Random Forest Modelling of the Lake Erie microbial community

Contents

Author: Michelle Berry

Date: 2/24/15

This is some sample code of how to run both regression and classification random forest models with microbial community composition datasets.

In this example, we are working with illumina 16s data that has already been processed into an otu and taxonomy table. The samples were collected from Western Lake Erie between May and November 2014 at three different locations. Bacterial DNA was extracted from three types of samples:

1. Full community samples: whole lake water was syringe-filtered onto a .22um PES filter to represent the "full" bacterial community (yes i know there are picoplankton that are smaller than .22um).
2. Algal/particle: This fraction targets cyanobacterial colonies and attached bacteria. Whole lake water was passed through a 100um mesh and the retentate was then filtered onto a glass fiber filter.
3. Free-living bacteria: This fraction targets just the free-living bacteria, which we defined as anything between 3um and .22um. Lake water was prefiltered through a 53um mesh and then passed through 3um and .22 um filters.

Data Import

```
# Load libraries
library(phyloseq)
packageVersion("phyloseq")
```

```
## [1] '1.10.0'
```

```
library(ggplot2)
library(reshape2)
library(plyr)
library(scales)
library(knitr)
library(randomForest)
```

```r
# Set working directory
setwd("~/habs/genomics/data/")

# mothur file import
sharedfile = "mothur/furthestneighbor.shared"
taxfile = "mothur/furthestneighbor.taxonomy"
mothurdata = import_mothur(mothur_shared_file = sharedfile ,
                           mothur_constaxonomy_file = taxfile )
mothurdata
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 15633 taxa and 221 samples ]
## tax_table()   Taxonomy Table:    [ 15633 taxa by 6 taxonomic ranks ]
```

```r
# Taxonomy names are listed as Rank1, Rank2 . . .
colnames(tax_table(mothurdata))
```

```
## [1] "Rank1" "Rank2" "Rank3" "Rank4" "Rank5" "Rank6"
```

```r
# Change to Kingdom, Phylum . . .
colnames(tax_table(mothurdata)) <- c("Kingdom","Phylum","Class","Order","Family","Genus")

# Import metadata and merge with other object
metadata = "other/metadata_bloom.txt"
map = import_qiime_sample_data(metadata)
map1 <-map[-1,]
colnames(otu_table(mothurdata))<-map1$SampleID
moth_merge = merge_phyloseq(mothurdata,map)

# What variables are in our metadata?
sample_variables(moth_merge)
```

```
##  [1] "SampleID"       "Samplenum"      "Date"           "DateContinuous"
##  [5] "Station"        "Fraction"       "Photonum"       "pH"
##  [9] "Type"           "Intensive"      "Arrival"        "Departure"
## [13] "Sky"            "StationDepth"   "Secchi"         "Temp"
## [17] "Turbidity"      "ParMC"          "DissMC"         "Phycocyanin"
## [21] "Chla"           "Bloom"          "LogPhyco"       "LogChla"
## [25] "LogParMC"
```

```r
# lets filter out samples such as blanks, mock communities and samples that didnt amplify
moth_good <- subset_samples(moth_merge,Type=="sample" & Intensive=="n")
moth_good <- subset_taxa(moth_good,
                         Kingdom == "Bacteria" &
                         Class != "Chloroplast" &
                        Family!="mitochondria"
                          )
moth_good <- prune_taxa(taxa_sums(moth_good)>0,moth_good)
moth_good
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 11765 taxa and 174 samples ]
## sample_data() Sample Data:       [ 174 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 11765 taxa by 6 taxonomic ranks ]
```

