Lab 5

Clustering and Ordination with Distances

This lab introduces a new set of techniques that are based on distance matrices that indicate similarity between observations. All these techniques require a two step process: (1) create a matrix of similarity between observations, choosing among dozens of ecological, genetic, or mathematical distance measures, and (2) using this matrix to ordinate or cluster the observations, choosing again among dozens of ordination and clustering techniques. This leaves you with (dozens)^2 possible ways to view your data.

There is no objective way to guide the choice of distance matrix and the choice of ordination and clustering techniques. Generally, Euclidean and Mahalanobis distances work well for normally distributed environmental data. The Bray-Curtis distance is a favorite for community ecologists to describe similarity in species composition in sample plots. The Bray-Curtis distance is a very robust distance measure. If you have an intuitive sense of similarity of your study subjects (or communities) spot-check the B-C distance matrix. The measure just seems to work on anything.

It is good practice to explore several distance metrics and clustering or ordination techniques: if they give you completely different answers, then likely your observations do not fall into clear groups. You can also test for significant differences among your groups (next lab) to confirm that your classification is not a sampling artifact based on random variation in your measurements.

5.1. Creating and importing distance matrices

Download the dataset AB_Climate_Means.csv available at http://tinyurl.com/mv690/lab/5/data. These are means of climate variables for ecosystem (natural subregions of Alberta) that were derived from the spatial data you are familiar with from the previous labs. We want to explore similarity with this simpler multivariate climate dataset. (Note: variable explanations on last page!)

- To set yourself up in R as usual, starting R from an empty workspace in a working directory, and import the dataset AB_Climate_Means.csv. The option row.names=1 converts the ecosystem column to row names, which facilitates automatic labeling. Note that row names must be unique. Here, we don't use the biome column, whch does not belong to the climate data, so we need to drop it.

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
head(dat1)
dat1=dat1[,-1]
head(dat1)
```

- An important consideration before applying distance metrics is whether or not to scale your data. If you have different units (like in this case temperature in degree and precipitation in mm), but you want each variable to have an equal influence on your distance metric, then you should standardize your data (i.e., for each variable column, subtract it's mean and divide by the standard deviation). You may want to use other transformations used in your field of research (e.g. "Hellinger" or "Wisconsin" in the field of community ecology, provided by the function `decostand` of the vegan package)

```
dat2=data.frame(scale(dat1))
head(dat2)
```

- Now, we use the distance function to create distance matrices. The default for `dist()` function is the normal Euclidean distance. You can exaggerate or reduce the relative importance of the largest distances with transformations of the distance matrix. Squares or the square root are commonly used:

```
euclid=dist(dat2, method ="euclidean")
euclid_sq=euclid^2
euclid_sqrt=sqrt(euclid)
```

- For ecologists, there are other important distance measure. For this, you have to install the R package “ecodist”, which allows you to calculate Bray-Curtis and Mahalanobis distances. The `distance()` function of “ecodist” actually calculates the squared Mahalanobis, so you have to take the square root to get the original.

