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

Tutorial Day 5

Day 5. Analyzing community patterns: The contributions of individual OTUs to community dynamics

Software: R

1. Get set up in R studio

As before, read in the Bray Curtis resemblance matrix, the OTU table, and the mapping file. Don't forget to assign the RDP column as "rdp" - we'll need these taxonomic IDs for today's analyses.

```
> braycurtis=read.table("BrayCurtis_even861.txt", header=TRUE, row.names=1, sep="\t")
> braycurtis.d=as.dist(braycurtis)
> otu=read.table("Manduca_otu_table_even861_sorted.txt", header=TRUE, row.names=1,
sep="\t", check.names=FALSE)
> rdp=otu[,77]
> names(rdp)=row.names(otu)
> otu=otu[,-77]

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

2. PROTEST: Procrustean superimposition analysis

We will use the **protest()** function from the vegan package to compare the time series of different treatments to one another. For this to work, it is essential that the same observations are in the same order across treatments. PROTEST cannot read the sample IDs, and so it is not checking to make sure that instar 1 from Treatment 1 and instar 1 from Treatment 2 are being compared, rather than, for example, instar 1 from Treatment 1 with instar 4 from Treatment 2. Order is important!

First, designate treatment groups from the map file using the **unique()** function, we did previously. Then, use the custom-function **makeRedresem.f()** to generate a resemblance matrix for each treatment. Inspect the function, and notice that the input is the full resemblance matrix (in table format), plus a Treatment ID (given in quotes), and that the output (return) is in vector format. The treatment ID must match one of those from the mapping file exactly.

```
> u=unique(map[,"Treatment"])
> makeRedresem.f=function(resem_fp,group){
   resem=resem_fp
   red=resem[map[,"Treatment"]==group, map[,"Treatment"]==group]
   red.d=as.dist(red)
   return(red.d)
 }

> pbs.d=makeRedresem.f(braycurtis,"PBS")
> LGG.d=makeRedresem.f(braycurtis,"LGG")
> Lp.d=makeRedresem.f(braycurtis,"Lp")
> Lr.d=makeRedresem.f(braycurtis,"Lr")
```

In the end, we have 6 "mini" resemblance matrix - one for each treatment. We check to make sure they are in the same order. I did this by exporting the text files of the resemblances and making sure that each treatment had samples in consecutive numerical order. Note that not all of the resulting tables were in order, but the example tests below were:

```
> protest(mrs.d,LGG.d, permutations=999)

Call:
protest(X = mrs.d, Y = LGG.d, permutations = 999)

Correlation in a symmetric Procrustes rotation:  0.3507
Significance:  0.399
Based on 1000 permutations.

> protest(Lr.d,LGG.d, permutations=999)

Call:
protest(X = Lr.d, Y = LGG.d, permutations = 999)

Correlation in a symmetric Procrustes rotation:  0.5912
Significance:  0.031
Based on 1000 permutations.
```

From these two tests, we can see that the MRS control and the LGG treatment were not correlated in their dynamics, while the Lr and LGG treatments were correlated. If we use the mantel() function to test with the previously introduced Mantel test for linear matrix correlation, we get a similar result:

```
> mantel(mrs.d, LGG.d, method="pearson", permutations=999)

Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = mrs.d, ydis = LGG.d, method = "pearson", permutations = 999)

Mantel statistic r: 0.03638
      Significance: 0.36

Empirical upper confidence limits of r:
  90%   95% 97.5%   99%
0.200 0.277 0.369 0.456

Based on 999 permutations

> mantel(Lr.d, LGG.d, method="pearson", permutations=999)

Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = Lr.d, ydis = LGG.d, method = "pearson", permutations = 999)

Mantel statistic r: 0.3409
```

```
      Significance: 0.027
```

Therefore, both methods agree. This is not always the case, but if both methods agree, it is okay to only report the Mantel results, as Mantel uncovers synchrony that is apparent without first distorting (stretching, rotating) the data.

3. Taxonomic Venn analysis

We want to know which OTUs are shared among treatments, and which are unique to certain treatments. A taxonomic Venn analysis is the most straightforward and common method to do so. It is based on binary (presence/absence) data. First, we combine the instars for each treatment, and create smaller OTU table that we call "**otu.trt**", with each treatment as a column, and all OTU IDs in rows. For example, if any instar from Treatment 1 contained OTU 1, then there would be a "1" value in the Treatment 1 column in this OTU table. (You could do a similar analysis by instar, combining all treatments within each instar.).

