Distances

Miquel Salicru, Sergi Civit, Ferran Reverter

Distances in R

Distance functions are available in at least three packages of the R language.

Package stats

Function dist : 6 distances (documentation: ?dist) The choice of coefficient is done by typing its name in quotes. Example: dist(data, method=“binary”) or dist(data, “binary”)

euclidean: Usual distance between the two vectors (2 norm aka L_2), sqrt(sum((x_i - y_i)^2)).

maximum: Maximum distance between two components of x and y (supremum norm)

manhattan: Absolute distance between the two vectors (1 norm aka L_1).

canberra: sum(|x_i - y_i| / |x_i + y_i|). Terms with zero numerator and denominator are omitted from the sum and treated as if the values were missing.

binary: (aka asymmetric binary): The vectors are regarded as binary bits, so non-zero elements are ‘on’ and zero elements are ‘off’. The distance is the proportion of bits in which only one is on amongst those in which at least one is on.

minkowski: The p norm, the pth root of the sum of the pth powers of the differences of the components.

```
x <- matrix(rnorm(100), nrow = 5)
dim(x)
```

```
## [1]  5 20
```

```
dist(x)
```

```
##          1        2        3        4
## 2 5.086051
## 3 7.294924 6.686224
## 4 5.702883 6.317966 6.755952
## 5 5.674455 6.605299 7.850960 7.538817
```

```
class(dist(x))
```

```
## [1] "dist"
```

```
dist(x, diag = TRUE)
```

```
##          1        2        3        4        5
## 1 0.000000
## 2 5.086051 0.000000
## 3 7.294924 6.686224 0.000000
## 4 5.702883 6.317966 6.755952 0.000000
## 5 5.674455 6.605299 7.850960 7.538817 0.000000
```

```
dist(x, upper = TRUE)
```

```
##          1        2        3        4        5
## 1          5.086051 7.294924 5.702883 5.674455
## 2 5.086051          6.686224 6.317966 6.605299
## 3 7.294924 6.686224          6.755952 7.850960
## 4 5.702883 6.317966 6.755952          7.538817
## 5 5.674455 6.605299 7.850960 7.538817
```

```
m <- as.matrix(dist(x))
m
```

```
##          1        2        3        4        5
## 1 0.000000 5.086051 7.294924 5.702883 5.674455
## 2 5.086051 0.000000 6.686224 6.317966 6.605299
## 3 7.294924 6.686224 0.000000 6.755952 7.850960
## 4 5.702883 6.317966 6.755952 0.000000 7.538817
## 5 5.674455 6.605299 7.850960 7.538817 0.000000
```

Package vegan

Function vegdist : 10 distances (documentation: ?vegdist)

The choice of coefficient is done by typing its name in quotes. Example: vegdist(data, method="bray") or vegdist(data, "bray")

euclidean d[jk] = sqrt(sum(x[ij]-x[ik])^2)

manhattan d[jk] = sum(abs(x[ij] - x[ik]))

gower d[jk] = (1/M) sum(abs(x[ij]-x[ik])/(max(x[i])-min(x[i])))

altGower d[jk] = (1/NZ) sum(abs(x[ij] - x[ik])) where NZ is the number of non-zero columns excluding double-zeros (Anderson et al. 2006).

canberra d[jk] = (1/NZ) sum (abs(x[ij]-x[ik])/(abs(x[ij])+abs(x[ik]))) where NZ is the number of non-zero entries.

bray d[jk] = (sum abs(x[ij]-x[ik]))/(sum (x[ij]+x[ik]))

. . .

Package ade4

Function dist.binary : 10 binary distances (documentation: ?dist.binary)

These similarities (S) are converted to distances through the transformation D = sqrt(1 − S) The choice of coefficient is done by typing its number in the list above. Example: dist.binary(data, method=1) or dist.binary(data, 1) or dist.binary(data, "1")

Let be the contingency table of binary data such as $n11 = a$, $n10 = b$, $n01 = c$ and $n00 = d$.

