AbxD01 Analysis

Alyx Schubert

September 29, 2014

# Introduction

This is a digital notebook to accompany the paper titled, “*” that will be published in* . It was written in R markdown and converted to html using the R knitr package. This enables us to embed the results of our analyses directly into the text to allow for a completely reproducible data analysis pipeline. A github repository is available where you can pull down your own version of the notebook to modify our analysis or adapt it to your analysis.

This document was generated using mothur v.1.33 and R.

# Prcessing 16S gene sequence data in mothur

## First, get the required files.

Sequence files can be found here \_.

The following files can be found through the mothur wiki: silva.bacteria.fasta trainset9\_032012.pds.tax trainset9\_032012.pds.fasta

The following files can be found on the github repository for this notebook: abxD01.files

## Now for the curation pipeline.

These sequences were generated using the Illumina MiSeq Platform. For data processing, we followed the MiSeq SOP outlined in the [Kozich et al. manuscript](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3753973/). These commands come from the outline described in the [MiSeq SOP](http://www.mothur.org/wiki/MiSeq_SOP).

make.contigs(file=abxD01.files, processors=14)  
summary.seqs(fasta=abxD01.trim.contigs.fasta, processors=14)  
screen.seqs(fasta=abxD01.trim.contigs.fasta, group=abxD01.contigs.groups, summary=abxD01.trim.contigs.summary, maxambig=0, maxlength=275, processors=14)  
unique.seqs(fasta=abxD01.trim.contigs.good.fasta)  
count.seqs(name=abxD01.trim.contigs.good.names, group=abxD01.contigs.good.groups)  
summary.seqs(count=abxD01.trim.contigs.good.count\_table)  
#Make sure silva.bacteria.fasta file in folder  
pcr.seqs(fasta=silva.bacteria.fasta, start=11894, end=25319, keepdots=F, processors=14)  
system(mv silva.bacteria.pcr.fasta silva.v4.fasta)  
summary.seqs(fasta=silva.v4.fasta)  
align.seqs(fasta=abxD01.trim.contigs.good.unique.fasta, reference=silva.v4.fasta, processors=14)  
summary.seqs(fasta=abxD01.trim.contigs.good.unique.align, count=abxD01.trim.contigs.good.count\_table)  
screen.seqs(fasta=abxD01.trim.contigs.good.unique.align, count=abxD01.trim.contigs.good.count\_table, summary=abxD01.trim.contigs.good.unique.summary, start=1968, end=11550, maxhomop=8)  
summary.seqs(fasta=current, count=current)  
filter.seqs(fasta=abxD01.trim.contigs.good.unique.good.align, vertical=T, trump=.)  
unique.seqs(fasta=abxD01.trim.contigs.good.unique.good.filter.fasta, count=abxD01.trim.contigs.good.good.count\_table)  
pre.cluster(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.fasta, count=abxD01.trim.contigs.good.unique.good.filter.count\_table, diffs=2)  
chimera.uchime(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.count\_table, dereplicate=t, processors=14)  
remove.seqs(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.fasta, accnos=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.accnos)  
summary.seqs(fasta=current, count=current)  
#Make sure the reference and training set files are in the analysis folder  
classify.seqs(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.count\_table, reference=trainset9\_032012.pds.fasta, taxonomy=trainset9\_032012.pds.tax, cutoff=80)  
remove.lineage(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.count\_table, taxonomy=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.taxonomy, taxon=Chloroplast-Mitochondria-unknown-Archaea-Eukaryota)  
remove.groups(count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.pick.count\_table, fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pick.fasta, taxonomy=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.pick.taxonomy, groups=mock1-mock2-mock3-mock4-mock5-mock6-mock7-mock8)  
  
#renamed fasta, count\_table, taxonomy to have shorter names  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.pick.pick.taxonomy abxD01.final.taxonomy)  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.pick.pick.count\_table abxD01.final.count\_table)  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pick.pick.fasta abxD01.final.fasta)  
  
set.current(fasta=abxD01.final.fasta, count=abxD01.final.count\_table, taxonomy=abxD01.final.taxonomy)

## Creating phylotypes

Continuing the pipeline outlined on the mothur wiki's MiSeq SOP.

