Biplots and clustering

gg

2015-11-16

To run this file: Rscript -e "rmarkdown::render('biplots.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 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. Distance is the building block for clustering, where the idea is to show the distance relationships among the samples. Distance 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 Principle 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 (We will send one around by email).

Now for some code. I have put up a dataset composed of a subset of the Human Microbiome project oral 16S rRNA gene 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.

```
# we need these libraries for the colored biplot, and for the 0 replacement function
library(compositions)
library(zCompositions)

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

# keep only those samples with > 10000 reads
# the which command returns a vector of numbers corresponding to the columns that have > 10000 reads. T
d.bf.0 <- d.bf.1[,which(apply(d.bf.1,2,sum) > 10000)]

# simplify the dataset, by keeping only taxa present in x fraction of the samples
# the which function returns a list of row numbers where the number of 0 values in the row is greater t
cutoff = .3
d.subset <- data.frame(d.bf.0[which(apply(d.bf.0, 1, function(x){length(which(x != 0))/length(x)}) > cu
tax.subset <- tax.0[which(apply(d.bf.0, 1, function(x){length(which(x != 0))/length(x)}) > cutoff)]

# oh boy, so much here to explain
# 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 n
short_tax <- gsub(".+__", "", tax.subset)

# 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.su
```

Herein ends the data munging. In the end we are left with 1770 OTUs and ňcol(d.subset)‘ samples. This is just for convenience, and the second example contains many more.

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
d.n0 <- cmultRepl(t(d.subset),  label=0, method="CZM", output="counts")
```

```
## No. corrected values:  5837
```

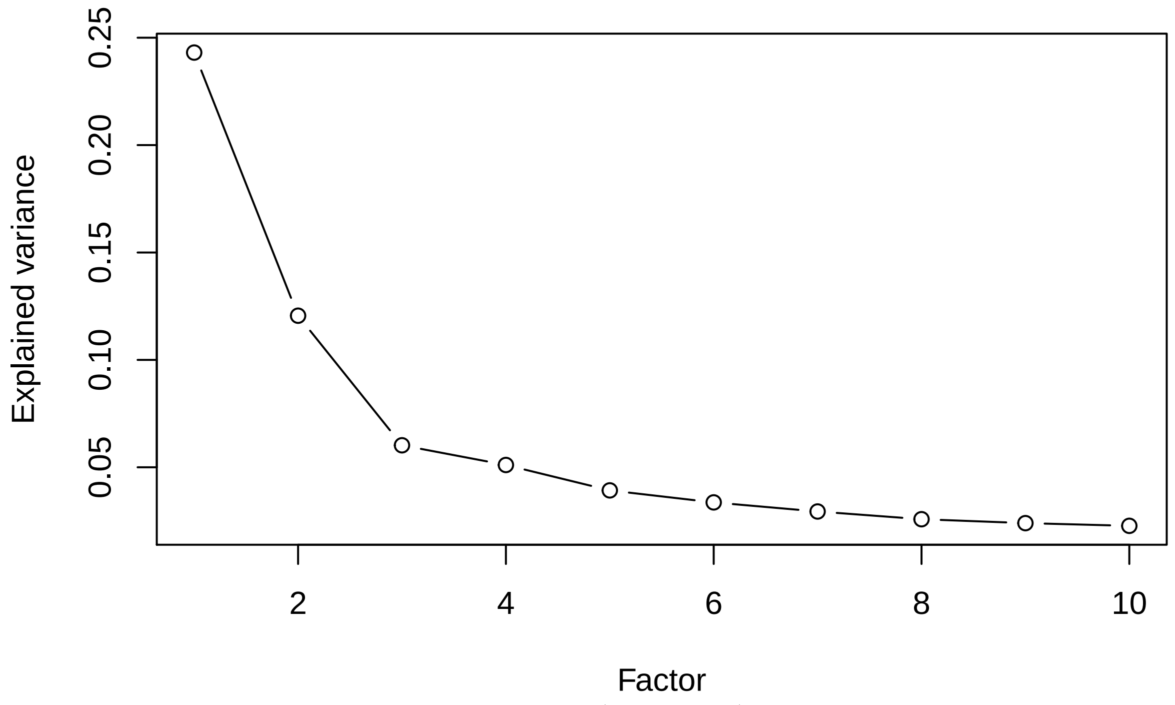
```
# convert everything to log-ratios
# equivalent to log(x/gx) for every value where gx is the geomtric mean of the vector X
# I prefer to use log to the base 2, although purists disagree
d.n0.clr <- apply(d.n0, 2, function(x){log2(x) - mean(log2(x))})