1 = Jaccard index (1901) S3 coefficient of Gower & Legendre s1 = a / (a+b+c)

2 = Simple matching coefficient of Sokal & Michener (1958) S4 coefficient of Gower & Legendre s2 = (a+d) / (a+b+c+d)

3 = Sokal & Sneath(1963) S5 coefficient of Gower & Legendre s3 = a / (a + 2(b + c))

4 = Rogers & Tanimoto (1960) S6 coefficient of Gower & Legendre s4 = (a + d) / (a + 2(b + c) +d)

5 = Dice (1945) or Sorensen (1948) S7 coefficient of Gower & Legendre s5 = 2a / (2a + b + c)

6 = Hamann coefficient S9 index of Gower & Legendre (1986) s6 = (a - (b + c) + d) / (a + b + c + d)

7 = Ochiai (1957) S12 coefficient of Gower & Legendre s7 = a / sqrt((a + b)(a + c))

8 = Sokal & Sneath (1963) S13 coefficient of Gower & Legendre s8 = ad / sqrt((a + b)(a + c)(d + b)(d + c))

9 = Phi of Pearson S14 coefficient of Gower & Legendre s9 = (ad - bc) / sqrt((a + b)(a + c)(d + b)(d + c))

10 = S2 coefficient of Gower & Legendre s10 = a / (a + b + c + d)

Practice

```
wd<-"C:/Users/User/Documents/docencia/curs17_18/ub/Multivariant/bloc1"
setwd(wd)
data<-read.table("FANGA TAUFA delimitado tabulaciones.txt",sep="\t",header=TRUE)   #gastropod species
str(data)
```

```
## 'data.frame':    28 obs. of  14 variables:
##  $ GRUP   : int  1 1 1 1 1 1 1 3 3 3 ...
##  $ Pat.fle: int  95 12 7 9 4 0 0 0 0 0 ...
##  $ Tur.set: int  18 5 14 13 5 2 3 0 0 0 ...
##  $ Ner.pli: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Tec.gra: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Dru.ric: int  24 13 10 13 25 2 4 2 2 1 ...
##  $ Mor.uva: int  0 0 0 0 0 0 0 14 10 3 ...
##  $ Mor.gra: int  0 0 0 0 0 0 0 7 0 0 ...
##  $ Mit.lit: int  0 0 0 0 0 0 0 1 1 0 ...
##  $ Con.ebr: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Con.mil: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Con.spo: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Con.nan: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Con.ver: int  0 0 0 0 0 0 0 3 0 1 ...
```

```
head(data)
```

```
##   GRUP Pat.fle Tur.set Ner.pli Tec.gra Dru.ric Mor.uva Mor.gra Mit.lit
## 1    1      95      18       0       0      24       0       0       0
## 2    1      12       5       0       0      13       0       0       0
## 3    1       7      14       0       0      10       0       0       0
## 4    1       9      13       0       0      13       0       0       0
## 5    1       4       5       0       0      25       0       0       0
## 6    1       0       2       0       0       2       0       0       0
##   Con.ebr Con.mil Con.spo Con.nan Con.ver
## 1       0       0       0       0       0
## 2       0       0       0       0       0
## 3       0       0       0       0       0
## 4       0       0       0       0       0
## 5       0       0       0       0       0
## 6       0       0       0       0       0
```

MDS

The function cmdscale of the stats package to carry out this analysis. “cmds” is the acronym of classical multidimensional scaling.

