Lab 2

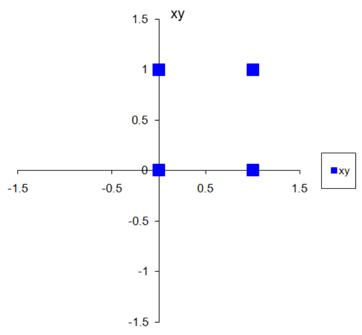
Rotation & Principal Component Analysis

Many classical multivariate techniques rely on rotating a dataset in multiple dimensions and then looking at the results through a 2-dimensional "window" (e.g. principal component analysis, factor analysis, discriminant analysis, or redundancy analysis). Mathematically, this works with matrix algebra, and if you like, you can analyze your data with published matrix algebra formulas. On the website you can find a paper by Wollenberg for programming a redundancy analysis, and below a very simple example to program the rotation of a coordinate system that requires just one matrix multiplication.

2.1. Matrix rotation

- In Excel create a dataset with columns x,y,z and a couple of rows of data (the sample dataset below represents the 8 corners of a 3D cube). Make a 2D scatter plot of 2 variables (e.g. x and y)

x	y	z
1	0	0
0	1	0
0	0	0
1	1	0
1	0	1
0	1	1
0	0	1
1	1	1



- If you use the example above, choose the z-rotation matrix below to rotate the "blue box" around the z-axis that represents the third dimension (sticking out of the page toward you). Use a spreadsheet cell to enter an angle in degrees (e.g. 30 entered in cell **E2**), another cell (e.g. **F2**) that converts the angle to radians =RADIANS(**E2**).
- Enter one of the rotation matrices below. If you want to rotate aroud the z-axis, use the first:

Rotation around z-axis

cos(**F2**)	-sin(**F2**)	0
sin(**F2**)	cos(**F2**)	0
0	0	1

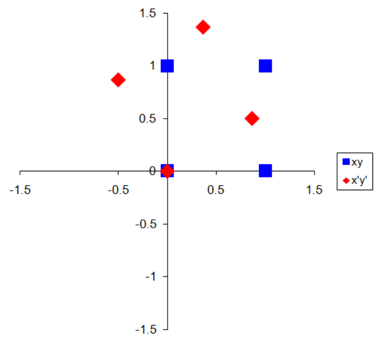
Rotation around y-axis

cos(**F2**)	0	sin(**F2**)
0	1	0
-sin(**F2**)	0	cos(**F2**)

Rotation around x-axis

1	0	0
0	cos(**F2**)	-sin(**F2**)
0	sin(**F2**)	cos(**F2**)

- Highlight the data matrix (without the x, y, z header). From the ribbon, choose "Formula" > "Define Name" > type "DataMatrix", repeat for "RotationMatrix"
- At a new cell insert the formula for multiplying matrices: =MMULT(DataMatrix, RotationMatrix). This will only give one value.
- Click on that field and hold the mouse button, hold the F2 button, drag to the same dimensions as the data matrix, release mouse, release F2
- Now, press and hold <CTRL><SHIFT>, hit <ENTER>, Done!
- Plot the original and rotated points in an Excel graph. You can now rotate the box to any degree by changing the angle in cell **E2**.



- You could now take the red box as input and rotate it around the y-axis and subsequently the x-axis. That's essentially what principal component analysis and other techniques do.

2.2. Matrix rotation in R

- You can program the same in R. If you import the data matrix as **d** and the rotation matrix as **r**, the code below will do the multiplication and plots. Of course, you could make the code more flexible for different angles by calculating the rotation matrix with `cos()` and `sin()` functions (rather than importing **r**). Try, if you have time to play with this.

```
d2=as.matrix(d) %*% as.matrix(r)
plot(d[,1],d[,2], xlim=c(-0.5, 1.5), ylim=c(-0.5, 1.5), col="blue")
points(d2[,1],d2[,2], col="red")
```

2.3. Principal component analysis with R

Let's start with a simple example of a PCA that uses a sample dataset that comes with the R base package. This data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states. Also given is the percent of the population living in urban areas. This is a fair amount of data and not trivial to effectively visualize.

You can use Principal Component Analysis to understand the content of this dataset, first by rotating the dataset so that most variance in the original variables is explained, then by overlaying the component loadings as vectors to indicate correlations with the original variables.

