Multivariate Analysis in R

Lab Goals

1. Verification of svd properties.
2. Comparison of classical multidimensional scaling (`cmdscale`) and pca.
3. Learn to interpret output from multivariate projections.

Again, we recommend making a .Rmd file in Rstudio for your own documentation. This can be used to automatically build a .html or a .pdf for you which makes this reproducible.

Note: This lab will focus on the CRAN package `ade4`.

```
require(ade4) # multivariate analysis
require(ggplot2) # fancy plotting
require(grid) # has the viewport function
require(phyloseq) # GlobalPatterns data
```

Note that you can use `=` and `<-` interchangeably in R when you are assigning a value to an object. This lab was put together by authors who have different preferences in this notation.

SVD review

Read sections 1, 2, and 3 of the Wikipedia article about SVD. Read section 1, 2, and 2.1 of the Wikipedia article about eigendecomposition of a matrix. We know that we can decompose a \(n\) row by \(p\) column rank 1 matrix \(X\) as follows.

\[{\bf X} = \begin{pmatrix} x_{11} & x_{12} &\ldots & x_{1p}\\ x_{21} & x_{22} &\ldots & x_{2p}\\ \vdots & \vdots &\vdots & \vdots \\ x_{n1} & x_{n2} &\ldots & x_{np} \end{pmatrix} = \begin{pmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{n1} \\ \end{pmatrix} \times \begin{pmatrix} v_{11} & v_{21} & \cdots & v_{p1} \end{pmatrix} \]

If \(X\) has no rows and no columns which are all zeros, then is this decomposition unique?

Fun with SVD and Eigendecomposition

For the statistically inclined, you can read the paper Multivariate Data Analysis: The French Way. The short version is that there is a unifying connection between many multivariate data analysis techniques.

First note that if you have an \(n\) by \(p\) matrix, then \(X'X\) is \(n\) by \(n\) and \(XX'\) is \(p\) by \(p\).

When we take the SVD of \(X\), we get \(X = UDV'\). Hence, \(X'X = (UDV')'(UDV') = (VDU')(UDV') = VDU'UD'V'\). Since \(U\) is orthonormal, \(U'U = I\) is the identity. Since \(D\) is diagonal, \(D = D'\). So \(X'X = VD^2V'\). We can do similar calculations for \(XX'\). In the OHMS questions, we ask you about the relationship between the SVD of \(X'X\), the eigendecomposition of \(X'X\), and the SVD of \(X\). To answer those questions, you can either do the math to figure out the right answer, or you can generate some random data and do small simulations to try to figure it out. You might consider generating some data like this: `x <- matrix(rnorm(20), ncol=4)`.

Generating a rank one matrix

Now we want to make a rank one matrix. We take a vector of length 15 with values from 2 to 30 in increments of 2, and a vector of length 4 with values 3,6,9,12:

```
u=seq(2,30,by=2)
v=seq(3,12,by=3)
X1=u%*%t(v) # transpose so the dimensions match in the multiplication
X1
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    6   12   18   24
##  [2,]   12   24   36   48
##  [3,]   18   36   54   72
##  [4,]   24   48   72   96
##  [5,]   30   60   90  120
##  [6,]   36   72  108  144
##  [7,]   42   84  126  168
##  [8,]   48   96  144  192
##  [9,]   54  108  162  216
## [10,]   60  120  180  240
## [11,]   66  132  198  264
## [12,]   72  144  216  288
## [13,]   78  156  234  312
## [14,]   84  168  252  336
## [15,]   90  180  270  360
```

Note that this is the exact same as

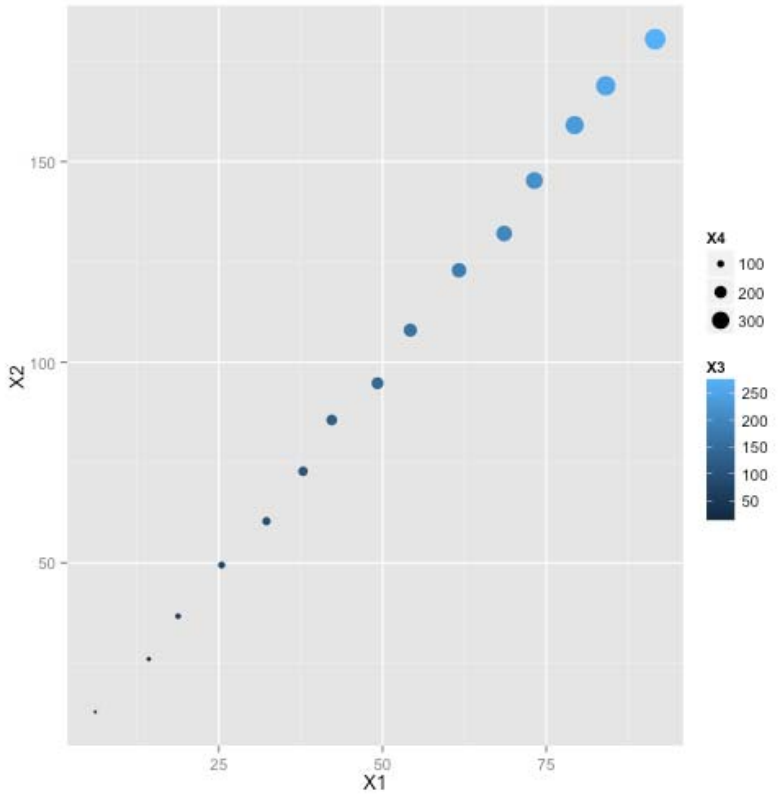
```
X1=outer(u,v)
```

Now let's add some noise to this, so we have an “approximately rank one” matrix.

```
# Make a noise matrix to add to X1
Matex <- matrix(rnorm(60,1),nrow=15,ncol=4)
X <- X1+Matex
```