Compute the matrix of Bray Curtis distances

```
library(vegan)
```

```
## Warning: package 'vegan' was built under R version 3.4.4
```

```
## Loading required package: permute
```

```
## Warning: package 'permute' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

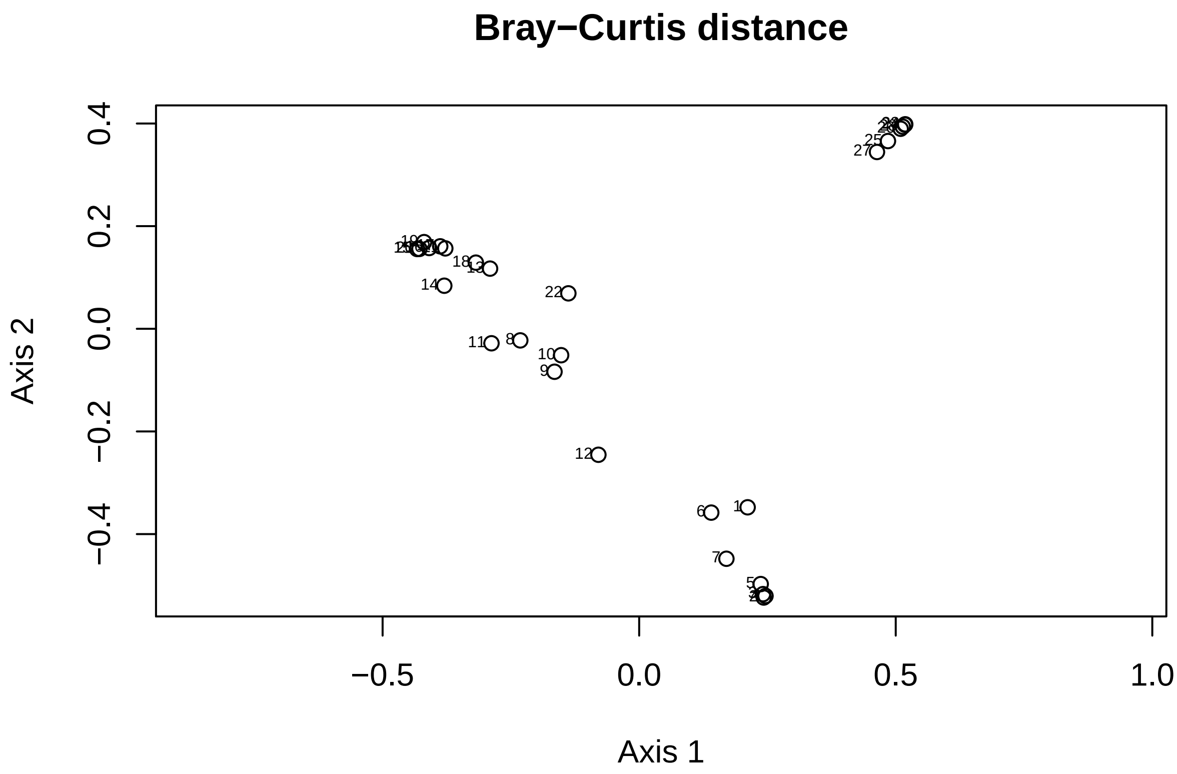
```
## This is vegan 2.4-6
```

```
data.D2<-vegdist(data[,-1], method="bray")
class(data.D2)
```

```
## [1] "dist"
```

Principal coordinate analysis. Save k=3 axes. Plot a graph of axes 1 and 2.

