MorphoTools version 1.0.0 tutorial

April 26, 2020

# 1. Introduction

The R package MorphoTools provides useful tools for

# 2. Obtaining and installing MorphoTools package

The R console and base system can be obtained at <http://www.r-project.org/>. Once R is installed, MorphoTools can be installed and loaded by typing the following commands into the R console:

install.packages(MorphoTools)  
library(MorphoTools)

If you quit and restart R, you will need to load it again (using the library function as shown above).

# 3. Data import, control and manipulation

## 3.1 Data import

As with any software, the first thing you want to do is import your raw data. Data can be imported from text files, import from Excel spreadsheets through the clipboard in Windows also usually works well. For this tutorial, go into the “extdata” directory of the MorphoTools package installation, and find a file called “centaurea.txt”. To find the path to the package location run path.package(package = "MorphoTools"). Open this file in a text editor and inspect its contents. This sample data include portions of data sets analysed by Koutecký (2007) and Koutecký et al. (2012), and contains 25 morphological characters of three diploid species of the *Centaurea phrygia* complex: s.str. (abbreviated “ph”), (“ps”) and (“st”) and the putative hybrid of the last two species abbreviated as “hybr”. As you find in centaurea.txt file, following structure of input data is required:

* the first row contains variable names.
* the next rows contains individuals, single individual per row.
* the first three columns include unique identifiers for individuals, populations and taxa/groups, respectively. Columns have to be named as “ID”, “Population” and “Taxon”.
* fourth and next columns stores morphological characters.

Avoid spaces and special characters for column names. Missing values have to be represented as empty cells or by the text NA (not quoted).

Read the centaurea.txt file using the read.morphodata() function, and assign the dataset a name of your choice (centaurea in this example) by typing:

centaurea = read.morphodata(file = system.file("extdata", "centaurea.txt", package = "MorphoTools"),  
 dec = ".", sep = "\t")

Argument dec stands for the character used in the file for decimal points, sep is a column delimiter character, usualy whitespace " ", coma "," or tab "\t". Alternatively, you can use build-in data running data(centaurea). The dataset now exists as a morphodata object in R.

The morphodata object, as well as other objects used later, is defined as a list. In R lists act as containers for data. Elements stored in morphodata object can be referenced by the $ notation. Type centaurea$ and press the tab key to see contained elements. Command centaurea$Taxon print values on R console. Run ?morphodata to see what a morphodata object contains.

Alternatively, the following commands display basic information about the dataset or show data in data viewer.

summary(centaurea)  
#> object of class 'morphodata'  
#> - contains 33 populations  
#> - contains 4 defined groups (taxa)  
#>   
#> Populations: BABL, BABU, BOL, BRT, BUK, CERM, CERV, CZLE, DEB, DOM, DUB, HVLT, KASH, KOT, KOZH, KRO, LES, LIP, MIL, NEJ, NSED, OLE1, OLE2, PREL, PRIS, PROS, RTE, RUS, SOK, STCV, STGH, VIT, VOL  
#> Groups (taxa): hybr, ph, ps, st

samples(centaurea)  
#> [1] "BABL1146" "BABL1147" "BABL1148" "BABL1149" "BABL1150" "BABL1151"  
#> [7] "BABL1152" "BABL1153" "BABL1154" "BABL1155" "BABL1156" "BABL1157"  
#> [13] "BABL1158" "BABL1159" "BABL1164" "BABL1165" "BABL1166" "BABL1170"  
#> [19] "BABL1171" "BABL1174" "BABU834" "BABU835" "BABU836" "BABU837"   
#> [25] "BABU838" "BABU839" "BABU840" "BABU841" "BABU842" "BABU843"   
#> [31] "BABU845" "BABU848" "BABU851" "BABU852" "BABU854" "BABU855"   
#> [37] "BABU856" "BABU857" "BABU859" "BABU860" "BOL1176" "BOL1177"   
#> [43] "BOL1178" "BOL1179" "BOL1180" "BOL1181" "BOL1182" "BOL1183"   
#> [49] "BOL1184" "BOL1185" "BOL1186" "BOL1187" "BOL1188" "BOL1189"   
#> [55] "BOL1190" "BOL1191" "BOL1192" "BOL1193" "BOL1194" "BOL1200"   
#> [ reached getOption("max.print") -- omitted 592 entries ]

populations(centaurea)  
#> [1] "BABL" "BABU" "BOL" "BRT" "BUK" "CERM" "CERV" "CZLE" "DEB" "DOM"   
#> [11] "DUB" "HVLT" "KASH" "KOT" "KOZH" "KRO" "LES" "LIP" "MIL" "NEJ"   
#> [21] "NSED" "OLE1" "OLE2" "PREL" "PRIS" "PROS" "RTE" "RUS" "SOK" "STCV"  
#> [31] "STGH" "VIT" "VOL"

taxa(centaurea)  
#> [1] "hybr" "ph" "ps" "st"

viewMorphodata(centaurea)

## 3.2 Data control and manipulation

Row data can contain errors (e.g., typos in numbers or in decimal points) and missing values which should be corrected or removed. We should also consider removing significantly correlated characters, that could potentially distort the results of some of the multivariate analyses. Discriminant analyses generally require a multivariate normal distribution of quantitative (not binary) characters within-groups; nevertheless, they have been shown to be considerably robust to deviations in this respect (Thorpe, 1976; Klecka, 1980). Following steps go through these issues and at the end, we will have cleaned data ready for exploring the morphological differentiation among the taxa (or any defined groups).