Classification rf: modelling algal-associated vs free-living

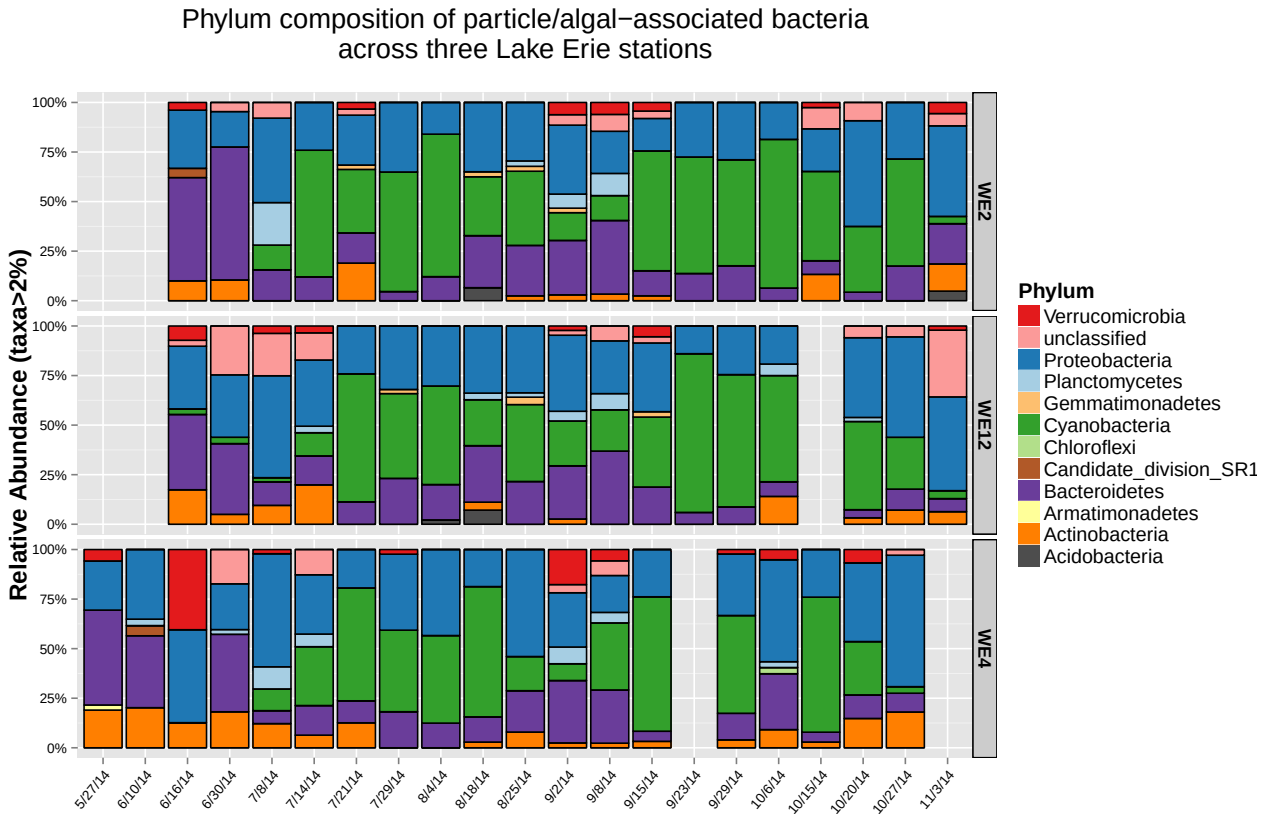
For this random forest we want to build a model which will classify samples as either "free-living" or "algal/particle". In doing this, we will also pull out which OTUs seem to be associated with the free-living lifestyle, vs attachment to colonial algae

First, we need to do a few things to filter our data and put it in the right format

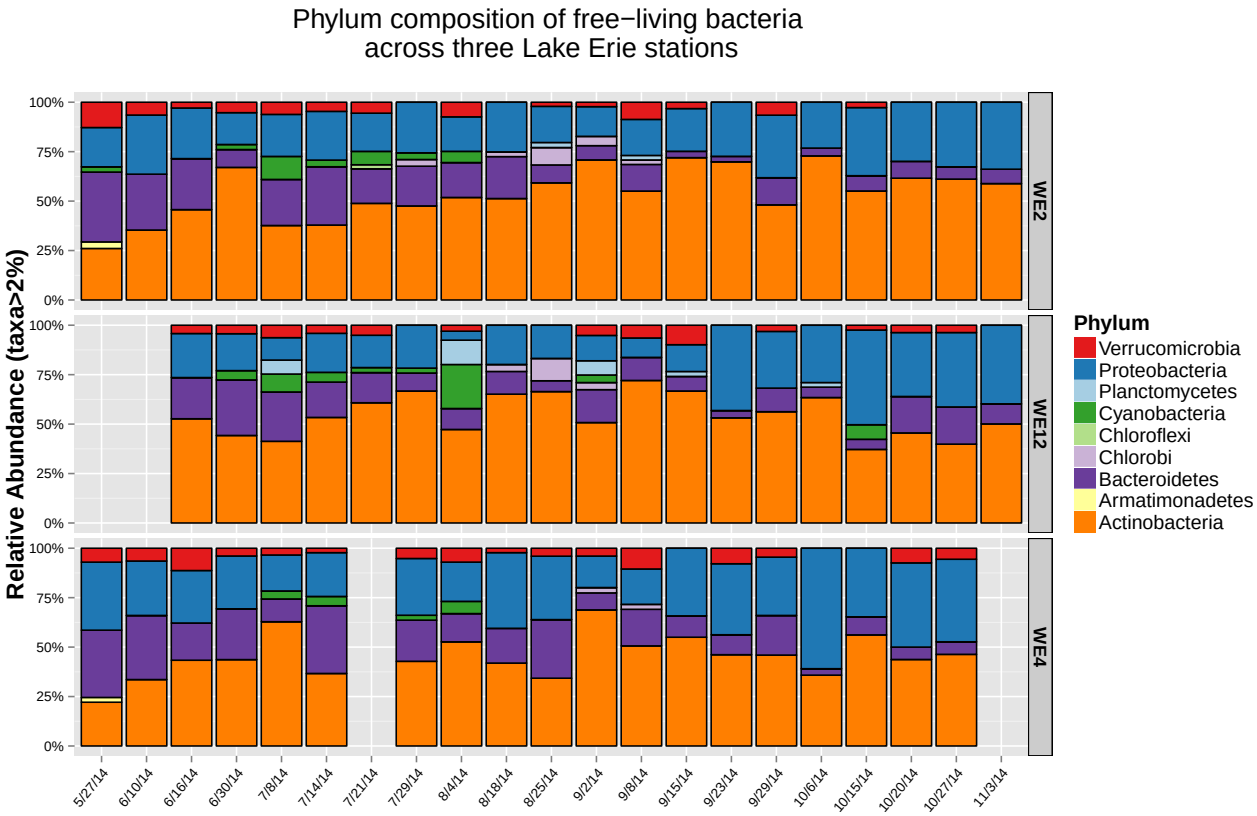
```
# Select only samples of type 100L (Algal/particle) or 22NA (free-living)
moth_subset<-subset_samples(moth_good,Fraction=="100L"|Fraction=="22NA")

# Normalize read counts by scaling to smallest library size
moth_scaled<-scale_reads(moth_subset,min(sample_sums(moth_subset)))
```

