SamplePop"] %>% as.matrix(.) %>% .[,

ggplot(summary_df_simpson, aes(y = estimate, x = SamplePop, fill = SamplePop)) + geom_violin(alpha=0.7)
  geom_jitter(height = 0, width = .2) + geom_boxplot(width=0.08, outlier.color = NA) +
  scale_fill_manual(values=c(IndonesiaCol,MaliCol,UKCol)) + ggtitle("Simpson's Diversity Index") +
  ylab("Estimate of Simpson Diversity")
```



Again, we can see that the Malian population has the highest diversity (remember, and index of 0 equates to infinite diversity), while the UK population has the lowest diversity.

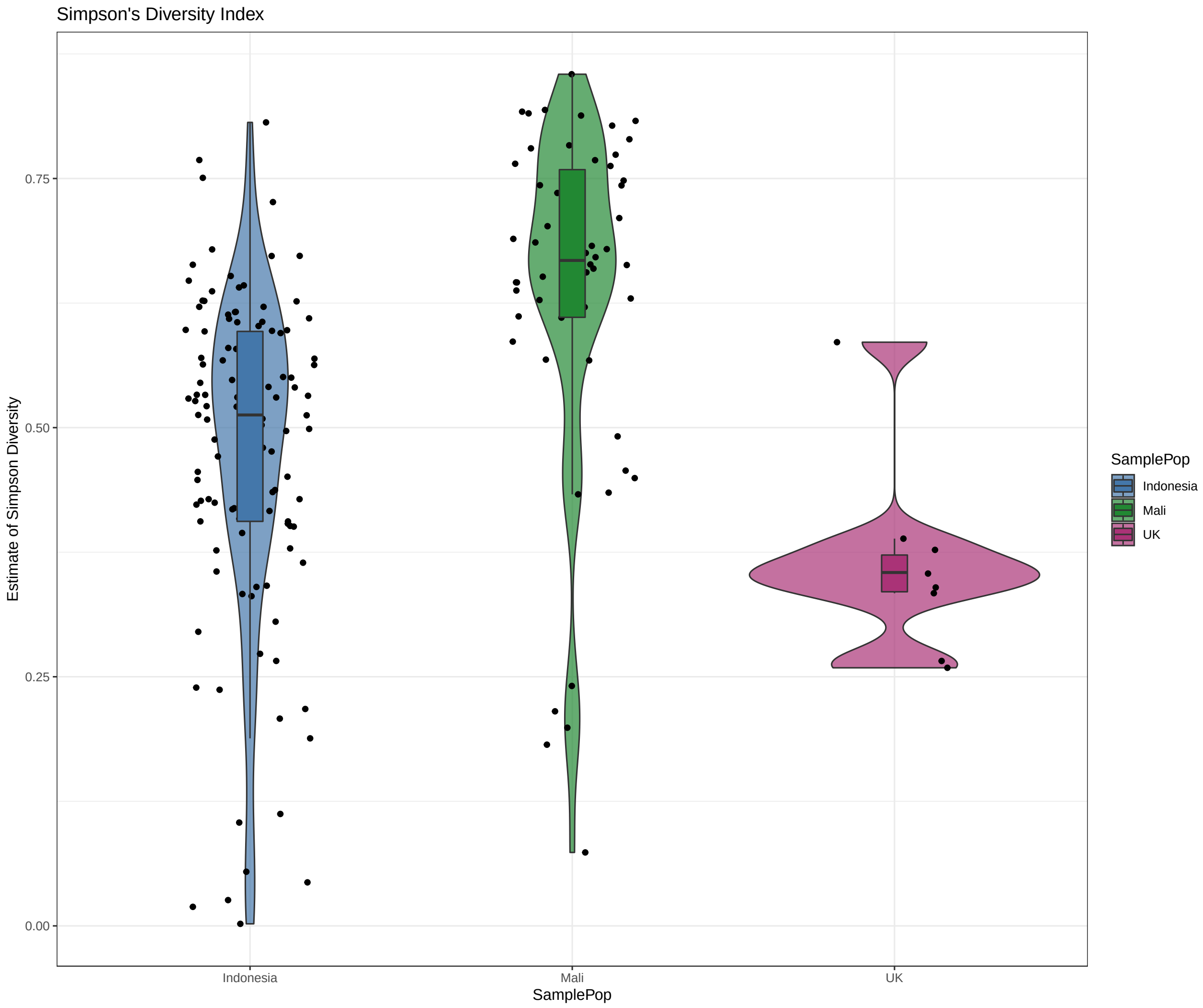
This is how the diversity looks like with SE included for each sample.

```
plot(dv_pop_comparison$simpson, pop_comparison, col = "SamplePop") + scale_colour_manual(values=c(Indone
```



Since a larger Simpson index value equates to a lower diversity index, many people find this confusing and not very intuitive. Therefore, the inverse Simpsone Index, or 1 - Simpson Index, is also commonly used. Let's plot that now.

```
# Subtract the Simpson estimate from one
summary_df_simpson$estimate = 1-summary_df_simpson$estimate
# Plot
ggplot(summary_df_simpson, aes(y = estimate, x = SamplePop, fill = SamplePop)) + geom_violin(alpha=0.7)
  geom_jitter(height = 0, width = .2) + geom_boxplot(width=0.08, outlier.color = NA) +
  scale_fill_manual(values=c(IndonesiaCol,MaliCol,UKCol)) + ggtitle("Simpson's Diversity Index") +
  ylab("Estimate of Simpson Diversity")
```



Now let's test the hypothesis that the diversity is different between islands, which is really what we're asking - are the taxa communities between our groups different? This is holding the assumption that the populations are part of a shared ecosystem that has similarities in, for example, pathogen load. We are now estimating the diversity of island/population being an ecosystem, so we're focusing on the ecosystem, not just the samples. If we wanted to reproduce this result, it is better to focus on the populations that the samples come from, not just the samples themselves.

Let's test this first using the Shannon diversity index.