# replace row and column names for display
# this only works because we now have a matrix, not a dataframe
colnames(d.n0.clr) <- short_tax
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

The scree plot.

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

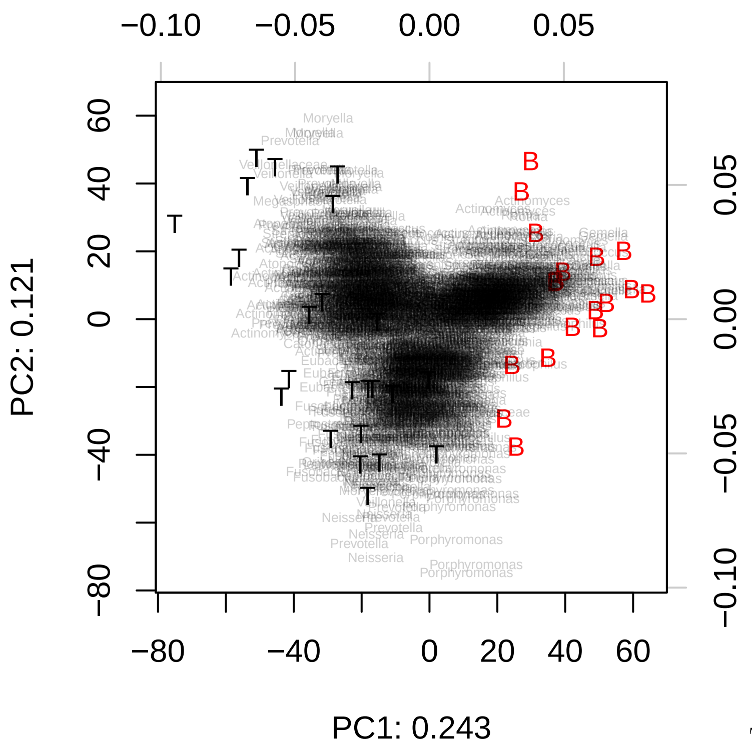
```
conds <- data.frame(c(rep(1,length(grep("td_", colnames(d.bf.0)))), rep(2, length(grep("bm_", colnames(
colnames(conds) <- "cond"

# this is the key function that performs the calculations
pcx <- prcomp(d.n0.clr)
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 0.364 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.

The form compositional biplot.

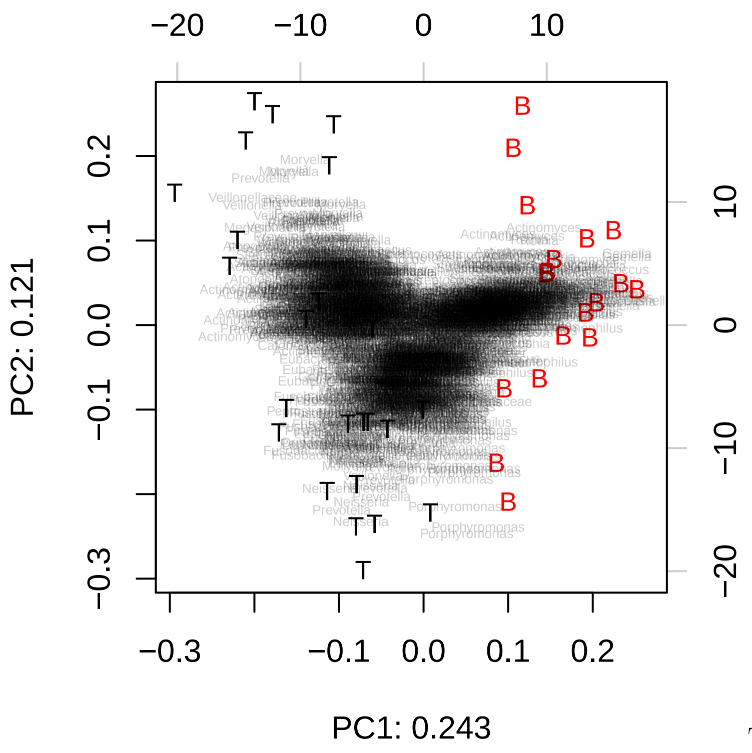
```
par(mfrow=c(1,1))
coloredBiplot(pcx,col=rgb(0,0,0,0.2),cex=c(0.8,0.4), xlabs.col=conds$cond, var.axes=F, arrow.len=0.05, 
```



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 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 0.3, 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.

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, 
```