#PHYLOTYPE analysis:  
#Note: label=1 is the genus level, which is the finest you can do, then label=2 is family and so forth  
phylotype(taxonomy=abxD01.final.taxonomy)  
make.shared(list=abxD01.final.tx.list, count=abxD01.final.count\_table)  
count.groups()  
#The following size 1625 was chosen based on the results of count.groups(). I tried to minimize the number of samples at Day 0 that were dropped at that minimum cutoff.  
sub.sample(shared=abxD01.final.tx.shared, size=1625)  
classify.otu(list=abxD01.final.tx.list, count=abxD01.final.count\_table, taxonomy=abxD01.final.taxonomy)  
#tx.2=family, tx.3=order, etc  
get.relabund(shared=abxD01.final.tx.2.subsample.shared)   
get.relabund(shared=abxD01.final.tx.3.subsample.shared)  
get.relabund(shared=abxD01.final.tx.4.subsample.shared)   
get.relabund(shared=abxD01.final.tx.5.subsample.shared)   
get.oturep(column=abxD01.final.dist, list=abxD01.final.tx.list, label=2-3-4-5, fasta=abxD01.final.fasta, count=abxD01.final.count\_table)

## Creating OTUs

It is worth noting that while phylotype analysis followed the listed protocol exactly, we deviated from the listed SOP for the OTU analysis. This was due to the extensive size of this dataset and limitations in our computing abilities. Thus, in a **separate** folder we copied over the following files from earlier in the protocol: abxD01.trim.contigs.good.unique.good.filter.unique.fasta abxD01.trim.contigs.good.unique.good.filter.count\_table trainset9\_032012.pds.fasta trainset9\_032012.pds.tax

**If this isn't done in a new folder, you will write over all your previous files because they have the same name.**

#Changed diffs from 2 to 3  
pre.cluster(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.fasta, count=abxD01.trim.contigs.good.unique.good.filter.count\_table, diffs=3)  
chimera.uchime(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.count\_table, dereplicate=t, processors=14)  
remove.seqs(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.fasta, accnos=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.accnos)  
summary.seqs(fasta=current, count=current)  
#Make sure the reference and training set files are in the analysis folder  
classify.seqs(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.count\_table, reference=trainset9\_032012.pds.fasta, taxonomy=trainset9\_032012.pds.tax, cutoff=80)  
remove.lineage(fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.count\_table, taxonomy=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.taxonomy, taxon=Chloroplast-Mitochondria-unknown-Archaea-Eukaryota)  
remove.groups(count=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.pick.count\_table, fasta=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pick.fasta, taxonomy=abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.pick.taxonomy, groups=mock1-mock2-mock3-mock4-mock5-mock6-mock7-mock8)  
  
#renamed fasta, count\_table, taxonomy to have shorter names  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pds.wang.pick.pick.taxonomy abxD01.final.taxonomy)  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.uchime.pick.pick.pick.count\_table abxD01.final.count\_table)  
system(cp abxD01.trim.contigs.good.unique.good.filter.unique.precluster.pick.pick.pick.fasta abxD01.final.fasta)  
  
set.current(fasta=abxD01.final.fasta, count=abxD01.final.count\_table, taxonomy=abxD01.final.taxonomy)  
  
#OTU ANALYSIS:  
#changed SOP taxlevel from 4 to 5  
cluster.split(fasta=abxD01.final.fasta, count=abxD01.final.count\_table, taxonomy=abxD01.final.taxonomy, splitmethod=classify, taxlevel=5, cutoff=0.20)  
make.shared(list=abxD01.final.an.unique\_list.list, count=abxD01.final.count\_table, label=0.03)  
count.groups()  
  
#Note: the size varies based on what you see on count.groups, need to check the distribution of day 0 groups and clindamycin group  
sub.sample(shared=abxD01.final.an.unique\_list.shared, size=1625)  
classify.otu(list=abxD01.final.an.unique\_list.list, count=abxD01.final.count\_table, taxonomy=abxD01.final.taxonomy, label=0.03, cutoff=80)  
get.relabund(shared=abxD01.final.an.unique\_list.0.03.subsample.shared)  
get.oturep(column=abxD01.final.dist, list=abxD01.final.an.unique\_list.list, label=0.03, fasta=abxD01.final.fasta, count=abxD01.final.count\_table)  
collect.single(shared=abxD01.final.an.unique\_list.0.03.subsample.shared, calc=chao-invsimpson-sobs-simpsoneven-shannon-jclass, freq=100)  
rarefaction.single(freq=100)  
#Note summary.single--use the original shared file, and put subsample as same as before, will subsample 1000 times and get avgs  
summary.single(calc=nseqs-coverage-chao-sobs-invsimpson-simpsoneven-shannoneven-shannon, shared=abxD01.final.an.unique\_list.shared, subsample=1625)

# Data Analysis and Paper Figures

## Figure 1

This figure shows the differences in the communities' structures following different antibiotic treatments. The average *C. difficile* CFU/g Feces is also calculated in this code for each treatment group.

This figure was built using the following R code. The input files are listed in Github.