```
library(ecodist)
braycurtis=distance(dat1, "bray-curtis")
mahal_sq=distance(dat1, "mahal")
mahal=sqrt(mahal_sq)
mahal   # check how a distance matrix looks like
```

- Distance matrices are a special data format in R. If you ever need to import a distance matrix or convert a data table to a distance matrix, you can do it with the code below. There are, for example, specialized distance metrics for geneticists to describe similarities in DNA sequences, etc. that may come from a different package or a different program. You can practice this with the Custom_Distance_Matrix.csv file that is included in the data package:

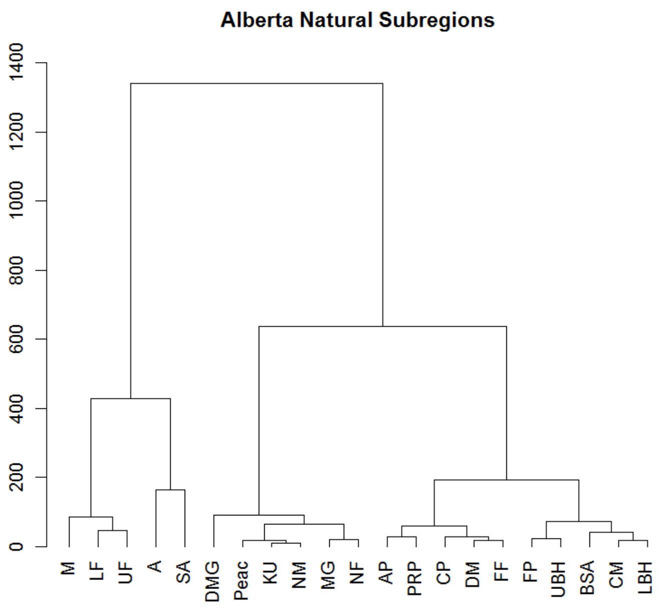
```
custom1=read.csv("Custom_Distance_Matrix.csv")
head(custom1) # check it
rownames(custom1)= custom1$ID; # row name modification as above
custom1=custom1[,-1] # drop ID column
custom2=as.dist(custom1) # conversion to distance matrix
head(custom1) # check it
```

5.2. Agglomerative cluster analysis

OK, now that you know everything about generating and importing distance matrices, let’s use them for building dendrograms. Below, I call the distance matrix uniformly “**dm**”. You have to replace this with the distance matrix of your choice that you generated above (i.e., **mahal**, **mahal_sq**, **euclid**, **euclid_sq**, **euclid_sqrt** or **braycurtis**).

- The code below builds dendrograms. `hclust` stands for hierarchical cluster analysis. There are many methods. Ward’s method “`ward.D`”, “`ward.D2`”, and “`centroid`” are widely used and usually yields good results, but you may try: "`single`", "`complete`", "`average`", "`mcquitty`", or "`median`" as well.

```
tree=hclust(dm,
method="ward.D")
plot(tree, hang=-1,
main="Alberta Natural
Subregions")
```



- Get a feel for the robustness of cluster analysis by trying various distances, transformations, and clustering methods. Save a few dendrograms and make up your mind if they make biological sense. For reference, the Alberta Natural Subregion system is given on the last page. For this dataset, the Mahalanobis distance is likely appropriate because some of the climate variables are correlated.

5.3. Cluster Analysis with significance testing

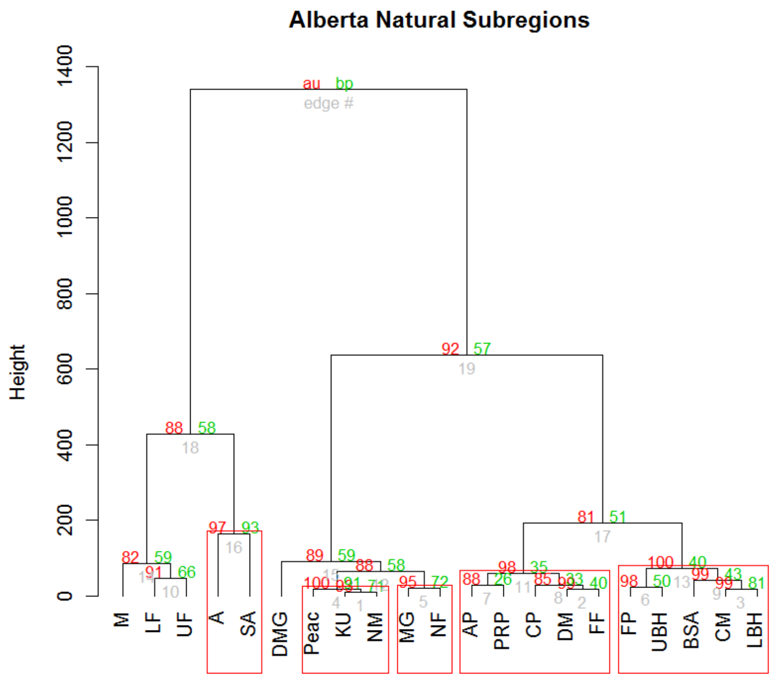
You can run a bootstrap-version of cluster analysis that evaluates how consistently the same clusters appear over hundreds or thousands of runs with randomly sub-sampled dataset omitting one or few variables (columns) and observations (row) at a time

- Install the package `pvclust` and try out the following code.

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
dat1=dat1[,-1]
head(dat1)

dat1t=t(dat1) # transpose dataset
head(dat1t) # check it

# Install package "pvclust"
library(pvclust)

# Be patient the next step may take a moment with larger datasets
tree=pvclust(dat1t, method.hclust="ward.D", nboot=1000,
method.dist="euclidean")

# Create the dendrogram with p values
plot(tree, hang=-1, main="Alberta Natural Subregions")

# add rectangles around groups highly supported by the data
pvrect(tree, alpha=.95)
```

