A primer for compositional analysis of microbiome data

gg

2016-03-09

To run this file: # Rscript -e "rmarkdown::render('CoDaseq.Rmd')"

you will need the rmarkdown and knitr packages installed from CRAN.

Exploring multivariate data

Multivariate data are complex, and not easily envisioned. For this reason we use data reduction techniques to examine the relationships between the samples and the data. These techniques require us to understand two fundamental issues: distance and variance.

Distance is simply what we imagine it is:

how far it is from point A to point B. But there are many ways to measure it. There are two main methods of distance measurement, city block and Euclidian. City Block is the distance around the two sides of a right angle triangle (called an L1 measure). Euclidian distance is the hypotenuse of that triangle (L2 measure). The Bray-Curtis dissimilarity is is a City Block measure normalized so that it is 0 if the samples are identical and 1 if the samples are completely different.

Distance is the building block for clustering, where the idea is to show the relationships among the samples. The choice of a distance measure is controversial, ** "researchers' choices of a proximity measure, as well as possible transformations of the initial data, are usually governed by their prior education, the school of thought they happen to follow, and the literature they are emulating, rather than an insight into the properties of the measure itself." **

(Michael Greenacre, Author Multivariate Analysis of Ecological Data)

Variance is also what we think it is:

dispersion around some centre. But, in a multivarite dataset, the dispersion is not easy to visualize because we have as many dimensions as we have datapoints. Thus we have Principal Component Analysis (PCA), that identifies and displays the axes of variance in the data. An approachable introduction to this (best I've seen anyway) is at: https://www.ling.ohio-state.edu/~kbaker/pubs/Singular_Value_Decomposition_Tutorial.pdf. You should read and understand this as best you can.

Distance and variance are not represented well when we are dealing with proportional (or constant sum) data. Thus, we need to transform these data to ensure that we can get reasonable, reproducible estimates of these values. You should read one of my polemics on this, but be prepared to spend some time (try https://github.com/ggloor/compositions/tree/master/background_reading Gloor et al. Workshop_report_R1.docx).

Now for some code. I have put up a dataset composed of a subset of the Human Microbiome project oral 16S rRNA gene sequencing survey dataset on github. The first chunk is just a lot of plumbing. It is here for reference. To use this you will need to get the table tongue_vs_cheek.txt from github and put it in the same directory as the .Rmd document.

```r
# ---- setup ----

# we need these libraries for the colored biplot, and for the 0 replacement function
```

```r
library(compositions)
library(zCompositions)
source("chunk/codaSeq_functions.R") # commonly used functions

# read the table, with column and row names, columns tab delimited
# samples are by column, variables are by row
d.bf.1 <- read.table("tongue_vs_cheek.txt", header=T, row.names=1, sep="\t")

# move taxonomy info to a vector
tax.0 <- data.frame(d.bf.1$tax)
rownames(tax.0) <- rownames(d.bf.1)

# remove the taxonomy column
d.bf.1$tax <- NULL

# keep only those samples with > min.reads
min.reads <- 5000
# keep only OTUs with an abundance of at least 0.001
min.prop = 0.01
# keep OTUs that are found in at least 30% of samples
cutoff = .3

# using function in codaMB_functions.R
d.subset <- codaSeq.filter(d.bf.1, min.reads=min.reads, min.fraction=cutoff, min.prop=min.prop,
    samples.by.row=FALSE)

tax.subset <- as.data.frame(tax.0[rownames(tax.0) %in% rownames(d.subset),])
rownames(tax.subset) <- rownames(tax.0)[rownames(tax.0) %in% rownames(d.subset)]

# basically I am generating a shortened name for each OTU for display purposes
# you could do this by hand if you don't know grep
# two ways to substitute row and column labels
# gsub replaces any number of characters denoted by the "." that are followed by
# two underscores with nothing from the list of taxa names in the reduced dataset
short_tax <- gsub(".+__", "", tax.subset[,1])

# com is being assigned a vector of values where the "T" is replicated the number of times
# that we observe td_ in the column names of the reduced data subset
com <- c(rep("T", length(grep("td_.", colnames(d.subset))) ),
    rep("B", length(grep("bm_.", colnames(d.subset))) ))
```

Herein ends the data munging. I first reduced the dataset to those samples with at least 5000 reads, and then removed all OTUs that were present at an abundance less than 0.01 in all samples and that were represented in less than 0.3 as a proportion of the samples. In the end we are left with 220 OTUs and 147 samples. This is just for convenience because it removes samples and OTUs with little data. Such samples and OTUs add little to the analysis because their values are estimated poorly.

You will need to alter these commands to get your dataset into shape for the analysis that follows. The important thing to remember is to try a number of different subsetting methods and cutoffs to make sure that you are not looking at an artefact of the exact way that you prepared your data. This is an important principle in compositional data analysis because these data are very sensitive to subsetting and cutoffs.

```
# replace 0 values with an estimate of the probability that the zero is not 0
# this is from the zCompositions package
##### IMPORTANT
# at this point samples are by row, and variables are by column
#####
# we can use the GBM or CZM methods. Function from zCompositions
d.n0 <- cmultRepl(t(d.subset),  label=0, method="CZM", output="counts")
```

```
## No. corrected values:  4278
```

```
# convert everything to log-ratios. Function from codaSeq
# equivalent to log(x/gx) for every value where gx is the geomtric mean of the vector X
# I prefer to use log to the base 2, although purists disagree.
# For the purposes of these plots and analyses there is no
# effect of the base on the results.
d.n0.clr <- codaSeq.clr(d.n0, samples.by.row=TRUE)

# replace row and column names for display
# this only works because we now have a matrix, not a dataframe
colnames(d.n0.clr) <-  paste(short_tax,colnames(d.n0.clr), sep=":")
rownames(d.n0.clr) <- com
```

At this point we have our data in a matrix, with rownames being samples and column names being taxa names