```
outBC=cmdscale(data.D2, 3, eig=TRUE)
plot(outBC$points[,1], outBC$points[,2], main="Bray-Curtis distance",asp=1, xlab="Axis 1", ylab="Axis 2
names = rownames(data)
text(outBC$points[,1], outBC$points[,2], labels= names, pos=2, cex=0.5, offset=0.15)
```

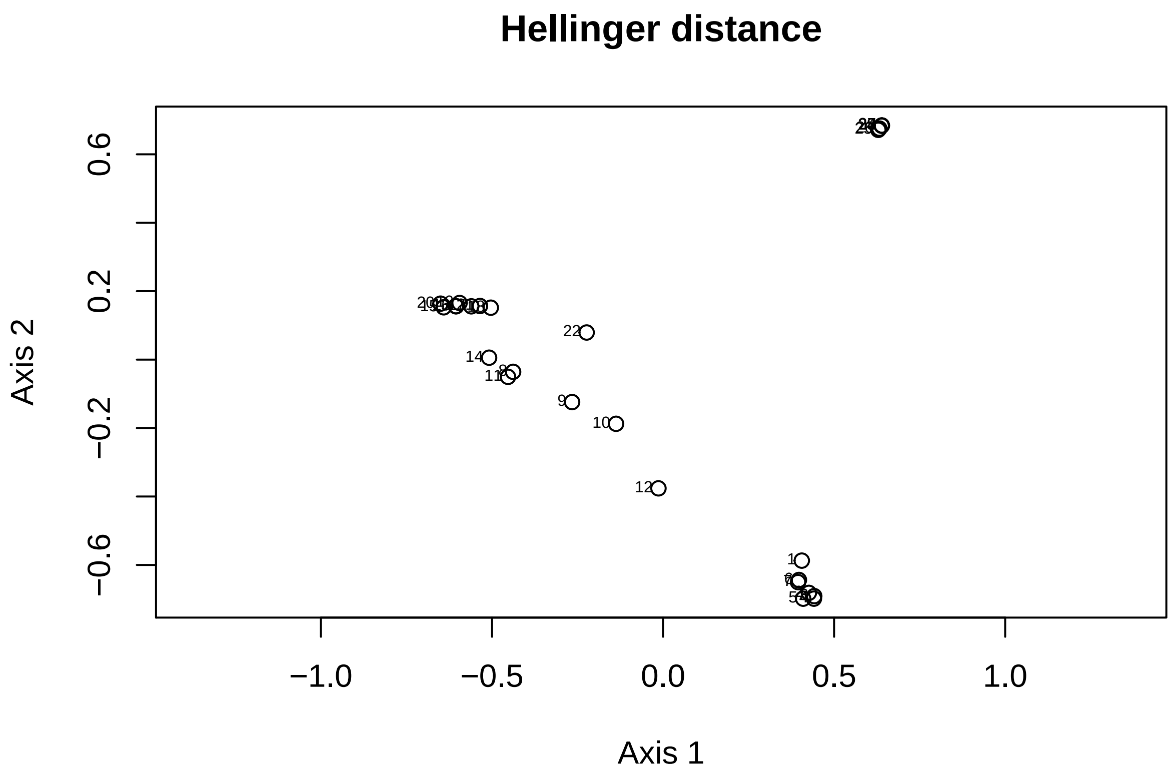


Repeat the analysis after applying the Hellinger transformation (using function decostand of the vegan package) to the data. Hellinger transformation, followed by calculation of Euclidean distances.

```
data.hel = decostand(data[,-1], "hel")
data.DHell = dist(data.hel)
```

Classical multidimensional scaling (MDS) of a data matrix. Also known as principal coordinates analysis

```
outDHell = cmdscale(data.DHell, 3, eig=TRUE)
plot(outDHell$points[,1], outDHell$points[,2], main="Hellinger distance",asp=1,
     xlab="Axis 1", ylab="Axis 2")
names = rownames(data)
text(outDHell$points[,1], outDHell$points[,2], labels= names, pos=2, cex=0.5, offset=0.15)
```



```
data.out = prcomp(data[,-1], scale=FALSE)
```

PCA

The values returned, by the function prcomp()

```
names(data.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

The standard deviations of the principal components (the square roots of the eigenvalues)

```
data.out$sdev
```

```
##  [1] 21.0738583 18.1325967  5.4141315  3.1737348  2.6736409  2.2126442
##  [7]  1.0668062  0.7732629  0.6220504  0.5196187  0.3523081  0.2447178
## [13]  0.1532113
```

