Taxonomic Analysis

Niels W. Hanson
Bioinformatics Ph.D. Candidate
Tuesday, February 11 2014

Hallam Laboratory
Hydrocarbon MetaPathways Workshop
The University of British Columbia, Vancouver

https://github.com/nielshanson/mp_tutorial/wiki/Taxonomic-Analysis

Prerequisites

- Install MEGAN - http://ab.inf.uni-tuebingen.de/software/megan5/
- Install R Studio - http://www.rstudio.com/

Downloads

- Presentation Slides - MetaPathways_Tutorial_Taxonomic_Analysis.pdf
- Complete R Script - mp_tutorial_taxonomic_analysis.R
- Megan-Compatible Files - HOT_Sanger_megan.csv.zip
- Megan-Table File - HOT_megan_table.txt

Tutorial Goals

1. Understand the basic background of the taxonomic signal related to the 16S rRNA, Clusters of Orthologous Genes (COGs), and MEGAN taxonomy.
2. Be able to prime taxonomic search results from MetaPathways for analysis in MEGAN via python scripts
3. Understand how to decline taxonomy in MEGAN and its ability to export samples
4. Outline some examples of downstream hierarchical clustering in the R statistical environment

1. Overview of Taxonomic Annotation

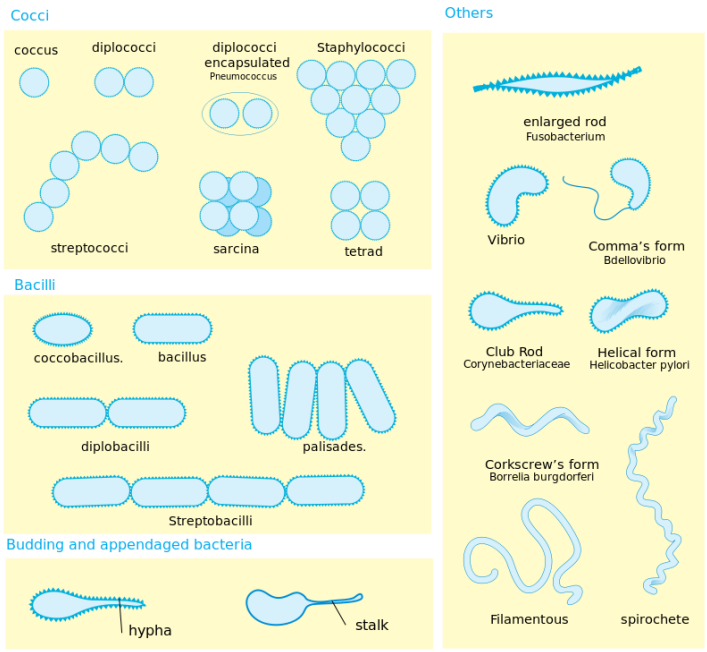
i. Ribosomal rRNA Genes

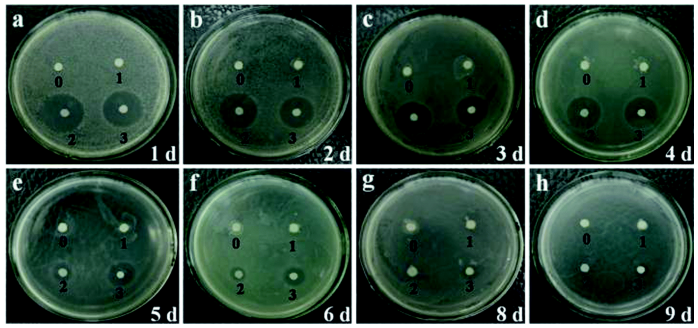
ii. Clusters of Orthologous Genes (COGs)

iii. RefSeq Functional Taxonomy
(LCA or MEGAN-taxonomy)

Ribosomal rRNA Genes (e.g., 16S, 18S)

- Differences in morphology & growth of microbes insufficient to determine phylogeny
  — thought high order organisms has most diversity (Five Kingdoms Model)
- Enter DNA sequencing: certain genes taxonomically relevant rRNA genes (widely found, stable sequence)
- Alignment of hyper variable regions of 16S lead to development of modern ‘tree-of-life

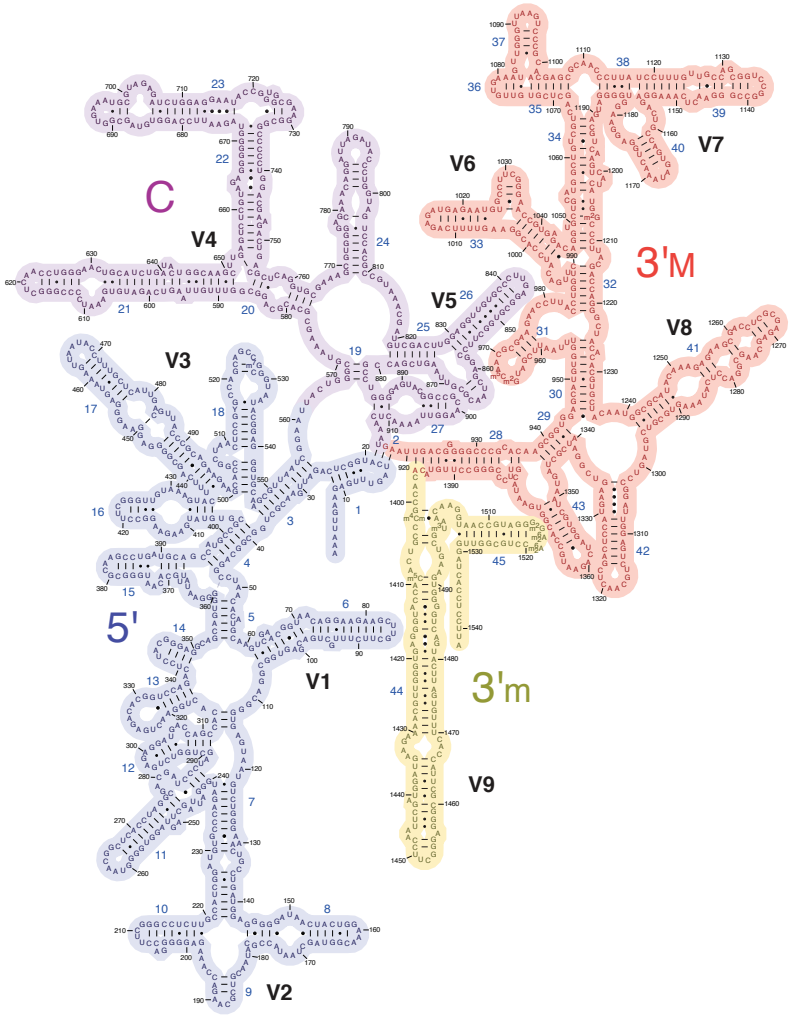




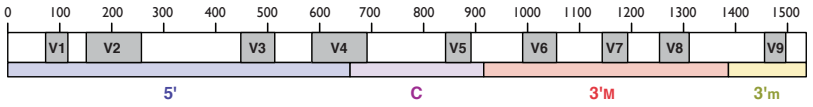
Ribosomal rRNA Genes (e.g., 16S, 18S)

Ribosomal rRNA

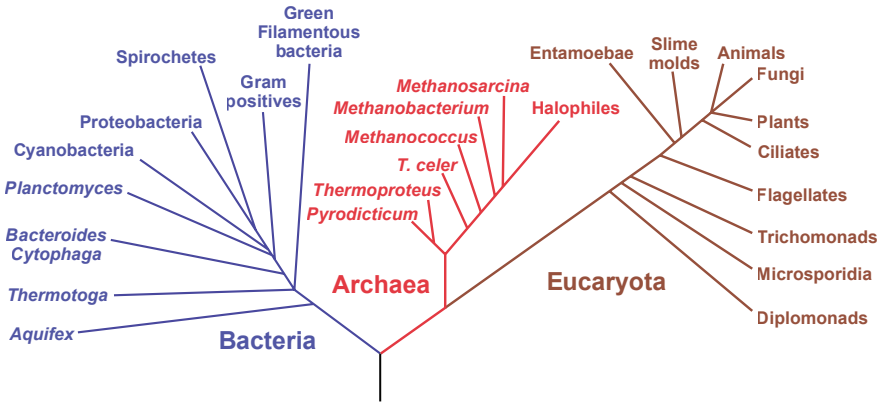
a



Alignment



"Tree-of-Life"