- Groups with high AU values, for example >0.95, are strongly supported by data. This means they really are very similar units that form a natural cluster. AU means “Approximately Unbiased P-value” whereas BP refers to raw “Bootstrap Probabilities” before statistical adjustments.

  For interpretation check what the ecosystem abbreviations stand for on the last page.



5.4. Nonmetric multidimensional scaling (NMDS)

Instead of using a dendrogram, we can also use ordination techniques. NMDS is a very robust technique for all kinds of normally and non-normally distributed data, including presence/absence data. Similarity is implied by proximity:

- For this analysis we want to color by the BIOME variable, so we'll pull that out before deletion:

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
BIOME= dat1[,1]
dat1=dat1[,-1]
head(dat1)
BIOME

library(ecodist)
dm=distance(scale(dat1), "euclidean") # standardized Euclidean
nmds_out=nmds(dm, mindim=2, maxdim=2) # runs NMDS
scores=nmds.min(nmds_out) # generate scores
nmds_out$stress # the last value indicates the final stress
```

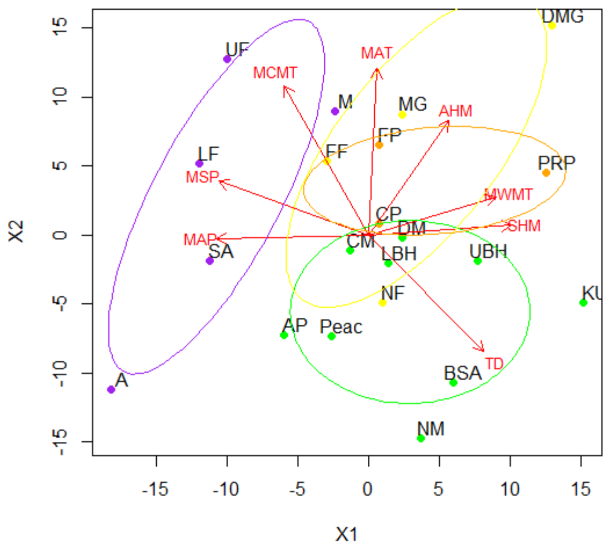
- The stress value by itself is not informative, but it should be stable (i.e. for the last permutations, and you should look at the stress values of 1, 2, 3, 4, 5, 6 or so dimensions (modify maxdim). A scree-plot of stress values over the number of dimensions will tell you how many dimensions you need to consider. Let's look at the first 2 dimensions.

- In the plot below, we color by biome, guided by the first command that shows the order of biomes. Then create a color legend in the same order, and plot with ecosystem labels. The +0.2 offsets the labels, so you can still see the points. For different distance metrics or datasets, you may need to adjust those offset values. Vectors are fitted as usual.

