Let’s try some real data in R!

This assignment is not radically different than the first tour in R that you did (using that tiny matrix). The data we will use for this is much larger (it has millions of cells in the data table), and can’t easily be opened, navigated, and/or manipulated.

The data

The data are a **subsampled** (we will get to this soon, in mothur) .shared file from work done with Earlham students - we sampled soil from 20 local agricultural fields (five collections per field, pooled for analysis). All odd numbered fields were planted in corn, while all evens in soy. Fields numbered 1-10 were categorized by their farmers as “no till” (i.e., no or minimal soil disturbance...roots and the rhizosphere stays intact...and perhaps also the microbial communities), and 11-20 were conventionally tilled.

The data file is on Moodle: Phylo2012subsampled.shared

**This is a table of phylotypes (not binned OTUs). Briefly, the phylotypes are the groupings of sequences based on their taxonomy while OTUs are binned solely based on their sequence similarity (i.e., taxonomy independent). So, there are fewer phylotypes than OTUs when you compare the mothur outputs...we are using phylotypes here because it is a smaller file (the OTU file takes ~20min to load into R).**

This file has not been modified, but will need to be as the output from Mothur isn’t quite what we want for Vegan. Here is how I do it, but you can do it your own way (note, you might need to look up, google?, how to do some of these steps).

1. import the file using read.table with a header, but leave out row names and leave out the as.matrix...we will add these components later.
2. now label the row names using “row.names(filename) = filename$Group” (Group is one of the first columns of the file)
3. change the file format into a matrix (this might not actually be necessary, but…) and get rid of the first 3 columns in the file (these are not part of the data)
   1. I do this by renaming the file… “newfile = as.matrix(oldfile[,-c(1,2,3)])
      1. The -c is combining and eliminating in one fell swoop

IMPORTANT NOTE: be careful with looking at this! This file is very long on one matrix dimension, and so even using “head” can waste a lot of space on your R work surface. If you want to check things out, you can just “head” on the short axis. Also note, you will want to check out the file set-up and dimensions, and make sure your row names are displaying correctly...but only read these out along the short axis of the matrix.

Go through a similar work-flow as you did previously using the tiny matrix.

That is, 1) make a distance matrix, 2) do a principal coordinates analysis (that was using the function “cmdscale”). See what stuff looks like! Note, because you are using a sub-sampled file it is not necessary to calculate relative abundance (it will be redundant to the process of sub-sampling). This means that you can make the distance matrix using the matrix you made above. To make sure that the subsampling has worked, you can look at the sums for the samples (make sure you choose appropriately between column or row before summing).

IMPORTANT NOTE: for some reason the last column in the matrix is labeled X and has no information (don’t believe me, go ahead and look)...this can be avoided if you don’t specify sep=”\t” when you import the table. Vegan does not like blanks in a matrix, and neither do I. So, you can tell vegdist to ignore any NAs (which is what that column is filled with). To do this, set na.rm = TRUE in your vegdist command. Alternatively, you can get rid of that last column.

**Save as a .pdf (for credit, to turn in later) the nicest graph you made with this data, up to this point! To save you can just “save as” from the upper menu when the graph screen is selected.**

Making nicer graphs, part 1

Why don’t you do some color coding of points? You can use the same syntax for the “pcoa” plot you did, but add color quite easily. Here is how:

1. use the same basic syntax as you did before, such as:
   1. plot(xaxis, yaxis, cex=0)
   2. next, add a new “variable” (really, an object) using the “c” (combine) command.
   3. CvsS2012OTU=c("darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod", "darkgreen", "darkgoldenrod")
      1. in the above, the fields are labeled by color codes (from an R color library) and the codes match the sample order based on whether the field was corn or soy (odd or even in number...**check your sample order to be sure this is correct!**)
   4. next replot adding col…
      1. plot(xaxis, yaxis, cex=0, col=CvsS2012OTU)
   5. repeat this, but now make an object that is specific to whether the fields were tilled or not. You can do this with the point shape...instead of “col” this is “pch”. Google what the codes are for pch (or find a link in my R shortcuts sheet).
   6. Now combine! Color code by crop type and use different symbols (pch=XXX) for till type. Google what the codes are for pch.
   7. Feeling adventurous? Why not add the field number to the plot as you did in the first R exercise. You can make a new variable with the field numbers as the data and then use the “text” to plot those numbers at the coordinates.