- First, we can make ourselves familiar with the raw data using these basic commands:

```
data(USArrests)
head(USArrests)
View(USArrests)
pairs(USArrests)
```

- Next, let's run a principal component analysis on this dataset. The syntax is very simple: we specify an output object, which can have any name, run the `princomp` function, and specify that we want to work with the correlation matrix rather than the covariance matix. The latter would require the option `cor=F`, and then the variables would retain their original units for rotation rather than being normalized.

```
out1=princomp(USArrests, cor=T)
```

- While many software packages will return pages of output for every analysis, R tends to return nothing, and you have to specifically extract the information you care about from the output object. Below we request principal component loading (correlations with original variables), component scores (new coordinate of points after rotation), and variance explained by the principal components (with the summary statement). Princomp does not report Eigenvalues, but you can to calculate them manually, and you can convert them to variance explained by dividing them by the number of variables.

```
out1$loadings
out1$score
summary(out1)
eigen(cor(USArrests))
eigen(cor(USArrests))$values/4
```

- Finally, we can make a quick plot of the rotated points, choosing the first and second principal component as axes, which we have seen above, explain much of the total variance. The bi-plot also adds the component loadings as vectors.

```
biplot(out1), choices=c(1,2))
```

- Can you now very concisely interpret what information the "USArrests" data contains?

2.4. Visualizing a more challenging spatial dataset for analysis

- Download the file AB_Climate.csv from the course website. This is a spatial, multivariate dataset of climate in Alberta. We'll be using this dataset throughout the course, so I'll explain what the variable names mean:

X, Y:	Coordinate of a sample location in UTM projection
ECOSYS:	The code of the ecosystem present at that location
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
PAS:	precipitation as snow (mm)

- Let's import this dataset, look at it, and try out the various options for checking the import:

```
ab=read.csv("AB_Climate.csv")
attach(ab)
head(ab)
str(ab)
data.frame(names(ab))
View(ab)
```

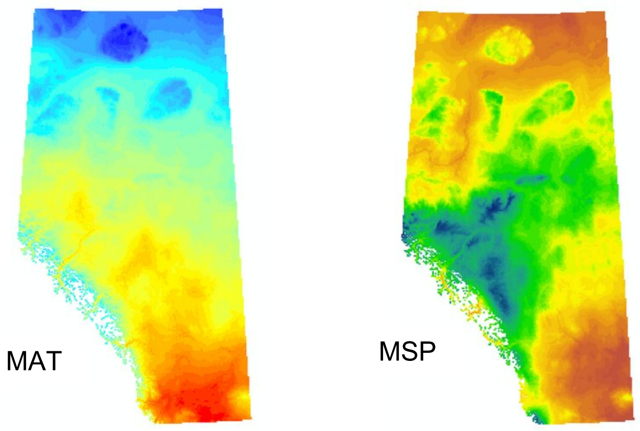
It's a fairly big table of spatial data, which is difficult to visualize. However, there are some functions in ad-on packages to R to display spatial data like that. Let's install the `lattice` extension package:

- If you work in R Studio install the package `lattice` from the Packages tab. If you work in base R, choose "Packages", then "Install packages …", choose a download mirror, then choose `lattice`. Load the package into memory with the `library(lattice)` command.
- Now you are ready to use `levelplot` to map the AB_Climate data. You can execute the new function levelplot with some custom color ramps that we make to color the map:

```
library(lattice)

tcol=colorRampPalette(c("blue","lightblue","yellow","red"))(100)

levelplot(MAT~X*Y, aspect="iso", cuts=99, col.regions=tcol)

pcol=colorRampPalette(c("brown","yellow","darkgreen",
            "darkblue"))(100)

levelplot(log(MAP)~X*Y, aspect="iso", cuts=99, col.regions=pcol)

levelplot(ECOSYS~X*Y, aspect="iso", cuts=20,
            col.regions=rainbow(21))
```

- To explain a few options in the commands above: the `aspect` option makes sure that your x and y intervals are the same (both measured in meters), the `cuts` option gives you the number of breaks in the color ramp, and in `col.regions` we specify one more color then we have breaks. You can use four built-in color palettes: `rainbow` , `heat.colors`, `topo.colors`, and `cm.colors`., or make custom color ramps with the `colorRampPalette()()` function. In the first bracket you place a vector of the base colors you like (at least two colors), in the second bracket you place the number of interpolated colors you want in the color ramp. Note that I also did a log-transformation of MAP to get better color separation (you can try without transformation to see the difference).

- Your output should look like this:



2.5. Principal Component Analysis in R with a custom bi-plot

If you were asked to describe the climate of Alberta based on a dataset like this, you would have to produce a lot of maps and somehow integrate that information from multiple variables into a simpler narrative. Principal component analysis can help with this “reduction of complexity” objective also in spatial datasets.

