16S Data Exploration

Next steps of processing 16S rRNA diversity data: multivariate analyses

Brazelton Lab, February 2016

**Familiarize yourself with the workflow on the training dataset provided in /srv/data/training/amplicons/mothur/mothur\_miseq\_sop prior to following it for the first time on real data.**

Follows initial [standardized protocol of processing raw data](https://docs.google.com/document/d/1k6p_iiQG_4qPX-ro-yezGYViPZNojDy9ZbsybZfpg9E)

and begins in the same way as the [edgeR commands for finding sample-specific taxa](https://docs.google.com/document/d/1PBlwXEeLaydt4bHRFDZt5YzHrxBps82GQTbH1vt21Wo)

The following commands make use of the R package **phyloseq**:

McMurdie, P. J., & Holmes, S. (2013). phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PloS One*, *8*(4), e61217. doi:10.1371/journal.pone.0061217

McMurdie, P. J., & Holmes, S. (2014). Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS Computational Biology*, *10*(4), e1003531. doi:10.1371/journal.pcbi.1003531

# Contents

1. [Introduction: Multivariate Analysis](#_25tcmlj3xgv9)
2. [Data Preparation](#_xf9tkc1ob9xc)
3. [Unconstrained Ordination](#_g7z0swy1lugd)
   1. [Interpreting unconstrained ordination with environmental fitting](#_cugz5q38mwf)
4. [Constrained Ordination](#_pxz06z4g04a7)
5. [Classification](#_qrci7zi333qp)
6. [Interpreting the Results](#_b1potovjd490)
   1. [Why did we do all of this?](#_dy84ykd2qgq1)
   2. [How should we interpret different results from the three beta diversity indices?](#_bnvwo6etxd0e)

# Introduction: Multivariate Analysis

Ordination and classification are the two main classes of multivariate analysis employed in community ecology. Both are used for revealing patterns in species composition. Though complementary, the two methods differ in strategy employed and how the results are typically visualized. Ordination is based around the arrangement of species and/or samples along gradients, and is visualized using scatter plots. While classification involves grouping objects that are more similar to each other than they are to other objects, and is generally viewed with dendrograms (trees).

# Data Preparation

Starting Materials:

- sequence by sample table (i.e. count table from mothur)

- sample info file with samples as rows and explanatory variables as columns. Both the first and second columns should contain the sample names, with an optional header for the first column.

- file containing sequence taxonomic classifications

\*check that all files have unix linebreaks.

\* note that you can use the same phyloseq objects you may have already created before

If your taxonomy file was generated with mothur, prepare it for phyloseq with this script:

srun taxonomy2tsv.py <filename>.taxonomy

Or use this script if you are using a taxonomy file created by classify.otu in mothur:

srun otutaxonomy2tsv.py <filename>.taxonomy

Or use this script if you are using a tax table created by DADA2:

taxonomy2tsv\_DADA2.py <filename>.csv

Now you have a new taxonomy file that ends in '.tsv' that you should use for the <taxonomy table> file below.

Launch R and load the required libraries:

srun --x11=first --pty R

library('phyloseq')

library('ggplot2')

theme\_set(theme\_bw())

Create the phyloseq objects, if you have not done so already:

count.table <- read.table('**<sequence-by-sample table>**', sep='\t', header=TRUE, row.names=1, check.names=FALSE)

meta.table <- read.table('**<meta-data table>**', sep=',', header=TRUE, row.names=1)

taxa.table <- read.table('**<taxonomy table>**', sep='\t', header=<BOOL>, row.names=1)

merged <- phyloseq(otu\_table(count.table, taxa\_are\_rows=TRUE), tax\_table(as.matrix(taxa.table)), sample\_data(meta.table))

merged

\* copy the output of merged into your notebook.

\* if the sequence-by-sample table was created in Mothur, use count.table <- subset(count.table, select=-c(1:1)) to delete the 'total' column from the table prior to creating the phyloseq-class object with phyloseq.

