# Software: 2blockPLS_and_ordinations.r



**Erik Otárola-Castillo November 2010**

```
# Below are general programs I wrote to conduct 2Block Partial Least Squares (from Rohlf and Corti 2000 systematic biology),
# Principal Components Analysis, Principal Coordinates Analysis, Correspondence Analysis, Canonical Correspondence Analysis, and Redundancy Analysis.
# 2B-PLS (from Rohlf and Corti 2000 sytematic biology)

# chiken measurements correlation data
# 2 sets (blocks) of data: skull length and skull breadth measured against limb measurements
# is there a correlation?

chickencor<-matrix(c(1,.584,.615,.610,.570,.600,.584,1,.576,.530,.526,.555,.615,.576,1,
                .940,.875,.878,.610,.530,.940,1,.877,.886,.570,.526,.875,.877,1,.924,.600,.555,.878,
                .886,.924,1),nrow=6,ncol=6,dimnames=list(c("Skull length","Skull
breadth","Fibula","Tibia","Humerus","Ulna"),
                                   c("Skull length","Skull
breadth","Fibula","Tibia","Humerus","Ulna")))

# partition of the correlation matrix: correlation between dataset 1 and dataset 2
r21<-
matrix(c(chickencor[3:6],chickencor[9:12]),nrow=4,ncol=2,dimnames=list(c("Fibula","Tibia"
,"Humerus","Ulna"),c("Skull length","Skull breadth")))

# this is r12 r12<-t(r21)
r12<-t(r21)
# decompose data using SVD. Unlike eigen analysis, SVD can decompose asymetric matrices,
# in this case 4-x-2 (4 limb bones x 2
S<-svd(r12)
lam<-sum(S$d^2)
lam1<-S$d[1]^2
cumvar<-S$d^2/lam
corr<-lam/(14*6)


x1<-rnorm(100)
x2<-rnorm(100)
int<-x1*x2
y1<-.9*x1+rnorm(100)
y2<-.9*x2+rnorm(100)
y3<-.9*x2*x1+rnorm(100)
```

```
mat<-as.matrix(cbind(x1,x2,int,y1,y2,y3))
matc<-scale(mat,center=TRUE,scale=FALSE)
vcv<-(1/(100-1))*t(matc)%*%matc
r12<-
matrix((vcv[4:6,1:3]),nrow=3,ncol=3,dimnames=list(c("y1","y2","y3"),c("x1","x2","int")))
r12<-matrix((vcv[3:5,2:1]),nrow=3,ncol=2,dimnames=list(c("y1","y2","y3"),c("x1","x2")))

S<-svd(r12)
lam<-sum(S$d^2)
lam1<-S$d[1]^2
cumvar<-S$d^2/lam
corr<-lam/(2*2)

########################################
# PCA
# Begin with a dataset of n rows x p columns named Y
Y<-cbind(c(2,3,5,7,9),c(1,4,0,6,2))
Y<-matrix(rnorm(100),ncol=10)
n<-dim(Y)[1]
p<-dim(Y)[2]

# mean-center columns (subtract the column means from respective colum; either create a
function or use "scale(Y,center=TRUE,scale=FALSE)")
  # call the new mean centered matrix Yc
cent<-function(x)(x-mean(x))
Yc<-apply(Y,2,cent)

# obtain S, the VCV matrix of the mean-centered data matrix Yc
  #(1/(n-1))*sum((Y[,1]-mean(Y[,2]))*(Y[,2]-mean(Y[,2])))
  # scalar-multiply 1/n-1 times the transpose Yc matrix-multiplied-by Yc (or use "cov(Yc)")
S<-(1/(n-1))*t(Yc)%*%Yc
# find the eigenvalues and eigenvectors of the VCV matrix S
  # subtract Yc-lambda%*%I (lambda matrix has lambda in diagonals and 0s elsewhere)
  # find characteristic polynomial equation of Yc-lambda%*%I matrix (Yca-lambda)*(Ycd-
lambda)-(Ycb*Ycc)
  # set the characteristic polynomial to 0 and solve - result is the eigenvalues. Eigen values
have the strenght of the component.
  # subtract matrix S from the first eigenvalue and set an equation where the result (the null
eigenspace) is multiplied some unknown vector named x
  # set right hand side to zero and solve for vector x, the eigenvectors of the first eigenvalue.
 # easier to conduct eigen analysis using "eigen" on the square matrix S (VCV matrix)
 # alternatively one can conduct singular value decomposition and use the d and u matrices
("svd(S)")
E<-eigen(S)$values
U<-eigen(S)$vectors

# obtain variable scores by "projecting" (matrix multiplying) Yc, the column-mean-centered
matrix by the eigenvectors.
