iii) 16S Diversity Analysis

Laura Mary Pugh

27/11/2019

The following pipeline was used to assess the OTU data that was output from the mothur 16S taxonomic classification.

Importing data into R

The data output from mothur is not in the ideal format to work with metacoder, the primary package used in this analysis so first it needed to be formatted. To import the data into R the line of code below was used by specifying the name of the '.shared' file which contains the abundance counts for the OTUs.

```
otu_data <- read_tsv("16S.shared")
```

Two of the columns in the table contain no useful information, so these were removed using the first two lines of code. The data table also needed to be transposed so columns became rows etc. The 3rd line did this. The 4th and 5th lines set the contents of the first row to be the column headers, and removed it. The 6th line ensured that the count data was recognised as being numeric information rather than characters.

```
otu_data <- otu_data[,-1]
otu_data <- otu_data[,-2]
otu_data <- t(otu_data)
colnames(otu_data) = otu_data[1, ]
otu_data <- otu_data[-1,]
class(otu_data)<-"numeric"
```

The next lines converted the OTU data table to a tibble.

```
otu_data<- as_tibble(otu_data, validate=NULL,.name_repair = NULL,
                     rownames = pkgconfig::get_config("tibble::rownames", NA))
otu_data<- otu_data %>% rownames_to_column("OTU_ID")
```

Then I specifed the name of the taxonomy file output by mothur.

```
tax_data <- read_tsv("16S.taxonomy")
print(tax_data)
```

```
## # A tibble: 100,484 x 3
##    OTU          Size Taxonomy
##    <chr>       <dbl> <chr>
##  1 Otu000001 156689 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  2 Otu000002 118564 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  3 Otu000003 111885 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  4 Otu000004  80149 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  5 Otu000005  77912 Bacteria(100);Bacteroidetes(100);Bacteroidia(100);Flavobact...
##  6 Otu000006  52500 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  7 Otu000007  52198 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
##  8 Otu000008  40795 Bacteria(100);Actinobacteria(100);Actinobacteria(100);Micro...
##  9 Otu000009  40479 Bacteria(100);Proteobacteria(100);Gammaproteobacteria(100);...
## 10 Otu000010  38806 Bacteria(100);Bacteroidetes(100);Bacteroidia(100);Sphingoba...
## # ... with 100,474 more rows
```

The 'Taxonomy' column in this table contained the taxonomy information for each OTU. The different taxonomic ranks were separated by a semicolon. The line below split this column where there was a semicolon into multiple columns.

```
tax_data<-separate(tax_data, col=3, remove=FALSE, sep = "[;]",
                   c("Kingdom","Phylum","Class","Order","Family","Genus"))
```

Each rank classification was followed by a number in (). This number is the % of the sequences in that OTU which were of the rank shown. The following lines of code split that information away from the classification into separate columns, then reordered the columns so the rank columns were all together which mades things easier later on.

```
tax_data<-separate(tax_data, col=3, remove=FALSE, sep = "[\\(|\\)]",
                   c("Kingdom","Kingdom_conf"))
tax_data<-separate(tax_data, col=6, remove=FALSE, sep = "[\\(|\\)]",
                   c("Phylum","Phylum_conf"))
tax_data<-separate(tax_data, col=6, remove=FALSE, sep = "[\\(|\\)]",
                   c("Class","Class_conf"))
tax_data<-separate(tax_data, col=10, remove=FALSE, sep = "[\\(|\\)]",
                   c("Order","Order_conf"))
tax_data<-separate(tax_data, col=10, remove=FALSE, sep = "[\\(|\\)]",
                   c("Family","Family_conf"))
tax_data<-separate(tax_data, col=14, remove=FALSE, sep = "[\\(|\\)]",
                   c("Genus","Genus_conf"))
tax_data<-tax_data[,c(1,2,3,4,6,7,10,11,14,5,8,9,12,13,15)]
tax_data<- tax_data %>% unite(col="taxonomy", c(4:9), remove=FALSE, sep=";")
print(tax_data)
```

```
## # A tibble: 100,484 x 16
##    OTU     Size Taxonomy taxonomy Kingdom Phylum Class Order Family Genus
##    <chr>  <dbl> <chr>    <chr>    <chr>   <chr>  <chr> <chr> <chr>  <chr>
##  1 Otu0... 156689 Bacteri... Bacteri... Bacter... Prote... Gamm... Ente... Enter... Ente...
##  2 Otu0... 118564 Bacteri... Bacteri... Bacter... Prote... Gamm... Pseu... Pseud... Pseu...
##  3 Otu0... 111885 Bacteri... Bacteri... Bacter... Prote... Gamm... Beta... Burkh... Mass...
##  4 Otu0...  80149 Bacteri... Bacteri... Bacter... Prote... Gamm... Pseu... Pseud... Pseu...
##  5 Otu0...  77912 Bacteri... Bacteri... Bacter... Bacte... Bact... Flav... Flavo... Flav...
##  6 Otu0...  52500 Bacteri... Bacteri... Bacter... Prote... Gamm... Ente... Enter... Yers...
##  7 Otu0...  52198 Bacteri... Bacteri... Bacter... Prote... Gamm... Xant... Xanth... Sten...
##  8 Otu0...  40795 Bacteri... Bacteri... Bacter... Actin... Acti... Micr... Micro... Pseu...
##  9 Otu0...  40479 Bacteri... Bacteri... Bacter... Prote... Gamm... Xant... Rhoda... Rhod...
## 10 Otu0...  38806 Bacteri... Bacteri... Bacter... Bacte... Bact... Sphi... Sphin... Pedo...
## # ... with 100,474 more rows, and 6 more variables: Kingdom_conf <chr>,
## #   Class_conf <chr>, Phylum_conf <chr>, Family_conf <chr>, Order_conf <chr>,
## #   Genus_conf <chr>
```

