Lake Erie HABs Community Ecology Manuscript

Contents

Load libraries

```r
library(phyloseq)
library(DESeq2)
library(ggplot2)
library(dplyr)
library(scales)
library(grid)
library(reshape2)
library(gridExtra)
library(vegan)
library(cowplot)
library(gtable)
library(pander)
library(tidyr)
```

Global

```r
## objects and functions that will be useful throughout this analysis

# Set the ggplot theme
theme_set(theme_bw())

# Color palette for stations
station_colors = c("red", "#ffa500", "#0080ff")

# Function to order date levels correctly
order_dates <- function(df) {
  df$Date <- factor(df$Date,
    levels = c("6/16","6/30","7/8","7/14","7/21",
      "7/29","8/4","8/11","8/18","8/25","9/2","9/8","9/15",
      "9/23","9/29","10/6","10/15","10/20","10/27"))
  return(df)
}

named_list <- function(...){
    names <- as.list(substitute(list(...)))[-1L]
    result <- list(...)
    names(result) <- names
    result
}

# Source some useful functions for data normalizatio
source("~/git_repos/MicrobeMiseq/R/miseqR.R")
```

Load Data

```
load("erie-data.RData")

# Inspect erie phyloseq object
erie
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 7192 taxa and 53 samples ]
## sample_data() Sample Data:       [ 53 samples by 32 sample variables ]
## tax_table()   Taxonomy Table:    [ 7192 taxa by 9 taxonomic ranks ]
```

Normalization

```
depth = 15000
thresh = 0.0001

# Scale reads in OTU table to even depth
erie_scale <- erie %>%
  scale_reads(n = depth, round = "round")


# Prune low abundance taxa using thresh as mean relative abundance
tax_mean <- taxa_sums(erie_scale)/nsamples(erie_scale)
erie_scale_0001 <- prune_taxa(tax_mean > thresh*depth, erie_scale)
```

Figure 1: Bloom temporal dynamics

```
# Import metadata file with nutrients, pigments and toxin
nutrient <- read.csv("other/nutrient_cleaned.csv")

# Format nutrient data
nutrient_sub <-
  nutrient %>%
    filter(!(Date %in% c("5/27", "6/10", "11/3"))) %>%
    order_dates()

# Calculate relative abundance of Cyanobacteria at each date
cyano_abundance <-
  erie_scale %>%
    tax_glom(taxrank = "Phylum") %>%                        # conglomerate OTUs to phylum level
    transform_sample_counts(function(x) {x/sum(x)} ) %>%    # transform to relative abundance
    subset_taxa(Phylum == "Cyanobacteria") %>%              # Subset to just Cyanobacteria
    psmelt() %>%                                            # melt phyloseq object
    rename(Cyanobacteria = Abundance) %>%
    select(Cyanobacteria, Date, Station)
```

```r
# Merge cyanobacteria data with nutrient df
bloom_df <-
  nutrient_sub %>%
    left_join(cyano_abundance, by = c("Station", "Date")) %>%
    mutate(Phycocyanin = ifelse(Phycocyanin > 80, 80, Phycocyanin)) %>%   # lower extreme values to plo
    select(Station, Date, Phycocyanin, Chla, ParMC, Cyanobacteria) %>%
    melt(id.vars = c("Station", "Date")) %>%
    order_dates()

# Make a faceted ggplot of the four bloom variables over time and grouped by station
bloom_plots <- ggplot(bloom_df,
  aes(x = Date, y = value, group = Station, color = Station, shape = Station)
) +
  facet_grid(variable~., scales = "free_y") +
  geom_point(size = 1.3) +
  geom_line(size = 1) +
  ylab("") +
  scale_x_discrete(
    breaks = c("7/8", "8/4", "9/2", "10/6"),
    labels = c("Jul", "Aug", "Sep", "Oct"),
    drop = FALSE
  ) +
  scale_color_manual(values = station_colors) +
  theme(
    strip.background = element_blank(),
    strip.text = element_text(size = 11),
    axis.title.x = element_blank()
  )

# function to extract a legend from a ggplot object
grab_legend <- function(a_ggplot) {
    tmp <- ggplot_gtable(ggplot_build(a_ggplot))
    leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
    legend <- tmp$grobs[[leg]]
    legend
}

station_legend <- grab_legend(bloom_plots)

bloom_plots
```

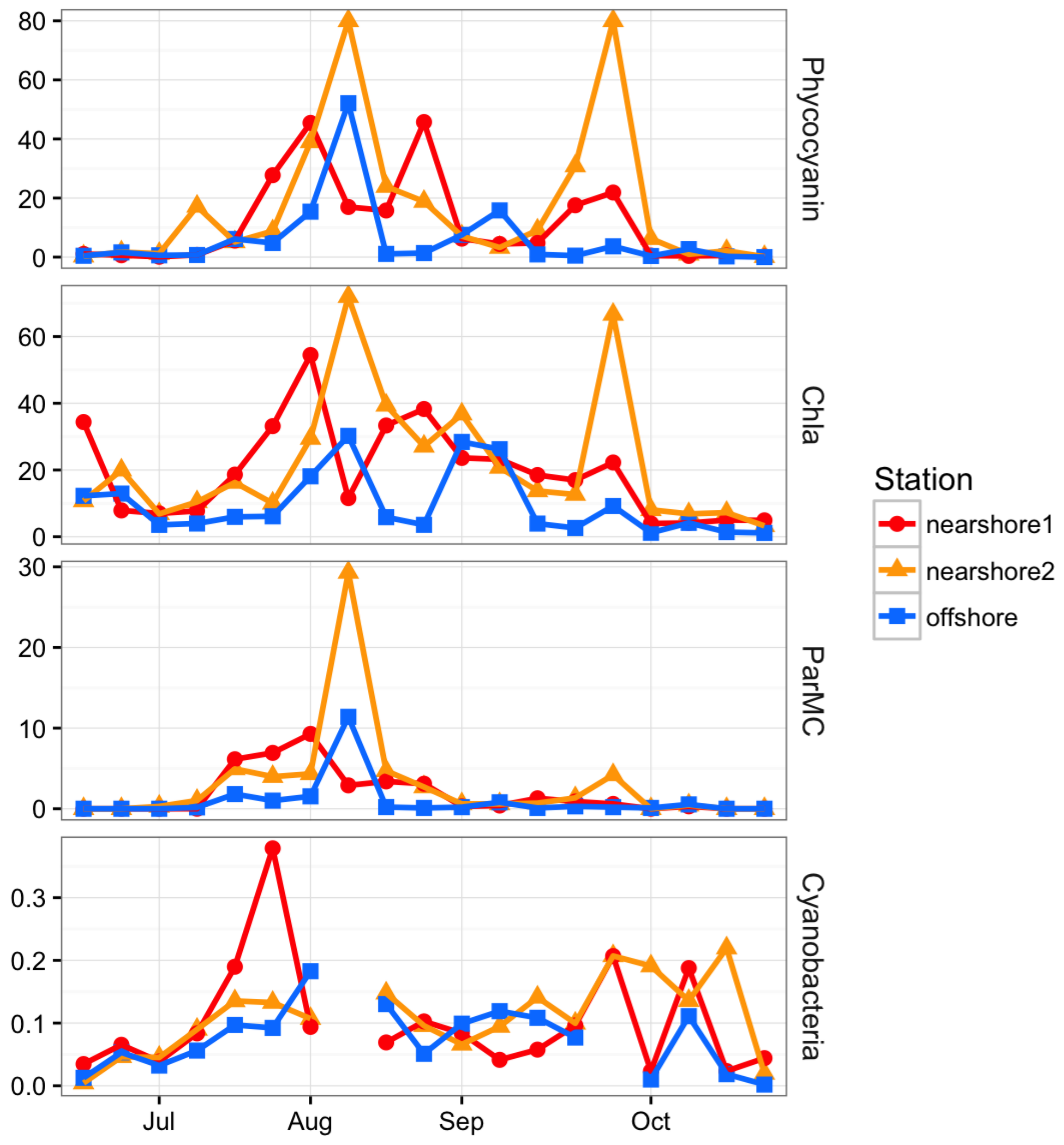


Figure 1 Statistics

Pearson correlations between chl a and phycocyanin

```r
# Calculate correlation between Chla and phycocyanin for all sites
cor.test(
  x = nutrient$Chla,
  y = nutrient$Phycocyanin,
  alternative = "greater",
  method = "pearson"
)
```

```
##
```

```
##  Pearson's product-moment correlation
##
## data:  nutrient$Chla and nutrient$Phycocyanin
## t = 8.88, df = 55, p-value = 1.645e-12
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
##  0.658666 1.000000
## sample estimates:
##       cor
## 0.7675306
```