Let's see what \(X\) actually looks like. Remember, \(X\) is four dimensional – so to try to visualize this it is easiest to do one of two things. We can look at lots of plots in two dimensions and even make a movie where we rotate which two dimensions we're looking from: this is the approach taken in `ggobi` which you can learn about on your own if you want. Another method is to plot the data in two dimensions and use plotting aesthetics such as point color and point size to try to visualize the other dimensions. When using plot aesthetics like this, I think about big points as being closer to me (so I can imagine 3 dimensions relatively easily), and for me color is the next easiest way to represent a dimension (I struggle with this for more than 2 colors though – the default in ggplot2 ranges from black to blue).

```
ggplot(data=data.frame(X), aes(x=X1, y=X2, col=X3, size=X4)) + geom_point()
```



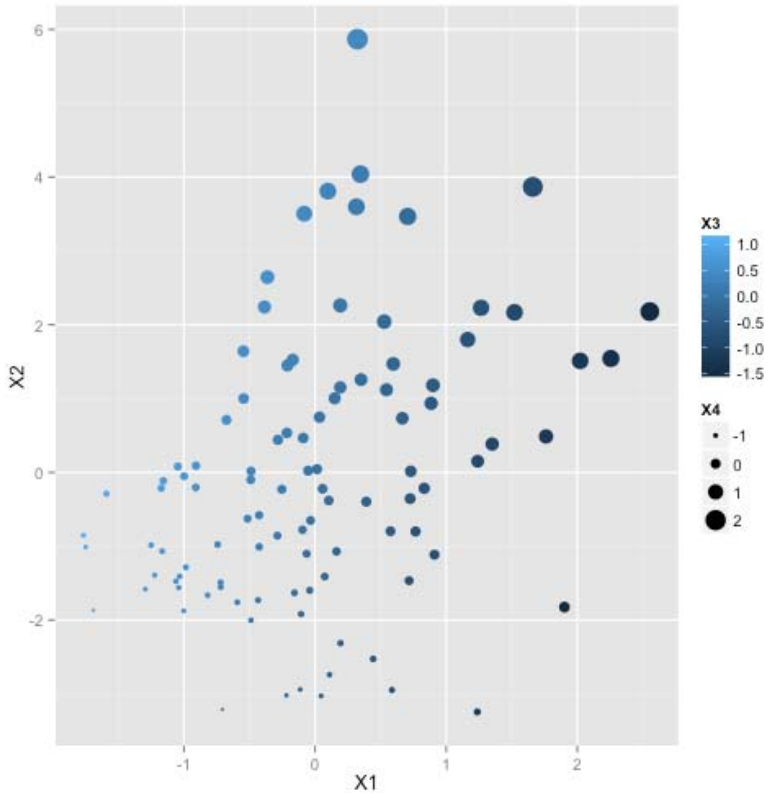
Here we see that the data looks linear in all four dimensions. This is what it means to be rank one.

Now let's consider a rank 2 matrix.

```
set.seed(0)
n <- 100
p <- 4
Y2 <- outer(rnorm(n), rnorm(p)) + outer(rnorm(n), rnorm(p))
head(Y2)
```

```
##          [,1]   [,2]      [,3]    [,4]
## [1,]  0.11069 -2.741 -0.364922 -0.8642
## [2,]  0.19225  1.152 -0.009778  0.4557
## [3,]  0.04632 -3.028 -0.351155 -0.9839
## [4,]  2.25561  1.543 -1.377745  1.3865
## [5,] -0.06417 -1.102 -0.072460 -0.3895
## [6,]  0.09772  3.810  0.335341  1.2987
```

```
ggplot(data=data.frame(Y2), aes(x=X1, y=X2, col=X3, size=X4)) + geom_point()
```



Now there are obviously at least two dimensions because if we project the data onto the first two coordinates (by default called `X1` and `X2` when you convert a matrix into a data frame in `R`), then the data varies in both dimensions. So the next step is to try to decide if there are more than two dimensions. The top right points are the closest to you (they're biggest) and as you go down and left in the plot those points are farther away. In the left are the bluest points and they seem to get darker linearly as you move right.

As you can probably tell, it is very hard to visually discover a low dimensional space in higher dimensions, even when “high dimensions” only means 4! This is one reason why we rely on the singular value decomposition.

```
svd(Y2)$d # two non-zero eigenvalues
```

```
## [1] 1.986e+01 1.025e+01 2.330e-15 1.120e-15
```

```
Y <- Y2 + matrix(rnorm(n*p, sd=0.01),n,p) # add some noise to Y2
svd(Y)$d # four non-zero eigenvalues (but only 2 big ones)
```

```
## [1] 19.85669 10.24460  0.10767  0.08972
```

Here we have two dimensions which are non-zero and two dimensions which are approximately 0 (for `Y2`, they are within square root of computer tolerance of 0).

Principal Component Analysis

Relating PCA to SVD

Let X be a centered but unscaled matrix. We will show that there is a matrix X_r whose principal component output (without rescaling the columns) is the same as the eigendecomposition of $X'X$.

The first k principal components of X are the first k directions explaining maximum variance. This is equivalent to the first k eigenvectors of the covariance matrix. We estimate the sample covariance matrix as $S = X'X/N$. Hence, the principal component analysis of X gives the first k eigenvectors of $X'X/N$.

So if we let $X_r = X*\sqrt{N}$, then the pca output will be the first k eigenvectors of $((X*\sqrt{N})‘(X*\sqrt{N}) / N = X'X$. That is the eigendecomposition of (the centered) X.