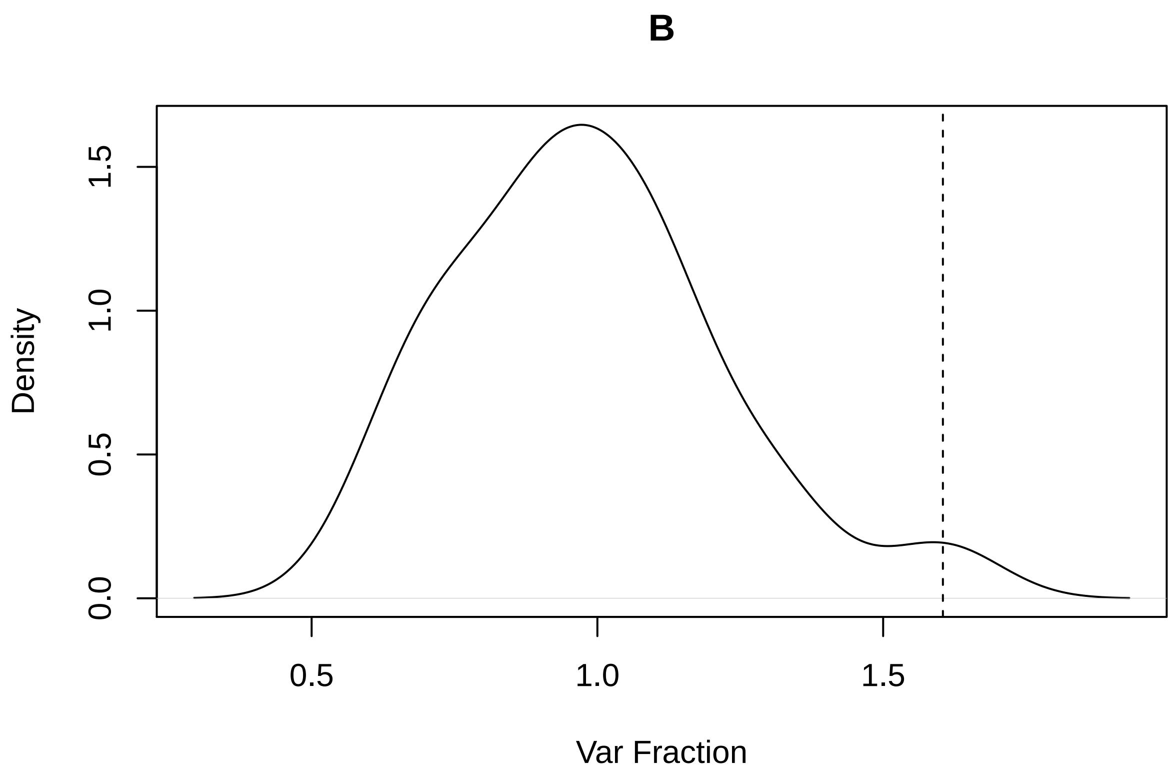
Within group consistency

The first thing we should do is to determine if our samples are consistent within each group. Note that this approach is only sensible if you have naturally defined groups. This is not an exploratory test.

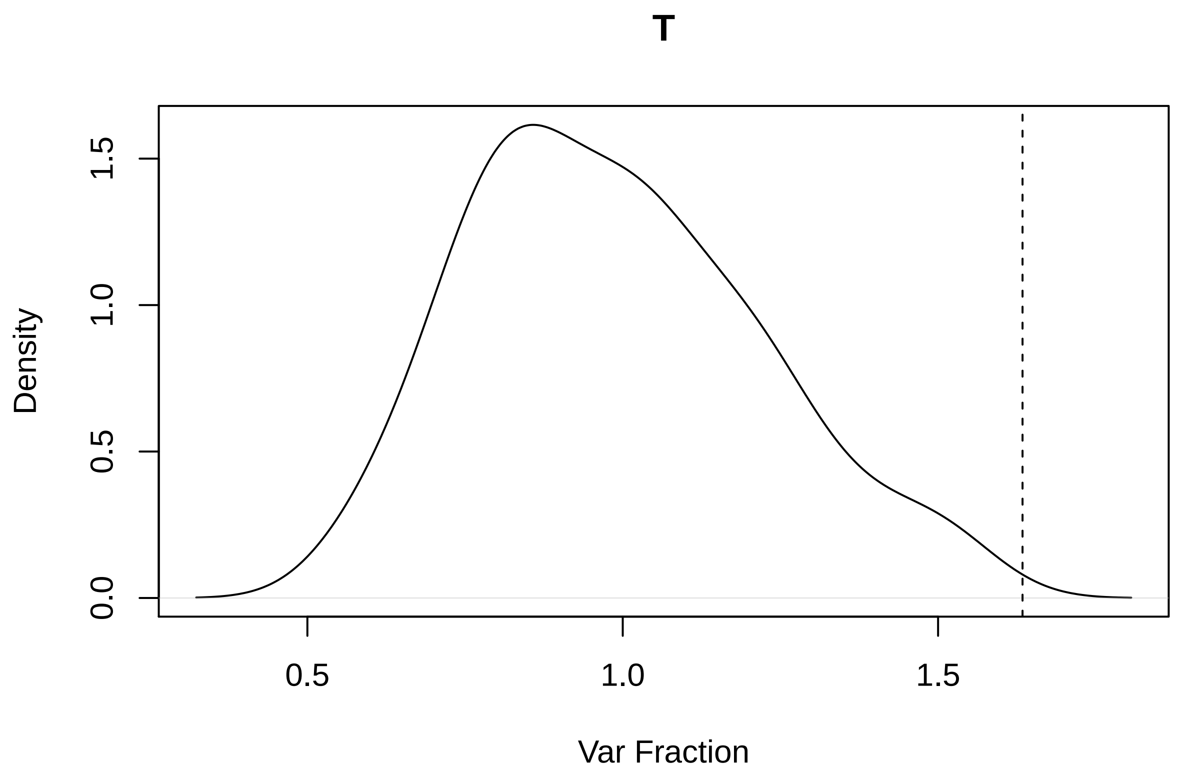
We will use a similar in concept to the ideas in (REF BARTON LAB), but using a simpler approach. This group demonstrated that outlying samples substantially reduce the power of the any downstream tests. Essentially, we will determine how much each sample contributes to the total variance of the group. Samples that contribute a lot of variance to the group are probable outliers and should be discarded. Outliers are defined as those that contribute more than the median + 2 IQR to the variance of the group. This approach, when used on RNA-seq data, captures a similar set of outliers to those of the Barton group in a recent report using an RNA-seq dataset. Note that there is no hard and fast rule for this, all approaches are rules of thumb anyway. To do this, we need to break the samples into T and B groups and calculate the total variance within groups. Then we use a Singular Value Decomposition to determine the amount of variance contributed by each sample within the group, and normalize by dividing by the total variance within the group. We can visualize the distribution, and remove outlying samples with the code below.

```
group.B <- d.n0.clr[which(rownames(d.n0.clr) == "B"),]
group.T <- d.n0.clr[which(rownames(d.n0.clr) == "T"),]

# get proportional variance per sample in each group
pvar.B <- codaSeq.outlier(group.B)
pvar.T <- codaSeq.outlier(group.T)

# density plot of the variance of each sample as a function
# of the total variance of the group identifies outliers
# I define outliers as > 2 IQR away from the median
```

```r
# in these cases the samples are fairly homogeneous
plot(density(pvar.B), main="B", xlab="Var Fraction", ylab="Density" )
cut.B <- median(pvar.B) + ( 2 * IQR(pvar.B) )
abline(v=cut.B, lty=2)
```



```r
plot(density(pvar.T), main="T", xlab="Var Fraction", ylab="Density" )
cut.T <- median(pvar.T) + ( 2 * IQR(pvar.T) )
abline(v=cut.T, lty=2)
```



Conclusions: It appears that the T and B groups are pretty homogeneous. In the B group, only the td_016125, bm_016040 sample is a potential outlier, and it is marginal being almost at the cutpoint (it is the hump on the B plot above. Remarkably, these samples are more homogeneous over a larger sample set than the Barton RNA-seq dataset. This could be because 16S rRNA gene sequencing is more reproducible than RNA-seq (not likely because of the clinical nature of the samples and the amplification needed to generate the samples) or because the total variance is larger in 16S rRNA gene sequencing than RNA-seq datasets. In any event, there is little reason to exclude any samples because they are grossly different than other members of their group.