Ribosomal rRNA Genes (e.g., 16S, 18S)

- small (16S and 18S) and large (23S and 28S) rRNA subunits catalogued in databases (Silva[1], GreenGenes[2])
- Analysis

  i. 16S Amplified by PCR (a.k.a., pyrotags)

  - PCR Biases within & differences in 'universal' primers makes relative abundance of taxa "semi-quantitative" at best[3]

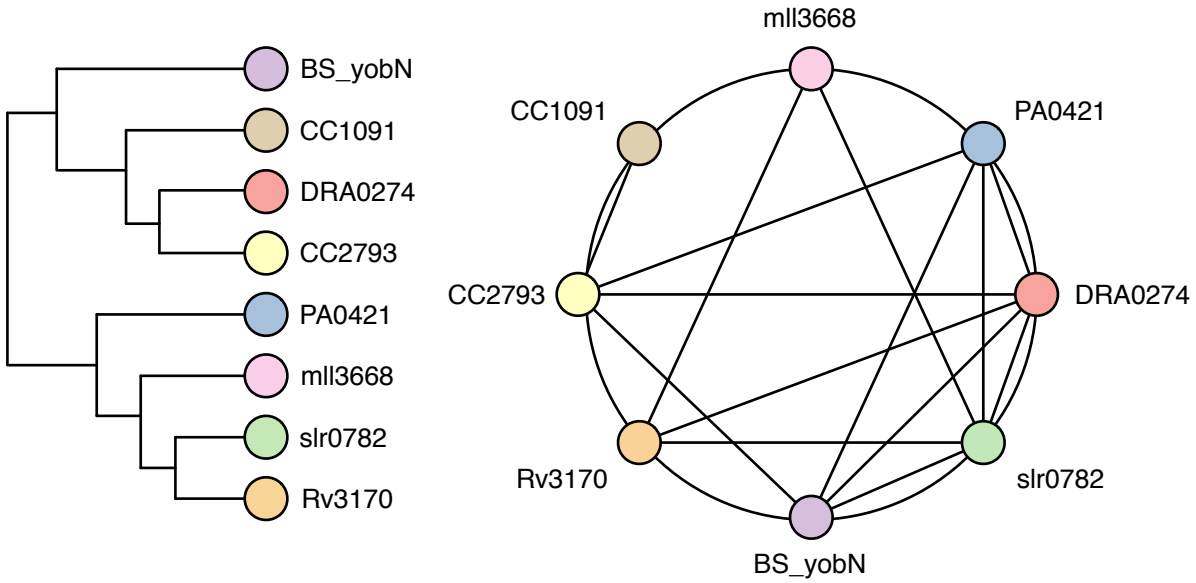
  ii. Metagenomes

  - Not sequencing specifically - taxonomic breadth limited
- - Possibly more quantitative – no 'universal' primer bias

1. E. Pruesse et al., SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB, Nucleic Acids Research 35, 7188–7196 (2007).
2. T. Z. DeSantis et al., Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB, Appl. Environ. Microbiol. 72, 5069–5072 (2006).
3. J. G. Caporaso *et al.*, QIIME allows analysis of high-throughput community sequencing data, *Nat Meth* **7**, 335–336 (2010).

Clusters of Orthologous Genes

- What's so special about the 16S rRNA gene? COG a collection of single copy genes for taxonomic signal[1]



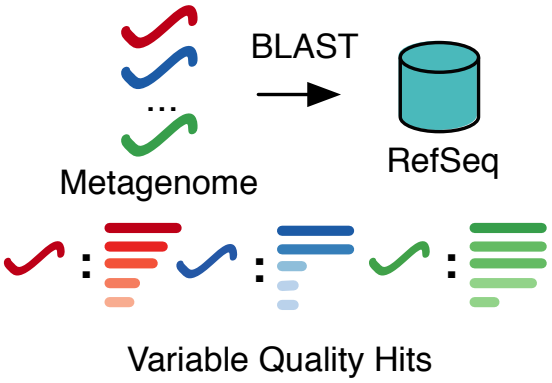
- Triangle Homology Rule

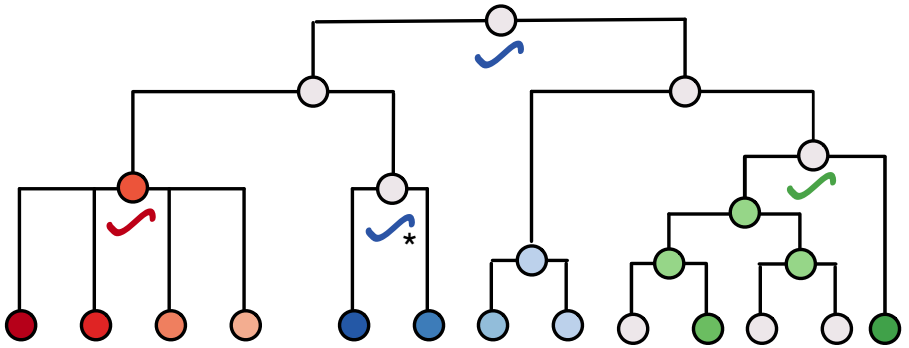
  "Any group of at least three proteins from distant genomes that are more similar to each other than they are to any other proteins are likely to form an orthologous set."

1. R. L. Tatusov *et al.*, The COG database: new developments in phylogenetic classification of proteins from complete genomes, *Nucleic Acids Research* **29**, 22–28 (2001).

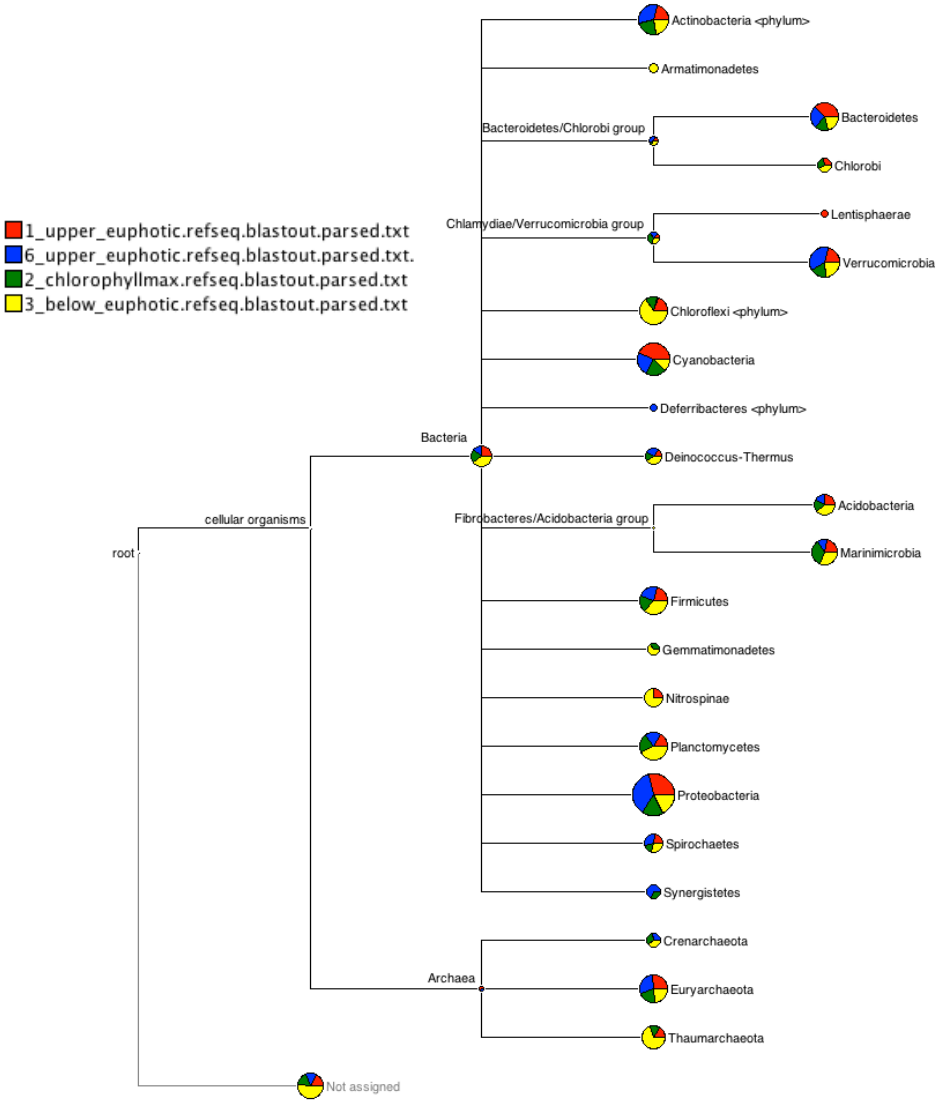
RefSeq Functional Taxonomy Metagenome Analyzer (MEGAN)