### 3.2.1 Shapiro-Wilk normality test

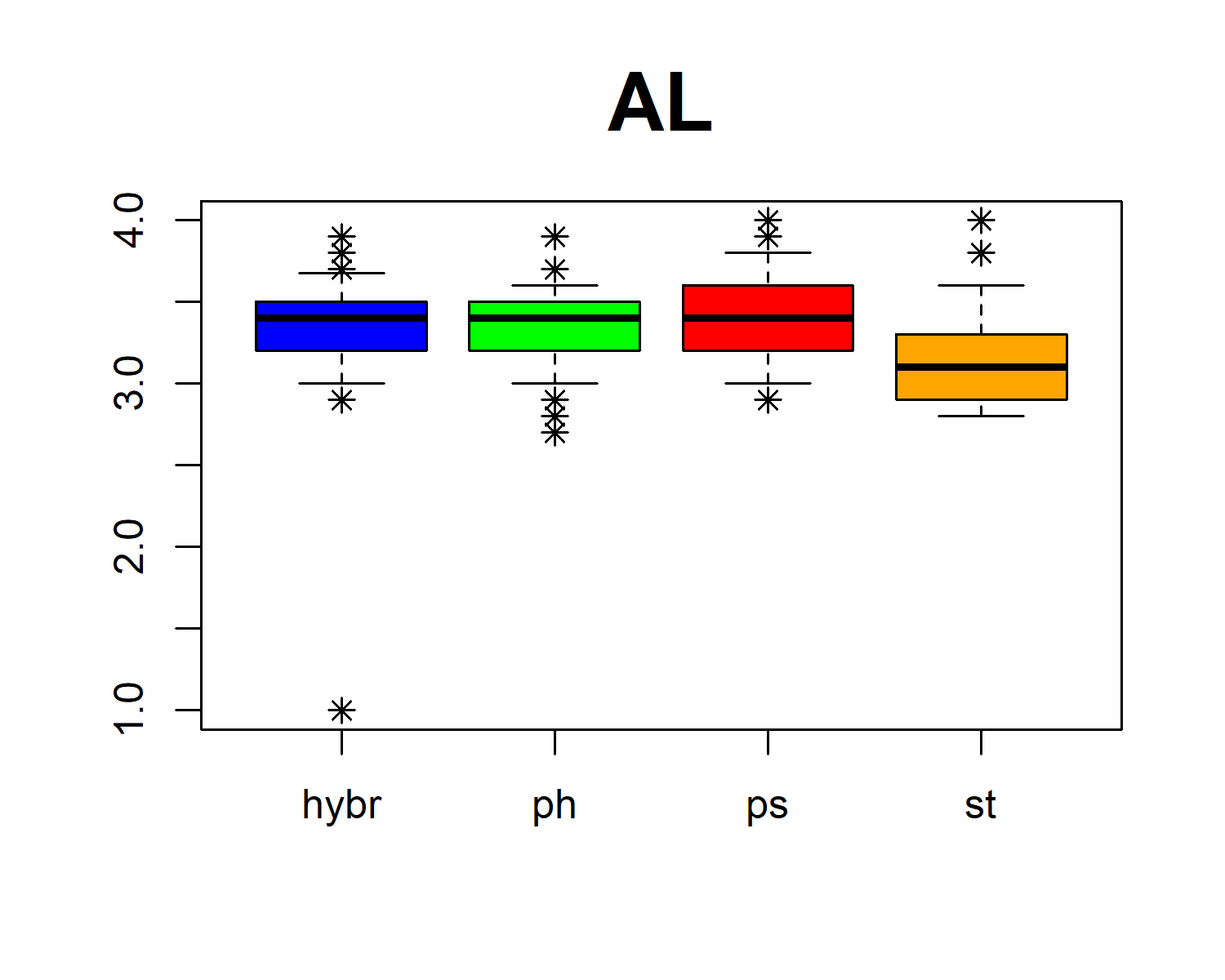
As the first step, the Shapiro-Wilk statistic for the test of normality of distribution was computed for each character  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

### 3.2.2 Box Plots

Boxplots are handy tools for detecting outlier values (potential typos, missing decimal points, etc.) and for detection between-species dissimilarities and critical morphological values discriminating among species. Exactly these values should be transformed into identification key enabling the correct identification of taxa.

Boxplots can be calculated for particular character using boxplot.character() or for all characters at once running boxplot.all() function, that saves all boxplots to a new folder in your working directory. The whiskers can be extended to desired percentiles using lowerWhisker and upperWhisker arguments. Missing data are omitted.

boxplot.character(centaurea, character = "AL", lowerWhisker = 0.05,  
 upperWhisker = 0.95, col = c("blue","green","red","orange"))



The default behaviour is to plot outliers. As you can see in the above figure, hybrids contain one markedly different value, which will require examination.

### 3.2.3 Descriptive statistics

The table of descriptive statistics is a less comfortable way of detecting outlier values, but it can be used for reporting descriptive statistics of morphological characters in your paper. This statistics can be calculated on the levels of populations, taxa or for the whole dataset, using descr.pop(), descr.tax() or descr.all(), respectively. Using argument format you can specify the desired output format. Keywords $MEAN, $SD, $MIN, $5%, $25%, $MEDIAN, $75%, $95%, $MAX will be replaced by actual values. Run ?descr.tax for more details.

descr.tax(centaurea, format = "($MEAN ± $SD)", decimalPlaces = 2)  
#> group hybr ph ps st  
#> 1 format (MEAN ± SD) (MEAN ± SD) (MEAN ± SD) (MEAN ± SD)  
#> 2 N 120 160 240 132  
#> 3 SN (48.32 ± 23.07) (45.22 ± 17.03) (49.07 ± 19.13) (55.47 ± 14.8)  
#> 4 SF (16.33 ± 9.83) (20.12 ± 10.21) (18.85 ± 9.08) (9.64 ± 7.96)  
#> 5 ST (64.65 ± 23.13) (65.34 ± 17.63) (67.93 ± 20.93) (65.11 ± 17.17)  
#> 6 SFT (0.27 ± 0.16) (0.31 ± 0.14) (0.29 ± 0.14) (0.14 ± 0.1)  
#> 7 LL (10.78 ± 2.56) (9.92 ± 2.41) (10.01 ± 2.17) (8.96 ± 2.32)  
#> 8 LW (3.6 ± 0.78) (3.29 ± 1.05) (3.38 ± 0.86) (2.57 ± 0.73)  
#> 9 LLW (3.02 ± 0.49) (3.17 ± 0.7) (3.07 ± 0.78) (3.58 ± 0.78)  
#> 10 LM (0 ± 0) (0.14 ± 0.35) (0.02 ± 0.14) (0.01 ± 0.09)  
#> 11 LBA (0.47 ± 0.5) (0.12 ± 0.33) (0.04 ± 0.2) (0.57 ± 0.5)  
#> 12 LBS (0.17 ± 0.37) (0.04 ± 0.19) (0.41 ± 0.49) (0.01 ± 0.09)  
#> [ reached 'max' / getOption("max.print") -- omitted 15 rows ]

The results can be assigned to a variable and copied to the clipboard (fails for large data) or exported to file using the export.res() function.

descr\_tax = descr.tax(centaurea, format = "($MEAN ± $SD)", decimalPlaces = 2)  
export.res(descr\_tax, file = "clipboard")  
export.res(descr\_tax, file = "descr\_tax.txt")

### 3.2.4 Correlations of characters

Highly correlated characters (r > |0.95|) should not be used in principal component analysis and especially in discriminant analysis, as this could potentially distort the results. Function cormat calculates the correlation coefficients of the characters, Pearson’s (default) or Spearman’s. The results can be exported with the export.res() function. One from the pair of highly correlated character can be removed from dataset using delete.charecter() function, see below.

correlations.s = cormat(centaurea, method = "spearman")  
export.res(correlations.s, file="correlations.spearman.txt")

Significance tests are usually unnecessary for morphometric analysis. Anyway, if tests are needed, they can be computed using the cormat.signifTest() function.

### 3.2.5 Populations as the taxonomic units

To simplify the overall structure, we can consider the use of populations as operational taxonomic units. This means that the whole population will be represented by a single individual, whose morphological values will be calculated as an average of the individuals’ values. Missing values will be removed.

pops = popul.otu(centaurea)  
#> Warning: Unable to calculate the means of characters AL AW ALW AP in populations  
#> PREL PRIS PROS RTE. Values are NA.

We received a warning that the values of some characters are NA. How to deal with missing data, we will discuss in the following section.