Checking to see which OTUs are highly correlated in the dataset

Uncovering correlation within compositional datasets is extremely tricky because everything correlates with everything. This is true regardless of the tool used - Pearsons or Spearmans approach which finds linear relationships either in the values or ranks, or with Kendall's Tau which examines the ordering relationships betwee the variables. All of these are prone to false positive identification, and it is not a stretch to say that essentially all correlation networks in microbial datasets are filled with false positives (See Lovell 2015).

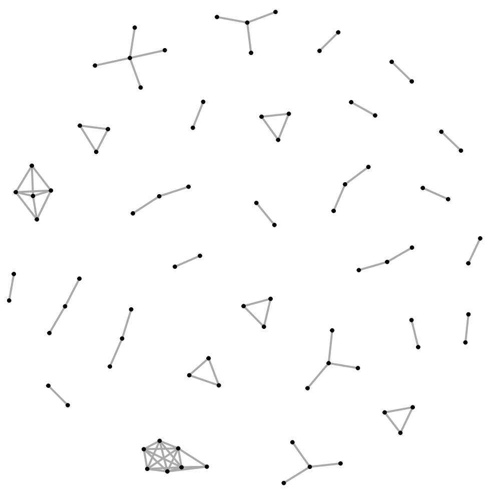
Lovell et al. showed that variables (OTUs) that have nearly constant ratios in all samples are by definition are highly correlated. They also demonstrate that the converse is not true. The phi statistic identifies those OTUs where the var(clr(x)/clr(y)) is essentially constant across samples. This method requires the propr-functions.R set of functions. The data can be explored using graphical approaches, and the code below illustrates how to retrieve the desired information from the graphs.

we will use a cutoff of for this analysis.

```
# I have placed this script in the directory
# change the path accordingly
source("~/git/proprBayes/R/propr-functions.R")
library(igraph)

###########
```

```r
# these are functins that you will need. Copy and paste into the console

sma.df <- function(df){
df.cor <- stats::cor(df, use="pairwise.complete.obs")
df.var <- stats::cov(df, use="pairwise.complete.obs")
df.sd <- sqrt(diag(df.var))
r.rf2 <-
    (outer(diag(df.var), diag(df.var), "-")^2 ) /
    (outer(diag(df.var), diag(df.var), "+")^2 - 4 * df.var^2 )
 diag(r.rf2) <- 0
  res.dof     <- nrow(df) - 2
  F           <- r.rf2/(1 - r.rf2) * res.dof
list(b=sign(df.cor) * outer(df.sd, df.sd, "/"),
p=1 - pf(F, 1, res.dof),
r2=df.cor^2)
}

lt.row.min <- function(X){

  result <- numeric(nrow(X) - 1)

  for(i in 2:nrow(X)){

    result[i-1] <- min(X[i, 1:i-1])

  }

  result

}


propr.phisym <- function (X)
{
 Cov    <- stats::var(X)
 tmp    <- 2 * Cov / outer(diag(Cov), diag(Cov), "+")
 return((1-tmp)/(1+tmp))
}
############ end functions
##

# We start with the log-ratio transformed data

# generate the phi statistic
# this is a measure of the concordance of ratios across samples
# low values are better, a good place to start is 0.05, but this has only been
# examined in one dataset
# this is from Lovell's R package in progress
d.n0.sym.phi <- propr.phisym(d.n0.clr)

# generate a symmetrical regression line of best fit
d.n0.sma <- sma.df(d.n0.clr)
```

```r
# make an empty dataset of the correct size
lt <- which(col(d.n0.sma$b)<row(d.n0.sma$b), arr.ind=FALSE)
lt.ind <- which(col(d.n0.sma$b)<row(d.n0.sma$b), arr.ind=TRUE)

# find the minimum value of each row
d.n0.phi.min <- lt.row.min(d.n0.sym.phi)

# dataframe to hold the info,
# data is a set of vectors where only the lower triangle is kept, because the matrix is symmetrical
# this is needed so the subset function works properly
d.n0.sma.df <- data.frame(row=factor(rownames(d.n0.sma$b)[lt.ind[,"row"]]), col=factor(colnames(d.n0.sma
d.n0.sma.df$b <- d.n0.sma$b[lt]
d.n0.sma.df$p <- d.n0.sma$p[lt]
d.n0.sma.df$r2 <- d.n0.sma$r2[lt]
d.n0.sma.df$phi <- d.n0.sym.phi[lt]

# find the set of connections with phi less than some value
# phi.cutoff <- 0.1
d.n0.sma.lo.phi <- subset(d.n0.sma.df, phi < phi.cutoff)

# generate a graphical object
g <- graph.data.frame(d.n0.sma.lo.phi, directed=FALSE)

# overview of all the proportional relationships
# this can take a long time!!!
 plot(g, layout=layout.fruchterman.reingold.grid(g, weight=0.05/E(g)$phi), vertex.size=1, vertex.color=
```



```r
# # get the clusters from the graph object
 g.clust <- clusters(g)
#
# # data frame containing the names and group memberships of each cluster
 g.df <- data.frame(Systematic.name=V(g)$name, cluster=g.clust$membership, cluster.size=g.clust$csize[g

max.clust <- induced.subgraph(g, which(g.clust$membership %in% which(g.clust$csize == max(g.clust$csize
max.clust.names <- V(max.clust)$name

#
```

```
# # to see the sized of clusters in the dataset (i.e., the number of members in clusters)
# g.clust$csize
#
# # make a subgraph object for later plotting with a specific cluster size
#
# g.10 <- induced.subgraph(g, which(g.clust$membership %in% which(g.clust$csize == 29)))
# # get the names
# g.10.names <- V(g.10)$name
#
# plot(g.10, layout=layout.fruchterman.reingold.grid(g.10, weight=0.05/E(g.10)$phi), vertex.size=3, ver
#
# # generate a variance variance plot with the clusters overlaid
# # this is the aldex_all.txt data.
# plot(x.all$diff.win, x.all$diff.btw, pch=19,cex=0.3)
# points(x.all[g.10.names,"diff.win"], x.all[g.10.names,"diff.btw"], pch=19, col="red", cex=0.4)
# abline(0,2)
# abline(0,-2)
#
```

Here we see a plot of all OTU clusters with phi <= 0.1 in the dataset. The cutoff of 0.05 used in Lovell's paper gave only single pairs so I upped the cutoff a bit: probably needed to do this because the dataset is very noisy compared to the RNA-seq dataset used by Lovell. 16S rRNA gene sequencing datasets have much greater variance and so less power than do RNA-seq datasets (Fernandes 2014). We can explore this grouping a bit, and see that there are two distinct clusters for Gemella in the data. The members of the first group can be found with the following commands:

```
Gem.clust <- g.df$cluster[grep("Gemella", rownames(g.df))]
Gem.clust <- factor(Gem.clust)
```

. The OTU membership in clusters can be obtained as well, and this will show the members that cluster in the Gemella cluster 2:

```
Gem.a.rnms <- rownames(g.df)[g.df$cluster == levels(Gem.clust)[1]]
Gem.b.rnms <- rownames(g.df)[g.df$cluster == levels(Gem.clust)[2]]
```

