Lab 3 Matrix Algebra and Ordination Part I

# Set up R session

## Download packages

Install and load packages

library(mvnormtest)  
library(MVN)  
library(MVA)  
library(psych)  
library(Hmisc)  
library(vegan)  
library(StatMatch)  
library(MASS)

# A primer of matrix algebra

Let’s start by making our own matrix:

newMatrix<- matrix(c(1,4,5,4,5,6,9,1,9),nrow=3, ncol=3)  
newMatrix

## [,1] [,2] [,3]  
## [1,] 1 4 9  
## [2,] 4 5 1  
## [3,] 5 6 9

The command c concatenates a list of numbers.

Now let’s check the dimensions of newMatrix:

dim(newMatrix)

## [1] 3 3

## 

## Matrix addition and subtraction

First you make a new matrix to either add to or subtract from “newMatrix”", Make a 3 x 3 matrix containing all ones and call it oneMatrix:

Now add oneMatrix to newMatrix:

newMatrix + oneMatrix

Then subtract oneMatrix from newMatrix:

newMatrix - oneMatrix

**Remember, because matrix addition and subtraction is performed on an element by element basis, matrices must have the same dimensions**

## 

## Scalar Multiplication

A **Scalar** is a single number. Scalar multiplication multiplies a scalar times a matrix:

3\*newMatrix

An *eigenvalue* is a scalar that is an essential component of multivariate analysis. We will explore this in a little bit.

## 

## Matrix Multiplication

You use % to signify that are using a matrix operation. Otherwise, R will attempt the operation element by element.

oneMatrix%\*%newMatrix

order matters:

newMatrix%\*%oneMatrix

**The number of columns in the first matrix must equal the number of rows in the second matrix.**

## 

## Matrix transposition

**Transposing** a matrix involves interchanging its rows and columns:

transMatrix<-t(newMatrix)

## 

## Identity Matrices

An **identity matrix** is a matrix where all the diagonal terms equal one and the remaining elements equal 0:

Identity<-diag(3)

## 

## Matrix Inversion

The inverse of matrix A is A-1.

invMatrix<-solve(newMatrix)

Multiplying a matrix times its inverse yields and identity matrix (A x A-1 = I):

invMatrix%\*%newMatrix

Let’s round it:

round(invMatrix%\*%newMatrix,10)

## 

## Eigenvalues and eigenvectors

Remember that an eigenvalue is a special scalar and the associated eigenvector is a vector that are key components of PCA.

eig<-eigen(newMatrix)

# Principal Component Analysis (PCA)

You are going to conduct a PCA on the USairpollution data.

First let’s call in the data:

usAir<-USairpollution

Now, look at the distributions (i.e., histograms) of the variables to determine if they need to be transformed. You should be able to make the histograms and transform them based on what you learned in the lab 2. Raise your hand and let me know which variables need to be transformed and which transformation you propose.

Let’s also test for multivariate and univariate normality (even though these tests are conservative)

mshapiro.test(t(usAir))

##   
## Shapiro-Wilk normality test  
##   
## data: Z  
## W = 0.59549, p-value = 2.025e-09

mvn(usAir, mvnTest = "mardia")

## $multivariateNormality  
## Test Statistic p value Result  
## 1 Mardia Skewness 226.612731693166 4.82491336101959e-15 NO  
## 2 Mardia Kurtosis 3.97754689564216 6.96298933924311e-05 NO  
## 3 MVN <NA> <NA> NO  
##   
## $univariateNormality  
## Test Variable Statistic p value Normality  
## 1 Shapiro-Wilk SO2 0.8117 <0.001 NO   
## 2 Shapiro-Wilk temp 0.9355 0.0221 NO   
## 3 Shapiro-Wilk manu 0.6055 <0.001 NO   
## 4 Shapiro-Wilk popul 0.6805 <0.001 NO   
## 5 Shapiro-Wilk wind 0.9806 0.6973 YES   
## 6 Shapiro-Wilk precip 0.9421 0.0373 NO   
## 7 Shapiro-Wilk predays 0.9654 0.2419 YES   
##   
## $Descriptives  
## n Mean Std.Dev Median Min Max 25th 75th Skew  
## SO2 41 30.048780 23.472272 26.00 8.00 110.0 13.00 35.00 1.584112608  
## temp 41 55.763415 7.227716 54.60 43.50 75.5 50.60 59.30 0.822975684  
## manu 41 463.097561 563.473948 347.00 35.00 3344.0 181.00 462.00 3.484603302  
## popul 41 608.609756 579.113023 515.00 71.00 3369.0 299.00 717.00 2.941257977  
## wind 41 9.443902 1.428644 9.30 6.00 12.7 8.70 10.60 0.002675131  
## precip 41 36.769024 11.771550 38.74 7.05 59.8 30.96 43.11 -0.692518149  
## predays 41 113.902439 26.506419 115.00 36.00 166.0 103.00 128.00 -0.550092270  
## Kurtosis  
## SO2 2.25541093  
## temp 0.09066032  
## manu 14.33200058  
## popul 10.57605759  
## wind 0.06015407  
## precip 0.49578021  
## predays 0.72033969

