Symbiodiniaceae *ITS2* community analyses

Ryan Eckert – ryan.j.eckert@gmail.com

10/16/2019

Contents

About this document

This is the code accompanies the publication **Eckert RJ, Reaume A, Sturm AB, Studivan MS, and Voss JD.(in review) Symbiodiniaceae community variation among Montastraea cavernosa populations along a depth gradient on the Belize Barrier Reef.** Here you will find all the code to repeat the analyses performed for this manuscript. All of the accompanying data can be found on my github.

If you download my entire accompanying github directory you should be able to re-run these analyses by following along with the included code in R Studio. If you download the code separtely or you are using this pipeline on your own data, you may need to change the working directory to whereever the associated files are housed (ie. `setwd("~/path/to/directory/with/data")`).

The data used for this analysis are OTUs calculated with R packages *dada2* and *LULU*. The raw Symbiodiniaceae *ITS2* sequences obtained from *Montastraea cavernosa* samples can be found in the NCBI SRA under project number XXX . Hopefully you are able to follow along this file and find it useful to use with your own data!

Basic setup of R environment and data

Loading required packages

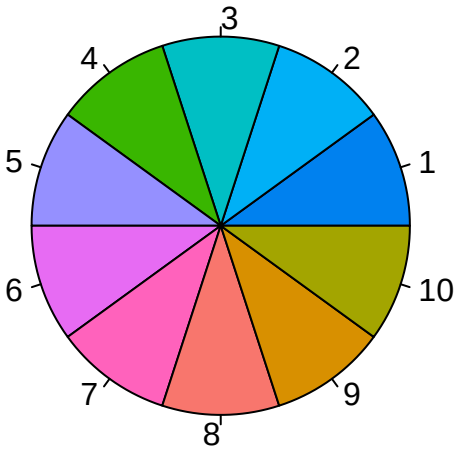
For the following analyses we will require the use of a number of different R packages. Most of these can be sourced from CRAN, but a couple need to be downloaded from GitHub or BioConducter. We can use the following code to quickly load in the packages and install any packages not previously installed in the R console.

```
if (!require("pacman")) install.packages("pacman")
if (!require("BiocManager")) install.packages("BiocManager", update = FALSE)
if (!require("pairwiseAdonis")) pacman::p_load_gh('pmartinezarbizu/pairwiseAdonis/
                                                   pairwiseAdonis')
if (!require("edgeR")) BiocManager::install("edgeR", update = FALSE)
pacman::p_load(car, cluster, dunn.test, edgeR, ggplot2, ggpubr, labdsv, MCMC.OTU,
               multcompView, pairwiseAdonis, pgirmess, plyr, reshape, vegan)

#setting seed allows randomized processes to be repeated later
set.seed(3254)
```

Creating a color palette to use for our data

We need 10 colors, I'm using mostly ggPlot colors, but in a better order for visualizing our OTU bar chart. I like to set a default palette at the beginning and call specific colors from it as needed throughout.

```
its2ColPal=c("#0081EF", "#00B0F6", "#00BFC4", "#39B600", "#9590FF", "#E76BF3", "#FF62BC",
             "#F8766D", "#D89000", "#A3A500")
```



Loading our data into R

First, we need to load in the data from dada2 and LULU analyses.

```
curated_result_97 = readRDS("curated_result_97.rds") #read in LULU output
ITS2data <- read.csv("ASVtable.csv") #read in dada2 output
head(ITS2data)
```

Preparing data for analyses

Now that the data are loaded in, we can continue to make these data the way we need them for our analyses. We can combine our curated OTUs with the metadata we added to our ASV file.

```
ITS2data = cbind(ITS2data[, 1:3], data.frame(t(curated_result_97$curated_table)))
```

And we can order sites from north to south so later data can be plotted shallow to deep; north to south. To do this we can set "Depth" and "Site" as factors

```
ITS2data$Depth = factor(ITS2data$Depth, levels = c("10", "16", "25", "35"))
levels(ITS2data$Site)
```

```
## [1] "GR" "RW" "SR" "TR"
```

```
ITS2data$Site = factor(ITS2data$Site, levels(ITS2data$Site)[c(4, 2, 3, 1)])
ITS2data = ITS2data[order(ITS2data$Site, ITS2data$Depth), ]
ITS2data$newOrder = 1:nrow(ITS2data)
ITS2data = cbind(ITS2data[, length(ITS2data), drop = FALSE],
                 ITS2data[, c(1:length(ITS2data) - 1)])
row.names(ITS2data) = ITS2data$Sample
head(ITS2data)
```

```
##     newOrder Sample Site Depth    sq1 sq10 sq11 sq14 sq17 sq18 sq25 sq272
## 107        1    107   TR    10   6456    0    0    0    0    0    0     0
## 108        2    108   TR    10 215412    0    0    0    0    0    0     0
## 109        3    109   TR    10  16407    0    0    0    0    0    0     0
## 110        4    110   TR    10 197896    0    0    6    3  618    1     0
## 111        5    111   TR    10 180119    0    0    0    0    0    0     0
## 112        6    112   TR    10  24462    0    0    0    0    0    0     0
##     sq300 sq32 sq360 sq442 sq474 sq485   sq5 sq503 sq531 sq537 sq547 sq555
## 107     0    0     0     0     0     0   649     0     0     0     0     0
## 108     0    0     0     0     0     0 17803     0     0     0     0     0
## 109     0    0     0     0     0     0  1397     0     0     0     0     0
## 110     0    2     0     0     0     0 19950     0     0     0     0     0
## 111     0    0     0     6     0     0 16261     3     0     1     0     0
## 112     0    0     0     0     0     0  2438     0     0     0     0     0
##     sq563 sq64  sq7 sq8
## 107     0    0  225   0
## 108     0    0 4736   0
## 109     0    0  266   0
## 110     0    0 7923  36
## 111     0    0 8595   0
## 112     0    0  862   0
```

Now we can call these data for the following analyses.

Purging outlying OTUs

```
goods = purgeOutliers(ITS2data, count.columns = 5:length(ITS2data), otu.cut = 0.001)
```

```
## [1] "samples with counts below z-score -2.5 :"
## [1] "107" "44"  "212" "220" "67"  "189" "186" "197"
## [1] "zscores:"
##       107        44       212       220        67       189       186
## -3.289894 -2.977162 -2.594488 -2.840026 -2.895177 -4.272415 -3.211249
##       197
## -4.032355
## [1] "OTUs passing frequency cutoff  0.001 : 10"
```

```
goods$newOrder = 1:nrow(goods)
row.names(goods) = goods$sample
head(goods)
```

```
##     newOrder sample Site Depth    sq1 sq10 sq11 sq14 sq17 sq18 sq25   sq5
## 108        1    108   TR    10 215412    0    0    0    0    0    0 17803
## 109        2    109   TR    10  16407    0    0    0    0    0    0  1397
## 110        3    110   TR    10 197896    0    0    6    3  618    1 19950
## 111        4    111   TR    10 180119    0    0    0    0    0    0 16261
## 112        5    112   TR    10  24462    0    0    0    0    0    0  2438
```

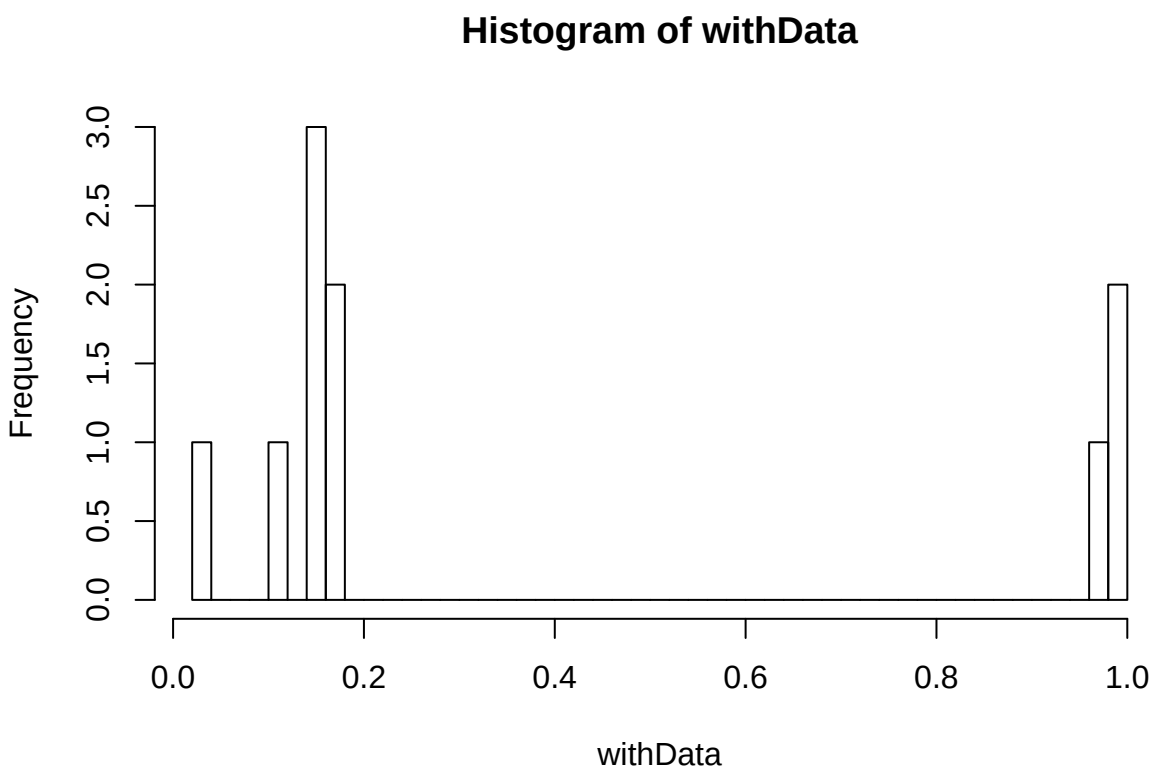
```
## 113        6    113    TR    10 195026    0    0    7   15  826   14 23224
##      sq7 sq8
## 108 4736   0
## 109  266   0
## 110 7923  36
## 111 8595   0
## 112  862   0
## 113 6242  21
```

```
row.names(goods) = goods$sample
```

Samples with data for OTUs

What is the proportion of samples with data for these OTUs?

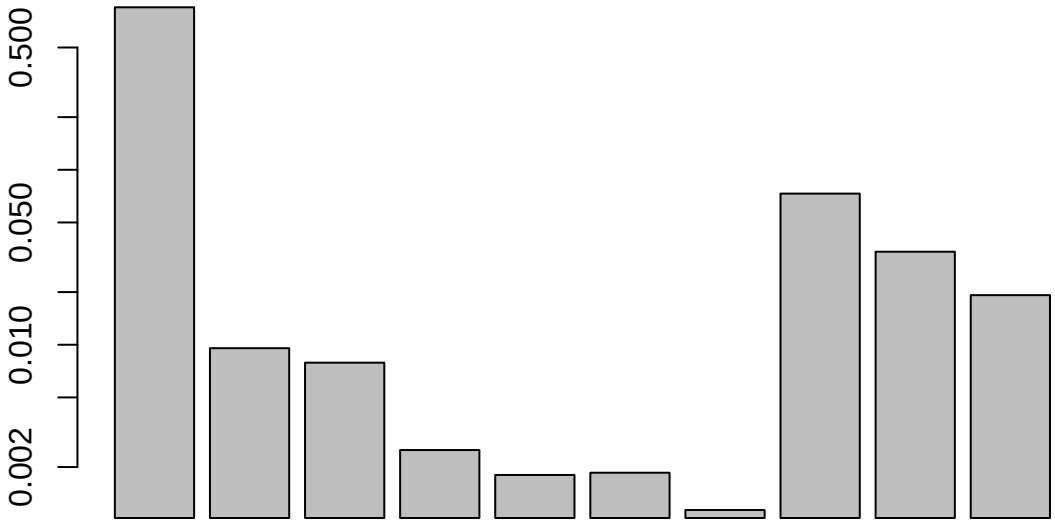
```
withData = apply(goods[, 5:length(goods[1, ])], 2, function(x) {
    sum(x > 0)/length(x)
})
hist(withData, breaks = 50)
```




```
props = apply(goods[, 5:length(goods[1, ])], 2, function(x) {
    sum(x)/sum(goods[, 5:length(goods[1, ])])
})

props
```

```
##         sq1        sq10        sq11        sq14        sq17        sq18
## 0.848909686 0.009562495 0.007899518 0.002501895 0.001801993 0.001855771
##        sq25         sq5         sq7         sq8
## 0.001136438 0.073101722 0.034023114 0.019207368
```

```
barplot(props, xaxt = "n", log = "y")
```



