By Ashley Shade, for use by EDAMAME workshop students at Michigan State University in August 2014

Tutorial Day 3

Day 3. Analyzing community patterns: Introduction, visualization, and identifying categorical groups

Software: QIIME, R

1. Make phylogenetic resemblance matrices for analysis of beta-diversity in QIIME

Welcome back!

We will make three kinds of resemblance matrices (sample by sample comparisons) for analysis of beta-diversity. We will use QIIME to calculate the phylogenetic metrics: weighted and unweighted UniFrac. We will then use R to calculate the taxonomic metrics: Bray-Curtis and Sørenson.

Open terminal, navigate to the data directory for analysis, and call QIIME (macqiime). We will use the beta_diversity function to calculate weighted and unweighted UniFrac matrices.

```
$ beta_diversity.py -i Manduca_otu_table_even861.biom -m
weighted_unifrac,unweighted_unifrac -o beta_div_even861/ -t Manduca_rep_phylo.tre
```

Open both the unweighted and weighted UniFrac tables in Excel, and compare their values. The unweighted UniFrac does not account for differences in the relative abundances of taxa, while weighted UniFrac does, thereby "up-weighting" prevalent taxa and "down-weighting" rare taxa accordingly. Later today, explore how weighting influences downstream analyses, including hypothesis testing.

2. Using QIIME for ordination: PCoA

We can also use QIIME functions to easily make an ordination, using principle components analysis (PCoA). Open terminal and navigate to the main Manduca analysis directory. We'll perform PCoA on both weighted and unweighted UniFac resemblance matrices, and compare them.

```
$ principal_coordinates.py -i beta_div_even861/ -o beta_div_even861_PCoA/
```

Then, we can make 2d or 3 d plots of the output, and map the colors to the categories in the mapping file:

```
$ make_2d_plots.py -i beta_div_even861_PCoA/ -m Manduca_map.txt -o
plotting_PCoA2d_even861/ --ellipsoid_opacity 0
```

```
$ make_3d_plots.py -i beta_div_even861_PCoA/ -m Manduca_map.txt -o
plotting_PCoA3d_even861/ --ellipsoid_opacity 0
```

Navigate to the .html files and click on the link. Take some time to explore these 2d and 3d plots carefully - toggle samples, note color categories, hover over points to examine sample IDs. What hypotheses can be generated based on exploring these ordinations?

3. Introduction to R: formatting files and reading in tables

Create a new directory for R analyses, above the Manduca_raw_data directory. This is helpful to keep analyses separate, and to not accidentally alter the output files from QIIME that you may be using repeatedly for various analyses. **Make copies** of the following files, and move the copies into the new directory:

The classic OTU table:
Manduca_otu_table_even861

The mapping file:
Manduca_map.txt

The phylogenetic resemblance matrices:
beta_div_even861/unweighted_unifrac_Manduca_otu_table_even861.txt
beta_div_even861/weighted_unifrac_Manduca_otu_table_even861.txt

Open each of these files in Excel and sort them by SAMPLE IDs so that every sample is in the same order. You will have to sort both rows and columns for the resemblance matrices. Also, because the samples IDs do not have the same length, they will not be in consecutive order (e.g., KM1, KM10, KM11… instead of KM1, KM2, KM3). This is okay, as long as all of the samples are in the same order in each file. Save the files with "_sorted" appended to their names in the same directory.

Hints for sorting a UniFrac matrix in Excel:

1. Sort by Rows. Highlight all of the data EXCEPT for the first column (which contains row IDs). From the excel Menu, select Data -> Sort. Click on the Bottom Left button "Sort Options." Select "sort left to right." Select OK. Select "Sort by Row 1". Click okay. Cells should be sorted left to right, starting with sample KM1 in the upper left.
2. Sort by Columns. Highlight all of the data. From the Excel Menu, select Data -> Sort. Click on the bottom left button "Sort options". Select "Sort top to bottom". Select OK. At the bottom of the box select "Header row." At the top of the box select "(Column A"). Select OK. Cells should be sorted from top to bottom.
3. Check that the diagonal is ZERO. If so, you have sorted effectively!