Let's apply this to the X above to verify that our calculations are correct. First, we need to center it, and we will call that centered version `Xc`. In our case, $N=15$.

```
Xmeans <- apply(X,2,mean)
Xc <- sweep(X,2,Xmeans)
Sc <- crossprod(Xc)
Sce <- eigen(Sc)
# here is Xr
Xr <- Xc * sqrt(15)
Xr.pca <- dudi.pca(Xr, scale=F, scannf=F, nf=4)
Xr.pca$eig
```

```
## [1] 3.036e+05 1.472e+01 1.019e+01 1.724e+00
```

```
Sce$values
```

```
## [1] 3.036e+05 1.472e+01 1.019e+01 1.724e+00
```

```
# eigenvectors are the same (up to sign)
Xr.pca$c1
```

```
##        CS1     CS2      CS3       CS4
## V1 -0.1821  0.1380  0.87966 -0.417089
## V2 -0.3648 -0.9282  0.07263  0.005302
## V3 -0.5463  0.1748 -0.46144 -0.676784
## V4 -0.7316  0.2980  0.08938  0.606607
```

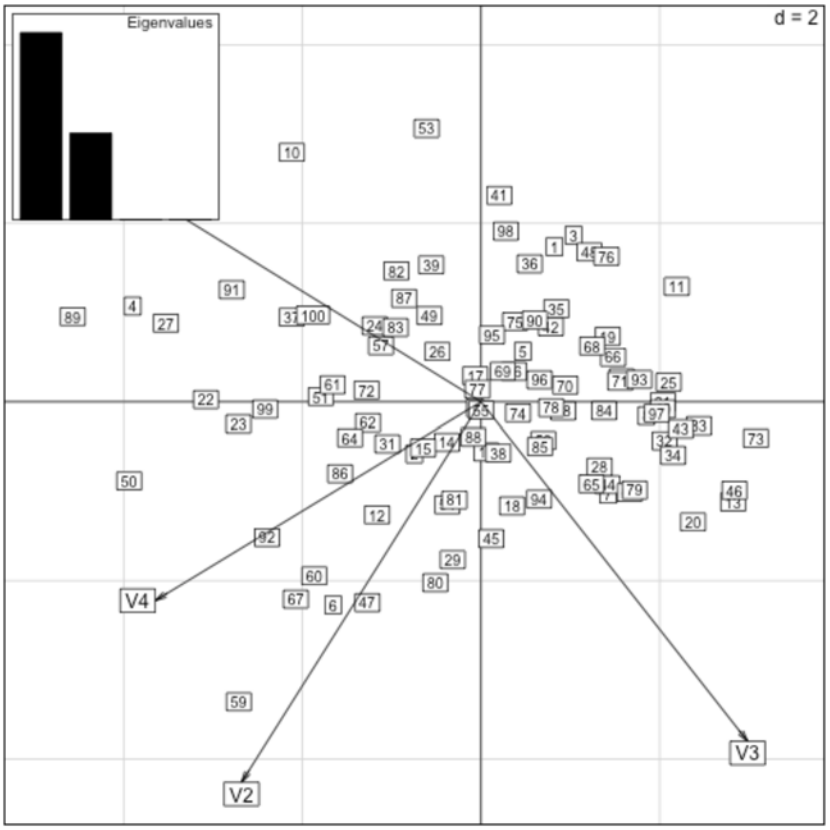
```
Sce$vectors
```

```
##         [,1]    [,2]     [,3]      [,4]
## [1,] -0.1821  0.1380  0.87966 -0.417089
## [2,] -0.3648 -0.9282  0.07263  0.005302
## [3,] -0.5463  0.1748 -0.46144 -0.676784
## [4,] -0.7316  0.2980  0.08938  0.606607
```

SVD can be used to determine the direction of the most variance (and next most variance, and next most variance, …) and how much of the variation is explained by each of those directions. This is exactly the goal of PCA.

When we use PCA to plot data, we only plot the directions in which there is the most change in the data. For example in two dimensional data `Y`, we can easily plot that in two dimensions now and there is very little (actually 0) variation in all other dimensions. By default (using `dudi.pca`), we center the data and then rescale it so each column has a Euclidean norm of 1. Here we show an example and use the default plotting function of the package `ade4` and then a fancy plot from `ggplot2`.

```
Y.pca <- dudi.pca(Y, scannf=F, nf=4)
scatter(Y.pca) # default quick plot
```