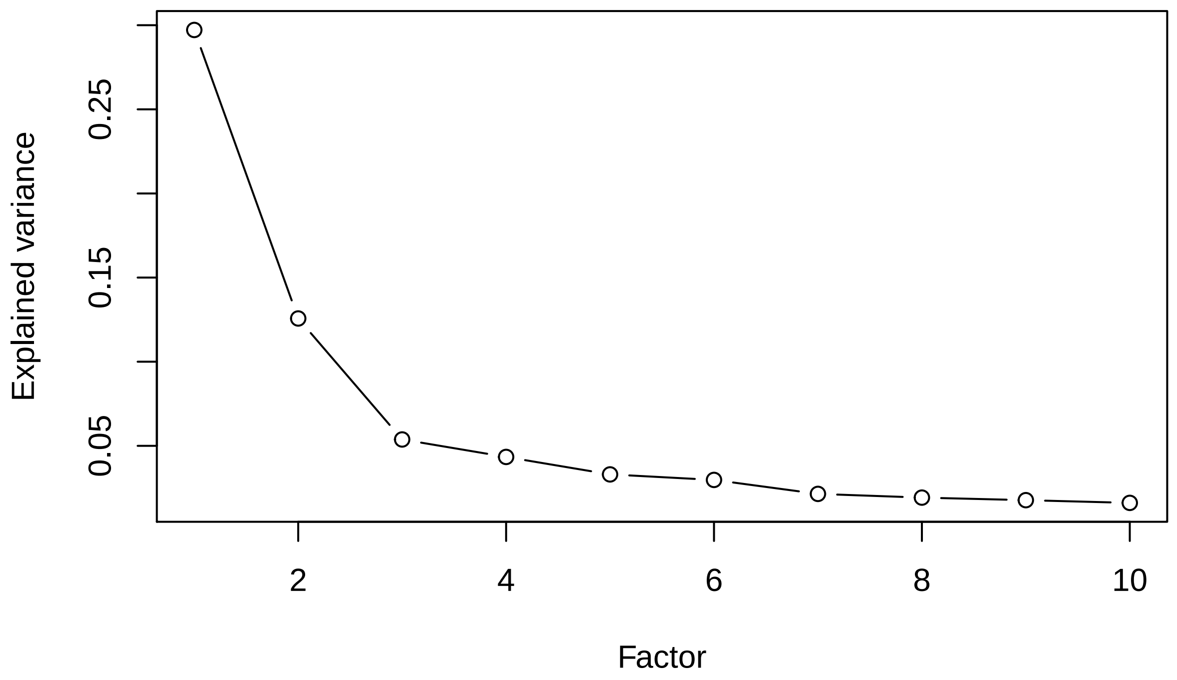
. In the phi network diagram above, this Gemella group is the largest group, and at this cutoff contains only members of this genus. At a cutoff of 0.1 group A membership includes Gemella:22791, Gemella:29014, Gemella:30378, Gemella:30902, Gemella:44273, Gemella:33289, Gemella:2221, and group B includes Gemella:3632, Gemella:6319, Gemella:45351, Gemella:3529. We can see that the majority of the group members are connected to several others.

Principle Component analysis 1: the scree plot.

Instructions for interpreting these are in Figure 8 and discussion of 'introduction_to_compositions.pdf' on the course github site.

```
conds <- data.frame(c(rep(1,length(grep("td_", colnames(d.subset)))),
    rep(2, length(grep("bm_", colnames(d.subset))))))
colnames(conds) <- "cond"

# this is the key function that performs the calculations
pcx <- prcomp(d.n0.clr)
```

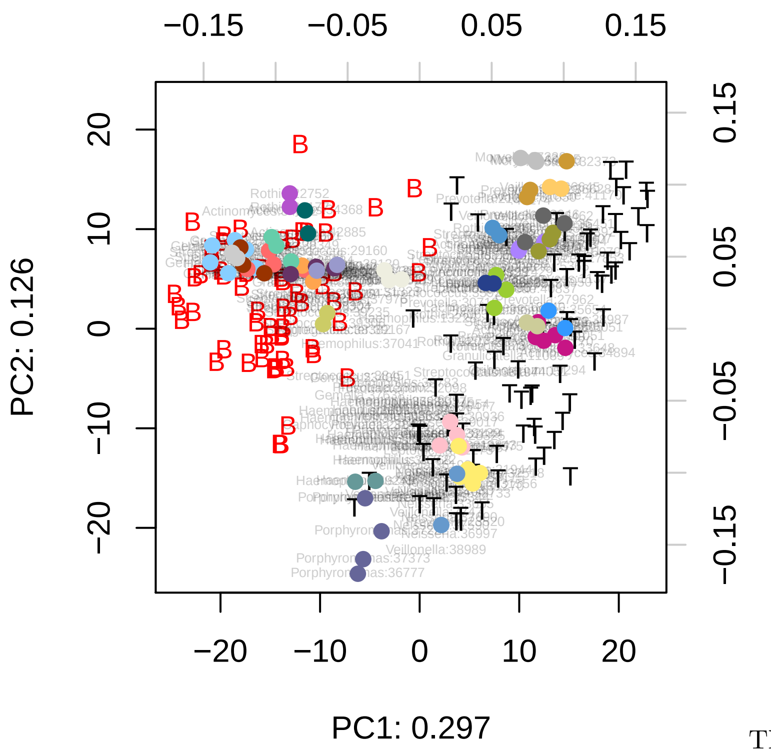
```r
mvar.clr <- mvar(d.n0.clr)

# the scree plot
plot((pcx$sdev^2/mvar(d.n0.clr))[1:10], type="b", xlab="Factor",
    ylab="Explained variance")
```



The biplot attempts to project a multimensional object (our dataset) onto two dimensions. Examining biplots is kind of like Plato's cave, where we can only determine how something looks by examining its shadow on a wall (except we have more than three dimensions in our data). We can examine the scree plot to tell us how well the projection represents the reality of our data. The more variance explained in the first two dimensions, the better the represention. In our case, we are explaining a proportion of 0.423 in the first two dimensions, indicating a good projection for such a large dataset. In addition, examination of the plot shows that there is a large difference between Factor 1-2, and 2-3, and that later factors have very small sequential differences.

PCA analysis 2: The form compositional biplot.

```r
par(mfrow=c(1,1))
coloredBiplot(pcx,col=rgb(0,0,0,0.2),cex=c(0.8,0.4), xlabs.col=conds$cond,
    var.axes=F, arrow.len=0.05,
    xlab=paste("PC1: ", round(sum(pcx$sdev[1]^2)/mvar.clr, 3), sep=""),
    ylab=paste("PC2: ", round(sum(pcx$sdev[2]^2)/mvar.clr, 3), sep=""), scale=0)
colours <- c("steelblue3", "skyblue1", "indianred1", "mediumpurple1", "olivedrab3", "pink", "#FFED6F",

# adding each group in a different coloured dot

lev <- factor(g.df$cluster)
for(i in as.numeric(levels(lev))){
  nms <- rownames(g.df)[g.df$cluster==i]
  points(pcx$rotation[nms,][,1],pcx$rotation[nms,][,2], col=colours[i],    pch=19)
}
```



This is a form biplot that best (given the quality of the projection) represents the distance relationships between samples. This is set with scale=0 in the biplot command. In addition, the length of a ray drawn from the center to a taxon represents how well the variation of that taxon is represented. If the length is 1, then we have a perfect representation. None of our rays are much over 0.1, so there is a lot of noise in these data. Here we can see that the T and B samples separate quite well on component 1, or the major axis. This is what we want to see.