Open R, and if you haven't already, install the vegan package for community ecology. On a Mac, this is down by going to the menu bar, Tools -> Install packages. Select the box to also install dependencies (dependencies = existing packages/functions that vegan needs to execute some of its functions)

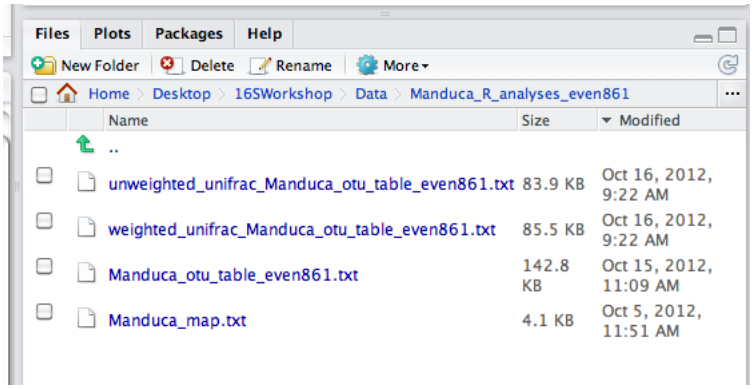
Note: Here is a link to info about the vegan package: http://vegan.r-forge.r-project.org/

Then, to load the functions of the vegan package into your current R environment, in the R console and type:

```
> library(vegan)
```

Now, change directory to the one that you just created. On the Mac menu bar, select Tools -> Set working directory -> Choose directory.

Or, if you are using R Studio, navigate to the lower right panel, click on the "**Files**" tab, and the click on the top right-hand side **"…"** button (below the refresh arrow, see below), to browse to the correct directory. Then click on the "**More**" tab, and select "set as working directory."



Or, you could use the **"setwd()"** command in the console. Lots of options.

Now, we will use our OTU table to calculate two other resemblance matrices using functions from the vegan package.

First, we must read in the OTU table. Before R will accept the OTU table, we must do a little bit of formatting. Open the tab-delimited file in Excel (or, if you are using R Studio, you can open it there), and clear the "#OTU ID" cell. Then, save the file (make sure it is a tab-delimited txt file). The "#" tells R to skip reading the row, but this would remove our sample names and cause an error with analysis. Then, we will read the table into R using the **"read.table()"** command. We name the table "otu" : this makes it easy to call in R.

```
> otu=read.table("Manduca_otu_table_even861_sorted.txt", header=TRUE, row.names=1,
sep="\t", check.names=FALSE)
```

Then, use the **"head()"** command to view the top of otu. Check that the sample IDs (KM1, KM59, etc), and the OTU IDs (0, 145, 786, etc), are correct, and that the formatting seems okay:

```
> head(otu)
   KM1 KM10 KM11 KM12 KM13 KM14 KM15 KM16 KM17 KM18 KM19 KM2 KM20 KM21 KM22 KM23 KM24
KM25 KM26
1    0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
2   14    0    2    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
6    0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
7    0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
8    0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
10   0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0    0
```

Now, let's explore working with tables. In R, we use brackets **[rows, columns]** to specify rows and columns within a table. What is the value in your OTU table that is in the 19th row and 23 column?

```
> otu[19,23]

[1] 0
```

You can also use colons to specify ranges, as in the example below, where we've selected the first five rows and the samples in columns 5 through 10.

```
> otu[1:5, 5:10]
  KM13 KM14 KM15 KM16 KM17 KM18
1    0    0    0    0    0    0
2    0    0    0    0    0    0
6    0    0    0    0    0    0
7    0    0    0    0    0    0
8    0    0    0    0    0    0
```

We can also extract row names (OTU IDs) and column names (sample IDs) using R:

```
> row.names(otu)
  [1] "1"   "2"   "6"   "7"   "8"   "10"  "12"  "15"  "19"  "20"  "24"  "27"  "29"
"30"  "31"
 [16] "36"  "38"  "39"  "40"  "41"  "42"  "44"  "45"  "46"  "48"  "49"  "50"  "51"
"54"  "55"
 [31] "57"  "59"  "60"  "61"  "64"  "66"  "72"  "74"  "79"  "80"  "81"  "83"  "88"
"89"  "92"
 [46] "96"  "98"  "109" "112" "115" "116" "118" "119" "121" "123" "125" "127" "130"
"132" "133" …

> colnames(otu)
 [1] "KM1"              "KM10"             "KM11"             "KM12"
 [5] "KM13"             "KM14"             "KM15"             "KM16"
 [9] "KM17"             "KM18"             "KM19"             "KM2"   …
```

