Clustering UNC23 neo microbiome data

```
### v2.6 add PCA loading
### UNC21.   v2.3 add 2d and 3d plot of pc1 vs pc2 (vs pc3) using weight unifrac with color coded cluste


### UNC21. v2.2 remove the subjects that didn't pass inclusion criterion:
### Following up from our morning meet, the two microbiome cases that were recently discovered
###  to fail our inclusion criteria are MA035 and MA034. MA035 received gripe water before the neo visi
### MA034 was in the NICU for several hours after birth and was intubated.



### v1.1 use the precomputed unifrac and weighted unifrac
### v1.2 adds prediction strength from Dan Knight
### v1.3 use the L6 as the genus otu table
### v1.4 adds network analysis by creating Cytoscape file

#setwd('Z:/Kai/bios_brain/microbiome/')
#setwd('C:/Dropbox/scripts/microbiome')
#setwd('/Users/kaixia/Dropbox/scripts/microbiome')


if(Sys.info()["sysname"] == 'Windows'){
  dir1 = 'C:/Users/kxia/OneDrive - University of North Carolina at Chapel Hill/github/gmia/scripts'
} else{
  dir1 = '/Users/kaixia/OneDrive - University of North Carolina at Chapel Hill/github/gmia/scripts'
}

setwd(dir1)



library("phyloseq")
library("clusterSim")
```

```
## Loading required package: cluster
```

```
## Loading required package: MASS
```

```
##
## This is package 'modeest' written by P. PONCET.
## For a complete list of functions, use 'library(help = "modeest")' or 'help.start()'.
```

```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
source('biome.R')
source('prediction_strength.R')

library(scatterplot3d)
library(rgl)
library(fpc)
```

```
##
## Attaching package: 'fpc'
## The following object is masked _by_ '.GlobalEnv':
##
##     prediction.strength
library(vegan)
## Warning: package 'vegan' was built under R version 3.5.2
## Loading required package: permute
## Warning: package 'permute' was built under R version 3.5.2
## Loading required package: lattice
## This is vegan 2.5-3
dir_mb = '../processed_data'
dir_raw = '../raw_data'
dir_cvrt = sprintf('%s/cvrt', dir_raw)
dir_brain = sprintf('%s/braindata', dir_raw)
dir_out = '../results'
dir_fig = '../fig'

dataset = 'neo'

dir_div = sprintf('%s/diversity/%s', dir_raw, dataset)
#dir_beta = sprintf('%s/diversity/%s/core_div/bdiv_even5000', dir_raw, dataset)


#dir1 = 'Z:/Kai/bios_brain/microbiome/2018_0502/160527_UNC23_0034_000000000-ANV4F'

dir_otu = sprintf('%s/core_div/taxa_plots',dir_div)
dir_beta = sprintf('%s/core_div/bdiv_even5000',dir_div)
#dir_tax = sprintf('%s/taxa_summary',dir1)
#dataset = 'UNC23'


### try phyloseq, not working at all
#file_biome1 = sprintf("%s/otu_table_no_singletons_no_chimeras.biom",'Z:/Kai/bios_brain/microbiome/2018
file_biome = sprintf("%s/table_mc5000_sorted_L6.txt",dir_otu)

#file_tree = sprintf("%s/rep_set.tre",dir_otu)
# data_biom = system.file("extdata", sprintf("%s/otu_table_no_singletons_no_chimeras.biom",dir_otu), pa
# treefilename = system.file("extdata", sprintf("%s/rep_set.tre",dir_otu),  package="phyloseq")
#biomot = import_biom(file_biome, file_tree, parseFunction=parse_taxonomy_greengenes)
#biomot = import_biom(file_biome, file_tree, parseFunction = parse_taxonomy_qiime)