Lets make some barplots to get a sense of what these communities look like. Here is a barplot of phylum composition from all three locations (WE2, WE12, WE4)

```
# Plot phylum level composition of Particle/algal fraction
F100L <- subset_samples(moth_scaled,Fraction=="100L")

# Set Colors (RColorBrewer)
phy <- c(Acidobacteria="#4D4D4D",Actinobacteria="#ff7f00",Armatimonadetes="#ffff99",
         Bacteroidetes="#6a3d9a",Candidate_division_SR1 ="#b15928" ,
         Chlorobi="#cab2d6", Chloroflexi="#B2DF8A",
         Chloroplast="darkgreen",Cyanobacteria="#33A02C",
         Gemmatimonadetes="#fdbf6f",Planctomycetes="#A6CEE3",
         Proteobacteria="#1F78B4", unclassified="#fb9a99",
         Verrucomicrobia ="#e31a1c")

F100L.long<-transform_and_melt(physeq = F100L,taxrank = "Phylum",prune = .02)

make_tax_barplot(df = F100L.long,x = "Date",tax = "Phylum",
                 title = "Phylum composition of particle/algal-associated bacteria
across three Lake Erie stations\n",
                 colors=phy,
                 xlab="",
                 ylab="Relative Abundance (taxa>2%)")
```




```
# Plot phylum level composition of Free-living fraction
F22NA <- subset_samples(moth_scaled,Fraction=="22NA")

F22NA.long<-transform_and_melt(physeq = F22NA,taxrank = "Phylum",prune = .02)

make_tax_barplot(df = F22NA.long,x = "Date",tax = "Phylum",
                 title = "Phylum composition of free-living bacteria
across three Lake Erie stations\n",
                 colors=phy,
                 xlab="",
                 ylab="Relative Abundance (taxa>2%)")
```