\* if the taxonomy table was created in Mothur, use header=FALSE when reading in the table.

\* use the merge\_phyloseq function instead of the phyloseq function if one or more of the objects to be merged is already a phyloseq-class object. The class of an object can be determined using class(object\_name).

# Unconstrained Ordination (Indirect Gradient Analysis)

With indirect gradient analysis methods, the important environmental gradients are inferred directly from the species compositional data. These techniques are useful in that they reduce the complexity of the data and allow for the relative importance of different gradients to be compared. However, unconstrained ordination is regarded as an exploratory approach to data analysis. Testing hypotheses on the same set of data in which the hypotheses were generated from is considered data dredging, and invalidates any conclusions drawn from them. [Here](http://ordination.okstate.edu/motivate.htm) are some suggestions on how to avoid the pitfalls of data dredging.

Calculate the dissimilarity between each pair of samples based on their shared species compositions and perform ordination on the resulting matrix:

mds.sor = ordinate(merged, method='MDS', distance='sor')

mds.bray = ordinate(merged, method='MDS', distance='bray')

mds.horn = ordinate(merged, method='MDS', distance='horn')

\* there are many different beta diversity indices to choose from for this analysis, and we will use three of them: Sorensen, Bray-Curtis, and Morisita-Horn. The result is a dissimilarity matrix which we will visualize by compressing all of the information into two dimensions using MDS/PCoA (multi-dimensional scaling/principal coordinates analysis).

\* it is possible to use other unconstrained ordination techniques in phyloseq as well. Methods include MDS, NMDS, PCA ('RDA' without constraints), CA ('CCA' without constraints), and DCA. See help(ordination) for a list of all ordination methods that phyloseq supports and [this](http://ordination.okstate.edu/key.htm) site for a guideline on when a given ordination technique might be most appropriate.

Plot the ordination eigenvalues:

plot\_scree(mds.sor)

plot\_scree(mds.bray)

plot\_scree(mds.horn)

\* this will help with choosing which axes to plot later on. Axes with higher eigenvalues explain more variation (or inertia) in the data.

See what environmental variables are available in the merged object:

sample\_variables(merged)

Plot the results of the ordinations and save them to a file:

png(filename='<project>.MDS\_sor.png')

plot\_ordination(merged, mds.sor, type='samples', color='variable1', shape='variable2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size = 4) + stat\_ellipse() + geom\_text(aes(label=<variable1>), size=4, vjust=1.5)

dev.off()

png(filename='<project>.MDS\_bray.png')

plot\_ordination(merged, mds.bray, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size = 4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

png(filename='<project>.MDS\_horn.png')

plot\_ordination(merged, mds.horn, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

\* **add actual column names in your sample info file** to the label, color, and shape variables.

\* the default behavior of plot\_ordination is to plot the first and second axes of the ordination, but any axis can be specified if the ordination method allows for it. For example, add axes=c(1, 3) as a parameter to plot\_ordination if you want to plot the first and third axes instead.

\* the numeric scale along the plot axes are not useful for interpretation for all methods except DCA, which has units of beta diversity. Instead, interpretation should be based on general trends in the data.

Repeat the procedure above, this time after converting the raw counts in the original count table into proportions, as a way of normalizing for the variation in number of sequences among samples:

merged.props = transform\_sample\_counts(merged, function(x) 100 \* x/sum(x))

mds.props.sor = ordinate(merged.props, 'MDS', distance='sor')

mds.props.bray = ordinate(merged.props, 'MDS', distance='bray')

mds.props.horn = ordinate(merged.props, 'MDS', distance='horn')

Plot the results and save them to a file:

png(filename='<project>.props.MDS\_sor.png')

plot\_ordination(merged.props, mds.props.sor, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

png(filename='<project>.props.MDS\_bray.png')

plot\_ordination(merged.props, mds.props.bray, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size = 4, vjust=1.5)

dev.off()

png(filename='<projects.props.MDS\_horn.png')

plot\_ordination(merged.props, mds.props.horn, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

\* remember to **add actual column names in your sample info file** to the label, color, and shape variables.