At this point the OTU ID's were all prefixed by 'Otu' - e.g. 'Otu000014'. The next lines of code simplified this and combined the two tables together by the OTU_ID column.

```
tax_data$OTU <- sub(tax_data$OTU, pattern = "Otu", replacement="")
otu_data$OTU_ID<-sub(otu_data$OTU_ID, pattern="Otu",replacement="")

tax_data$OTU <- as.character(tax_data$OTU)
otu_data$OTU_ID <- as.character(otu_data$OTU_ID)
otu_data <- left_join(otu_data, tax_data, by=c("OTU_ID" = "OTU"))
```

The metadata table of all the samples was already formatted nicely and was just read in like so.

```r
sample_data <- read_tsv("sample_data.txt")
```

To find out which samples match certain conditions, the line below could be used. It returns which samples are from the Durie site, are the Laureate genotype, flowering growth stage which were from the Plough treatment group.

```r
print(filter(sample_data, Site=="Durie",
             Genotype=="Laureate",
             Growth_Stage=="flowering",
             Treatment=="Plough"))
```

```
## # A tibble: 3 x 10
##   Sample_ID type  Site  Plant Genotype Growth_Stage Treatment Bed   Plot
##   <chr>     <chr> <chr> <chr> <chr>    <chr>        <chr>     <chr> <chr>
## 1 JM19      rhiz... Durie Spri... Laureate flowering    Plough    8 & 9 6 & ...
## 2 JM22      rhiz... Durie Spri... Laureate flowering    Plough    11 &... 5 & ...
## 3 JM28      rhiz... Durie Spri... Laureate flowering    Plough    26 &... 9
## # ... with 1 more variable: Sample_Date <chr>
```

From the counts of sequences per sample it could be seen that samples JM5, JM13 and JM29 had very low counts. These needed to be removed from the metadata table: this line of code was also used to remove outliers like sample JM19.

```r
sample_data <- filter(sample_data, Sample_ID !="JM5")
sample_data <- filter(sample_data, Sample_ID !="JM13")
sample_data <- filter(sample_data, Sample_ID !="JM19")
sample_data <- filter(sample_data, Sample_ID !="JM29")
```

At this point the sample data table contained information on all of the samples in the study: the barley, pea, strawberry (not part of this study), and control samples. The line below filtered the samples down to only those which were from the 'Durie' site, but it can be altered to subsample to any experimental group of samples.

```r
sample_data <- filter(sample_data, Site=="Durie")
```

To save these files (just incase) the following lines of code could be used.

```r
#save(otu_data, file="durie_otu_data.RData")
#save(sample_data, file="durie_data.RData")
```

Creating a taxmap object

Converting data to 'taxmap' format: The lines below created a 'taxmap' object from the otu_data tibble using the taxa package. It used the individual taxonomic rank columns created earlier. These columns differ depending on how many samples were being looked at, but it was pretty easy to work out how many there should be by counting the number of samples which the first line does.

```r
n<-nrow(sample_data)
obj <- parse_tax_data(otu_data, class_cols=c((n+5):(n+10)), named_by_rank = TRUE)
names(obj$data) <- "otu_counts"
obj$data$otu_counts <- obj$data$otu_counts[c("taxon_id", "OTU_ID", sample_data$Sample_ID)]
```

Since the taxonomy file was created using all the samples together, there was a chance that some might not be present in the subset being looked at after it was filtered to just the Durie site samples. The following lines of code removed OTUs which had no reads in the samples of interest.

```r
has_no_reads <- rowSums(obj$data$otu_counts[,sample_data$Sample_ID])==0
sum(has_no_reads)
```

```
## [1] 36298
```

```r
obj <- filter_obs(obj, "otu_counts", ! has_no_reads, drop_taxa=TRUE)
```

Some OTUs were found in samples in very small numbers. Counts less than 10 were set to zero to simplify things. OTUs which may now have no reads needed to be removed. The code below did this.

```r
obj$data$otu_counts <- zero_low_counts(obj, "otu_counts", min_count=10, other_cols =TRUE)
no_reads <- rowSums(obj$data$otu_counts[,sample_data$Sample_ID])==0
sum(no_reads)
```

```
## [1] 60281
```