### 3.2.6 Missing data

Missing values are not accepted in morphological analyses, and MorphoTools do not edit user’s data in the background. User has to do his own decision, what to do with missing values. Basically, there are two options: remove or replace.

#### Replace missing values

Missing values can be substituted by the average value of the respective character in the respective population. This approach is acceptable only if: there are relatively few missing values; these missing values are scattered throughout many characters (each character includes only a few missing values); and removing all individuals or all characters with missing data would unacceptably reduce the data set. To substitute missing values by average value, run:

centaurea = na.meanSubst(centaurea)  
#> Warning: Unable to replace NAs in characters AL, AW, ALW, AP in population LIP.  
#> Probably all values of that character are NA.  
#> Warning: Unable to replace NAs in characters AL, AW, ALW, AP in population PREL.  
#> Probably all values of that character are NA.

#### Remove items

After the execution of the previous code, we received warnings. Characters AL, AW, ALW, AP in populations LIP and PREL contains only NA values. We have to decide between deleting characters using delete.charecter() or populations using delete.population() function. As character AP looks promising for delimitation of and , we will keep the characters and remove the populations.

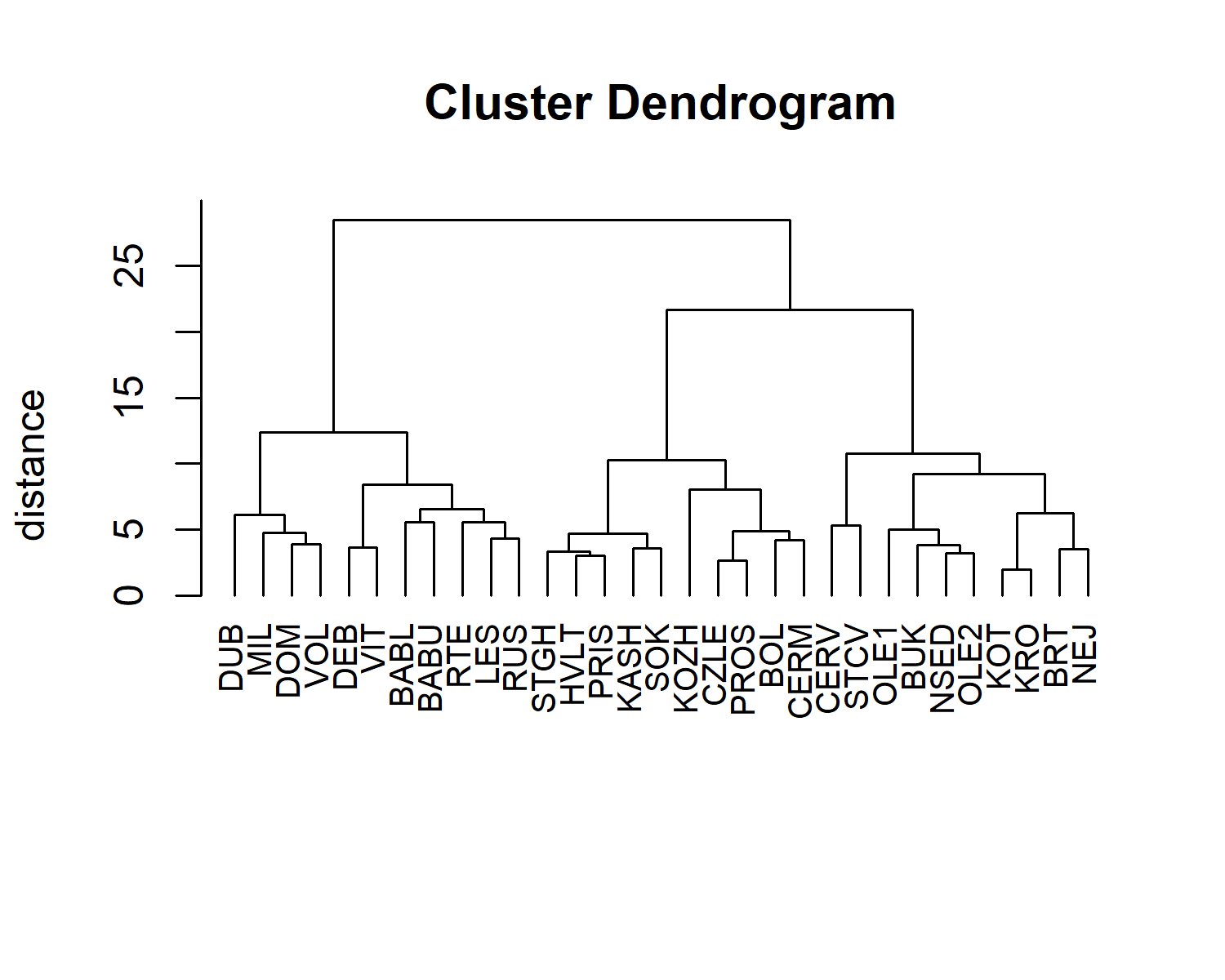
centaurea = delete.population(centaurea, populationName = c("LIP", "PREL"))  
pops = delete.population(pops, populationName = c("LIP", "PREL"))

We examined outlier value in “hybr” group and ensured that data do not contain highly correlated characters. Missing values were replaced by the average values, and populations containing only NAs in some characters were removed. Our dataset is now prepared for further analyses. It is not a bad idea to save a copy of it using export.res() function.

# 4. Hierarchical clustering

Hierarchical classification is one of the methods that do not require specification of the samples’ membership in taxa (groups). Therefore, this method is recommended to use first in order to be able to get an insight on the existence of group structure in your data. Typically, populations are used as OTUs. Characters are standardised to a zero mean and a unit standard deviation using the scale function. Various measures of distance between the observations (rows) are applicable, and various procedures of clustering based on pre-computed distances are available. For further details, run ?clust.

pops\_hierClust = clust(pops, distMethod = "euclidean", clustMethod = "ward")  
plot(pops\_hierClust, hang = -1, cex = 0.8, sub = "", xlab = "", ylab = "distance")



Four main clusters were formed in the histogram above, however, some populations (BABL, LES, PROS, BUK, and NSED) were “incorrectly” clustered, what will require further inspection.