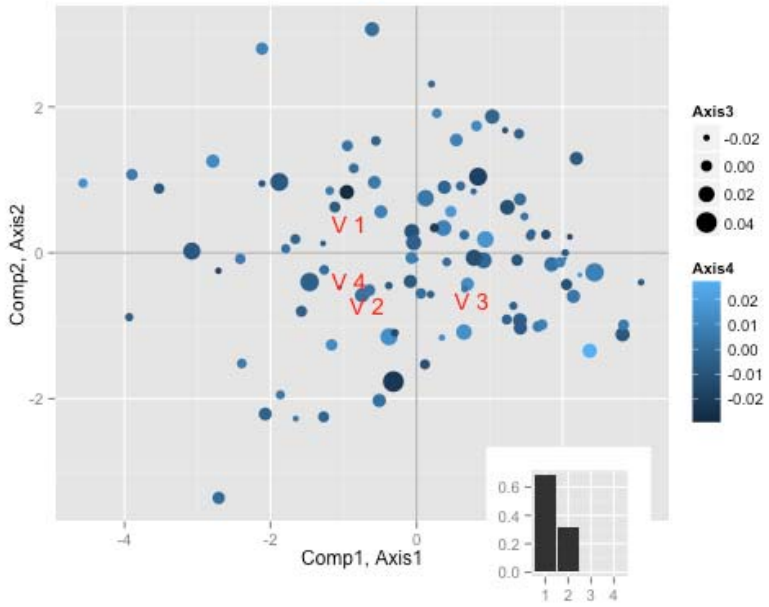


To save time later, we'll save a default plot and a screeplot making function.

```
# set up a plot we'll use later
ppp <- ggplot() + coord_fixed() +
  labs(x="Comp1, Axis1", y="Comp2, Axis2") +
  geom_hline(yintercept=0, col="darkgrey") +
  geom_vline(xintercept=0, col="darkgrey")
# make the scree plot in a viewport
myscree <- function(eigs, x=0.8, y=0.1, just=c("right","bottom")){
  vp <- viewport(x=x, y=y, width=0.2, height=0.2, just=just)
  sp <- qplot(factor(1:length(eigs)), eigs,
              geom="bar", stat="identity") +
    labs(x = NULL, y = NULL)
  print(sp, vp=vp)
}
```

Now actually make the plot

```
ppp + geom_point(data=Y.pca$li, aes(x=Axis1, y=Axis2, size=Axis3, col=Axis4)) +
  geom_text(data=Y.pca$co, aes(x=Comp1, y=Comp2, label=paste("V",1:4)), col="red")
myscree(Y.pca$eig / sum(Y.pca$eig))
```



Here we can see that there is lots of variation in the first two axes (horizontal axis has the most variation, vertical axis has second most variation). There is very little variation in the other axes (notice the scale for axis 3 only goes from -0.02 to 0.04 compared to the scale for axis 1 which goes from -4 to 2).

Centering and Scaling

Why is the default to center and to scale?

Suppose that we did not center. Recall (above) that we can relate PCA to directions with highest covariance. When we calculate sample covariance, we subtract the mean from each observation. If we skip this step (not centering), then the first axis of the PCA would always be pointing towards the center of mass.

Some functions in `R` that calculate the PCA do not center by default. There might be a good reason to not center (e.g., you centered a large dataset already and you are only looking at a subsample), but in general, you should always center your data when doing a PCA.

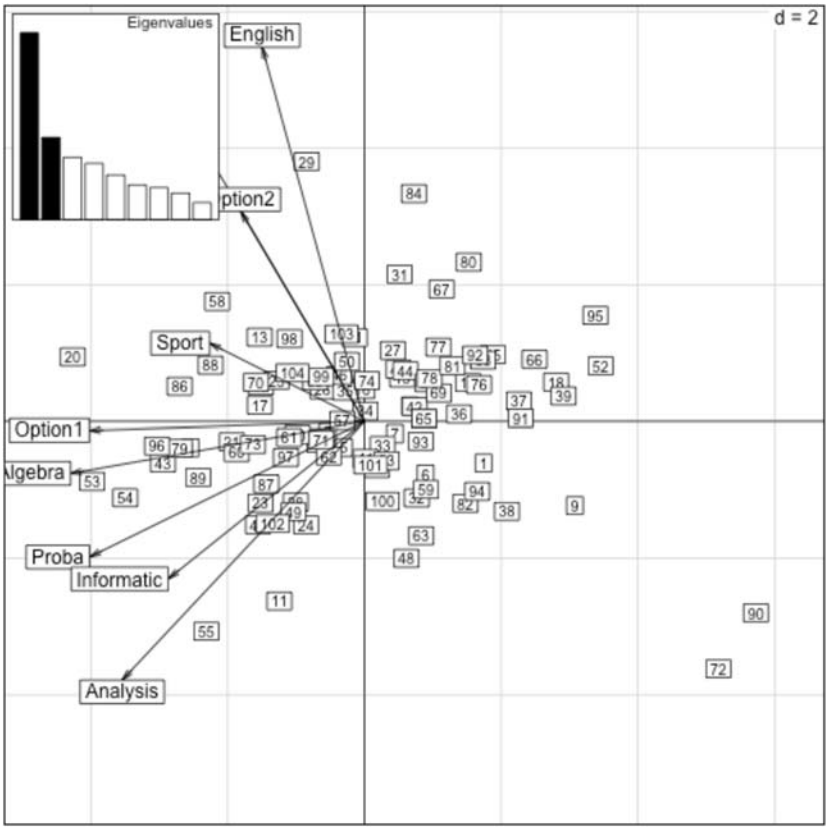
Why is the default to rescale the data?

Recall the difference between correlation and covariance. In correlation you rescale by dividing by the norm of each dimension. This is more in line with what we're interested in. If one of our variable is measured in inches and then we decide to change that measurement to feet, the variance decreases by a factor of \(12^{-2}\). We don't want the result of our PCA to change based on the units a dimension is measured in. To avoid problems like this, we rescale our data so that each dimension has variance 1.

Real example for PCA

The dataset `deug` contains data on 104 French students' scores in 9 subjects: Algebra, Analysis, Proba, Informatic, Economy, Option1, Option2, English, Sport. We will look at the PCA of this data.