- Genes identified in RefSeq have underlying genome of origin.
  - High quality hits possibly contain a (noisy) taxonomic signal
- Megan an interactive software to implement the Lowest Common Ancestor (LCA) algorithm





RefSeq Functional Taxonomy Metagenome Analyzer (MEGAN)



2. Priming MetaPathways output for MEGAN

- Take rRNA (Silva, GreenGenes), RefSeq, and COG hits to MEGAN
- MEGAN can easily parse comma-separated files (.csv) files in the following format:
  <read>, <taxa>, <score>

```
1_upper_euphotic_5292_1, Caulobacter vibrioides, 813
1_upper_euphotic_5292_1, Brucella melitensis, 786
1_upper_euphotic_5292_1, Pseudomonas aeruginosa, 774
1_upper_euphotic_5296_0, Mycobacterium tuberculosis CDC1551, 757
1_upper_euphotic_5296_0, Mycobacterium tuberculosis H37Rv, 757
1_upper_euphotic_5292_1, Salmonella typhimurium LT2, 743
```

Python Scripts: metapathways_[rRNA/last/COG]_to_megan.py

metapathways_rRNA_to_megan.py

```
$ python metapathways_rRNA_to_megan.py -i *.rRNA.stats.txt
                                       -o output_folder .
                                       [--dsv]
```

```
$ python metapathways_rRNA_to_megan.py -i HOT_Sanger/*/results/rRNA/
*ssu*.rRNA.stats.txt -o .
```

metapathways_COG_to_megan.py

```
$ python metapathways_cog_to_megan.py -i *.cog.lastout.parsed.txt
                                       -o output_folder .
                                       [--dsv]
```

```
$ python metapathways_cog_to_megan.py -i HOT_Sanger/*/blast_results/*cog*parsed.txt -
o .
```

metapathways_last_to_megan.py

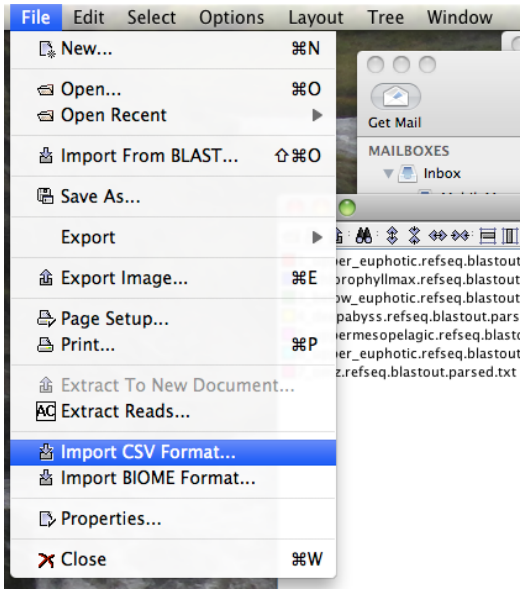
```
$ python metapathways_last_to_megan.py -i *refseq.lastout.parsed.txt
                                       -o output_folder .
                                       [--dsv]
```

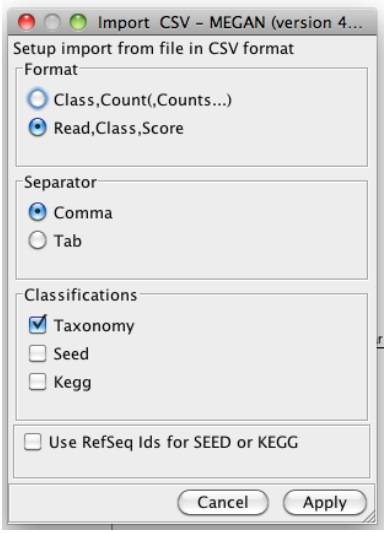
```
$ python metapathways_last_to_megan.py -i HOT_Sanger/*/blast_results/*refseq*parsed.txt
-o .
```

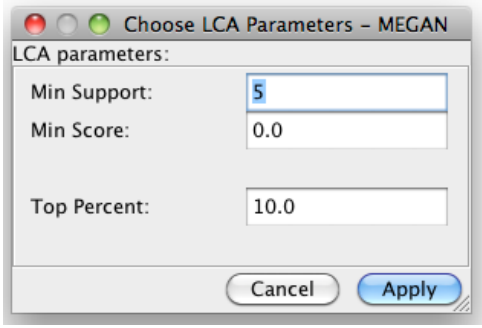
Results: HOT_Sanger_megan.csv.zip, HOT_Sanger_megan_dsvs.zip

3. Loading Results into MEGAN

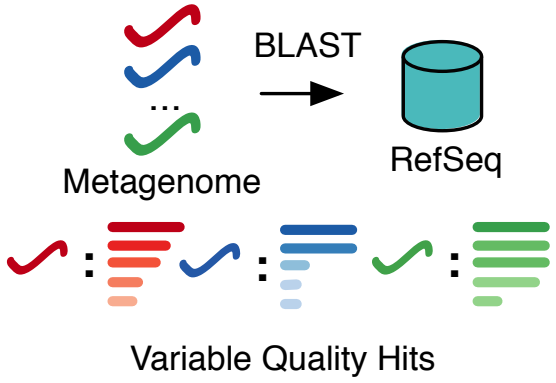
- We will now load our .megan.csv.txt files into MEGAN