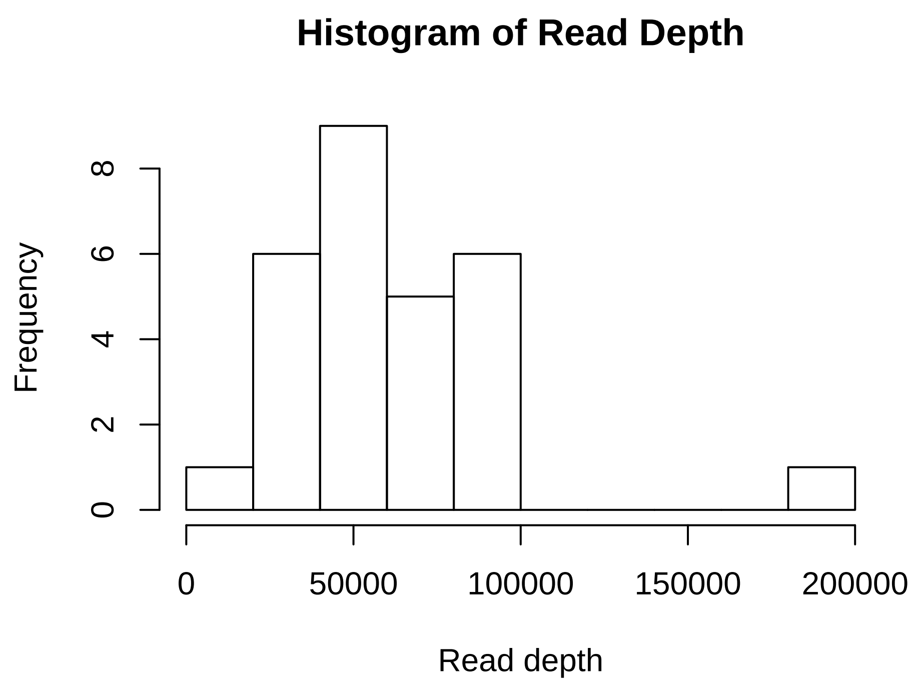
```r
obj <- filter_obs(obj, "otu_counts", ! no_reads, drop_taxa=TRUE)
```

Quality Control

Rarefaction

Rarefaction is used to simulate even numbers of reads between samples. If some samples had very low read depth they could be removed. The lowest samples JM5, JM13 and JM29 had already been removed so the remainder should have been fine, but the following line visualised read depth using a histogram to make sure.

```r
hist(colSums(obj$data$otu_counts[,sample_data$Sample_ID]),
     main="Histogram of Read Depth",xlab="Read depth")
```

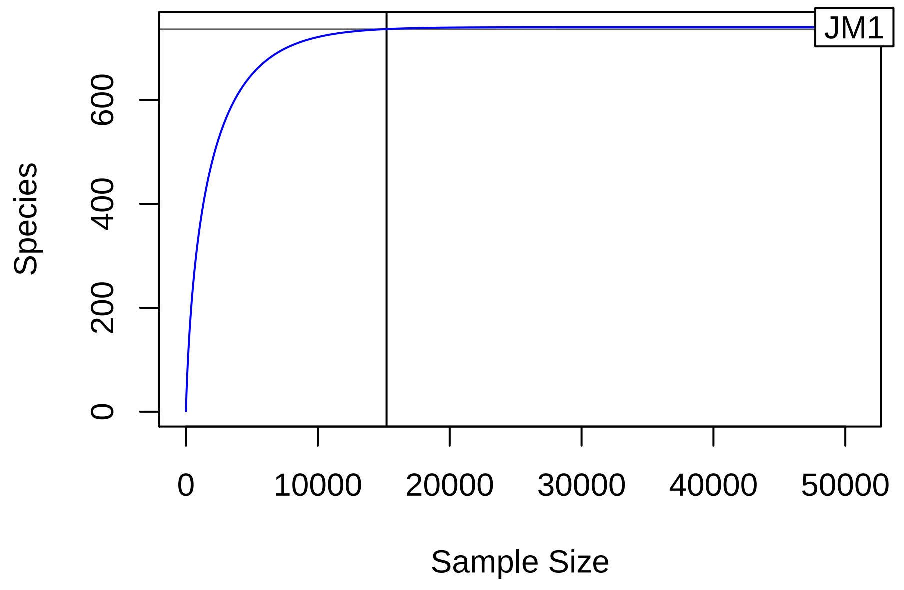


The next lines rareified the samples to the lowest read count. Some OTUs may now have had no counts in them so needed to be removed.

```
obj$data$otu_rarefied <- rarefy_obs(obj, "otu_counts", other_cols=TRUE)
no_reads <- rowSums(obj$data$otu_rarefied[,sample_data$Sample_ID])==0
obj <- filter_obs(obj, "otu_rarefied",! no_reads)
```