Figure 2: Cyano OTUs

The goal of this code is to generate lineplots for each non-rare (rel abundance > 0.0001) cyanobacterial OTU

```
# Function to make a ggplot lineplot of an OTU's relative abundance over time
#
# Args:
#   df: a melted data frame generated by calling psmelt() on a phyloseq object.
#       Contains an "Abundance" column for the OTU's abundance
#   otu: the OTU to generate a lineplot for
#   taxrank: the taxonomic rank to appear in the plot title (e.g. "Genus")
# Returns:
#   a ggplot lineplot
plot_otus <- function(df, otu, taxrank) {
  ggplot(df,
    aes(x = Date, y = Abundance, group = Station, color = Station, shape = Station)) +
    geom_point(size = 2) +
    geom_line(size = 0.7) +
    ggtitle(paste(df[1, taxrank], otu)) +
    ylab("rel. abund") +
    scale_color_manual(values = station_colors) +
    scale_x_discrete(
      breaks = c("7/8", "8/4", "9/2", "10/6"),
      labels = c("Jul", "Aug", "Sep", "Oct"),
      drop = FALSE
    ) +
    theme(
      axis.title.x = element_blank(),
      axis.title.y = element_text(size = 9),
      legend.position = "none",
      plot.title = element_text(size = 10, face = "bold")
    )
}

## Select only cyano OTUs that have mean relative abundace > 0.0001
n = 15000
thresh = 0.0001

# Prune low abundance taxa using thresh as mean relative abundance
tax_mean <- taxa_sums(erie_scale)/nsamples(erie_scale)
erie_prune_0001 <- prune_taxa(tax_mean > thresh*n, erie_scale)
```

```r
# Create a melted data frame of selected cyanobacteria OTUs
cyano_otus <-
  erie_prune_0001 %>%
    transform_sample_counts(function(x) {x/sum(x)}) %>%
    subset_taxa(Class == "Cyanobacteria") %>%
    psmelt() %>%
    order_dates()

cyano_otu_names <- as.list(levels(cyano_otus$Species))
names(cyano_otu_names) <- levels(cyano_otus$Species)


# Generate a lineplot for each cyanobacteria with mean relative abundance > 0.0001
cyano_otu_plots <- lapply(cyano_otu_names,
  function(otu) {
    df_otu <- filter(cyano_otus, OTU == otu)
    plot <- plot_otus(df = df_otu, otu = otu, taxrank = "Genus")
    return(plot)
  }
)


##################### Compile plots ####################################

grid.arrange(
  cyano_otu_plots$Otu00007, cyano_otu_plots$Otu00037, cyano_otu_plots$Otu00005,
  cyano_otu_plots$Otu00044, cyano_otu_plots$Otu00063, cyano_otu_plots$Otu00049,
  cyano_otu_plots$Otu00147, cyano_otu_plots$Otu00193, cyano_otu_plots$Otu00304,
  cyano_otu_plots$Otu00177, station_legend,           cyano_otu_plots$Otu00403,
  ncol = 3
)
```