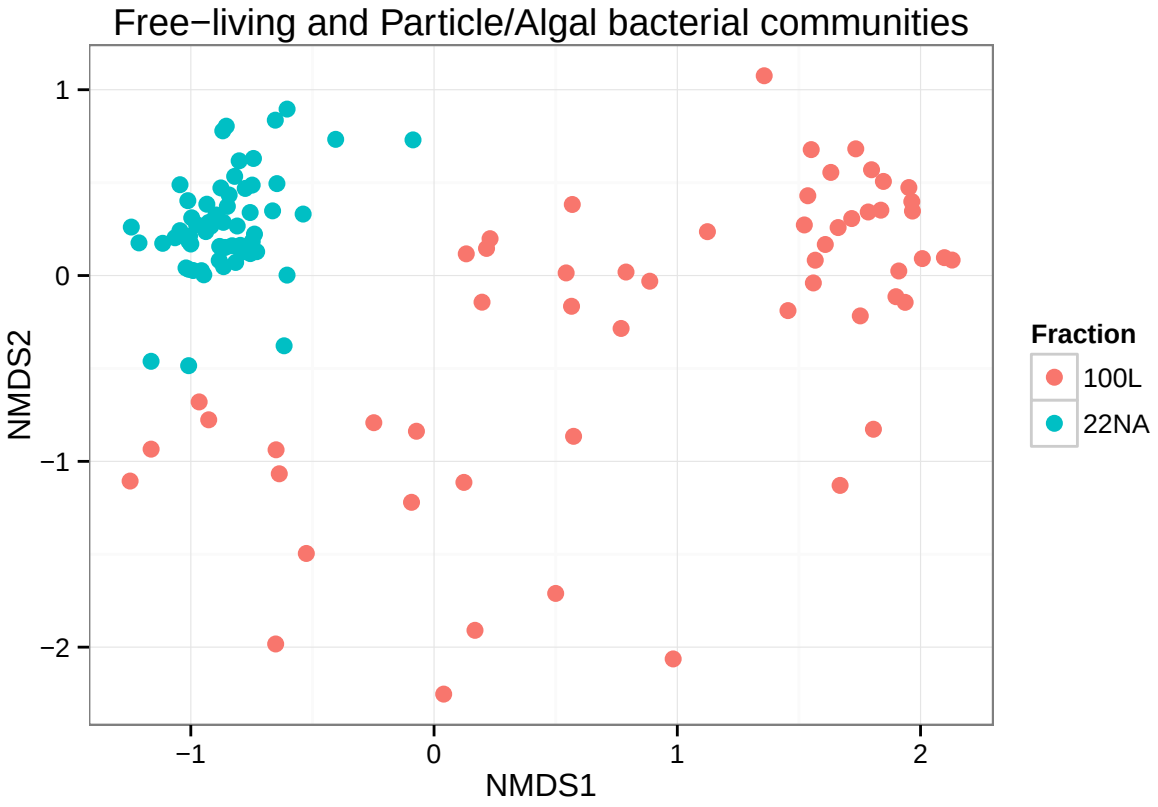
Here is an NMDS of the samples colored by free-living or particle/algal. We can already see vsee that these communities are very well segregated, so our random forest model will probably work very well.

```
# Make an NMDS colored by sample type
# Plot NMDS
theme_set(theme_bw())
bray.nmds = ordinate(moth_scaled,method="NMDS","bray")
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.1432346
## Run 1 stress 0.1598872
## Run 2 stress 0.1625284
## Run 3 stress 0.1538138
## Run 4 stress 0.1526169
## Run 5 stress 0.16099
## Run 6 stress 0.159927
## Run 7 stress 0.1560044
## Run 8 stress 0.1613005
## Run 9 stress 0.1633926
## Run 10 stress 0.1638689
## Run 11 stress 0.160981
## Run 12 stress 0.1512389
## Run 13 stress 0.1593861
## Run 14 stress 0.1568865
## Run 15 stress 0.1579673
## Run 16 stress 0.1597686
## Run 17 stress 0.1624696
```

```
## Run 18 stress 0.1503891
## Run 19 stress 0.1603744
## Run 20 stress 0.1539144
```

```
plot_ordination(moth_scaled,bray.nmds,color = "Fraction") +
  geom_point(size=3) +
  ggtitle("Free-living and Particle/Algal bacterial communities")
```




```
# Make a dataframe of training data with OTUs as column and samples as rows
t_otu <- t(otu_table(moth_scaled))
dim(t_otu)
```

```
## [1] 115 788
```

We have 115 samples and 788 OTUs

```
# Make one column for our predictor variable "Fraction"
Fraction <- as.factor(sample_data(moth_scaled)$Fraction)

# Combine them into 1 data frame
rf_data<-data.frame(Fraction,t_otu)
```

Now we will use the randomForest package to train and test our random forest model using the "out of bag" error to estimate our model error. OOB is a nice feature of random forest models whereby since the training data is bootstrapped, you only use approximately 2/3 of the data at each iteration. The remaining 1/3 or "out of bag" data can be used to validate your model. This removes the need to use another form of cross-validation such as using a separate validation set or k-folds.

Results

```
# It is important to set a seed for reproducability
# By default, randomForest uses p/3 variables when building
# a random forest of regression trees and root(p)
# variables when building a random forest of classification trees.
# In this case, root(p) = 28
set.seed(1)
rf.erie <- randomForest(Fraction~.,data=rf_data,ntree=100)
print(rf.erie)
```

```
##
## Call:
##  randomForest(formula = Fraction ~ ., data = rf_data, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 28
##
##         OOB estimate of  error rate: 1.74%
## Confusion matrix:
##      100L 22NA class.error
## 100L   56    0  0.00000000
## 22NA    2   57  0.03389831
```

Our out of bag error is 1.74%

```
# What variables are stored in the output?
names(rf.erie)
```

```
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
## [19] "terms"
```