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 should have a strong correlation.

Unsupervised clustering.

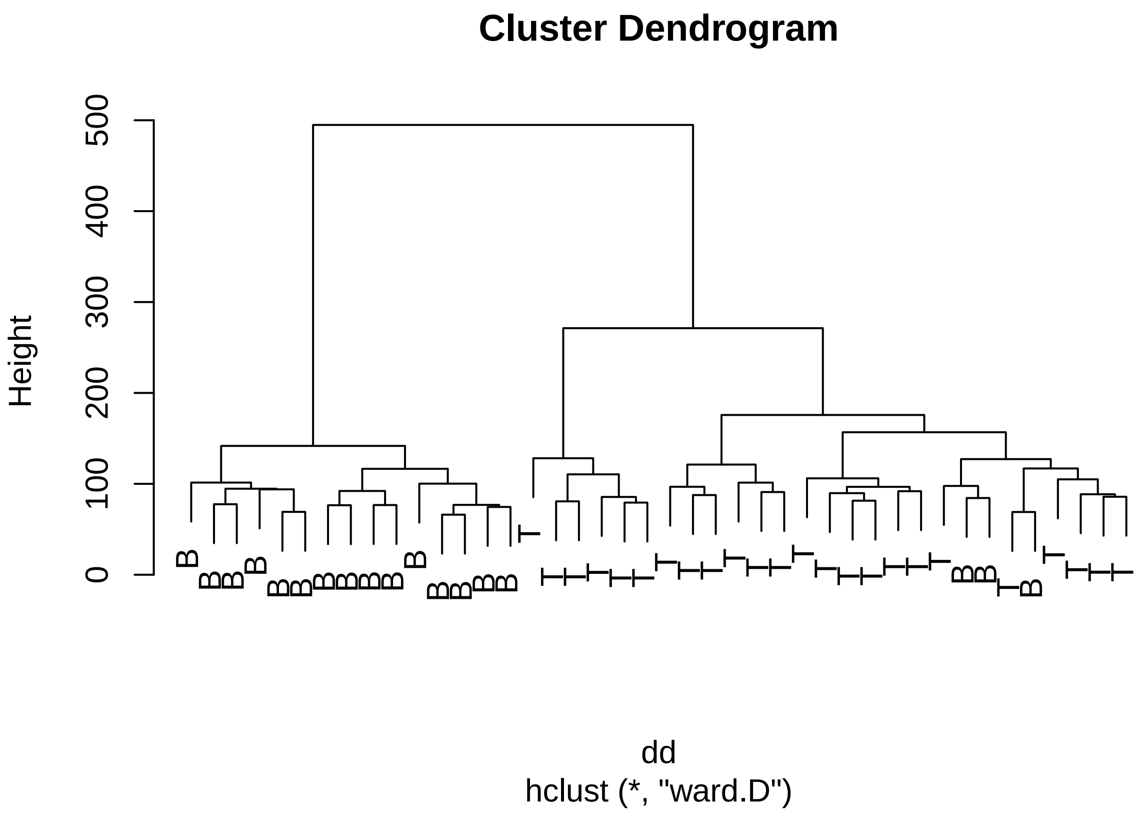
This approach is used to examine the overall similarity between samples by grouping them by their distance (more distance is less similarity). There are many different methods to determine distance, but most are based on either the city block (L1) or Euclidian (L2) distance (see wikipedia)