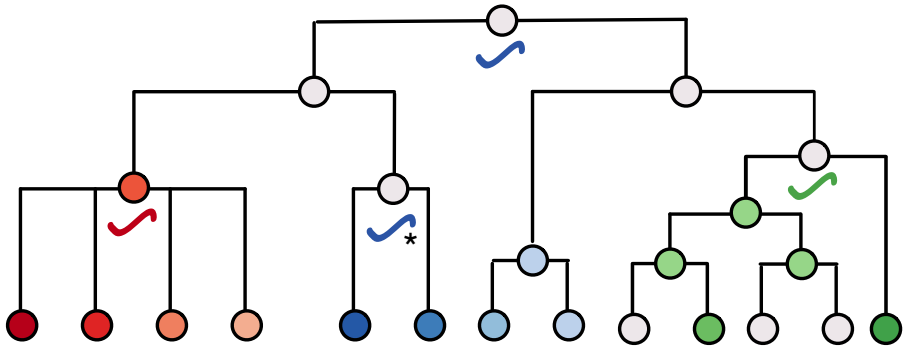






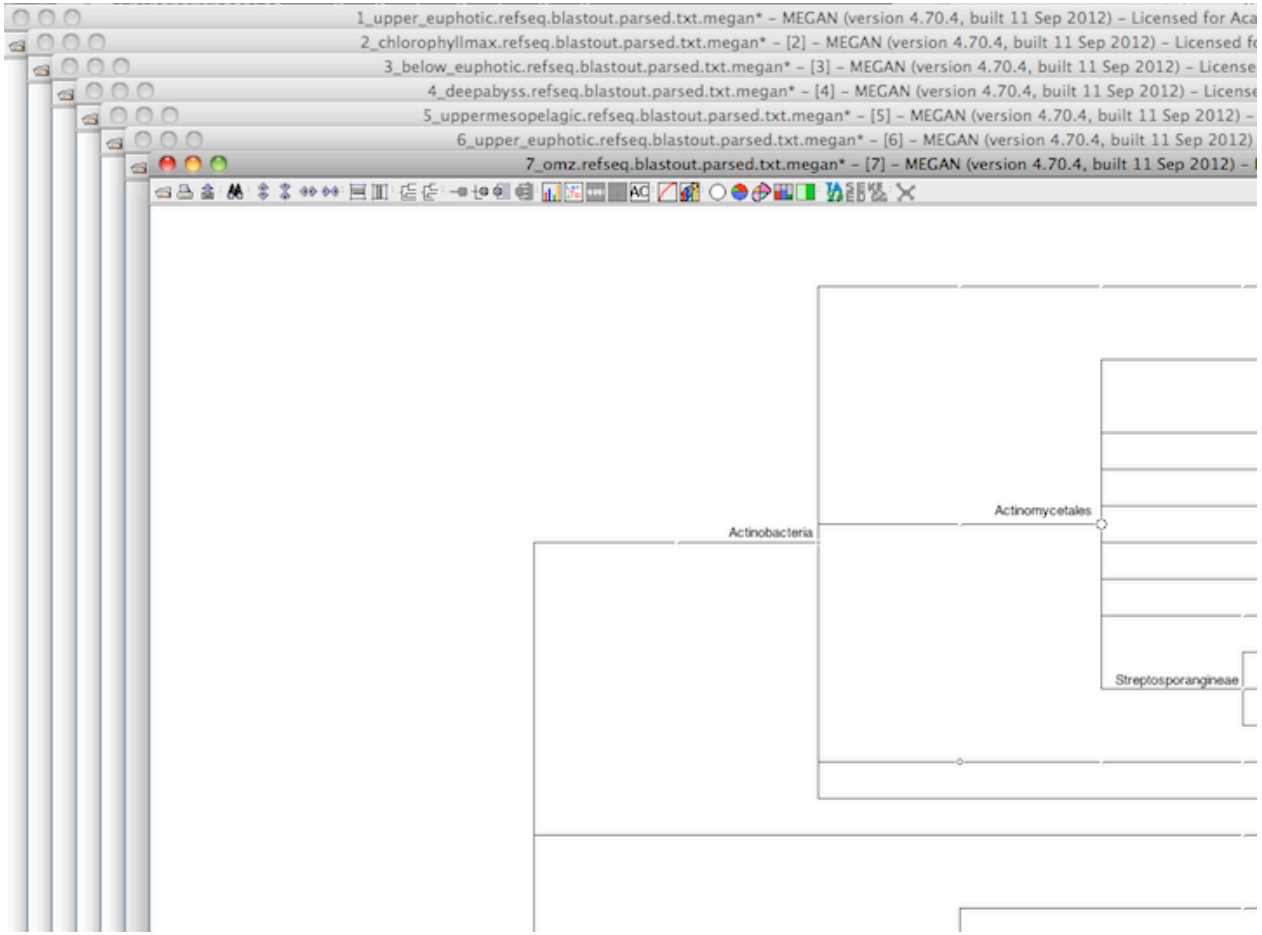
- May want to change Min Support to 1 for rRNA





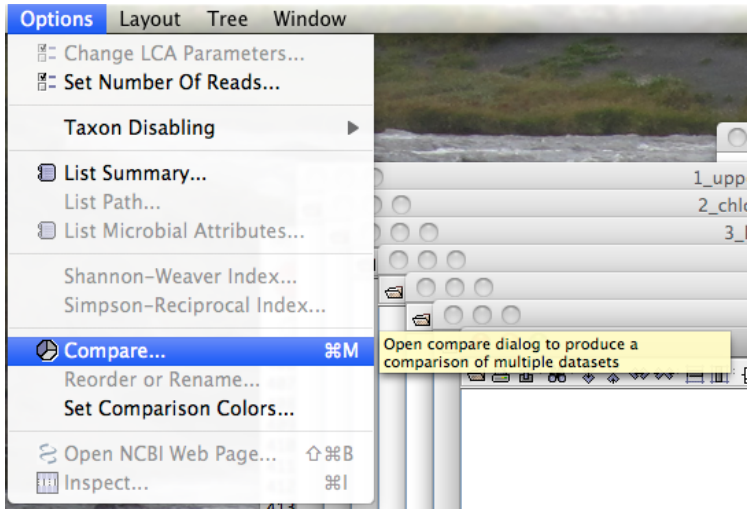
3. Loading Results into MEGAN

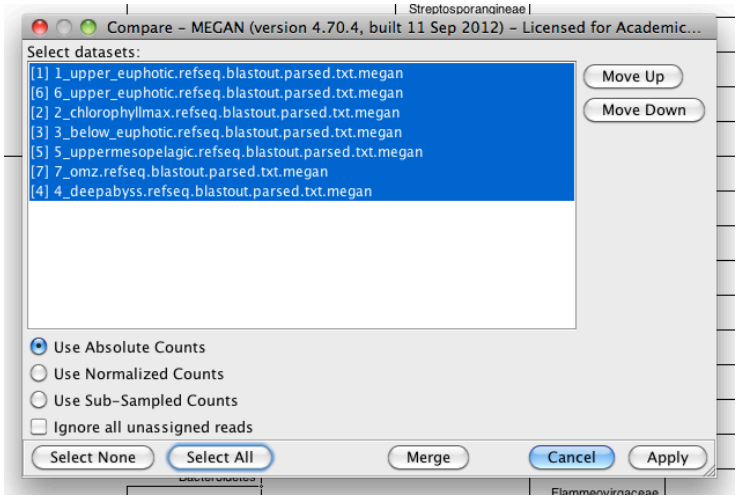
- Repeat the process for all your samples



3. Loading Results into MEGAN

- Options > Compare > Organize Samples





3. Loading Results into MEGAN

- Repeat the process for all your samples

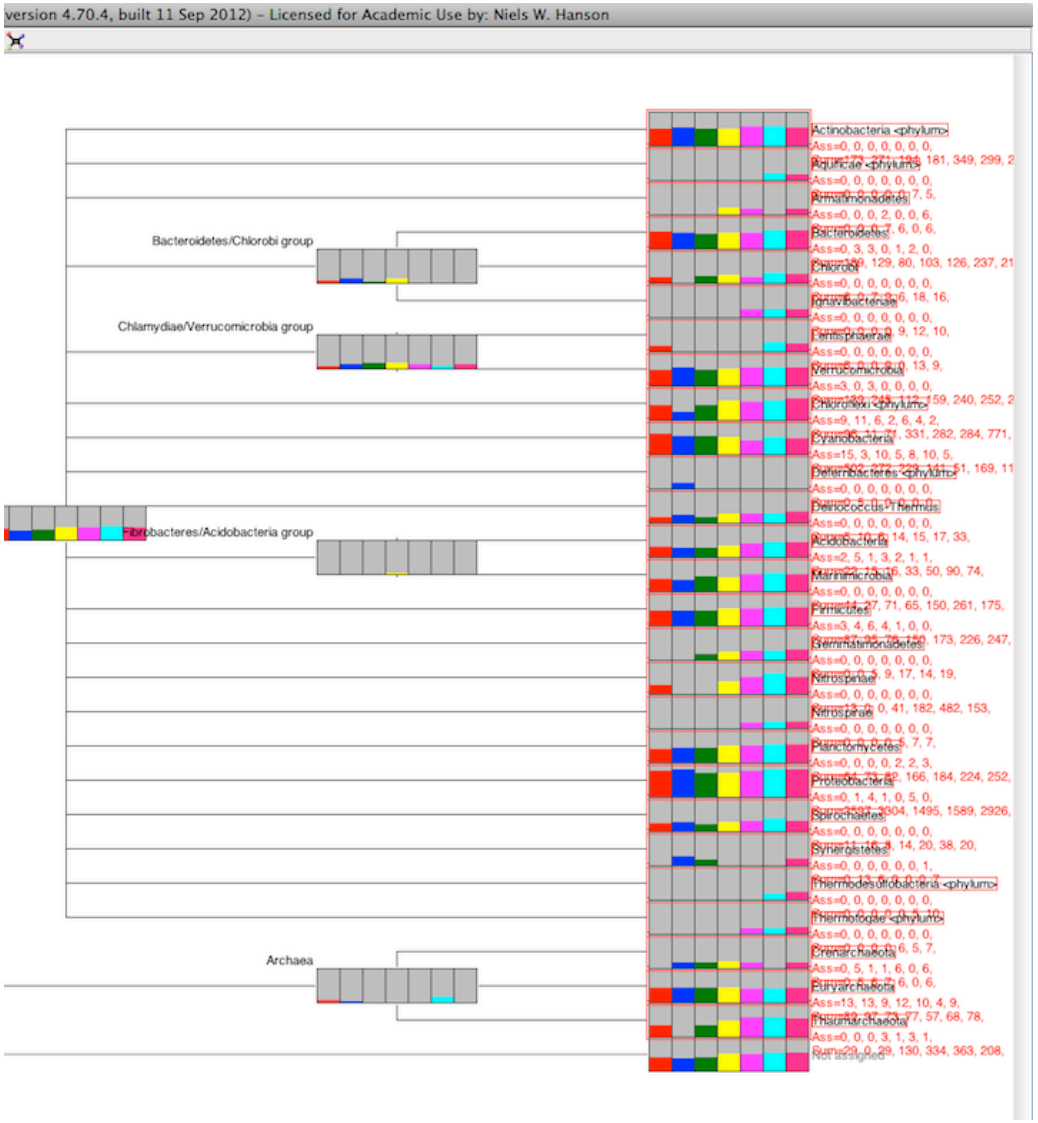
4. Export Taxonomy Results from MEGAN

- Now are going to export the taxonomy hits from MEGAN in an R-compatible format

1. Decline your taxonomy hits to your preferred taxonomic level
   - this will really depend on your samples and biological goals
2. e.g., Tree > Collapse at Level or
   right-click options to un-collapse a given node (e.g., expand *proteobacteria* to the *alpha-*, *beta-*, *gamma-*, and *delta-* subgroups)