```r
# test the hypothesis that the diversity is differnet between islands
dv_pop_comparison_cov <- pop_comparison %>%
  divnet(X = "SamplePop", ncores = 8)
```

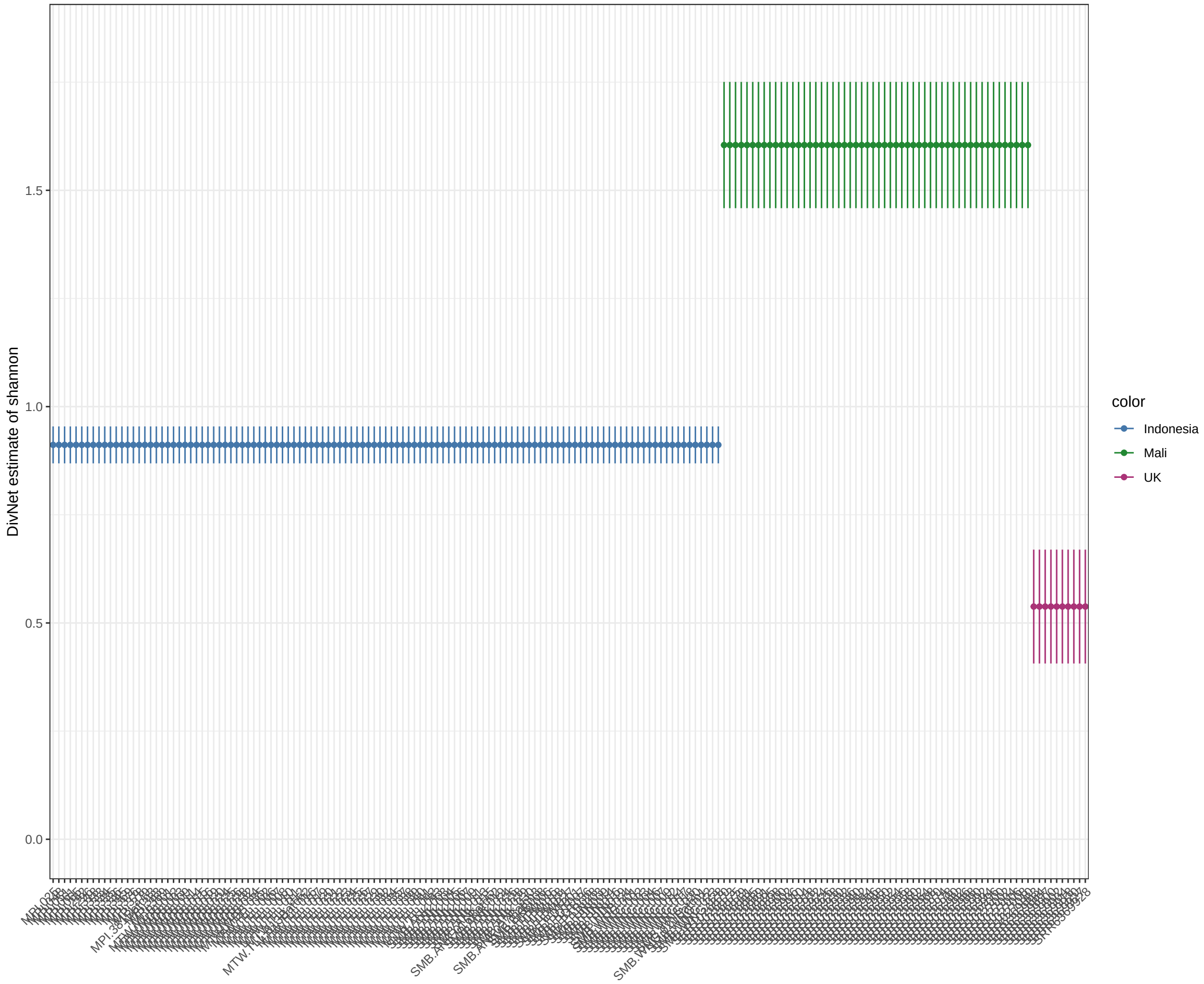
```
##   |                                                                              |
```

```
## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing
```

```
##   |                                                                              |============
```

```
## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================
##   |                                                                               |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                               |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing
##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================
##   |                                                                              |

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |============

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                              |==================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                      |===================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                      |===================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                      |===================

## Warning in MCmat(Y = Y_p, W = W, eY = eY, N = N, Q = Q, base = base, sigma =
## sigma, : Running in series; one of the packages doParallel, foreach or doSNOW is
## missing