#################################################  
#Purpose: Convert csv matrix of columns you want converted into barcharts  
#input: csv file following this heading format format: sampleID, expgroup, nextDayCFU, Otu001, Otu002, Otu003, etc  
#ex:file<-read.csv("~/Desktop/mothur/abxD01/barcharts/abxD01.final.tx.5.subsample.relabund.topdose.forlogscale2.csv  
#Note: It is assumed that anything after the expgroup column are vectors containing data to be split by the groups in 'expgroup'   
# and averaged with SD measured.   
#Output: matrices avg[], std[], bp[y] with the barplot coordinates info stored for each subgraph  
#################################################  
# Parameters to change:  
# CSV file: Group expgroup Otu001... (limited by most abund, end with 'Other', OTUs normalized +0.0001, expgroups #'d by graph order & sorted by first graph)  
file<-read.csv("~/Desktop/mothur/abxD01/barcharts/abxD01.final.tx.1.subsample.relabund.topdose2.forlogscale.csv", header=T)  
fileIDS<-read.csv("~/Desktop/mothur/abxD01/barcharts/topdose\_tx1\_barchart\_ids.csv", header=T)  
# Y Labels for each graph:   
abx<-c("Untreated", "Ciprofloxacin", "Clindamycin", "Vancomycin", "Streptomycin", "Cefoperazone", "Ampicillin", "Metronidazole")  
  
# If you want each OTU on the Y axis to be sorted by the most abundant phylum and then decreasing abundance within that phylum, change to TRUE  
# The default is false, which means sort be decreasing relative abundance in the top group (untreated/control)  
sortbyphyl<-TRUE  
  
# Highlight all and run!  
#################################################  
  
file<-file[,-1] #delete the first col of group names  
avgs=NULL  
avgs <-data.frame(levels(file$expgroup))  
colnames(avgs) <- c("expgroup")  
stds=NULL  
stds <-data.frame(levels(file$expgroup))  
colnames(stds) <- c("expgroup")  
  
  
#Calculate average cdiff  
avgCD=NULL  
sdCD=NULL  
avgCD<-tapply(file$nextDayCFU, file$expgroup, mean)  
avgCD<-format(avgCD, scientific=TRUE, digits=2)  
sdCD<-tapply(file$nextDayCFU, file$expgroup, sd)  
file<-file[,-2] #delete the nexDayCFU column  
  
  
for(i in 1:ncol(file)){  
   
 if(i==1){  
 ids<-names(file)  
 }  
 else{  
 columni <- data.frame(file[,i])  
 colnames(columni) <- c("columndata")  
   
 #calculates mean for each column in file by expgroup and stores it in avgs data frame  
 poop<- aggregate(columni$columndata~file$expgroup, FUN=mean) #poop stores the aggregated means for each group for this specific column  
 colnames(poop) <- c(ids[1], ids[i])  
   
 avgs<- merge(avgs, poop, by ="expgroup")  
   
 #calculates st dev for each column in file by expgroup and stores it in stds data frame  
 poop2<- aggregate(columni$columndata~file$expgroup, FUN=sd) #poop stores the aggregated means for each group for this specific column  
 colnames(poop2) <- c(ids[1], ids[i])  
   
 stds<- merge(stds, poop2, by ="expgroup")  
 }  
}  
  
ordIDS <- fileIDS[order(fileIDS[,2]), ] #sort by the phyla  
totalOTU <- dim(ordIDS)[1]  
  
########for doing sort by phylum then within phylum (by relabund)  
if(sortbyphyl == TRUE){  
 ordIDS<-cbind(ordIDS, 0)  
   
 for (i in 1:totalOTU){ #loop to fill out the final column for the phyla groups, while phyla is sorted!!  
 if(i==1){  
 ordIDS[i, 4] <- 1  
 }  
 else{  
 if(ordIDS[i, 2]==ordIDS[i-1, 2]){  
 ordIDS[i, 4] <- ordIDS[i-1, 4]  
 }  
 else{  
 ordIDS[i, 4] <- ordIDS[i-1, 4] + 1  
 }  
   
 }  
 }  
   
 names(ordIDS)[4]<-paste("phynum")  
   
 ##add row for phylum number to the avgs?  
 #avgs<-rbind(avgs, 0)[c(9, 1, 2, 3, 4, 5, 6, 7, 8),] #this is hard coded for the top doses... also dont rerun this multiple times!!! might need to change  
 avgs<-rbind(0,avgs) #this is hard coded for the top doses... also dont rerun this multiple times!!! might need to change, will give error  
   
 numphyla<-max(ordIDS[,4])  
   
 #test\_avg <- avgs[, rev(order(avgs[1,2:length(avgs)-1]))] #sort all but other column (last)  
   
 physums <- data.frame(c(1:numphyla))  
 physums <- cbind(physums, 0)  
 colnames(physums) <- c("Num", "Sum")  
   
 leng<-length(avgs)  
   