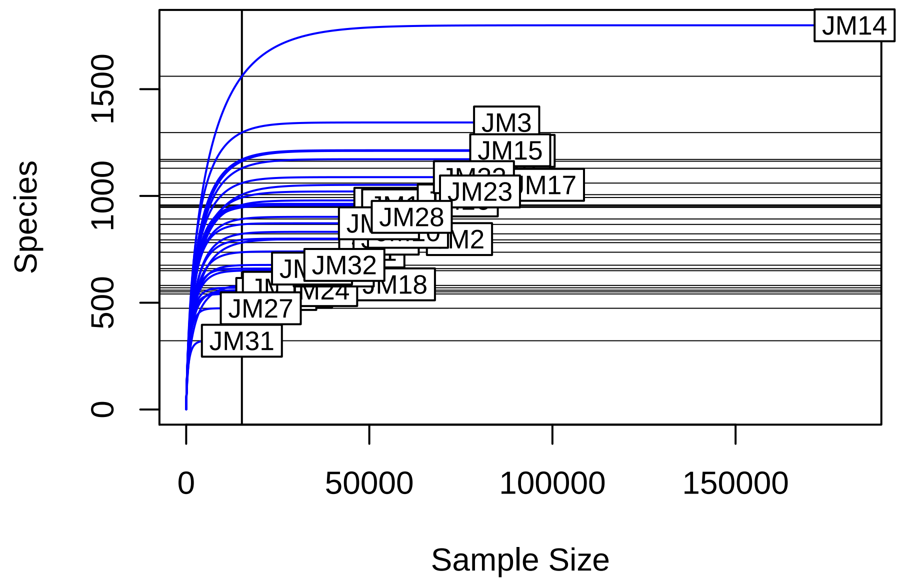
Rarefaction curves plot the relationship between read depth and the number of OTUs observed. If a curve plateaus it is an indicator of sufficient read depth.The following line was used to plot the rarefaction curve for sample JM1.

```
rarecurve(t(obj$data$otu_counts[,"JM1"]),step=20,
          sample=min(colSums(obj$data$otu_counts[,sample_data$Sample_ID])),
          col="blue",cex=1, add=TRUE)
```



It is easier to view all the curves on a single plot. The following lines did this.

```
n2<-ncol(obj$data$otu_counts)
curves<- obj$data$otu_counts[,3:n2]
rarecurve(t(curves),step=20,
          sample=min(colSums(obj$data$otu_counts[,sample_data$Sample_ID])),
          col="blue",cex=0.8, add=TRUE)
```



Diversity Analysis

Time to make some pretty pictures.

Alpha Diversity

Adding the alpha diversity, using the Inverse Simpson statistic, of each sample as a column in sample data made it easier to plot. The next line of code did this.

```
sample_data$alpha <- diversity(obj$data$otu_rarefied[,sample_data$Sample_ID],
                               MARGIN=2, index="invsimpson")
```

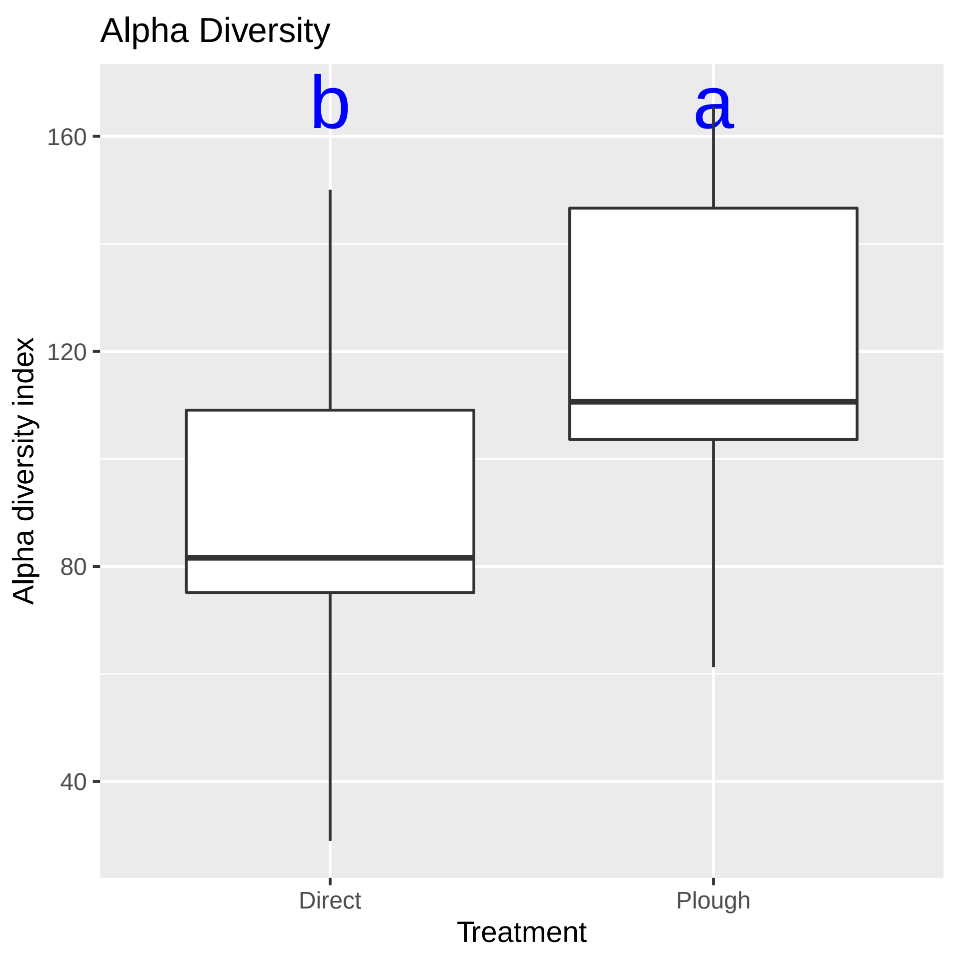
The following line made a function that creates a boxplot of alpha diversity measurements. It then uses a Tukey's Honest Significance Difference (HSD) test to do parewise comparisons to see if the mean diversity of a group is significantly different from the other. It will then write in a letter in blue above the groups. Different letters indicate the mean alpha diversity of one group is significantly different from the other.

```
compare_alpha<- function(sample_data, grouping_var)
  {
  sample_data$alpha <- diversity(obj$data$otu_rarefied[,sample_data$Sample_ID],
                                 MARGIN=2, index="invsimpson")
  sample_data$grouping <- sample_data[[grouping_var]]
  anova_result <- aov(alpha ~ grouping, sample_data)
  tukey_result <- HSD.test(anova_result, "grouping",group=TRUE)
  group_data <- tukey_result$groups[order(rownames(tukey_result$groups)),]
  my_plot <- ggplot(sample_data, aes(x=grouping, y=alpha)) +
    geom_text(data=data.frame(),
              aes(x=rownames(group_data),
                  y=max(sample_data$alpha)+1,
                  label =group_data$groups),
              col='blue', size =10)+
    geom_boxplot() +
    ggtitle("Alpha Diversity")+
    xlab(grouping_var)+
    ylab("Alpha diversity index")
  print(tukey_result)
  return(my_plot)
}
```