##   |                                                                      |===================
```

```r
# Plot the results for each individual
plot(dv_pop_comparison_cov$shannon, pop_comparison, col = "SamplePop") + scale_colour_manual(values=c(I
```



We can see that, as a population, the Malian samples have the highest diversity, followed by the Indonesian samples, then the UK samples.

Let's test this hypothesis formally.

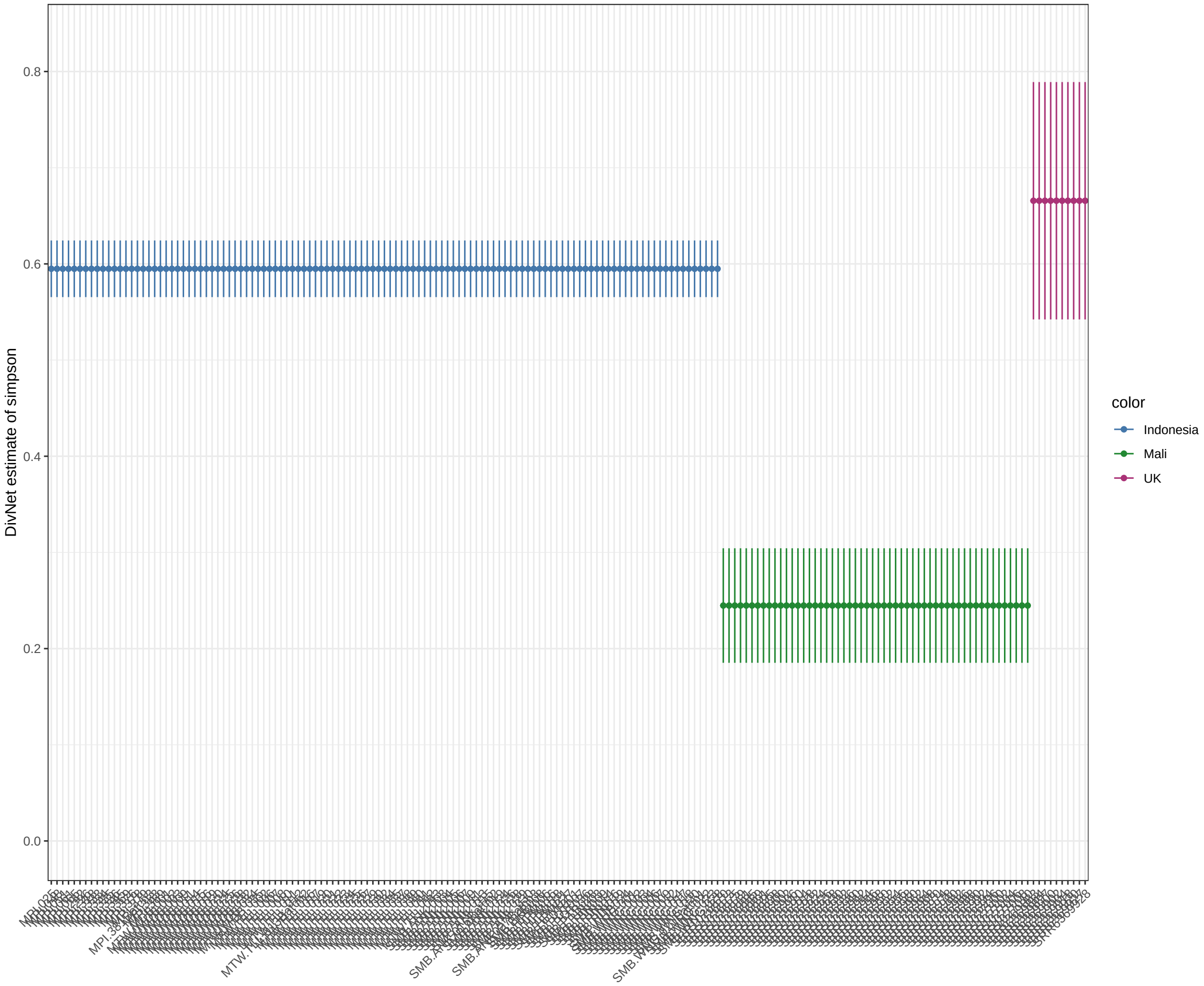
```r
# test that these populations are actually different
testDiversity(dv_pop_comparison_cov, "shannon")
```

```
## Hypothesis testing:
##   p-value for global test: 0

##                  Estimates Standard Errors p-values
## (Intercept)      0.9115722     0.001925960        0
## predictorsMali   0.6930572     0.009926869        0
## predictorsUK    -0.3736744     0.020806487        0
```

The result tells us that the mean Shannon diversity in the Indonesian population at the Phylum-level is 0.92, and it is significantly higher by 0.76 orders, on average, in the Malian population. We can also see that the UK population is significantly lower than the Indonesian population by 0.38 orders, on average.

Let's do the same thing for Simpson diversity.

```r
plot(dv_pop_comparison_cov$simpson, pop_comparison, col = "SamplePop") + scale_colour_manual(values=c(I
```



With the Simpson index, the same trend seems to hold at the population-level: the UK population has the highest Simpson index (i.e., lowest diversity), followed by the Indonesian population, then the Malian population.

Again, let's test this formally.