```
mygroups = sort(unique(BIOME)); mygroups       # shows order
mycol = c("green","yellow","purple","orange")  # color in order
plot(scores, pch=19, col=mycol[as.factor(BIOME)])
text(scores+0.2, labels=row.names(dat1))       # add offset labels
vectors = vf(scores, dat1, nperm=10)           # calculate vectors
plot(vectors, len=0.1, col="red")              # add vectors
```

- The overlay of vectors indicate how the original variables are correlated with observations. For example the "DMG" ecosystem (Dry Mixed Grass) is associated with high mean annual temperature (MAT) and dryness (AHM).

- You can guide the reader eye by adding ellipses that represent groups. The basic command is requires your coordinates (`scores`) the color criteria (`BIOME`), the color scheme (`mycol`) and the group for which you want to draw ellipses (`mygroups`). The option for confidence interval `conf` allows to scale the size of the ellipses to your liking. 0.6 means that if the data were multivariate normally distributed 60% of observations would fall within:



```
library(vegan)
ordiellipse(scores, BIOME, conf=0.6, col=mycol, show.groups=mygroups)
```

If you run the NMDS code multiple times, you may often (but not always) notice that the plots differ substantially in appearance. A big part of this is that the canvas is randomly rotated to a new angle every time. As a consequence, loadings and variance explained by X1 and X2 are not particularly informative. Variance explained by X1 and X2 is not informative for another reason: NMDS is not an orthogonal rotation, and because X1 and X2 are not at right angles (uncorrelated), only the cumulative variance explained should be reported.

Besides the random rotation, you will also see that the sample points shift relative to each other and relative to the vectors by small amounts (hopefully). Thus, NMDS really provides a different ordination every time you run it. There is nothing inherently wrong with this. It is equivalent to looking at your dataset from slightly different angles as in PCA versus FACTOR analysis. If you have a run that has substantially lower stress than all others, you should probably pick that one. Otherwise, you are free to choose a run that you like for esthetic reasons.

5.5. MetaMDS

If you feel uncomfortable with a subjective decision what run you like best or what run is easiest to interpret, the vegan package offers a metaMDS function that executes multiple runs and looks for stable configurations. One advantage is that this package normally produces more repeatable results in that you get the same or a similar ordination output every time.

- In addition to the number of dimensions (`k`), you can also specify the number of random start configurations (`trymax`) that metaMDS executes in search of a stable solution. If it does not find a stable solution, it will use PCA as starting point for consistency. Check `?metaNMDS` for all options in this package. You can see that metaMDS will also calculate distance matrices for you, but for better control and transparency let’s keep doing it manually:

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
BIOME= dat1[,1]
dat1=dat1[,-1]
head(dat1)
BIOME

library(ecodist)
dm=distance(scale(dat1), "euclidean") # standardized Euclidean

library(vegan)
out1 = metaMDS(dm, k=2, trymax=500)
out1

mygroups = sort(unique(BIOME)); mygroups      # shows order
mycol = c("green","yellow","purple","orange") # color in order
plot(scores, pch=19, col=mycol[as.factor(BIOME)])
text(scores+0.2, labels=row.names(dat1))      # add offset labels
vectors = vf(scores, dat1, nperm=10)          # calculate vectors
plot(vectors, len=0.1, col="red")             # add vectors
ordiellipse(scores, BIOME, conf=0.6, col=mycol,
            show.groups=mygroups)             # add ellipses
```

5.6. Principal Coordinate Analysis (PCoA)

Another well regarded and well behaved ordination technique is Principal Coordinate Analysis (PCoA). Like NMDS it is an ordination that is based on a distance matrix of your choice, so you are free to use a distance measure that is suits your data, i.e. Bray-Curtis for species community data, which I am using in the example below (just because I can, and because I want to use something that's non-Euclidean, which is actually the point of this analysis).

The difference to NMDS is that I do not constrain this ordination to just few or two dimensions. Instead, I allow as many dimensions as required to honor the relative position of all points in my dataset, and that is (to be mathematically 100% on the safe side), n-1 dimensions. So, if you have a dataset with n=50 observations (rows) and 10 variables (columns), we allow 49 dimensions for this ordination.