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

```r
# 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.

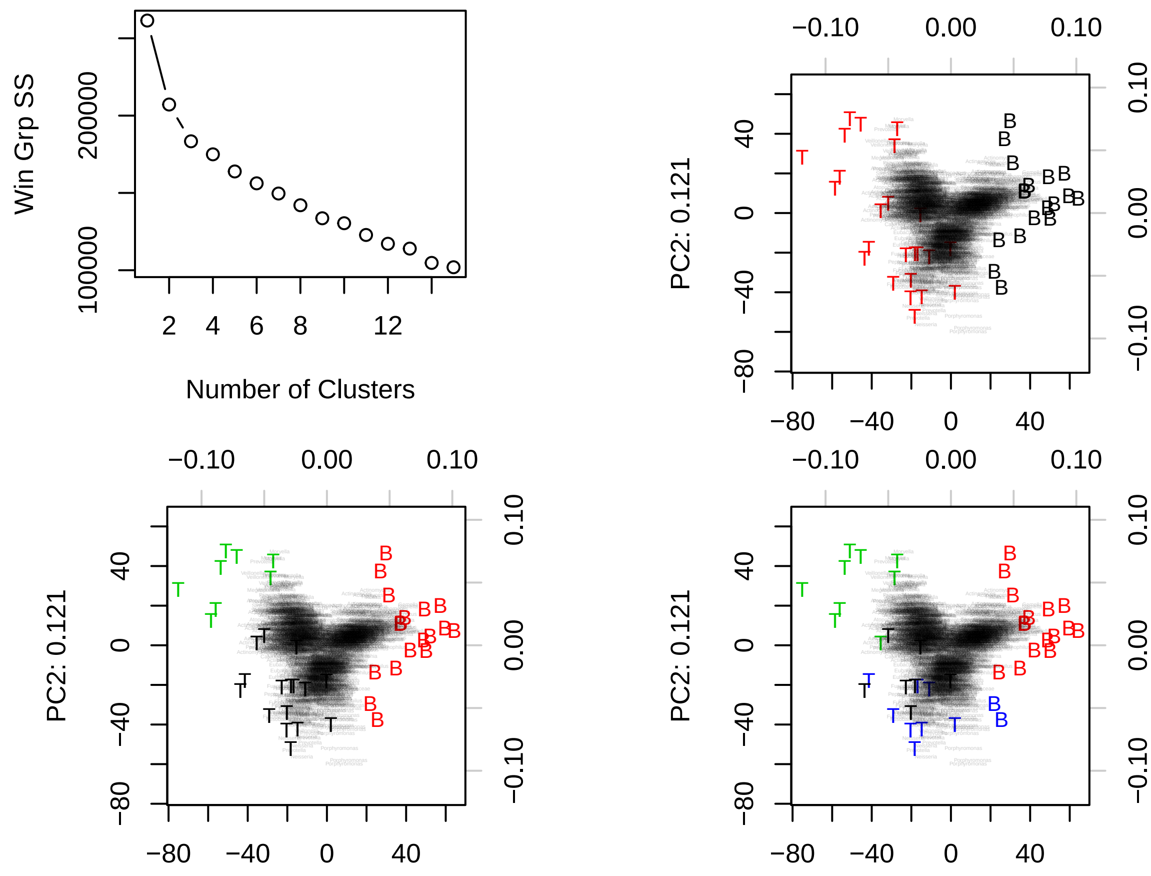
k-means clustering

k-means clustering is somewhat non-deterministic. Here the middle of a group is estimated randomly, and the center and group membership is iteratively approached. 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, because the results can change if the effect is small.