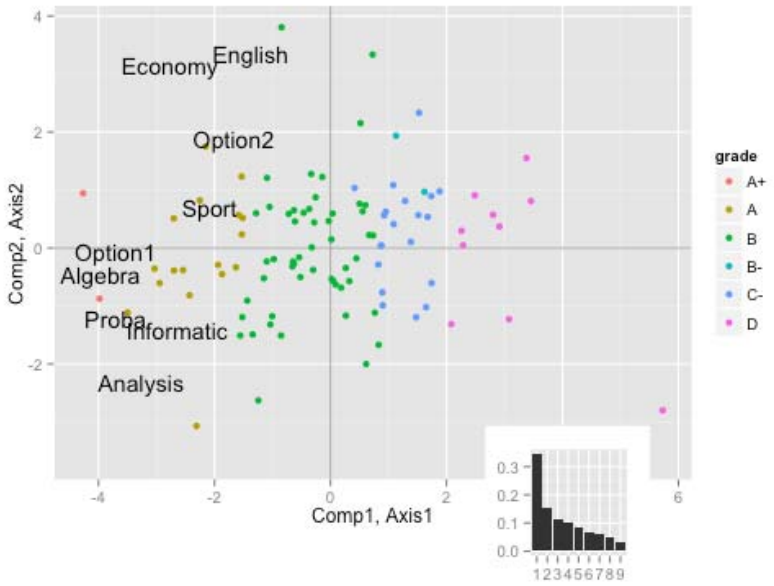
```
data(deug)
pca1 <- dudi.pca(deug$tab, scan = FALSE)
biplot(pca1)
```



The default plotting functions in ade4 are very limitted.

We use `ggplot2` here to show what's going on.

```
grade <- factor(deug$result, levels=c("A+", "A", "B", "B-", "C-", "D"))
pca1.dfs <- data.frame(pca1$li, grade)
# multiply the loadings by 5 so they are more spread out
subject <- names(deug$tab)
pca1.dfl <- data.frame(5*pca1$co[,1:2], subject)
ppp + geom_point(data=pca1.dfs, aes(x=Axis1, y=Axis2, col=grade)) +
  geom_text(data=pca1.dfl, aes(x=Comp1, y=Comp2, label=subject))
myscree(pca1$eig / sum(pca1$eig))
```



Here we see what is called a “size effect”. The first principal component has the largest variation, and it corresponds to an overall effect of how well the students did. Since we also know the students' overall grades, it makes sense that those students who are very far in the direction of all subjects should be the ones who got an “A+” and the ones on the opposite extreme got a “D”.

The second component seems to break up “Analysis” on the one end versus “English” on the other. This component gives us an idea of what the students were good at. Looking at the screeplot though, it is evident that this dimension is not very well defined since there is a small jump in variance explained from this direction to the direction with next most variance.

The screeplot suggests that there is one very important dimension (which corresponds to a size effect). There is not a low rank structure left after accounting for this effect, and plotting this in two dimenions tells us little more than plotting only in one dimension.

Correspondence Analysis

Correspondence analysis takes a different sort of approach to figuring out where data changes the most. Instead of looking for a direction with a high variance, correspondence analysis looks for the directions where the data is “most surprising” from a chi-squared test perspective. This means that correspondence analysis is best suited for count data.

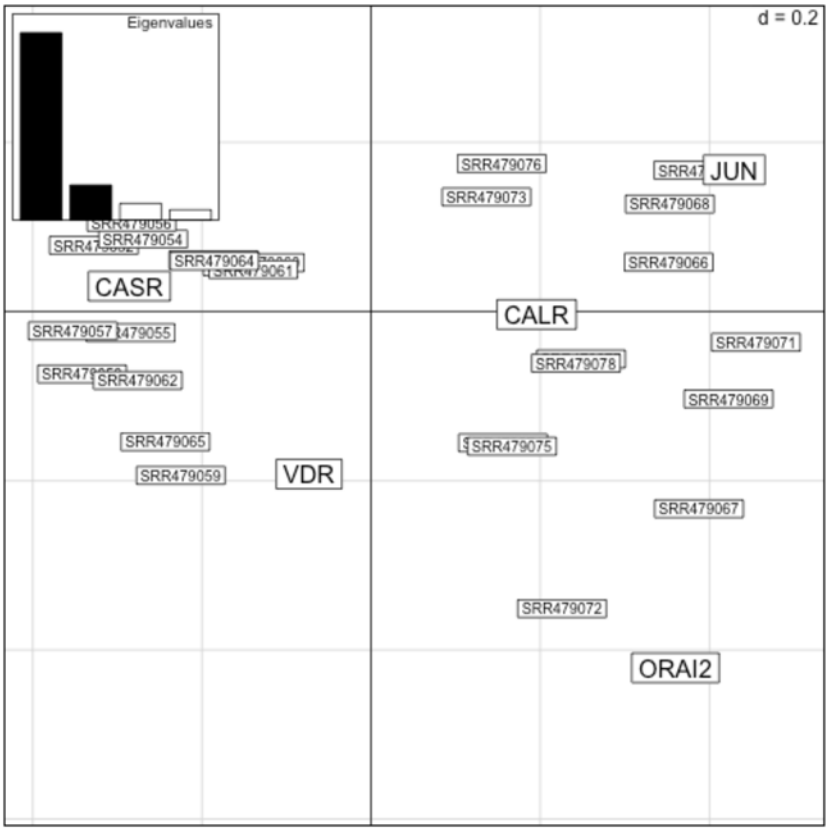
Let's look at an example with the parathyroid data from before. First we load this data and clean it some.

```
library(parathyroid)
data("parathyroidGenes")
# This time we won't look at the estrogen genes because there are very
# few counts of them (no observation has more than 20 counts)
gene.names <- matrix(ncol=3, byrow=T, data=c(
#  "ESR1",  "ENSG00000091831", "estrogen",
#  "ESR2",  "ENSG00000140009", "estrogen",
  "CASR",  "ENSG00000036828", "parathyroid",
  "VDR",   "ENSG00000111424", "parathyroid",
  "JUN",   "ENSG00000177606", "parathyroid",
  "CALR",  "ENSG00000179218", "parathyroid",
  "ORAI2", "ENSG00000160991", "parathyroid"))
gene.counts <- t(counts(parathyroidGenes)[gene.names[,2],])
colnames(gene.counts) <- gene.names[,1]
gene.dat <- parathyroidGenes@phenoData@data
# change dat$time to a numeric 24 and 48 instead of factor 24h and 48h
gene.dat$time <- as.numeric(gsub("h","",(as.character(gene.dat$time))))
```

```
dim(gene.counts)
```