```
testDiversity(dv_pop_comparison_cov, "simpson")
```

```
## Hypothesis testing:
##   p-value for global test: 0

##                  Estimates Standard Errors p-values
## (Intercept)     0.59490753     0.001286851        0
## predictorsMali -0.35013171     0.004050794        0
## predictorsUK    0.07074015     0.019510844        0
```

The result tells us that the mean Simpson diversity index in the Indonesian population at the Phylum-level is 0.59, and it is significantly lower by 0.36 orders, on average, in the Malian population. We can also see that the UK population is significantly higher than the Indonesian population by 0.07 orders, on average.

# Beta Diversity

Beta diversity is a measure of dissimilarity metric between samples to compare differneces in species composition. It's helpful to know not only how taxonomically/pathogenically rich each sample is, but also to see differences in samples and populations.

There are multiple beta diversity measures to use, including Bray-curtis dissimilarity (based on abundances), Jaccard distance (based on presence or absence), Euclidean distance, and Unifrac (based on sequence distances using a phylogenetic tree).

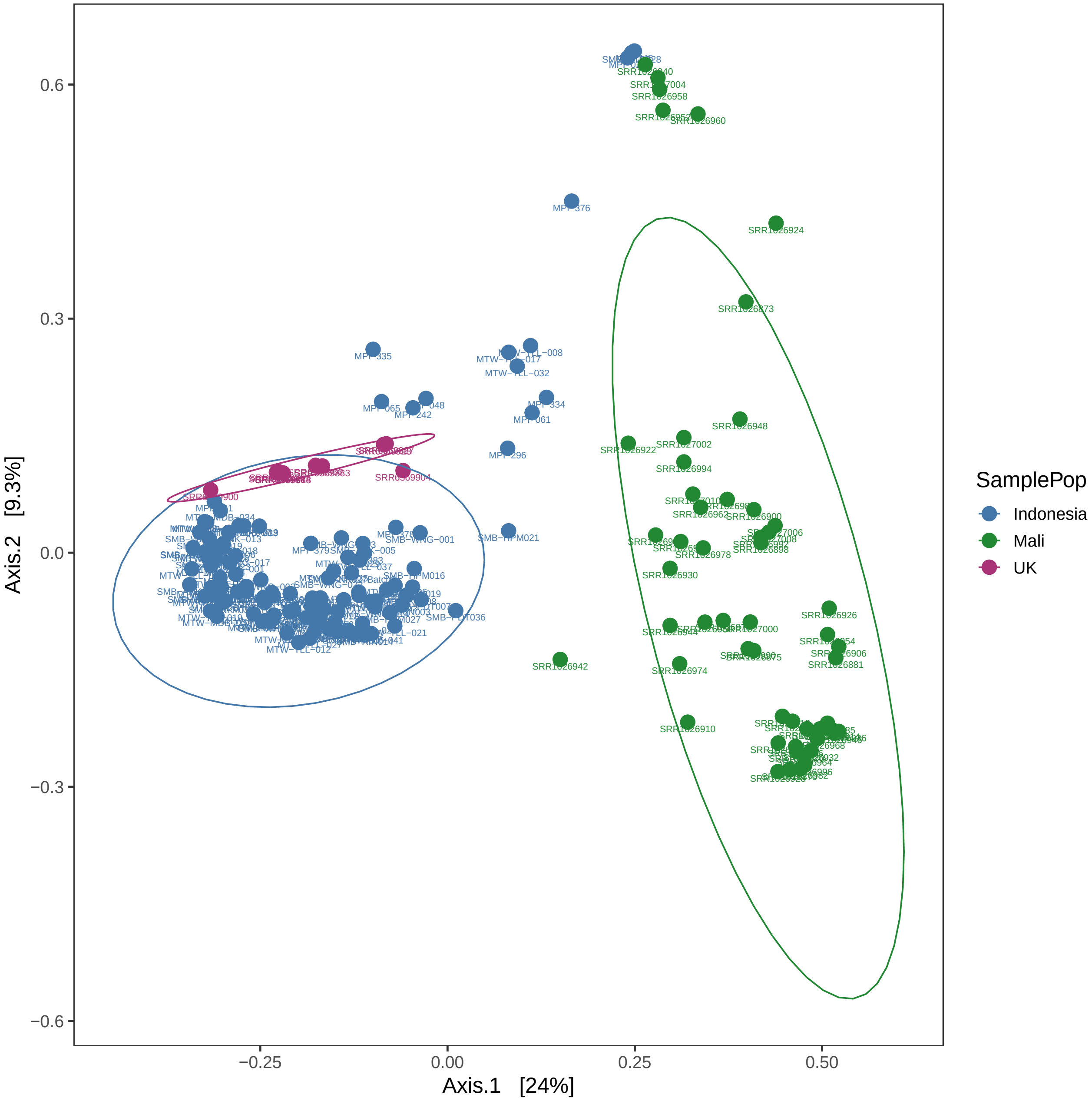