Compare test results to the histograms you plotted.

If you do transform any variables, use the transformed data matrix going forward.

Next, apply a z-score standardization:

ZusAir<-scale(usAir)

# Running the PCA:

You are going to use the package princomp function in the stats package. Take some time to read about this function:

?princomp

Run the princomp function:

usAir\_pca <- princomp(ZusAir, cor = F)

cor = F, because you are using the covariance matrix instead of the correlation matrix.

**It should be noted that the covariance matrix of a z-standardized data matrix is equivalent to the correlation matrix of the unscaled data.**

Let’s look at a summary of our PCA:

summary(usAir\_pca)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5  
## Standard deviation 1.6314350 1.2146805 1.1665973 0.9328641 0.58165337  
## Proportion of Variance 0.3897314 0.2160478 0.1992819 0.1274273 0.04953981  
## Cumulative Proportion 0.3897314 0.6057792 0.8050611 0.9324884 0.98202821  
## Comp.6 Comp.7  
## Standard deviation 0.3127963 0.157773920  
## Proportion of Variance 0.0143268 0.003644989  
## Cumulative Proportion 0.9963550 1.000000000

Notice that the summary gives the standard deviation instead of the eigenvalue (variance). Let’s calculate the eigenvalues using what we know about the relationship between standard deviation and variance (var = sd^2):

eigenVal<- (usAir\_pca$sdev\*sqrt(41/40))^2

The *sqrt(41/40)* is to correct for the fact that princomp calculates variances with the divisor N instead of N-1 as is customary. This adjustment will allow direct comparison with “hand” calculated eigenvalues using the function eigen below.

Let’s make the PCA table with the eigenvalues instead of the standard deviations:

propVar<-eigenVal/sum(eigenVal)  
cumVar<-cumsum(propVar)  
pca\_Table<-t(rbind(eigenVal,propVar,cumVar))  
pca\_Table

## eigenVal propVar cumVar  
## Comp.1 2.72811968 0.389731383 0.3897314  
## Comp.2 1.51233485 0.216047836 0.6057792  
## Comp.3 1.39497299 0.199281856 0.8050611  
## Comp.4 0.89199129 0.127427327 0.9324884  
## Comp.5 0.34677866 0.049539809 0.9820282  
## Comp.6 0.10028759 0.014326799 0.9963550  
## Comp.7 0.02551493 0.003644989 1.0000000

**This calculation and table are just to show you that the eigenvalues and the output from princomp, the stadard deviations, are the same thing. YOU DO NOT NEED TO DO THIS WHEN YOU RUN A PCA**

the factor loadings from princomp:

loadings(usAir\_pca)

##   
## Loadings:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7  
## SO2 0.490 0.404 0.730 0.183 0.150  
## temp -0.315 0.677 -0.185 0.162 0.611   
## manu 0.541 -0.226 0.267 -0.164 -0.745  
## popul 0.488 -0.282 0.345 -0.113 -0.349 0.649  
## wind 0.250 -0.311 -0.862 0.268 0.150   
## precip 0.626 0.492 -0.184 0.161 -0.554   
## predays 0.260 0.678 -0.110 0.110 -0.440 0.505   
##   
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7  
## SS loadings 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## Proportion Var 0.143 0.143 0.143 0.143 0.143 0.143 0.143  
## Cumulative Var 0.143 0.286 0.429 0.571 0.714 0.857 1.000

and the factor scores:

scores(usAir\_pca)