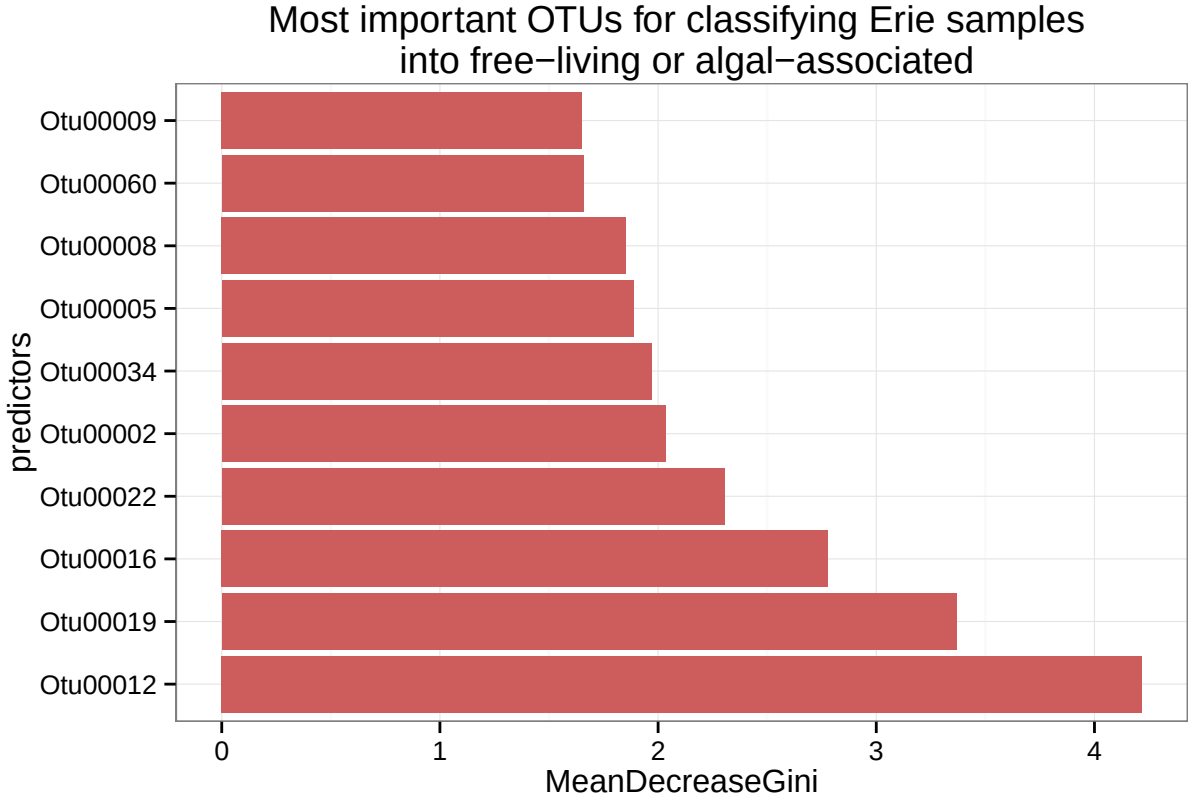
This model worked really well! We can very accurately classify our bacterial communities into free-living or particle/algal-associated

plots

Lets make some plots of the most important variables in our model

```
# Select the importance vactor for the predictors.
# For a regression tree this is measured by mean decrease in MSE
# due to that variable
imp <- importance(rf.erie)
imp <- data.frame(predictors=rownames(imp),imp)

# Order the predictor levels by importance
```

```
imp.sort <- arrange(imp,desc(MeanDecreaseGini))
imp.sort$predictors <- factor(imp.sort$predictors,levels=imp.sort$predictors)

# Select the top 10 predictors
imp.10<- imp.sort[1:10,]


# ggplot
ggplot(imp.10,aes(x=predictors,y=MeanDecreaseGini))+
  geom_bar(stat="identity",fill="indianred") +
  coord_flip()+
  ggtitle("Most important OTUs for classifying Erie samples \n into free-living or algal-associated")
```




```
# What are those OTUs?
otunames<- imp.10$predictors
r <- rownames(tax_table(moth_scaled)) %in% otunames
kable(tax_table(moth_scaled)[r,])
```

	Kingdom	Phylum	Class	Order	Family	Genus
Otu00002	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	hgcI_clade
Otu00005	Bacteria	Cyanobacteria	Cyanobacteria	SubsectionI	FamilyI	Microcystis
Otu00008	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	unclassified
Otu00009	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	hgcI_clade
Otu00012	Bacteria	Proteobacteria	Betaproteobacteria	Burkholderiales	Oxalobacteraceae	unclassified
Otu00016	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	Candidatus_Plank
Otu00019	Bacteria	Proteobacteria	Betaproteobacteria	Methylophilales	Methylophilaceae	LD28_freshwater_

	Kingdom	Phylum	Class	Order	Family	Genus
Otu00022	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	hgcI_clade
Otu00034	Bacteria	Proteobacteria	Alphaproteobacteria	Rhodobacterales	Rhodobacteraceae	Rhodobacter
Otu00060	Bacteria	Proteobacteria	Betaproteobacteria	Burkholderiales	Alcaligenaceae	MWH-UniP1_aqua

Regression rf: modelling microcystin levels of the whole bacterial community

In this random forest we will attempt to predict the microcystin toxin levels of a bacterial community based on its OTU composition. This is essentially a regression problem using OTU relative abundances as predictors and toxin levels as our outcome. Because there is a time component to these samples we will also include time as a variable.

```
# Subset the data to select just full community samples
FCNA <- subset_samples(moth_good,Fraction=="CNA")
FCNA.scaled<-scale_reads(FCNA, min(sample_sums(FCNA)))

FCNA.scaled
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 4703 taxa and 59 samples ]
## sample_data() Sample Data:       [ 59 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 4703 taxa by 6 taxonomic ranks ]
```

This is a large number of OTUs, so to speed up the model a bit we will remove all rare OTUs that are probably not very biologically relevant

```
# Transform to proportions
FCNA.proportions = transform_sample_counts(FCNA.scaled,function(x){x/sum(x)})

# Remove OTUs below .001 within a sample
otu_table(FCNA.proportions)[otu_table(FCNA.proportions)<.001] <- 0
FCNA.pruned = prune_taxa(taxa_sums(FCNA.proportions)>0, FCNA.proportions)
FCNA.pruned
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 488 taxa and 59 samples ]
## sample_data() Sample Data:       [ 59 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 488 taxa by 6 taxonomic ranks ]
```