The Bray–Curtis dissimilarity metric is probably the most popular beta diversity metric and is bounded between 0 and 1, where 0 means the two sites have the same composition (all species are shared), and 1 means the two sites do not share any species.

Unlike alpha diversity, beta diversity is not as sensitive to singletons, and it has even been suggested that 'using simple proportions' (i.e., relative abundance) on non-rarefied data is fine. I haven't found enough information on this one way or another (it seems like people are far more opinionated on alpha diversity than beta diversity), but regardless, let's try it out a few different methods. First we'll try out Bray-curtis dissimilarity and Jaccard distances on the relative abundances using phyloseq, then we'll use the Bray-curtis dissimilarity and Euclidean beta diversity metrics from DivNet (these are the only two that are available in DivNet).

First, let's test the Phyloseq method using Bray-curtis dissimilarity and Jaccard.

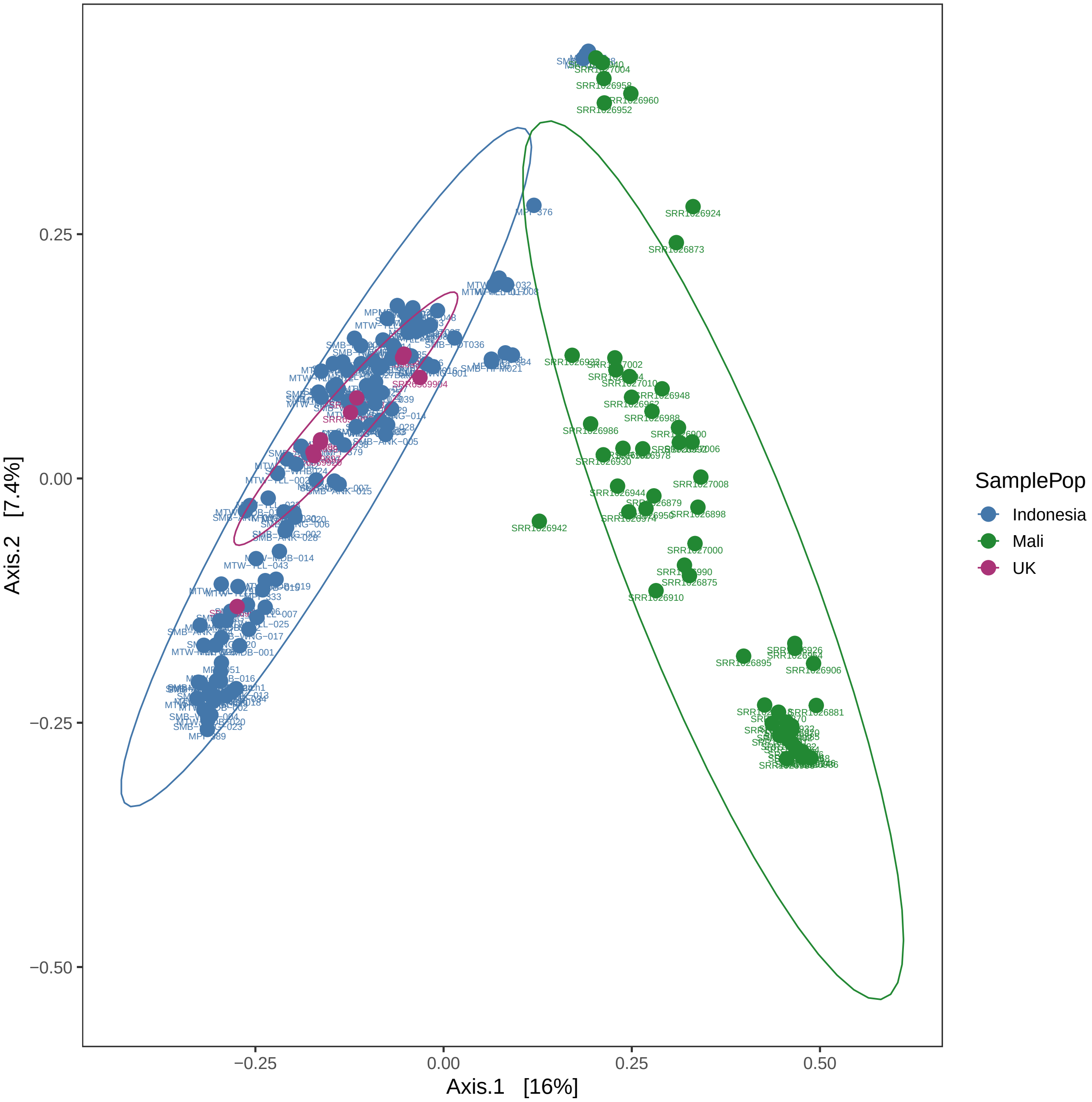
```
# Change counts to relative abundances
ps4.rel <- microbiome::transform(merged_phylo_counts_withSingletons, "compositional")
# calculat Bray-curtis dissimilarity
bx.ord_pcoa_bray <- ordinate(ps4.rel, "PCoA", "bray")