```
> otu.trt=matrix(0,ncol=length(u),nrow=nrow(otu))
> for(i in 1:length(u)){
  temp=otu[,map$Treatment==u[i]]
  dim(temp)
  temp2=rowSums(temp)
  otu.trt[,i]=temp2
}
> row.names(otu.trt)=row.names(otu)
> colnames(otu.trt)=u
```

Then, we install the library limma and load it. We use the function **vennCounts()** to calculate all OTUs that are unique and shared among all combinations of treaments, and then write out the results to the file we name "VennCounts.txt."

```
> library(limma)
> v=vennCounts(otu.trt)
> #Write out the results of venncounts
> write.table(v, "VennCounts.txt", quote=FALSE, sep="\t")
```

Inspect the vennCounts output, which we called v. To interpret this output, the "1" says that the treatment group is included in the final count (rightmost column). This, there are 0 OTUs that are not included in any of the 6 groups, 63 OTUs that are unique to the MRS group, and 6 OTUs that are shared between the Lr and the MRS groups.

```
> v
   Eggs PBS LGG Lp Lr MRS Counts
 [1,]    0   0   0  0  0   0      0
 [2,]    0   0   0  0  0   1     63
 [3,]    0   0   0  0  1   0     61
 [4,]    0   0   0  0  1   1      6
 [5,]    0   0   0  1  0   0     90
 [6,]    0   0   0  1  0   1      6
 [7,]    0   0   0  1  1   0     12
```

It is often of interest to be able to identify exactly who those 63 unique OTUs were in the MRS treatment. We use the rdp names and the OTU IDs to create files of a list for each treatment combination. There are a total of 64 possible treatment combinations, but some of them have no OTUs. There will be one file written out for each combination that includes OTUs, but the label for each file must be referenced to the VennCounts.txt file to determine to which group the OTUs belong. Below is a custom script that I wrote to do the job, with VennOTUGroups as a vector of each OTU's assignment among the 64 possible Venn Treatment Group combinations:

```
#Make a presence/absence (binary) table
otu.trt.pa=1*(otu.trt>0)
#u=unique(map[,"Treatment"])

#This loop will output one text file per VennCounts category that has a list of all of
the OTU IDs and their RDP taxonomic affiliations that belong to each VennCount group.
#The group name is the row of the Venn Count, so you will have to compare the
VennCount file to the output to determine which group.
tmp=NULL
VennOTUGroups=vector(length=nrow(otu)

for(y in 1:nrow(v)){
  if(v[y,ncol(v)]!=0){

    for(z in 1:nrow(otu.trt.pa)){

      if(sum(1*(as.vector(otu.trt.pa[z,]) == as.vector(v[y,(1:ncol(v)-
1)])))==length(u)){
        VennOTUGroups[z]=y
        n=c(row.names(otu.trt.pa)[z],paste(rdp[z]))
        tmp=rbind(tmp,n)
      }
    }