   
 #jtime<-NULL  
 #itime<-NULL  
 #inside<-FALSE  
 #count<-0  
 j<-1  
 ##This loop is to fill the physums data frame with the sum of the relabund for each phylum represented  
 for(i in 2:(leng-1)){ #searching through avgs, so leng is number of otus, -1 because i dont want to deal with "Other" column  
 found<-FALSE  
 #itime<-rbind(itime, i)  
 j<-1  
 while(found==FALSE){ #searching through ordIDS  
 test<-ordIDS[j, 1] == names(avgs)[i]  
 if(test) {   
 pnum<-ordIDS[j, 4]  
 found<-TRUE  
 inside<-TRUE  
 avgs[1,i]<-pnum  
 }  
 else{  
 j <- j + 1  
 #count<-count+1  
 #jtime<-rbind(jtime, j)  
   
   
 }  
 } #WARNING: could cause an error if dont find the ID, theres no check for the end of the list  
   
 physums[pnum, 2] <- physums[pnum, 2] + avgs[2, i] ##add the relabund of the otu   
 }  
   
   
 y<-rev(order(physums$Sum))  
 x<-avgs[1, 2:(leng-1)]  
   
   
 ord\_physums<-physums[rev(order(physums[,2])),]  
 sort\_avgs<-data.frame(avgs$expgroup)  
 names(sort\_avgs)[1]<-c("expgroup")  
   
 for(i in 1:numphyla){ #loop is to sort by decreasing phyla and perform internal sort within phylum, return the new sorted avgs to go into mavgs  
 val<-NULL  
 phy<-which(x %in% c(ord\_physums[i,1])) #phy returns the indices for where that phyNum is  
 for(j in 1:length(phy)){ #using locations of phylum i to find in avgs  
 val[j]<-avgs[2,phy[j]+1]  
 }  
 phy<-phy[rev(order(val))]  
   
 for(j in 1:length(phy)){ #now loop through the sorted sub cols and add them to the sort\_avgs  
 sort\_avgs<-cbind(sort\_avgs, avgs[,phy[j]+1])  
 names(sort\_avgs)[length(sort\_avgs)]<-names(avgs)[phy[j]+1]  
 }  
 }  
   
 attach(avgs)  
 sort\_avgs<-cbind(sort\_avgs, Other)  
 detach(avgs)  
   
 sort\_avgs <- sort\_avgs[-1,] #get rid of phylum first row from earlier sorting  
   
 #loop for getting the right ids in the right order as sort  
 ids<-data.frame(1:(totalOTU))  
   
 row.names(sort\_avgs)<-sort\_avgs$expgroup  
 sort\_avgs<-sort\_avgs[,-1]  
   
 for (i in 1:(length(sort\_avgs)-1)){  
 found<-FALSE  
 #itime<-rbind(itime, i)  
 j<-1  
 while(found==FALSE){ #searching through ordIDS  
 test<-ordIDS[j, 1] == names(sort\_avgs)[i]  
 if(test) {   
 ids[i,2]<-ordIDS[j, 3]  
 found<-TRUE  
 }  
 else{  
 j <- j + 1  
 #count<-count+1  
 #jtime<-rbind(jtime, j)  
 }  
 }  
   
 }  
   
   
 names(ids)<-c("x", "name")  
 ids<-rbind(ids, data.frame(x=totalOTU+1, name= "Other"))  
   
   
 mavgs<-as.matrix(sort\_avgs)  
   
 leng<-dim(mavgs)[2]  
   
 numgr <- nlevels(file$expgroup)  
} ##end if(sortbyphyl == TRUE){  
  
  
####order the results by the first group alpha numerically  
if(sortbyphyl == FALSE){  
   
 row.names(avgs)<-avgs$expgroup  
 avgs<-avgs[,-1]  
 ordered\_avgs<- avgs[, rev(order(avgs[1,1:length(avgs)-1]))] #sort all but other column (last)  
 attach(avgs)  
 ordered\_avgs <- cbind(ordered\_avgs, Other) #put other back on  
 detach(avgs)  
   
 mavgs<-as.matrix(ordered\_avgs)  
 leng<-dim(mavgs)[2]  
   
 numgr <- nlevels(file$expgroup)  
   
   
 ids<-data.frame(1:(totalOTU))  
   
 for (i in 1:(length(ordered\_avgs)-1)){  
 found<-FALSE  
 #itime<-rbind(itime, i)  
 j<-1  
 while(found==FALSE){ #searching through ordIDS  
 test<-ordIDS[j, 1] == names(ordered\_avgs)[i]  
 if(test) {   
 ids[i,2]<-ordIDS[j, 3]  
 found<-TRUE  
 }  
 else{  
 j <- j + 1  
 #count<-count+1  
 #jtime<-rbind(jtime, j)  
 }  
 }  
   