Repeat the procedure yet again, this time after eliminating an arbitrary number of the least abundant sequences:

topN = 10000

most\_abundant\_taxa = sort(taxa\_sums(merged.props), TRUE)[1:topN]

merged.props.topN = prune\_taxa(names(most\_abundant\_taxa), merged.props)

mds.props.topN.sor = ordinate(merged.props.topN, 'MDS', distance='sor')

mds.props.topN.bray = ordinate(merged.props.topN, 'MDS', distance='bray')

mds.props.topN.horn = ordinate(merged.props.topN, 'MDS', distance='horn')

\* change the topN variable to indicate how many of the most abundant species (in this case, species = taxa = unique sequences) you want to include in the analysis.

Plot the ordination and save the output to a file:

png(filename='<project>.props.topN.MDS\_sor.png')

plot\_ordination(merged.props.topN, mds.props.topN.sor, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

png(filename='<project>.props.topN.MDS\_bray.png')

plot\_ordination(merged.props.topN, mds.props.topN.bray, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

png(filename='<project>.props.topN.MDS\_horn.png')

plot\_ordination(merged.props.topN, mds.props.topN.horn, type='samples', color='yourcolumn1', shape='yourcolumn2') + theme\_bw() + theme(text=element\_text(size=16)) + geom\_point(size=4) + stat\_ellipse() + geom\_text(aes(label=<yourcolumn1>), size=4, vjust=1.5)

dev.off()

\* remember to **add actual column names in your sample info file** to the label, color, and shape variables.

Open a new terminal window and log into the server with sftp. Navigate to the folder where your files are located and then download all of the .png files with the MDS plots by using this command:

get \*MDS\_\*

# The End. Open the .png files with an image viewer on your local machine.

# If you want to explore how to visualize these results in more creative ways, check out the phyloseq tutorials (e.g. [distance](http://joey711.github.io/phyloseq/distance.html) and [subset\_ord\_plot](http://joey711.github.io/phyloseq/subset_ord_plot-examples.html)).

## Interpreting unconstrained ordination with environmental fitting

In unconstrained ordination, the major compositional variation of the community data is found first and the results related to observed environmental variation afterwards. The purpose of this section is to show how to overlay potentially ecologically relevant information onto ordination diagrams to aid the interpretation of patterns in the data illuminated by unconstrained dimension reduction techniques.

Launch R and load the required libraries:

srun --x11=first --pty R

library('phyloseq')

library('ggplot2')

library('vegan')

library('grid')

library('ggrepel')

Create the ordination object:

nmds.horn = ordinate(merged, method='NMDS', distance='horn')

\* the method used in this example is NMDS (non-metric multidimensional scaling). It is similar to MDS in that it is a distance-based technique (as opposed to eigenvector-based techniques such as PCA or CA), but differs in algorithm and properties. An advantage to using NMDS is that it maximizes the rank order correlation as opposed to a linear correlation, and can therefore handle non-linear species responses to a gradient.

\* if a method such as MDS was used instead of NMDS, set the variable to include only the vector attributes of the object (<ord\_object> = <ord\_object>$vector). This is not necessary for NMDS because the function called by the ordinate wrapper finds and returns only the axes with minimum stress. Use attributes(object) to list the associated attributes of an R object.

\* the number of dimensions must be pre-specified prior to ordination with NMDS, as the solution changes depending on number of axes viewed. This can be done by providing ordinate with the parameter k=<dimensions> (default: k=2). The measure of ‘stress’ (the mismatch between the rank order of distances in the data and the rank order of distances in the ordination) in the solution may be significantly reduced by increasing the number of dimensions, but a solution with more than 2 dimensions may be difficult to interpret visually.

Fit the environmental variables onto the ordination:

nmds.fit = envfit(nmds.horn ~ variable1 + variable2 + ..., as.data.frame(meta.table), perm=1000)

nmds.fit

\* explanatory variables need not be continuous; they can be continuous, ratio scale, or even nominal. The envfit function returns vectors for continuous variables and factors for categoricals. Dichotomous variables - or nominal variables with only two categories (male or female, yes or no, etc.) - will be treated as continuous if first converted into binomials (0, 1). This should be done whenever possible.