- If you have closed R, open it again via the “StartR.Rdata” shortcut and import the Climate_AB.csv file. Attach it and run the principal component analysis just as in the Exercise 2.3 above. There is only one important change – you have to specify the variables that you want to use for your principal component analysis because the dataset contains additional variables. We do this with the `[rows,cols]` subset command, where the climate variables are in columns 6 to 18:

```
data.frame(names(ab))
out1=princomp(ab[,6:18], cor=T)
```

- Can you interpret the component loadings and variance explained? The bi-plot certainly is a mess. If we have so many data points, we need a custom plot. We’ll cover graphics in another lab, but let’s do a simple scatter plot, where ecosystems are colored, and we add a legend as well:

```
plot(out1$score[,1:2],cex=0.3,asp=1,col=rainbow(21)[ab$ECOSYS])

legend(9~-12.5, cex=0.8, bty="n", pch=c(16),
       col=rainbow(21), legend=sort(unique(ab$ECOSYS)))
```

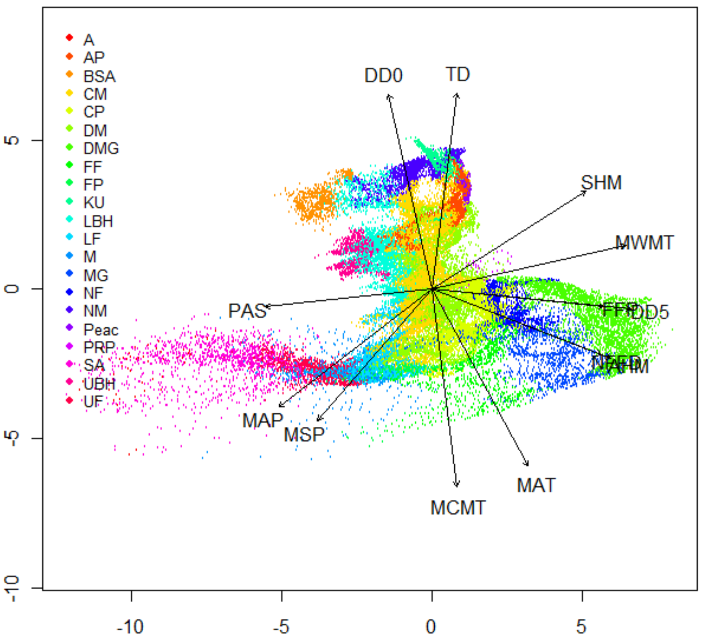
The `plot` command plots the first and second component scores, and the options `cex` makes the points smaller, `asp` forces x and y units to be the same, `col` specifies 21 colors for 21 ecosystems. We'll cover the legend command in another lab.

- All that's missing is the display of vectors, which we can do with a function from the vegan add-on package. You can also install via command line. I added a #, so that you don't accidentally run it twice. # blanks the line out, but you can run it by highlighting the command without the #.

```
# install.packages("vegan")
library(vegan)
vec1=envfit(out1$score[,1:2], ab[,6:18], permutations=0)
plot(vec1, col="black")
```

- The `envfit` function fits vectors indicating the correlation between the component scores that you plotted **out1**`$score[,1:2]` and the original variables **ab**`[,6:18]`. Note, that you can always remind yourself what data you are working with by quickly checking the elements that go into the function. For example with:

```
head(out1$score[,1:2])
head(ab[,6:18])
```



This is how you final bi-plot should look like

- If you have too many vectors, you can also customize which vectors you want to display. Re-run the `plot` and `legend` commands above, then select a few variables that you want to highlight for the vector plot. These may be the variables with the strongest loadings or important variables for your research questions:

```
data.frame(names(ab))
vectors=envfit(out1$score[,1:2], ab[,c(7,9,18)], permutations=0)
plot(vectors, col="black")
```

- Try to interpret the bi-plot you generated (selected ecosystems: UBH, LBH, NM: northern highlands, BSA, AP=boreal subarctic, DMG, MG=grasslands, CM, DM=boreal mixedwood, LF,UF=foothills, SA=subalpine).

- This is not bad, but it would be even better to visualize the principal components spatially. Let's plot them as a map and interpret them together with the component loadings:

```
out1$loadings[,1:3]
library(lattice)

pccol=colorRampPalette(c("green","yellow","red"))(100)
```

```
levelplot(out1$score[,1]~X*Y, aspect="iso", cuts=99,
          col.regions=pccol)
levelplot(out1$score[,2]~X*Y, aspect="iso", cuts=99,
          col.regions=pccol)
levelplot(out1$score[,3]~X*Y, aspect="iso", cuts=99,
          col.regions=pccol)
levelplot(log(out1$score[,3]+3)~X*Y, aspect="iso", cuts=99,
          col.regions=pccol)
```

- Note that in the last command I did a log transformation after adding a constant to the PCA score 3. This is simply to get better color resolution in the values most frequent in Alberta.

- Now you should be able to describe the main climatic gradients in Alberta in just a few words. Explain it to me or the TA and get some feedback whether you missed anything important.