**Multivariate Statistics for the Life Sciences**

**WIS 6934**

*Lab 3: Matrix Algebra and Ordination Part I*

The goal of this lab is to become familiar with the basics of matrix algebra and learn the ordination techniques of *principal components analysis* (PCA).

----

1. **Set up R session**

**a) Set your working directory**

path<-“C:/Users/bbaiser/Documents/Multivariate/lab”

setwd(path)

**b) Download packages**

install.packages(“MVA”)

install.packages(“psych”)

install.packages(“Hmisc”)

install.packages(“vegan”)

install.packages("StatMatch")

install.packages("MASS")

library(MVA)

library(psych)

library(Hmisc)

library(vegan)

library(StatMatch)

library(MASS)

1. **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

the command *c* concatenates a list of numbers and *cbind* binds the columns together into matrix form.

Now let’s check the dimensions of *newMatrix*:

dim(newMatrix)

**a) 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

and 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

**b) 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.

**c) 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.

**d) Matrix transposition**

Transposing a matrix involves interchanging its rows and columns:

transMatrix<-t(newMatrix)

transMatrix

**e) Identity Matrices**

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

Identity<-diag(3)

Identity

**f) Matrix Inversion**

The inverse of matrix **A** is **A-1.**

invMatrix<-solve(newMatrix)

invMatrix

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)

**g) 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)

eig

1. **Principal Component Analysis (PCA)**

You are going to conduct a PCA on the US air pollution data.

First let’s call in the data:

usAir<-USairpollution

usAir

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.**

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

Next, apply a z-score standardization:

ZusAir<-scale(usAir)

ZusAir

**a) 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)

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

the factor loadings:

loadings(usAir\_pca)

and the factor scores:

scores(usAir\_pca)

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

**b) 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 *pca\_Table* you made to explore this:

pca\_Table

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

**c) 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

}

sigpca2(ZusAir, permutations=1000)

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)

**d) 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(-4,4), 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(usAir\_pca$scores,usAir\_pca$loading,xlab="PC 1, 39%", ylab="PC 2, 21%",ylim=c(-2,6), xlim=c(-4,7))

to replace city names with a symbol:

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

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

eig<-eigen(cov(ZusAir))

eig

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))

eigVec

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

biplot(scores,eigVec,xlab="PC 1, 39%", ylab="PC 2, 21%",ylim=c(-2,6), xlim=c(-4,7))