# Make an ordination plot using bray's dissimilarity metric
beta.ps1 <- plot_ordination(ps4.rel,
                            bx.ord_pcoa_bray,
                            color="SamplePop",
                            label = "SampleName") +
  geom_point(aes(), size= 4) +
  theme(plot.title = element_text(hjust = 0, size = 12))

# add in an ellipse
```

```
beta.ps1 + stat_ellipse() + theme_bw(base_size = 14) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  scale_colour_manual(values=c(IndonesiaCol,MaliCol,UKCol))
```



```
# Change counts to relative abundances
ps4.rel <- microbiome::transform(merged_phylo_counts_withSingletons, "compositional")
# calculat Bray-curtis dissimilarity
bx.ord_pcoa_jaccard <- ordinate(ps4.rel, "PCoA", "jaccard")

# Make an ordination plot using bray's dissimilarity metric
beta.ps1 <- plot_ordination(ps4.rel,
                            bx.ord_pcoa_jaccard,
                            color="SamplePop",
                                    label = "SampleName") +
  geom_point(aes(), size= 4) +
  theme(plot.title = element_text(hjust = 0, size = 12))

# add in an ellipse
beta.ps1 + stat_ellipse() + theme_bw(base_size = 14) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  scale_colour_manual(values=c(IndonesiaCol,MaliCol,UKCol))
```