Normalizing reads across all samples

Here we normalize OTU counts with weighted trimmed mean of M-values (TMM; Robinson and Oshlack 2010; https://doi.org/10.1186/gb-2010-11-3-r25). This helps to account for disparity in sequencing depth across libraries. First we transpose the data to work with *edgeR*

```
itsGoodsTransposed = t(goods[, 5:length(goods[1, ])])
itsGoodsList = DGEList(counts = itsGoodsTransposed)
head(itsGoodsList$samples)
```

```
##     group lib.size norm.factors
## 108     1   237951            1
## 109     1    18070            1
## 110     1   226433            1
## 111     1   204975            1
## 112     1    27762            1
## 113     1   225375            1
```

Now we can use TMM normalization in *edgeR*

```
its2Norm =  calcNormFactors(itsGoodsList, method = "TMM")
head(its2Norm$samples)
```

```
##     group lib.size norm.factors
## 108     1   237951     1.090520
## 109     1    18070     1.095378
## 110     1   226433     1.057821
## 111     1   204975     1.063382
## 112     1    27762     1.066040
## 113     1   225375     1.048051
```

```
its2TMM = t(cpm(its2Norm, normalized.lib.sizes = TRUE))
its2Norm = cbind(goods[,c(2:4)], its2TMM)
head(its2Norm)
```

```
##     sample Site Depth      sq1 sq10 sq11     sq14     sq17     sq18
## 108    108   TR    10 830134.8    0    0  0.00000  0.00000    0.000
```

```
## 109    109   TR    10 828909.5    0    0  0.00000  0.00000    0.000
## 110    110   TR    10 826199.7    0    0 25.04951 12.52476 2580.100
## 111    111   TR    10 826360.2    0    0  0.00000  0.00000    0.000
## 112    112   TR    10 826546.9    0    0  0.00000  0.00000    0.000
## 113    113   TR    10 825665.6    0    0 29.63533 63.50427 3496.969
##          sq25      sq5      sq7       sq8
## 108  0.000000 68607.55 18251.16   0.00000
## 109  0.000000 70578.82 13438.77   0.00000
## 110  4.174918 83289.62 33077.88 150.29706
## 111  0.000000 74603.14 39432.63   0.00000
## 112  0.000000 82377.62 29126.13   0.00000
## 113 59.270653 98321.55 26426.24  88.90598
```

Renaming OTU sequence columns

Finally, I want to rename the columns with the Symbiodiniaceae sequences identified through our previose BLASTn query

```
colnames(its2Norm)[4:ncol(its2Norm)] = c("sq01_C3", "sq10_C3", "sq11_C3g", "sq14_C3g",
                                         "sq17_C3g", "sq18_C3g", "sq25_C3g", "sq05_C3z",
                                         "sq07_C3e", "sq08_C3g")
head(its2Norm)
```

```
##     sample Site Depth  sq01_C3 sq10_C3 sq11_C3g sq14_C3g sq17_C3g sq18_C3g
## 108    108   TR    10 830134.8       0        0  0.00000  0.00000    0.000
## 109    109   TR    10 828909.5       0        0  0.00000  0.00000    0.000
## 110    110   TR    10 826199.7       0        0 25.04951 12.52476 2580.100
## 111    111   TR    10 826360.2       0        0  0.00000  0.00000    0.000
## 112    112   TR    10 826546.9       0        0  0.00000  0.00000    0.000
## 113    113   TR    10 825665.6       0        0 29.63533 63.50427 3496.969
##      sq25_C3g sq05_C3z sq07_C3e  sq08_C3g
## 108  0.000000 68607.55 18251.16   0.00000
## 109  0.000000 70578.82 13438.77   0.00000
## 110  4.174918 83289.62 33077.88 150.29706
## 111  0.000000 74603.14 39432.63   0.00000
## 112  0.000000 82377.62 29126.13   0.00000
## 113 59.270653 98321.55 26426.24  88.90598
```

```
levels(its2Norm$Site)
```

```
## [1] "TR" "RW" "SR" "GR"
```

```
levels(its2Norm$Depth)
```

```
## [1] "10" "16" "25" "35"
```

After all that everything looks good to go. On to the fun stuff!

OTU α-diversity

Looking at α-diversity of Symbiodiniaceae OTU sequences in each of our samples for comparison. We will look at species richness, Shannon's index and Simpson's index ##Calculate α-diversity metrics

```
its2Richness = specnumber(its2Norm[,4:ncol(its2Norm)])
its2Shannon = diversity(its2Norm[,4:ncol(its2Norm)],index = "shannon")
its2Simpson = diversity(its2Norm[,4:ncol(its2Norm)],index = "simpson")
its2Div = cbind(its2Norm[,1:3], its2Richness, its2Shannon, its2Simpson)
```

```
colnames(its2Div)[4:6] = c("richness", "shannon", "simpson")
head(its2Div)
```

```
##     sample Site Depth richness   shannon   simpson
## 108    108   TR    10        3 0.3620252 0.1744764
## 109    109   TR    10        3 0.3476677 0.1693987
## 110    110   TR    10        8 0.4670561 0.2271793
## 111    111   TR    10        3 0.4476261 0.2197705
## 112    112   TR    10        3 0.4329306 0.2149295
## 113    113   TR    10        8 0.4816535 0.2397877
```

Statistical tests of α-diversity

Now we can test whether or not there a significant differences in any of the calculated α-diversity metrics across Site or Depth.

Assumption testing

First we need to see if our data are normally distributed. We can use Shapiro-Wilk's test.

```
tapply(its2Div$richness, its2Div$Site, shapiro.test)
```

```
## $TR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.49821, p-value = 8.713e-13
##
##
## $RW
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.59852, p-value = 2.451e-11
##
##
## $SR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.51981, p-value = 1.024e-12
##
##
## $GR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.58742, p-value = 2.111e-11
```

```
tapply(its2Div$shannon, its2Div$Site, shapiro.test)
```

```
## $TR
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.55244, p-value = 4.961e-12
##
##
## $RW
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.61603, p-value = 4.648e-11
##
##
## $SR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.69458, p-value = 6.984e-10
##
##
## $GR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.51309, p-value = 1.8e-12
```

```
tapply(its2Div$simpson, its2Div$Site, shapiro.test)
```

```
## $TR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.56825, p-value = 8.468e-12
##
##
## $RW
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.63812, p-value = 1.073e-10
##
##
## $SR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.73532, p-value = 4.536e-09
##
```

```
##
## $GR
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.55991, p-value = 8.211e-12
```

```
tapply(its2Div$richness, its2Div$Depth, shapiro.test)
```

```
## $`10`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.7758, p-value = 3.534e-08
##
##
## $`16`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.67303, p-value = 4.331e-10
##
##
## $`25`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.2022, p-value = 3.019e-16
##
##
## $`35`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.17992, p-value = 4.725e-16
```

```
tapply(its2Div$shannon, its2Div$Depth, shapiro.test)
```

```
## $`10`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.67641, p-value = 3.197e-10
##
##
## $`16`
##
##  Shapiro-Wilk normality test
##
```

```
## data:  X[[i]]
## W = 0.68356, p-value = 6.73e-10
##
##
## $`25`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.71312, p-value = 1.977e-09
##
##
## $`35`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.77289, p-value = 6.555e-08
```

```
tapply(its2Div$simpson, its2Div$Depth, shapiro.test)
```

```
## $`10`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.70942, p-value = 1.353e-09
##
##
## $`16`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.70544, p-value = 1.734e-09
##
##
## $`25`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.74479, p-value = 8.794e-09
##
##
## $`35`
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.83899, p-value = 2.827e-06
```

We see that our data are NOT normally distributed.

We can use Levene's test to test for equal variance among sites and depths.

```
leveneTest(its2Div$richness, its2Div$Depth)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   3  25.117 4.368e-14 ***
##       229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(its2Div$shannon, its2Div$Depth)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   3  6.7836 0.0002115 ***
##       229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(its2Div$simpson, its2Div$Depth)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   3  6.6744 0.0002442 ***
##       229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(its2Div$richness, its2Div$Site)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   3  1.0165 0.3861
##       229
```

```
leveneTest(its2Div$shannon, its2Div$Site)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   3  0.4804 0.6962
##       229
```

```
leveneTest(its2Div$simpson, its2Div$Site)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   3  0.5514 0.6477
##       229
```

Data are non-normally distributed and also heteroschedastic. Using non-parametric tests moving forward.

Kruskal-Wallis Tests

Since our data did not fit the assumptions of ANOVA, we will use single factor Krskal-Wallis tests on these data.

K-W test on Shannon's indices

```
kruskal.test(shannon ~ Site, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  shannon by Site
## Kruskal-Wallis chi-squared = 2.7884, df = 3, p-value = 0.4254
```

```
kruskal.test(shannon ~ Depth, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  shannon by Depth
## Kruskal-Wallis chi-squared = 3.0743, df = 3, p-value = 0.3803
```

Nothing significant

K-W test on Simpson's indices

```
kruskal.test(simpson ~ Site, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  simpson by Site
## Kruskal-Wallis chi-squared = 3.4937, df = 3, p-value = 0.3216
```

```
kruskal.test(simpson ~ Depth, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  simpson by Depth
## Kruskal-Wallis chi-squared = 3.8446, df = 3, p-value = 0.2787
```

Again, no significant tests

K-W test on species richness

```
kruskal.test(richness ~ Site, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  richness by Site
## Kruskal-Wallis chi-squared = 2.7628, df = 3, p-value = 0.4297
```

```
kruskal.test(richness ~ Depth, data = its2Div)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  richness by Depth
## Kruskal-Wallis chi-squared = 38.993, df = 3, p-value = 1.741e-08
```

Here we see that species richness changes significantly with Depth

Pairwise comparisons with Dunn's test

We now use Dunn's test to do multiple comparisons with bonferroni correction. This will reveal where the differences in species richness are across Depth

```
dunn.test(its2Div$richness, g = its2Div$Depth, method = "bonferroni", table = FALSE,
          list = TRUE)
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 38.9935, df = 3, p-value = 0
##
##
##                             Comparison of x by group
##                                   (Bonferroni)
##
## List of pairwise comparisons: Z statistic (adjusted p-value)
## ------------------------------
## 10 - 16 :  1.449108 (0.4419)
## 10 - 25 :  4.989355 (0.0000)*
## 16 - 25 :  3.504135 (0.0014)*
## 10 - 35 :  5.057011 (0.0000)*
## 16 - 35 :  3.591117 (0.0010)*
## 25 - 35 :  0.133160 (1.0000)
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```

This shows us that species richness is different between our deepest sites and our shallowest sites

Plotting α-diversity metrics

We can take all of our α-diversity metrics and plot them all into a single figure panel to visualize the results of the previous tests. Here we make plots for richness, Simpson's and Shannon's indices by site and depth zone.

Plot species richness by site

```
its2RichPlotS = ggplot(data = its2Div, aes(x = Site, y = richness)) +
                geom_point(aes(fill = Depth),shape = 21, color = "gray20", size = 2,
                           position = position_jitterdodge()) +
                geom_boxplot(alpha = 0, outlier.shape = NA, color = "gray30") +
                scale_fill_manual(values = its2ColPal[c(1, 4, 6, 2)],
                                  labels = c("10 m", "16 m", "25 m", "35 m")) +
                xlab("Reef Site") +
                ylab("Richness") +
                stat_compare_means(size = 5, geom = "label", label.x = 1, label.y = 10.7) +
                coord_cartesian(ylim = c(1, 11)) +
                scale_y_continuous(breaks = seq(2, 10, 2)) +
                guides(fill = guide_legend(ncol = 2)) +
                theme_bw()

its2RichPlotSite = its2RichPlotS +
```

```
                    theme(axis.title.x = element_blank(),
                          axis.text.x = element_blank(),
                          axis.title.y = element_text(color = "black", size = 14),
                          axis.text.y = element_text(color = "black", size = 12),
                          legend.title = element_text(color = "black", size = 14),
                          legend.text = element_text(color = "black", size = 12),
                          panel.border = element_rect(size = 1.1, color = "black"),
                          panel.background = element_rect(fill = "white"),
                          plot.background = element_rect(fill = "white"),
                          legend.position = "bottom",
                          legend.background = element_rect(color = "black"),
                          plot.margin = unit(c(0.5, 0.1, 0.1, 0.61), "cm")
                          )

its2RichPlotSite
```

Create pairwise comparison letters

We know from our Kruskal-Wallis tests that there was a significant difference in the species diversity across depth zones. We can plot letters to denote similarity between groups on our plot.