I have over-plotted the location of each cluster with phi < 0.1. Here we can see that many OTUs with the same name are close together in the plot (look for clusters of dots with the same color). In the parlance of biplot interpretation, they have short links. Thus, in some sense phi is giving us similar information to what we could gather from the biplot, but is providing a numerical interpretation on the strength of association.

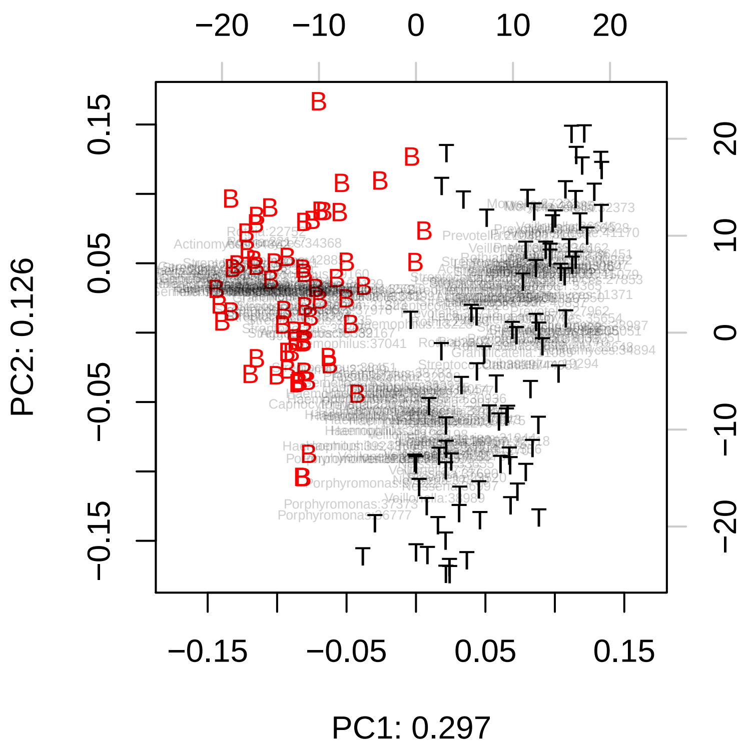
Note, however that some of the pairs of links are long, but transitively, all links are short (note that we acutally need to look at the covariance plot for the more rigorous interpretion, but I have not yet figured out how to plot this). Thus, phi is providing valuable additional insights into the structure of the dataset:

- Some of the individual pairs have short links but other members have fairly long links (see the Porphyromonas in purple at the bottom).

So how can we explain the (relatively) low phi value with the observation that some of the links are somewhat long? The simplest explanation, and the one that I favour, is that the links are only apparently long because the quality of the projection is poor since less than half of the the variance in the dataset is explained by the first two principle components.

PCA analysis 3: The covariance compositional biplot.

```
par(mfrow=c(1,1))
coloredBiplot(pcx,col=rgb(0,0,0,0.2),cex=c(0.8,0.4),
    xlabs.col=conds$cond, var.axes=F, arrow.len=0.05,
    xlab=paste("PC1: ", round(sum(pcx$sdev[1]^2)/mvar.clr, 3), sep=""),
    ylab=paste("PC2: ", round(sum(pcx$sdev[2]^2)/mvar.clr, 3), sep=""), scale=1)
```



This is a covariance biplot that best represents the variance relationships between OTUs. This is set with scale=1. The basic rule for a covariance biplot is that length and direction of each OTU is proportional to the standard deviation of the OTU in the dataset (given the quality of the projection). This allows us to determine how the OTUs variation affects the partitioning of the samples. In addition, the cosine of the angle between two arrows drawn from the center to two taxa approximates the correlation between those two taxa. So the two Porphyromonas in the bottom right that are essentially on top of each other have a strong correlation (see the form biplot above).

Unsupervised clustering.

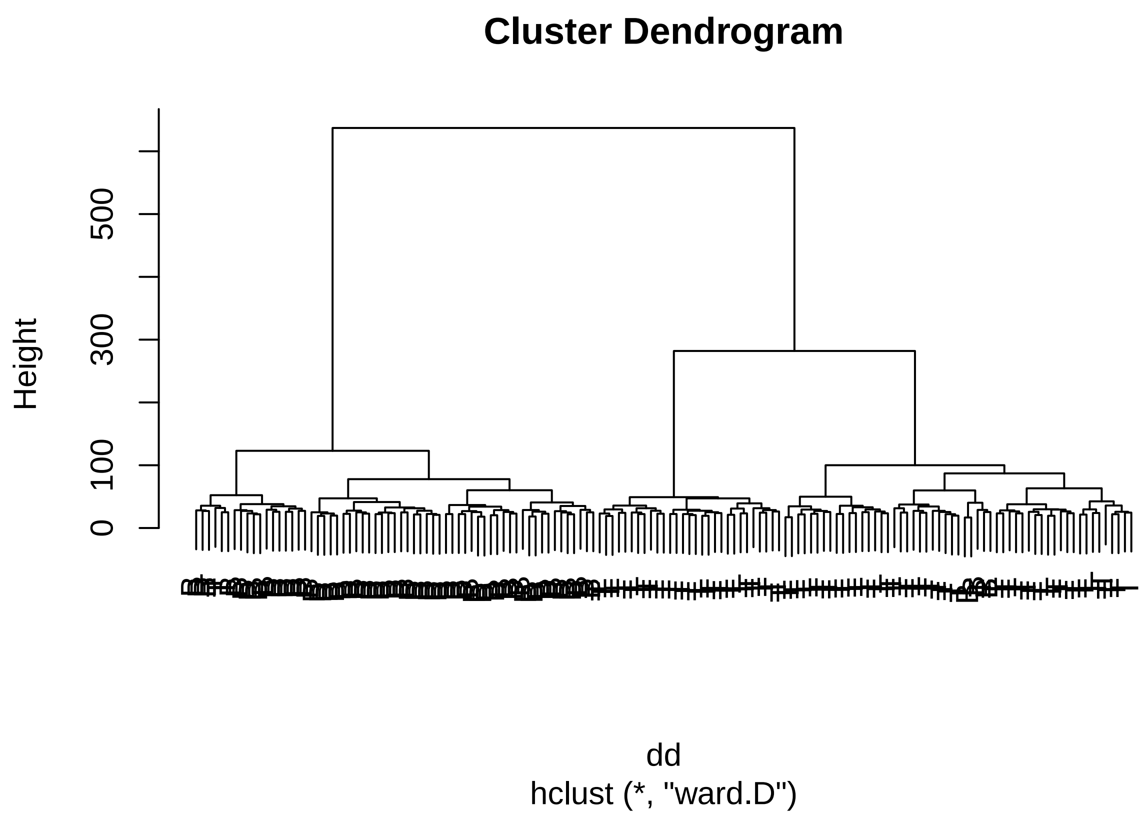
Unsupervised clustering is used to examine the overall similarity between samples by grouping them by their distance (more distance is less similarity). There are many different methods to determine distance, but most are based on either the city block (L1) or Euclidian (L2) distance (see wikipedia)