    write.table(tmp, paste("VennCounts_Condition_",y,"_OTUList.txt",sep=""),
row.names=FALSE, sep="\t", quote=FALSE)
    print(dim(tmp))
  }
  tmp=NULL
}
```

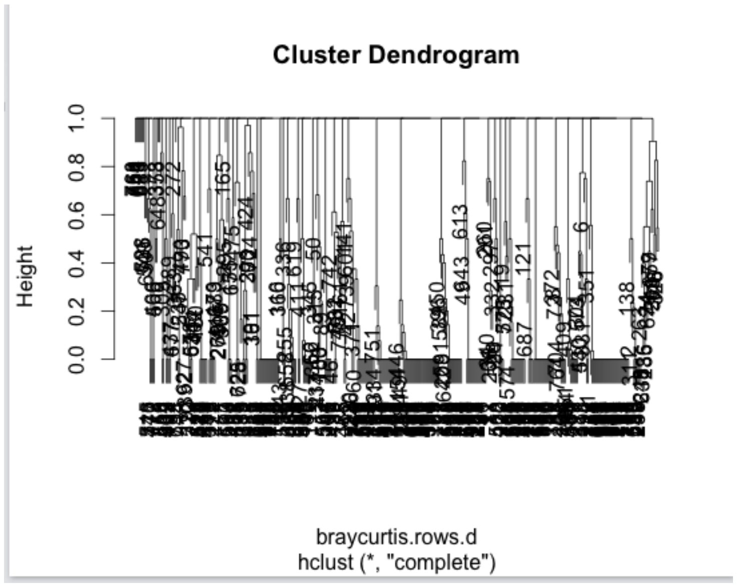
We check the output to make sure it makes sense with what we expect (and that the script is working correctly). First, we check the length of VennOTUGroups - this should be the same length as the number of OTUs that we have (389). Then, we use a logical statement, "VennOTUGroups==2" to sum the number of times that it is TRUE that VennOTUGroups has a 2. From the head of v, we know that there should be 63 OTUs that are only viewed in MRS, which is VennOTUGroups 2. We could check other groups as well. There are 16 OTUs shared across all groups (Venn Group 64), which also matches v.

```
> length(VennOTUGroups)
> sum(1*(VennOTUGroups==2))
> sum(1*(VennOTUGroups==64))
```

4. Clustering OTUs by similar dynamics

By making use of the abundance data of OTUs, we can also identify OTUs that have common dynamics. This especially is useful when considering temporal or spatial series. First, we make an OTU table standardized by rows (OTU patterns) so that prevalent and rare taxa that have similar dynamics will be clustered together. Otherwise, the abundances of OTUs will drive the patterns, and not their dynamics. Then, we use the **hclust()** function (hierarchical clustering) based on Bray-Curtis dissimilarities calculated *between all pairs of OTU*s (rather than all pairs of samples/communities, as we did previously). The resulting dendrogram is large (as many nodes as OTUs), but informative.

```
> rSums=rowSums(otu)
> otu.relrows=otu/rSums
> braycurtis.rows.d=vegdist(otu.relrows, method="bray")
> otu.cluster=hclust(braycurtis.rows.d, method="complete")
> plot(otu.cluster)
```



We can also label nodes by Venn Groups, or any other categorical value, like rdp names (abbreviated using the **make.cepnames()** function from vegan)

```
> plot(otu.cluster,labels=VennOTUGroups)
> rdp2=make.cepnames(rdp)
> plot(otu.cluster, labels=rdp2)
```

Explore these plots in detail on your own time, and also explore the related functions to hclust given at the bottom of the R help files, under "See Also".

5. Species abundance and species occurrence distributions

The vegan package has a few options for exploring and plotting species abundance distributions. The most common is called using the **radfit()** function (*aka* rank abundance distribution). We apply this function to rSums, a vector of rowSums (OTU occurrences) calculated from the OTU table. The algorithm fits a few different but commonly used SAD models so that you can compare them.

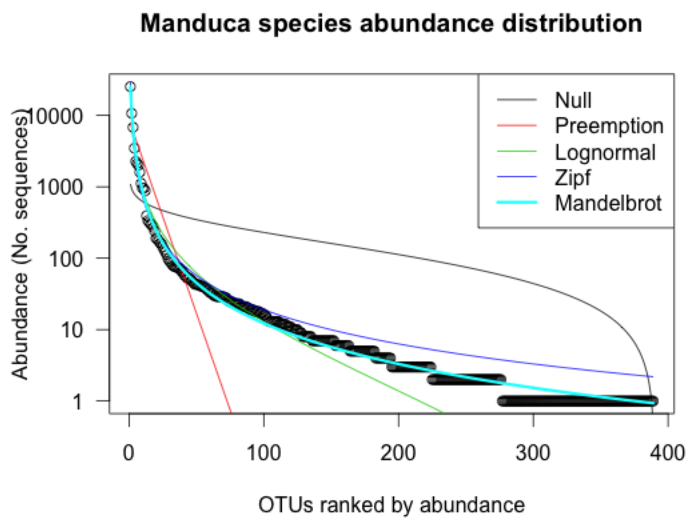
```
> r.sad=radfit(rSums)
> r.sad