```
## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure
```



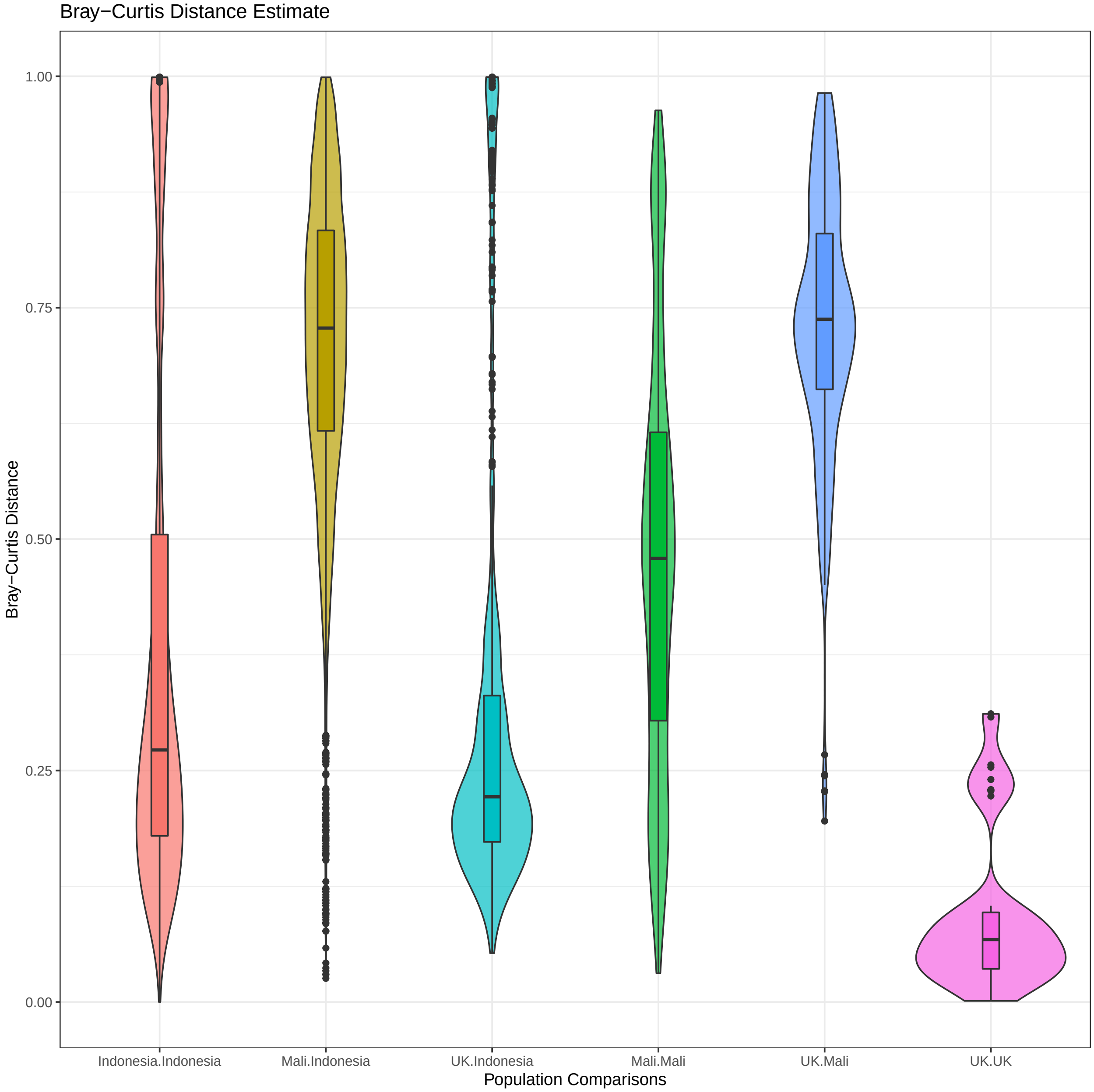
For both estimates, we can see that the Malian data is the least similar from the Indonesian and UK data. We can also see that some samples, particularly those with higher Plasmodium load, cluster closer to the

Malian samples.

Now let's use our results from DivNet, which corrects for sequencing depth, and see how this compares.

For the first bray-curtis dissimilarity analysis, we'll look at all of our samples individually, then we'll look at how bray-curtis dissimilarity looks like between islands (with hypothesis testing).

```r
# First, let's look at Bray-curtis dissimilarity at the individual sample level
bray_est <- simplifyBeta(dv_pop_comparison, pop_comparison, "bray-curtis", "SamplePop")

# add in group comparisons and plot
bray_est$group=paste(bray_est$Covar1,bray_est$Covar2,sep="_")
ggplot(bray_est, aes(x = interaction(Covar1, Covar2), y = beta_est, fill=group)) +
  geom_violin(alpha=0.7) + geom_boxplot(width=0.1) + xlab("Population Comparisons") +
  theme(legend.position="none") + ggtitle("Bray-Curtis Distance Estimate") +
  ylab("Bray-Curtis Distance")
```

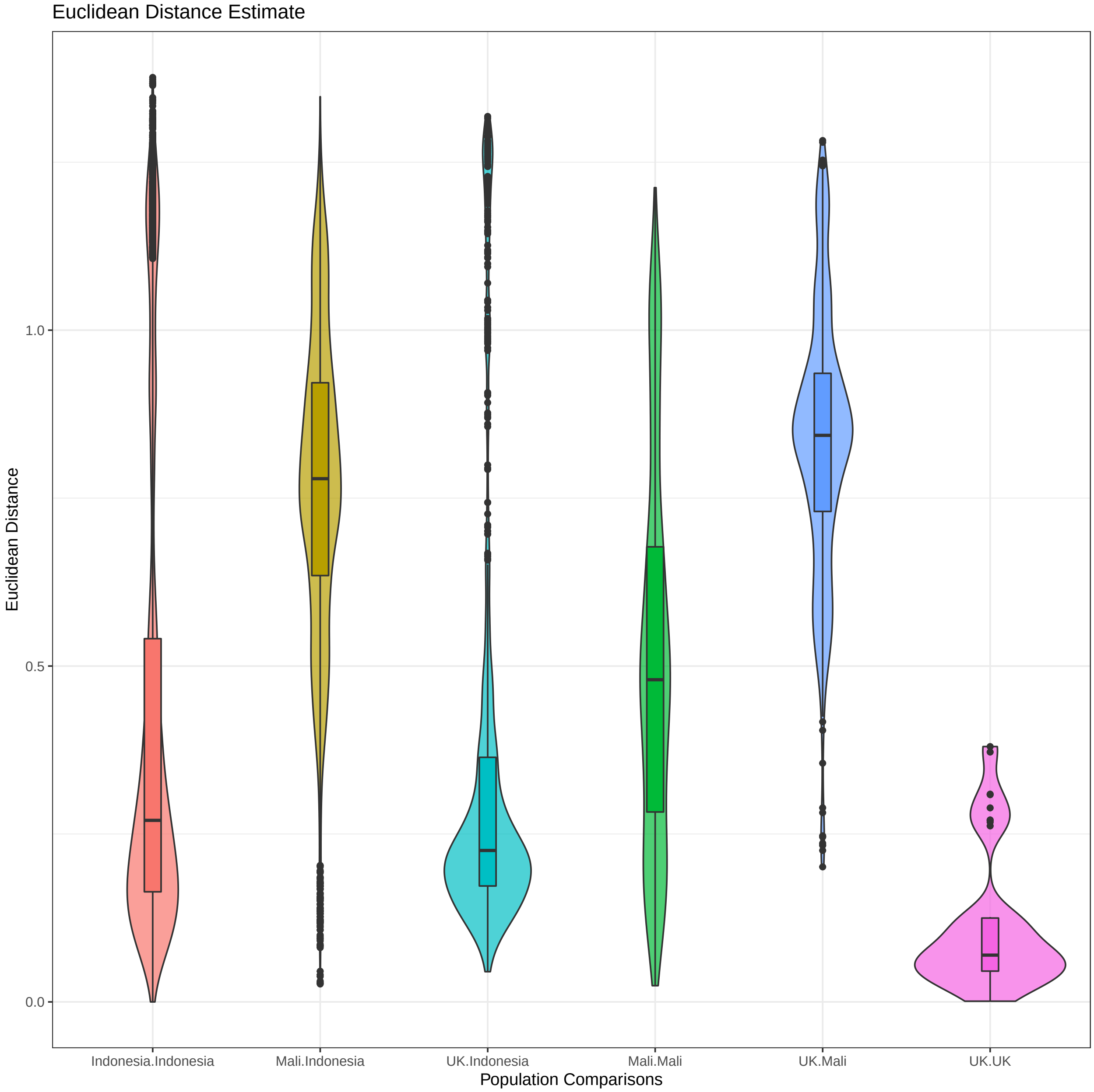


From DivNet, again we can see that the greatest dissimilarity is between the Malian population. Unsurpiringly, the least dissimilar samples are comparing UK samples to UK samples and Indonesian samples to Indonesian samples, however this isn't the case for the Malian samples - we can see that the mean Malian Bray-Curtis dissimilarity estimate is nearly 0.5, which is even higher than comparing UK samples to Indonesian samples. We also see quite a but of spread in distance estimates within Indonesia, but in the UK, they seem to be relatively similar to one another.

Now let's test beta diversity in DivNet using Euclidean distance.