You should explore how each clustering and distance metric changes the clustering

```
# we are doing Euclidian distance (the default) on the log-ratio values
# the log-ratio values are linearly different, which is required for accurate
# use of Euclidian distance
# this is called an Aitchison distance (by some) and is the valid way to determine
# distance for compositional data

# this generates a matrix of distances between each pair of samples
dd <- dist(d.n0.clr)

hc <- hclust(dd, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```r
plot(hc)
```



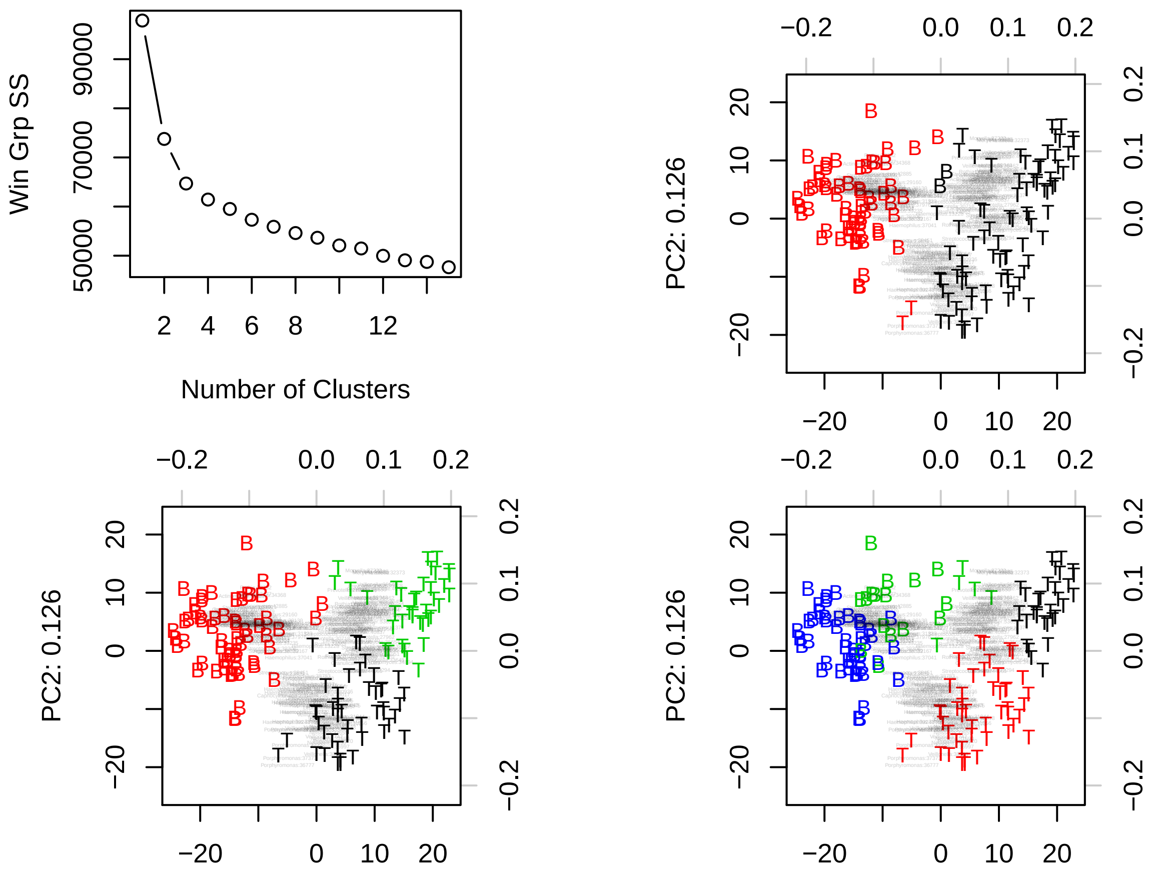
Here we can see that most B and T samples separate very well. There are a few B samples that are embedded within the T samples. We also observed this in the compositional biplots. This type of clustering is deterministic - you will get the same answer every time. This is not a guarantee that it is the right answer, it is just consistent.

k-means clustering

k-means clustering is somewhat non-deterministic. Here randomly chosen points are used as initial centroids within the dataset, with the number of points determined by the user. The data is partitioned and new centroids are chosen based on the centre of each group and the clustering is repeated until convergence (i.e., the old centroid and the new centroid are the same). Of note, is that the initial points are chosen randomly, and so each attempt may give a different answer. This means that group membership can change from one trial to another, and that we should not get hung up on trying to chase small effects. This is because K-means clustering (indeed any stochastic process) is sensitive to the local minimum problem, where the process reaches an apparent best solution. A second issue with this approach is that the expected sizes of the groups are similar. That is, if three groups are chosen, then the data is partitioned into three roughly equally-sized groups, where size is measured as the dispersion (not number of members) of each group. Nevertheless, this is a useful approach to try, if you think your groups show equal dispersion.

```r
mydata <- as.data.frame(d.n0.clr)

# calculate the within group sum of squares
wss <- (nrow(mydata) -1) * sum(apply(mydata,2,var))

# summarize the fit for the first 15 groups
```

```r
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
        centers=i)$withinss)

# the plot suggests a reasonable fit can be had with about 2-3 clusters
# and after that there are only small improvements with additional clusters
layout( matrix(c(1,2,3,4),2,2, byrow=T), widths=c(10,10), height=c(10,10))

par(mar=c(5,4,0,5)+0.1)

# make the sum of squares distance plot
plot(1:15, wss[1:15], type="b", xlab="Number of Clusters",
    ylab="Win Grp SS")

par(mar=c(2,0,2,0)+0.1)

# plot the data for cluster sizes 2-4
for(i in 2:4){
    fit <- kmeans(mydata,i)
    mydata$fit.cluster <- NULL # clear any previous data
    mydata <- data.frame(mydata, fit$cluster) # add the cluster data

    coloredBiplot(pcx,col=rgb(0,0,0,0.2),cex=c(0.8,0.2),
     xlab=paste("PC1: ", round(sum(pcx$sdev[1]^2)/mvar.clr, 3), sep=""),
     ylab=paste("PC2: ", round(sum(pcx$sdev[2]^2)/mvar.clr, 3), sep=""),
     xlabs.col=mydata$fit.cluster, arrow.len=0.05, var.axes=F, expand=0.8,  scale=0)
}
```



The top left plot shows the distance between the center of each group as a function of the number of groups. It is likely that 2 groups best explains the data because there is a large difference in distance between 1 and 2 groups, but not much difference between 2 and 3, or any other successive pair. The group memberships for 2, 3 and 4 groups are shown successively. Note that the actual group memberships change for 3 or 4 groups, but not for 2 groups with different k-means instances.

Categorical separation of T and B

The above methods are unguided methods designed to determine if there is a difference between the T and B groups. The next two methods are guided by our data and tell us if our groups are truly separable, how separable they are, and which taxa separate the groups.

ALDEx2: Which OTUs are different between T and B?