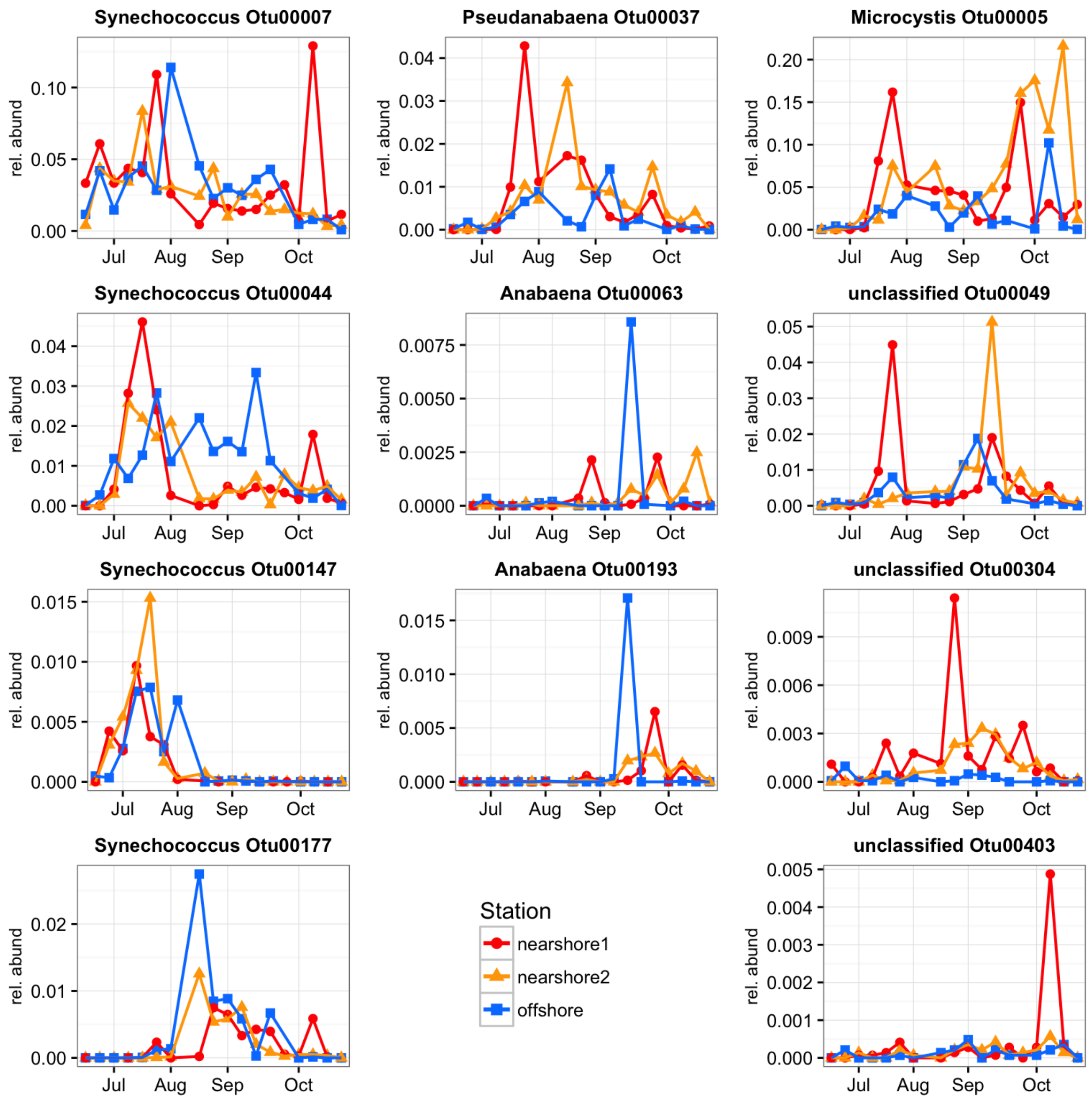


Figure 3: Alpha diversity

```
# My own subsetting function, similar to phyloseq::subset_taxa, except taxa can
# be passed as arguments within functions without weird environment errors
#
# Args:
#   physeq: a phyloseq object
#   taxrank: taxonomic rank to filter on
#   taxa: a vector of taxa groups to filter on
#
# Returns:
```

```r
#   a phyloseq object subsetted to the x taxa in taxrank
my_subset_taxa <- function(physeq, taxrank, taxa) {
  physeq_tax_sub <- tax_table(physeq)[tax_table(physeq)[ , taxrank] %in% taxa, ]
  tax_table(physeq) <- physeq_tax_sub
  return(physeq)
}
```

```r
# Initialize parameters
trials = 100
min_lib = 15000 # Depth we are rarefying to

# Groups to estimate alpha diversity for
mytaxa <- c("Bacteria", "NcBacteria", "Actinobacteria", "Alphaproteobacteria", "Betaproteobacteria",
            "Bacteroidetes", "Gammaproteobacteria", "Deltaproteobacteria", "Verrucomicrobia")
names(mytaxa) <- mytaxa

# Taxonomic ranks of mytaxa
mytaxa_taxrank <- c("Kingdom", "Class", "Phylum", "Class", "Class", "Phylum", "Class", "Class", "Phylum
names(mytaxa_taxrank) <- mytaxa

# Data frame to hold alpha diversity estimates over trials
alphadiv_df <- data.frame(matrix(nrow = nsamples(erie), ncol = trials))

# Initialize empty df's for richness and evenness of all taxa in mytaxa
richness <- lapply(mytaxa, function(x) {return(alphadiv_df)} )
inv_simpson <- lapply(mytaxa, function(x) {return(alphadiv_df)} )

alphadiv_list <- list(richness = richness, inv_simpson = inv_simpson)


# It is always important to set a seed when you subsample so your result is replicable
set.seed(3)

# Run trials to subsample and estimate diversity
for (i in 1:trials) {
  # Subsample
  rarefied_physeq <- rarefy_even_depth(erie, sample.size = min_lib, verbose = FALSE, replace = TRUE)

  # Generate alpha-diversity estimates for each taxonomic group
  for (t in mytaxa) {
    # Subset physeq object to taxa in mytaxa
    if (t != "NcBacteria") {
      physeq_sub <- my_subset_taxa(
        physeq = rarefied_physeq,
        taxrank = mytaxa_taxrank[t],
        taxa = t
      )
    } else {
      physeq_sub <- subset_taxa(physeq = rarefied_physeq, Class != "Cyanobacteria")
    }

    # Calculate observed richness for that group and store value in a df column
    alphadiv_list$richness[[t]][ ,i] <- estimate_richness(physeq_sub, measures = "Observed")[ ,1]
    # Calculate Inv. Simpson for that group and store value in a df column
    alphadiv_list$inv_simpson[[t]][ ,i] <- estimate_richness(physeq_sub, measures = "InvSimpson")[ ,1]

  }
}
```

```r
# Calculate the means of richness and inverse simpson from the 100 trials
alphadiv_est <- lapply(alphadiv_list, function(div_measure) {
    lapply(div_measure, function(taxa_group) {
        alpha_mean <- rowMeans(taxa_group)
        return(alpha_mean)
    })
})

# Convert alphadiv_est richness and inv_simpsons lists into wide data frames
l <- lapply(alphadiv_est, function(x) {
  # convert from list to data.frame
  est_df <- plyr::ldply(.data = x, .fun = data.frame)
  names(est_df) <- c("Taxa", "Diversity")

  # Add in SampleID column and spread to wide format
  r <- est_df %>%
    mutate(SampleID = rep(sample_names(erie), length(mytaxa)))
  return(r)
})

# Merge sample metadata with these estimates
chla_dat <- data.frame(sample_data(erie)) %>%
  select(SampleID, Chla, Station, Date, Days)

# Create a df with a "Diversity" column that includes richness and inv. simpson,
# and log-chl a values from erie sample_data
alpha_comb <- l$richness %>%
  left_join(y = l$inv_simpson, by = c("Taxa", "SampleID")) %>%  # Join the richness and inv_simp df's
  rename(Richness = Diversity.x, InvSimpson = Diversity.y) %>%  # rename columns to avoid confusion
  left_join(chla_dat, by = "SampleID") %>%                      # Join with chla data
  mutate(logChla = log(Chla)) %>%
  gather(key = "Alphadiv", value = "Estimate", Richness, InvSimpson) %>%
  order_dates()
```

```r
# Function to test whether there is a linear relationship
# between log chla and alpha diversity of a group
#
# Args:
#   df: a data frame with a column called logChla and value
#
# Returns:
#   a vector with the pvalue and R2 of the linear model
test_alphadiv_pp <- function(df) {

  df_sub <-
    df %>%
      select(logChla, Estimate) %>%
      na.omit()

  # Fit a linear model
  fit <- lm(Estimate ~ logChla, data = df_sub)

  # Grab model outputs
  fit_pvalue <- summary(fit)$coef[2,4]
  fit_r2 <- summary(fit)$r.squared

  return(c(fit_pvalue, fit_r2))
}

divs <- named_list("Richness", "InvSimpson")

# apply alpha div test to each diversity index for each group
alpha_models <- lapply(divs, function(d) {
  alpha_sub <- alpha_comb %>% filter(Alphadiv == d)
  lapply(mytaxa, function(t) {
    alpha_sub <- alpha_sub %>% filter(Taxa == t)
    # Fit linear model
    fit <- test_alphadiv_pp(alpha_sub)
    return(fit)
  })
})

# Unlist richness
alpha_results <- lapply(alpha_models, function(x) {
  f <- x %>%
     unlist(use.names = FALSE) %>%
     matrix(
        nrow = length(mytaxa),
        ncol = 2,
        byrow = TRUE,
        dimnames = list(mytaxa, c("pvalue","r2"))
    )

  # fdr correction on pvalues
  f[ ,1] <- p.adjust(f[ ,1], method = "fdr")
  # Round to three significant digits
  f <- round(f, digits = 3)
})
```

```
# Function to make a ggplot scatterplot of logChla vs an alpha-diversity metric.
# If the pvalue is below 0.05, it will also plot the fitted line
#
# Args:
#   df: a melted data frame with a column called logChla and value for alpha-diversity
#   measure: Alpha-diversity measure (e.g. "InvSimpson" or "Observed")
#   group: Taxonomic group to plot (e.g. "Betaproteobacteria")
#   pvalue: pvalue from linear model returned by test_alphadiv_pp
#   r2: r2 from linear model returned by test_alphadiv_pp
#
# Returns:
```

```
#   a ggplot
make_alphadiv_plot <- function(df, measure, group, pvalue, r2) {

  g <- ggplot(df, aes(x = logChla, y = Estimate)) +
    geom_point() +
    ylab(measure) +
    ggtitle(group)

  # Since we rounded to 3 sigfigs, pEstimates of 0 need to actually say "p < 0.001"
  if (pvalue != 0) {
    g <- g + annotate(
      "text",
      x = 1,
      y = max(df$Estimate) - 0.03*max(df$Estimate),
      size = 3,
      label = paste("p =", pvalue)
      )
  } else {
    g <- g + annotate(
      "text",
      x = 1,
      y = max(df$Estimate) - 0.03*max(df$Estimate),
      size = 3,
      label = "p < 0.001"
      )
  }

  if (pvalue < 0.05) {
    g <- g + annotate(
      "text",
      x = 1,
      y = max(df$Estimate) - 0.08*max(df$Estimate),
      size = 3,
      label = paste("R2 =", r2)
    ) + geom_smooth(method = "lm", size = 1)
  }

  return(g)

}

## Make plots for inverse simpson index vs log chla
simp_plots <- list()

for (i in 1:length(mytaxa)) {
  df <- filter(alpha_comb, Taxa == mytaxa[i]) %>%
    filter(Alphadiv == "InvSimpson")
  simp_plots[[i]] <- make_alphadiv_plot(
    df = df,
    measure = "InvSimpson",
    group = mytaxa[i],
    pvalue = alpha_results$InvSimpson[i, 1],
    r2 = alpha_results$InvSimpson[i, 2]
  )
}


## Make plots for observed richness vs log chla
obs_plots <- list()

for (i in 1:length(mytaxa)) {
  df <- filter(alpha_comb, Taxa == mytaxa[i]) %>%
    filter(Alphadiv == "Richness")
  obs_plots[[i]] <- make_alphadiv_plot(
    df = df,
    measure = "Richness",
    group = mytaxa[i],
    pvalue = alpha_results$Richness[i, 1],
    r2 = alpha_results$Richness[i, 2]
  )
}

## Arrange plots for final figure
grid.arrange(obs_plots[[2]],  obs_plots[[4]],  obs_plots[[5]],  obs_plots[[6]],
             simp_plots[[2]], simp_plots[[4]], simp_plots[[5]], simp_plots[[6]],
             ncol = 4)
```