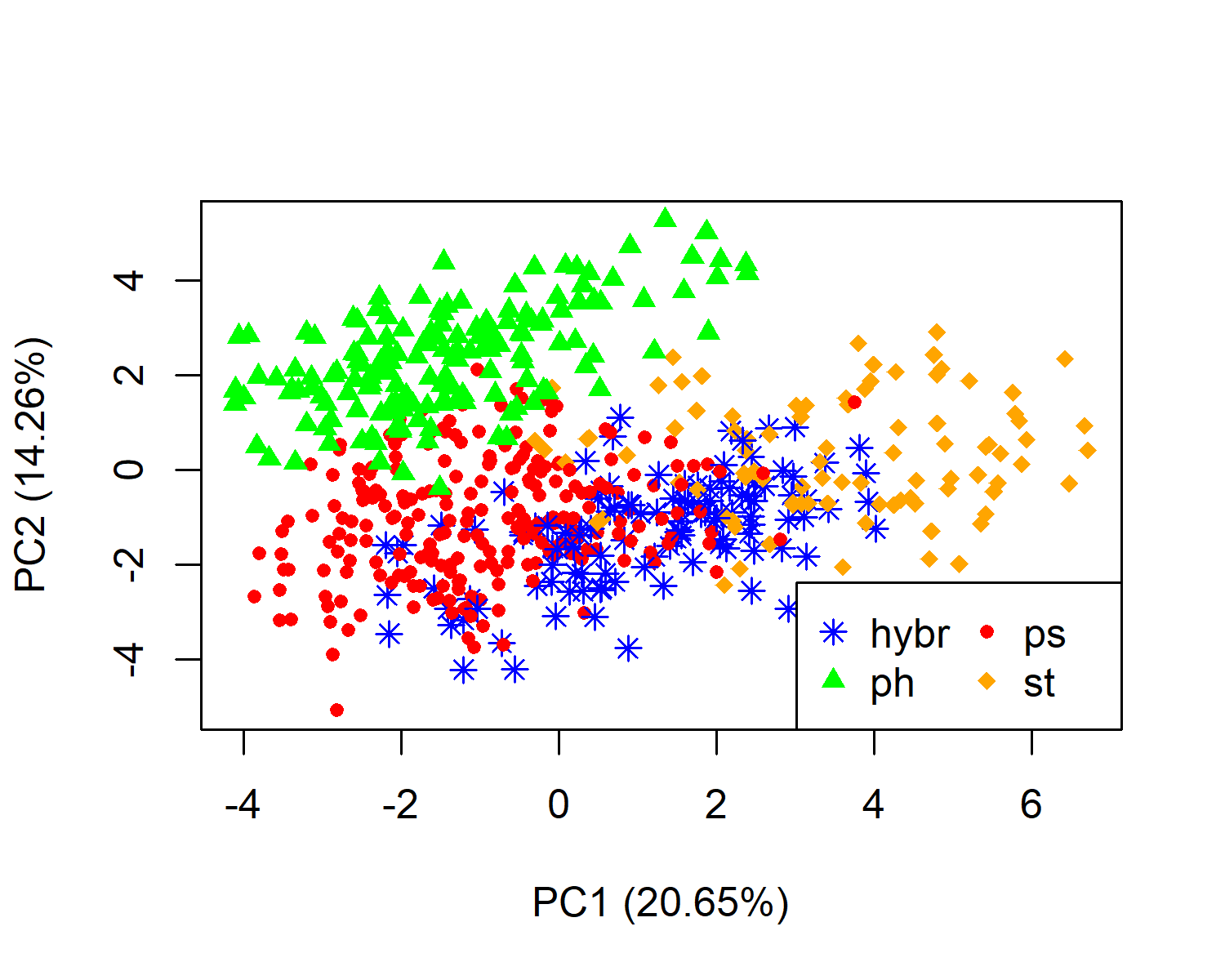
# 5. Principal component analysis (PCA)

Principal components analysis (PCA) is another method without the requirement of specification of the samples’ membership in taxa (groups). PCA reduce the measured variables into a number of principal components (artificial variables). The first few of them extracts most of the variance in the measured variables. PCA is calculated by pca.calc() function, the result is object of class pcadata. Run ?pcadata for the help page about the elements of this object.

pca.centaurea = pca.calc(centaurea)

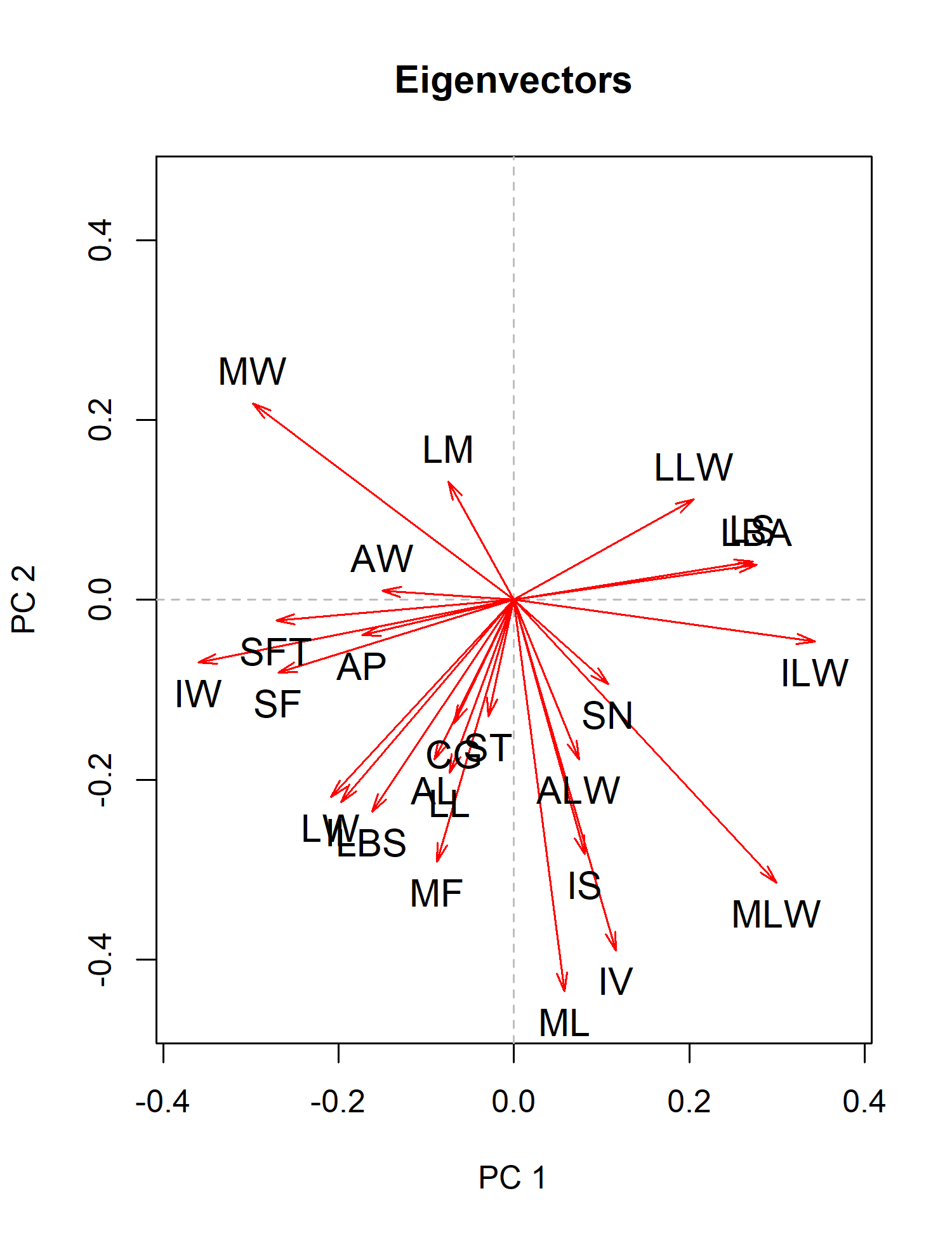
Now we can plot the result by typing following commands:

plot.points(pca.centaurea, col = c("blue","green","red","orange"), pch = c(8,17,20,18),  
 legend = T, ncol = 2, legend.pos="bottomright")