```
#palette <- old.palette
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 10 groups
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 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)

# this is 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)

fit <- kmeans(mydata,2) # fit to 2 clusters
mydata$fit.cluster <- NULL # clear any previous data
mydata <- data.frame(mydata, fit$cluster) # add the cluster data
```

```
## Warning in data.row.names(row.names, rowsi, i): some row.names duplicated:
## 2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,27,28,29,30,31,32,33,34,35,36,37,38,
## --> row.names NOT used
```

```
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)

fit <- kmeans(mydata,3) # fit to 2 clusters
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)

fit <- kmeans(mydata,4) # fit to 2 clusters
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.

Now to examine tongue and saliva

you will need to grab the tongue_saliva.txt file from github

```r
# read the entire data table
# tongue samples start with td_
# saliva samples start with sa_
d <- read.table("tongue_saliva.txt", header=T, row.names=1, sep="\t")

# get the taxon list and remove that column from the table
tax <- d$tax.0
d$tax.0 <- NULL

# samples must be by rows so use t()
d.n0 <- cmultRepl(t(d), label=0, method="CZM")
```

```
## No. corrected values:  649623
```

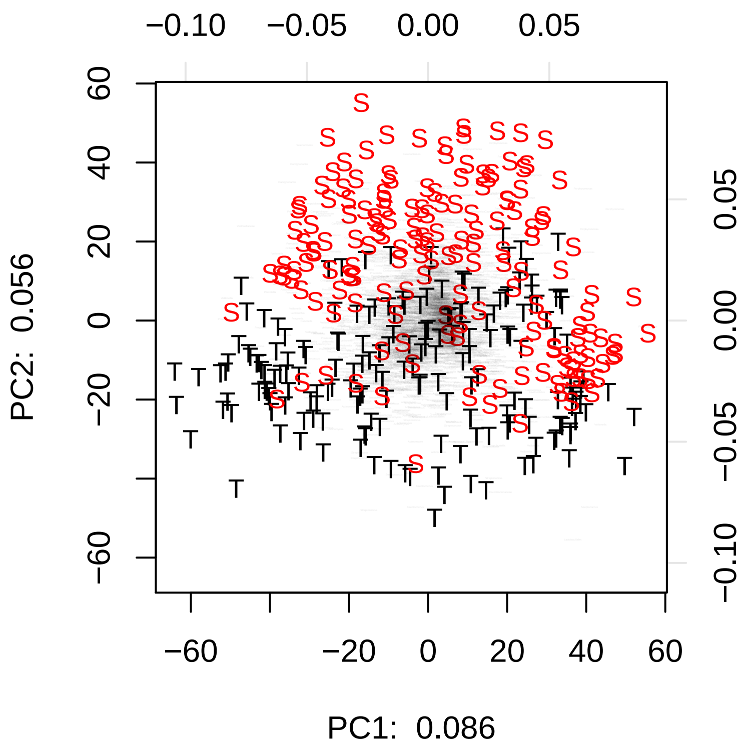
```
# the apply function rotates the data when by row, so turn it back to samples by row
# how I hate R
d.n0.clr <- t(apply(d.n0, 1, function(x){log2(x) - mean(log2(x))}))

####
# this set of code replaces row and column names with single values
# and the genus name of each taxon, if known
# replace rownames in d.n0.clr with one character values
# gsub is substitute the every occurence of the pattern with the replacement
# pattern is read as: "sa_ followed by one or more other characters (.+)"
# See Regular_expression on wikipedia if you are keen, this is POSIX grep
rownames(d.n0.clr) <- gsub("sa_.+", "S", rownames(d.n0.clr))
rownames(d.n0.clr) <- gsub("td_.+", "T", rownames(d.n0.clr))

# make sure you are using the correct taxon list if you reduce the dataset by cutoff
# as it is now, it is using the entire dataset
# see class code for example
colnames(d.n0.clr) <- gsub(".+__", "", tax)

conds <- data.frame(c(rep(1,length(grep("T", rownames(d.n0.clr)))), rep(2, length(grep("S", rownames(d.n
colnames(conds) <- "cond"
#### We are done loading and munging data
```