#data_biom0 = import_biom(file_biome, file_tree)
#data_biom1 = import_biom(file_biome)

data_biom0 = read.delim(file_biome,skip=1,strings=FALSE)
Names = colnames(data_biom0)[-1]
data_matrix = data_biom0[,-1]
rownames(data_matrix) = data_biom0[,1]
taxNames = data_biom0[,1]
```

```
taxNames = as.character((taxNames))
list0 = strsplit(taxNames,';')
taxmat = matrix(unlist(list0),ncol=6,byrow=TRUE)

#data_biom

otumat= otu_table(data_matrix,taxa_are_rows=TRUE)


#taxmat = matrix(sample(letters, 70, replace = TRUE), nrow = nrow(otumat), ncol = 7)
rownames(taxmat) <- rownames(otumat)
colnames(taxmat) <- c("Domain", "Phylum", "Class", "Order", "Family", "Genus")

OTU = otu_table(otumat, taxa_are_rows = TRUE)
TAX = tax_table(taxmat)
data_biom = phyloseq(OTU,TAX)

### remove 'MA021.neo' since it is removed requested Alex

idx_rm  = c('MA021.NEO')
Names = sample_names(data_biom)

Names = Names[!Names %in% idx_rm]
#Names = Names[!Names %in% id_rm]

data_biom = prune_samples(Names,data_biom)



data_matrix = otu_table(data_biom)

dist_methods = c('unifrac','wunifrac','jsd','rjsd','bray')
dist_methods_Names = c('UniFrac','Weighted UniFrac','JSD','rJSD','Bray-Curtis')

#dist_methods = c('jsd','rjsd','bray')
#dist_methods_Names = c('JSD','rJSD','Bray-Curtis')

### unifrac
dist_wunifrac = read.table(sprintf('%s/weighted_unifrac_dm.txt',dir_beta),header=TRUE,sep='\t')
dist_wunifrac = dist_wunifrac[,-1]
colnames(dist_wunifrac) = gsub('^X','',colnames(dist_wunifrac))
rownames(dist_wunifrac) = colnames(dist_wunifrac)

dist_wunifrac = dist_wunifrac[rownames(dist_wunifrac) %in% Names, colnames(dist_wunifrac) %in% Names]

## unweighted unifrac
dist_unifrac = read.table(sprintf('%s/unweighted_unifrac_dm.txt',dir_beta),header=TRUE,sep='\t')
dist_unifrac = dist_unifrac[,-1]
colnames(dist_unifrac) = gsub('^X','',colnames(dist_unifrac))
rownames(dist_unifrac) = colnames(dist_unifrac)

dist_unifrac = dist_unifrac[rownames(dist_unifrac) %in% Names, colnames(dist_unifrac) %in% Names]
```

```
# source('prediction_strength.R');  prediction.strength(dist_unifrac)



n_cluster = 8


#nclusters = matrix(NA,n_cluster,length(dist_methods))
#nclusters = NULL
list_cluster = list()
list_cluster_2c = list()
nclusters = list()
score_methods = c('Average silhouette width','Calinski-Harabasz score','Prediction Strength')
score_Names = c('SH','CH','PS')
for(j in 1:length(score_Names)){
    mx1 = matrix(NA,n_cluster,length(dist_methods))
    colnames(mx1) = dist_methods
    nclusters[[score_Names[j]]] = mx1
}

for(i in 1:length(dist_methods)){
    if(dist_methods[i] == 'rjsd'){
        data_dist = dist.JSD(data_matrix)
    } else if(dist_methods[i] == 'unifrac') {
        data_dist = dist_unifrac
    } else if(dist_methods[i] == 'wunifrac'){
        data_dist = dist_wunifrac
    } else if(dist_methods[i] == 'bray'){
      data_dist = vegdist(t(data_matrix),method='bray')
    } else{
        data_dist = distance(data_biom,method = dist_methods[i])
    }
    for (k in 1:n_cluster) {
        if (k==1) {
            nclusters[['CH']][k,i] = NA
            nclusters[['SH']][k,i] = NA
        } else {