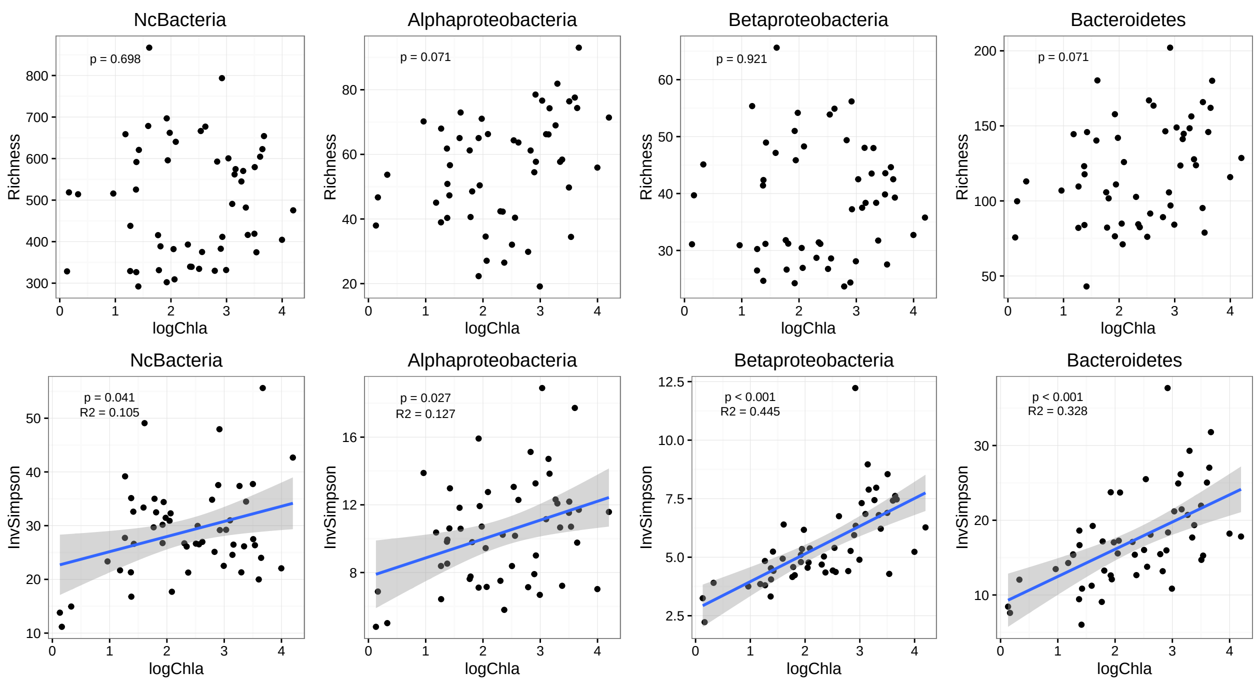


Figure 4: PCoA analyses

PCoA ordination plots

```
# Subset to cyanobacteria and scale internally
cyanos <-
  erie %>%
    subset_taxa(Class == "Cyanobacteria") %>%
    scale_reads(round = "round")


# Subset to cyanobacteria and scale internally
non_cyanos <-
  erie %>%
    subset_taxa(Class != "Cyanobacteria") %>%
    scale_reads(round = "round")

# Make a list of phyloseq objects for the full community, cyanos, and non-cyanos
physeq_subsets <- list(erie_scale, cyanos, non_cyanos)
names(physeq_subsets) <- c("full", "Cyanobacteria", "NcBacteria")


# Generate pcoa scores for each subset
pcoas <- lapply(physeq_subsets,
  function(x) {
    ordinate(
      physeq = x,
      method = "PCoA",
      distance = "bray"
    )
  }
)


# Generate a df to plot pcoa for each subset
pcoa_dfs <- lapply(pcoas,
  function(x, names) {
    p <- plot_ordination(
      physeq = erie_scale,
      axes = 1:3,
      ordination = x,
      justDF = TRUE
    )
    p$Month <- factor(p$Month,
      levels = c("June", "July", "August", "September", "October"))
    p <- p %>%
      rename(PC1 = Axis.1, PC2 = Axis.2, PC3 = Axis.3) %>%
      order_dates()
    return(p)
  }
)

# Flip orientation of PC2 for Cyanobacteria (does not affect interpretation)
```

```
pcoa_dfs$NcBacteria$PC2 <- -pcoa_dfs$NcBacteria$PC2

# Generate relative, lingoes-corrected eigenvalues for PC1 and PC2
eigs <- lapply(pcoas,
  function(x) {
    pcs <- c(PC1 = signif(x$values$Rel_corr_eig[1]*100, 3),
             PC2 = signif(x$values$Rel_corr_eig[2]*100, 3),
             PC3 = signif(x$values$Rel_corr_eig[3]*100, 3)
            )
    return(pcs)
  }
)


pcoa_plots <- Map(
  function(x, n) {
    ggplot(data = x, aes(x = PC1, y = PC2,
           color = Station, shape = Station)) +
      geom_point(aes(alpha = Month), size = 2.5) +
      scale_color_manual(values = station_colors) +
      xlab(paste("PC1 ", eigs[[n]][1], "%")) +
      ylab(paste("PC2 ", eigs[[n]][2], "%")) +
      theme(plot.margin = unit(c(0, 0.2, 0, 0.2), "cm"))
  },
  pcoa_dfs, names(pcoa_dfs)
)


# Extract legend
pcoa_legend <- grab_legend(pcoa_plots$full)

# Remove legend from plots
pcoa_plots <- lapply(pcoa_plots,
  function(x) {x + theme(legend.position = "none")}
)
```

PCoA time series plots

which PC's exceed the broken stick model for cyanos?

```
which(
  pcoas$Cyanobacteria$values$Rel_corr_eig >
  pcoas$Cyanobacteria$values$Broken_stick
)
```

```
##  [1]  1  2 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
```

We will inspect PC1 and PC2 for cyanobacteria

which PC's exceed the broken stick model for nc-bacteria?

```r
# Determine which PC's exceed broken stick model for nc-bacteria
which(
  pcoas$NcBacteria$values$Rel_corr_eig >
  pcoas$NcBacteria$values$Broken_stick
)
```

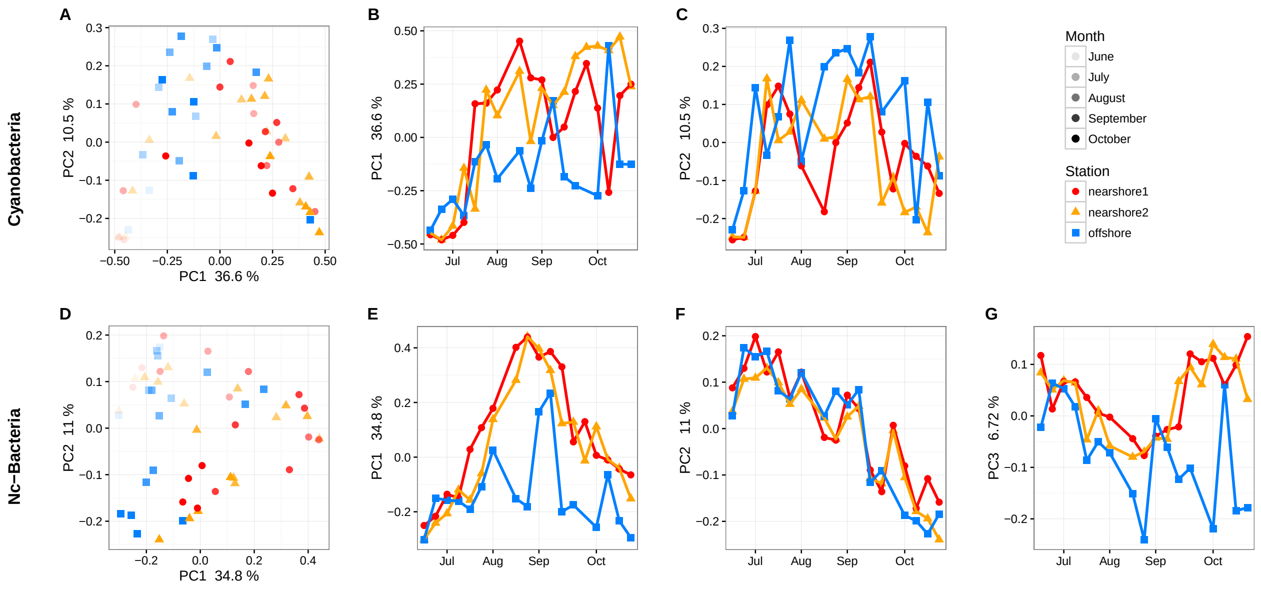
```
## [1]  1  2  3 48 49 50 51
```

We will inspect PC1, PC2, and PC3 for nc-bacteria

```r
# Function to create a plot of time (x-axis) vs PC scores (y-axis)
plot_pcts <- function(df, pc, eigs) {
  ggplot(df,
    aes_string(x = "Date", y = pc, group = "Station", color = "Station", shape = "Station")) +
      geom_point(size = 2.5) +
      geom_line(size = 1.1) +
      scale_color_manual(values = station_colors) +
      scale_x_discrete(
        breaks = c("7/8", "8/4", "9/2", "10/6"),
        labels = c("Jul", "Aug", "Sep", "Oct"),
        drop = FALSE
      ) +
      ylab(paste(pc, " ", eigs[pc], "%")) +
      theme(
        axis.title.x = element_blank(),
        plot.title = element_text(face = "bold", size = 16),
        legend.position = "none",
        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "cm")
      )
}
# Cyano PC time series plots
cyano_pcs <- named_list("PC1", "PC2")

cyano_pc_plots <- lapply(cyano_pcs,
  function(x) {
    plot_pcts(pcoa_dfs$Cyanobacteria, x, eigs = eigs$Cyanobacteria)
  }
)

# Non-cyano PC time series plots
non_cyano_pcs <- named_list("PC1", "PC2", "PC3")

non_cyano_pc_plots <- lapply(non_cyano_pcs,
  function(x) {
    plot_pcts(pcoa_dfs$NcBacteria, x, eigs = eigs$NcBacteria)
  }
)
```

Compiled plot

```
ggdraw() +
  ## Cyano
  draw_plot(pcoa_plots$Cyanobacteria, x = 0.05, y = 0.5, width = 0.22, height = 0.43) +
  draw_plot(cyano_pc_plots$PC1, x = 0.29, y = 0.52, width = 0.22, height = 0.42) +
  draw_plot(cyano_pc_plots$PC2, x = 0.53, y = 0.52, width = 0.22, height = 0.42) +
  ## non-cyano
  draw_plot(pcoa_plots$NcBacteria, x = 0.05, y = 0, width = 0.22, height = 0.43) +
  draw_plot(non_cyano_pc_plots$PC1, x = 0.29, y = 0.02, width = 0.22, height = 0.42) +
  draw_plot(non_cyano_pc_plots$PC2, x = 0.53, y = 0.02, width = 0.22, height = 0.42) +
  draw_plot(non_cyano_pc_plots$PC3, x = 0.77, y = 0.02, width = 0.22, height = 0.42) +
  ## legend and labels
  draw_plot(pcoa_legend, x = 0.82, y = 0.57, width = 0.1, height = .35) +
  draw_plot_label(c("A", "B", "C", "D", "E", "F", "G"),
                  c(0.05, 0.29, 0.53, 0.05, 0.29, 0.53, 0.77),
                  c(0.97, 0.97, 0.97, 0.47, 0.47, 0.47, 0.47),
                  size = 14) +
  draw_plot_label(c("Cyanobacteria", "Nc-Bacteria"),
                  c(0.01, 0.01), c(.5, 0.05), size = 14, angle = 90)
```