The matrix of variable loadings (columns are eigenvectors)

```
data.out$rotation
```

```
##                  PC1          PC2          PC3         PC4          PC5
## Pat.fle -0.489926693 -0.806837465 -0.319464254 -0.02543856  0.003799339
## Tur.set -0.126269799 -0.152783728  0.376319777 -0.09639962  0.247890408
## Ner.pli  0.038327483 -0.023116751 -0.001402709 -0.01314119  0.015564458
## Tec.gra  0.841006319 -0.531006033  0.037470986  0.06200510 -0.048032019
## Dru.ric -0.185381440 -0.190721189  0.808957423  0.30258706 -0.309315885
## Mor.uva -0.018033174  0.049212036 -0.195733807  0.88958026  0.271632278
## Mor.gra -0.014752452  0.040411365 -0.157031168  0.27903635 -0.439297328
## Mit.lit -0.016945543  0.046821665 -0.175760510  0.01186565 -0.727136406
## Con.ebr -0.001115264  0.003232606 -0.011083720 -0.01573996  0.004589836
## Con.mil -0.007364408  0.020487212 -0.069046836 -0.07133257 -0.203835033
## Con.spo -0.001135945  0.003230676 -0.010847172 -0.01991969 -0.019562428
## Con.nan -0.001889411  0.005422943 -0.019784991 -0.02794473 -0.035194023
## Con.ver -0.003248274  0.008768235 -0.035301180  0.13643618 -0.040318496
##                  PC6          PC7           PC8          PC9          PC10
## Pat.fle  0.078410324  0.009769905 -0.0024812313  0.002484073 -0.0021785372
## Tur.set -0.863042792 -0.053077104  0.0181069835 -0.014413992  0.0044302375
## Ner.pli  0.016632681  0.054593571 -0.0579047572 -0.992434448  0.0698441540
## Tec.gra -0.032790150 -0.018488002  0.0071904799  0.041342981  0.0002640451
## Dru.ric  0.293495024 -0.033202576  0.0061803827 -0.008761684  0.0127101166
## Mor.uva -0.098646491 -0.253505171 -0.1158471990 -0.017015186 -0.0179646748
## Mor.gra -0.255313707  0.597091314  0.4390239180  0.009469781  0.1677332764
## Mit.lit -0.288123281 -0.268040931 -0.5135388570 -0.011472542 -0.0866763151
## Con.ebr  0.010407987 -0.069668473 -0.0110094316  0.045017278  0.4653212457
## Con.mil -0.027317143 -0.640871457  0.6742624338 -0.087486154 -0.2118626181
## Con.spo  0.002555158 -0.102113228  0.0001297512  0.014706871  0.1888235891
## Con.nan -0.001505530 -0.254396076  0.0715315188  0.024434330  0.8072280696
## Con.ver -0.052184281  0.096537294  0.2578188816 -0.045490666 -0.1014135311
##                 PC11          PC12         PC13
## Pat.fle  0.001475831 -0.0009347317  0.001556315
## Tur.set -0.007240477  0.0047083620 -0.008004362
## Ner.pli -0.024836821  0.0128989549 -0.019546218
## Tec.gra -0.001494375  0.0010950223 -0.001932735
## Dru.ric -0.006173145  0.0038644436 -0.006363865
## Mor.uva -0.070266763  0.0288751594 -0.002024941
## Mor.gra -0.209366373  0.1261861480  0.014852421
## Mit.lit  0.080798795 -0.0698149082 -0.011427318
## Con.ebr -0.130262080 -0.3177082697 -0.811206659
## Con.mil -0.160139376  0.0156620263 -0.059997684
## Con.spo  0.377584987  0.8562411757 -0.277610264
## Con.nan  0.106262118 -0.0934921163  0.505443423
## Con.ver  0.864588433 -0.3675459576 -0.070395049
```