\* it is possible to use either all of the variables in the metadata table or just some of them. To use all variables, replace the right-hand side of the formula (delimited by ~) with a period (nmds.horn ~ .).

\* specific axes of the ordination can be chosen by providing envfit with the parameter choices=c(<axis1>, <axis2>).

\* the parameter perm of envfit instructs the algorithm on how many random permutations of the data to perform for each explanatory variable. The resultant significance value is reported as Pr(>r). It is generally only going to be useful to fit the variables shown to be significant.

Get the values of the fitted vectors, scaled by correlation coefficient:

nmds.fit.vectors = scores(nmds.fit, display=c('vectors'))

multiplier = ordiArrowMul(nmds.fit.vectors, fill=<fraction>)

nmds.fit.vectors = data.frame(labels=rownames(nmds.fit.vectors), nmds.fit.vectors\*multiplier)

\* scaled values are extracted with the display=c('vectors') parameter of scores. Scaling is done by multiplying the vector axes by the square root of the coefficient of determination. This is done so that 'weak' predictors have shorter arrows than 'strong' predictors.

\* the function ordiArrowMul allows the size of the arrow graphic to be adjusted using a constant multiplier. The parameter fill determines the proportion of the plot that the span of the arrows will fill. A value of 0.8 means that the arrows will span 80% of the plot. This is for aesthetic purposes and does not affect the relative strengths of the vectors.

\* to extract only the significant vectors (with the significance level set at 5%), replace scores(nmds.fit, display=c('vectors')) with scores(nmds.fit, display=c('vectors'))[rownames(scores(nmds.fit, display=c('vectors')))[nmds.fit$vectors$pvals<=0.05], ].

(Optional) Generate mean site values, grouped by factor of interest:

nmds.mean = aggregate(scores(nmds.horn), by=list(labels=meta.table$<factor>), mean)

\* specific axes of the ordination can be chosen by providing scores with the parameter choices=c(<axis1>, <axis2>).

Create the plot:

p.nmds = plot\_ordination(merged, nmds.horn, type='split', color='<factor1>', shape='<factor2>')

p.nmds

\* if using methods where information about species is collapsed during ordination, such as MDS/PCoA, the parameter type must be set to type='samples'. For other methods, type can be set to split, samples, species, or biplot.

Add vector arrows to the plot:

p.nmds = p.nmds + geom\_segment(data=nmds.fit.vectors, aes(x=0, y=0, xend=NMDS1, yend=NMDS2), inherit.aes=FALSE, arrow=arrow(length=unit(0.2, 'cm')), color='#808080', alpha=0.8)

p.nmds

\* explanatory variables are represented by arrows and samples and/or species are represented as points.

\* the arrows point to the direction of the gradient (i.e. the direction of most rapid change in the explanatory variable) and the relative lengths of the arrows are proportional to the correlation between variable and ordination (i.e. the strength of the gradient).

\* substitute NMDS1 and NMDS2 with the appropriate axes names. The names of the axes can be found with colnames(nmds.fit.vectors).

(Optional) Add factor ellipses:

p.nmds = p.nmds + stat\_ellipse(linetype=<integer>, type='<distribution>', level=<conf\_level>, alpha=0.4)

p.nmds

\* ellipses are generated from the factors used in plot\_ordination.

(Optional) Add centroid and arrow labels:

p.nmds = p.nmds + geom\_label\_repel(data=rbind(nmds.fit.vectors, nmds.mean), aes(NMDS1, NMDS2, label=labels), inherit.aes=FALSE, max.iter=5000, color='#4C005C', alpha=0.4)

p.nmds

\* substitute NMDS1 and NMDS2 with the appropriate axes names. The names of the axes can be found with colnames(nmds.labels).