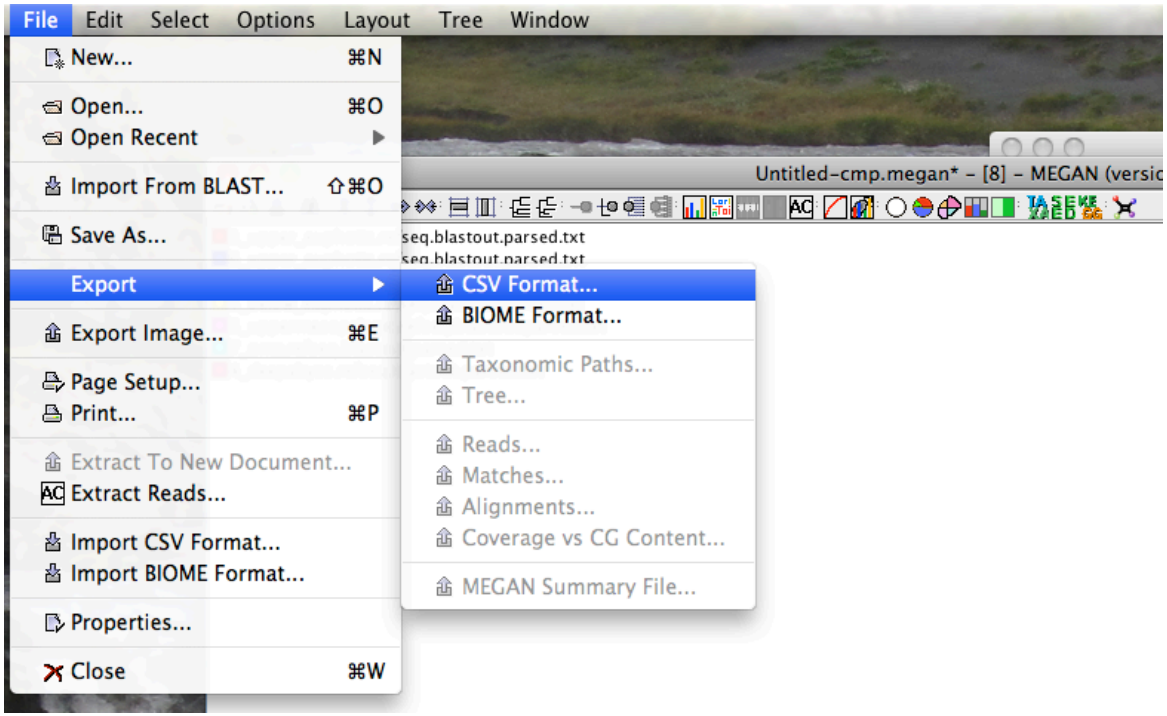
4. Export Taxonomy Results from MEGAN

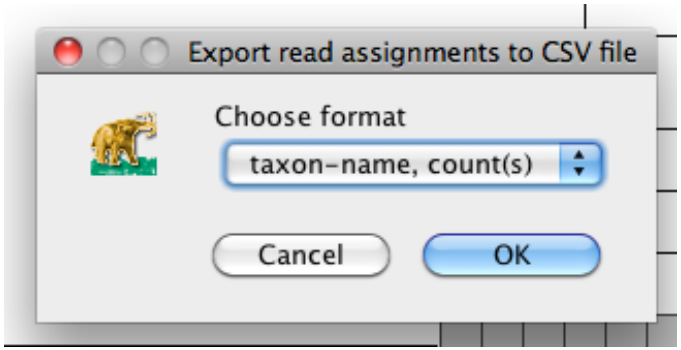
- Select desired taxonomy: e.g. Select > All Leaves

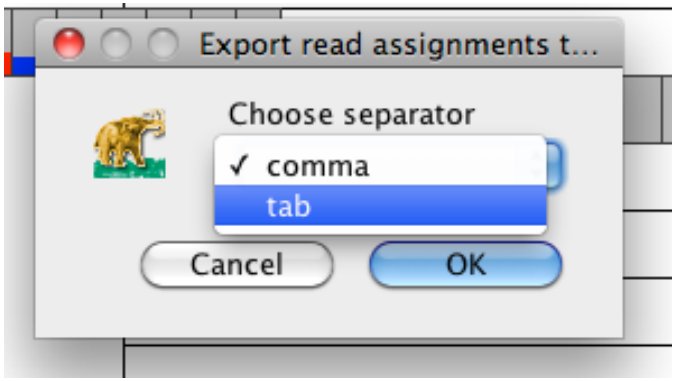


4. Export Taxonomy Results from MEGAN

- Files > Export > CSV Format...







- Tabs separator is nicer as commas can get used in Taxonomy Names

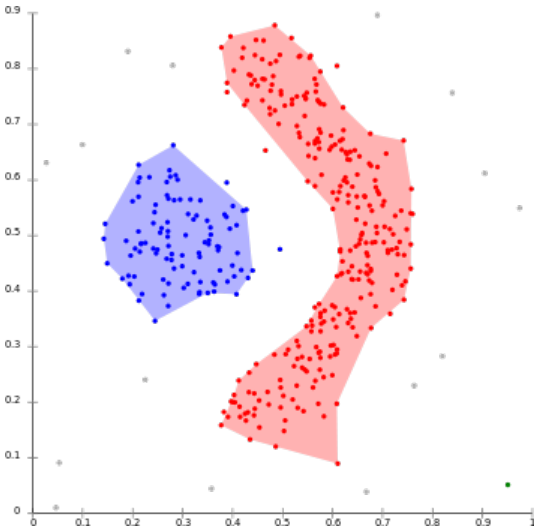
4. Export Taxonomy Results from MEGAN

- So then we have our MEGAN Taxonomy Matrix: HOT_megan_table.txt

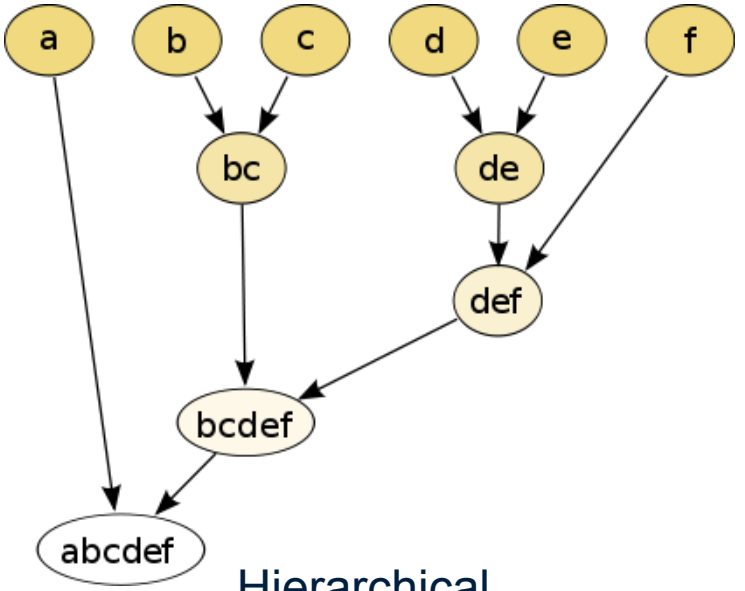
```
head HOT_megan_table.txt
#Datasets    1_upper_euphotic.refseq.blastout.parsed.txt     6_upper_euphotic.refseq.blastout.parsed.txt
4_deepabyss.refseq.blastout.parsed.txt
Actinobacteria <phylum>  173     271     194     181     349     299     252
Aquificae <phylum>   0   0   0   0   7   5
Armatimonadetes  0   0   0   7   6   0   6
Bacteroidetes    189     129     80  103     126     237     211
Chlorobi     6   0   7   9   6   18  16
Ignavibacteriae  0   0   0   0   9   12  10
Lentisphaerae    6   0   0   0   0   13  9
Verrucomicrobia  139     245     112     159     240     252     279
Chloroflexi <phylum>     96  11  71  331     282     284     771
```