```
richDkmc = kruskalmc(its2Div$richness ~ its2Div$Depth) # multiple-comparison test
print(richDkmc)
```

```
## Multiple comparison test after Kruskal-Wallis
## p.value: 0.05
## Comparisons
##         obs.dif critical.dif difference
## 10-16 14.040805     32.74642      FALSE
## 10-25 48.134463     32.60500       TRUE
## 10-35 49.441667     33.04241       TRUE
## 16-25 34.093659     32.88254       TRUE
## 16-35 35.400862     33.31630       TRUE
## 25-35  1.307203     33.17732      FALSE
```

```
richDkmcDiff = richDkmc$dif.com$difference # select logical vector
names(richDkmcDiff) = row.names(richDkmc$dif.com)# add comparison names

# create a list with "homogenous groups" coded by letter
richDsigLetters = multcompLetters(richDkmcDiff, compare="<", threshold=0.05,
                                  Letters=c(letters, LETTERS, "."), reversed = FALSE)
richLetters = as.data.frame(richDsigLetters$Letters)
richLetters$depth = row.names(richLetters)
colnames(richLetters)[1] = "id"
richLetters$y = c(1, 1, 1, 1)
head(richLetters)
```

```
##    id depth y
## 10  a    10 1
## 16  a    16 1
## 25  b    25 1
## 35  b    35 1
```

These letters can now be added to the corresponding plot

Plot species richness by depth

```
its2RichPlotD = ggplot(data = its2Div, aes(x = Depth, y = richness)) +
                geom_point(aes(fill = Site), shape = 21, color = "gray 20", size = 2,
                           position = position_jitterdodge()) +
                geom_boxplot(alpha = 0, outlier.shape = NA, color = "gray30") +
                scale_fill_manual(values = its2ColPal[c(8,5,3,9)],
                                  labels = c("Tobacco Reef", "Raph's Wall",
                                             "South Reef", "Gover's Reef")) +
                stat_compare_means(size = 5, geom = "label", label.x = 1.07,
                                   label.y = 10.7) +
                geom_text(data = richLetters, aes(x = depth, y = y, label = id),
                          size = 6) +
                coord_cartesian( ylim = c(1, 11)) +
                scale_y_continuous(breaks = seq(2, 10, 2))+
                xlab("Depth (m)") +
                ylab("Richness") +
                guides(fill=guide_legend(ncol=2))+
                theme_bw()

its2RichPlotDepth = its2RichPlotD +
                     theme(axis.title.x = element_blank(),
                           axis.text.x = element_blank(),
                           axis.title.y = element_blank(),
                           axis.text.y = element_blank(),
                           legend.title = element_text(color = "black", size = 14),
                           legend.text = element_text(color = "black", size = 12),
                           panel.border = element_rect(size = 1.1, color = "black"),
                           panel.background = element_rect(fill = "white"),
                           plot.background = element_rect(fill = "white"),
                           legend.position = "bottom",
                           legend.background = element_rect(color = "black"),
                           plot.margin = unit(c(0.5, 0.5, 0.1, 0.85), "cm")
                           )
its2RichPlotDepth
```

Make legends grobs

Now we want to take the Site and Depth legends and make them graphical objects (grobs). This will allow us to place them separately in our figure grid, so each plot won't have its own legend.

```
get_legend <- function(its2RichPlotSite) {
  tmp <- ggplot_gtable(ggplot_build(its2RichPlotSite))
  leg <-
  which(sapply(tmp$grobs, function(x)
  x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}

legend.site = get_legend(its2RichPlotSite)

get_legend = function(its2RichPlotDepth) {
  tmp <- ggplot_gtable(ggplot_build(its2RichPlotDepth))
```

```
  leg <-
  which(sapply(tmp$grobs, function(x)
  x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}

legend.depth = get_legend(its2RichPlotDepth)

# re-plot without legends
its2RichPlotSite = its2RichPlotSite + theme(legend.position = "none")
its2RichPlotDepth = its2RichPlotDepth + theme(legend.position = "none")
```

Plot Shannon's index by site

```
its2ShannonPlotS = ggplot(data = its2Div, aes(x = Site, y = shannon)) +
                   geom_point(aes(fill = Depth), shape = 21, color = "gray20",
                              size = 2, position = position_jitterdodge()) +
                   geom_boxplot(alpha = 0, outlier.shape = NA, color = "gray30") +
                   scale_fill_manual(values = its2ColPal[c(1, 4, 6, 2)],
                                     labels = c("10 m", "16 m", "25 m", "35 m")) +
                   stat_compare_means(size = 5, geom = "label", label.x = 1,
                                      label.y = 1.7) +
                   expand_limits(y = c(0, 1.75)) +
                   xlab("Reef Site") +
                   ylab("Shannon's index") +
                   theme_bw()

its2ShannonPlotSite = its2ShannonPlotS +
                      theme(axis.title.x = element_blank(),
                            axis.text.x = element_blank(),
                            axis.title.y = element_text(color = "black", size = 14),
                            axis.text.y = element_text(color = "black", size = 12),
                            legend.title = element_text(color = "black", size = 14),
                            legend.text = element_text(color = "black", size = 12),
                            panel.border = element_rect(size = 1.1, color = "black"),
                            panel.background = element_rect(fill = "white"),
                            plot.background = element_rect(fill = "white"),
                            legend.position = "none",
                            legend.background = element_rect(color = "black"),
                            plot.margin = unit(c(0.1, 0.1, 0.1, 0.5), "cm")
                            )

its2ShannonPlotSite
```

Plot Shannon's index by depth

```
its2ShannonPlotD = ggplot(data = its2Div, aes(x = Depth, y = shannon))+
                   geom_point(aes(fill = Site), shape = 21, color = "gray20", size = 2,
                              position = position_jitterdodge()) +
                   geom_boxplot(alpha = 0, outlier.shape = NA, color = "gray30") +
                   scale_fill_manual(values = its2ColPal[c(8,5,3,9)],
                                     labels = c("Tobacco Reef", "Raph's Wall",
```

```
                                    "South Reef", "Gover's Reef")) +
                    stat_compare_means(size = 5, geom = "label", label.x = 1,
                                       label.y = 1.7) +
                    expand_limits(y = c(0, 1.75)) +
                    xlab("Depth (m)") +
                    ylab("Shannon's index") +
                    theme_bw()

its2ShannonPlotDepth = its2ShannonPlotD +
                        theme(axis.title.x = element_blank(),
                              axis.text.x = element_blank(),
                              axis.title.y = element_blank(),
                              axis.text.y = element_blank(),
                              legend.title = element_text(color = "black", size = 14),
                              legend.text = element_text(color = "black", size = 12),
                              panel.border = element_rect(size = 1.1, color = "black"),
                              panel.background = element_rect(fill = "white"),
                              plot.background = element_rect(fill = "white"),
                              legend.position = "none",
                              legend.background = element_rect(color = "black"),
                              plot.margin = unit(c(0.1, 0.5, 0.1, 0.85), "cm")
                              )

its2ShannonPlotDepth
```

Plot Simpson's index by site

```
its2SimpsonPlotS = ggplot(data = its2Div, aes(x = Site, y = simpson)) +
                    geom_point(aes(fill = Depth), shape = 21, color = "gray20", size = 2,
                               position = position_jitterdodge()) +
                    geom_boxplot(alpha = 0, outlier.shape = NA) +
                    scale_fill_manual(values = its2ColPal[c(1,4,6,2)],
                                      labels = c("10 m", "16 m", "25 m", "35 m")) +
                    stat_compare_means(size = 5, geom = "label", label.x = 1,
                                       label.y = 0.77) +
                    expand_limits(y = c(0, 0.8)) +
                    xlab("Reef Site") +
                    ylab("Simpson's index") +
                    theme_bw()

its2SimpsonPlotSite = its2SimpsonPlotS +
                       theme(axis.title.x = element_text(color = "black", size = 14),
                             axis.text.x = element_text(color = "black", size = 12),
                             axis.title.y = element_text(color = "black", size = 14),
                             axis.text.y = element_text(color = "black", size = 12),
                             legend.title = element_text(color = "black", size = 14),
                             legend.text = element_text(color = "black", size = 12),
                             panel.border = element_rect(size = 1.1, color = "black"),
                             panel.background = element_rect(fill = "white"),
                             plot.background = element_rect(fill = "white"),
                             legend.position = "none",
                             legend.background = element_rect(color = "black"),
                             plot.margin = unit(c(0.11, 0.1, 0, 0.5), "cm")
```

```
                            )

its2SimpsonPlotSite
```

Plot Simpson's index by depth

```
its2SimpsonPlotD = ggplot(data = its2Div, aes(x = Depth, y = simpson)) +
                   geom_point(aes(fill = Site), shape = 21, color = "gray20", size = 2,
                              position = position_jitterdodge()) +
                   geom_boxplot(alpha = 0, outlier.shape = NA) +
                   scale_fill_manual(values = its2ColPal[c(8,5,3,9)],
                                     labels = c("Tobacco Reef", "Raph's Wall",
                                     "South Reef", "Gover's Reef")) +
                   stat_compare_means(size = 5, geom = "label", label.x = 1,
                                      label.y = 0.77) +
                   expand_limits(y = c(0, 0.8)) +
                   xlab("Depth (m)") +
                   ylab("Simpson's index") +
                   theme_bw()

its2SimpsonPlotDepth = its2SimpsonPlotD +
                       theme(axis.title.x = element_text(color = "black", size = 14),
                             axis.text.x = element_text(color = "black", size = 12),
                             axis.title.y = element_blank(),
                             axis.text.y = element_blank(),
                             legend.title = element_text(color = "black", size = 14),
                             legend.text = element_text(color = "black", size = 12),
                             panel.border = element_rect(size = 1.1, color = "black"),
                             panel.background = element_rect(fill = "white"),
                             plot.background = element_rect(fill = "white"),
                             legend.position = "none",
                             legend.background = element_rect(color = "black"),
                             plot.margin = unit(c(0.1, 0.5, 0, 0.85), "cm")
                             )

its2SimpsonPlotDepth
```

Create a single figure panel

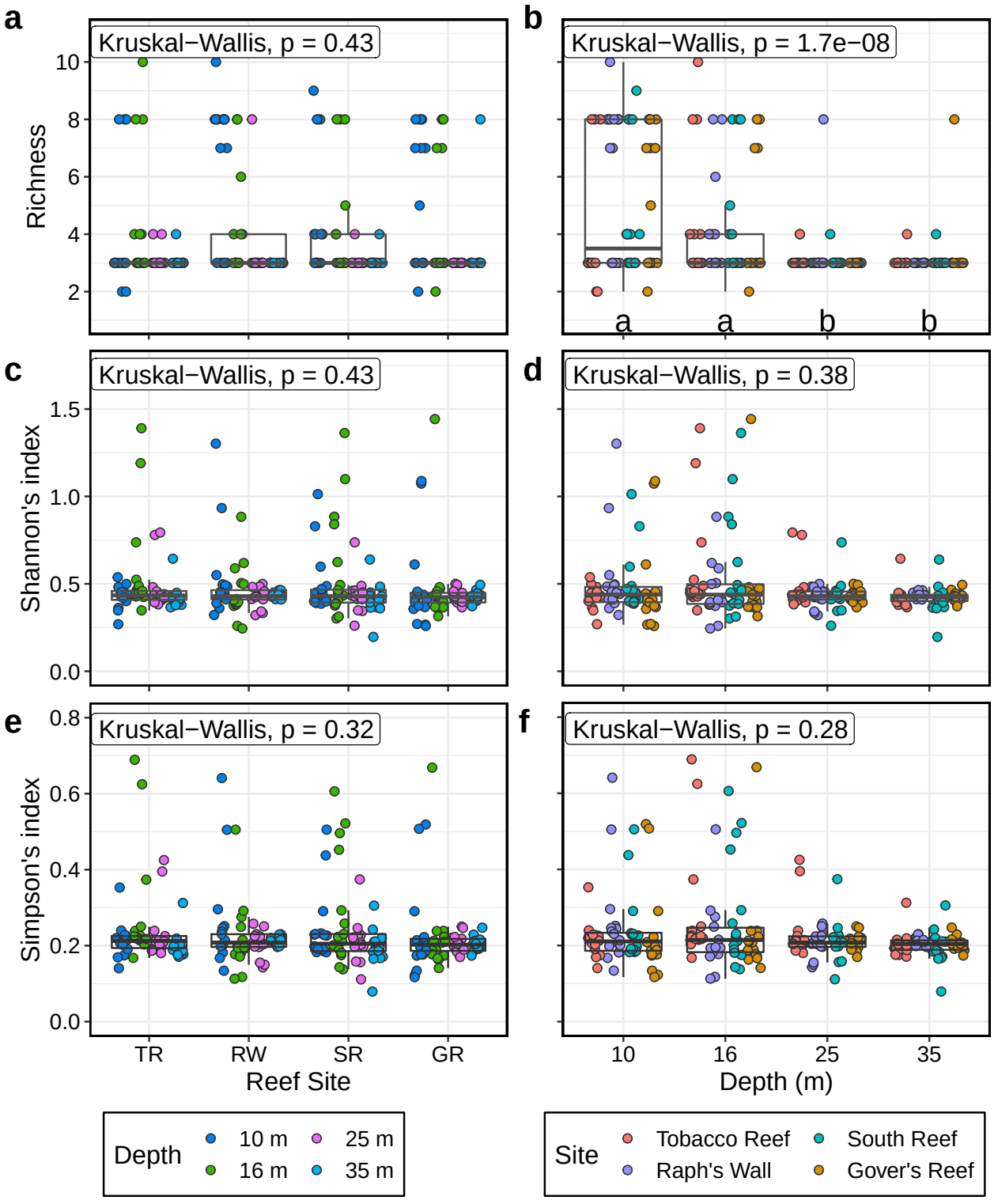
Now we want to throw all those bad boys into a single figure grid.