 }  
   
   
 names(ids)<-c("x", "name")  
 ids<-rbind(ids, data.frame(x=totalOTU+1, name= "Other"))  
   
   
  
} #end if(sortbyphyl == FALSE){  
  
  
###started to do the stds table... didnt complete  
#stds<-stds[,-1]  
#mstds<-as.matrix(stds)  
  
  
###PLOT PARAMETERS  
par(mfrow=c(numgr+1, 1)) #+1 to give extra labeling space  
par(mar=c(0.3, 8, 0.5, 2) +0.1, mgp=c(4.5, 1, 0)) #default is 5.1 4.1 4.1 2.1, bot/left/top/right, also default mgp is c(3,1,0)  
  
  
  
for(j in 1:numgr){  
   
 if(j != numgr){  
 barplot(mavgs[j, 1:leng], ylab=NULL, col="black", yaxt="n", xaxt="n", ylim=c(0.0001, 1), log="y", cex.names=3)  
 #error.bar(bp[k], mavgs[j, 1:leng[2]], mstds[j, 1:leng[2]]) #the bp[k] was for storing the barplot locations to use for these errors  
 # k <- k+1  
 axis(2, las=1, at=c(.0001, .001, .01, .1, 1), labels=c(0, .001, .01, .1, 1), cex.axis=1.1)  
 mtext(abx[j], side=2, line=6, cex=.8)  
 mtext(avgCD[j], side=2, line=4.5, cex=.8)  
 abline(h=c(0.0001, 1), lwd=3)  
 abline(h=c(0.001, 0.01, 0.1), col="black", lty="longdash", lwd=1.5)  
 abline(h=c(.25,.5, .75), col="black", lty="dashed")  
 }  
   
 else{ #the last graph needs different margins   
 label<-barplot(mavgs[j, 1:leng], col="black",ylab=NULL, yaxt="n", xaxt="n", ylim=c(0.0001, 1), log="y", cex.names=3)  
 #error.bar(bp[k], mavgs[j, 1:leng[2]], mstds[j, 1:leng[2]]) #the bp[k] was for storing the barplot locations to use for these errors  
 axis(2, las=1, at=c(.0001, .001, .01, .1, 1), labels=c(0, .001, .01, .1, 1), cex.axis=1.1)  
 mtext(abx[j], side=2, line=6, cex=.8)  
 mtext(avgCD[j], side=2, line=4.5, cex=.8)  
 axis(1, at=(label[,1]), labels=FALSE)  
 text(label[,1]+.13, .00005, label=ids[,2], xpd=NA, pos=2, srt=45, cex=1.2)  
 #text(-3.9,.0001, label=expression(paste(italic("C.d."), " CFU/g Feces:")), xpd=NA, pos=2, srt=90, cex=1.1)  
 abline(h=c(0.0001,1), lwd=3)  
 abline(h=c(0.001, 0.01, 0.1), col="black", lty="longdash", lwd=1.5)  
 abline(h=c(.25,.5, .75), col="black", lty="dashed")  
   
 }   
  
}  
par(mfrow=c(1, 1))  
  
  
##############END CODE###########################  
#################################################

The graph can be sorted 2 ways either by the untreated's decreasing relabund, or by the most abundant within each phylum. For each treatment, the resulting *C. difficile* CFU/g Feces is also labeled on the Y axis below the treatment name. File saved as "topdose2\_tx1\_sorted\_10x20.pdf"

### TO DO STILL

* change the "0" to be min of 0.001 instead of 0.0001 relabund
  + **completed 11/20/14, just make sure future files are changed for other graphs using the "abxD01.barcharts.xOTU.test.r"**
* add the 1/2 dashed lines
  + **completed 11/20/14**
* also wanted to try to make this table in the other direction: Phyla/OTUs as each graph with different shadings or colors for each treatment... will be good for titration graphs

## Figure 2

This figure shows the correlation analysis of bacterial species present on Day 0 with *C. difficile* levels on Day 1.

### Figure 2 Correlation Calculations

Before we get to how this graph was made, we first used an R script to calculate the spearman correlation of OTUs' relative abundance.