**Save your best graph from this section as a .pdf and save it for handing in later. You will need to combine several graph pdfs into a single pdf to hand in.**

Ok, decent graphs. Now let’s switch to ggplot2 and make even nicer graphs!

*Making nicer graphs, part 2 (ggplot2)*

We are going to make a simple plot using a nicer graphics library that has lots of bells and whistles, should you choose to use them. It is also very versatile and easy to use for multiple types of plotting (we will make a mean +/- error plot in a sec).

Ok, so let’s assume you called your distance matrix df.dist and your principal coordinate results as df.pcoa. The output of this is kinda lame, so let’s first give names to those unnamed variables (currently they are, [,1] and [,2].

>colnames(df.pcoa)=c(“X”, “Y”) #this just renames those two columns X and Y...easy.

Next load (and/or install) the ggplot2 library.

>library(ggplot2)

Unforunately, ggplot doesn’t know what to do with a matrix, and that is what your df.pcoa is. So, let’s turn it into a data frame using:

>df.pcoa = as.data.frame(df.pcoa)

Now let’s build a graph. The first part is:

>ggplot(df.pcoa, aes(x = X, y = Y)) #btw, aes is for “aesthetics”

...bummer, this produces a graph, but with nothing, this is because we didn’t tell ggplot what to plot. Let’s try this, then.

>ggplot(df.pcoa, aes(x = X, y = Y))+geom\_point()

Better. Now we have points. Now let’s color code them as we did before. You can just use the same objects you made before, such as “CvsS2012” for whether they were corn or soy fields.

>ggplot(df.pcoa, aes(x = X, y = Y, color = CvsS2012))+geom\_point()

This color codes them, but not like we wanted. This is because ggplot and “plot” take their arguments in different ways. ggplot wants you to specifically tell it what colors you will be using as an argument.

>ggplot(df.pcoa, aes(x = X, y = Y, color=CvsS2012OTU)) + geom\_point() + scale\_color\_manual(values=c("darkgoldenrod", "darkgreen"))

...this is actually a bit annoying because our legend is listing “darkgoldenrod” as something in our plot. This is because we used that old object (CvsS2012OTU) as describing fields that were corn vs. soy. Yes, if we want it to list Corn and Soy as items in our data we need to actually make a new column with those names. Try this, change the above ggplot call, and see if it worked.

**Save your final graph as a pdf and add this to your same graph made with “plot”.**

...we can keep messing with this and adjust the size, shape, etc. of our points, but I will leave you to check out some of the bells and whistles of ggplot. If you google “ggplot cookbook for R” you will get a lot of helpful tutorials.

Let’s test a hypothesis. (Formalizing what you can see in the ordination)

There are several oft used methods for testing hypotheses about whether different groups of points are separate. That is, do the groups of points (such as in ordination space) have more between group difference than within group difference than would be predicted by chance? All of these tests are permutational, and it is typical to do 1000 permutations - that is, if you randomly sample all points in your data do they tend to cluster into groups or not...is your observed data different than random?

The following test is coded within the Vegan package. I suggest you make your grouping as a separate object (for ease) as we did when we made the colors in the graphics section above; for example, make an object that has “Y” or “N” for each field depending on whether it was tilled or not..

* PermANOVA (permutational analysis of variance)
  + this is quite similar, in principle, to the ANOVA that many of us know and love...but it is permutational.
  + the function is called adonis...why, I don’t know.
  + also note that this doesn’t want a distance matrix either, it does that as part of the function..
  + adonis(data~groupingobject, method=”bray”, permutations=1000).
    - In the above, substitute “data” with the name of your data object, which is your modified .shared file from above. Substitute “groupingobject” with the object you made that codes corn/soy or till/notill.