```
divPlots = ggarrange(its2RichPlotSite, its2RichPlotDepth, its2ShannonPlotSite,
                     its2ShannonPlotDepth, its2SimpsonPlotSite, its2SimpsonPlotDepth,
                     legend.site, legend.depth, ncol = 2, nrow = 4, widths = c(3.5,3.4),
                     heights = c(3,3,3.4,1),
                     labels = c("a","b", "c", "d", "e", "f","",""),
                     font.label = list(size = 18, color = "black", face ="bold")
                     )

divPlots
```

Save as high-res images

Finally, we will save the resulting α-diversity plot panel as a high resoultion .eps figure. Alternatively you could save it as a HQ .tiff by simply changing the extension in the code below.

```
ggsave("its2_diversityPlots.eps", plot = divPlots, width = 8.25, height = 10,
       unit = "in", dpi = 600)
```



Ordination with PCoA

We can use principal coordinates analysis (PCoA) to visualize our samples based on significant OTUs. But, first we need to set up data for PCoA analysis in R.

Create distance matrix

We will create a distance matrix with Bray-Curtis similarity using the package *vegan*

```
its2Dist = vegdist(its2Norm[, c(4:ncol(its2Norm))], method = "bray")
```

Perform PCoA

This is the actual PCoA step

```
its2Mds = cmdscale(its2Dist, eig = TRUE, x.ret = TRUE)
```

Determine percent variation captured on each axis

Calculate the eigenvalues so later we can figure out % variation shown on each Principal Coordinate

```
its2Var = round(its2Mds$eig/sum(its2Mds$eig)*100, 1)
its2Var
```

```
##   [1] 83.3 10.5  3.0  1.8  1.3  0.9  0.4  0.3  0.2  0.2  0.1  0.1  0.1  0.0
##  [15]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [29]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [43]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [57]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [71]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [85]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  [99]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [113]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [127]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [141]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [155]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [169]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [183]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [197]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [211]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## [225]  0.0  0.0 -0.1 -0.1 -0.1 -0.2 -0.3 -0.4 -0.6
```

Format data to plot

```
its2Values = its2Mds$points

its2Values =as.data.frame(its2Values, sample = rownames(its2Values))
its2Pcoa = data.frame(sample = rownames(its2Values), site = as.factor(its2Norm$Site),
                      depth = as.factor(its2Norm$Depth),PCo1 = its2Values[,1],
                      PCo2 = its2Values[,2])

head(its2Pcoa)
```

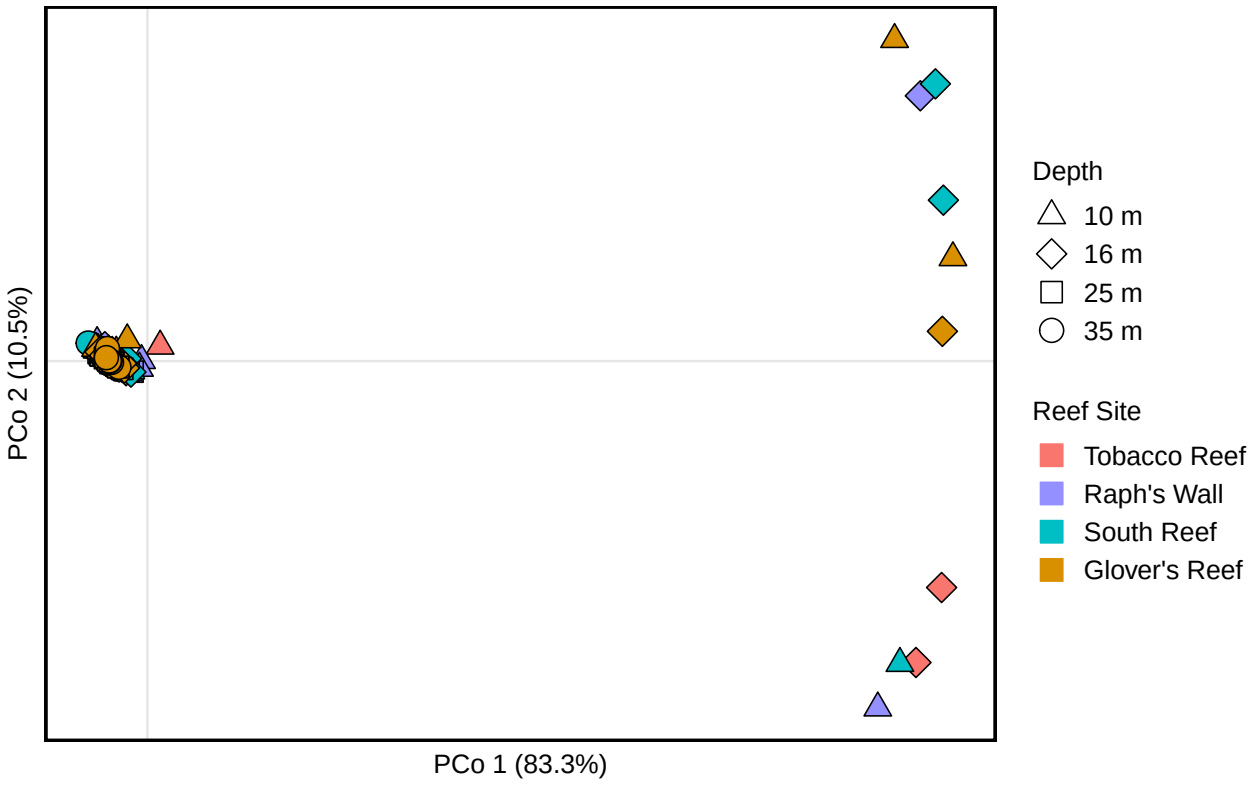
```
##   sample site depth        PCo1         PCo2
## 1    108   TR    10 -0.04147518  0.004611102
## 2    109   TR    10 -0.04211077  0.005071409
## 3    110   TR    10 -0.03039080 -0.004883738
## 4    111   TR    10 -0.03416250 -0.004766676
## 5    112   TR    10 -0.03216465 -0.003956257
## 6    113   TR    10 -0.02452031 -0.006515757
```

PCoA biplot

```
its2PcoaA = ggplot(its2Pcoa, aes(x = PCo1, y = PCo2)) +
            geom_hline(yintercept = 0, color = "gray90", size = 0.5) +
            geom_vline(xintercept = 0, color = "gray90", size = 0.5) +
            geom_point(aes(shape = factor(depth), size = depth, fill = site),
                       color = "black") +
            scale_fill_manual(values = its2ColPal[c(8,5,3,9)], name = "Reef Site",
                              labels = c("Tobacco Reef", "Raph's Wall", "South Reef",
                                         "Glover's Reef")) +
            scale_shape_manual(values = c(24,23,22,21), name = "Depth",
                               labels = c("10 m", "16 m", "25 m", "35 m")) +
            scale_size_manual(values = c(4, 4.75, 4.75, 4.75)) +
            guides(shape = guide_legend(override.aes = list(size = c(4, 4.75, 4.75, 4.75),
                   fill = "white")), fill = guide_legend(override.aes =
                   list(shape = 22, size = 5.75, color = "white")), size = FALSE,
                   color = FALSE)+
             xlab(paste ("PCo 1 (", its2Var[1],"%)", sep = "")) +
             ylab(paste ("PCo 2 (", its2Var[2],"%)", sep = "")) +
             theme_bw()

its2Pcoa = its2PcoaA +
           theme(axis.title.x = element_text(color = "black", size = 12),
                 axis.text.x = element_blank(),
                 axis.ticks.x = element_blank(),
                 axis.line.x = element_blank(),
                 axis.title.y = element_text(color = "black", size = 12),
                 axis.text.y = element_blank(),
                 axis.ticks.y = element_blank(),
                 axis.line.y = element_blank(),
                 legend.position = "right",
                 legend.title = element_text(color = "black", size = 12),
                 legend.text = element_text(color = "black", size = 12),
                 legend.key = element_blank(),
                 panel.border = element_rect(color = "black", size = 1.2),
                 panel.background = element_rect(fill = "white"),
                 plot.background = element_rect(fill = "white"),
                 panel.grid.major = element_blank(),
                 panel.grid.minor = element_blank()
                 )

its2PcoaA
```

Save the generated PCoA plot

```
ggsave("ITS2_OTU_PCoA.eps", plot = its2Pcoa, width = 10, height = 6.25, dpi = 600,
       device="eps")
```



Ordination with nMDS

Now we can view the data with nMDS using the package *vegan* again. First we have to run the calculations for the nMDS. This is similar to PCoA, but can be scaled freely and uses an iterative process. We'll run it with 50 iterations, which should be enough to arrive at a solution.

```
its2Nmds = metaMDS(its2Norm[4:ncol(its2Norm)], try = 50)
its2Nmds
```

Prepare data to plot

```
its2Scores = as.data.frame(scores(its2Nmds))
its2Scores$site = factor(its2Norm$Site)
its2Scores$depth = as.factor(its2Norm$Depth)
its2Scores$sample = row.names(its2Scores)
head(its2Scores)
```

```
##             NMDS1        NMDS2 site depth sample
## 108 -0.052295737 -0.058459256   TR    10    108
## 109 -0.048164717 -0.091740894   TR    10    109
## 110 -0.022926842  0.009377931   TR    10    110
## 111 -0.069887493  0.011330805   TR    10    111
## 112 -0.056906218 -0.016534503   TR    10    112
## 113  0.002797965 -0.011414792   TR    10    113
```

```
its2Clades = as.data.frame(scores(its2Nmds,"species"))
its2Clades$seq = row.names(its2Clades)
its2Clades
```

```
##                 NMDS1       NMDS2      seq
## sq01_C3  -0.074715881 -0.03014023  sq01_C3
## sq10_C3   0.031219752  0.73387836  sq10_C3
## sq11_C3g  1.736981629  0.77272569 sq11_C3g
## sq14_C3g  1.528327383 -0.65265579 sq14_C3g
## sq17_C3g  1.581939252  0.10004674 sq17_C3g
## sq18_C3g  0.511145818 -0.33556718 sq18_C3g
## sq25_C3g  1.610315632 -0.01124022 sq25_C3g
## sq05_C3z -0.006323011 -0.04149856 sq05_C3z
## sq07_C3e -0.049387470  0.03985393 sq07_C3e
## sq08_C3g  1.670623995 -0.13028495 sq08_C3g
```