\* to leave out centroid labels, substitute data=rbind(nmds.fit.vectors, nmds.mean) with data=nmds.fit.vectors.

# Constrained Ordination (Direct Gradient Analysis)

In multivariate direct gradient analysis (DGA), the important gradients are already “known” beforehand and the species composition is directly related to these measured variables. Instead of acting as an interpretive tool *post hoc*, the choice of environmental variables in DGA actively influences the outcome of the constrained ordination, and results in gradients constrained to be a function of the measured variables. One of the advantages of DGA is that it can be used to test statistical hypotheses, unlike indirect methods, which are regarded as purely exploratory analyses.

The purpose of this section is to demonstrate the use of constrained correspondence analysis (CCA). Like correspondence analysis (CA), CCA maximizes the correlation between species and sample scores, only constrained to be the best linear combination of the explanatory variables. In this way it is related to both CA and multiple regression.

Launch R and load the required libraries:

srun --x11=first --pty R

library('phyloseq')

library('ggplot2')

library('vegan')

library('grid')

library('ggrepel')

Create the ordination object:

ord.cca = ordinate(merged, method='CCA', formula=merged ~ variable1 + variable2 + ...)

ord.cca

\* it is possible to use either all of the variables in the metadata table or just some of them. To use all variables, replace the constraining model, or right-hand side of the formula delimited by a tilde, with a period (nmds.horn ~ .). Doing this is inadvisable, however, as the final configuration of points in the ordination is directly dependant on the importance of the explanatory variables in determining the makeup of the community. As the number of constraints increases, the ordination becomes more similar to the unconstrained one. In other words it is possible to overfit the data, just like in multiple regression. The authors of vegan recommend that no more than five constraints be used at a time.

\* as with environmental fitting in unconstrained ordination, explanatory variables can be either continuous or nominal. However, it is recommended that nominal variable be converted to continuous ones whenever possible.

\* it is possible with the formula interface of ordinate to partial out the influence of certain factors (conditioned factors) prior to ordination. This is done by including the term Condition(variable1 + variable2 + ...) in the constraining model. Conditioned factors are termed covariates and the result is called a partial ordination. A good overview of when partial ordination is appropriate can be found [here](http://ordination.okstate.edu/partial.htm).

\* if the eigenvalues for the constrained axes are not less than the eigenvalues for the unconstrained axes, then the constraints used had no effect.

Test the significance of the constraints:

anova(ord.cca)

\* significance is assessed through the use of permutation tests. Results are said to be significant when the observed result is better than a defined percentage (significance level) of the random models.

\* the default is to analyse all constraints simultaneously. To analyse the terms separately, include the parameter by=term in the function. The number of permutations of the data to perform can be specified by adding the parameter permutations=<int>. It is not necessary to specify the number of permutations when analysing constraints together. Analyzing terms separately may require corrections for multiple testing.

Extract the scores from the ordination object:

ord.cca.scrs = scores(ord.cca, display=c('bp', 'sp', 'wa', 'cn', 'lc'))

Extract the values for the vectors:

multiplier = ordiArrowMul(ord.cca.scrs$biplot, fill=<fraction>)

ord.cca.vectors = data.frame(labels=rownames(ord.cca.scrs$biplot), ord.cca.scrs$biplot\*multiplier)

Create the plot:

p.cca = plot\_ordination(merged, ord.cca, type='split', color='<factor1>', shape='<factor2>')

p.cca

Add arrows to the plot:

p.cca = p.cca + geom\_segment(data=ord.cca.vectors, aes(x=0, y=0, xend=CCA1, yend=CCA2), inherit.aes=FALSE, arrow=arrow(length=unit(0.2, 'cm')), color='#808080', alpha=0.8)

p.cca

\* explanatory variables are represented by arrows and samples and/or species are represented as points.

\* the arrows point to the direction of the gradient (i.e. the direction of most rapid change in the explanatory variable) and the relative lengths of the arrows are proportional to the correlation between variable and ordination (i.e. the strength of the gradient).