F<-Yc%*%U
summary(prcomp(Y))

########################################
```

```
# PCoA
D<-as.matrix(dist(Y,diag = TRUE, upper = TRUE))
ahi<--1/2*D^2
ah<-rowMeans(ahi)
ai<-colMeans(ahi)
a<-mean(ahi)
cent<-function(x,y)(x-y)
col<-apply(ahi,2,cent,y=ai)
row<-apply(col,1,cent,y=ah)
deltahi<-row+a

c<-scale(ahi,center=TRUE,scale=FALSE)
r<-scale(t(c),center=TRUE,scale=FALSE)


evals<-eigen(deltahi)$values
evects<-eigen(deltahi)$vectors
steval<-as.matrix(t(sqrt(evals[1:p])))
mul<-function(x,y)(x*y)
stevect<-t(apply(evects[,1:p],1,mul,steval))
plot(stevect[,1],stevect[,2])
#contribution
sqrt(evals/(n-1))
(evals/(n-1))/sum((evals/(n-1)))
cumsum((evals/(n-1))/sum((evals/(n-1))))
# general "loop" for distance matrix calculation
di<-matrix(NA,nrow=n,ncol=n)
for (i in 1:n)   {
  for (j in 1:n)          {
# Euclidean distance formula
di[i,j]<-sqrt((Y[i,1]-Y[j,1])^2+(Y[i,2]-Y[j,2])^2)
                                    }
                                                }


########################################
# Correspondence Analysis
# spec<-lag[,1:9]
spec<-matrix(c(10,10,15,10,15,5,20,10,5),nrow=3,ncol=3)
spec<-strat.mat
pij<-spec/sum(spec)
pisum<-rowSums(spec)/sum(spec)
pjsum<-colSums(spec)/sum(spec)

# Qbar matrix
Qbar<-matrix(NA,nrow=dim(spec)[1],ncol=dim(spec)[2])
for (i in 1:dim(Qbar)[1])     {
  for (j in 1:dim(Qbar)[2])            {
Qbar[i,j]<-(pij[i,j]-(pisum[i]*pjsum[j]))/sqrt(pisum[i]*pjsum[j])
                                                        }
                                                                    }
# Qtilde matrix
```

```
Qtilde<-matrix(NA,nrow=dim(spec)[1],ncol=dim(spec)[2])
for (i in 1:dim(Qbar)[1])     {
  for (j in 1:dim(Qbar)[2])            {
Qtilde[i,j]<-(pij[i,j])/sqrt(pisum[i]*pjsum[j])
                                        }
                                                }

chidist<-t(Qbar)%*%Qbar # column scores
chidist<-Qbar%*%t(Qbar) # row scores
# Eigenvalues and eigenvectors of Qbar transpose times Qbar
lamb<-eigen(t(Qbar)%*%Qbar)$values
U<-eigen(t(Qbar)%*%Qbar)$vectors

# proportional contribution of lambda the eigenvalues
picont<-lamb/sum(lamb)

# remove any eigenvectors with 0 contribution
U<-eigen(t(Qbar)%*%Qbar)$vectors[,1:2]


# Eigenvalues and eigenvectors of Qbar times transpose Qbar
lamb2<-eigen(Qbar%*%t(Qbar))$values
Uhat<-eigen(Qbar%*%t(Qbar))$vectors

# proportional contribution of lambda the eigenvalues
picont2<-lamb2/sum(lamb2)

# remove any eigenvectors with 0 contribution
Uhat<-eigen(Qbar%*%t(Qbar))$vectors[,1:2]


########################################
# RDA
# open data
lag<-read.csv(file.choose())

# create data matrix Y with responses
Y<-as.matrix(lag[,1:5])

# create data matrix X with predictors
X<-as.matrix(cbind(rep(1,20),lag[,6:8]))

# calculate B, the multiple regression coefficients for each Y as a function of all X
B<-solve(t(X)%*%X)%*%t(X)%*%Y

# compute Yhat, the fitted values of Y
Yhat<-X%*%B
Yhat<-scale(Yhat,center=TRUE,scale=FALSE) # center Yhat
# calculate Syhyhthe VCV of Yhat (the fitted values of the response; vals along the line)
Syhyh<-(1/(20-1))*(t(Yhat)%*%Yhat)

# Conduct eigenanalysis on Syhyh
```

```
evals<-eigen(Syhyh)$values
U<-eigen(Syhyh)$vectors
steval<-as.matrix(t(sqrt(evals[1:3])))
mul<-function(x,y)(x*y)
stevect<-t(apply(U[,1:3],1,mul,steval))

# project Y through the eigenvectors to obtain "site scores"
F<-Y%*%U

# project Yhat through the eigenvectors to obtain "fitted site scores"
Z<-Yhat%*%U

C<-B%*%U
X%*%C

## covariance loop function - needs a little bit of work
covar<-matrix(NA,nrow=5,ncol=5)
for (i in 1:5)   {
  for (j in 1:5)          {
    Emi<-mean(Yhat[,i]);Emj<-mean(Yhat[,j])
    covar[i,j]<-mean((Yhat[,i]-Emi)*(Yhat[,j]-Emj))
                          }
                                  }

########################################
# CCA
# open data
lag<-read.csv(file.choose())

# create data matrix Y with responses
Y<-as.matrix(lag[,1:5])

# create data matrix X with predictors
X<-as.matrix(cbind(rep(1,20),lag[,6:8]))


spec<-lag[,1:9]
spec<-matrix(c(10,10,15,10,15,5,20,10,5),nrow=3,ncol=3)

pij<-spec/sum(spec)
pisum<-rowSums(spec)/sum(spec)
pjsum<-colSums(spec)/sum(spec)

Qbar<-matrix(NA,nrow=3,ncol=3)
for (i in 1:dim(Qbar)[1])     {
  for (j in 1:dim(Qbar)[2])              {
    Qbar[i,j]<-(pij[i,j]-(pisum[i]*pjsum[j]))/sqrt(pisum[i]*pjsum[j])
  }
}
```