The next line used this function. It specificed that the samples would be grouped by 'Treatment'. This could be changed to view other variables such as 'Growth_Stage'.

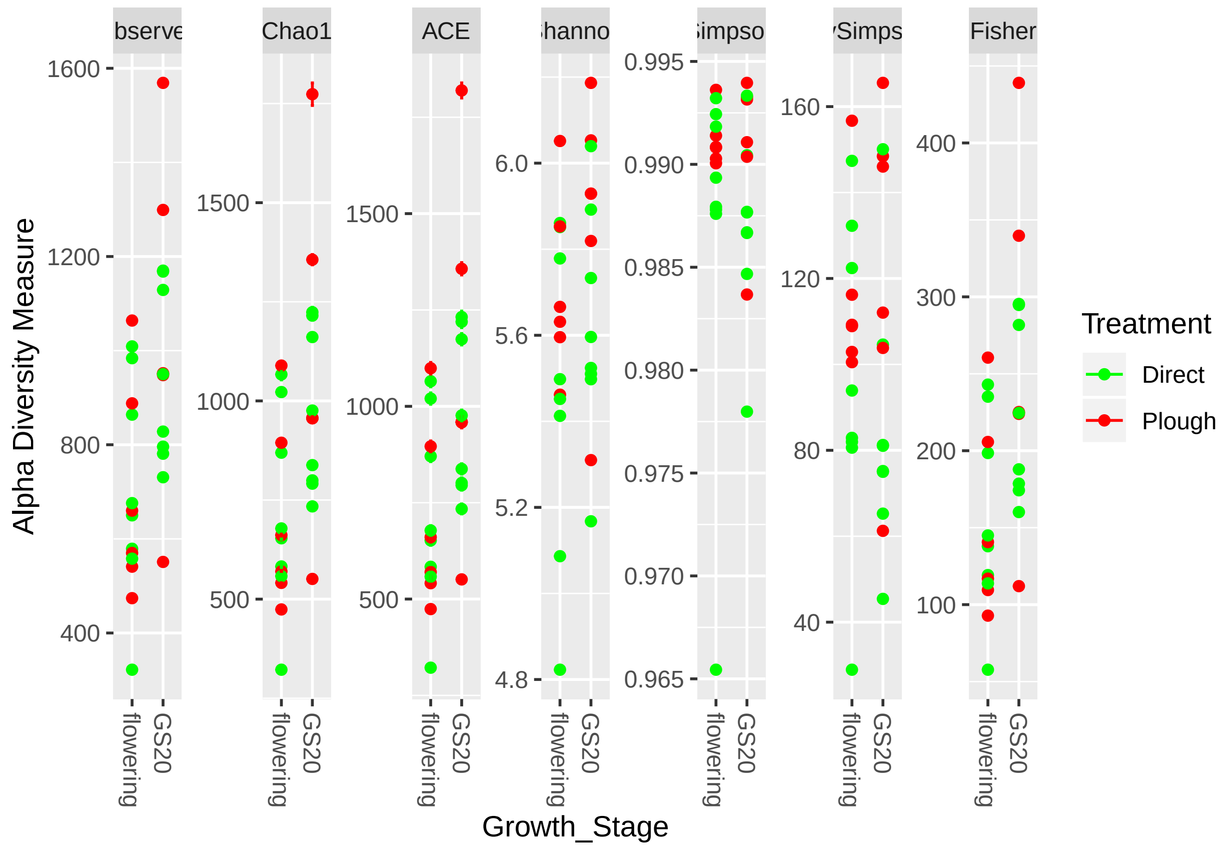
```
compare_alpha(sample_data, "Treatment")
```

```
## $statistics
##   MSerror Df     Mean       CV
##   1026.85 26 102.8501 31.15651
##
## $parameters
##    test   name.t ntr StudentizedRange alpha
##   Tukey grouping   2         2.906958  0.05
##
## $means
##            alpha      std  r      Min      Max       Q25      Q50      Q75
## Direct  90.51287 33.75822 16 28.93823 150.0676  75.12724  81.6020 109.0653
## Plough 119.29971 29.54787 12 61.25486 165.5215 103.59428 110.6314 146.6452
##
## $comparison
## NULL
##
## $groups
##            alpha groups
## Plough 119.29971      a
## Direct  90.51287      b
##
## attr(,"class")
## [1] "group"
```