Arrows (eigenvectors) in the next figure shows the influence of the original characters to the main components. Eigenvectors are stored in pca.centaurea$eigenVectors and can be exported by export.res() function. The dollar sign, $, lets you access elements within a object, same as above.

plot.characters(pca.centaurea, cex = 1.2)



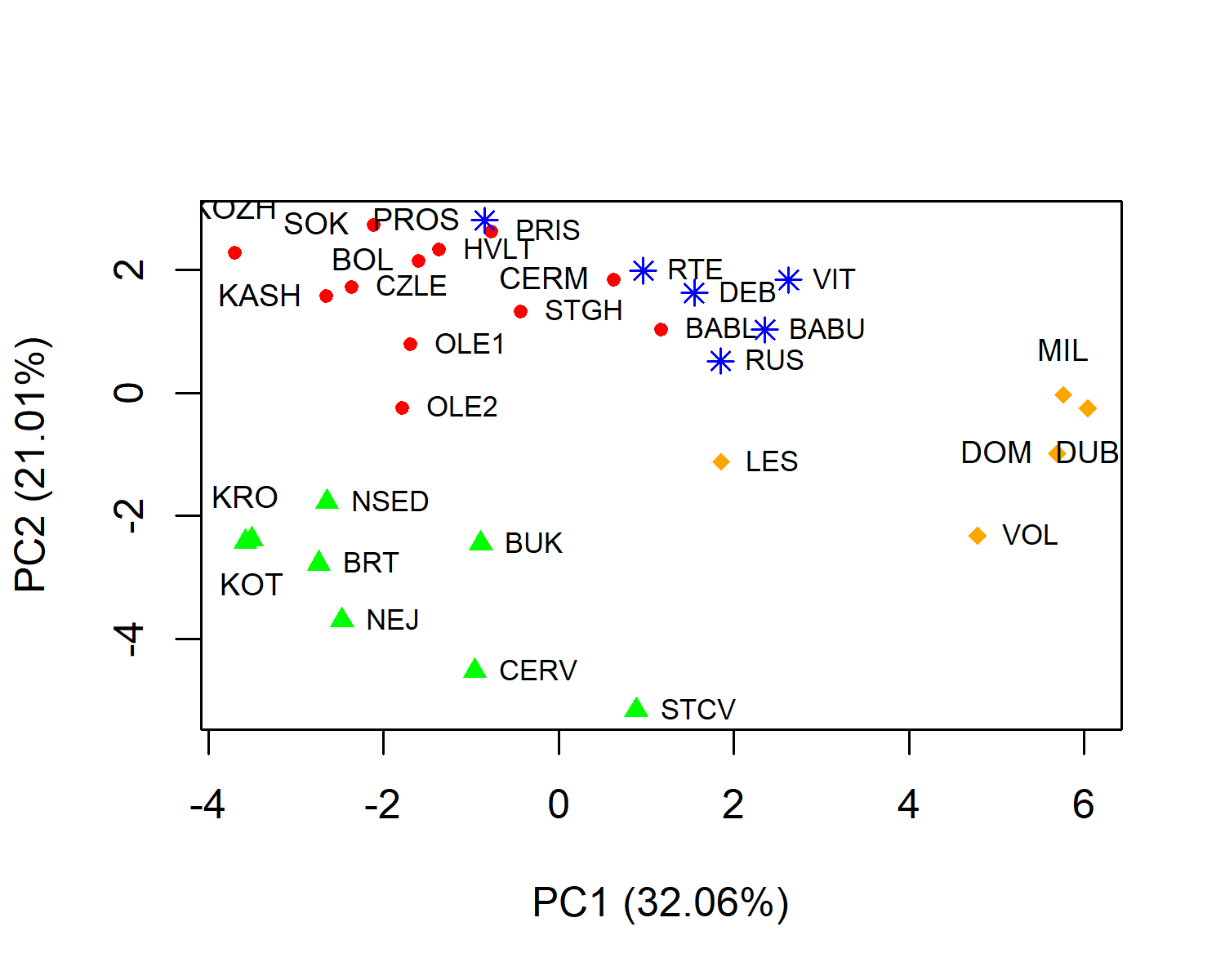
export.res(pca.centaurea$eigenVectors, file="eigenVectors.centaurea.txt")

As you can see in the figures above, ordination diagrams of PCA showed relatively compact groupings corresponding to taxa with partial overlaps. First two components (axes) extracted 20.65% and 14.26% of the overall variability in data. Characters ILW, IW, are strongly correlated with the direction of separation of taxa , (“ps”, “st”) and their putative hybrid “hybr”. The s.str. (“ph”) is separated in the diagonal direction, highly correlated with MW, ML, IV, and MLW characters.

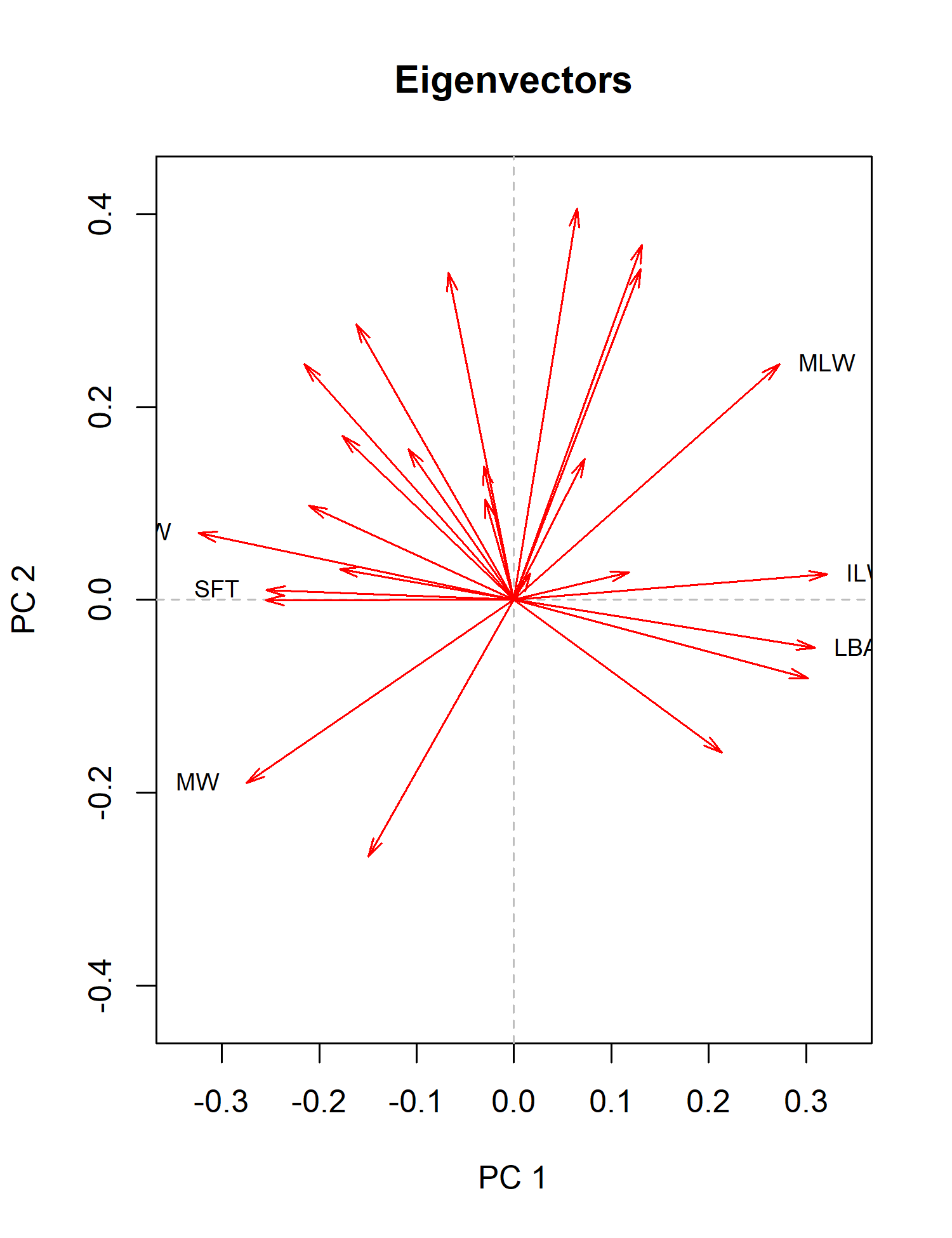
The plot.points() and plot.characters() are a default plotting functions. You can add simple labels using argument labels = TRUE, or point’s legend using legend = TRUE. If you need more precise control about plotting or want to add other elements to points, use some of the following functions:

* plot.addLabels.points(), plot.addLabels.characters() allows you to inclde or exclude labels, speciffy label’s position, offset, colors, etc.
* plot.addLegend() allows you to specify exact coordinates, expansion and interspacing factors, line width, borders parameters, etc.
* plot.addEllipses() draws confidence (prediction) ellipses around taxa.
* plot.addSpiders() connects taxa’s points with its centroids, thus forms a “spider” diagram.

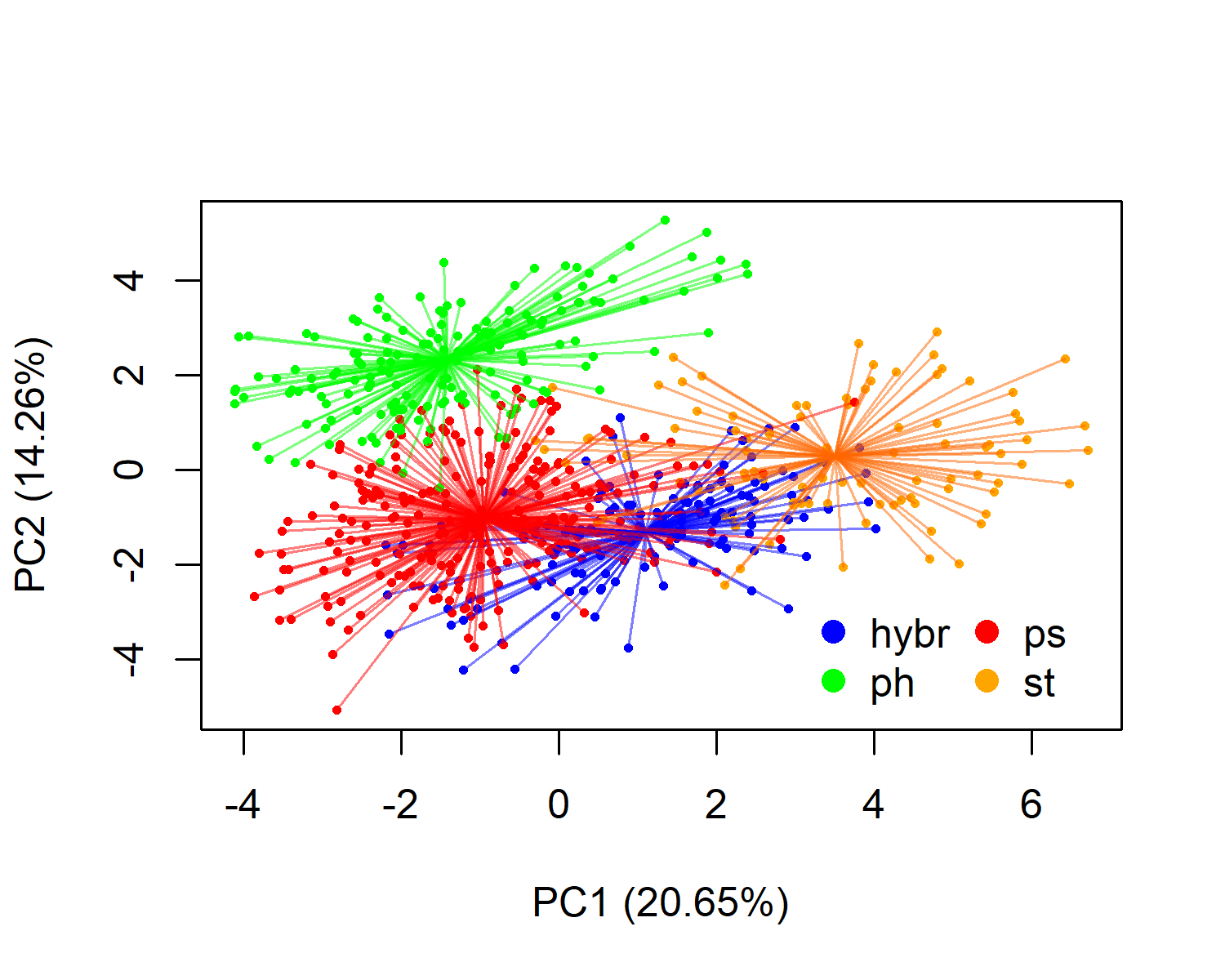
pca.pops = pca.calc(pops)  
  
plot.points(pca.pops, col = c("blue","green","red","orange"), pch = c(8,17,20,18),   
 legend = F, labels = F)  
  
plot.addLabels.points(pca.pops, labels = c("PROS","SOK","KASH","BOL","KRO","DUB","MIL",  
 "CERM","DOM" ,"KOZH","KOT"), include = FALSE, pos = 4, cex=0.7)  
plot.addLabels.points(pca.pops, labels = c("PROS","SOK","KASH","BOL","CERM", "DOM"),   
 pos = 2,cex = 0.75)  
plot.addLabels.points(pca.pops,labels=c("KRO","MIL","KOZH"),pos=3,offset=0.7,cex=0.75)  
plot.addLabels.points(pca.pops, labels=c("DUB","KOT"), pos=1, offset=0.7, cex=0.75)