RAD models, family poisson
No. of species 389, total abundance 65436

           par1     par2    par3     Deviance  AIC       BIC
Null                                 246011.46 247438.38 247438.38
Preemption  0.11723                   80071.71  81500.63  81504.59
Lognormal   0.42911  3.2503            3611.64   5042.55   5050.48
Zipf        0.44257 -1.5915            3055.97   4486.88   4494.81
Mandelbrot  1.2284  -1.9046  0.84008    886.59   2319.50   2331.39
```

The output from radfit provides some metrics for evaluating the fit of the model, as well as a built-in plotting algorithm. Here we can use either the Akaike (AIC) or the Bayesian (BIC) Information Criterion to evaluate the fit of the model- in both cases, a lower number is a better fit. In actuality, none of these models are awesome fits, but Mandelbrot is best. We can also see from the plot that the Mandelbrot function fits best.

```
>  plot(r.sad, las=1,main = "Manduca species abundance distribution", ylab="Abundance
(No. sequences)", xlab="OTUs ranked by abundance")
```



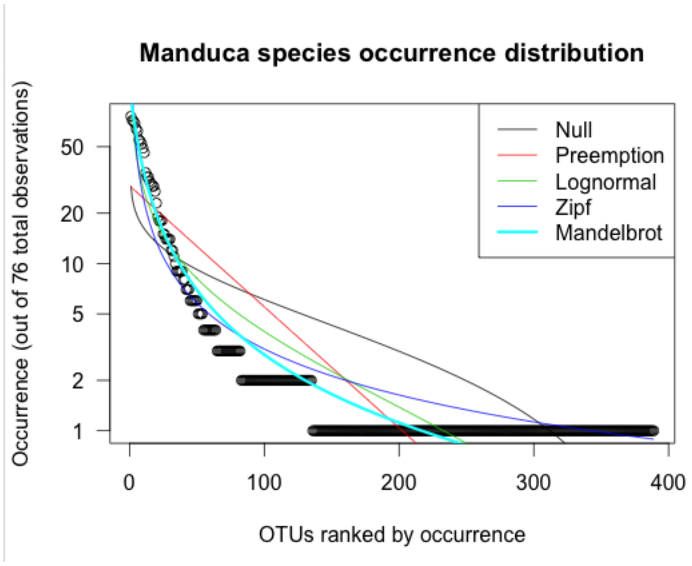
We use the same strategy for species occurrence distributions, but as input we use the total number of occurrences per OTU (out of 76 total observations) instead of the total OTU abundances. From this, there are a few models that are equally acceptable:

```
> otu.pa=1*(otu>0)
> rSums.pa=rowSums(otu.pa)
> r.sod=radfit(rSums.pa)
> r.sod

RAD models, family poisson
No. of species 389, total abundance 1736

           par1      par2     par3    Deviance AIC     BIC
Null                                  1342.28  2326.35 2326.35
Preemption  0.016538                  1127.46  2113.53 2117.49
Lognormal   0.37367   1.5129           453.30  1441.37 1449.30
Zipf        0.11967  -0.91434          393.53  1381.60 1389.53
Mandelbrot  1.3223   -1.4293   7.8843  112.64  1102.72 1114.61

> plot(r.sod, las=1,ylab="Occurrence (out of 76 total observations)", xlab="OTUs
ranked by occurrence", main="Manduca species occurrence distribution")
```



6. MultiCOLA

I've written a custom script to perform MultiCOLA at different species cut-offs, as per Gobet et al. 2010. The authors of the work also provide some R scripts that you could use instead. I used Mantel tests to compare the "full" dataset with each "reduced" dataset, but you could also use PROTEST. The output of the custom function is a table of cut-offs, number of remaining OTUs, and the Mantel correlation R and p-value for significance.