The required file is located on Github.

meta <- read.table('~/Desktop/mothur/abxD01/correlation/abxD01.final.an.unique\_list.0.03.subsample.filter16mintotal.shared.correl.topdose2.txt',header=T)  
#meta<-meta[1:96] #change based on number of OTUs above .05%, then add 2 for first two cols  
c<-1  
otu <- c()  
cor.spear <- c()  
pval.spear <- c()  
cor.ken = c()  
pval.ken = c()  
for(i in 3:length(meta)){  
 otu[c] <- colnames(meta[i])  
 cor.spear[c] <- cor.test(meta[,2],meta[,i], method="spearman")$estimate  
 pval.spear[c] <- cor.test(meta[,2],meta[,i], method="spearman")$p.value  
 cor.ken[c] <- cor.test(meta[,2],meta[,i], method="kendall")$estimate #good to see because kendall handles ties  
 pval.ken[c] <- cor.test(meta[,2],meta[,i], method="kendall")$p.value #but only works if this is tao-b and not tao-a which im not sure about  
 c <- c+1  
}  
  
pval.spear<-p.adjust(pval.spear, method='BH') #adjust for multiple comparisons  
pval.ken<-p.adjust(pval.ken, method='BH') #adjust for multiple comparisons  
  
results = NULL  
results <- matrix(c(otu, cor.spear, pval.spear, cor.ken, pval.ken), ncol=5)  
colnames(results) <- c('otu','corSpear','pvalSpear', "corKen", "pvalKen")  
results <- results[order(results[,3]),] #order by pvalue column=3  
write.table(results[1:dim(results)[1],], file="~/Desktop/mothur/abxD01/correlation/abxD01.final.an.unique\_list.0.03.subsample.filter16mintotal.shared.correl.topdose2.results.txt", sep="\t", row.names=FALSE)

### Figure 2 Graph

This figure shows correlation values for OTUs with next day *C. difficile* CFU levels calculated across the original set of antibiotic experiments.

Necessary files are on github, and it was made using the following code:

##Paper Figure: correlations for topdose   
c<-read.csv("~/Desktop/mothur/abxD01/correlation/corr\_allSig\_topdose2\_stripchart.csv", header=T)  
labels=c("Lachnospiraceae", "Ruminococcaceae", "Clostridia", "Lactobacillales", "Firmicutes", "Bacillales", "Porphyromonadaceae", "Bacteroidales", "Bacteroidetes", "Actinobacteria", "Proteobacteria", "Anaeroplasma", "Deinococcus", "Unclassified")  
ns=c("n=18", "n=9", "n=8", "n=4", "n=2", "n=1", "n=15", "n=4", "n=1", "n=4", "n=3", "n=1", "n=1", "n=3")  
  
par(mar=c(12, 7, 2, 2) +0.1, mgp=c(5, 1, 0)) #default is 5.1 4.1 4.1 2.1 [bottom, left, top, right space], mgp=c(3, 1, 0) [label line location for x/y location labels, tick mark labels location, tick mark locations]  
stripchart(c$cor~c$order, vertical=TRUE, ylab="Correlation", ylim=c(.85, -.85), cex.axis=1.1, cex=2, pch=21, lwd=3, col="black", xaxt="n", method="jitter", cex.lab=1.7, yaxt="n")  
axis(1, cex.axis=1, at=(1:nlevels(c$name)), labels=FALSE)  
text(1:nlevels(c$name)+.1, par("usr")[3]-0.16, label=labels, xpd=NA, pos=2, srt=45, cex=1.2)  
text(1:nlevels(c$name), par("usr")[3]-0.01, label=ns, xpd=NA, pos=1, cex=.9)  
axis(2, cex.axis=1.1, at=seq(-0.85,+0.85, by=.1), las=1)  
abline(v=c(1:nlevels(c$name)), col="dark gray")  
#abline(h=c(.3, -.3), col="dark gray", lty="dashed", lwd=4) #change to something else  
abline(h=0, col="black", lwd=2)

Figure saved as "corr\_allSig\_topdose2\_10x8.pdf".

### TO DO STILL

* Do I want to change the groupings?

## Figure 3

This figure shows the effect of titrating antibiotics on the community as well as the resulting *C. difficile* levels.

The code for this is still in the works. But is uploaded to github under Figure 3 folder.

Input files on github: 1. abxD01.final.tx.2.subsample.allvanctitr.forlogscale.csv 2. abxD01/barcharts/allvanctitr\_tx2\_barchart\_ids.csv

### TO DO STILL

* I want to change the graph so that the OTUs are their own graphs and the titrations are side by side -\*\*11/20/14, completed most of the graphical parameters.\*
* once code is working then put up the other titration data files
* change min relabund from 0.0001 to 0.001 for cef and strep
  + **11/20/14, changed the vanc titrations min**
* *Possibly work on merging singles*
* *Possibly work on getting plots on top of grid lines*

## Figure 4

This figure is a heatmap showing the correlation analysis results for the original set of antibiotic treatments and doses with the titration data sets.