Comparing multiple Alpha Diversity statistics by 2 variables:

```
ps <- as_phyloseq(obj, otu_table = "otu_rarefied", otu_id_col = "OTU_ID",
                  sample_data = sample_data, sample_id_col = "Sample_ID")
a_div_ps<-plot_richness(ps, color="Treatment", x="Growth_Stage")
a_div_ps + scale_color_manual(values=c("Direct"="green","Plough"="red"))
```



##Ploting taxon abundance with heat trees for a subgroup

A heat tree is a type of taxonomic tree. It shows the makeup of the species in the samples. The code below plotted a heat tree for a subgroup of data. In this case it was by 'Treatment'. It plotted OTUs that had counts of over 50 in samples grown under the 'Direct' treatment regime.

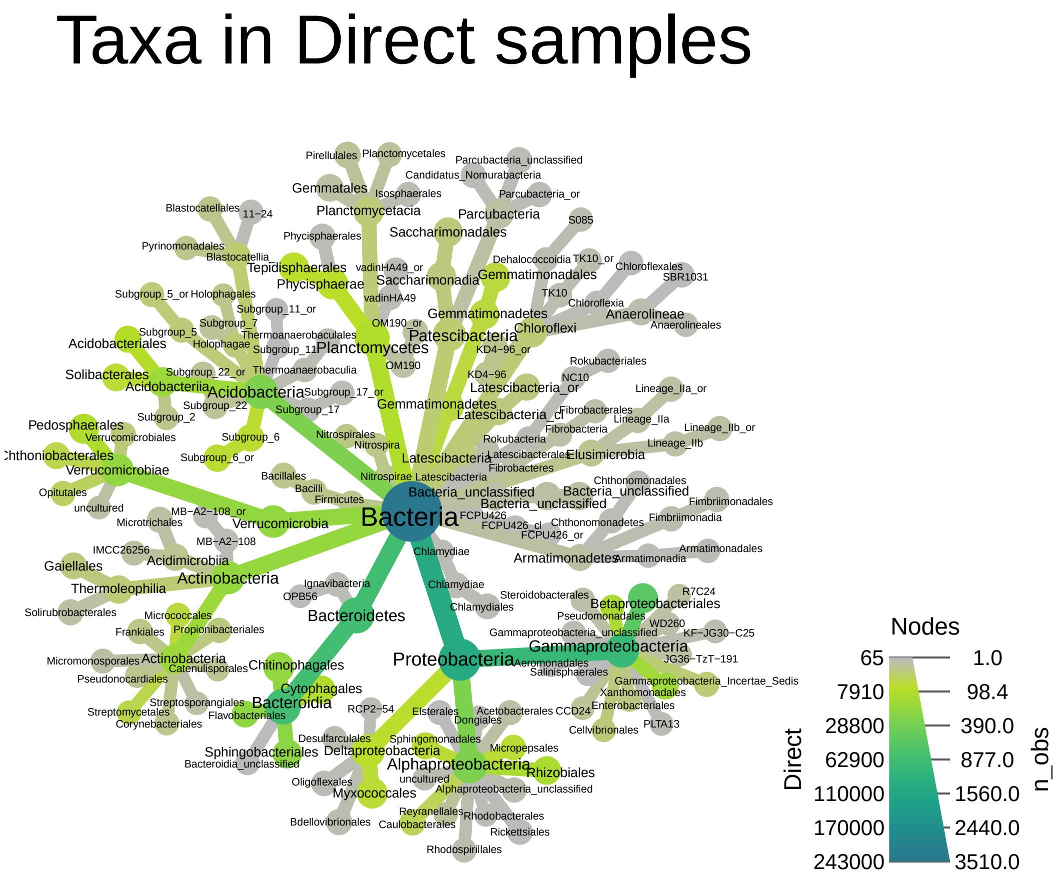
```
obj$data$type_abund <- calc_taxon_abund(obj,
                                        "otu_rarefied",
                                        cols=sample_data$Sample_ID,
                                        groups=sample_data$Treatment)
```

```
## Summing per-taxon counts from 28 columns in 2 groups for 1328 taxa
```

```
print(obj$data$type_abund)
```

```
## # A tibble: 1,328 x 3
##    taxon_id Direct Plough
##    <chr>     <dbl>  <dbl>
##  1 aab      243408 182556
##  2 aac      102548  65358
##  3 aad       62152  29268
##  4 aae       15980  15871
##  5 aaf         880   3089
##  6 aag       16336  12182
##  7 aah       24543  29594
##  8 aai        1097   2100
##  9 aaj        8036   7973
## 10 aak        1812    925
## # ... with 1,318 more rows
```

```
set.seed(1)
obj %>% taxa::filter_taxa(Direct>50)%>%
  taxa::filter_taxa(taxon_ranks=="Order", supertaxa=TRUE) %>%
  heat_tree(node_label=taxon_names,
            node_size=n_obs,
            node_color=Direct,
            layout="da",
            initial_layout="re",
            title="Taxa in Direct samples",
            output_file="Direct_heattree.pdf")
```



Beta diversity

The line below calculated beta diversity using the Bray-Curtis measure of dissimilarity.

```
beta_dist<- vegdist(t(obj$data$otu_rarefied[,sample_data$Sample_ID]),index="bray")
```

NMDS Plot The code below created a Non-MultiDimentional Scaling plot from the beta diversity statistics.