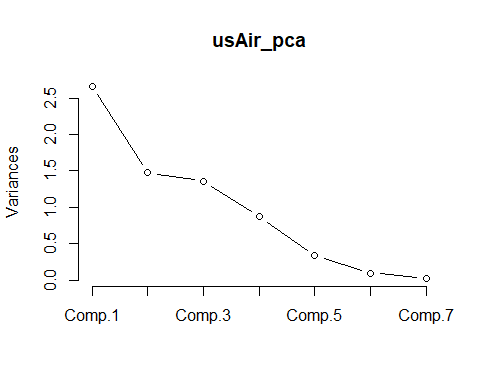
**Let’s refer back to the lecture slides (front of the class) to discuss.**

## 

## How many PC Axes to keep?

You now have 7 PC axes. Which ones give us vital information and which ones can we toss? One method for selecting the number of Axes is a **Scree plot**:

plot(usAir\_pca, type="lines")



How about the **latent root criterion (i.e., keep components with eigenvalues > 1)** and the **relative percent variance criteria**. Check the summary of the PCA explore this:

summary(usAir\_pca)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5  
## Standard deviation 1.6314350 1.2146805 1.1665973 0.9328641 0.58165337  
## Proportion of Variance 0.3897314 0.2160478 0.1992819 0.1274273 0.04953981  
## Cumulative Proportion 0.3897314 0.6057792 0.8050611 0.9324884 0.98202821  
## Comp.6 Comp.7  
## Standard deviation 0.3127963 0.157773920  
## Proportion of Variance 0.0143268 0.003644989  
## Cumulative Proportion 0.9963550 1.000000000

**Raise your hand and tell me how many axes you should keep.**

## 

## Significance of factor loadings.

While many use the “rule of thumb” where a loading > 0.30 dictates an “important” variable. Another method for determining significance of factor loadings is bootstrapping. Details and comparisons of the many way to assess significance of factor loadings are presented in Peres-Neto et al. (2003), which is supplemental reading for this week. Here we will run the method that they found to have the lowest type I error rates, Bootstrapped eigenvector. For reference, this is the method 6 in Peres-Neto et al. (2003).

sigpca2<-function (x, permutations=1000, ...)  
{  
 pcnull <- princomp(x, ...)  
 res <- pcnull$loadings  
 out <- matrix(0, nrow=nrow(res), ncol=ncol(res))  
 N <- nrow(x)  
 for (i in 1:permutations) {  
 pc <- princomp(x[sample(N, replace=TRUE), ], ...)  
 pred <- predict(pc, newdata = x)  
 r <- cor(pcnull$scores, pred)  
 k <- apply(abs(r), 2, which.max)  
 reve <- sign(diag(r[k,]))  
 sol <- pc$loadings[ ,k]  
 sol <- sweep(sol, 2, reve, "\*")  
 out <- out + ifelse(res > 0, sol <= 0, sol >= 0)  
 }  
 out/permutations  
}  
  
set.seed(4)  
   
sigpca2(ZusAir, permutations=1000)