PERMANOVA

Cyanobacteria

```
# Remove dates for which we are missing samples for any of the sites
cyano_permanova_subset <- subset_samples(cyanos, Date != "9/29")

# Calculate bray-curtis distance
cyano_bdist <- phyloseq::distance(physeq = cyano_permanova_subset, method = "bray")

# Convert sample_data to df
sampledf <- data.frame(sample_data(cyano_permanova_subset))
```

time + site adonis

```r
# Adonis test
adonis(cyano_bdist ~ Date + Station, data = sampledf)
```

```
##
## Call:
## adonis(formula = cyano_bdist ~ Date + Station, data = sampledf)
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##           Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
## Date      16    3.9636 0.247728  3.9065 0.59944  0.001 ***
## Station    2    0.6194 0.309683  4.8835 0.09367  0.003 **
## Residuals 32    2.0293 0.063414         0.30689
## Total     50    6.6123                  1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Homogeneity of dispersion test
beta <- betadisper(cyano_bdist, sampledf$Date)
permutest(beta)
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df  Sum Sq  Mean Sq      F N.Perm Pr(>F)
## Groups    16 0.37818 0.023637 0.8318    999  0.647
## Residuals 34 0.96616 0.028416
```

shore + site adonis

```r
# Adonis test
adonis(cyano_bdist ~ Shore + Station, data = sampledf)
```

```
##
## Call:
## adonis(formula = cyano_bdist ~ Shore + Station, data = sampledf)
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##           Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## Shore      1    0.5639 0.56391  4.5167 0.08528  0.020 *
## Station    1    0.0555 0.05545  0.4441 0.00839  0.692
## Residuals 48    5.9929 0.12485         0.90633
## Total     50    6.6123                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Homogeneity of dispersion test
beta <- betadisper(cyano_bdist, sampledf$Shore)
permutest(beta)
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df  Sum Sq  Mean Sq      F N.Perm Pr(>F)
## Groups     1 0.04806 0.048055 1.7795    999  0.208
## Residuals 49 1.32325 0.027005
```

Nc-Bacteria

```r
# Remove dates for which we are missing samples for any of the sites
non_cyano_permanova_subset <- subset_samples(non_cyanos, Date != "9/29")

# Calculate bray-curtis distance
non_cyano_bdist <- phyloseq::distance(physeq = non_cyano_permanova_subset, method = "bray")

# Convert sample_data to df
sampledf <- data.frame(sample_data(non_cyano_permanova_subset))
```

time + site adonis

```r
adonis(non_cyano_bdist ~ Date + Station, data = sampledf)
```

```
##
## Call:
## adonis(formula = non_cyano_bdist ~ Date + Station, data = sampledf)
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##           Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## Date      16    3.5749 0.22343  3.8102 0.58443  0.001 ***
## Station    2    0.6655 0.33275  5.6745 0.10880  0.001 ***
## Residuals 32    1.8765 0.05864         0.30677
## Total     50    6.1169                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
beta <- betadisper(non_cyano_bdist, sampledf$Date)
permutest(beta)
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df  Sum Sq  Mean Sq      F N.Perm Pr(>F)
## Groups    16 0.16704 0.010440 0.5618    999  0.909
## Residuals 34 0.63179 0.018582
```

shore + site adonis

```
adonis(non_cyano_bdist ~ Shore + Station, data = sampledf)
```

```
##
## Call:
## adonis(formula = non_cyano_bdist ~ Shore + Station, data = sampledf)
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##           Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## Shore      1    0.5849 0.58490  5.1501 0.09562  0.001 ***
## Station    1    0.0806 0.08060  0.7097 0.01318  0.640
## Residuals 48    5.4514 0.11357         0.89120
## Total     50    6.1169                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
beta <- betadisper(non_cyano_bdist, sampledf$Shore)
permutest(beta)
```

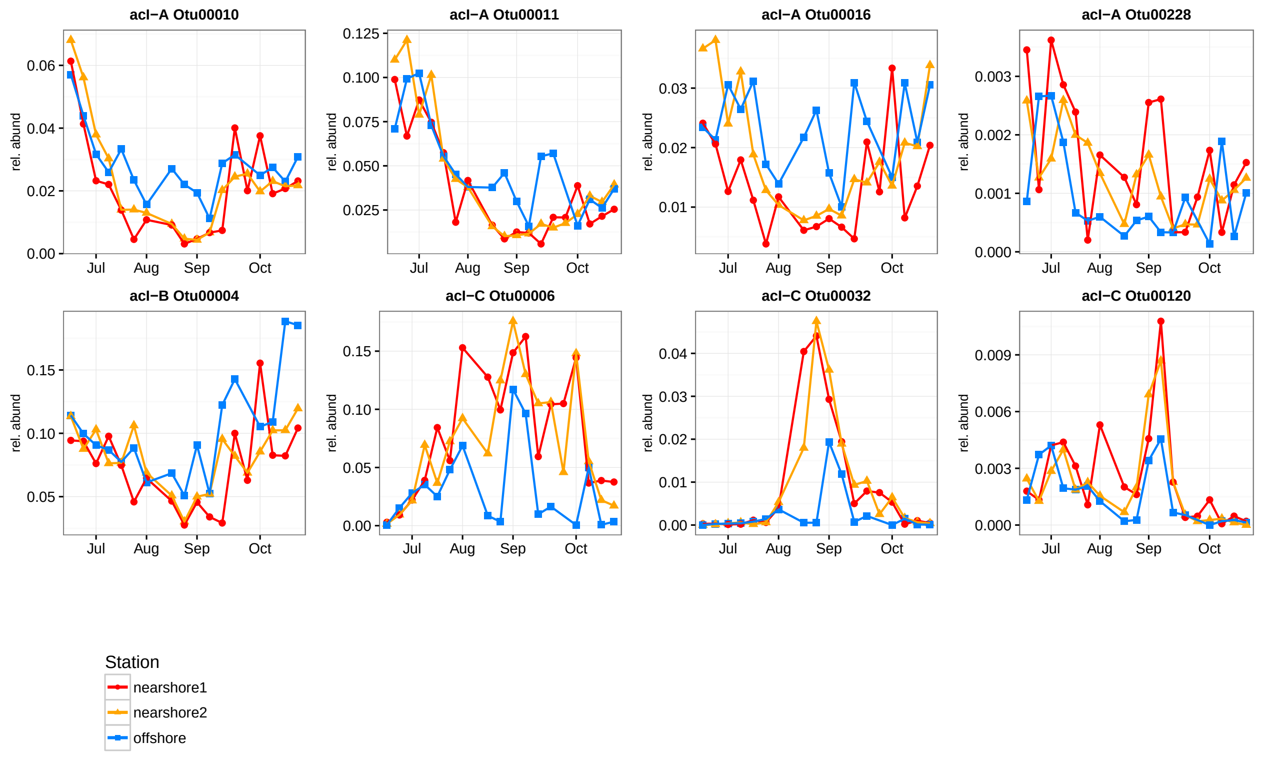
```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df   Sum Sq  Mean Sq      F N.Perm Pr(>F)
## Groups     1 0.032703 0.032703 6.1042    999  0.016 *
## Residuals 49 0.262518 0.005358
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 5: Actinobacteria OTU dynamics

```r
# AcI is the most abundant lineage
erie_scale %>%
  tax_glom(taxrank = "Family") %>%
  psmelt() %>%
  group_by(Family) %>%
  summarise(mean = mean(Abundance)) %>%
  arrange(desc(mean))
```

```
## Source: local data frame [274 x 2]
##
##               Family      mean
##               (fctr)     (dbl)
## 1                acI 3627.1509
## 2               bacI 1295.5849
## 3               betI  778.9057
## 4  Planctomycetaceae  751.5094
## 5               bacV  569.0189
## 6              betIV  479.6981
## 7            FamilyI  468.7610
## 8               acIV  303.2830
## 9              betII  292.2453
## 10             bacVI  284.8302
## ..               ...       ...
```