We can also use the sample or OTU IDs to explore values in the OTU table. Look at the sample called KM41 by calling this column name in quotations:

```
> otu[,"KM41"]
  [1]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
0   0
 [23]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  19   0   0   0
0   0
 [45]   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   3   0
0   0
 [67]   0   0  37   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
 [89]   0   0  82   0   0   0   0  10   0  28   0   0   0   0   0   0   0   0   0   0
0   0
[111]   0   0   0   4   0 421   0   0   0   0   0   2   0   1   0   0  94   0   0   1
0   0
[133]  43   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[155]   0   0   0   1   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0
2   0
[177]   0   0   0   0   0   0   0   0   0   2   0   3   0   0   0   1   3   0   0   0
0   2
[199]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[221]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[243]   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0
0   0
[265]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[287]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[309]  83   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[331]   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[353]   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
[375]   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
```

Look at OTU 134 by calling this row name in quotations:

```
> otu["134",]
    KM1 KM10 KM11 KM12 KM13 KM14 KM15 KM16 KM17 KM18 KM19 KM2 KM20 KM21 KM22 KM23 KM24
KM25
134   0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
0
    KM26 KM27 KM28 KM29 KM3 KM30 KM31 KM32 KM33 KM34 KM36 KM37 KM38 KM39 KM4 KM40 KM41
KM42
134    0    0    0    0   0    0    0    0    0    0    0    0    0    0   0    0    0
0
    KM44 KM45 KM46 KM47 KM48 KM49 KM5 KM50 KM51 KM52 KM53 KM54 KM55 KM56 KM57 KM58
KM59 KM6
134    0    0    0    0    0    0   0    0    0    0    0    0    0    0    0    0
0   0
    KM60 KM61 KM62 KM63 KM64 KM65 KM66 KM67 KM68 KM69 KM7 KM70 KM71 KM72 KM73 KM74
KM75 KM76
```

```
134    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0
17    0
    KM77 KM78 KM8 KM9
134    0    0   0   0
                                                                  ConsensusLineage
134 Bacteria; Firmicutes; Bacilli; Lactobacillales; Carnobacteriaceae; Dolosigranulum
```

Finally, we can check the dimensions of the table, to make sure that it is the size we expect:

```
> dim(otu)
[1] 389  77
```

Get into the habit of checking your tables to make sure they match your expectations. Let's start with the columns. This should be the number of samples PLUS one column that has the taxonomic IDs (labeled "ConsensusLineage"). We originally had 78 samples, but one did not amplify (KM35), leaving us with 77 to begin our QIIME workflow. Then, at the QIIME alpha_rarefaction step, we removed one sample that had too few sequences to continue with analysis (KM43), leaving us with 76 in total. 76 samples plus 1 ConsensusLineage column = 77. Thus, the number of columns are as expected.

Now, what about the rows? Originally, we had picked 782 OTUs using uclust in QIIME (uclust_picked_otus/seqs_otus.log). Then, we found 159 chimeras using ChimeraSlayer, leaving us with 623 OTUs in the `Manduca_otu_table.biom` OTU table before rarefaction. Then, we subsampled (rarefied) to an equal sequencing depth of sequences per sample. This also reduced the number of OTUs, as many OTUs that had low sequence abundance may not have randomly selected to be included among the 861. Note that not everyone will have the same number of OTUs - this will depend on the random subsampling, and will be slightly different after different iterations, though we would expect that prevalent taxa will be similarly represented because there is a higher likelihood of randomly sampling a sequence from an abundant taxon. When using the `per_library_stats.py -i Manduca_otu_table_even861.biom` command, I found that, after rarefaction, I had 389 OTUs remaining. This matches the number of rows in otu, and so my expectations of the input table are met.

Note: If for any reason your expectations of your own dataset are not met, check formatting of the input files. In my experience, formatting issues are the number one reason for errors.

Now, we will read in the mapping file and use the "attach" function to link the information to the OTU table. But, there is some formatting that we must do first. **First**, open the file and remove the "#" on the first line, but maintain the column name "SampleID". The R analyses only work if the samples are in the same order across files! **Second**, remember that there was one sample, KM43 that had low sequences and was omitted from the final OTU table. We need to omit it from the map file also.

```
> map=read.table("Manduca_map_sorted.txt", header=TRUE, sep="\t")
```

For the next step, we will remove the ConsensusLineage labels from the table, as we do not need them at the moment. We do this by first calling the labels as a vector, rdp, and then

removing the whole column (column 77) from the table. Now, we have a vector of rdp ConsensusLineage assignments called **rdp**, plus a "classic" sparse OTU table, called **otu**. You can use your R workspace browser (upper right window in R studio) to keep track of the objects you have in your R environment, and their dimensions.

```
> rdp=otu[,77]
> names(rdp)=row.names(otu)
> otu=otu[,-77]
> dim(otu)
[1] 389  76
```

