```
`PCA` <-
   function(Y, stand=FALSE)
#
# Principal component analysis (PCA) with option for variable standardization
#
# stand = FALSE : center by columns only, do not divide by s.d.
# stand = TRUE  : center and standardize (divide by s.d.) by columns
#
# License: GPL-2
# Author: Pierre Legendre, May 2006
{
   Y = as.matrix(Y)
   obj.names = rownames(Y)
   var.names = colnames(Y)
   size = dim(Y)
   Y.cent = apply(Y, 2, scale, center=TRUE, scale=stand)
   Y.cov = cov(Y.cent)
   Y.eig = eigen(Y.cov)
   k = length(which(Y.eig$values > 1e-10))
   U  = Y.eig$vectors[,1:k]
   F  = Y.cent %*% U
   U2 = U %*% diag(Y.eig$value[1:k]^(0.5))
   G  = F %*% diag(Y.eig$value[1:k]^(-0.5))
   rownames(F)  = obj.names
   rownames(U)  = var.names
   rownames(G)  = obj.names
   rownames(U2) = var.names
   axenames <- paste("Axis",1:k,sep=" ")
   colnames(F)  = axenames
   colnames(U)  = axenames
   colnames(G)  = axenames
   colnames(U2) = axenames

#
# Fractions of variance
   varY = sum(diag(Y.cov))
   eigval = Y.eig$values[1:k]
   relative = eigval/varY
   rel.cum = vector(length=k)
   rel.cum[1] = relative[1]
   for(kk in 2:k) { rel.cum[kk] = rel.cum[kk-1] + relative[kk] }
#
out <- list(total.var=varY, eigenvalues=eigval, rel.eigen=relative,
       rel.cum.eigen=rel.cum, U=U, F=F, U2=U2, G=G, stand=stand,
       obj.names=obj.names, var.names=var.names, call=match.call() )
class(out) <- "PCA"
out
}

`print.PCA` <-
    function(x, ...)
{
    cat("\nPrincipal Component Analysis\n")
    cat("\nCall:\n")
    cat(deparse(x$call),'\n')
    if(x$stand) cat("\nThe data have been centred and standardized by column",'\n')
```

```
    cat("\nTotal variance in matrix Y: ",x$total.var,'\n')
    cat("\nEigenvalues",'\n')
    cat(x$eigenvalues,'\n')
    cat("\nRelative eigenvalues",'\n')
    cat(x$rel.eigen,'\n')
    cat("\nCumulative relative eigenvalues",'\n')
    cat(x$rel.cum.eigen,'\n')
    invisible(x)
}

`biplot.PCA` <-
    function(x, scaling=1, plot.axes=c(1,2), color.obj="black", color.var="red",
...)
# scaling = 1 : preserves Euclidean distances among the objects
# scaling = 2 : preserves correlations among the variables
{
    #### Internal function
    larger.frame <- function(mat, percent=0.07)
    # Produce an object plot 10% larger than strictly necessary
    {
    range.mat = apply(mat,2,range)
    z <- apply(range.mat, 2, function(x) x[2]-x[1])
    range.mat[1,]=range.mat[1,]-z*percent
    range.mat[2,]=range.mat[2,]+z*percent
    range.mat
    }
    ####

    if(length(x$eigenvalues) < 2) stop("There is a single eigenvalue. No plot can
be produced.")
    if(length(which(scaling == c(1,2))) == 0) stop("Scaling must be 1 or 2")

    par(mai = c(1.0, 0.75, 1.0, 0.5))

    if(scaling == 1) {

    # Distance biplot, scaling type = 1: plot F for objects, U for variables
    # This projection preserves the Euclidean distances among the objects

    lf.F = larger.frame(x$F[,plot.axes])
    biplot(x$F[,plot.axes],x$U[,plot.axes],col=c(color.obj,color.var),
xlim=lf.F[,1], ylim=lf.F[,2], arrow.len=0.05, asp=1)
    title(main = c("PCA biplot","scaling type 1"), family="serif", line=3)

    } else {

    # Correlation biplot, scaling type = 2: plot G for objects, U2 for variables
    # This projection preserves the correlation among the variables

    lf.G = larger.frame(x$G[,plot.axes])
    biplot(x$G[,plot.axes],x$U2[,plot.axes],col=c(color.obj,color.var),
xlim=lf.G[,1], ylim=lf.G[,2], arrow.len=0.05, asp=1)
    title(main = c("PCA biplot","scaling type 2"), family="serif", line=3)

    }
invisible()
```

```
}

# mite.hel = decostand(mite, "hel")
# mite.hel.D = dist(mite.hel)
# mite.correlog = mantel.correlog(mite.hel.D, XY=mite.xy, nperm=99, cutoff=FALSE)
# mite.correlog
# mite.correlog$mantel.res
# plot(mite.correlog)
```

From:
https://anadat-r.davidzeleny.net/ - **Analysis of community ecology data in R**

Permanent link:
https://anadat-r.davidzeleny.net/doku.php/cs:numecolr:pca

Last update: **2017/10/11 20:36**