Having satisfied ourselves that T and B are separable, we can ask which OTUs are reliably different in abundance between the two datasets using the ALDEx2 tool (Fernandes 2013, 2014). This tool takes into account both the sampling variability, and the compositional nature of the data. When evaluating the results of this tool, we prefer to use effect size rather than adjusted P values, since P values are sensitive to both the effect size and the sample size. It is well known that a low P value will be observed even when datasets are trivially different, given a large enough sample size. However, effect size, which measures the standardized difference between groups (similar to a Z score) is a much more stable estimate of significance (ref here), and has the further advantage of providing a measure of biological relevance as well.

The ALDEx2 effect size for each OTU is the calculated as the difference between groups divided maximum dispersion within group A or B.

```r
library(ALDEx2)
```

```
## Loading required package: GenomicRanges
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following objects are masked from 'package:igraph':
##
##     normalize, union
##
## The following object is masked from 'package:compositions':
##
##     normalize
##
## The following object is masked from 'package:stats':
##
##     xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, as.vector, cbind,
##     colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##     intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unlist, unsplit
##
## Loading required package: S4Vectors
## Loading required package: stats4
## Creating a generic function for 'nchar' from package 'base' in package 'S4Vectors'
##
## Attaching package: 'S4Vectors'
##
## The following object is masked from 'package:igraph':
##
##     compare
##
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
##
## The following object is masked from 'package:igraph':
##
##     simplify
##
```

```
## Loading required package: GenomeInfoDb
```

```
# vector of conditions
conditions <- com

# estimate 16 sets of data that are consistent with the counts observed
# convert to centred log-ratios
rownames(d.subset) <- paste(short_tax,rownames(d.subset), sep=":")

x <- aldex.clr(d.subset, mc.samples=16, verbose=FALSE)
```

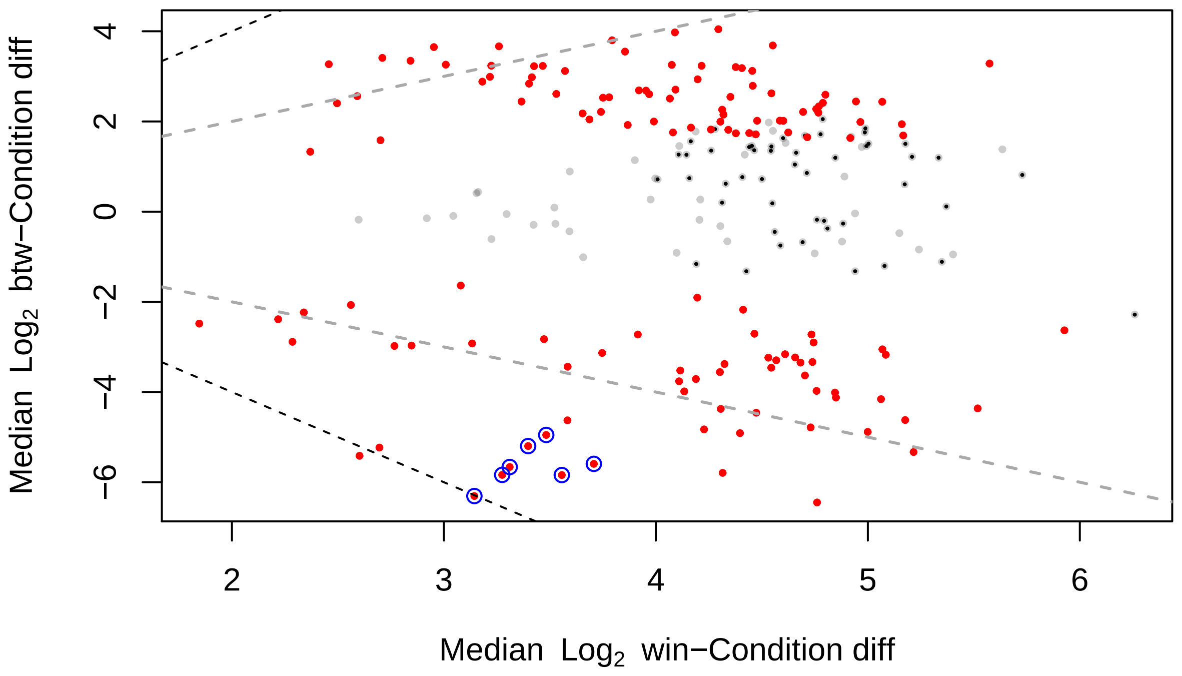
```
## [1] "operating in serial mode"
```

```
## Warning in aldex.clr.function(reads, mc.samples, verbose, useMC,
## summarizedExperiment = FALSE): values are unreliable when estimated with so
## few MC smps
```

```
# calculate the median clr values for summary statistics
# and for effect sizes
x.e <- aldex.effect(x, conditions, verbose=FALSE)
```

```
## [1] "operating in serial mode"
```

```
# conduct parametric and nonparametric hypothesis tests
# report the expected values
x.t <- aldex.ttest(x, conditions)

# combine into one dataframe, and add rownames with taxon names
x.all <- data.frame(x.e,x.t)

# generate an effect plot (Gloor, 2015)
aldex.plot(x.all, method="wilcox", cutoff=0.01)
# identify the Gemella low phi cluster
points(x.all[Gem.a.rnms,"diff.win"], x.all[Gem.a.rnms,"diff.btw"], col="blue")
abline(0,2, lty=2)
abline(0,-2, lty=2)
```



The above figure shows an effect plot of the data where each point represents an individual OTU (Gloor, 2015). The x-axis plots the expected value of the maximum dispersion of the OTU within either group, and the y-axis plots expected value of the difference between groups. The position of the point in each plot thus reflects the underlying constituents of effect size, or standardized difference between groups. Sectors of this plot with an origin at y=0 reflect sectors of similar effect size. The inner dotted lines indicate an effect size of 1 (that is the dispersion within at least one group was equal to the difference between groups), and the outer dotted lines indicate an effect size of 2 (that is, the difference between groups is twice that of the maximum dispersion within either group). Points colored in red indicate OTUs with Benjamini-Hochburg corrected P value less than 0.01. Points in grey or black did not meet this threshold, and the difference in color represents whether the OTUs were abundant or rare.