            data_cluster_temp=pam.clustering(data_dist, k)
            if(k == 3){
                list_cluster[[i]] = data_cluster_temp
            }
            if(k == 2){
                list_cluster_2c[[i]] = data_cluster_temp
            }
            val_CH=index.G1(t(data_matrix),data_cluster_temp,  d = data_dist, centrotypes = "medoids")
            nclusters[['CH']][k,i] = val_CH

            val_SH=mean(silhouette(data_cluster_temp, data_dist)[,3])
            # PS =  prediction.strength(data_matrix,k,k,M=3)
            # val_PS =PS$mean.pred[k]
            nclusters[['SH']][k,i] = val_SH
        }
    }
```

```
    ### add prediction strength
    val_PS = prediction.strength(data_dist, K=2:n_cluster, M=10)
    val_PS = as.numeric(val_PS)
    nclusters[['PS']][,i] = c(NA,val_PS)
}

#### output the 3 cluster results
mx_out = data.frame(id = Names)
for(i in 1:length(list_cluster)){
    mx_out = cbind(mx_out,list_cluster[[i]])
}
colnames(mx_out) = c('ID',dist_methods)
write.table(mx_out,file=sprintf('%s/cluster_results_c3_%s.txt',dir_out,dataset),sep='\t',quote=FALSE,co

#### output the 2 cluster results
mx_out = data.frame(id = Names)
for(i in 1:length(list_cluster_2c)){
    mx_out = cbind(mx_out,list_cluster_2c[[i]])
}
colnames(mx_out) = c('ID',dist_methods)
write.table(mx_out,file=sprintf('%s/cluster_results_c2_%s.txt',dir_out,dataset),sep='\t',quote=FALSE,co



### load the OTUs
# data1=read.table(sprintf('%s/otu_table_no_singletons_no_chimeras.txt',dir_otu), header=TRUE, skip=1,
# rownames(data1)=data1[,1]
# Ncol = dim(data1)
# data1 = data1[,-c(1,Ncol-1,Ncol)]


pdf(sprintf('%s/cluster_eval_%s.pdf',dir_fig,dataset),paper='special',width=8,height=8)
#layout(matrix(1:4,2,2,byrow=TRUE))
pchs = c(0:4)
for(j in 1:length(score_Names)){
    #ylims = c(0,0.3)
    if(score_Names[j] == 'CH'){
        #ylims = c(-20,25)
        ylims = c(-35,35)
    } else if(score_Names[j] == 'SH'){
            ylims = c(0,0.6)
    } else if(score_Names[j] == 'PS'){
        ylims = c(0,1)
    }
    for(i in 1:length(dist_methods)){
        if(i == 1){
            plot(1:n_cluster,nclusters[[score_Names[j]]][,i], type="b",pch=pchs[i], xlab="k clusters", y
        } else{
            points(1:n_cluster,nclusters[[score_Names[j]]][,i],pch=pchs[i],type='b')
        }
    }
    legend('topright',legend=dist_methods_Names,pch=pchs)
}
```

```
#dev.off()


### after the evalution, I choose to use three clusters to get the plot:
library(ade4)
```

```
## Warning: package 'ade4' was built under R version 3.5.2
```

```
sub_size = 2
k=3
data_dist = distance(data_biom,method = 'jsd')
data_cluster = pam.clustering(data_dist,k)

 obs.pca=dudi.pca(data.frame(t(data_matrix)), scannf=F, nf=10)
 obs.bet=bca(obs.pca, fac=as.factor(data_cluster), scannf=F, nf=k-1)
 s.class(obs.bet$ls, fac=as.factor(data_cluster), grid=F,sub='JSD',csub=sub_size)
 #text(obs.bet$ls,labels=colnames(data_biom),adj=c(-.6,-2),cex=0.3)

 ## rjsd
 # k=3
# data_dist = dist.JSD(data_matrix)
# data_cluster = pam.clustering(data_dist,k)

 # obs.pca=dudi.pca(data.frame(t(data_matrix)), scannf=F, nf=10)
 # obs.bet=bca(obs.pca, fac=as.factor(data_cluster), scannf=F, nf=k-1)
 # s.class(obs.bet$ls, fac=as.factor(data_cluster), grid=F,sub='JSD',csub=sub_size)


 k=3