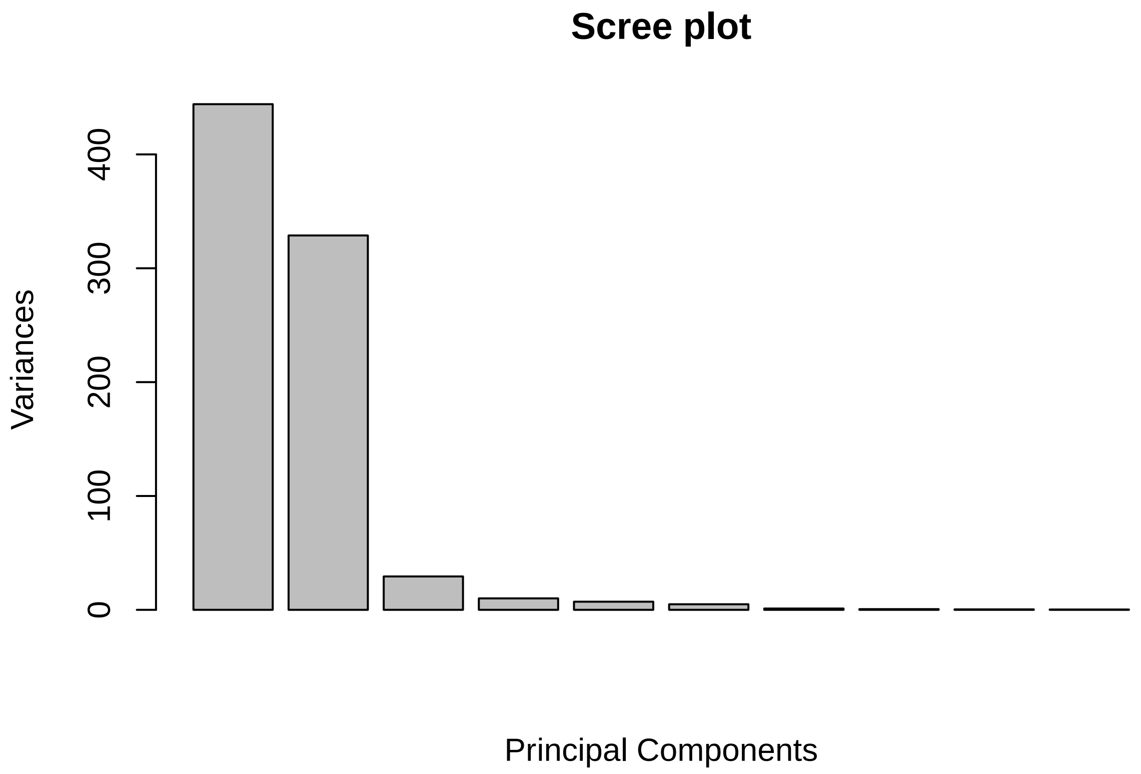
PCA outputs:

```
summary(data.out)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     21.0739 18.1326 5.41413 3.17373 2.67364 2.21264
## Proportion of Variance  0.5371  0.3976 0.03545 0.01218 0.00864 0.00592
## Cumulative Proportion   0.5371  0.9347 0.97011 0.98229 0.99093 0.99686
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     1.06681 0.77326 0.62205 0.51962 0.35231 0.24472
## Proportion of Variance 0.00138 0.00072 0.00047 0.00033 0.00015 0.00007
## Cumulative Proportion  0.99823 0.99895 0.99942 0.99975 0.99990 0.99997
##                           PC13
## Standard deviation     0.15321
## Proportion of Variance 0.00003
## Cumulative Proportion  1.00000
```