Add factor ellipses:

p.cca = p.cca + stat\_ellipse(linetype=<integer>, type='<distribution>', level=<conf\_level>, alpha=0.4)

p.cca

\* ellipses are generated from the factors used in plot\_ordination.

Add factor and vector labels:

p.cca = p.cca + geom\_label\_repel(data=ord.cca.vectors, aes(CCA1, CCA2, label=labels), inherit.aes=FALSE, max.iter=5000, color='#4C005C', alpha=0.4)

p.cca

# Classification (clustering)

Cluster analysis is divided into two main classes - hierarchical and fixed-cluster (partitioning) - based on whether the final number of clusters is specified *a priori* or not. This section shows how to use agglomerative hierarchical clustering. In this method, sample points start out as individuals and are iteratively grouped based on similarity until all points are contained in a final cluster. Cluster analysis is the technique also used to form OTUs.

Note that grouping sites as a nested hierarchy may not be the best way to visualize similarities in species composition between sites, as [communities tend to be more continuous in nature](http://ordination.okstate.edu/overview.htm#Figure_2). One of the more popular alternatives to hierarchical clustering is k-means clustering. K-means clustering requires that the number of partitions be specified beforehand, however, so its application to poorly characterized communities may be limited. The function kmeans can be used to perform k-means clustering in R.

Launch R and load the required libraries:

srun --x11=first --pty R

library('phyloseq')

library('vegan')

library('ggplot2')

library('ggdendro')

Compute the sample dissimilarity matrix:

dist.horn = distance(merged, method='horn', type='samples')

\* the Morisita-Horn index is used here. For compatibility with the standardized workflow, repeat this workflow using the Bray-Curtis (“bray”) and the Sorenson (“sor”) indices.

Cluster samples using a hierarchical classification method:

dist.clust = hclust(dist.horn, method='average')

\* the method parameter specifies what linkage criterion to use to determine if an object belongs to a given cluster. The mothur command tree.shared clusters groups using the UPGMA algorithm, or average-linkage clustering (method='average'), and the default clustering strategy used in hclust is average-linkage. It is also the strategy recommended by the authors of the R package vegan.

Compute a cophenetic correlation value for the hierarchical cluster:

dist.coph = cophenetic(dist.clust)

mantel(dist.horn, dist.coph, permutations=1000)

\* the cophenetic distance describes how well a hierarchical cluster analysis portrays the actual distances between sites. If the correlation between cophenetic distances and original distances is high, the dendrogram can be said to be a good representation of the data. The mantel statistic is used to assess the correlation between distance matrices, and the significance evaluated by permutations of the data.

Create the dendrogram object:

dist.dendro = dendro\_data(as.dendrogram(dist.clust), type='rectangle')

\* the parameter type determines how the tree will be displayed. The options are rectangle and triangle.

Add grouping information:

dist.dendro$labels$<VariableName> = meta.table[as.character(dist.dendro$labels$label), <col\_number>]

\* replace col\_number with the column number of the variable to group samples by. Use colnames(meta.table) to see the order of the available environmental variables.

Plot the diagram:

p.dendro = ggplot(data=dist.dendro$segment) + geom\_segment(aes(x=x, y=y, xend=xend, yend=yend)) + geom\_point(data=dist.dendro$label, aes(x=x, y=y, label=label, color=<VariableName>)) + geom\_text(data=dist.dendro$label, aes(x=x, y=y, label=label, hjust=1.1), angle=90, lineheight=10) + scale\_y\_continuous(expand=c(0, 0.2)) + xlab("Distance (beta diversity = horn)")

p.dendro

\*\*\* Note that all of the above clustering commands can be used with non-normalized, normalized ('props'), or with an arbitrary number of most abundant sequences ('topN') - just as we did above for the MDS plots.