5. Hierarchical Clustering in R

- *Clustering* – Methods to put objects into groups that are more similar within each other than between each other
- *Agglomerative* – Bottom-up approach. Groups are built-up one at a time
- *Divisive – Top-down approach. Start with one big group and break it down.*



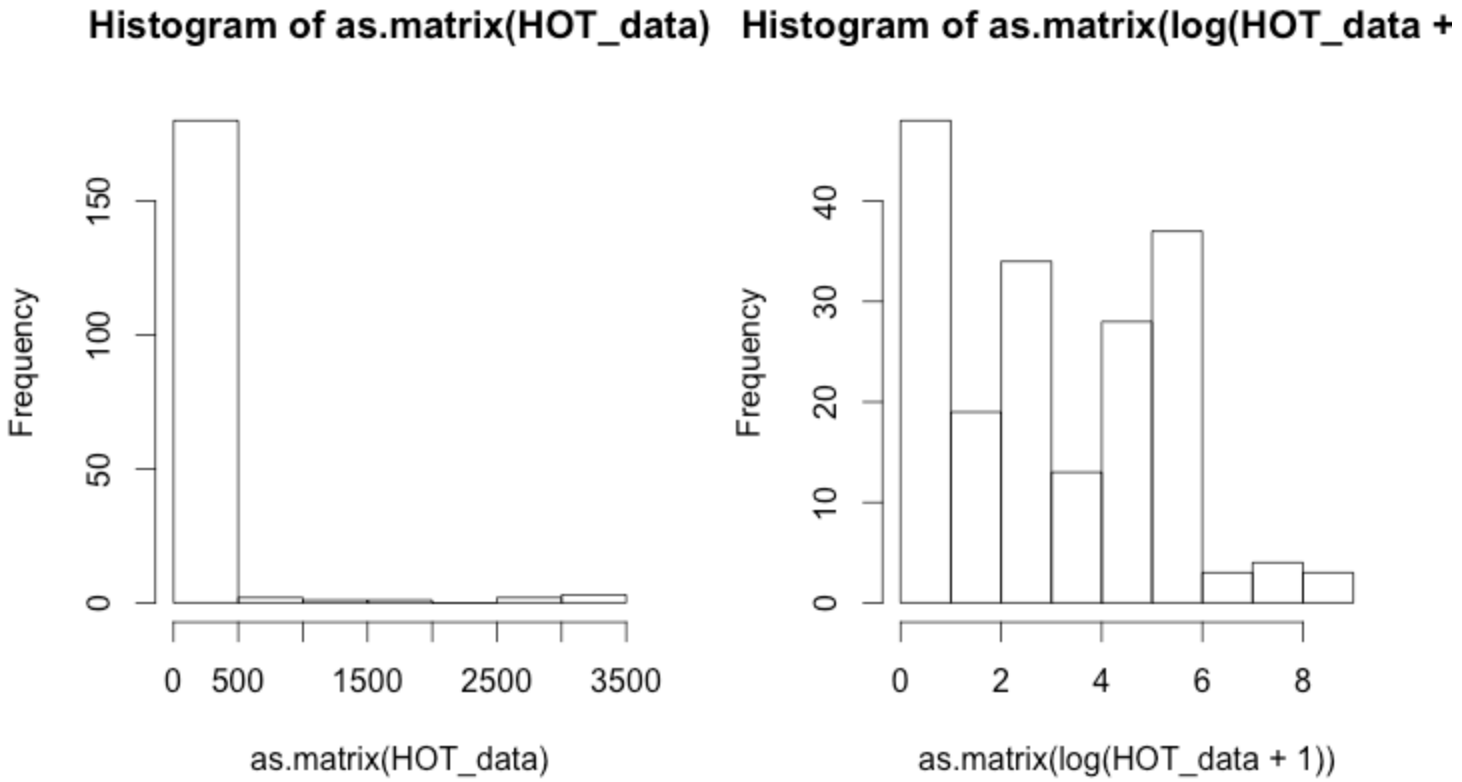
Non-hierarchical



Hierarchical

5. Data Scaling

- *Important to always plot your data to get an idea of relative data ranges*



Distance Metrics: Euclidian & Bray-Curtis

- *Distance Metric* to compare samples. Samples are treated as point in a multi-dimensional taxa space.
- *Euclidian distance*

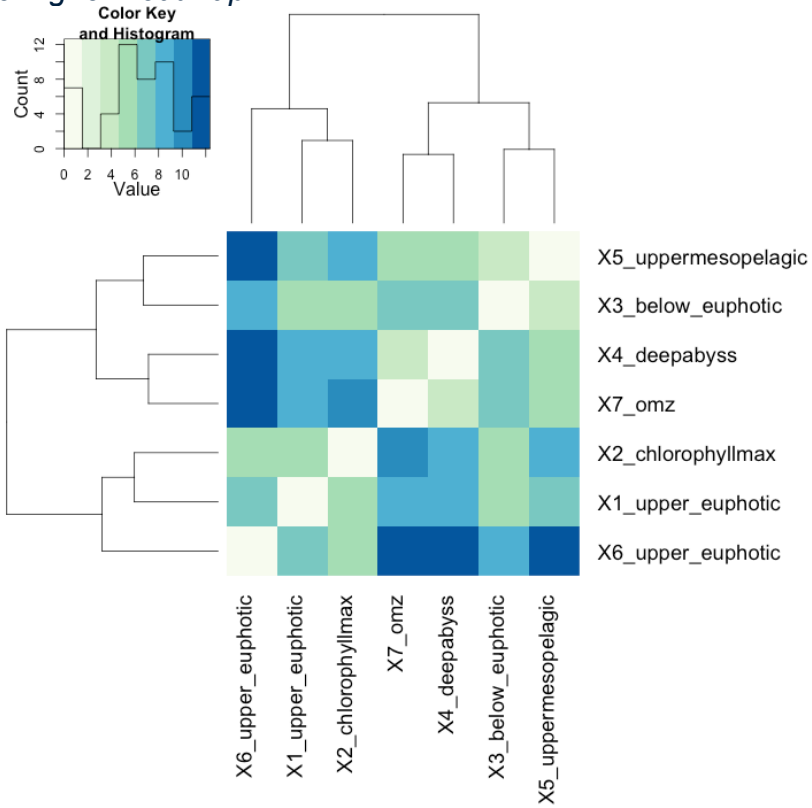
$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

- *Bray-Curtis Dissimilarity* – Set comparison between shared species over *observed* species
- Avoids the *double zero* problem by only looking at observed taxa