This makes the ordination procedure a no-brainer because I don't need to make any compromises. There is zero stress in my ordination and no complicated algorithms to minimize stress are required. That said, if your objective is to reduce complexity, we have just made things substantially worse, going from 10 original dimensions (or variables) to 49.

However, the party trick in the second step: I run a regular PCA on that 49 dimensional ordination. So you can think of PCoA as a representation of non-Euclidean data (i.e. since you normally would not use Euclidean distances for your distance matrix) in a Euclidean space.

- Here is the full code to run a PCoA with all customizations that we got used to, rather than using the quick and dirty biplot functions. Sometimes, you may run into trouble with negative Eigenvalues when trying to rotate your n-1 dimensional distance matrix, and there are some corrections available for that (see the `?pcoa` help file for references and details):

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
BIOME= dat1[,1]
dat1=dat1[,-1]
head(dat1)
BIOME

library(ecodist)
dm = bcdist(dat1)

library(ape)
out1 = pcoa(dm, correction="none")
scores = out1$vectors[,1:2]
head(scores)

mygroups = sort(unique(BIOME)); mygroups      # shows order
mycol = c("green","yellow","purple","orange") # color in order
plot(scores, pch=19, col=mycol[as.factor(BIOME)])
text(scores+0.003, labels=row.names(dat1))    # add offset labels
vectors = vf(scores, dat1, nperm=10)          # calculate vectors
plot(vectors, len=0.1, col="red")             # add vectors
ordiellipse(scores, BIOME, conf=0.6, col=mycol,
            show.groups=mygroups)             # add ellipses
```

5.7. Divisive cluster analysis (e.g. k-means)

Another way to execute a cluster analysis is a top-down approach, where you start with the entire dataset at once, and divide it into groups. In that case you don't build a dendrogram up from individual observations through agglomeration like we have done in sections 5.2 and 5.3 above As such, a dendrogram is also not a visualization option. The outcome of a divisive cluster analysis is just a vector of cluster membership that you can use to visualize in an ordination, as we have done in sections 5.4 to 5.6 above.

There are a large number of divisive cluster algorithms. Popular ones include k-means (e.g. function `kmeans` in the stats package) and PAM (e.g. function `pam` in the cluster package), which both work very similarly. One partitions around averages, the other around a slightly different metric of central tendency in multivariate space, medoids. Another well regarded and relatively new clustering technique is t-distributed stochastic neighbor embedding (package `tsne`), which works well for very high dimensionality datasets with hundreds or thousands of variables.

All of these methods have in common that you need to specify a priori how many clusters you want as your outcome. That may be difficult to decide for you, but it's even harder for a computer algorithm to decide that. There are some useful plots that can help you with that decision, though.

A practical use-case for divisive clustering is if you have datasets with many experimental or sampling units (rows). Dendrograms can get a bit crowded and ugly with more than one or two dozen observations. You may be more interested in a prescribed number of higher level clusters.

- The code below executes a k-means divisive clustering algorithm. It is a recursive code, that starts with a specified number of points (it just uses a random selection of k rows from your dataset), then calculates which of all the other points are closest to it and assigns them to this cluster. The second step is a re-calculation of the cluster center based on the newly assigned points. Rinse and repeat, i.e. based on my new cluster center, I recalculate which observations are closest, etc.

- The option `centers=5` presets the number of clusters, `iter.max=50` determines how many times you would repeat the two steps at most. The option `nstart=25` determines how many repeat runs of the whole analysis you want to do with different starting points to find the best solution.