**Run this test on either till vs. no-till fields, or corn vs. soy fields (base your decision on which comparison to make on your previous ordination...which groups look different?).**

**Save the results of these tests to a text file for later reference. Use the capture.output command to save the output to a file.**

Mothur can give you much more info.

Go to the MiSeq SOP and scroll down to “OTU Based Analyses”, and further down to “Alpha Diversity”. These are all fun things that can be done within Mothur (some can also easily be done in vegan...and we already did some of these).

If you were to run the following command (substituting the file name in the SOP for your .shared) you will get a summary table of some basic community information from your .shared file. First, you put in the least number of reads for all of your samples under “subsample”. This is because you are comparing the samples at the lowest common denominator - i.e., if one field was only lightly sampled we need to compare all fields at that lowest amount of sampling else our comparison is highly biased by how many sequences we got. The “calc” is just telling mothur to calculate for us the number of OTUs (that is “sobs”) and the diversity of the sample (that is “invsimpson”). Diversity is simply the number of OTUs weighted by their relative abundances.

mothur > summary.single(shared=X.shared, calc=sobs-invsimpson, subsample = XXX)

This will give us a file with observed number of OTUs, and an estimate of diversity (#OTUs weighted by relative abundance...we are asking for Inverse Simpson’s index...the same that we use in EcoBio).

The file, “2012phylotype.groups.summary” on Moodle is such a file for the 2012 fields you have been working with. Note, there are a few other bits in this file, but we’ll ignore these for now. *Read this file into R using the read.table command.*

The way to visualize these data is with a bar graph.

Install/load the library “Rmisc”. This is a hand library that can summarize data for you.

Add columns for “crop” (corn/soy) and “till” (yes/no) to the file you imported into R above. To save time, you can copy and paste the following into R.

crop = c("Corn", "Soy", "Corn", "Soy", "Corn", "Soy", "Corn", "Soy", "Corn", "Soy", "Corn", "Soy", "Soy", "Corn", "Soy", "Corn", "Soy", "Corn", "Soy", "Corn")

till = c("No", "No", "Yes","Yes","Yes","Yes","Yes","Yes","Yes","Yes","Yes","No", "Yes", "No", "No", "No", "No", "No", "No", "No")

Once done you will need to combine these new descriptors with your summary file...for convenience, I called my summary file “summary”. You can combine these using the function “cbind”.

summary2 = cbind(summary, crop, till)

Go ahead and open summary2 to make sure it looks ok.

Now we will summarize using Rmisc and the “summarySE” function. For this you simply input the continuous variables you want summarized and which variables you want them to be broken down by, in this case we are summarizing “sobs” and breaking it down by crop and till.

summarysummary=summarySE(summary2, measurevar="sobs", groupvar=c("crop", "till"))

Look at the resulting object and make sure it makes sense to you. It is simply the mean and errors around the mean; standard deviation, standard error, and 95% confidence intervals.

Now let’s graph this using ggplot2.

ggplot(summarysummary, aes(x=till, y=sobs, fill=crop)) + geom\_bar(position=position\_dodge(), stat="identity") + geom\_errorbar(aes(ymin=sobs-ci, ymax=sobs+ci), width=.2, position=position\_dodge(.9))

Let’s dissect this set of commands. The first part with the aes should be familiar. Next we asked for a bar geometric object...the position dodge is just making it so they aren’t put on top of each other, and the stat=identity is just saying we are not transforming our data, and we want the value of the bar to be the value of our data. Then we next add a geometric object that is error bars and we specify (from our summarysummary what values go into these.

**Save this graph and add it to the others as a .pdf.**

And if you thought this couldn’t get worse. Let’s run a quick analysis of variance (ANOVA) to see if these groups are different...that is, do crops differ in species richness (sobs) or do tilled/not fields differ.

ANOVA is run using the “aov” function.

>aov = aov(sobs~till, data=summary2)

> summary(aov) #this will give you the answers you seek...Pr(>F) is your “P value”.

**Compile the files you saved into a single pdf and turn them in to the “R assignment” in moodle.**

* **a basic ordination plot**
* **a nicer looking ordination plot**
* **an even nicer looking ordination plot using ggplot2**
* **a bar plot comparing sobs with 95% CI**