data_dist = dist_wunifrac
data_cluster = pam.clustering(data_dist,k)

 obs.pca=dudi.pca(data.frame(t(data_matrix)), scannf=F, nf=10)
 obs.bet=bca(obs.pca, fac=as.factor(data_cluster), scannf=F, nf=k-1)
 s.class(obs.bet$ls, fac=as.factor(data_cluster), grid=F,sub='wunifrac',csub=sub_size)


 k=3
data_dist = dist_unifrac
data_cluster = pam.clustering(data_dist,k)

 obs.pca=dudi.pca(data.frame(t(data_matrix)), scannf=F, nf=10)
 obs.bet=bca(obs.pca, fac=as.factor(data_cluster), scannf=F, nf=k-1)
 s.class(obs.bet$ls, fac=as.factor(data_cluster), grid=F,sub='unifrac',csub=sub_size)
#  k=3
# data_dist = distance(data_biom,method = 'bray')
# data_cluster = pam.clustering(data_dist,k)
#
# data_denoized=noise.removal(data_matrix, percent=0.01)
#
#  obs.pca=dudi.pca(data.frame(t(data_matrix)), scannf=F, nf=10)
#  obs.bet=bca(obs.pca, fac=as.factor(data_cluster), scannf=F, nf=k-1)
#  s.class(obs.bet$ls, fac=as.factor(data_cluster), grid=F,sub='Bray-Curtis',csub=sub_size)
```

```
#dev.off()

#########################################################################################################################
#### plot the pc vs clustering membership

#### plot the 2d cluster with weighted unifrac
file_cluster = sprintf('%s/cluster_results_c2_%s.txt',dir_out,dataset)
data1_cluster = read.delim(file_cluster,header=TRUE,strings=FALSE)

clusterNames = colnames(data1_cluster)[c(2,3,4,6)]

### weighted unifrac
file_pc = sprintf('%s/weighted_unifrac_pc_mod.txt',dir_beta)
mx_pc = read.delim(file_pc,header=FALSE,strings=FALSE)


mch1 = match(data1_cluster[,1], mx_pc[,1])
pc1 = mx_pc[mch1,2]
pc2 = mx_pc[mch1,3]
pc3 = mx_pc[mch1,4]

layout(matrix(1:4,2,2,byrow=TRUE))
colors = c('red','blue','green')
pchs = c(1,2,3)
for(clusterName in clusterNames){
    cluster_id = data1_cluster[,clusterName]
    #cols = colors[cluster_id]
    plot(pc1,pc2,xlab='PC 1',ylab='PC 2',main=clusterName,col=colors[cluster_id],pch=pchs[cluster_id])
    legend('topleft',legend = c('wunifrac PC 1','wunifrac PC 2'), col =  colors[1:2], pch = pchs[1:2])
}

#### plot 3d cluster with weighted unifrac
file_cluster = sprintf('%s/cluster_results_c3_%s.txt',dir_out,dataset)
data1_cluster = read.delim(file_cluster,header=TRUE,strings=FALSE)

layout(matrix(1:4,2,2,byrow=TRUE))
for(clusterName in clusterNames){
    cluster_id = data1_cluster[,clusterName]

    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    legend('topleft',legend = c('wunifrac PC 1','wunifrac PC 2','wunifrac PC 3'), col =  colors, pch = p

    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
}

### unweighted unifrac
file_pc = sprintf('%s/unweighted_unifrac_pc_mod.txt',dir_beta)
mx_pc = read.delim(file_pc,header=FALSE,strings=FALSE)

file_cluster = sprintf('%s/cluster_results_c2_%s.txt',dir_out,dataset)
data1_cluster = read.delim(file_cluster,header=TRUE,strings=FALSE)
```

```
clusterNames = colnames(data1_cluster)[c(2,3,4,6)]


mch1 = match(data1_cluster[,1], mx_pc[,1])
pc1 = mx_pc[mch1,2]
pc2 = mx_pc[mch1,3]
pc3 = mx_pc[mch1,4]