4. Beta diversity in R: making, reading in, and comparing resemblance matrices

We will use the vegdist function to calculate a Bray-Curtis and a Sørenson resemblance matrix from the OTU table. Navigate to the help window, and type in **"vegdist()"** to see the arguments for the function. We first need to transpose the OTU table using the **"t()"** command to have species in columns and samples in rows, just a formatting difference for the function - no big deal. It is easy to transpose the table directly inside the vegdist command.

```
> braycurtis.d=vegdist(t(otu), method="bray")
> head(braycurtis.d)
[1] 0.1974448 0.1858304 0.1254355 0.1684088 0.1742160 0.1463415
```

Now braycurtis.d is the name of our new resemblance matrix. I use the .d ending to remind myself that it is a resemblance matrix. R sees a resemblance matrix as something different than a table or data frame, as it sees the otu table. Instead, R sees a very long vector. That is why the "**head()"** command returns the first few values of a vector instead of a table with rows and columns.

Sorenson's similarity has the exact same calculations as bray-curtis, except that it does not weight species. Therefore, we make a Sorenson matrix using the exact same protocol, except that we *set the binary argument to TRUE* to tell R not to use the information about the taxa relative abundances in the OTU table. Compare the head of sorenson.d to that of braycurtis.d.

Note: Bray-Curtis can be calculated as a similarity or dissimilarity metric, but it is traditionally used as a dissimilarity, as it is in the vegan package. This means that the first two samples compared are 0.19 (%) dissimilar (or, 0.81 similar) using Bray-Curtis, but are 0.45 dissimilar (0.55 similar) using Sørenson. This is typical, as not using information about the relative contributions of taxa may make patterns become less apparent. UniFrac is also a metric of dissimilarity, so the same logic applies for their interpretation.

```
> sorenson.d=vegdist(t(otu), method="bray", binary=TRUE)
> head(sorenson.d)
[1] 0.4545455 0.4000000 0.4736842 0.4285714 0.4375000 0.3548387
```

Now, we read in the UniFrac resemblance matrices made in QIIME. When we read them into the R environment, R recognizes them as tables. We use the "**as.dist()"** command to let R know that these are resemblance matrices. Run the command lines one at a time, and inspect the head of each object as you go along. We name the new resemblance matrices "weighted_u.d" and "unweighted_u.d"

```
> weighted_u=read.table("weighted_unifrac_Manduca_otu_table_even861_sorted.txt",
header=TRUE, row.names=1, sep="\t")

> head(weighted_u)

> weighted_u.d=as.dist(weighted_u)

> head(weighted_u.d)

> unweighted_u=read.table("unweighted_unifrac_Manduca_otu_table_even861_sorted.txt",
header=TRUE, row.names=1, sep="\t")

> head(unweighted_u)

> unweighted_u.d=as.dist(unweighted_u)

> head(unweighted_u.d)
```