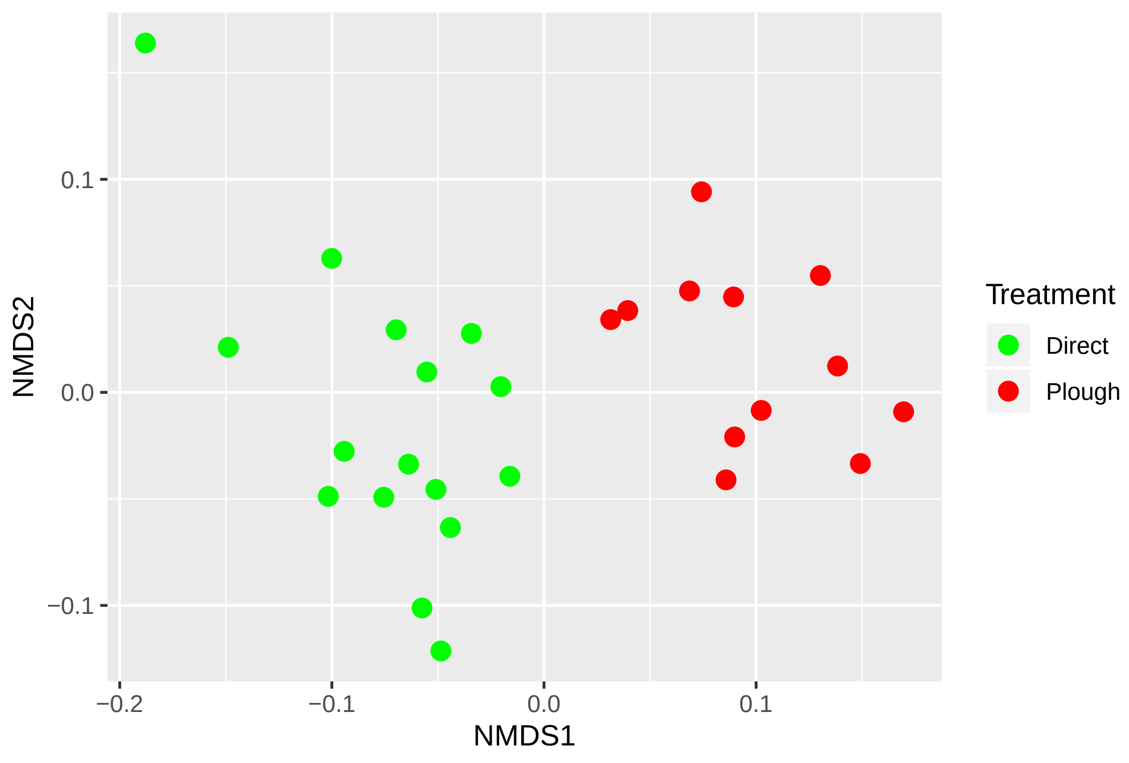
```
ps_ord <- ordinate(ps, method="NMDS",distance="jsd")
```

```
## Run 0 stress 0.1006149
## Run 1 stress 0.14982
## Run 2 stress 0.1009226
## ... Procrustes: rmse 0.01903096  max resid 0.08890496
## Run 3 stress 0.1205738
## Run 4 stress 0.119909
## Run 5 stress 0.1006149
## ... Procrustes: rmse 4.392819e-05  max resid 0.0002012265
## ... Similar to previous best
## Run 6 stress 0.1060755
## Run 7 stress 0.1470106
## Run 8 stress 0.1060755
## Run 9 stress 0.1199089
## Run 10 stress 0.1611045
## Run 11 stress 0.1199093
## Run 12 stress 0.1205181
## Run 13 stress 0.1611055
## Run 14 stress 0.1009221
## ... Procrustes: rmse 0.01901525  max resid 0.08874544
## Run 15 stress 0.1060752
## Run 16 stress 0.1006152
## ... Procrustes: rmse 0.0002213164  max resid 0.00101382
## ... Similar to previous best
## Run 17 stress 0.1060751
## Run 18 stress 0.1006151
## ... Procrustes: rmse 0.0001293563  max resid 0.000584511
## ... Similar to previous best
## Run 19 stress 0.1060757
## Run 20 stress 0.1006157
## ... Procrustes: rmse 0.0003473615  max resid 0.001596984
## ... Similar to previous best
## *** Solution reached
```

```
plot <- plot_ordination(ps, ps_ord, type="samples", color="Treatment")+geom_point(size=3)
#+ geom_label_repel(aes(label=mds_data$Sample_ID))
```

The plot can be coloured by variable, in this case 'Treatment'. Removing the # from infront of the lines allows the right coloring to be chosen for the right variable/experiment.

```
##Pick the plot colouring you want from options below
##Growth Stage
#plot + scale_color_manual(values=c("GS20"="purple","flowering"="orange"))
##Genotype
#plot + scale_color_manual(values=c("Laureate"="magenta","Concerto"="cyan"))
##Treatment
#plot+ scale_color_manual(values=c("Sustainable"="green","Conventional"="red"))
plot+ scale_color_manual(values=c("Direct"="green","Plough"="red"))
```