```
# First, let's look at Bray-curtis dissimilarity at the individual sample level
bray_est_eucl <- simplifyBeta(dv_pop_comparison, pop_comparison, "euclidean", "SamplePop")

# add in group comparisons and plot
bray_est_eucl$group <- paste(bray_est_eucl$Covar1, bray_est_eucl$Covar2,sep="_")
```

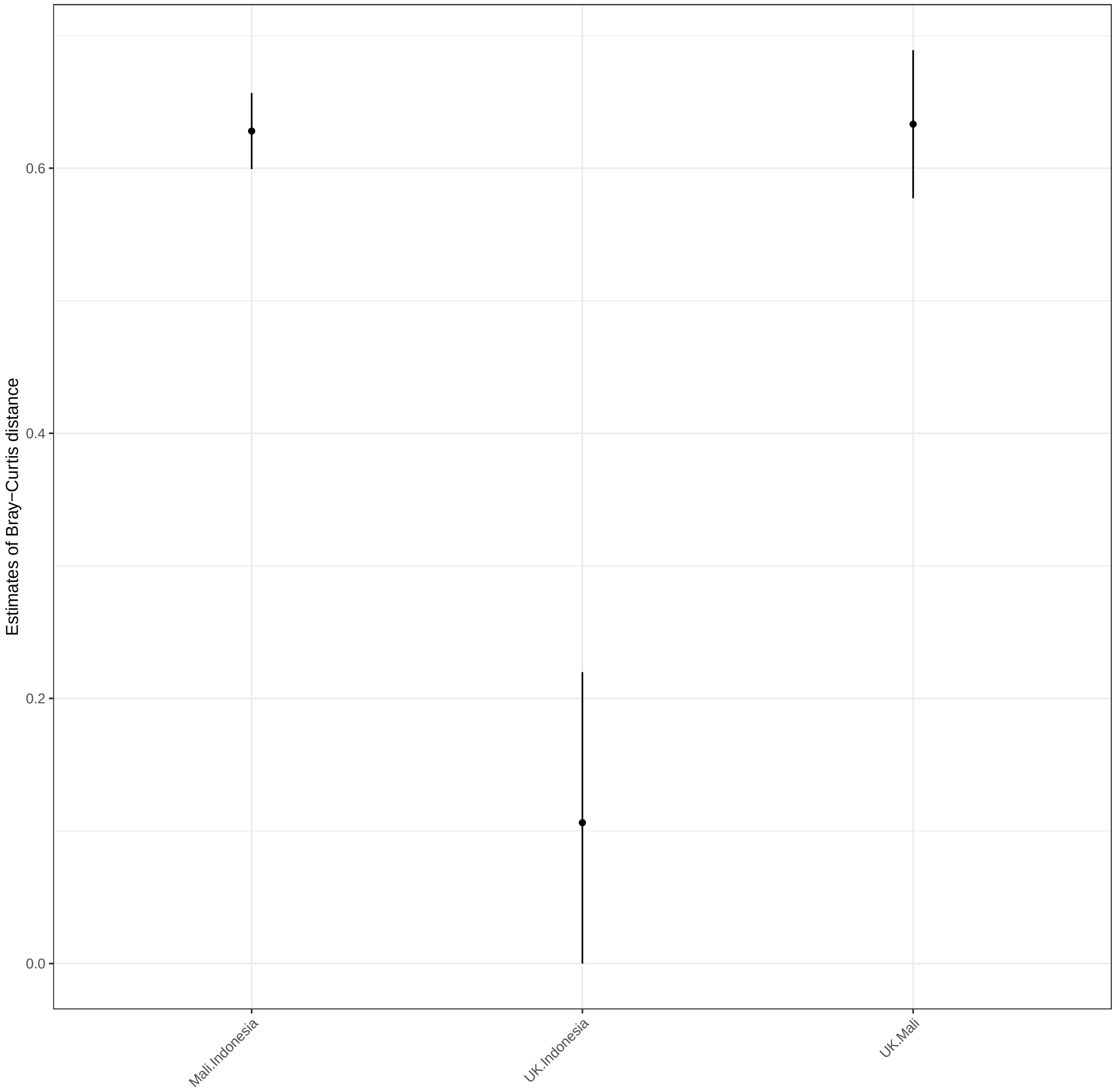
```
ggplot(bray_est_eucl, aes(x = interaction(Covar1, Covar2), y = beta_est, fill=group)) +
  geom_violin(alpha=0.7) + geom_boxplot(width=0.1) + xlab("Population Comparisons") +
  theme(legend.position="none") + ggtitle("Euclidean Distance Estimate") +
  ylab("Euclidean Distance")
```



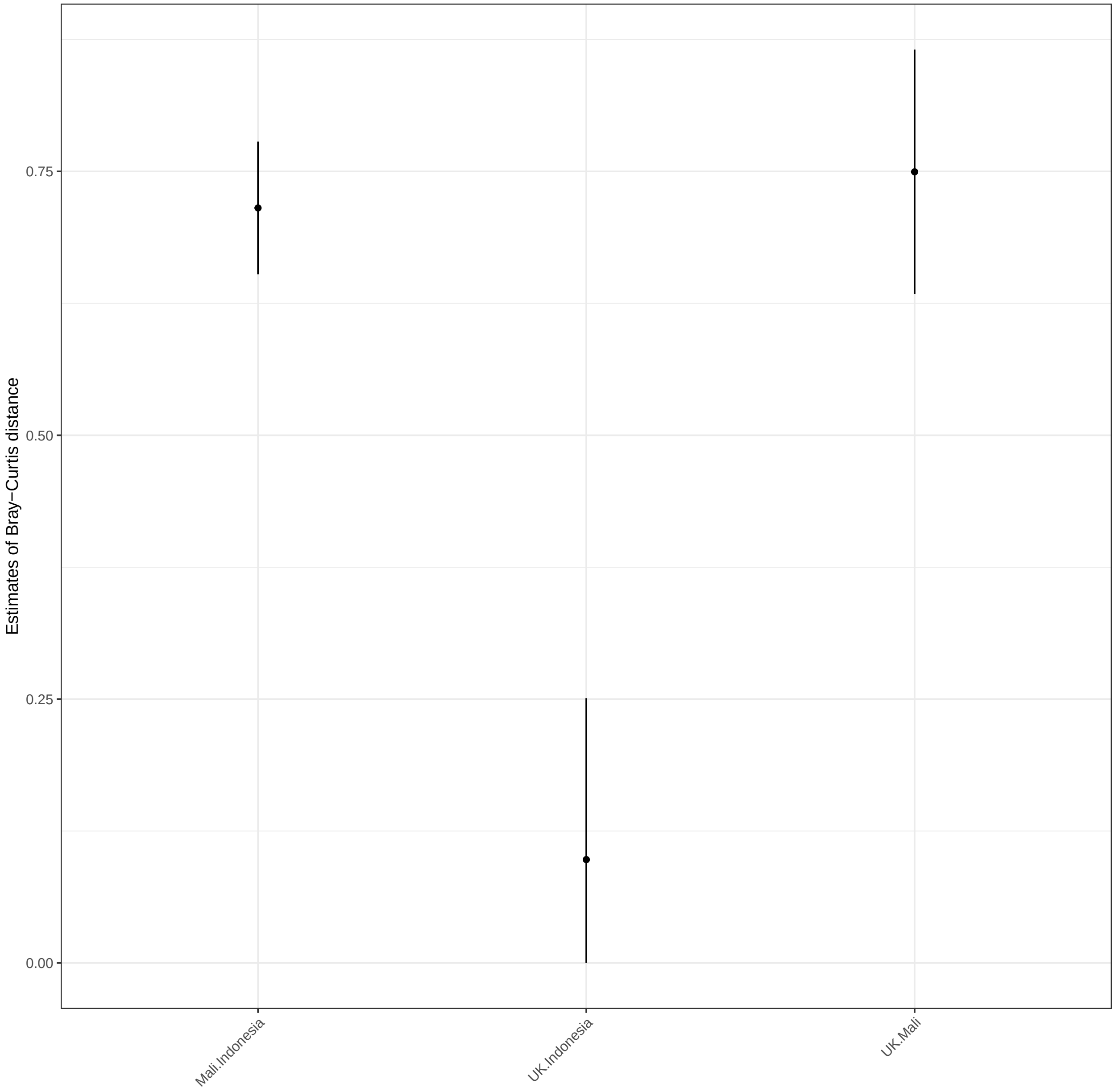
We can see similar trends in using Euclidean distance, compared to the Bray-Curtis dissimilarity metric.

Now let's see how this looks like for island-level comparisons.

```
# Bray-Curtis dissimilarity
simplifyBeta(dv_pop_comparison_cov, pop_comparison, "bray-curtis", "SamplePop") %>%
  ggplot(aes(x = interaction(Covar1, Covar2),
             y = beta_est)) +
  geom_point() +
  geom_linerange(aes(ymin = lower, ymax = upper)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  xlab("") + ylab("Estimates of Bray-Curtis distance")
```



```
# Euclidean distance
simplifyBeta(dv_pop_comparison_cov, pop_comparison, "euclidean", "SamplePop") %>%
  ggplot(aes(x = interaction(Covar1, Covar2),
             y = beta_est)) +
  geom_point() +
  geom_linerange(aes(ymin = lower, ymax = upper)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  xlab("") + ylab("Estimates of Bray-Curtis distance")
```



This confirms that indeeed, the Malian population is the most dissimilar from the other two populations. Thanks for sticking through to the end!