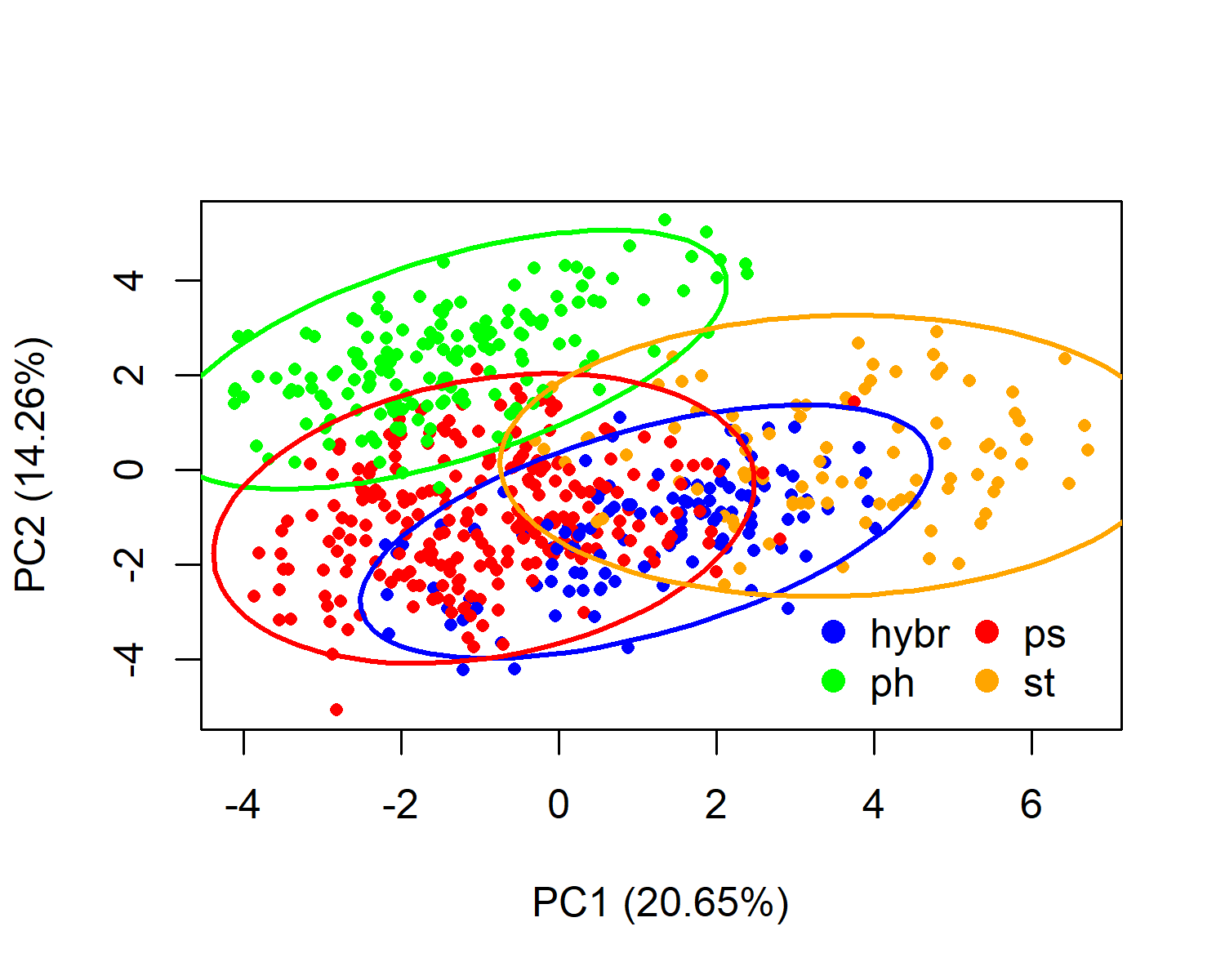
plot.characters(pca.pops, labels = F)  
  
plot.addLabels.characters(pca.pops, labels = c("ILW","MLW", "LBA"), pos = 4,cex = 0.75)  
plot.addLabels.characters(pca.pops,labels=c("IW","SFT","MW"),pos=2,offset=0.7,cex=0.75)



plot.points(pca.centaurea, col = c("blue","green","red","orange"), cex = 0.5)  
plot.addLegend(pca.centaurea, col = c("blue","green","red","orange"), x = "bottomright", pt.cex = 1.3, box.type = "n" ,ncol = 2)  
  
plot.addSpiders(pca.centaurea, col = c(rgb(0,0,255,max=255,alpha=130), # blue  
 rgb(0,255,0,max=255,alpha=130), # green  
 rgb(255,0,0,max=255,alpha=130), # red  
 rgb(255,102,0,max=255,alpha=130))) # orange



plot.points(pca.centaurea, col = c("blue","green","red","orange"), cex = 0.7)  
plot.addLegend(pca.centaurea, col = c("blue","green","red","orange"),   
 x = "bottomright", pt.cex = 1.3, box.type = "n" ,ncol = 2)  
  
plot.addEllipses(pca.centaurea, col = c("blue","green","red","orange"), lwd = 2)