```
dat1=read.csv("AB_Climate_Means.csv", row.names=1)
head(dat1)
dat1=dat1[,-1]
head(dat1)
dm=dist(scale(dat1), method ="euclidean")

library(stats)
kmeans5=kmeans(dm, centers=5, iter.max=50, nstart=25)
kmeans5$size; kmeans5$centers; kmeans5$cluster
```

- Since we can't plot a dendrogram, you can visualize your clusters via an ordination. Choose the most appropriate distance metric and ordination technique according to your type of data and your objectives (PCA, Factor, CanDisk, NMDS, or PCoA are all options that we have previously covered). I am going with the metaMDS here, using the same **dm** as for k-menas:

```
out1=metaMDS(dm, k=2)
scores=out1$points[,1:2]
```

```
mycol=c("pink2","yellow2","purple","green","orange")
plot(scores, pch=19, col=mycol[as.factor(kmeans5$cluster)])
text(scores+0.2, labels=row.names(dat1))

library(vegan)
mygroups = sort(unique(kmeans5$cluster))
ordiellipse(scores, as.factor(kmeans5$cluster),
   conf=0.6, col=mycol, show.groups=mygroups)
```

- Here is some additional analysis that you can try to help you determine the number of clusters that may be best. This does scree plots based on different criteria. The `method="wss"` will calculate the within cluster sums of squares (i.e. variance). More clusters means less variance within clusters, which is good, but there will be a point of diminishing returns where more clusters don't do much. The other methods evaluate the shape and distance between natural clusters. Sometimes, you don't have natural clusters where these metrics will not provide much guidance. This works for other cluster techniques as well, e.g. `FUNcluster=pam` . Check: `?fviz_nbclust`

```
library(factoextra)
dat2=data.frame(scale(dat1)) # forces a scaled Euclidean
fviz_nbclust(dat2, FUNcluster=kmeans, method="wss")
fviz_nbclust(dat2, FUNcluster=kmeans, method="silhouette")
fviz_nbclust(dat2, FUNcluster=kmeans, method="gap_stat")
```

- Since there is some space left on this page, another visualization that's built into factoextra using ggplot2. I don't recommend it, because it makes choices for you. You don't have control over the distance metric and ordination technique. It does a PCA only. For customization options, check `?fviz_cluster`, `?ggscatter` and `?ggpar`.

```
fviz_cluster(kmeans5, data=dat2, shape=1,
    show.clust.cent=T, ellipse.type="t",
    ellipse.level=0.7, ellipse.alpha=0.1,
    palette=c("pink3","yellow3","purple","green","orange"),
    ggtheme=theme_classic())
```

Some Reference information, so that you can interpret the results more easily:

Abbreviations of Ecosystems:

Mountains	
Alpine	A
Subalpine	SA
Montane	M
Upper Foothills	UF
Lower Foothills	LF
Grasslands	
Dry Mixedgrass	DMG
Mixedgrass	MG
Northern Fescue	NF
Foothills Fescue	FF
Parklands	
Foothills Parkland	FP
Central Parkland	CP
Peace River Parkland	PRP
Boreal Forest	
Dry Mixedwood	DM
Central Mixedwood	CM
Lower Boreal Highlands	LBH
Upper Boreal Highlands	UBH
Athabasca Plain	AP
Peace–Athabasca Delta	Peac
Northern Mixedwood	NM
Boreal Subarctic	BSA

Abbreviations of Climate Variables:

MAT: mean annual temperature (°C),
MWMT: mean warmest month temperature (°C),
MCMT: mean coldest month temperature (°C),
TD: temperature difference between MWMT and MCMT, or continentality (°C),
MAP: mean annual precipitation (mm),
MSP: mean annual summer (May to Sept.) precipitation (mm),
AHM: annual heat:moisture index (MAT+10)/(MAP/1000))
SHM: summer heat:moisture index ((MWMT)/(MSP/1000))
DD0: degree-days below 0°C, chilling degree-days
DD5: degree-days above 5°C, growing degree-days
NFFD: the number of frost-free days
FFP: frost-free period
BFFP: the Julian date on which FFP begins
EFFP: the Julian date on which FFP ends
PAS: precipitation as snow (mm)