Now we only have 488 OTUs to put into our model

```
# Make a dataframe with OTUs as columns, samples as rows,
# and one column for our predictor variable "Microcystin"
t_otu <- t(otu_table(FCNA.pruned))
dim(t_otu)
```

```
## [1]  59 488
```

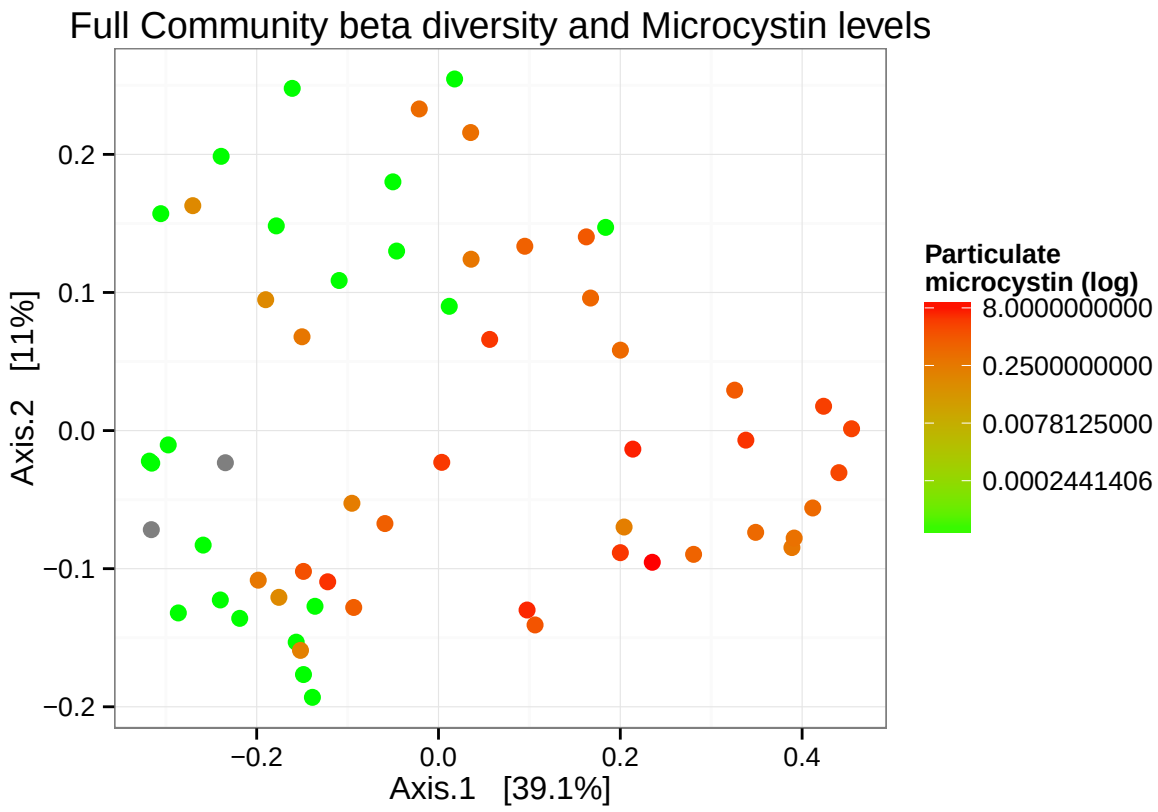
```r
# We have 59 samples and 488 OTUs

# Assign particulate Microcystin to a variable
ParMC <- as.vector(sample_data(FCNA.scaled)$ParMC)
# Log transform ParMC because it is highly skewed
ParMC <- log2(ParMC)


# Assign time to a variable
time <- as.vector(sample_data(FCNA.scaled)$DateContinuous)

# Combine the otu table with ParMC
train <- cbind(ParMC,t_otu,time)
dim(train)
```

```
## [1]  59 490
```

```r
# Remove the NA rows
NA_idx <- is.na(ParMC)
train_noNA <- subset(train,!NA_idx)

# Lastly, remove the OTUs that are now all zeros
zero_otus_idx <- which(colSums(train_noNA[,-1])==0)
train_noNA<-train_noNA[,-zero_otus_idx]
dim(train_noNA)
```

```
## [1]  57 488
```

```r
# Make an NMDS colored by particulate microcystin lvels
# Plot NMDS
bray.nmds = ordinate(FCNA.pruned,method="PCoA","bray")
plot_ordination(FCNA.pruned,bray.nmds,color = "ParMC") +
  geom_point(size=3) +
  ggtitle("Full Community beta diversity and Microcystin levels") +
  scale_colour_gradient(name="Particulate \nmicrocystin (log)",
                        trans="log2",low="green", high="red")
```



results

```
# It is important to set a seed for reproducability
# By default, randomForest uses p/3 variables when building
# a random forest of regression trees and root(p)
# variables when building a random forest of classification trees.
# In this case, p/3 = 163
set.seed(2)
rf.mc <- randomForest(ParMC~.,data=train_noNA,ntree=100,keep.inbag = T)
print(rf.mc)
```

```
##
## Call:
##  randomForest(formula = ParMC ~ ., data = train_noNA, ntree = 100,      keep.inbag = T)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 162
##
##           Mean of squared residuals: 32.8791
##                     % Var explained: 49.8
```