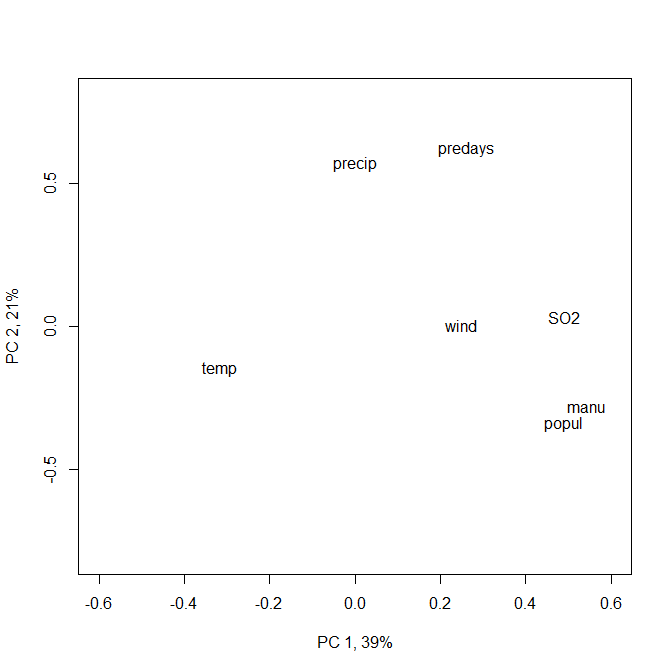
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7  
## SO2 0.034 0.355 0.301 0.174 0.081 0.069 0.021  
## temp 0.031 0.405 0.260 0.261 0.154 0.016 0.440  
## manu 0.037 0.334 0.237 0.488 0.094 0.343 0.011  
## popul 0.082 0.326 0.259 0.364 0.076 0.186 0.014  
## wind 0.039 0.427 0.384 0.250 0.103 0.044 0.333  
## precip 0.505 0.246 0.294 0.275 0.137 0.015 0.415  
## predays 0.043 0.220 0.489 0.374 0.068 0.031 0.461

#Piece-by-piece (read along step-by-step with pg. 2350 section 6) Bootstrapped eigenvector (V vectors)” of Peres-Neto et al. 2003.  
pcnull<-princomp(ZusAir)  
res <- pcnull$loadings  
out <- matrix(0, nrow=nrow(res), ncol=ncol(res))  
N <- nrow(ZusAir)  
pc<-princomp(ZusAir[sample(N, replace=TRUE), ])  
pred <- predict(pc, newdata = ZusAir)   
r <- cor(pcnull$scores, pred)  
k <- apply(abs(r), 2, which.max)  
reve <- sign(diag(r[k,]))  
sol <- pc$loadings[ ,k]  
sol <- sweep(sol, 2, reve, "\*")  
out <- out + ifelse(res > 0, sol <= 0, sol >= 0)

## PCA plots

Plot out the factor loadings for the first 2 PC axes:

plot(usAir\_pca$loadings,type="n",xlab="PC 1, 39%", ylab="PC 2, 21%",ylim=c(-.8,.8), xlim=c(-.6,.6))  
  
text(usAir\_pca$loadings, labels=as.character(colnames(ZusAir)), pos=1, cex=1)

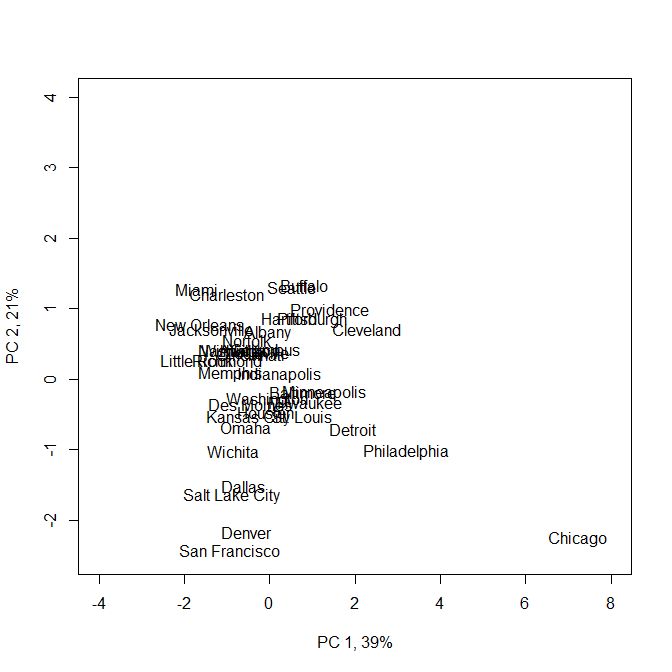


How do you interpret these axes? Come up with a name for each.