$$BC_{ij} = \frac{2C_{ij}}{S_i + S_j}$$

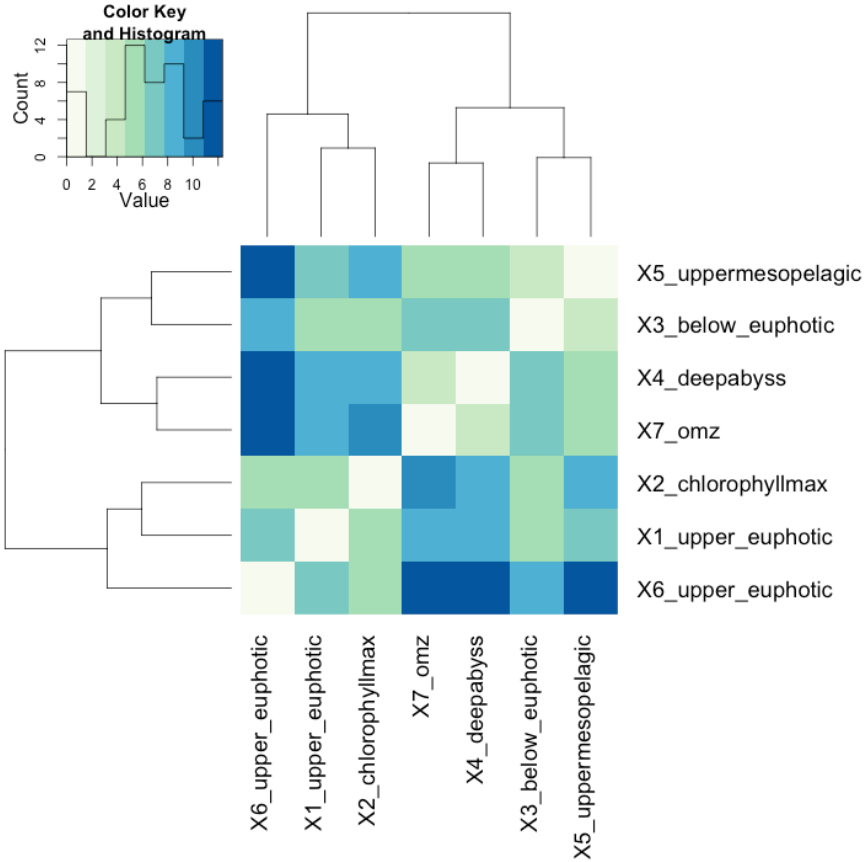
heatmap2()

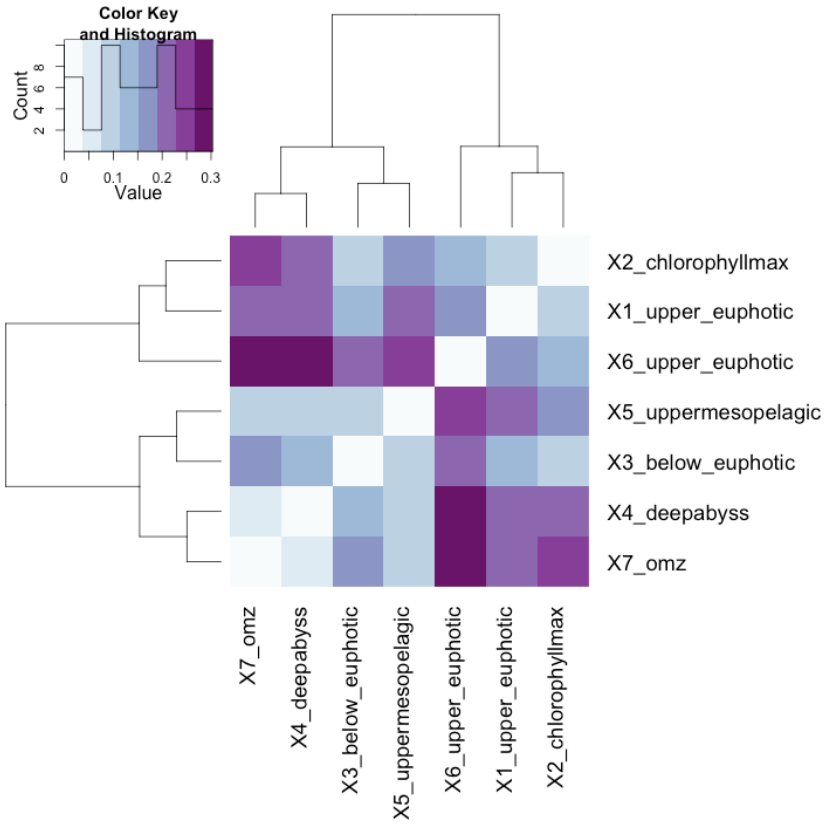
```
setwd("~/Desktop/")
HOT_data <- read.table("HOT_megan_table.txt", header=TRUE, sep="\t", row.names=1)
HOT_data <- log(HOT_data + 1) # transpose
HOT_data.t <- t(HOT_data) # transpose HOT_data
HOT_data.t.dist <- dist(HOT_data.t) # calculate the euclidian distance between the rows
HOT_data.euclid.fit <- hclust(HOT_data.t.dist)
library("gplots")
library("RColorBrewer") # for colours
euclid_dend <- as.dendrogram(HOT_data.euclid.fit) # get ordering for heatmap
my_colours <- brewer.pal(8,"GnBu") # my colours
heatmap.2(as.matrix(HOT_data.t.dist), margin=c(14,14),
                                      Rowv=euclid_dend,
                                      Colv=euclid_dend,
                                      col=my_colours,
                                      trace="none",
                                      denscol="black")
```



heatmap2() with multiple distance metrics

```
try(library("ecodist"), install.packages("ecodist")) # try load, otherwise install
library("ecodist") # try to load again
HOT_data.t.bcdist <- bcdist(HOT_data.t)
HOT_data.bcdist.ward.fit <- hclust(HOT_data.t.bcdist, method="ward")
bc_dend <- as.dendrogram(HOT_data.bcdist.ward.fit) # get ordering for heatmap
my_bc_colours <- brewer.pal(8,"BuPu") # my colours
heatmap.2(as.matrix(HOT_data.t.bcdist), margin=c(14,14), Rowv=bc_dend, Colv=bc_dend,
col=my_bc_colours, trace="none", denscol="black")
```

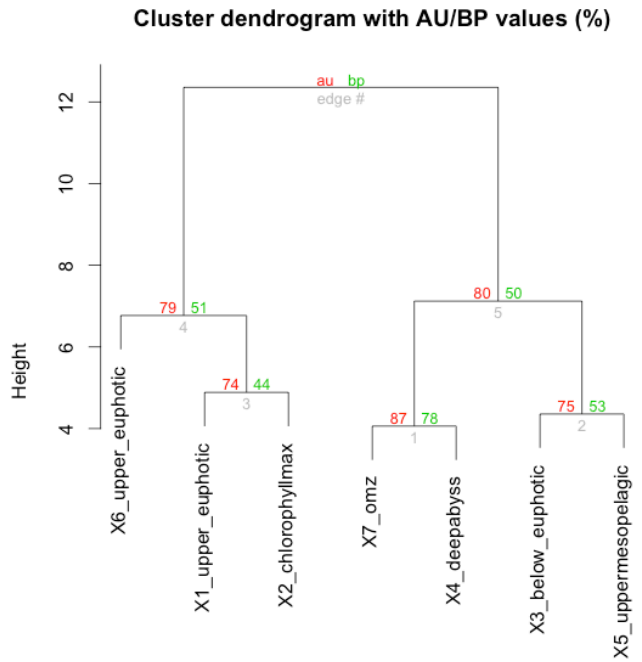




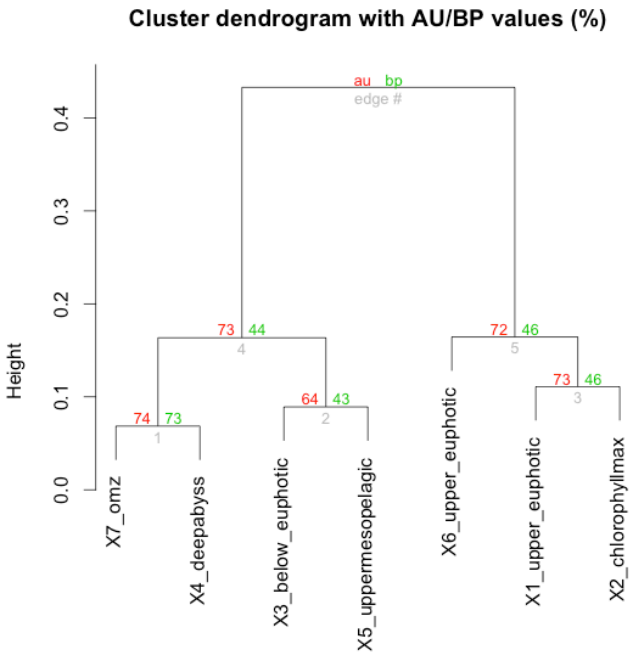
Assessing significance: bootstrapping p

```
library(pvclust)
HOT_data.pv_fit <- pvclust(HOT_data,
                           method.hclust="complete",
                           method.dist="euclidian", n=1000) # in this case no transform is need
plot(HOT_data.pv_fit)

library("devtools") # used to source functions from the internet
source_url('http://raw.github.com/nielshanson/mp_tutorial/master/taxonomic_analysis/code/pvclust_bcdist.R')
HOT_data.bcdist.pv_fit <- pvclust(HOT_data, method.hclust="ward", method.dist="bray–curtis", n=1000)
plot(HOT_data.bcdist.pv_fit)
```