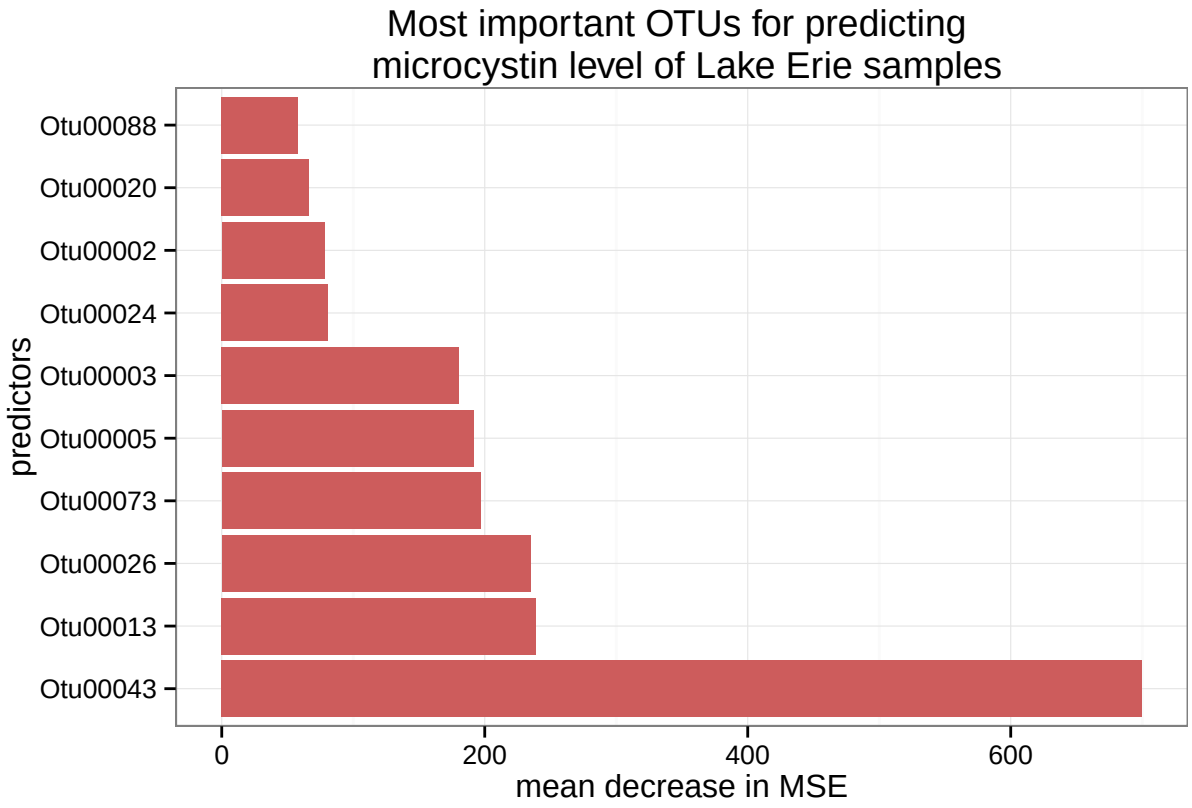
It looks like this model has a test Mean squared error of 32.6 and is able to explain half of the variation in Microcystin among the samples

```r
names(rf.mc)
```

```
##  [1] "call"            "type"            "predicted"
##  [4] "mse"             "rsq"             "oob.times"
##  [7] "importance"      "importanceSD"    "localImportance"
## [10] "proximity"       "ntree"           "mtry"
## [13] "forest"          "coefs"           "y"
## [16] "test"            "inbag"           "terms"
```

plots

```r
# Select the importance vactor for the predictors.
# For a regression tree this is measured by mean decrease in MSE
# due to that variable
imp <- importance(rf.mc)
imp <- data.frame(predictors=rownames(imp),imp)

# Order the predictor levels by importance
imp.sort <- arrange(imp,desc(IncNodePurity))
imp.sort$predictors <- factor(imp.sort$predictors,levels=imp.sort$predictors)

# Select the top 10 predictors
imp.10 <- imp.sort[1:10,]

# ggplot
ggplot(imp.10,aes(x=predictors,y=IncNodePurity))+
  geom_bar(stat="identity",fill="indianred") +
  coord_flip()+
  ggtitle("Most important OTUs for predicting \n microcystin level of Lake Erie samples") +
  ylab("mean decrease in MSE")
```




```
# What are those OTUs?
otunames<- imp.10$predictors
r <- rownames(tax_table(FCNA)) %in% otunames
kable(tax_table(FCNA)[r,])
```