**Close the plot window after viewing**

Let’s now plot the PC score for each sample (city):

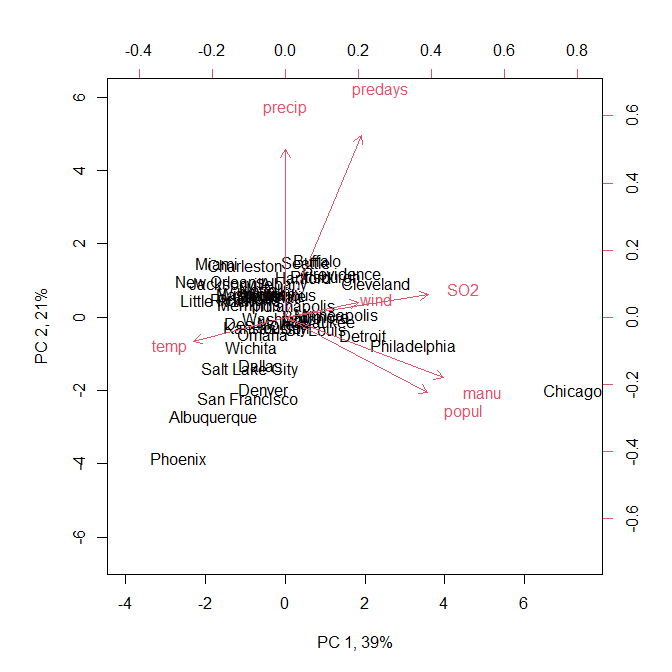
plot(usAir\_pca$scores,type="n",xlab="PC 1, 39%", ylab="PC 2, 21%",ylim=c(-2.5,4), xlim=c(-4,8))  
text(usAir\_pca$scores, labels=as.character(rownames(ZusAir)), pos=1, cex=1)



And now all together in a biplot:

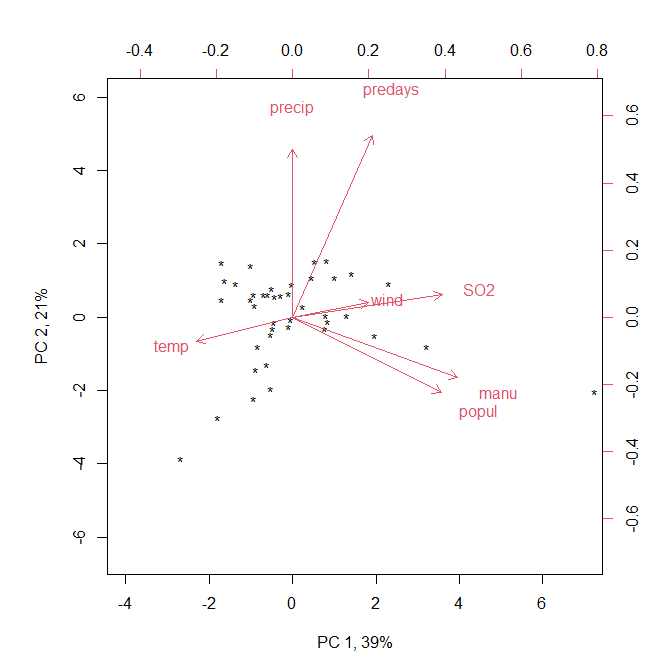
?biplot

biplot(usAir\_pca$scores,usAir\_pca$loading,xlab="PC 1, 39%", ylab="PC 2, 21%",expand= 4, ylim=c(-6.5,6), xlim=c(-4,7.5))



to replace city names with a symbol:

biplot(usAir\_pca$scores,usAir\_pca$loading, expand= 4, xlabs= rep("\*",41),xlab="PC 1, 39%", ylab="PC 2, 21%",ylim=c(-6.5,6), xlim=c(-4,7))



# Eigen Analysis

You can also just simply use the Eigen analysis function, eigen and calculate your own scores by hand:

eig<-eigen(cov(ZusAir))

Extract the first two eigenvectors (because that is what we are interested in plotting):

eigVec<-as.matrix(eig$vectors[,1:2])  
rownames(eigVec) <- rownames(cov(ZusAir))

Then simply multiply each eigenvector times the matrix of standardized observation values (ZusAir) and plot!

scores<-t(rbind(eigVec[,1]%\*%t(ZusAir),eigVec[,2]%\*%t(ZusAir)))#hand calculated scores  
  
# and plot  
  
biplot(scores,eigVec,xlab="PC 1, 39%", ylab="PC 2, 21%",expand= 4, ylim=c(-6.5,6), xlim=c(-4,7.5))