layout(matrix(1:4,2,2,byrow=TRUE))
colors = c('red','blue','green')
pchs = c(1,2,3)
for(clusterName in clusterNames){
    cluster_id = data1_cluster[,clusterName]
    #cols = colors[cluster_id]
    plot(pc1,pc2,xlab='PC 1',ylab='PC 2',main=clusterName,col=colors[cluster_id],pch=pchs[cluster_id])
    legend('topleft',legend = c('unifrac PC 1','unifrac PC 2'), col =  colors[1:2], pch = pchs[1:2])
}



#### plot 3d cluster with unweighted unifrac
file_cluster = sprintf('%s/cluster_results_c3_%s.txt',dir_out,dataset)
data1_cluster = read.delim(file_cluster,header=TRUE,strings=FALSE)

layout(matrix(1:4,2,2,byrow=TRUE))
for(clusterName in clusterNames){
    cluster_id = data1_cluster[,clusterName]

    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    legend('topleft',legend = c('unifrac PC 1','unifrac PC 2','unifrac PC 3'), col =  colors, pch = pchs

    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
    scatterplot3d(x = pc1, y = pc2, z = pc3, color = colors[cluster_id], pch=pchs[cluster_id], xlab='PC
}

dev.off()
```

```
## pdf
##   2
```

```
#########################################################################################################
#### get the genus tables
### change the name of the data_matrix table to genus name
rowNames = rownames(data_matrix)
#gsub('\\*\\_','',rowNames)
split1 = strsplit(rowNames,'__')
genusNames = rowNames
for(i in 1:length(split1)){
    Names = split1[[i]]
    genusNames[i] = Names[length(Names)]
}
```

```
k=3

for(i in 1:length(dist_methods)){
    if(dist_methods[i] == 'rjsd'){
        data_dist = dist.JSD(data_matrix)
    } else if(dist_methods[i] == 'unifrac') {
        data_dist = dist_unifrac
    } else if(dist_methods[i] == 'wunifrac'){
        data_dist = dist_wunifrac
    } else if(dist_methods[i] == 'bray'){
      data_dist = vegdist(t(data_matrix),method='bray')
    }
  else{
        data_dist = distance(data_biom,method = dist_methods[i])
    }
#data_dist = distance(data_biom,method = method)
    data_cluster = pam.clustering(data_dist,k)

    pdf(sprintf('%s/boxplot_abundance_%s_%s_c%s.pdf',dir_fig,dataset,dist_methods[i],k),paper='special'
    layout(matrix(1:16,4,4,byrow=TRUE))

    for(i in 1:dim(data_matrix)[1]){
        if(genusNames[i] == ';g'){next}
        if(max(data_matrix[i,]) < 0.01) {next}
        if(i == dim(data_matrix)[1]){next}
        plot(as.factor(data_cluster),as.numeric(data_matrix[i,]),main=genusNames[i],xlab='Enteroptype')

    }
    dev.off()
}
```

```
### weighted unifrac
file_pc = sprintf('%s/weighted_unifrac_pc_mod.txt',dir_beta)
mx_pc = read.delim(file_pc,header=FALSE,strings=FALSE)

pdf(sprintf('%s/PCA_loading_%s.pdf',dir_fig, dataset),width=4*3,height=4*5,paper='special')
layout(matrix(1:2,1,2,byrow=TRUE))
mch1 = match(colnames(data_matrix), mx_pc[,1])
#pc1 = mx_pc[mch1,2]
#pc2 = mx_pc[mch1,3]
pcs = mx_pc[mch1,2:3]
cor_mx=cor(pcs,t(data_matrix))

for(i in 1:dim(pcs)[2]){
  barplot(cor_mx[i,],main=paste('PC',i,'loading'),horiz=TRUE,names.arg=genusNames,las=2,cex.names = 0.7
}
dev.off()
```

```
## pdf
##   2
```

```
################################################################################