	Kingdom	Phylum	Class	Order	Family	Genus
Otu00002	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	hgcI_clade
Otu00003	Bacteria	Actinobacteria	Actinobacteria	Frankiales	Sporichthyaceae	hgcI_clade
Otu00005	Bacteria	Cyanobacteria	Cyanobacteria	SubsectionI	FamilyI	Microcystis
Otu00013	Bacteria	Proteobacteria	Alphaproteobacteria	Rhodospirillales	Acetobacteraceae	Roseomonas
Otu00020	Bacteria	Bacteroidetes	Cytophagia	Cytophagales	Cytophagaceae	unclassified
Otu00024	Bacteria	Cyanobacteria	Cyanobacteria	SubsectionI	FamilyI	unclassified
Otu00026	Bacteria	Verrucomicrobia	OPB35_soil_group	unclassified	unclassified	unclassified
Otu00043	Bacteria	Cyanobacteria	Cyanobacteria	SubsectionIII	FamilyI	Pseudanabaena
Otu00073	Bacteria	Cyanobacteria	Cyanobacteria	SubsectionI	FamilyI	unclassified
Otu00088	Bacteria	Proteobacteria	Alphaproteobacteria	Rickettsiales	Rickettsiaceae	Rickettsia

This is really interesting. Its telling us that the presence of pseudoanabena is the best predictor of toxicity. From other data we have, we know that psudoanabena only blooms in the later part of the season when Microcystis abundance is still high but toxin levels are much lower. There could be an interesting ecological interaction going on between microcystis and psuedoanabena which may relate to toxin production.

Functions

Here are some functions that i used previously in the code

scale_reads

```
# scale reads to the smallest library size
# using method in McMurdie and Holmes, Plos CompBio (2014)
scale_reads <- function(physeq,n){
  physeq.scale <- transform_sample_counts(physeq, function(x) {(n*x/sum(x))})
  otu_table(physeq.scale) = floor(otu_table(physeq.scale))
  physeq.scale = prune_taxa(taxa_sums(physeq.scale)>0,physeq.scale)
  return(physeq.scale)
}
```

transform_and_melt

```
# This function takes a phyloseq object,
# agglomerates OTU's to the desired taxonomic rank,
# prunes out OTUs below a certain relative proportion
# in a sample (ie 1% ) and melts the phyloseq object into long format.
transform_and_melt<-function(physeq,taxrank,prune){

  # Agglomerate all otu's by phylum level
  physeq_phylum<-tax_glom(physeq,taxrank = "Phylum")

  # Create a new phyloseq object which removes taxa from each sample
  # below the prune parameter
  physeq_phylum.prune = transform_sample_counts(physeq_phylum,
                                                function(x){x/sum(x)})
  otu_table(physeq_phylum.prune)[otu_table(physeq_phylum.prune)<prune] <- 0
  physeq_phylum.prune  = prune_taxa(taxa_sums(physeq_phylum.prune)>0,
                                    physeq_phylum.prune)

  # Melt into long format and sort by samplenum and phylum
  physeq_phylum.long<-psmelt(physeq_phylum.prune)
  physeq_phylum.long<-arrange(physeq_phylum.long,Samplenum,Phylum)


  # Reorder factor levels for dates
  physeq_phylum.long<-order_dates(physeq_phylum.long)

  # Reorder factor levers for stations
  physeq_phylum.long$Station <-factor(physeq_phylum.long$Station,
                                      levels=c("WE2","WE12","WE4"))
  return(physeq_phylum.long)
}
```

make_tax_barplot

```
# This function takes  a data frame in long format
# (such as the output from transform_and_melt) and
# produces a stacked barplot of community composition.
```

```
make_tax_barplot<-function(df,x,tax,title,colors,xlab,ylab){
  ggplot(df,aes_string(x=x,y="Abundance",fill=tax)) +
        facet_wrap(~Station,scales="free")+
        geom_bar(stat="identity")+
        geom_bar(stat="identity",position="fill",colour="black",show_guide=FALSE)+
   scale_y_continuous(labels = percent_format())+
  scale_fill_manual(values = colors) +
    theme(axis.title.x = element_text(size=16,face="bold"),
          axis.text.x = element_text(angle=50, colour = "black", vjust=1, hjust = 1, size=9),
          axis.text.y = element_text(colour = "black", size=9),
          axis.title.y = element_text(face="bold", size=16),
          plot.title = element_text(size = 18),
          legend.title = element_text(size=14),
          legend.text = element_text(size = 13),
          legend.position="right",
          strip.text.x = element_text(size=12, face="bold"),
          strip.text.y = element_text(size=12, face="bold"),
          strip.background = element_rect(colour="black"))+
    guides(fill = guide_legend(reverse= TRUE,keywidth=1,keyheight=1))+
    xlab(xlab)+
    ylab(ylab)+
    ggtitle(title)

}
```