The location of the low phi Gemella group is identified as those points circled in blue. Again note that these tend to co-locate in the plot. Not surprisingly since they were observed in the biplots to have among the largest standard deviation, these OTUs are among the those with the largest absolute effect sizes (being close to the effect -2 line).

```
x.all[Gem.a.rnms,]
```

```
##                    rab.all rab.win.B  rab.win.T  diff.btw diff.win
## Gemella:22791  0.13069659  3.726544 -1.8223738 -5.662724 3.310466
## Gemella:29014  2.04046178  6.158344 -0.1288325 -6.306304 3.143908
## Gemella:30378  0.19486451  3.876567 -1.7958786 -5.837331 3.275093
## Gemella:30902  0.08778532  3.931642 -1.8011478 -5.840586 3.556345
## Gemella:44273 -0.10252218  3.676551 -1.8917465 -5.592745 3.707472
## Gemella:33289 -0.17974381  3.084898 -2.0530897 -5.198801 3.397427
## Gemella:2221  -0.86672496  2.455111 -2.5511189 -4.951725 3.482799
##                   effect     overlap        we.ep       we.eBH        wi.ep
## Gemella:22791 -1.708366 0.033664537 1.821190e-21 4.008300e-20 6.480264e-22
## Gemella:29014 -1.969470 0.007941621 1.588644e-26 5.825067e-25 2.565608e-24
## Gemella:30378 -1.723487 0.027779212 8.856322e-28 2.532829e-26 3.512907e-22
## Gemella:30902 -1.584167 0.049603958 9.739505e-22 2.381310e-20 1.364962e-20
## Gemella:44273 -1.425811 0.069444991 3.629138e-18 4.699698e-17 1.898728e-18
## Gemella:33289 -1.406785 0.047619866 7.624499e-23 1.797745e-21 1.685611e-20
## Gemella:2221  -1.376328 0.069307476 2.132806e-19 3.431734e-18 4.591773e-18
##                     wi.eBH
## Gemella:22791 2.506803e-20
## Gemella:29014 3.740023e-22
## Gemella:30378 1.458477e-20
## Gemella:30902 3.455208e-19
## Gemella:44273 3.296810e-17
## Gemella:33289 4.691787e-19
## Gemella:2221  6.845559e-17
```

This table shows the summary statistics for the low phi Gemella group. All values are expected values of the 16 random samples of the dataset. The rab columns give the clr value of all, B or T groups. The diff columns give the within group (dispersion) or between group difference. The effect column give information on the standardized difference between groups. The overlap column indicates the fraction of the distributions that overlap. Finally, the we and wi columns give the raw and Benjamini-Hochberg corrected P values for each OTU.

LDA: Linear discriminate analysis

This method is from: http://little-book-of-r-for-multivariate-analysis.readthedocs.org/en/latest/src/multivariateanalysis.html

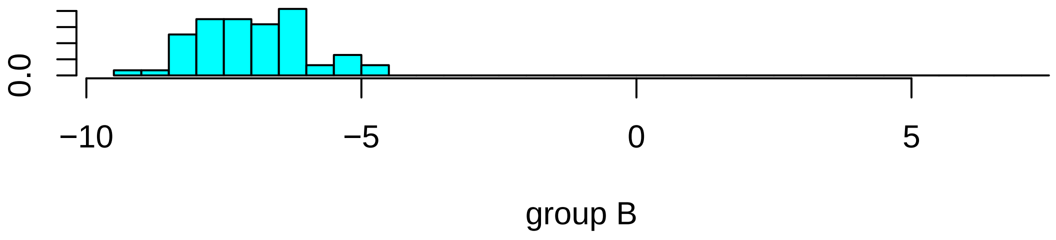
The idea here is to determine which linear combination of OTUs best separates the B and T samples. In some way this is the opposite of determining which OTUs are different since here we assume that the OTUs are different and we are determining the optimum loading function for the OTU abundances that can be used to separate the sample groups. This will work best when those OTUs that are not different between groups are removed. We will first try it with the full set, and then with the smaller set of OTUs that have an absolute effect size greater than 1 as determined by ALDEx2.

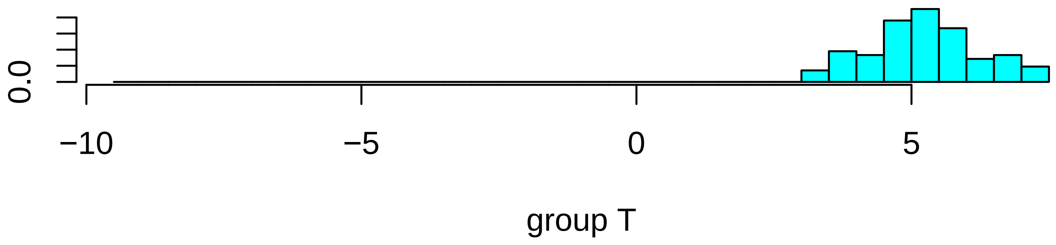
The number of discriminatory functions is the lesser of the number of groups -1, or the number of variables. Thus, we can only have 1 discriminate function for a two group comparison. However, we see below that this approach can completely separate the B and T groups.

```r
# this is from the MASS package loaded by compositions
# remake the clr dataset as a dataframe
myData <- data.frame(d.n0.clr <- codaSeq.clr(d.n0, samples.by.row=TRUE) )
colnames(myData) <- rownames(d.subset)

# com is the vector of B or T names
# ~ . means use all columns of myData as discriminate OTUs
# generate the loadings
fit <- lda(com ~ ., data=myData)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
# use the loadings to generate the funciton
fit.val <- predict(fit)

# plot the separation between the groups.
ldahist(data=fit.val$x[,1], com)
```



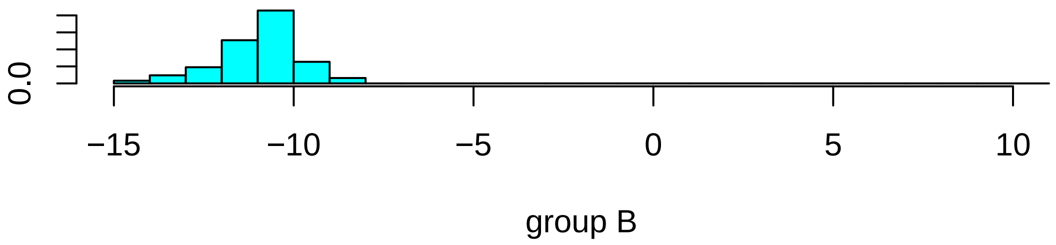


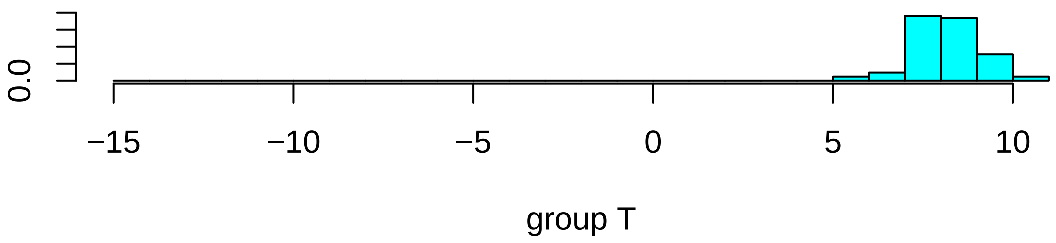
Now lets try this with a subset of the data where only those OTUs with an absolute effect size of at least 0.25: that is, they show at least some difference between groups. Here we can see that the difference between groups is even larger. In practice, increasing the absolute effect size does not increase discriminatory power, likely because there are fewer discriminatory factors (OTUs).

```
## vector of large effect OTUs
big.e <- rownames(x.all)[abs(x.all$effect) >= 0.25]

# subset to include only these OTUs
big.e.data <- myData[,big.e]

# generate the loadings
fit.e <- lda(com ~ ., data=big.e.data)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
# use the loadings to generate the function
fit.e.val <- predict(fit.e)

# plot the separation between the groups.
ldahist(data=fit.e.val$x[,1], com)
```


0.0
−15
−10
−5
0
5
10
group B


0.0
−15
−10
−5
0
5
10
group T