```
its2Clades$seq
```

```
##  [1] "sq01_C3"  "sq10_C3"  "sq11_C3g" "sq14_C3g" "sq17_C3g" "sq18_C3g"
##  [7] "sq25_C3g" "sq05_C3z" "sq07_C3e" "sq08_C3g"
```

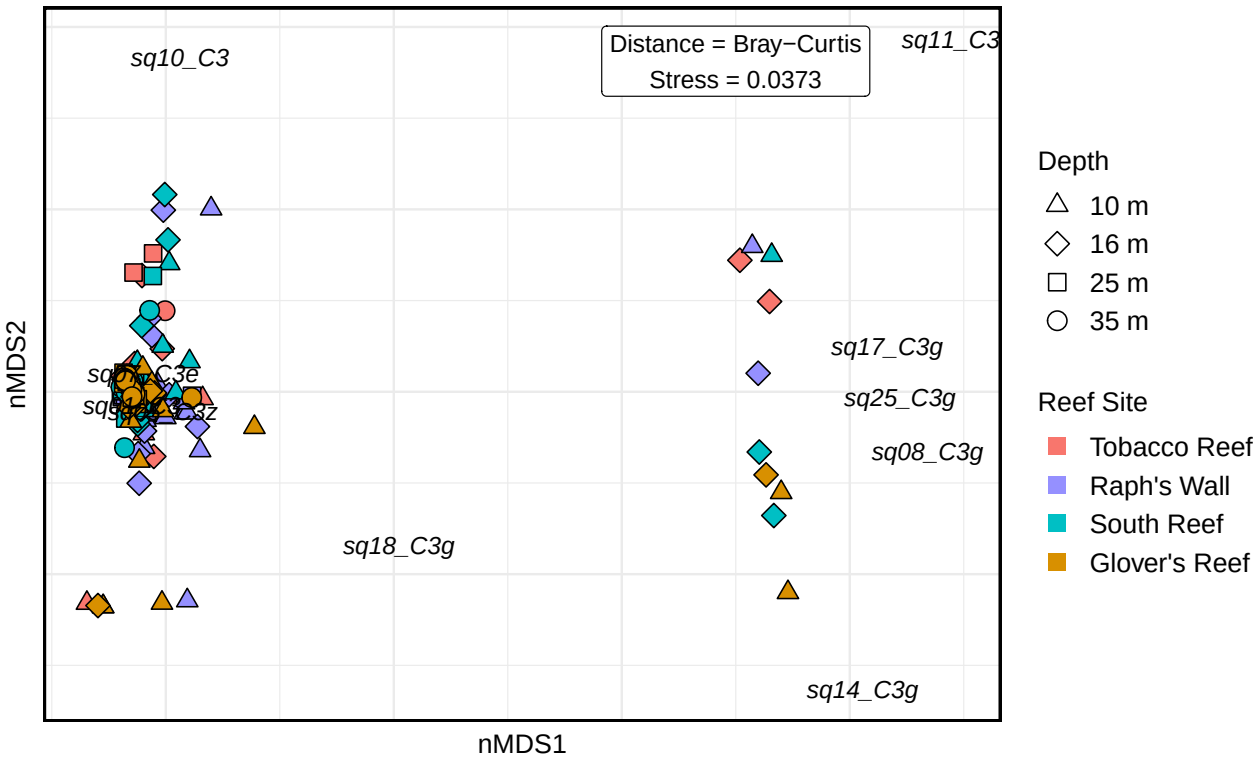
Construct nMDS biplot

```
its2NmdsPlotA = ggplot() +
                geom_point(data = its2Scores, aes(x = NMDS1, y = NMDS2,
                           shape = depth, size = depth, fill = site),
                           color = "black") + # add the site points
                scale_fill_manual(values = its2ColPal[c(8,5,3,9)],
                                  name = "Reef Site", labels = c("Tobacco Reef",
                                  "Raph's Wall", "South Reef", "Glover's Reef")) +
                scale_shape_manual(values = c(24, 23, 22, 21), name = "Depth",
                                   labels = c("10 m", "16 m", "25 m", "35 m")) +
                scale_size_manual(values = c(3, 3.75, 3.75, 3.75)) +
                guides(shape = guide_legend( override.aes = list(size =
                       c(3, 3.75, 3.75, 3.75), fill = "white")), fill =
                         guide_legend(override.aes = list(shape = 22, size = 3.75,
                         color = NA)), size = FALSE)+
                geom_text(data = its2Clades, aes(x = NMDS1, y = NMDS2,
                                                 label = seq),
                            color = "#000000", size = 4, fontface = "italic") +
                          # add seq labels
                annotate("label", x = 1.25, y = 0.725, label = paste
                           ("Distance = Bray-Curtis\nStress = ",
                           round(its2Nmds$stress, 4), sep = ""), size = 4) +
                labs(x = "nMDS1", y = "nMDS2") +
                coord_equal() +
                theme_bw()

its2NmdsPlot = its2NmdsPlotA +
               theme(axis.title.x = element_text(color = "black", size = 12),
                     axis.text.x = element_blank(),
                     axis.ticks.x = element_blank(),
                     axis.title.y = element_text(color = "black", size = 12),
                     axis.text.y = element_blank(),
                     axis.ticks.y = element_blank(),
                     legend.position = "right",
                     legend.title = element_text(color = "black", size = 12),
                     legend.text = element_text(color = "black", size = 12),
```

```
                      legend.key = element_blank(),
                      legend.background = element_blank(),
                      panel.border = element_rect(color = "black", size = 1.2),
                      panel.background = element_rect(fill = "white"),
                      plot.background = element_blank()
                      )

its2NmdsPlot
```

Saving nMDS plot

```
ggsave("nMDS_ITS2_OTU.eps", plot = its2NmdsPlot, width = 8, height = 5, dpi = 600,
       device = "eps")
```



Remove outlying samples

Samples with the C3g types skewing the data: 42, 80, 104, 98, 20, 57, 149, 152, 58, 163 let's look at these samples specifically

```
itsOuts = its2Norm[its2Norm$sample %in% c("42", "80", "104", "98", "20", "57",
                                          "149", "152", "58", "163"), ]
itsOuts$sums = apply(itsOuts[,4:13], 1, function(x) sum(x))
itsNoOut = its2Norm
itsNoOut <- itsNoOut[!itsNoOut$sample %in% c("42", "80", "104", "98", "20", "57",
                                             "149", "152", "58", "163"), ]
itsNoOut$sums <- apply(itsNoOut[,4:13], 1, function(x) sum(x))

itsOuts
```

```
##     sample Site Depth    sq01_C3  sq10_C3 sq11_C3g  sq14_C3g  sq17_C3g
```

```
## 98      98   TR    16   5962.282 2442.380  1152540       0.0 107105.56
## 104    104   TR    16  14559.120 3065.078  1047363       0.0 306539.72
## 42      42   RW    10   3327.416 3104.598  1241319       0.0 200848.17
## 20      20   RW    16  34844.366    0.000        0       0.0 219056.79
## 80      80   SR    10   7562.987 9943.468  2905229       0.0 187462.89
## 57      57   SR    16  59249.888    0.000        0  438911.4 361382.30
## 58      58   SR    16  69160.226    0.000        0  704896.7 155133.96
## 152    152   GR    10   6045.766    0.000        0  560943.6  79987.07
## 163    163   GR    10 171477.140    0.000        0 1164666.6 101868.88
## 149    149   GR    16  14587.886    0.000        0  499950.3 168854.14
##      sq18_C3g  sq25_C3g sq05_C3z sq07_C3e  sq08_C3g    sums
## 98       0.00  61370.79 169218.7 13145.75  884093.8 2395879
## 104      0.00 143260.47 165801.6 61908.19 1215750.4 2958247
## 42       0.00 129932.62 130690.2 38369.26  596691.8 2344283
## 20       0.00 122446.05 383743.7 22680.39 5925715.7 6708487
## 80       0.00 107791.16 164873.1 31417.39  947977.0 4362257
## 57  184315.48 189620.65 346779.9 47641.53 2467484.7 4095386
## 58  312760.62 116130.53 474786.2 20674.75 4183582.7 6037126
## 152  62068.54  53954.48 187677.3 15253.63 1879099.6 2845030
## 163 158405.47 281910.22 621065.3     0.00 5294413.0 7793807
## 149 169574.53  92441.40 243388.7 29973.35 1338843.8 2557614
```

```
max(itsOuts$sum)
```

```
## [1] 7793807
```

```
max(itsNoOut$sum)
```

```
## [1] 1372236
```

```
mean(itsNoOut$sum)
```

```
## [1] 942949
```

```
summary(itsOuts$sum)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 2344283 2629468 3526817 4209812 5618408 7793807
```

```
summary(itsNoOut$sum)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  871864  923180  932594  942949  943096 1372236
```

Nothing unusual here, neither the highest nor the lowest sequencing depth of all samples

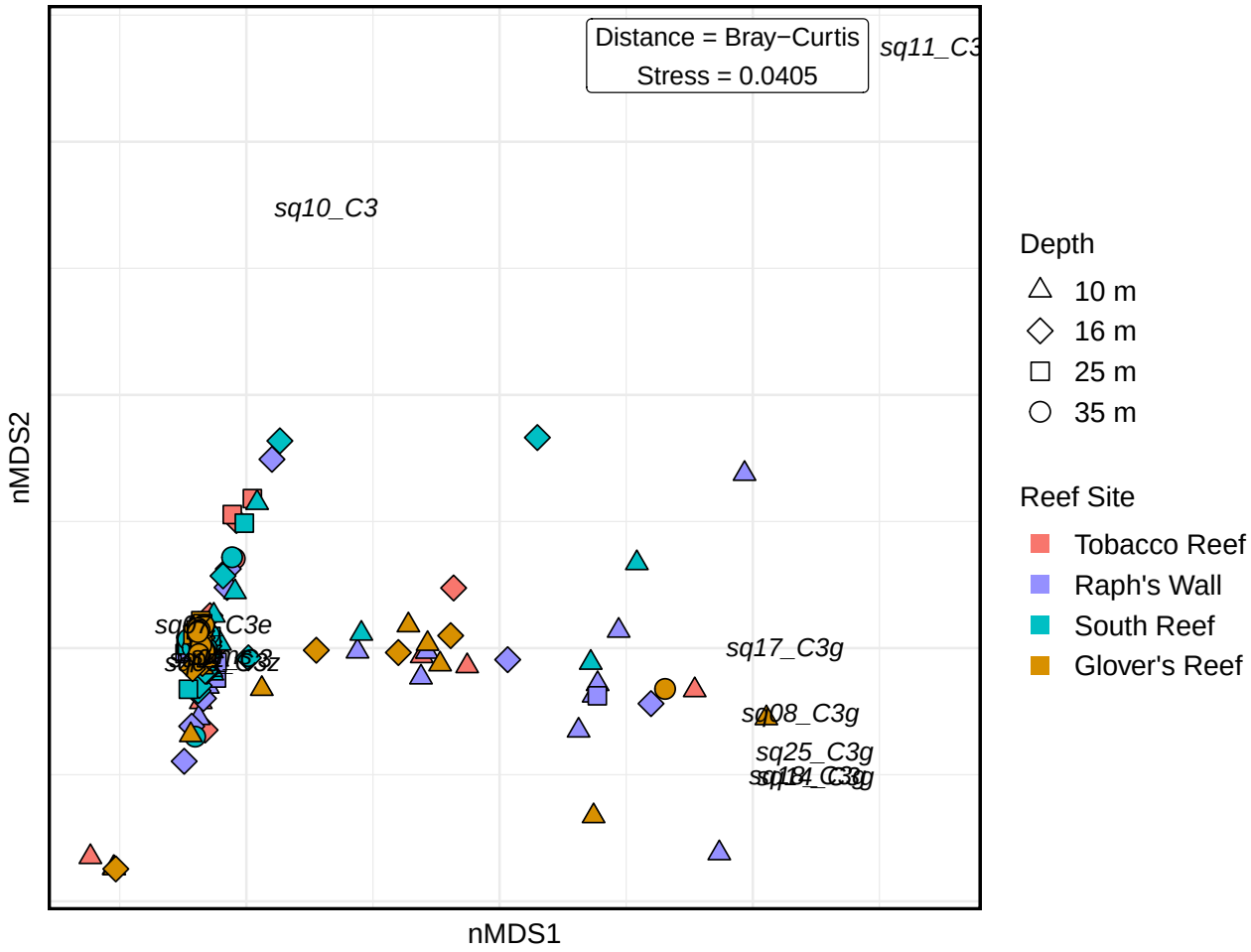
nMDS without outlying samples

Same idea as above, just with the new data lacking 10 C3g dominated samples. This way we can see if these samples are skewing our plot and obscuring a larger pattern in rest of the data.