Herein ends the data munging. In the end we are left with 3166 OTUs and 352 samples. This second example contains many more OTUs and variables because I have essentially not filtered.

make the PCA object

Here we are doing just exactly as before, but I minimized the taxon names because there are so many. If you are interested in finding the taxa that are outliers change the plot rgb value in the command below 0.1 to 0.5 (make text a darker grey), and change the second value in the cex command to about 0.5 as well (make text larger). It will be ugly!!!!

```
# generate the PCA object
pcx <- prcomp(d.n0.clr)
mvar.clr <- mvar(d.n0.clr)

# plot it
# you control the color of the taxa with the rgb(red,green,blue,transparency) function
# you control the size of the sample and tax labels with the cex() function
coloredBiplot(pcx,col=rgb(0,0,0,0.1),cex=c(0.8,0.1), xlabs.col=conds$cond,
    xlab=paste("PC1: ", round(sum(pcx$sdev[1]^2)/mvar.clr, 3)),
    ylab=paste("PC2: ", round(sum(pcx$sdev[2]^2)/mvar.clr, 3)),
var.axes=F, arrow.len=0.05, scale=0)
```



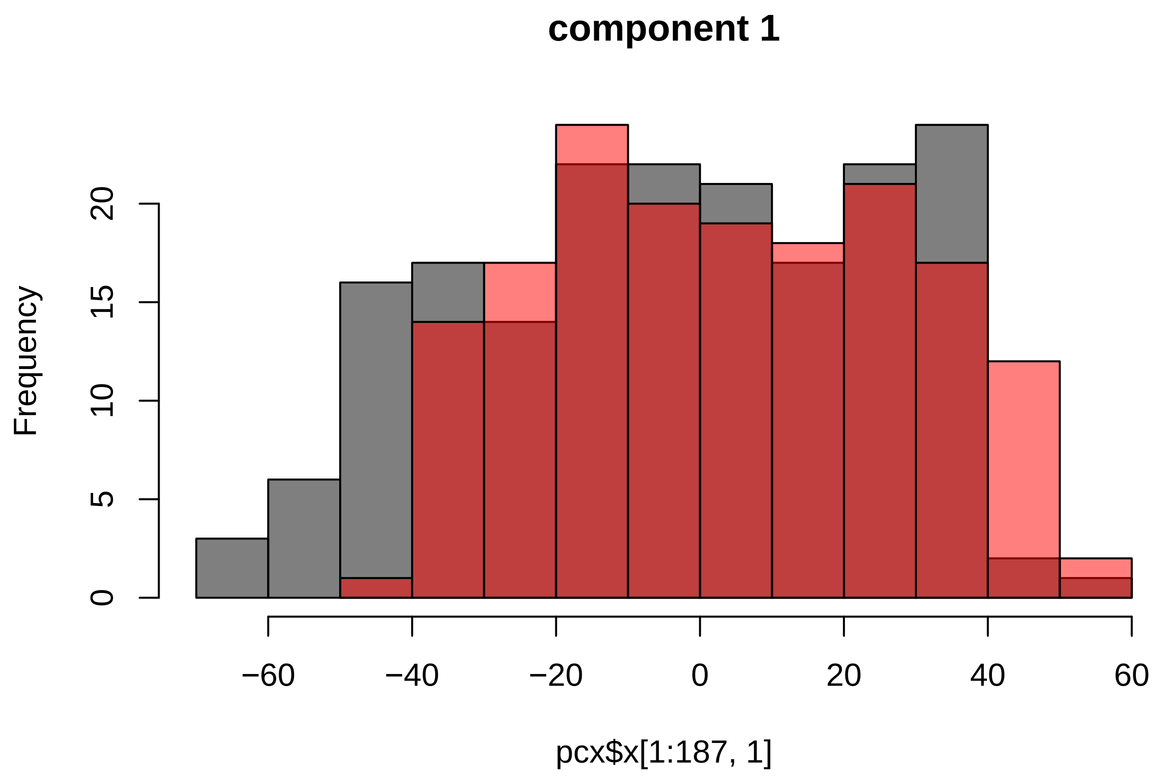
```
# this gives the proportion of variance explained in the first five components
```

Here we can see that the proportion of variation explained is much smaller. In other words, this is not a very good projection: in other words these data are not very efficient at separating anything. We need to really temper any conclusions we make from these data since basically we know from the outset that they are not very explanatory.