# Interpreting the Results

## Why did we do all of this?

Which of these analysis is the 'best'? And why did we not just choose that one? The answer is that there is no such thing as 'the best' approach for all possible cases, and comparing the multiple analyses we performed above can be very informative. To explain, we should first discuss the diversity indices a bit more:

The [Sorensen](https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient) index, in its original form, considers only presence/absence data. In other words, the abundance of a species is not considered, only whether it is present or absent. Therefore, this index is said to calculate dissimilarity with binary data.

The [Bray-Curtis](https://en.wikipedia.org/wiki/Bray%E2%80%93Curtis_dissimilarity) index can be thought of as the Sorensen index extended to incorporate differences in species abundances. Therefore, if a species is present in two samples, but much more abundant in one sample compared to the other, then this will be reflected in the calculation.

The [Morisita-Horn](https://en.wikipedia.org/wiki/Morisita%27s_overlap_index) index is similar to Bray-Curtis in that it is weighted by differences in species abundances among samples, but it is independent of differences in sample sizes or diversity. For example, two samples that are nearly identical to each other except that one has additional species will have much lower (i.e. less dissimilar) scores with Morisita-Horn than with Bray-Curtis.

Mothur also provides the [Sorensen](http://mothur.org/wiki/Sorclass), [Bray-Curtis](http://mothur.org/wiki/Braycurtis), and [Morisita-Horn](http://mothur.org/wiki/Morisitahorn) indices, which you can find in the '.count\_table.summary' file that you probably already generated.

## How should we interpret different results from the three beta diversity indices?

First, let's do a sanity check and verify the following results:

* First, let's be clear that each sample is represented by a single point in the MDS plot. Samples that cluster together in the 2D space of the MDS plot are similar to each other. This means they have low dissimilarity values. When speaking and writing, it's easy to get confused with these words, but just remember that low dissimilarity = high similarity and close together on the MDS plot.
* The Sorensen results should be the same whether we converted raw counts to proportions or not because presence/absence data is not affected by this transformation (i.e. MDS\_sor.png should be equivalent to MDS\_sor\_props.png.)
* Similarly, the Morisita-Horn results should be the same whether we converted raw counts to proportions or not because it is independent of sample size. (i.e. MDS\_horn.png should be equivalent to MDS\_horn\_props.png. 'Equivalent to' means they should look identical or would look identical if you rotated and/or flipped them.)
* The Bray-Curtis results should be at least slightly affected by converting raw counts to proportions. The effect could be small or large depending on the variation of sample sizes in your dataset. If all of your samples are very similar in size (number of total sequences), then you won't see much difference between MDS\_bray.png and MDS\_bray\_props.png. One rule of thumb is that Bray-Curtis results are only meaningful after converting counts to proportions. This is usually true for our purposes, but it is important to know that this is not always the case and depends on the study design.

If you passed that sanity check, let's try to interpret the differences in results. It is hard to generalize this, but here are a few examples:

* If two samples appear to be very similar according to Sorensen but less similar according to Bray-Curtis, then these two samples probably share many species, but those species are much more abundant in one sample compared to another. You should check whether this relationship holds with both MDS\_bray and MDS\_bray\_props.
* In the converse scenario, two samples that are very similar according to Bray-Curtis but less similar with Sorensen probably share very few species, but those few species are very abundant in both samples.
* If two samples appear to be very similar according to Morisita-Horn but less similar with Bray-Curtis (after converting to proportions), then one of these samples is probably a subset (to some degree) of the other. One prediction of this interpretation is that the alpha diversity (number of species) of one of these samples is much lower than the other. One of many possible real-world interpretations of this pattern is that the lower-diversity sample could be a 'more pure' representation of the true community while the higher-diversity sample could be affected by contamination and/or mixing with other communities. This is just an example; there are many other possible interpretations as well.
* If any of the results are substantially different after removing the rarest species, keeping only the most abundant (i.e. the 'topN' results), then this indicates the previous results were being dominated by the rare species and not reflecting the patterns in the most abundant species. The Sorensen results are more likely to be strongly affected by removing rare species, while Bray-Curtis and Morisita-Horn are unlikely to be affected. Again, there is no right or wrong here, but the comparison provides valuable information about your dataset.