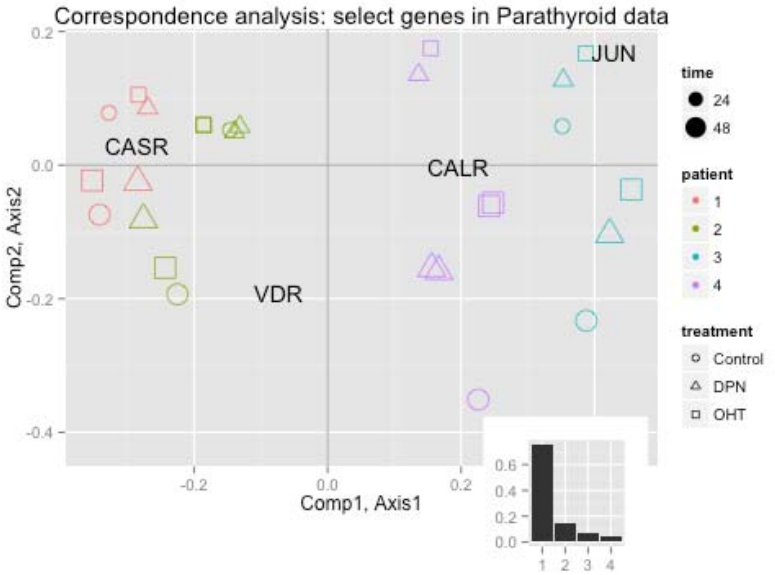
```
## [1] 27  5
```

```
gene.coa <- dudi.coa(gene.counts, scannf=F, nf=2)
scatter(gene.coa)
```



Or you can make it very fancy in ggplot2.

```
# main plot
ppp + geom_point(data=data.frame(gene.coa$li, gene.dat),
             aes(x=Axis1, y=Axis2, col=patient, shape=treatment, size=time)) +
  geom_text(data=gene.coa$co, aes(x=Comp1, y=Comp2, label=gene.names[,1])) +
  scale_size_area(breaks=c(24,48)) +
  scale_shape(solid=F) +
  labs(title="Correspondence analysis: select genes in Parathyroid data")
myscree(gene.coa$eig / sum(gene.coa$eig))
```



Now it's obvious that the first component separates the patients (with patients 1 and 2 slightly overlapping), and the second component separates out 24 hours from 48 hours (perfectly). The first component is obviously the most important though (look at the eigenvalues in the screeplot).

Maybe there's something more important going on with the full structure of the dataset. Above, we cut the data to only focus on the genes we were interested in. We can also look at the full dataset to see if we are able to pick out these interesting genes. First, get rid of the noise from tags with small counts: keep only the genes where at least two observations have more than 1000 reads.

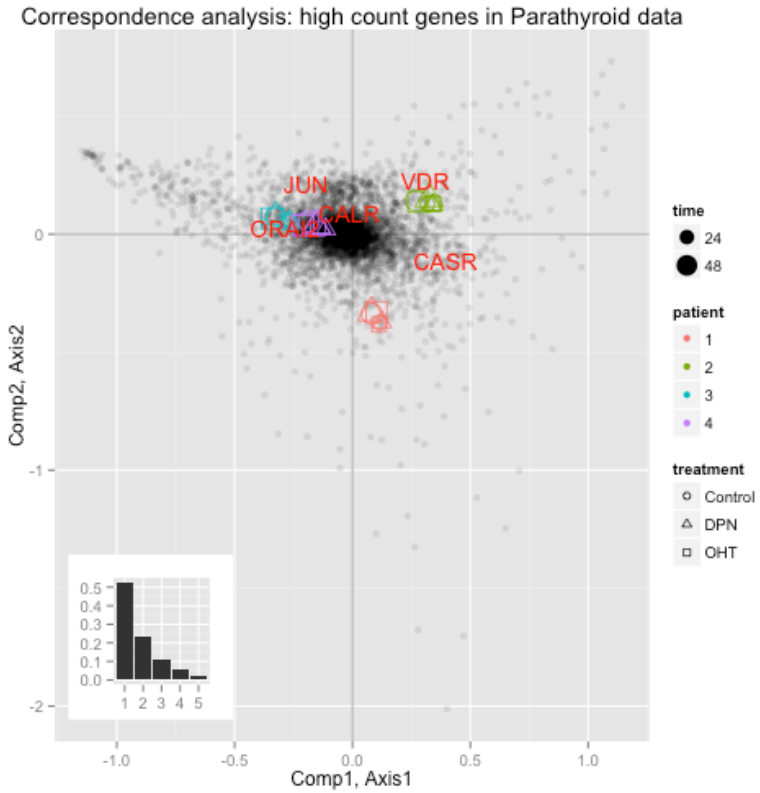
```
cts <- counts(parathyroidGenes)
big.idx <- which(rowSums(cts > 1000) >= 2)
cts <- cts[big.idx,]
dim(cts)
```

```
## [1] 4724   27
```

Note that now we have the samples as the columns and the genes on the rows. Since we are looking at where this data is surprising from a chi-squared test perspective, it doesn't matter that we transposed the data.