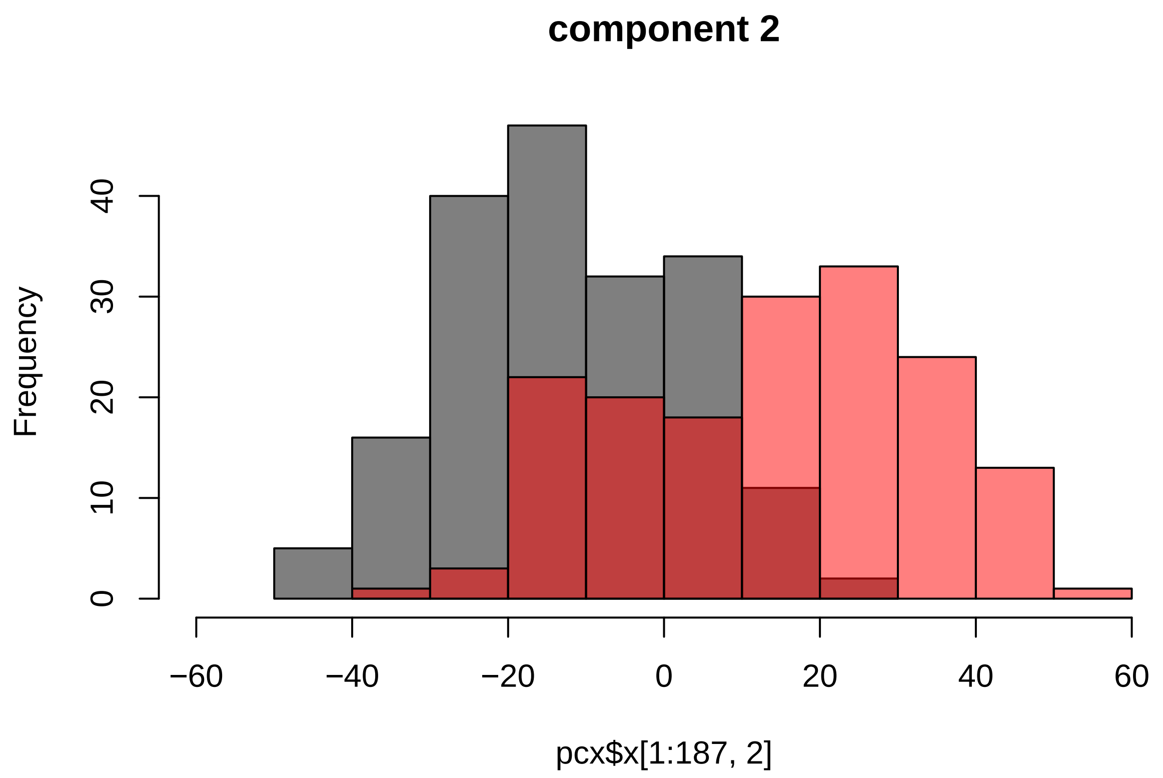
We can see from the biplot that the Saliva and Tongue samples appear to separate on component 2. This has a very simple explanation. There is more variation within samples (smeared out along component 1) than between samples. In other words they are really not that different and we would need huge sample sizes to distinguish them.