2 input files found in Figure 4 github folder: 1. correl\_heatmap\_0.01rel\_topdose2\_newtitr.csv 2. correl\_heatmapSIDE\_0.01rel\_topdose2\_newtitr.csv

library(gplots)  
correl<- read.csv("~/Desktop/mothur/abxD01/correlation/correl\_heatmap\_0.01rel\_topdose2\_newtitr.csv", header=T)  
  
row.names(correl)<-correl$OTU  
correl\_matrix<-data.matrix(correl)  
correl\_matrix<-correl\_matrix[,-1]  
correl\_matrix<-data.matrix(correl\_matrix)  
my\_palette <- colorRampPalette(c("blue", "white", "red"))(n = 299)  
breaks = c(seq(-1,-.33,length=100),seq(-.33,.33,length=100),seq(.33,1,length=100))#changed to equal numbers  
  
#par(mar=c(4, 6, 3, 4.5) +0.1) #default is 5.1 4.1 4.1 2.1  
  
side<-scan("~/Desktop/mothur/abxD01/correlation/correl\_heatmapSIDE\_0.01rel\_topdose2\_newtitr.csv", what=",", sep=",")  
side2<-read.csv("~/Desktop/mothur/abxD01/correlation/correl\_heatmapSIDE\_0.01rel\_topdose2\_newtitr.csv", header=T)  
side3<-as.matrix(side2)  
#lmat=rbind(c(4,3), c(2,1))  
#lmat=rbind(c(4,3), c(1,2))  
  
#lmat order: 1)row dendrogram, 2) heatmap, 3)col?, 4)key?--key disappeared  
lmat = rbind(3:4, 1:2)  
lmat = rbind(c(2,4), c(1,3))  
lmat = rbind(c(0,3), c(2,1))  
lmat=NULL  
lwid=NULL  
lhei=NULL  
lwid = c(.5, 2, 4)  
lhei = c(.7, 4)  
   
correl\_heatmap<-heatmap.2(correl\_matrix, scale="none", col=my\_palette, breaks=breaks, density.info="none", lhei=lhei,lwid=lwid, RowSideColors=side3[,2], labRow=side3[,1], trace="none",margins=c(8, 25), dendrogram="none", Rowv=FALSE, Colv=FALSE, na.color="black", cexCol=1.5, key=TRUE, keysize=1, cexRow=1) #then plot was 7x8in, portrait

Each row in this heatmap represents a single OTU. The side colors show broad taxonomic groups of each OTU. The subsequent columns show the correlation value for each OTU (Day 0) with Day 1 *C. difficile* colonization levels by plating. The first column represents the correlations calculations over all the original antibiotics used in the study. The last 3 columns represent the correlation calculations over the titration experiments for the given antibiotic. An example pdf of this figure is on github.

### TO DO STILL

* Add "ns" to the graph where not significant
* Is this the order of drugs I want-keep consistent
* for all the significant values for each pair of original + abx rows that are both significant, quantify that significance for each column (abx treatment)... see personal notes for more
* change xlabel names
* change code to get the key back

## Figure 5

This figure shows the difference between the Day 0's in the delayed and original treatments for metronidazole. The figure is shown as a pdf on github.

Input files found on github: 1. abxD01.final.tx.2.subsample.2.pick.metro.relabund.d0s.txt

#First calculating:  
b<-read.delim("~/Desktop/mothur/abxD01/barcharts/abxD01.final.tx.2.subsample.2.pick.metro.relabund.d0s.txt", header=T)  
  
avg1<-tapply(b$Otu001, b$order, mean)  
sd1<-tapply(b$Otu001, b$order, sd)  
  
avg11<-tapply(b$Otu011, b$order, mean)  
sd11<-tapply(b$Otu011, b$order, sd)  
  
avg02<-tapply(b$Otu002, b$order, mean)  
sd02<-tapply(b$Otu002, b$order, sd)  
  
avg03<-tapply(b$Otu003, b$order, mean)  
sd03<-tapply(b$Otu003, b$order, sd)  
  
avg7<-tapply(b$Otu007, b$order, mean)  
sd7<-tapply(b$Otu007, b$order, sd)  
  
avg05<-tapply(b$Otu005, b$order, mean)  
sd05<-tapply(b$Otu005, b$order, sd)  
  
avg16<-tapply(b$Otu016, b$order, mean)  
sd16<-tapply(b$Otu016, b$order, sd)  
  
avg06<-tapply(b$Otu006, b$order, mean)  
sd06<-tapply(b$Otu006, b$order, sd)  
  
avg04<-tapply(b$Otu004, b$order, mean)  
sd04<-tapply(b$Otu004, b$order, sd)  
  