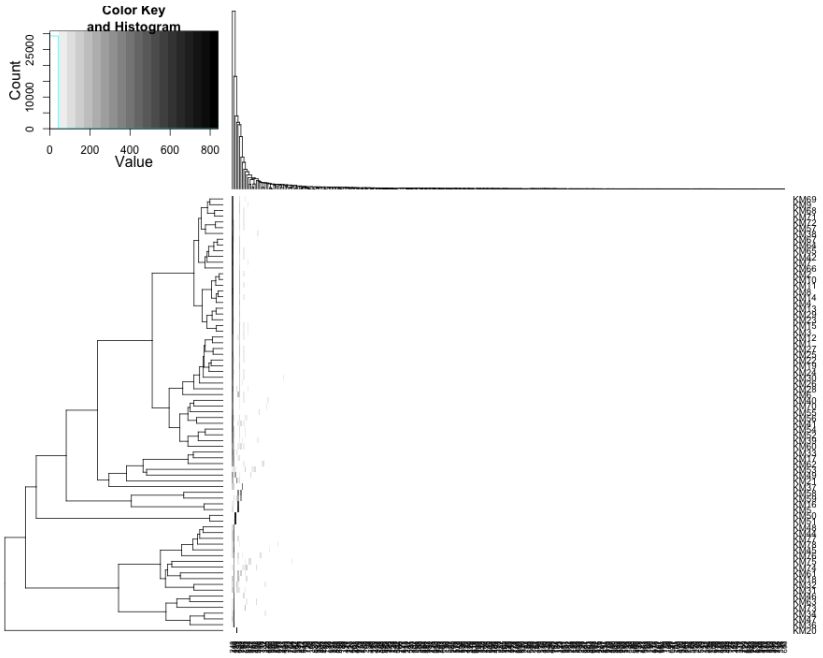
5. Visualizing community data in R

A. Heatmaps + clustering

It is quick and easy to generate a heatmap from an otu table. First, install the "gplots" package in the same way that you installed vegan. Don't forget the dependencies. We will set colors to range from white to black, using the "**colorRampPalette()"** command with twenty steps between the minimum and the maximum values. Then, we call heatmap.2(), again transpose the OTU table, and, set the argument trace = "none" so that obnoxious turquoise lines don't show up in the final plot.

```
> library(gplots)
> colors <- colorRampPalette(c("white", "black"))(20)
> heatmap.2(t(otu), trace="none",col=colors)
```

Inspect the plot. Zoom in if you can, as it may be hard to distinguish the sample IDs and the OTU IDs because the dataset is large. The default heatmap.2 algorithm clusters samples that have similar compositions, and also OTUs that have similar abundances and occurrence patterns. However, remember that these OTUs vary widely in their abundances, and so similarly abundant OTUs will cluster together. Futhermore, because most of the OTUs in the dataset are in low abundance (rare), most of the heatmap is white, meaning that these OTUs have values close to zero.



B. Ordination in R

We will use the metaMDS function to apply non-metric multidimensional scaling analysis to ordinate our data. NMDS is a robust method that makes few assumptions about the data, so it is a good "default" choice for ordination in the absence of any other information. When plotting, we set type = t to show the sample IDs.

```
> braycurtis.mds <- metaMDS(braycurtis.d)
> plot(braycurtis.mds, type="t")
```

We apply metaMDS to all of the resemblance matrices and compare them by dividing the plot space into 2 rows and 2 columns with the **par()** command. Inspect the plots- zoom in if you like. How are they different? How to binary metrics (Sørenson, unweighted UniFrac) differ in pattern from weighted metrics (Bray-Curtis, weighted UniFrac)? How do phylogenetic metrics (weighted UniFrac, unweighted UniFrac) differ in pattern from taxonomic matrics (Bray-Curtis, Sørenson)?

```
> sorenson.mds=metaMDS(sorenson.d)
> weighted_u.mds=metaMDS(weighted_u.d)
> unweighted_u.mds=metaMDS(unweighted_u.d)

> par(mfrow=c(2,2))
> plot(braycurtis.mds, type="t", main= "Bray-Curtis")
> plot(sorenson.mds, type="t", main="Sorenson")
> plot(weighted_u.mds, type="t", main="Weighted UniFrac")
> plot(unweighted_u.mds, type="t", main="Unweighted UniFrac")
```

7. Writing out (exporting) tables from R

We will save our resemblance matrices to our working directory so that we don't have to re-calculate them every time we continue analyses, or if we want to use these matrices. Remember that resemblance matrices are actually very long vectors. R needs to think of these as tables so that it can write them out with the correct delimitations. Thus, we take the opposite approach as we did when we read in the table. We use the "**as.matrix()"** command to convert the vector to a table.

```
> braycurtis=as.matrix(braycurtis.d)
```

Then, we use the the **"write.table()"** command to export the data. In this command, the first argument is the object that we want to export, the second argument (given in quotations) is what we want the name of the exported file to be - don't forget the extension, we set row.names and col.names to TRUE because we have both, we specify that we want the exported file to be tab-delimited by setting the sep="\t" argument, and, finally, we specify quote=FALSE, as otherwise R will put every value in the exported table in quotes, which is annoying. Use the same commands with the Sorenson matrix.

```
> write.table(braycurtis, "BrayCurtis_even861.txt",row.names=TRUE,
col.names=TRUE,sep="\t", quote=FALSE)

> sorenson=as.matrix(sorenson.d)
> write.table(sorenson, "Sorenson_even861.txt",row.names=TRUE,
col.names=TRUE,sep="\t", quote=FALSE)
```

If you open these resemblance matrices in Excel, you will notice another formatting issue. You must insert a top leftmost cell so that the sample IDs (column names) are shifted one over. This may seem confusing, but inspect one of the exported files in Excel and everything will be illuminated.