```r
# Subset to just acI
aci <-
  erie_scale %>%
    transform_sample_counts(function(x) {x/sum(x)} ) %>%
    subset_taxa(Family == "acI") %>%
    psmelt() %>%
    order_dates()

aci_otus <- levels(aci$Species)

aci_plots <- lapply(aci_otus,
  function(x) {
    df_otu <- filter(aci, OTU == x)
    aci_plot <- plot_otus(df = df_otu, otu = x, taxrank = "Genus")
    return(aci_plot)
  }
)


grid.arrange(
  aci_plots[[3]], aci_plots[[4]], aci_plots[[5]], aci_plots[[8]],
  aci_plots[[1]], aci_plots[[2]], aci_plots[[6]], aci_plots[[7]],
  station_legend,
  ncol = 4, nrow = 3
)
```



Linear models

```r
library(leaps)

# Function to extract the best subset multiple linear regression model
#
# Args:
#   vars: vectors of all variables to consider in the model
#   response: response variable of the model
#   dat: dataframe with vars and response
#
# Returns: a list with the variables in the best model, bic, cp, and adjusted r2
get_bestsub_summary <- function(vars, response, dat) {
  formula = reformulate(termlabels = vars, response = response)
  lm_model <- regsubsets(formula, dat)
  bic <- summary(lm_model)$bic
  cp <- summary(lm_model)$cp
  adjr2 <- summary(lm_model)$adjr2
  best_model <- summary(lm_model)$which[which.min(bic), ]
  return(list(model = best_model, bic = bic, cp = cp, adjr2 = adjr2))
}

# Variables to include in cyano models
cyano_vars <- c("Nitrate", "SRP", "Temp", "H2O2", "SpCond", "Ammonia", "Turbidity", "Days")

# Variables to include in nc-bacteria models
non_cyano_vars <- c(cyano_vars, "pH",  "ParMC", "Chla")
```

```r
# Impute SpCond values for nearshore 1 on Sep 2 and Sep 8 with value for nearshore 2
pcoa_dfs_impute <- lapply(pcoa_dfs,
  function(x) {
    # Change 9/2 value
    x$SpCond[x$Date == "9/2" & x$Station == "nearshore1"] <-
      x$SpCond[x$Date == "9/2" & x$Station == "nearshore2"]
    # Change 9/8 value
     x$SpCond[x$Date == "9/8" & x$Station == "nearshore1"] <-
      x$SpCond[x$Date == "9/8" & x$Station == "nearshore2"]
    return(x)
  }
)
```

Cyano models

```r
# Get the variables best subset model for the cyano community and then
# fit the model to extract coefficients and p-values.
cyano_models <- lapply(cyano_pcs,
  function(x) {
    best_model <- get_bestsub_summary(cyano_vars, x, dat = pcoa_dfs_impute$Cyanobacteria)
    model <- lm(
      formula = reformulate(cyano_vars[best_model$model[-1]], x),
      data = pcoa_dfs_impute$Cyanobacteria
    )
    return(model)
  }
)
```

PC1

```r
summary(cyano_models$PC1)
```

```
##
## Call:
## lm(formula = reformulate(cyano_vars[best_model$model[-1]], x),
##     data = pcoa_dfs_impute$Cyanobacteria)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48009 -0.12766 -0.02878  0.14444  0.24356
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.4272225  0.3376777  -4.227 0.000106 ***
## H2O2         0.0003124  0.0001181   2.646 0.010982 *
## SpCond       0.0026639  0.0011588   2.299 0.025915 *
## Turbidity    0.0143916  0.0032658   4.407 5.87e-05 ***
## Days         0.0051998  0.0006812   7.633 7.93e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1769 on 48 degrees of freedom
## Multiple R-squared:  0.6755, Adjusted R-squared:  0.6485
## F-statistic: 24.99 on 4 and 48 DF,  p-value: 3.188e-11
```

PC2

```
summary(cyano_models$PC2)
```

```
##
## Call:
## lm(formula = reformulate(cyano_vars[best_model$model[-1]], x),
##     data = pcoa_dfs_impute$Cyanobacteria)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18917 -0.05410 -0.01098  0.06232  0.32464
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.105e-01  3.363e-01  -0.626 0.534289
## Nitrate      3.749e-04  8.879e-05   4.223 0.000107 ***
## Temp         4.077e-02  8.167e-03   4.993 8.27e-06 ***
## SpCond      -4.174e-03  7.787e-04  -5.360 2.34e-06 ***
## Days         4.661e-03  9.585e-04   4.863 1.28e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.108 on 48 degrees of freedom
## Multiple R-squared:  0.5506, Adjusted R-squared:  0.5131
## F-statistic:  14.7 on 4 and 48 DF,  p-value: 6.55e-08
```

Non-cyano models

```
# get the variables best subset model for the non-cyano community and then
# fit the model to extract coefficients and p-values
non_cyano_models <- lapply(non_cyano_pcs,
  function(x) {
    best_model <- get_bestsub_summary(non_cyano_vars, x, dat = pcoa_dfs_impute$NcBacteria)
    model <- lm(
      formula = reformulate(non_cyano_vars[best_model$model[-1]], x),
      data = pcoa_dfs_impute$NcBacteria
    )
    return(model)
  }
)
```

PC1

```r
summary(non_cyano_models$PC1)
```

```
##
## Call:
## lm(formula = reformulate(non_cyano_vars[best_model$model[-1]],
##     x), data = pcoa_dfs_impute$NcBacteria)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.230086 -0.048647 -0.002318  0.065187  0.248261
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.5495599  0.3924083 -11.594 3.01e-15 ***
## SRP          0.0144208  0.0034016   4.239 0.000107 ***
## Days         0.0019157  0.0005042   3.799 0.000424 ***
## pH           0.5087169  0.0450809  11.285 7.60e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1084 on 46 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.7687, Adjusted R-squared:  0.7536
## F-statistic: 50.96 on 3 and 46 DF,  p-value: 1.153e-14
```

PC2

```r
summary(non_cyano_models$PC2)
```

```
##
## Call:
## lm(formula = reformulate(non_cyano_vars[best_model$model[-1]],
##     x), data = pcoa_dfs_impute$NcBacteria)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.102535 -0.020498 -0.005607  0.023581  0.103218
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.6578822  0.0725671  -9.066 5.67e-12 ***
## SRP         -0.0045399  0.0013103  -3.465  0.00113 **
## Temp         0.0226581  0.0018100  12.518  < 2e-16 ***
## SpCond       0.0008462  0.0002631   3.217  0.00232 **
## Turbidity   -0.0016045  0.0007930  -2.023  0.04863 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.04235 on 48 degrees of freedom
## Multiple R-squared:  0.888,  Adjusted R-squared:  0.8786
## F-statistic: 95.11 on 4 and 48 DF,  p-value: < 2.2e-16
```

PC3

```
summary(non_cyano_models$PC3)
```

```
##
## Call:
## lm(formula = reformulate(non_cyano_vars[best_model$model[-1]],
##     x), data = pcoa_dfs_impute$NcBacteria)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12411 -0.03042  0.00210  0.03355  0.10428
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.8494228  0.0960755  -8.841 1.02e-11 ***
## SRP          0.0048398  0.0013824   3.501 0.000999 ***
## SpCond       0.0030774  0.0003579   8.598 2.36e-11 ***
## ParMC       -0.0148944  0.0041118  -3.622 0.000692 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05596 on 49 degrees of freedom
## Multiple R-squared:  0.6679, Adjusted R-squared:  0.6476
## F-statistic: 32.85 on 3 and 49 DF,  p-value: 8.709e-12
```

Deseq tests

```
# Function to run deseq2 differential abundance analysis.
#
# Args:
#   physeq: a phyloseq object
#   var: factor column in sample_data
# Returns:
#   a df with log2fold ratios and pvalues of differentially abundant taxa
get_PC_deseq_OTUs <- function(physeq, var) {

  station_deseq <- phyloseq_to_deseq2(physeq, reformulate(var))
  station_deseq <- DESeq(station_deseq, test = "Wald", fitType = "parametric")

  my_alpha = 0.05
  res <- data.frame(results(station_deseq, cooksCutoff = FALSE, alpha = my_alpha))
  res <- data.frame(OTU = row.names(res), res)

  sigtab <-
```

```
    res %>%
      filter(padj < my_alpha)

  sigtab_station <- cbind(sigtab, as(tax_table(physeq)[sigtab$OTU, ], "matrix"))
  return(sigtab_station)
}
```

```
# Create factor levels for samples on PCs
noncyano_df <-
  pcoa_dfs$NcBacteria %>%
    mutate(PC1group = ifelse(PC1 > 0, "2", "1")) %>%
    mutate(PC2group = ifelse(PC2 > 0, "2", "1")) %>%
    mutate(PC3group = ifelse(PC3 > 0, "2", "1"))

# Deseq tests should be run on raw count data.
# Create a new physeq object with raw counts using the OTUs from erie_prune_0001
otu_names <- tax_table(erie_prune_0001)[ ,"Species"]

non_cyano_deseq <-
  erie %>%
    subset_taxa(Species %in% otu_names) %>%
    subset_taxa(Phylum != "Cyanobacteria")