```
its2Nmds2 = metaMDS(itsNoOut[4:ncol(itsNoOut)], try = 50)
its2Nmds2

its2Scores2 = as.data.frame(scores(its2Nmds2))
its2Scores2$site = factor(itsNoOut$Site)
its2Scores2$depth = as.factor(itsNoOut$Depth)
its2Scores2$sample = row.names(its2Scores2)
head(its2Scores2)
```

```
its2Clades2 = as.data.frame(scores(its2Nmds2,"species"))
its2Clades2$seq = row.names(its2Clades2)
its2Clades2
its2Clades2$seq

## Plot the data
its2NmdsPlotB = ggplot() +
                geom_point(data = its2Scores2, aes(x = NMDS1, y = NMDS2,
                           shape = depth, size = depth, fill = site),
                           color = "black") + # add the site points
                scale_fill_manual(values = its2ColPal[c(8,5,3,9)], name =
                                  "Reef Site",labels = c("Tobacco Reef",
                                  "Raph's Wall", "South Reef", "Glover's Reef")) +
                scale_shape_manual(values = c(24, 23, 22, 21), name = "Depth",
                                   labels = c("10 m", "16 m", "25 m", "35 m")) +
                scale_size_manual(values = c(3, 3.75, 3.75, 3.75)) +
                guides(shape = guide_legend( override.aes = list(size =
                       c(3, 3.75, 3.75, 3.75), fill = "white")), fill =
                       guide_legend(override.aes = list(shape = 22, size = 3.75,
                       color = NA)), size = FALSE)+
                geom_text(data = its2Clades2, aes(x = NMDS1, y = NMDS2,
                          label = seq), color = "#000000", size = 4,
                          fontface = "italic") +
                          # add seq labels
                annotate("label", x = 0.95, y = 1.17, label = paste
                         ("Distance = Bray-Curtis\nStress = ",
                         round(its2Nmds2$stress, 4), sep = ""), size = 4) +
                labs(x = "nMDS1", y = "nMDS2") +
                coord_equal() +
                theme_bw()

its2NmdsPlot2 = its2NmdsPlotB +
                theme(axis.title.x = element_text(color = "black", size = 12),
                      axis.text.x = element_blank(),
                      axis.ticks.x = element_blank(),
                      axis.title.y = element_text(color = "black", size = 12),
                      axis.text.y = element_blank(),
                      axis.ticks.y = element_blank(),
                      legend.position = "right",
                      legend.title = element_text(color = "black", size = 12),
                      legend.text = element_text(color = "black", size = 12),
                      legend.key = element_blank(),
                      legend.background = element_blank(),
                      panel.border = element_rect(color = "black", size = 1.2),
                      panel.background = element_rect(fill = "white"),
                      plot.background = element_blank()
                      )

its2NmdsPlot2
```

```
# saving new nMDS plot
ggsave("nMDS_ASVs_noOut.eps", plot = its2NmdsPlot2, width = 9.2, height = 5.75,
       dpi = 600, device = "eps")
```



Here we can see that there is not a larger pattern in the data that was being masked by the C3g dominant samples

Relative abundance of Symbiodiniaceae OTUs

Now we can calculate the relative abundance of each unique OTU per sample. This will allow us to view the community structure in a faceted barplot.

Caluculate OTU relative abundances

```
head(its2Norm)
```

```
##     sample Site Depth  sq01_C3 sq10_C3 sq11_C3g sq14_C3g sq17_C3g sq18_C3g
## 108    108   TR    10 830134.8       0        0  0.00000  0.00000    0.000
## 109    109   TR    10 828909.5       0        0  0.00000  0.00000    0.000
## 110    110   TR    10 826199.7       0        0 25.04951 12.52476 2580.100
## 111    111   TR    10 826360.2       0        0  0.00000  0.00000    0.000
## 112    112   TR    10 826546.9       0        0  0.00000  0.00000    0.000
## 113    113   TR    10 825665.6       0        0 29.63533 63.50427 3496.969
##      sq25_C3g sq05_C3z sq07_C3e  sq08_C3g
## 108  0.000000 68607.55 18251.16   0.00000
## 109  0.000000 70578.82 13438.77   0.00000
```

```
## 110  4.174918 83289.62 33077.88 150.29706
## 111  0.000000 74603.14 39432.63   0.00000
## 112  0.000000 82377.62 29126.13   0.00000
## 113 59.270653 98321.55 26426.24  88.90598
```

```
its2NormPerc = its2Norm
its2NormPerc$sum = apply(its2NormPerc[, c(4:length(its2NormPerc[1,]))], 1, function(x) {
  sum(x, na.rm = T)
})

its2NormPerc = cbind(its2NormPerc[, c(1:3)], (its2NormPerc[,
                    c(4:(ncol(its2NormPerc)-1))] / its2NormPerc$sum))
head(its2NormPerc)
```

```
##     sample Site Depth   sq01_C3 sq10_C3 sq11_C3g     sq14_C3g     sq17_C3g
## 108    108   TR    10 0.9052788       0        0 0.000000e+00 0.000000e+00
## 109    109   TR    10 0.9079690       0        0 0.000000e+00 0.000000e+00
## 110    110   TR    10 0.8739716       0        0 2.649790e-05 1.324895e-05
## 111    111   TR    10 0.8787364       0        0 0.000000e+00 0.000000e+00
## 112    112   TR    10 0.8811325       0        0 0.000000e+00 0.000000e+00
## 113    113   TR    10 0.8653400       0        0 3.105935e-05 6.655574e-05
##        sq18_C3g     sq25_C3g   sq05_C3z   sq07_C3e     sq08_C3g
## 108 0.000000000 0.000000e+00 0.07481792 0.01990326 0.000000e+00
## 109 0.000000000 0.000000e+00 0.07731046 0.01472053 0.000000e+00
## 110 0.002729284 4.416317e-06 0.08810553 0.03499048 1.589874e-04
## 111 0.000000000 0.000000e+00 0.07933163 0.04193194 0.000000e+00
## 112 0.000000000 0.000000e+00 0.08781788 0.03104964 0.000000e+00
## 113 0.003665003 6.211869e-05 0.10304603 0.02769606 9.317804e-05
```

Now a quick sanity check. If this worked the sum of each row should = 100% (i.e. "1")

```
# test that all are now 100% = 1
apply(its2NormPerc[, c(4:(ncol(its2NormPerc)))], 1, function(x) {
  sum(x, na.rm = T)
})
```

```
## 108 109 110 111 112 113 114 116 117 118 119 120 121 156  93  94  95  96
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  97  98  99 100 101 102 103 104 105 106 137 122 123 124 125 126 127 128
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 129 130 131 132 133 134 135 136 226 227 228 229 230 231 232 233 234 236
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 237 238 239 240  29  30  31  32  33  34  35  36  37  38  39  40  41  42
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  43  17  18  19  20  21  22  23  24  25  26  27  28  45  46   1   2   3
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##   4   5   6   7   8   9  10  11  12  13  14  15  16 211 213 214 215 216
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 217 218 219 221 222 223 224 225  76  77  78  79  80  81  82  83  84  85
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  86  87  88  89  90  91  47  48  49  50  51  52  53  54  55  56  57  58
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  59  60  92  61  62  63  64  65  66  68  69  70  71  72  73  74  75 198
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 199 200 201 202 203 204 205 206 207 208 209 210 241 242 115 152 153 154
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
## 155 157 158 159 160 161 162 163 164 165 166 148 149 150 151 176 177 178
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 179 180 181 182 183 187 188 145 146 147 167 168 169 170 171 172 173 174
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 175 184 185 138 139 140 141 142 143 144 190 191 192 193 194 195 196
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

Everything adds up to 1, this is good! The code works.

Add plot order to data frame

I added an additional column to sort better for the stacked barplot. This was just a work around to get the `facet_grid()` function to play nice with our data. I added a coulumn "barPlotOrder" and for each population I filled in a series 1:n foreach sample at each Site:Depth combo, so now there's no large blank expanses on the plot.

```
its2ra = its2NormPerc
sampleCounts = plyr::count(its2ra, c('Site','Depth'))
meltedList = melt(lapply(sampleCounts$freq,function(x){c(1:x)}))
its2ra$barPlotOrder = meltedList$value
colnames(its2ra)[c(5:ncol(its2ra)-1)] = c("sq01_C3", "sq10_C3", "sq11_C3g","sq14_C3g", "sq17_C3g",
                                          "sq18_C3g", "sq25_C3g", "sq05_C3z", "sq07_C3e",
                                          "sq08_C3g")
its2ra=its2ra[c(1,ncol(its2ra),2:(ncol(its2ra)-1))]
head(its2ra)
```

```
##     sample barPlotOrder Site Depth   sq01_C3 sq10_C3 sq11_C3g     sq14_C3g
## 108    108            1   TR    10 0.9052788       0        0 0.000000e+00
## 109    109            2   TR    10 0.9079690       0        0 0.000000e+00
## 110    110            3   TR    10 0.8739716       0        0 2.649790e-05
## 111    111            4   TR    10 0.8787364       0        0 0.000000e+00
## 112    112            5   TR    10 0.8811325       0        0 0.000000e+00
## 113    113            6   TR    10 0.8653400       0        0 3.105935e-05
##         sq17_C3g    sq18_C3g     sq25_C3g   sq05_C3z   sq07_C3e
## 108 0.000000e+00 0.000000000 0.000000e+00 0.07481792 0.01990326
## 109 0.000000e+00 0.000000000 0.000000e+00 0.07731046 0.01472053
## 110 1.324895e-05 0.002729284 4.416317e-06 0.08810553 0.03499048
## 111 0.000000e+00 0.000000000 0.000000e+00 0.07933163 0.04193194
## 112 0.000000e+00 0.000000000 0.000000e+00 0.08781788 0.03104964
## 113 6.655574e-05 0.003665003 6.211869e-05 0.10304603 0.02769606
##         sq08_C3g
## 108 0.000000e+00
## 109 0.000000e+00
## 110 1.589874e-04
## 111 0.000000e+00
## 112 0.000000e+00
## 113 9.317804e-05
```

Create OTU stack for plotting

Stack the OTU data and adjest our columns and factor names for plotting

```
gss = otuStack(its2ra, count.columns = c(5:length(its2ra[1, ])),
               condition.columns = c(1:4))[1:2330,] # remove summ rows

levels(gss$otu)
```

```
##  [1] "summ"     "sq01_C3"  "sq10_C3"  "sq11_C3g" "sq14_C3g" "sq17_C3g"
##  [7] "sq18_C3g" "sq25_C3g" "sq05_C3z" "sq07_C3e" "sq08_C3g"
```

```
gss$otu = factor(gss$otu, levels(gss$otu)[c(1:8, 11, 9, 10)])
levels(gss$otu)
```

```
##  [1] "summ"     "sq01_C3"  "sq10_C3"  "sq11_C3g" "sq14_C3g" "sq17_C3g"
##  [7] "sq18_C3g" "sq25_C3g" "sq08_C3g" "sq05_C3z" "sq07_C3e"
```

```
levels(gss$Depth)
```

```
## [1] "10" "16" "25" "35"
```

```
levels(gss$Depth) = c("10 m", "16 m", "25 m", "35 m")
levels(gss$Site)
```

```
## [1] "TR" "RW" "SR" "GR"
```

```
levels(gss$Site) = c("Tobacco Reef", "Raph's Wall", "South Reef", "Glover's Reef")
levels(gss$Depth)
```

```
## [1] "10 m" "16 m" "25 m" "35 m"
```

```
levels(gss$Site)
```

```
## [1] "Tobacco Reef"  "Raph's Wall"   "South Reef"    "Glover's Reef"
```