### create network
## enterotype 1: Faecalibacterium
## enterotype 2: Veillonella
```

```
## enterotype 3: Blautia or [Eubacterium]

get_simple_genus_name = function(rowNames){
    split1 = strsplit(rowNames,'__')
    Names1 = rowNames
    for(ii in 1:length(split1)){
        Names = split1[[ii]]
        name_tmp = Names[length(Names)]
        if(name_tmp == ";g"){
            print(name_tmp)
            name_tmp = paste(Names[length(Names)-1],'_',Names[length(Names)],sep='')
        }
        Names1[ii] = name_tmp
    }
    return(Names1)
}

### change the name of the data_matrix table to genus name
rowNames = rownames(data_matrix)
#gsub('\\*\\_','',rowNames)
split1 = strsplit(rowNames,'__')
genusNames = rowNames
for(i in 1:length(split1)){
    Names = split1[[i]]
    genusNames[i] = Names[length(Names)]
}

genus = c('k__Bacteria;p__Bacteroidetes;c__Bacteroidia;o__Bacteroidales;f__Bacteroidaceae;g__Bacteroides
            ,'k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae;g__'
            ,'k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Veillonellaceae;g__Veillonella
            ,'k__Bacteria;p__Actinobacteria;c__Actinobacteria;o__Bifidobacteriales;f__Bifidobacteriacea
            ,'k__Bacteria;p__Proteobacteria;c__Gammaproteobacteria;o__Enterobacteriales;f__Enterobacter
            ,'k__Bacteria;p__Firmicutes;c__Bacilli;o__Lactobacillales;f__Enterococcaceae;g__Enterococcus
genusNames1 = c('Bacteroides','Lachnospiraceae','Veillonella','Bifidobacterium','Enterobacteriaceae','E

### cal correlation matrix
#mx1 = data_matrix
#rownames(mx1) = rowNames
cutoff = 0.4
t_matrix = t(data_matrix)
cor1 = cor(t_matrix,method='spearman')
#colnames(cor1) = genusNames
#rownames(cor1) = genusNames
n_cluster = 2

genusNames = rownames(data_matrix)

for( i in 1:length(genus)){
    # i = 1;
    fileout = sprintf('cor_network_%s_%s_%s.sif',dataset,n_cluster,i)
    fout = file(fileout,open='w')

    name1 = genus[i]
```

```
    print(name1)
    idx = which(genusNames == name1)
    corNames1 = genusNames[which(cor1[idx,] > cutoff)]
    corNames2 = genusNames[which(cor1[idx,] < -cutoff)]
    idx_all = genusNames %in% c(corNames1,corNames2)
    mx2 = cor1[idx_all,idx_all]
    print(dim(mx2))
    # print(name1)
    # print('pos')
    # print(corNames1)
    # print('anti')
    # print(corNames2)
    # print('\n')

    rowNames = rownames(mx2)
    #gsub('\\*\\_','',rowNames)
    split1 = strsplit(rowNames,'__')
    Names1 = rowNames
    for(ii in 1:length(split1)){
        Names = split1[[ii]]
        name_tmp = Names[length(Names)]
        if(name_tmp == ";g"){
            print(name_tmp)
            name_tmp = paste(Names[length(Names)-1],'_',Names[length(Names)],sep='')
        }
        Names1[ii] = name_tmp
    }


    #print(Names1)
    #Names1 = colnames(mx2)
    for(j in 1:dim(mx2)[1]){
        for(k in 1:dim(mx2)[1]){
            if(j <= k){next}
            if(abs(mx2[j,k]) < cutoff){next}

            if(mx2[j,k] > 0){
                inter_type = 'pos'
            } else{
                inter_type = 'neg'
            }
            line1 = paste(Names1[j],'\t',inter_type,'\t',Names1[k],sep='')