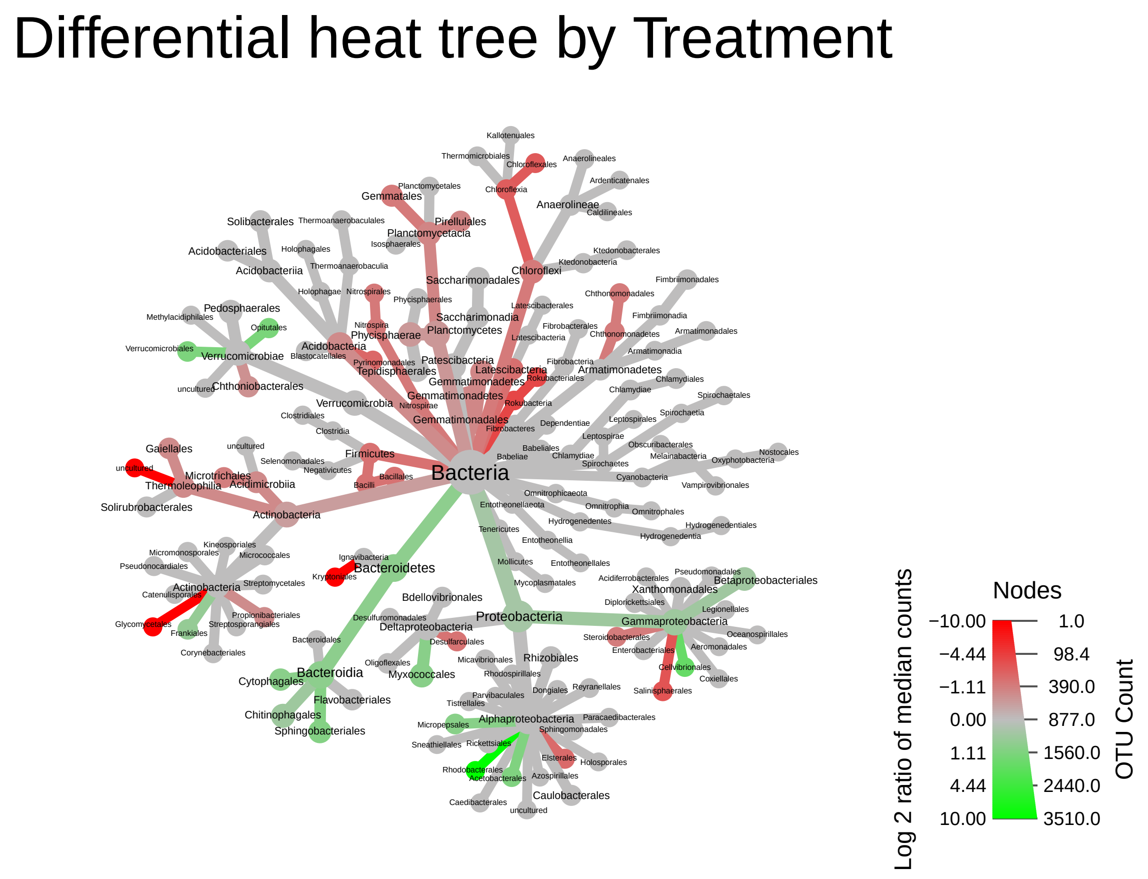
Differential Heat Trees

The following code was used to plot a heat tree that the shows which taxa are more abundant in one group than another - indicated by colour. The code below produced a differential heat tree that compared treatment types. Taxa in red (the first colour input in node_colour_range) are more abundant in treatment2 - in this case Plough. The image was saved as a PDF, which will allowing zooming.

```
obj$data$tax_abund <- calc_taxon_abund(obj, "otu_rarefied",
                                        cols=sample_data$Sample_ID)
obj$data$diff_table <- compare_groups(obj, dataset="tax_abund",
                                       cols=sample_data$Sample_ID,
                                       groups=sample_data$Treatment)
obj <- mutate_obs(obj,"diff_table", wilcox_p_value = p.adjust(wilcox_p_value, method="fdr"))
obj$data$diff_table$log2_median_ratio[obj$data$diff_table$wilcox_p_value>0.05]<- 0
```

```
set.seed(1)
obj %>% mutate_obs("cleaned_names", gsub(taxon_names, pattern="\\[|\\]", replacement="")) %>%
  taxa::filter_taxa(grepl(cleaned_names, pattern="^[a-z|A-Z]+$")) %>%
  taxa::filter_taxa(taxon_ranks=="Order", supertaxa=TRUE) %>%
  heat_tree(node_label=cleaned_names,
            node_size=n_obs,
            node_color=log2_median_ratio,
            node_color_interval = c(-10,10),
            node_color_range=c("red","gray","green"),
            node_size_axis_label="OTU Count",
            node_color_axis_label="Log 2 ratio of median counts",
            layout="da", initial_layout="re",
            title="Differential heat tree by Treatment",
            output_file="Differential_heat_tree_treatment.pdf")
```

```
## Adding a new "character" vector of length 1328.
```



The code for this pipeline has been adapted from an online workshop by the Grunwald lab available at https://grunwaldlab.github.io/analysis_of_microbiome_community_data_in_r/02--quick_example.html