#Now graphing:  
par(mfrow=c(4, 3))   
  
par(mar=c(2, 4, 2, 3) +0.1)  
plot1<-barplot(avg1, col="darkblue", cex.axis=1.3, xaxt="n")   
error.bar(plot1, avg1, sd1) #plot SD  
  
par(mar=c(2, 3, 2, 3) +0.1)  
plot03<-barplot(avg03, col="steelblue", cex.axis=1.3, xaxt="n")  
error.bar(plot03, avg03, sd03) #plot SD  
  
plot11<-barplot(avg11, col="darkgreen", cex.axis=1.3, xaxt="n")  
error.bar(plot11, avg11, sd11) #plot SD  
  
par(mar=c(2, 4, 2, 3) +0.1)  
plot02<-barplot(avg02, col="darkred", cex.axis=1.3, xaxt="n")  
error.bar(plot02, avg02, sd02) #plot SD  
  
par(mar=c(2, 3, 2, 3) +0.1)  
plot05<-barplot(avg05, col="orange2", cex.axis=1.3, xaxt="n")  
error.bar(plot05, avg05, sd05) #plot SD  
  
plot04<-barplot(avg04, col="gold1", cex.axis=1.3, xaxt="n")  
error.bar(plot04, avg04, sd04) #plot SD  
  
par(mar=c(2, 4, 2, 3) +0.1)  
plot16<-barplot(avg16, col="deeppink3", cex.axis=1.3, xaxt="n")  
error.bar(plot16, avg16, sd16) #plot SD  
axis(1, at=plot16, labels=c("D0", "D0 Recovered"), cex.axis=1.7)  
  
par(mar=c(2, 3, 2, 3) +0.1)  
plot7<-barplot(avg7, col="blueviolet", cex.axis=1.3, xaxt="n")  
error.bar(plot7, avg7, sd7) #plot SD  
axis(1, at=plot7, labels=c("D0", "D0 Recovered"), cex.axis=1.7)  
  
plot06<-barplot(avg06, col="black", cex.axis=1.3, xaxt="n")  
error.bar(plot06, avg06, sd06) #plot SD  
axis(1, at=plot06, labels=c("D0", "D0 Recovered"), cex.axis=1.7)  
  
#Now figure legend:  
plot(1, type="n", axes=F, xlab="", ylab="")  
  
leg<-c("Porphyromonadaceae", "Bacteroides","Erysipelotrichaceae" )  
col<-c("darkblue", "steelblue", "darkgreen" )  
#pch<-c(16, 16, 16)  
legend("left", legend=leg, pch=15, col=col, cex=1.7, bty="n", ncol=1)  
  
plot(1, type="n", axes=F, xlab="", ylab="")  
  
leg<-c("Lachnospiraceae", "Lactobacillus", "Ruminococcaceae" )  
col<-c("darkred", "orange2", "gold1" )  
#pch<-c(16, 16, 16)  
legend("left", legend=leg, pch=15, col=col, cex=1.7, bty="n", ncol=1)  
  
plot(1, type="n", axes=F, xlab="", ylab="")  
  
leg<-c("Bifidobacterium", "Enterobacteriaceae","Akkermansia" )  
col<-c("deeppink3", "blueviolet", "black" )  
#pch<-c(16, 16, 16)  
legend("left", legend=leg, pch=15, col=col, cex=1.7, bty="n", ncol=1)  
  
  
par(mfrow=c(1, 1))

### TO DO STILL

* Do for ampicillin too
* Pick the same groups showed in Figure 2. AND as side heatmap.

## Figure 6

This figure shows the results of our model predicting *C. difficile* colonization given relative abundances of a subset of bacteria.

### Figure 6: Model Building

In order to determine the best possible set of models, I first narrowed down the list of candidate OTUs to be considered in the model. This candidate list was created using several criteria, based on the data from the original set of antibiotic experiments alone:

1. Using top results of random forest's feature selection.
2. Incorporating both strongly positively correlated OTUs and strongly negatively correlated OTUs. We used the spearman correlation analysis for this, see Figure 2.
3. Using sparCC in mothur to determine any strong correlations (>90%) among the candidate list OTUs. We found the highest correlation between OTUs to be \_\_\_% This was not high enough to consider eliminating from the list.
4. Limiting the candidate list to 15 OTUs.

For the model, I limited models to include anywhere from 1 to 10 parameters. I log transformed the *C. difficile* CFU/g feces on Day 1 for inclusion in a linear model. The OTU data was in the form of a shared file.

The script for running this code is called "abxD01.lmModel.compare.R". The input files for this script are: 1. abxD01.final.an.unique\_list.0.03.subsample.filter16mintotal.shared.topdose2.logtrans.15otus.rfnegpos.csv

I realize I could've done the modelling comparison script in a loop and shortened the code length significantly, but deal with it.

### TO DO STILL

* Make figure! for modelling results
* The criteria for picking the otus, these results can be listed in supplementary tables possibly.
* Consider rerunning the model building code using a bigger candidate list, or using higher level taxonomy!!!
* Calculate the BIC for each--change code for this