            print(line1)
            writeLines(line1,fout,sep='\n')

        }
    }

    ### get the node attribute output
    # fileout1 = sprintf('cor_network_%s_%s_%s.noa',dataset,n_cluster,i)
    # fout1 = file(fileout1,open='w')

    # corNames1 = get_simple_genus_name(corNames1)
```

```
    # writeLines("NodeCorrelation",fout1,sep='\n')
    # writeLines(paste(get_simple_genus_name(name1),' = seed',sep=''),fout1,sep='\n')
    # for(Name1 in corNames1){
        # line1 = paste(Name1,' = corr',sep='')
        # writeLines(line1,fout1,sep='\n')
    # }
    # corNames2 = get_simple_genus_name(corNames2)
    # #writeLines("NodeCorrelation",fout1,sep='\n')
    # for(Name2 in corNames2){
        # line1 = paste(Name2,' = anti',sep='')
        # writeLines(line1,fout1,sep='\n')
    # }
    # close(fout1)

    # if(i == 2){
        # break
    # }
    close(fout)
}









#
#
# ##############################################################################
# ### run some testing
#
#
# ### v1.1 use the precomputed unifrac and weighted unifrac
# ### v1.2 adds prediction strength from Dan Knight
# ### v1.3 use the L6 as the genus otu table
# ### v1.4 adds network analysis by creating Cytoscape file
#
# setwd('C:/Dropbox/scripts/microbiome')
#
# library("phyloseq")
# library("clusterSim")
# library("ggplot2")
# source('biome.R')
# source('prediction_strength.R')
#
# library(fpc)
```

```
#
# #dir1 = 'raw_data'
# dir_otu = 'raw_data/otus'
# dir_beta = 'raw_data/beta_diversity'
# dir_tax = 'raw_data/taxa_summary'
#
#
# ### try phyloseq, not working at all
# #file_biome = sprintf("%s/otu_table_no_singletons_no_chimeras.biom",dir_otu)
# file_biome = sprintf("%s/otu_table_no_singletons_no_chimeras_L6.biom",dir_tax)
#
# file_tree = sprintf("%s/rep_set.tre",dir_otu)
# # data_biom = system.file("extdata", sprintf("%s/otu_table_no_singletons_no_chimeras.biom",dir_otu),
# # treefilename = system.file("extdata", sprintf("%s/rep_set.tre",dir_otu),  package="phyloseq")
# #biomot = import_biom(file_biome, file_tree, parseFunction=parse_taxonomy_greengenes)
# #biomot = import_biom(file_biome, file_tree, parseFunction = parse_taxonomy_qiime)
# #data_biom0 = import_biom(file_biome, file_tree)
# data_biom0 = import_biom(file_biome)
#
# Names = sample_names(data_biom0)
# Names = Names[Names != 'Blank']
# data_biom = prune_samples(Names,data_biom0)
#
# data_matrix = otu_table(data_biom)
# pdf(sprintf('%s/hist_genera_all_%s.pdf',dir_fig, dataset),width=4*3,height=4*4,paper='special')
# layout(matrix(1:12,4,3,byrow=TRUE))
# for(i in 1:dim(data_matrix)[1]){
#   hist(as.numeric(data_matrix[i,]),breaks=100)
# #hist(as.numeric(data_matrix[2,]),breaks=100)
# }
# dev.off()
#
#
#
# # #### nature paper
# # data1 = read.delim('MetaHIT_SangerSamples.genus.txt',header=TRUE)
# # data_matrix = data1[-1,]
# # pdf('fig/hist_genera_nature.pdf',width=4*3,height=4*4,paper='special')
# # layout(matrix(1:12,4,3,byrow=TRUE))
# # for(i in 1:dim(data_matrix)[1]){
# #     hist(as.numeric(data_matrix[i,]),breaks=100)
# # #hist(as.numeric(data_matrix[2,]),breaks=100)
# # }
# # dev.off()
#
#
```