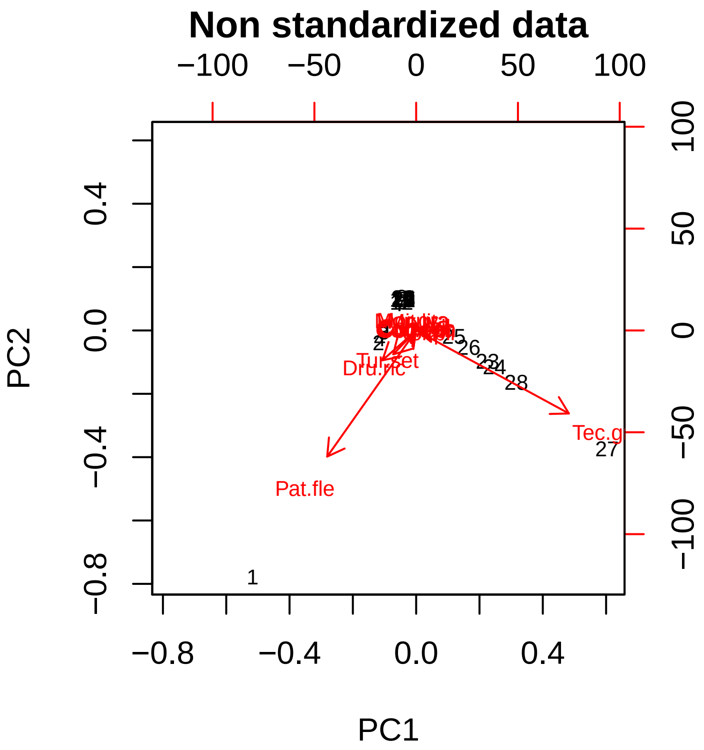
Shows a screeplot.

```
plot(data.out, main="Scree plot", xlab="Principal Components")
```



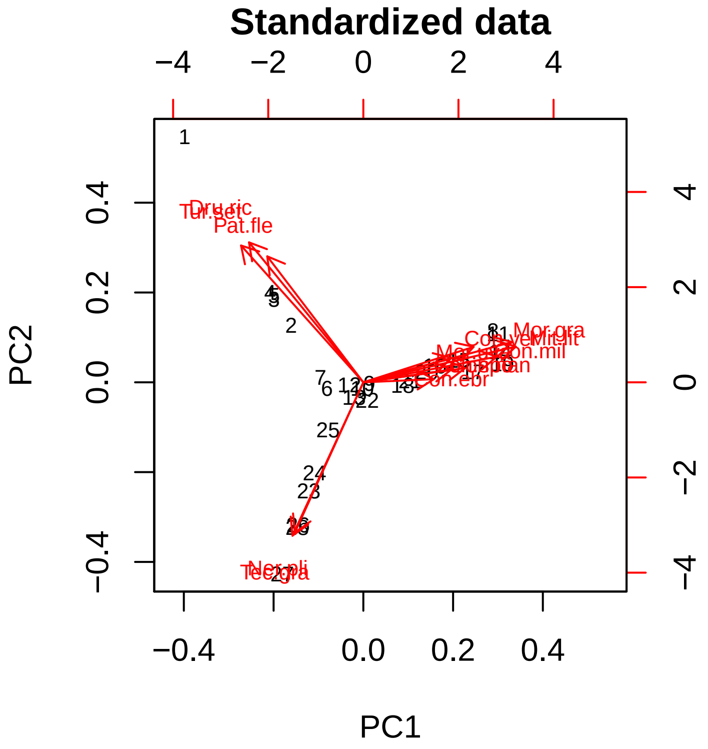
Shows coordinates of individuals on the principal components. Shows a biplot graph

```
biplot(data.out, main="Non standardized data", cex=0.7)
```



Repeat the analysis using the argument "scale=TRUE" means that the data is standardized and show a biplot graph.

```
data.out = prcomp(data[,-1], scale=TRUE)
biplot(data.out, main="Standardized data", cex=0.7)
```


Standardized data
−4
−2
0
2
4
4
2
0
−2
−4
0.4
0.2
0.0
−0.4
−0.4
0.0
0.2
0.4
PC1
PC2
1
Dru.ric
Tur.sel
Pat.fle
4
5
3
2
7
6
25
24
23
26
Ner.pli
Tec.gra
22
18
8
11
Mor.gra
Con.vel
Mit.lit
Mar.uva
Con.mil
Con.ebr
17