One way to look at this is to make a histogram of the distributions of values from components 1 and 2. In this plot, grey is tongue, light red is saliva, and dark red is tongue and saliva. We can see that they are essentially completely overlapping on component 1, and somewhat separable on component 2.

```
hist(pcx$x[1:187,1], col=rgb(0,0,0,0.5), main="component 1")
hist(pcx$x[188:352,1], col=rgb(1,0,0,0.5), add=T)
```



```
hist(pcx$x[1:187,2], col=rgb(0,0,0,0.5), xlim=c(-60,60), main="component 2")
hist(pcx$x[188:352,2], col=rgb(1,0,0,0.5), add=T)
```

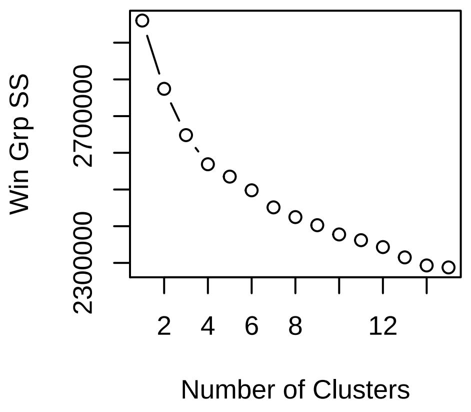


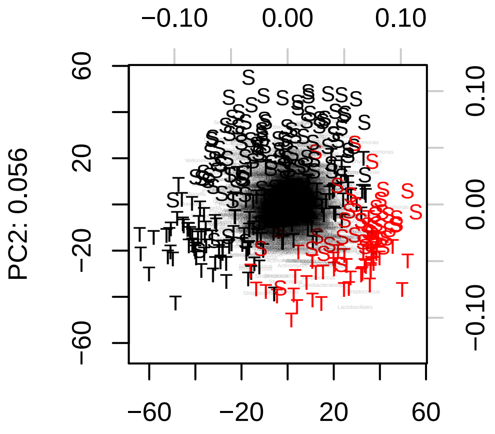
One way to think about this is what information have I gained? If I gave you a blinded sample and it ended up anywhere on component 1, then we do not know if it is tongue or saliva. Likewise if it ends up between

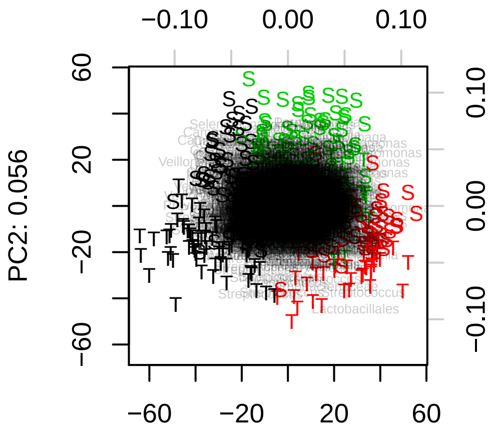
-20 and 20 on component 2, then we don't really know either. But if it ends up at the extremes of component 2, then we have some predictive power.

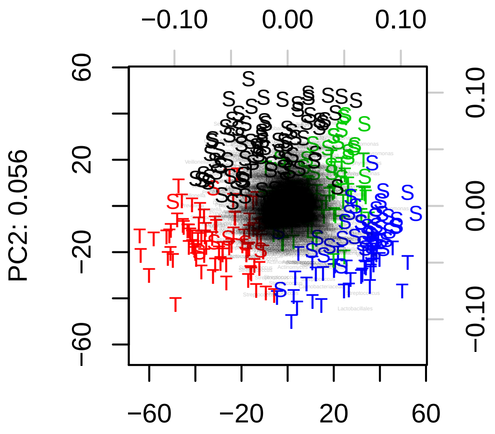
Finally, we can try k-means clustering on these data as before.

```
mydata <- as.data.frame(d.n0.clr)
# k-means clustering
# strongly suggests 2 groups
wss <- (nrow(mydata) -1) * sum(apply(mydata,2,var))
# summarize the fit for the first 10 groups
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
        centers=i)$withinss)

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)

# this is 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)

fit <- kmeans(mydata,2) # fit to 2 clusters
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)

fit <- kmeans(mydata,3) # fit to 2 clusters
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(0.8,0.5),
 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)

fit <- kmeans(mydata,4) # fit to 2 clusters
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)
```









These data suggest 3 groups is the best natural fit, although there is not nearly as much support for natural groups as with the tongue and cheek comparison.