Here, we're looking at the case with lots of genes and seeing if we can pick out the important ones. Note that the columns of `cts` are the samples and the rows are the genes (that's the opposite of the case above for `gene.counts`.

```
cts.coa <- dudi.coa(cts, scannf=F, nf=2)
ppp + geom_point(data=cts.coa$li, aes(x=Axis1, y=Axis2), alpha=0.1) +
  geom_point(data=data.frame(cts.coa$co, gene.dat),
             aes(x=Comp1, y=Comp2, col=patient, shape=treatment, size=time)) +
  geom_text(data=cts.coa$li[gene.names[,2],],
             aes(x=Axis1, y=Axis2), label=gene.names[,1], col="Red") +
  scale_size_area(breaks=c(24,48)) +
  scale_shape(solid=F) +
  labs(title= "Correspondence analysis: high count genes in Parathyroid data")
myscree(cts.coa$eig[1:5] / sum(cts.coa$eig),
        x=0.1, y=0.1, just=c("left", "bottom"))
```



Again we see that with all this additional data, patients 3 and 4 are near JUN, ORAI2, and CALR. Patient 2 is near VDR. Before it looked like patient 1 was very near CASR, but when we include all this additional data, the top components and axes suggest that patient 1 is far from all 5 of the genes labelled as interesting by the paper.

Multidimensional Scaling

In some cases, we only have a distance matrix to work with, but we want to visualize the data so that points which are close to each other in the distance matrix appear close to each other in a figure. One way to do this is with multidimensional scaling.

Here we look at a dataset from the `phyloseq` package in R. This data has a phylogenetic tree associated with it, so we will calculate the UniFrac distance based on that, and then look at the data.

First we will read in the `GlobalPatterns` dataset which has an OTU table for 26 sample locations (12 of which are from humans) and 19216 taxa. Since my computer is slow, we will only consider the top 200 taxa from the top 5 phyla (this will result in only 182 taxa).

```
data(GlobalPatterns)
# prune OTUs that are not present in at least one sample
GP <- prune_taxa(taxa_sums(GlobalPatterns) > 0, GlobalPatterns)
# Define a human-associated versus non-human categorical variable:
human <- get_variable(GP, "SampleType") %in% c("Feces", "Mock", "Skin", "Tongue")
# Add new human variable to sample data:
sample_data(GP)$human <- factor(human)
# Take a subset of the GP dataset, top 200 species
topsp <- names(sort(taxa_sums(GlobalPatterns), TRUE)[1:200])
GP <- prune_taxa(topsp, GlobalPatterns)
# Subset further to top 5 phyla, among the top 200 OTUs.
top5ph <- sort(tapply(taxa_sums(GP), tax_table(GP)[, "Phylum"], sum), decreasing = TRUE)[1:5]
GP <- subset_taxa(GP, Phylum %in% names(top5ph))
GP
```

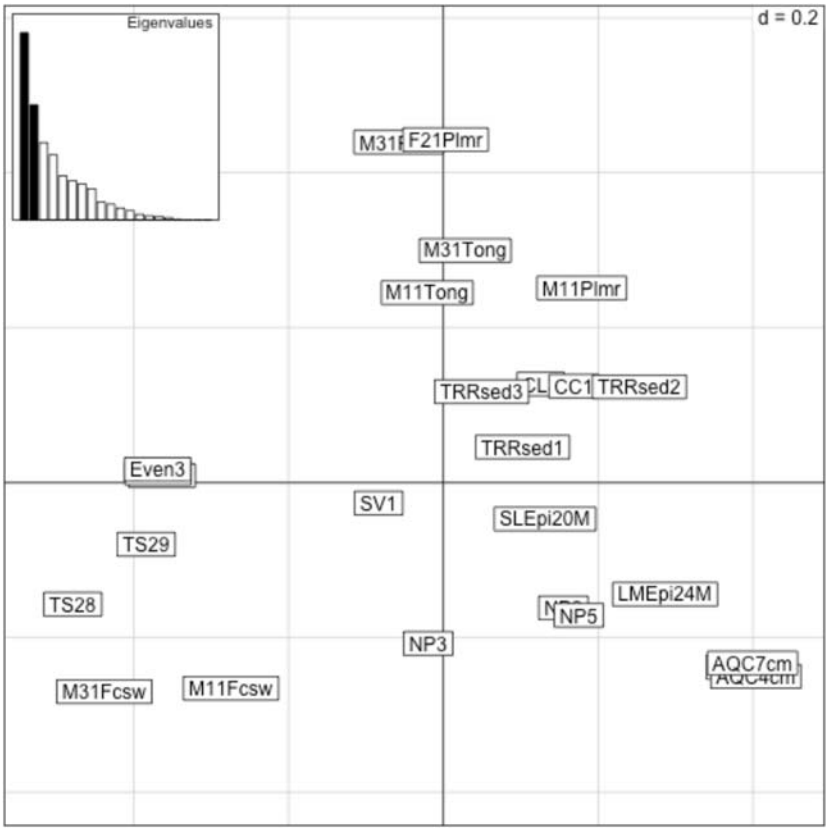
```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 182 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 182 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 182 tips and 181 internal nodes ]
```

Now we will calculate the unifrac distance, and do an MDS. The figure below uses the default plotting function in ade4.

```
GPUF <- phyloseq::distance(GP, method="unifrac", weighted=TRUE) # Weighted UniFrac
set.seed(3) # MDS is only unique up to sign
mds.gpuf <- dudi.pco(GPUF, scannf=F)
```