```
> MantelMultiCOLA.f=function(otu_fp){
  #Step 1.  Read in full dataset:
  otu2=otu_fp
  library(vegan)

  cutoff=c(1.00,0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10, 0.05, 0.025,
0.01, 0.005, 0.001)

  m.out=NULL

  otu.pa=1*(otu2>0)

  r=rowSums(otu2)

  #Create a vector of indexes of the ranked OTUs
  r2=sort(r,decreasing=TRUE, index=TRUE)
  r2$ix
  r2$x

  for(j in 1:length(cutoff)){
    print(cutoff[j])

    no.keep=ceiling(cutoff[j]*nrow(otu2))
    otu.index.keep=r2$ix[1:no.keep]
    otu.keep=otu2[otu.index.keep,]
    print(head(otu.keep))


    #Write out otu tables at each cutoff
    #write.table(otu.keep, paste(u[i],"_",cutoff[j],"_otu.txt", sep=""),sep="\t",
quote=FALSE)

    all.dist=vegdist(t(otu2), method="bray")
    subset.dist=vegdist(t(otu.keep),method="bray")

    m1=mantel(all.dist,subset.dist, method="pearson",permutations=999)


    m=c(paste(cutoff[j]),m1$statistic,m1$signif,dim(otu.keep)[1])
    m.out=rbind(m.out,m)
  }

  colnames(m.out)=c("Cutoff", "AllvSubsetPearsonR", "AllvSubset_pvalue",
"NoOTUsSubset")

  #write.table(m.out, "MantelMultiCOLA.txt", sep="\t", quote=FALSE, row.names=FALSE)
  return(m.out)
}
```

Run the function, and name the output "MultiCOLA.test"

```
> MultiCOLA.test=MantelMultiCOLA.f(otu)
```

Inspect the output, MultiCOLA.test. The first cut-off is not a cut-off at all, as it contains 100% of the most prevalent OTUs. As expected, this "reduced" table is correlated with a strength of R=1 (perfect correlation) and a significance of p = < 0.001. The final cutoff, the top 0.001% most prevalent OTUs included, contains only 1 OTU, but is (amazingly) still correlated with a firm R= 0.89 and p < 0.001.

```
> MultiCOLA.test

  Cutoff  AllvSubsetPearsonR  AllvSubset_pvalue NoOTUsSubset
m "1"     "1"                 "0.001"           "389"
m "0.9"   "0.999998762445507" "0.001"           "351"
m "0.8"   "0.999997479336161" "0.001"           "312"
m "0.7"   "0.999993644544533" "0.001"           "273"
m "0.6"   "0.999984756488421" "0.001"           "234"
m "0.5"   "0.999964182731846" "0.001"           "195"
m "0.4"   "0.999900942304155" "0.001"           "156"
m "0.3"   "0.999748338426395" "0.001"           "117"
m "0.2"   "0.99932881215884"  "0.001"           "78"
m "0.1"   "0.9971398024914"   "0.001"           "39"
m "0.05"  "0.993506180164891" "0.001"           "20"
m "0.025" "0.987266965435431" "0.001"           "10"
m "0.01"  "0.970160900902557" "0.001"           "4"
m "0.005" "0.95021814945195"  "0.001"           "2"
m "0.001" "0.888037161574205" "0.001"           "1"
```

From this, we see that there is one OTU that is driving the dynamics. We can deduce this because the most reduced dataset, including only 1 taxon comprising the upper 0.001 percent abundance of all taxa, is still highly correlated (0.88) to the original MultiCOLA.test. Who is the lone taxon that is so powerful?? We check the rdp vector to find it is an Enterobacteriaceae. We could BLAST the sequence of OTU 248 (remember, the sequence is in the **rep_set.fna** file from the QIIME analysis) to see if we could get an improved identification.

```
> rdp["248"]

248
Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriales; Enterobacteriaceae;
Escherichia/Shigella
145 Levels: Bacteria ... Unclassified
```