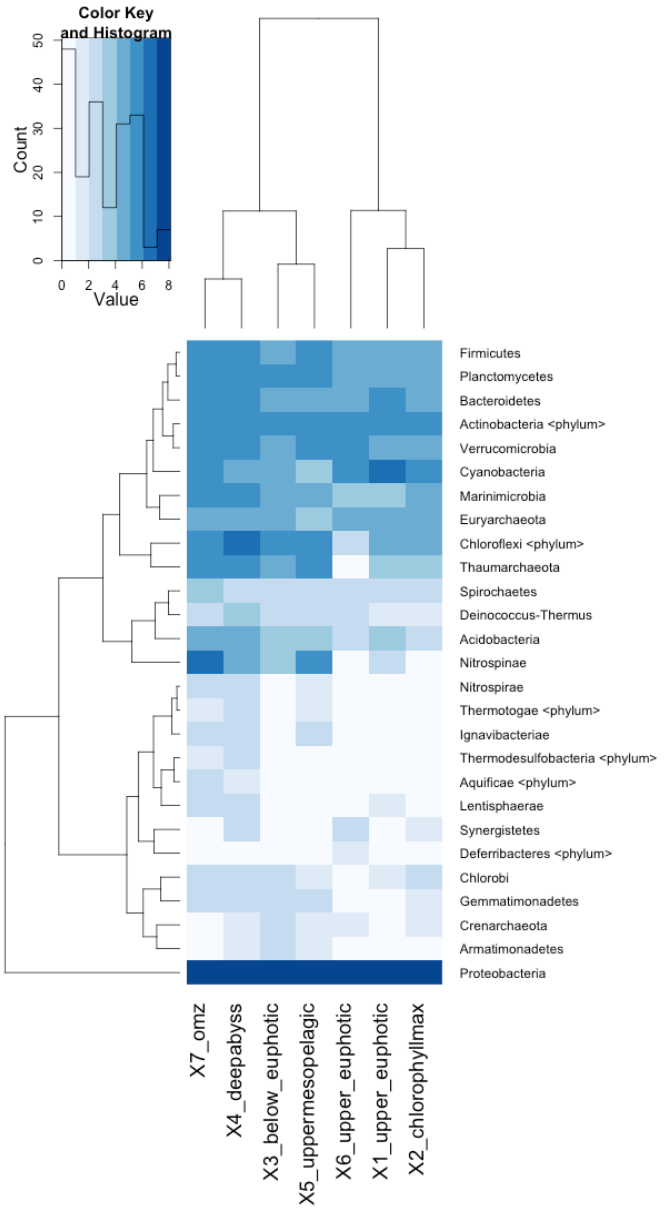
Distance: euclidean
Cluster method: complete



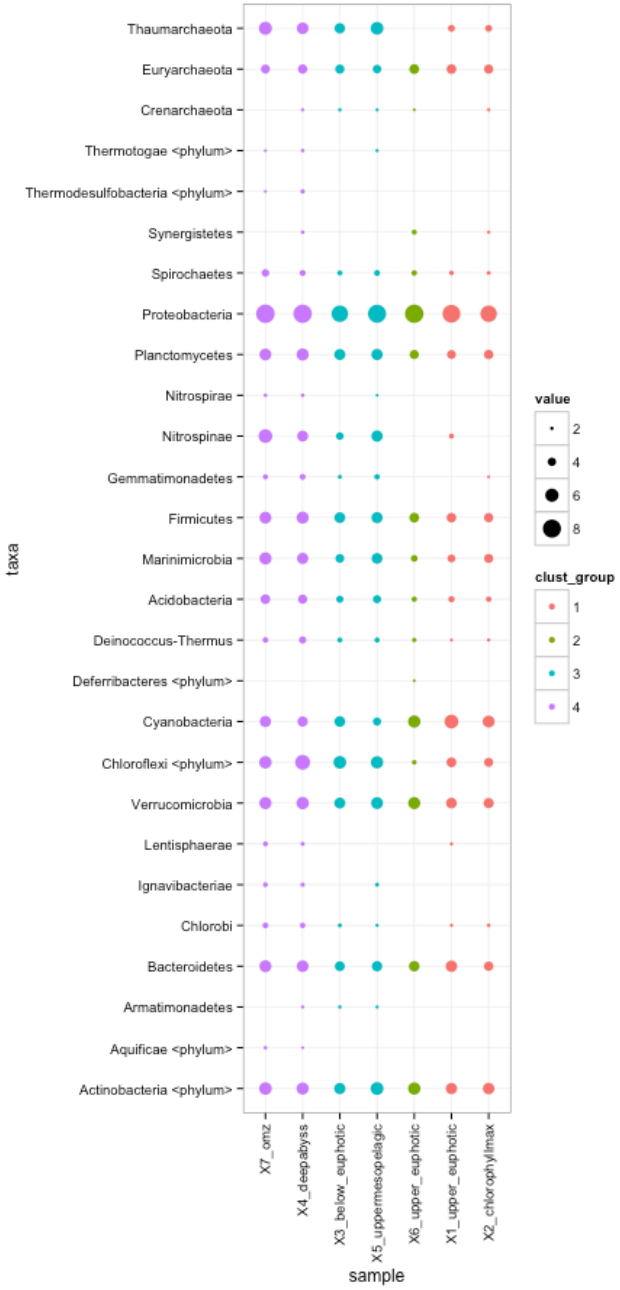
Distance: bray-curtis
Cluster method: ward

Visualizing Clustering on Taxa Profile

```
my_colours <- brewer.pal(8,"Blues") # another colour scheme
HOT_heat <- heatmap.2(as.matrix(HOT_data),
                      margin=c(14,14),
                      col=my_colours,
                      Colv=bc_dend,
                      trace="none",
                      denscol="black")
```



ggplot()


Thaumarchaeota
Euryarchaeota
Crenarchaeota
Thermotogae <phylum>
Thermodesulfobacteria <phylum>
Synergistetes
Spirochaetes
Proteobacteria
Planctomycetes
Nitrospirae
Nitrospinae
Gemmatimonadetes
Firmicutes
Marinimicrobia
Acidobacteria
Deinococcus-Thermus
Deferribacteres <phylum>
Cyanobacteria
Chloroflexi <phylum>
Verrucomicrobia
Lentisphaerae
Ignavibacteriae
Chlorobi
Bacteroidetes
Armatimonadetes
Aquificae <phylum>
Actinobacteria <phylum>
taxa
X7_omz
X4_deepabyss
X3_below_euphotic
X5_uppermesopelagic
X6_upper_euphotic
X1_upper_euphotic
X2_chlorophyllmax
sample
value
2
4
6
8
clust_group
1
2
3
4

ggplot()

```
library(ggplot2)
library(reshape2) # transform our Data from wide to long format
HOT_data$taxa = rownames(HOT_data) # add taxa from rownames to Data Frame
HOT_data.m <- melt(HOT_data)
colnames(HOT_data.m)[2] = "sample" # rename variable column to sample

# in order to plot things in properly, the order of each variable has to be explicitly set
name_order <- HOT_data.bcdist.ward.fit$labels[HOT_data.bcdist.ward.fit$order] # get order of samples from clustering
HOT_data.m$sample <- factor(HOT_data.m$sample, levels=name_order) # set order of samples
HOT_data.m$taxa <- factor(HOT_data.m$taxa, levels=unique(HOT_data.m$taxa)) # set order of taxa

# cut bray-curtis clustering to get groups
bc_ward_groups <- cutree(HOT_data.bcdist.ward.fit, h=0.12) # slice dendrogram for groups (hight=0.2)
HOT_data.m$clust_group <- as.vector(bc_ward_groups[as.vector(HOT_data.m[,"sample"])])
HOT_data.m$clust_group <- as.factor(HOT_data.m$clust_group) # set group numbers as factors

# finally create the bubble plot
g <- ggplot(subset(HOT_data.m, value >0), aes(x=sample, y=taxa, color=clust_group))
g <- g + geom_point(aes(size=value)) # plot the points and scale them to value
g <- g + theme_bw() # use a white background
g <- g + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) # rotate and centre labels
g # plot it, whew!
```

Questions?