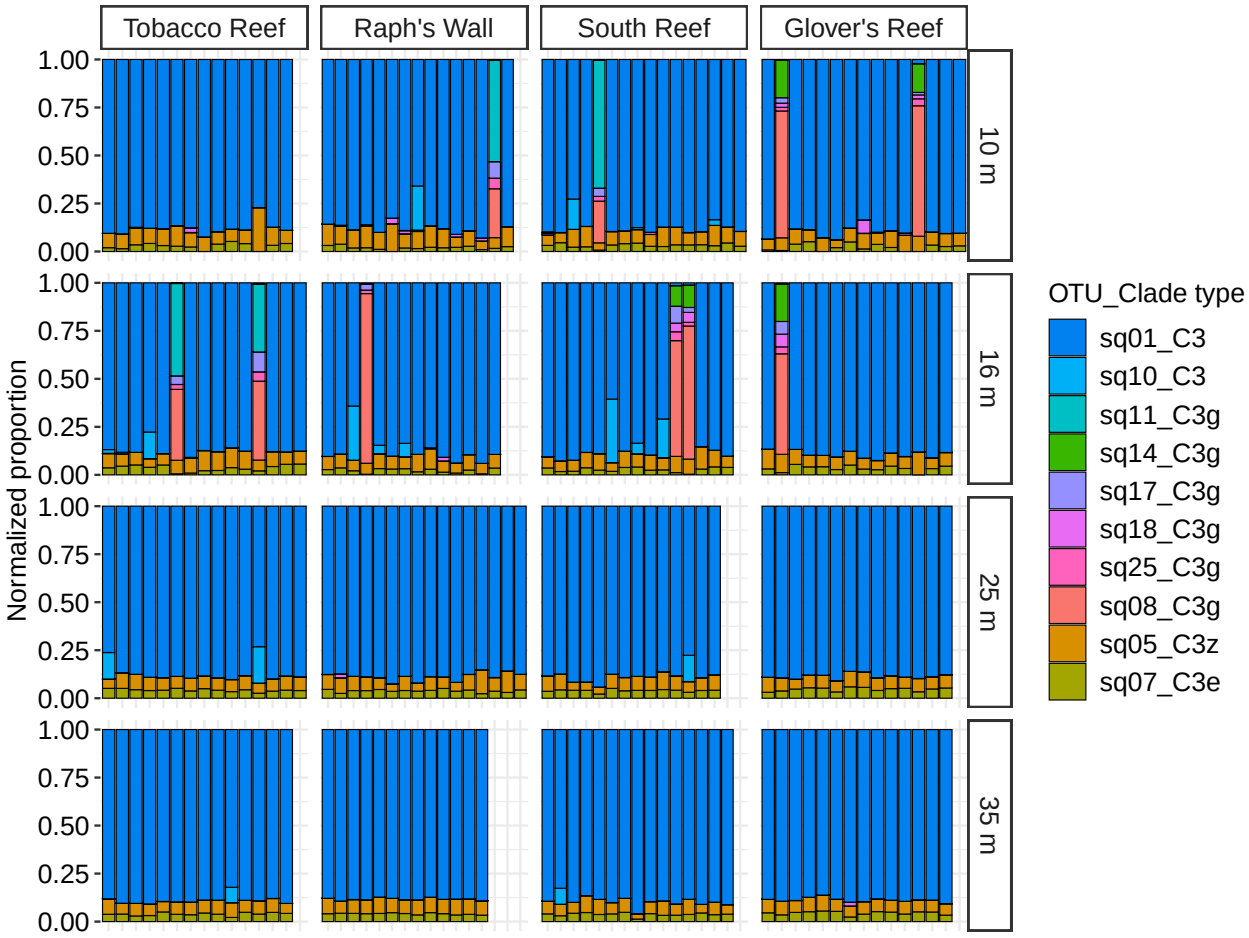
Consruct OTU barplot

```
OTUplotA = ggplot(gss, aes(x = barPlotOrder, y = count, fill = factor(otu))) +
          geom_bar(position = "stack", stat = "identity", color = "black",
                   size = 0.25) +
            ylab("Normalized proportion") +
            scale_fill_manual(values=its2ColPal)+
            labs(fill = "OTU_Clade type") +
            facet_grid(Depth ~ Site, scales = "free_x") + #faceting plots by Depth and Site
            theme_bw()

OTUplot = OTUplotA +
          theme(axis.title.x = element_blank(),
                axis.text.x = element_blank(),
                axis.ticks.x = element_blank(),
                axis.title.y = element_text(color = "black", size = 12),
                axis.text.y = element_text(color = "black", size = 12),
                legend.position = "right",
                legend.title = element_text(color = "black", size = 12),
                legend.text = element_text(color = "black", size = 12),
                legend.key = element_blank(),
                legend.background = element_blank(),
                panel.border = element_blank(),
                panel.background = element_rect(fill = "white"),
                plot.background = element_blank(),
                strip.text.x = element_text(size = 12),
                strip.text.y = element_text(size = 12),
                strip.background = element_rect(fill = "white", size = 0.9)
                )
```

```
OTUplot
```

Save the OTU barplot

```
ggsave("OTUbarPlot2.eps", plot = OTUplot, width = 8, height = 6, unit = "in",
       dpi = 600)
```



ITS2 community differences

Now we can use permutational multivariate analysis of variance (PERMANOVA) to test for differences in Symbiodiniaceae β-diversity across Site and Depth based on our OTUs.

Cheking dispersion

Using `betadisper()` in *vegan* to look at multivariate homogeneity of dispersion between sites and depths. This is using Bray-Curtis similarity.

```
anova(betadisper(its2Dist, its2Norm$Depth))
```

```
## Analysis of Variance Table
##
## Response: Distances
```

```
##            Df Sum Sq  Mean Sq F value   Pr(>F)
## Groups      3 0.5242 0.174742  5.0941 0.001971 **
## Residuals 229 7.8553 0.034303
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dispersion is heteroschedastic, but PERMANOVA is robust to deviations in homgeneity of variance (Anderson and Walsh, 2013; https://esajournals.onlinelibrary.wiley.com/doi/10.1890/12-2010.1).

Running PERMANOVA in R

Now let's see how different communities are from each other with PERMANOVA. We will utilize the `adonis()` function in *vegan*. We will use Bray-Curtis similarity for our distance matrix and run a total 0f 9,999 permutations, and test the effects of Site, Depth, and the interaction between Site and Depth.

```
its2Adonis = adonis(its2Norm[, c(4:ncol(its2Norm))] ~ Depth*Site,
                    data = its2Norm, permutations = 9999, method = "bray")
its2Adonis
```

```
##
## Call:
## adonis(formula = its2Norm[, c(4:ncol(its2Norm))] ~ Depth * Site,      data = its2Norm, permutations =
##
## Permutation: free
## Number of permutations: 9999
##
## Terms added sequentially (first to last)
##
##             Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
## Depth        3    0.3946 0.131523  3.4741 0.04443 0.0082 **
## Site         3    0.0349 0.011633  0.3073 0.00393 0.9114
## Depth:Site   9    0.2364 0.026268  0.6938 0.02662 0.7949
## Residuals  217    8.2153 0.037859         0.92502
## Total      232    8.8812                  1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that **Depth** has a signifcant effect on Symbiodiniaceae community structure in our *M. cavernosa* samples.

Pairwise PERMANOVA for multiple comparisons

Since we found that Depth was a significant factor in our PERMANOVA we can now use pairwise PERMANOVA to reveal where differences occur across depth. This utilizes the package *pairwiseAdonis*, where we will again use Bray-Curtis similarity and 9,999 permutations. We also have added false discovery rate (FDR) corrections since we are perfoming multiple comparisons.

```
its2PWAdonis = pairwise.adonis(its2Norm[, c(4:ncol(its2Norm))],
                   factors = its2Norm$Depth, sim.method = "bray",
                   p.adjust.m = "BH", perm = 9999)
its2PWAdonis
```

```
##      pairs Df    SumsOfSqs   F.Model          R2 p.value p.adjusted sig
## 1 10 vs 16  1 3.791992e-02 0.5212379 0.004473330  0.5278    0.63336
## 2 10 vs 25  1 1.215016e-01 4.0807976 0.033703094  0.0003    0.00180   *
## 3 10 vs 35  1 1.209040e-01 3.9840789 0.033767936  0.0026    0.00580   *
## 4 16 vs 25  1 2.553940e-01 5.8424234 0.048347453  0.0029    0.00580   *
```

```
## 5 16 vs 35  1 2.551471e-01 5.7118059 0.048523645  0.0095    0.01425   .
## 6 25 vs 35  1 7.599081e-05 0.1802355 0.001592465  0.8247    0.82470
```

We see that again see differences between our deeper (25 + 35 m) and shallower (10 + 16 m) samples.

PERMANOVA without 35 m samples

Subset our dataframe

First we need to remove the deep samples from the dataframe. We will use our dataframe of good OTUs, which haven't been normalized yet. This way we can calculate the normalization based on only the samples we are keeping in the analysis.

```
goods2 = subset(goods, !Depth=="35")
goods2[] = lapply(goods2, function(x) if(is.factor(x)) factor(x) else x)
summary(goods2)
```

```
##     newOrder         sample       Site    Depth        sq1
##  Min.   :  1.0   Min.   :  1.00   TR:44   10:60   Min.   :   224
##  1st Qu.: 59.0   1st Qu.: 46.00   RW:45   16:58   1st Qu.: 90703
##  Median :103.0   Median : 91.00   SR:45   25:59   Median :132450
##  Mean   :109.6   Mean   : 92.51   GR:43           Mean   :132460
##  3rd Qu.:160.0   3rd Qu.:136.00                   3rd Qu.:176321
##  Max.   :219.0   Max.   :188.00                   Max.   :430489
##       sq10            sq11            sq14            sq17
##  Min.   :    0   Min.   :     0   Min.   :    0   Min.   :    0.0
##  1st Qu.:    0   1st Qu.:     0   1st Qu.:    0   1st Qu.:    0.0
##  Median :    0   Median :     0   Median :    0   Median :    0.0
##  Mean   : 1739   Mean   :  1592   Mean   :  504   Mean   :  363.1
##  3rd Qu.:    0   3rd Qu.:     0   3rd Qu.:    0   3rd Qu.:    0.0
##  Max.   :53422   Max.   :117162   Max.   :28206   Max.   :13521.0
##       sq18            sq25           sq5            sq7
##  Min.   :   0.0   Min.   :   0   Min.   :  935   Min.   :    0
##  1st Qu.:   0.0   1st Qu.:   0   1st Qu.: 7391   1st Qu.: 2389
##  Median :   0.0   Median :   0   Median :11115   Median : 4938
##  Mean   : 370.1   Mean   : 229   Mean   :11759   Mean   : 5134
##  3rd Qu.:   0.0   3rd Qu.:   0   3rd Qu.:14646   3rd Qu.: 7149
##  Max.   :9958.0   Max.   :8747   Max.   :56300   Max.   :15419
##       sq8
##  Min.   :     0
##  1st Qu.:     0
##  Median :     0
##  Mean   :  3870
##  3rd Qu.:     0
##  Max.   :169042
```

Normalize reads

We will normalize samples again, as we did with the entire dataframe orignially

```
itsgoods2Transposed = t(goods2[, 5:length(goods2[1, ])])
itsgoods2List = DGEList(counts = itsgoods2Transposed)
head(itsgoods2List$samples)
```

```
##     group lib.size norm.factors
## 108     1   237951            1
## 109     1    18070            1
```

```
## 110     1   226433            1
## 111     1   204975            1
## 112     1    27762            1
## 113     1   225375            1
```

```
its2Norm2 =  calcNormFactors(itsgoods2List, method = "TMM")
head(its2Norm2$samples)
```

```
##     group lib.size norm.factors
## 108     1   237951     1.080109
## 109     1    18070     1.084610
## 110     1   226433     1.047130
## 111     1   204975     1.052500
## 112     1    27762     1.055361
## 113     1   225375     1.037930
```

```
its2TMM = t(cpm(its2Norm2, Normalized.lib.sizes = TRUE))
its2Norm2 = cbind(goods2[,c(2:4)], its2TMM)
head(its2Norm2)
```

```
##     sample Site Depth      sq1 sq10 sq11     sq14     sq17     sq18
## 108    108   TR    10 838136.9    0    0  0.00000  0.00000    0.000
## 109    109   TR    10 837139.0    0    0  0.00000  0.00000    0.000
## 110    110   TR    10 834635.2    0    0 25.30527 12.65263 2606.443
## 111    111   TR    10 834904.0    0    0  0.00000  0.00000    0.000
## 112    112   TR    10 834911.1    0    0  0.00000  0.00000    0.000
## 113    113   TR    10 833716.8    0    0 29.92431 64.12352 3531.068
##          sq25      sq5      sq7       sq8
## 108  0.000000 69268.90 18427.09   0.00000
## 109  0.000000 71279.53 13572.19   0.00000
## 110  4.217545 84140.02 33415.61 151.83161
## 111  0.000000 75374.47 39840.33   0.00000
## 112  0.000000 83211.23 29420.87   0.00000
## 113 59.848614 99280.30 26683.93  89.77292
```

Run PERMANOVA on subset of data

```
its2Dist2 = vegdist(its2Norm2[, c(4:ncol(its2Norm2))], method = "bray")
anova(betadisper(its2Dist2, its2Norm2$Depth))
```

```
## Analysis of Variance Table
##
## Response: Distances
##            Df Sum Sq  Mean Sq F value  Pr(>F)
## Groups      2 0.3443 0.172163  3.9313 0.02138 *
## Residuals 174 7.6199 0.043793
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
its2Adonis2 <- adonis(its2Norm2[, c(4:ncol(its2Norm2))] ~ Depth*Site,
                      data = its2Norm2, permutations = 9999, method = "bray")
its2Adonis2
```

```
##
## Call:
## adonis(formula = its2Norm2[, c(4:ncol(its2Norm2))] ~ Depth *      Site, data = its2Norm2, permutatio
```

```
##
## Permutation: free
## Number of permutations: 9999
##
## Terms added sequentially (first to last)
##
##             Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
## Depth        2    0.2828 0.141396 2.92686 0.03326 0.0368 *
## Site         3    0.0367 0.012230 0.25316 0.00432 0.9386
## Depth:Site   6    0.2111 0.035186 0.72835 0.02483 0.6873
## Residuals  165    7.9711 0.048310         0.93759
## Total      176    8.5017                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Depth is still has a significant effect on community structure

Pairwise PERMANOVA

Let' see where differences lie across depth using pairwise permanova again.