# Add PC groups into phyloseq sample_data
sample_data(non_cyano_deseq)$PC1group <- noncyano_df$PC1group
sample_data(non_cyano_deseq)$PC2group <- noncyano_df$PC2group
sample_data(non_cyano_deseq)$PC3group <- noncyano_df$PC3group

## Make physeq objects for each station
nearshore1 <-
    non_cyano_deseq %>%
      subset_samples(Station == "nearshore1")

nearshore2 <-
    non_cyano_deseq %>%
      subset_samples(Station == "nearshore2")

offshore <-
    non_cyano_deseq %>%
      subset_samples(Station == "offshore")


stations_physeq <- named_list(nearshore1, nearshore2, offshore)

# Get the deseq OTUs for each station and each pc
stations_deseq <- lapply(stations_physeq,
  function(x) {
    pc1_deseq <- get_PC_deseq_OTUs(x, var = "PC1group")
    pc2_deseq <- get_PC_deseq_OTUs(x, var = "PC2group")
    pc3_deseq <- get_PC_deseq_OTUs(x, var = "PC3group")
    return(list(pc1 = pc1_deseq, pc2 = pc2_deseq, pc3 = pc3_deseq))
  }
)
```

```r
## Find OTUs that are overabundant on the positive scores
positives <- lapply(stations_deseq,
  function(x) {
    lapply(x, function(x) {
      filter(x, log2FoldChange > 0)
    })
  }
)

## Find OTUs that are overabundant on negative scores
negatives <- lapply(stations_deseq,
  function(x) {
    lapply(x, function(x) {
      filter(x, log2FoldChange < 0)
    })
  }
)

tax_table <- data.frame(tax_table(erie))
```

```r
make_pc_table <- function(score_sign, pc) {
  pc_score <- intersect(score_sign$nearshore1[[pc]][ ,"OTU"],
    intersect(score_sign$nearshore2[[pc]][ ,"OTU"],
              score_sign$offshore[[pc]][ ,"OTU"]
    )
  )
}
```

PC1

Positive

```r
# Taxa that are overabundant on positive PC1 axis
pc1_pos <- make_pc_table(positives, "pc1")

pander(tax_table %>% filter(Species %in% pc1_pos))
```

Table 1: Table continues below

Kingdom	Phylum	Class	Order
Bacteria	Actinobacteria	Actinobacteria	Actinomycetales
Bacteria	Actinobacteria	Actinobacteria	Actinomycetales
Bacteria	Chlorobi	Chlorobia	Chlorobiales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Proteobacteria	Deltaproteobacteria	Myxococcales
Bacteria	Bacteroidetes	Cytophagia	Cytophagales
Bacteria	Proteobacteria	Betaproteobacteria	Burkholderiales
Bacteria	Proteobacteria	Alphaproteobacteria	Rickettsiales
Bacteria	Planctomycetes	Planctomycetacia	Planctomycetales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales

Kingdom	Phylum	Class	Order
Bacteria	Bacteroidetes	Cytophagia	Cytophagales
Bacteria	Proteobacteria	Deltaproteobacteria	Myxococcales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Bacteroidetes	Sphingobacteriia	Sphingobacteriales
Bacteria	Verrucomicrobia	Opitutae	Opitutales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Planctomycetes	Phycisphaerae	Phycisphaerales

Family	Genus	Rank7	Rank8	Species
acI	acI-C	acI-C2	unclassified	Otu00006
acI	acI-C	acI-C2	unclassified	Otu00032
OPB56	unclassified	NA	NA	Otu00046
bacI	unclassified	unclassified	unclassified	Otu00093
0319-6G20	unclassified	NA	NA	Otu00148
Cytophagaceae	unclassified	unclassified	unclassified	Otu00157
betI	unclassified	unclassified	unclassified	Otu00160
unclassified	unclassified	NA	NA	Otu00183
Planctomycetaceae	Blastopirellula	NA	NA	Otu00213
bacI	unclassified	unclassified	unclassified	Otu00225
bacI	unclassified	unclassified	unclassified	Otu00292
Cytophagaceae	unclassified	NA	NA	Otu00308
0319-6G20	unclassified	NA	NA	Otu00416
bacVI	unclassified	unclassified	unclassified	Otu00534
Saprospiraceae	unclassified	NA	NA	Otu00568
Opitutaceae	unclassified	unclassified	unclassified	Otu00775
bacI	unclassified	unclassified	unclassified	Otu00983
Phycisphaeraceae	SM1A02	NA	NA	Otu02322

Negative

```r
# Taxa that are overabundant on negative PC1 axis
pc1_neg <- make_pc_table(negatives, "pc1")

pander(tax_table %>% filter(Species %in% pc1_pos))
```

Table 3: Table continues below

Kingdom	Phylum	Class	Order
Bacteria	Actinobacteria	Actinobacteria	Actinomycetales
Bacteria	Actinobacteria	Actinobacteria	Actinomycetales
Bacteria	Chlorobi	Chlorobia	Chlorobiales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Proteobacteria	Deltaproteobacteria	Myxococcales
Bacteria	Bacteroidetes	Cytophagia	Cytophagales
Bacteria	Proteobacteria	Betaproteobacteria	Burkholderiales
Bacteria	Proteobacteria	Alphaproteobacteria	Rickettsiales
Bacteria	Planctomycetes	Planctomycetacia	Planctomycetales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales

Kingdom	Phylum	Class	Order
Bacteria	Bacteroidetes	Cytophagia	Cytophagales
Bacteria	Proteobacteria	Deltaproteobacteria	Myxococcales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Bacteroidetes	Sphingobacteriia	Sphingobacteriales
Bacteria	Verrucomicrobia	Opitutae	Opitutales
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales
Bacteria	Planctomycetes	Phycisphaerae	Phycisphaerales

Family	Genus	Rank7	Rank8	Species
acI	acI-C	acI-C2	unclassified	Otu00006
acI	acI-C	acI-C2	unclassified	Otu00032
OPB56	unclassified	NA	NA	Otu00046
bacI	unclassified	unclassified	unclassified	Otu00093
0319-6G20	unclassified	NA	NA	Otu00148
Cytophagaceae	unclassified	unclassified	unclassified	Otu00157
betI	unclassified	unclassified	unclassified	Otu00160
unclassified	unclassified	NA	NA	Otu00183
Planctomycetaceae	Blastopirellula	NA	NA	Otu00213
bacI	unclassified	unclassified	unclassified	Otu00225
bacI	unclassified	unclassified	unclassified	Otu00292
Cytophagaceae	unclassified	NA	NA	Otu00308
0319-6G20	unclassified	NA	NA	Otu00416
bacVI	unclassified	unclassified	unclassified	Otu00534
Saprospiraceae	unclassified	NA	NA	Otu00568
Opitutaceae	unclassified	unclassified	unclassified	Otu00775
bacI	unclassified	unclassified	unclassified	Otu00983
Phycisphaeraceae	SM1A02	NA	NA	Otu02322

PC2

Positive

```r
# Positive
pc2_pos <- make_pc_table(positives, "pc2")

pander(tax_table %>% filter(Species %in% pc2_pos))
```

Table 5: Table continues below

Kingdom	Phylum	Class	Order	Family
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales	bacI
Bacteria	Verrucomicrobia	Opitutae	Opitutales	Opitutaceae
Bacteria	Bacteroidetes	Sphingobacteria	Sphingobacteriales	bacI

Genus	Rank7	Rank8	Species
unclassified	unclassified	unclassified	Otu00103

Genus	Rank7	Rank8	Species
unclassified	unclassified	unclassified	Otu00109
unclassified	unclassified	unclassified	Otu00822

Negative

```r
# Negative
pc2_neg <- make_pc_table(negatives, "pc2")

pander(tax_table %>% filter(Species %in% pc2_neg))
```

Table 7: Table continues below

Kingdom	Phylum	Class	Order
Bacteria	Verrucomicrobia	OPB35_soil_group	unclassified
Bacteria	Proteobacteria	Gammaproteobacteria	unclassified
Bacteria	Proteobacteria	Betaproteobacteria	unclassified
Bacteria	Verrucomicrobia	Verrucomicrobiae	Verrucomicrobiales
Bacteria	Verrucomicrobia	S-BQ2-57_soil_group	unclassified

Family	Genus	Rank7	Rank8	Species
unclassified	unclassified	NA	NA	Otu00090
unclassified	unclassified	NA	NA	Otu00108
unclassified	unclassified	unclassified	unclassified	Otu00210
Verrucomicrobiaceae	unclassified	NA	NA	Otu00238
unclassified	unclassified	NA	NA	Otu00662

PC3

Positive

```r
pc3_pos <- make_pc_table(positives, "pc3")

pander(tax_table %>% filter(Species %in% pc3_pos))
```

Table 9: Table continues below

Kingdom	Phylum	Class	Order	Family
Bacteria	Bacteroidetes	Flavobacteria	Flavobacteriales	bacII
Bacteria	Planctomycetes	OM190	unclassified	unclassified

Genus	Rank7	Rank8	Species
bacII-A	Flavo-A3	unclassified	Otu00062
unclassified	NA	NA	Otu00114

Negative