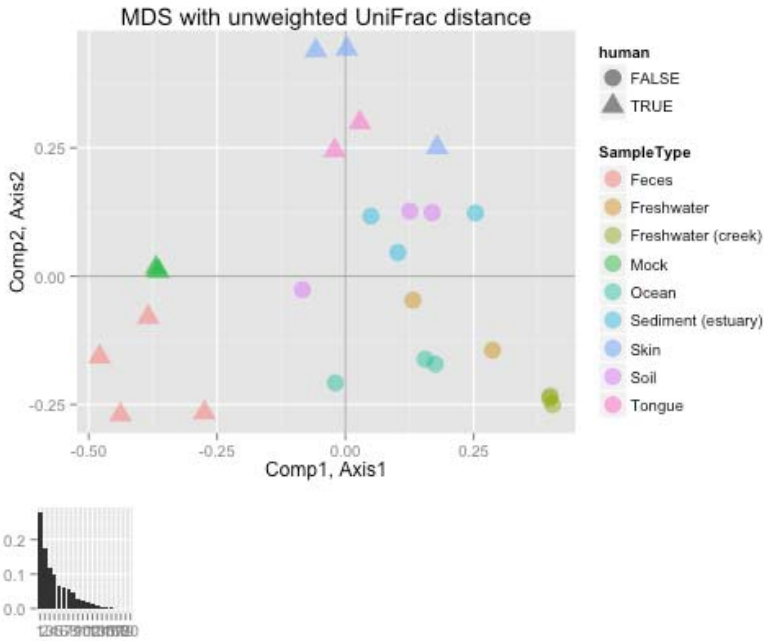
```
## Warning: Non euclidean distance
```

```
scatter(mds.gpuf)
```



We can make this look nicer and include more information using ggplot to add colors and shapes.

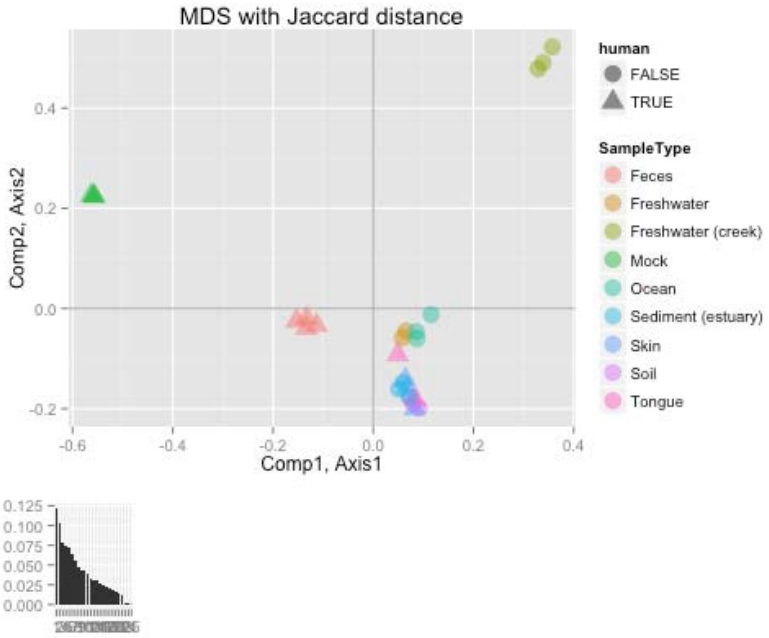
```
ppp + geom_point(data=data.frame(mds.gpuf$li, sample_data(GP)),
             aes(x=A1, y=A2, col=SampleType, shape=human), size=5, alpha=0.5) +
  labs(title="MDS with unweighted UniFrac distance")
myscree(mds.gpuf$eig / sum(mds.gpuf$eig), x=0, y=0, just=c("left", "bottom"))
```



In this case, it is easy to separate the human and non-human samples since we can draw a box around all the non-human samples which does not include any of the human samples. Here we see that the simulated “mock human” samples are close to the feces, and the tongue and skin are close to each other. The freshwater, freshwater creek, and ocean are all together. The soil and sediment are near the middle (close to the freshwater creek). This analysis is rather satisfying since it agrees with what we might have expected before actually plotting the data.

Since this data is annotated with more than just the phylogenetic tree, we can also make a distance on other characteristics. Some choices can be found in `help(vegdist)`. Here, we use the jaccard distance (note that in `vegdist`, Jaccard index is computed as 2B/(1+B), where B is Bray–Curtis dissimilarity).

```
GPJA <- phyloseq::distance(GP, "jaccard") # vegdist jaccard
mds.gpja <- dudi.pco(GPJA, scannf=F)
ppp + geom_point(data=data.frame(mds.gpja$li, sample_data(GP)),
             aes(x=A1, y=A2, col=SampleType, shape=human), size=5, alpha=0.5) +
    labs(title="MDS with Jaccard distance")
myscree(mds.gpja$eig / sum(mds.gpja$eig), x=0, y=0, just=c("left","bottom"))
```



Here we see something very different. This is to show that the choice of a distance metric is very important when working with data. Choosing the right metric can provide useful insights while others do not.

Exploratory Analysis Versus Predictive Analysis

In this lab, we have focused entirely on exploratory methods. In data with a small number of samples, this is often an important first step. By doing this sort of analysis, we can make informed hypotheses and do experiments to test them.

In addition to just doing exploratory methods, we can also consider predictive analysis such as regression and classification.

Questions

Answer the questions on OHMS “Lab 5: Multivariate”. Go to https://web.stanford.edu/class/bios221/cgi-bin/index.cgi/ to answer some questions. You may NOT work with other students to answer the questions on OHMS. You will need a Stanford ID to log in to OHMS.