```
its2PWAdonis2 = pairwise.adonis(its2Dist2, factors = its2Norm2$Depth,
                                perm = 9999, p.adjust.m = "BH")
its2PWAdonis2
```

```
##      pairs Df  SumsOfSqs   F.Model          R2 p.value p.adjusted sig
## 1 10 vs 16  1 0.03164863 0.4486379 0.003852667  0.6030    0.60300
## 2 10 vs 25  1 0.13184809 4.5463670 0.037404384  0.0001    0.00030  **
## 3 16 vs 25  1 0.26179196 6.1925677 0.051096926  0.0023    0.00345   *
```

The differences are still between 25 m and the shallower sites (10 + 16 m). This gives us confidence that our deeper samples aren't changing our interpretation.

Generalized linear mixed model of OTUs

To see how significantly changing OTUs across depth we will use the package *MCMC.OTU*, which performs Bayesian analysis of multivariate count data. This will create a MCMC-based GLMM analysis of our OTU data.

Stack OTU data table

```
its2Glm = its2Norm
colnames(its2Glm)[4:ncol(its2Glm)] = c("sq01_C3", "sq10_C3", "sq11_C3g",
                                       "sq14_C3g", "sq17_C3g", "sq18_C3g",
                                       "sq25_C3g", "sq05_C3z", "sq07_C3e",
                                       "sq08_C3g")
glmStack=otuStack(data=its2Glm, count.columns = c(4:ncol(its2Glm)),
                  condition.columns=c(1:3))

levels(glmStack$otu)
```

```
##  [1] "summ"     "sq01_C3"  "sq10_C3"  "sq11_C3g" "sq14_C3g" "sq17_C3g"
##  [7] "sq18_C3g" "sq25_C3g" "sq05_C3z" "sq07_C3e" "sq08_C3g"
```

```
glmStack$otu = factor(glmStack$otu, levels(glmStack$otu)[c(1:8, 11, 9, 10)])
levels(glmStack$otu)
```

```
##  [1] "summ"     "sq01_C3"  "sq10_C3"  "sq11_C3g" "sq14_C3g" "sq17_C3g"
##  [7] "sq18_C3g" "sq25_C3g" "sq08_C3g" "sq05_C3z" "sq07_C3e"
```

```
glmStack$count = round(glmStack$count, 0)
head(glmStack)
```

```
##    count     otu sample Site Depth
## 1 830135 sq01_C3    108   TR    10
## 2 828910 sq01_C3    109   TR    10
## 3 826200 sq01_C3    110   TR    10
## 4 826360 sq01_C3    111   TR    10
## 5 826547 sq01_C3    112   TR    10
## 6 825666 sq01_C3    113   TR    10
```

Creating the model

Now that we have our data setup correctly we can create the GLMM

Fitting the model

Here we are using Depth as a fixed effect and running 10^6 iterations with a 5000 burn-in period.

```
its2MmD = mcmc.otu(fixed = "Depth", data = glmStack,
                   nitt=100000, thin=25, burnin=5000)
```

Calculate effect size and *p*-values

```
ssD = OTUsummary(its2MmD, glmStack, summ.plot = FALSE)
ssD = padjustOTU(ssD)
sigsD = signifOTU(ssD)

ss2D = OTUsummary(its2MmD, glmStack, otus = sigsD, whiskers = "sd",
                  ptype = "mcmc")
```

```
its2MCMCglmD = ss2D$summary
levels(its2MCMCglmD$otu)
```

```
## [1] "sq01_C3"  "sq07_C3e" "sq08_C3g" "sq10_C3"  "sq14_C3g" "sq17_C3g"
## [7] "sq18_C3g" "sq25_C3g"
```

```
its2MCMCglmD$Depth = as.factor(its2MCMCglmD$Depth)
its2MCMCglmD$otu = factor(its2MCMCglmD$otu, levels = c("sq01_C3", "sq10_C3",
                          "sq14_C3g", "sq17_C3g", "sq18_C3g", "sq25_C3g",
                          "sq08_C3g", "sq07_C3e"))

ssD$otuWise[sigsD]
```

```
## $sq01_C3
##       difference
## pvalue         10           16        25         35
##     10         NA -0.061342676 0.1662458 0.16790133
##     16 0.49373133           NA 0.2275885 0.22924401
##     25 0.03691520  0.005191352        NA 0.00165554
##     35 0.03737995  0.005401985 0.9983354         NA
##
## $sq10_C3
```

```
##       difference
## pvalue        10         16         25        35
##     10        NA 2.09427370 -3.6073627 -5.064233
##     16 0.3776807         NA -5.7016364 -7.158507
##     25 0.2228695 0.03645170         NA -1.456871
##     35 0.1137648 0.02472841  0.7422496        NA
##
## $sq14_C3g
##       difference
## pvalue          10          16         25         35
##     10          NA -2.23508519 -7.0628806 -6.8949807
##     16 0.040969131          NA -4.8277954 -4.6598955
##     25 0.001607853  0.02499062         NA  0.1678999
##     35 0.001178064  0.02313775  0.9983354         NA
##
## $sq17_C3g
##       difference
## pvalue          10           16         25         35
##     10          NA -1.388122781 -7.5474321 -7.4377685
##     16 0.206788675           NA -6.1593093 -6.0496457
##     25 0.001245135  0.006000357         NA  0.1096636
##     35 0.001241094  0.006041373  0.9983354         NA
##
## $sq18_C3g
##       difference
## pvalue          10          16        25           35
##     10          NA -3.16142817 -9.626732 -9.632088918
##     16 0.024865893          NA -6.465303 -6.470660752
##     25 0.000949371  0.01900819        NA -0.005357278
##     35 0.001178064  0.02169629  0.998571           NA
##
## $sq25_C3g
##       difference
## pvalue          10           16         25          35
##     10          NA -1.272909432 -7.0865630 -7.01764595
##     16 0.236396410           NA -5.8136536 -5.74473652
##     25 0.001245135  0.006000357         NA  0.06891705
##     35 0.001477129  0.007169117  0.9983354          NA
##
## $sq08_C3g
##       difference
## pvalue          10           16         25          35
##     10          NA -1.799719114 -8.7903141 -8.69496991
##     16 0.152251158           NA -6.9905950 -6.89525079
##     25 0.000949371  0.004565093         NA  0.09534424
##     35 0.000949371  0.004776074  0.9983354          NA
##
## $sq07_C3e
##       difference
## pvalue           10          16        25          35
##     10           NA 0.34735180 0.6075924  0.59452579
##     16 0.0190342629          NA 0.2602406  0.24717399
##     25 0.0001186316 0.07833303        NA -0.01306663
##     35 0.0001186316 0.10555832 0.9983354          NA
```

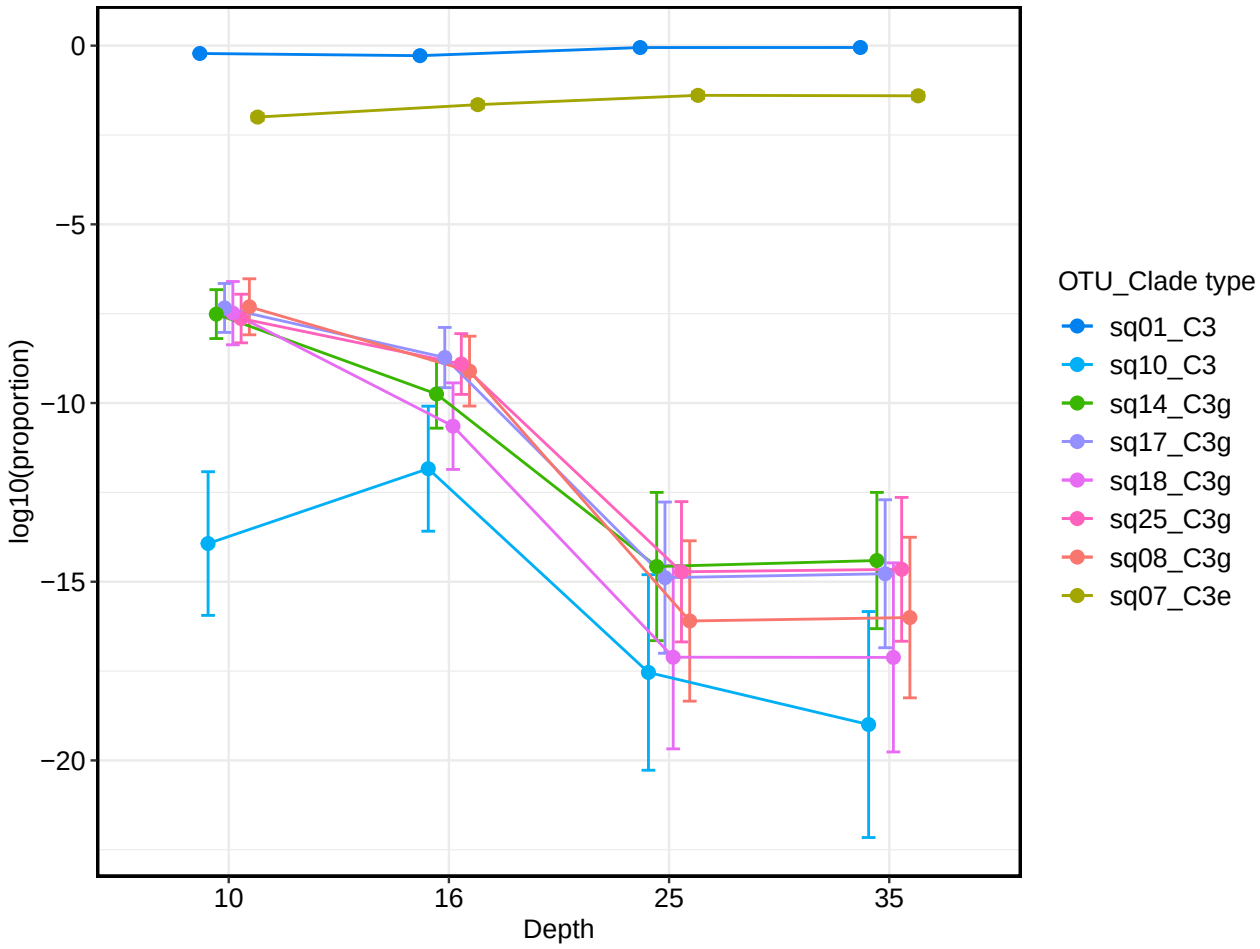
Plotting the GLMM

We are now plotting the GLMM using our color pallete we created for the OTUs

```
pd = position_dodge(0.3)

its2glmPlotA = ggplot(its2MCMCglmD, aes(x = Depth, y = mean, group = otu,
                       colour =otu))+
                       geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd),
                                     lwd = 0.6, width = 0.5,     position = pd) +
                       geom_line(aes(group = otu), lwd = 0.6, position = pd)+
                       geom_point(aes(group = otu), position = pd, size = 2.5)+
                       scale_color_manual(values = its2ColPal[c(1,2,4:8,10)],
                                          name = "OTU_Clade type") +
                       xlab("Depth")+
                       ylab("log10(proportion)") +
                       theme_bw()

its2glmPlotD = its2glmPlotA +
                theme(axis.title.x = element_text(color = "black", size = 12),
                      axis.text.x = element_text(color = "black", size = 12),
                      axis.title.y = element_text(color = "black", size = 12),
                      axis.text.y = element_text(color = "black", size = 12),
                      legend.position = "right",
                      legend.title = element_text(color = "black", size = 12),
                      legend.text = element_text(color = "black", size = 12),
                      legend.key = element_blank(),
                      legend.background = element_blank(),
                      panel.border = element_rect(color = "black", size = 1.2),
                      panel.background = element_rect(fill = "white"),
                      plot.background = element_blank()
                      )

its2glmPlotD
```

Save GLMM plot

```
ggsave("its2glmDepth.eps", plot=its2glmPlotD, width = 8, height = 6, dpi = 600)
```


0
−5
−10
−15
−20
log10(proportion)
10
16
25
35
Depth
OTU_Clade type
sq01_C3
sq10_C3
sq14_C3g
sq17_C3g
sq18_C3g
sq25_C3g
sq08_C3g
sq07_C3e