```r
# Negative
pc3_neg <- make_pc_table(negatives, "pc3")

pander(tax_table %>% filter(Species %in% pc3_neg))
```

Table 11: Table continues below

Kingdom	Phylum	Class	Order	Family
Bacteria	Proteobacteria	Deltaproteobacteria	Myxococcales	0319-6G20
Bacteria	Bacteroidetes	Flavobacteria	Flavobacteriales	bacII

Genus	Rank7	Rank8	Species
unclassified	NA	NA	Otu00148
bacII-A	unclassified	unclassified	Otu00720

```r
# make a list of all sig OTUs
sig_otus <- named_list(pc1_pos, pc1_neg, pc2_pos, pc2_neg, pc3_pos, pc3_neg)
```

Microcystis associates

```r
library(psych)

# Run correlation test with full community

# Subset to just the Microcystis OTU
mc <-
  erie_scale %>%
  subset_taxa(Species == "Otu00005")

# Create a list of physeq objects for Microcystis at each station
mc_list <- list()
mc_list$nearshore1 <- subset_samples(mc, Station == "nearshore1")
mc_list$nearshore2 <- subset_samples(mc, Station == "nearshore2")
mc_list$offshore <- subset_samples(mc, Station == "offshore")

# Create a list of physeq objects for nc-bacteria at each station
erie_scale_0001_nc <- subset_taxa(erie_prune_0001, Class != "Cyanobacteria")
nc_bacteria_stations <- list()
nc_bacteria_stations$nearshore1 <- subset_samples(erie_scale_0001_nc, Station == "nearshore1")
nc_bacteria_stations$nearshore2 <- subset_samples(erie_scale_0001_nc, Station == "nearshore2")
nc_bacteria_stations$offshore <- subset_samples(erie_scale_0001_nc, Station == "offshore")

mc_corrs <- list()
Stations = c("nearshore1", "nearshore2", "offshore")
```

```r
# Loop through stations, performing a spearman test between Microcystis and all non-cyanos
for (st in Stations) {
  mc_corrs[[st]] <-
    corr.test(
      x = t(otu_table(mc_list[[st]])),
      y = t(otu_table(nc_bacteria_stations[[st]])),
      method = "pearson",
      adjust = "fdr"
    )
}

sig_corrs <- lapply(mc_corrs,
  function(x) {
    which_sigs <- which(x$p < 0.5)
    return(colnames(x$p)[which_sigs])
  }
)

# Intersection of significant OTUs across all three sites
sig_corrs <- intersect(sig_corrs$nearshore1,
             intersect(sig_corrs$nearshore2, sig_corrs$offshore)
            )

pander(tax_table %>% filter(Species %in% sig_corrs))
```

Table 13: Table continues below

Kingdom	Phylum	Class	Order
Bacteria	Proteobacteria	Alphaproteobacteria	Rhodobacterales
Bacteria	Proteobacteria	Alphaproteobacteria	Rhodospirillales
Bacteria	Proteobacteria	Alphaproteobacteria	Rickettsiales

Family	Genus	Rank7	Rank8	Species
Rhodobacteraceae	Rhodobacter	NA	NA	Otu00030
I-10	unclassified	NA	NA	Otu00180
Rickettsiales_Incertae_Sedis	Candidatus_Captivus	NA	NA	Otu00278

Cyanobacteria correlations

```r
cyano_otu_prune <-
  erie_scale_0001 %>%
    transform_sample_counts(function(x) {x/sum(x)}) %>%
    subset_taxa(Class == "Cyanobacteria")

# Run pairwise pearson correlation tests between all non-rare cyano OTUs
# with an fdr correction for multiple hypotheses.
cyano_corrs_pearson <- corr.test(
  t(otu_table(cyano_otu_prune)),
  use = "pairwise",
  method = "pearson",
  adjust = "fdr"
)

#emphasize.strong.cells(which(cyano_corrs_pearson$p < 0.05, arr.ind = TRUE))
pander(signif(cyano_corrs_pearson$r, digits = 2))
```

Table 15: Table continues below

	Otu00005	Otu00007	Otu00037	Otu00044	Otu00049	Otu00063
Otu00005	1	-0.0099	0.47	0.033	0.27	0.19
Otu00007	-0.0099	1	0.25	0.43	0.22	-0.042
Otu00037	0.47	0.25	1	0.14	0.49	-0.009
Otu00044	0.033	0.43	0.14	1	0.23	0.22
Otu00049	0.27	0.22	0.49	0.23	1	0.046
Otu00063	0.19	-0.042	-0.009	0.22	0.046	1
Otu00147	-0.2	0.48	-0.052	0.41	-0.11	-0.15
Otu00177	-0.043	0.12	0.24	0.11	0.11	-0.081
Otu00193	0.14	0.028	-0.05	0.26	0.093	0.94
Otu00304	0.14	-0.075	0.27	-0.14	0.17	0.19
Otu00403	0.031	0.49	-0.055	0.13	0.1	-0.021

	Otu00147	Otu00177	Otu00193	Otu00304	Otu00403
Otu00005	-0.2	-0.043	0.14	0.14	0.031
Otu00007	0.48	0.12	0.028	-0.075	0.49
Otu00037	-0.052	0.24	-0.05	0.27	-0.055
Otu00044	0.41	0.11	0.26	-0.14	0.13
Otu00049	-0.11	0.11	0.093	0.17	0.1
Otu00063	-0.15	-0.081	0.94	0.19	-0.021
Otu00147	1	-0.24	-0.15	-0.2	-0.14
Otu00177	-0.24	1	-0.093	0.18	0.13
Otu00193	-0.15	-0.093	1	0.065	0.067
Otu00304	-0.2	0.18	0.065	1	0.016
Otu00403	-0.14	0.13	0.067	0.016	1

```r
# Save objects for the supplement
save(
  list = c("simp_plots", "obs_plots",    # alpha diversity plots
    "cyano_models", "non_cyano_models", # linear model results
    "sig_otus",                         # Deseq2 results
    "plot_otus",                        # OTU plotting function
    "alpha_comb"),                      # alpha diversity df
  file = "supplement.RData"
)
```

```r
sessionInfo()
```

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.9.5 (Mavericks)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] psych_1.5.8              leaps_2.9
##  [3] tidyr_0.4.1              pander_0.6.0
##  [5] gtable_0.2.0             cowplot_0.6.1.9999
##  [7] vegan_2.3-1              lattice_0.20-33
##  [9] permute_0.8-4            gridExtra_2.0.0
## [11] reshape2_1.4.1           scales_0.4.0
## [13] dplyr_0.4.3              ggplot2_2.1.0
## [15] DESeq2_1.8.2             RcppArmadillo_0.6.100.0.0
## [17] Rcpp_0.12.3              GenomicRanges_1.20.8
## [19] GenomeInfoDb_1.4.3       IRanges_2.2.9
## [21] S4Vectors_0.6.6          BiocGenerics_0.14.0
## [23] phyloseq_1.12.2
##
## loaded via a namespace (and not attached):
##  [1] Biobase_2.28.0       splines_3.2.2        foreach_1.4.3
##  [4] Formula_1.2-1        assertthat_0.1       latticeExtra_0.6-26
##  [7] yaml_2.1.13          RSQLite_1.0.0        chron_2.3-47
## [10] digest_0.6.9         RColorBrewer_1.1-2   XVector_0.8.0
## [13] colorspace_1.2-6     htmltools_0.2.6      Matrix_1.2-2
## [16] plyr_1.8.3           XML_3.98-1.3         genefilter_1.50.0
## [19] zlibbioc_1.14.0      xtable_1.7-4         BiocParallel_1.2.22
## [22] annotate_1.46.1      mgcv_1.8-7           lazyeval_0.1.10
## [25] nnet_7.3-11          mnormt_1.5-3         proto_0.3-10
## [28] survival_2.38-3      RJSONIO_1.3-0        magrittr_1.5
## [31] evaluate_0.8         nlme_3.1-122         MASS_7.3-44
## [34] foreign_0.8-66       tools_3.2.2          data.table_1.9.6
## [37] formatR_1.2.1        stringr_1.0.0        munsell_0.4.3
## [40] locfit_1.5-9.1       cluster_2.0.3        AnnotationDbi_1.30.1
## [43] lambda.r_1.1.7       Biostrings_2.36.4    ade4_1.7-2
## [46] futile.logger_1.4.1  iterators_1.0.8      biom_0.3.12
## [49] igraph_1.0.1         labeling_0.3         rmarkdown_0.9.5
## [52] codetools_0.2-14     multtest_2.24.0      DBI_0.3.1
## [55] R6_2.1.1             knitr_1.11           Hmisc_3.17-0
## [58] futile.options_1.0.0 ape_3.3              stringi_1.0-1
## [61] geneplotter_1.46.0   rpart_4.1-10         acepack_1.3-3.3
```