# 3D

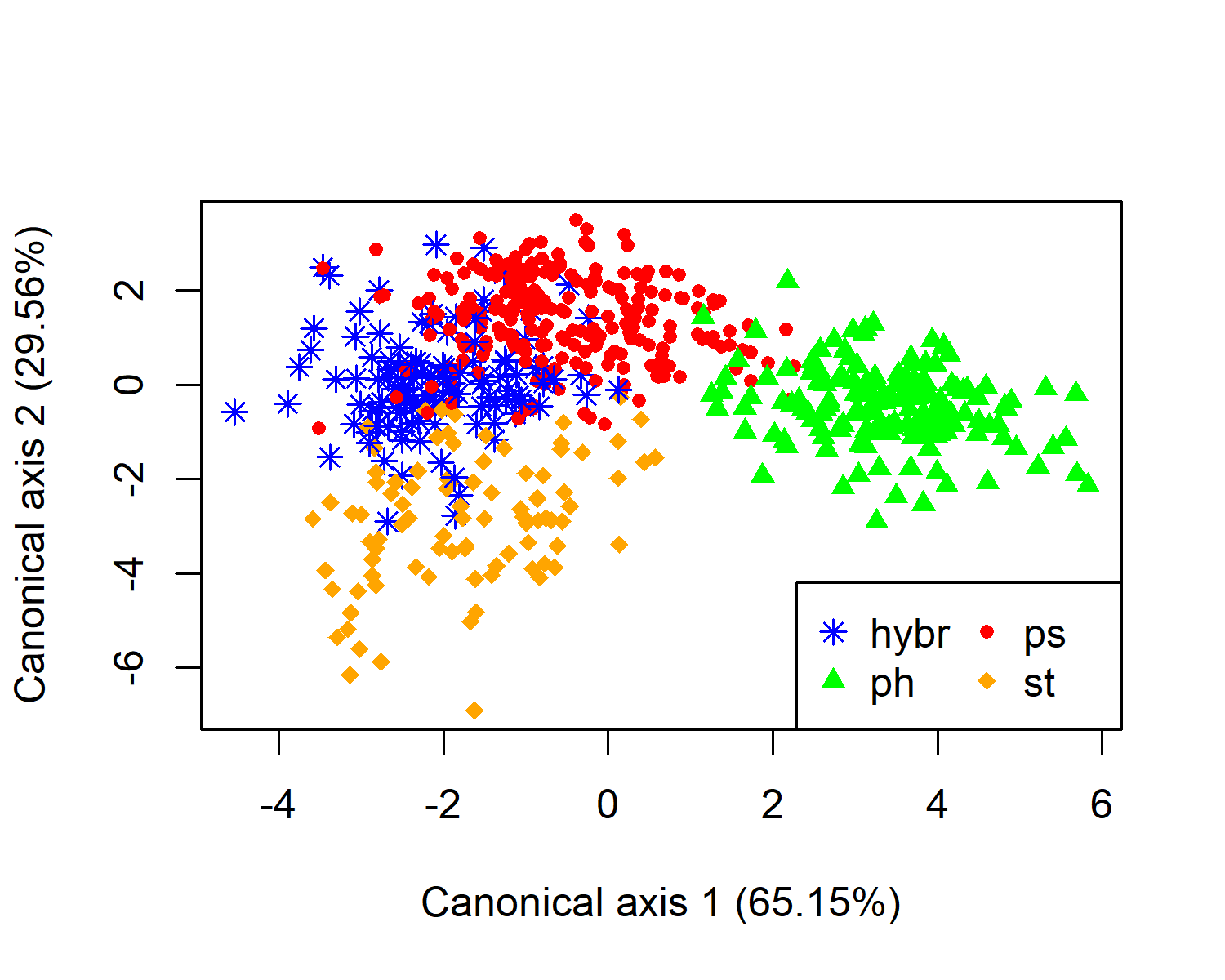
# 3D

# 3D

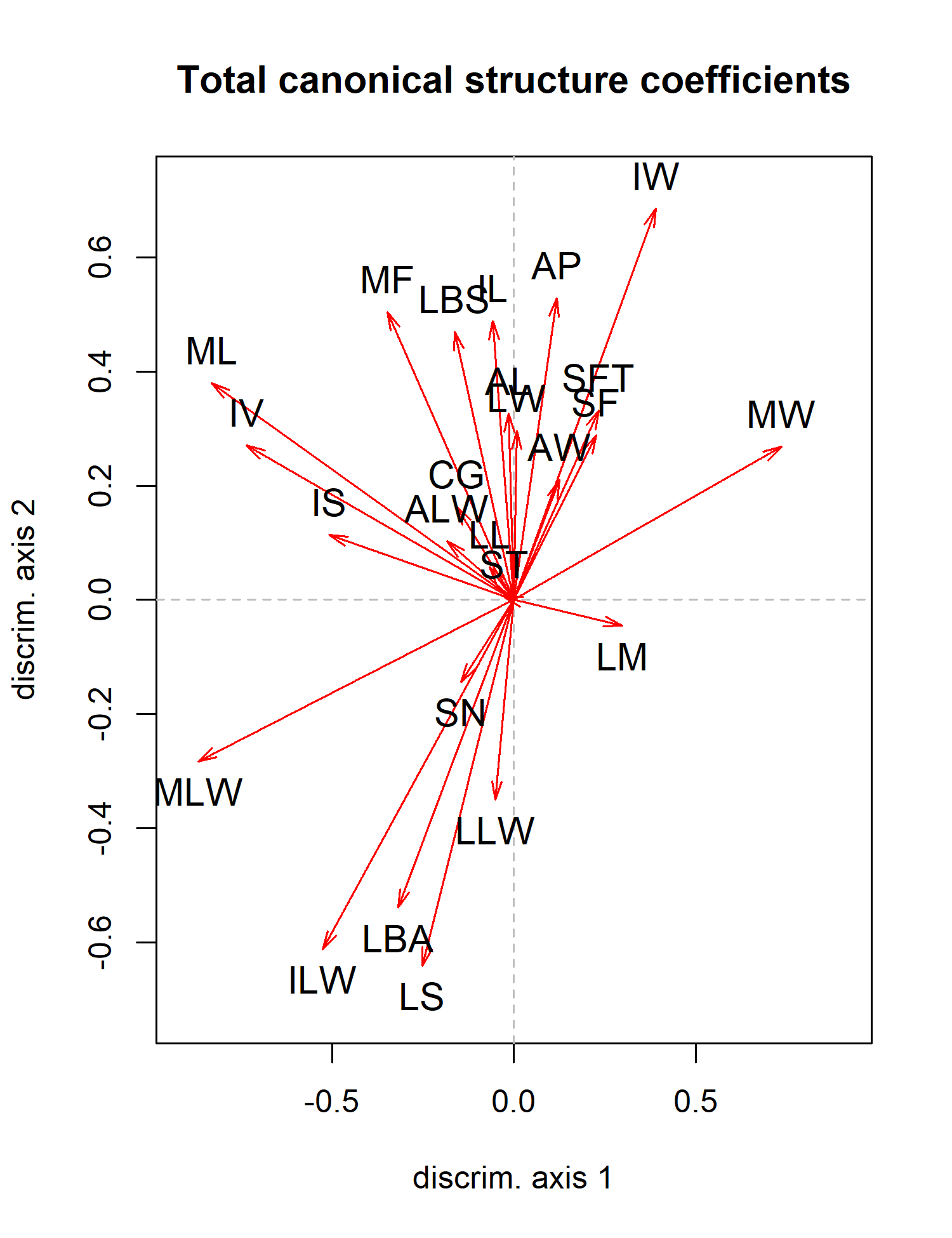
# 6. Canonical discriminant analysis (CDA)

The Canonical discriminant analysis finds linear combinations of the original variables that provide maximal separation among defined groups (Taxon column in input data). CDA can be calculated by cda.calc() function. A result is an object of class cdadata, and among other elements (run ?cdadata for details) it stores total canonical structure coefficients, i.e., total-sample correlations between the original variables and the canonical variables. These coefficients are roughly equivalent to eigenvectors of PCA and we used them to interpret the character’s contribution to group separation.

cda.centaurea = cda.calc(centaurea)  
  
plot.points(cda.centaurea, col = c("blue","green","red","orange"), pch = c(8,17,20,18),   
 legend = T, ncol = 2, legend.pos="bottomright")



plot.characters(cda.centaurea, cex = 1.2)



The CDA ordination diagram unequivocally support morphological differentiation of s.str. (“ph”) along the first axis. Characters ML, MLW, IV and MW are oriented in the direction of separation, thus these characters contributed the most significantly, which is in accordance with the results of PCA. The values of correlation of characters with the first canonical axis (toatal canonical structure coefficients) are one of the elements of the object of class cdadata and can be accesed by the $ notation. Results can be exported using the export.res() function.

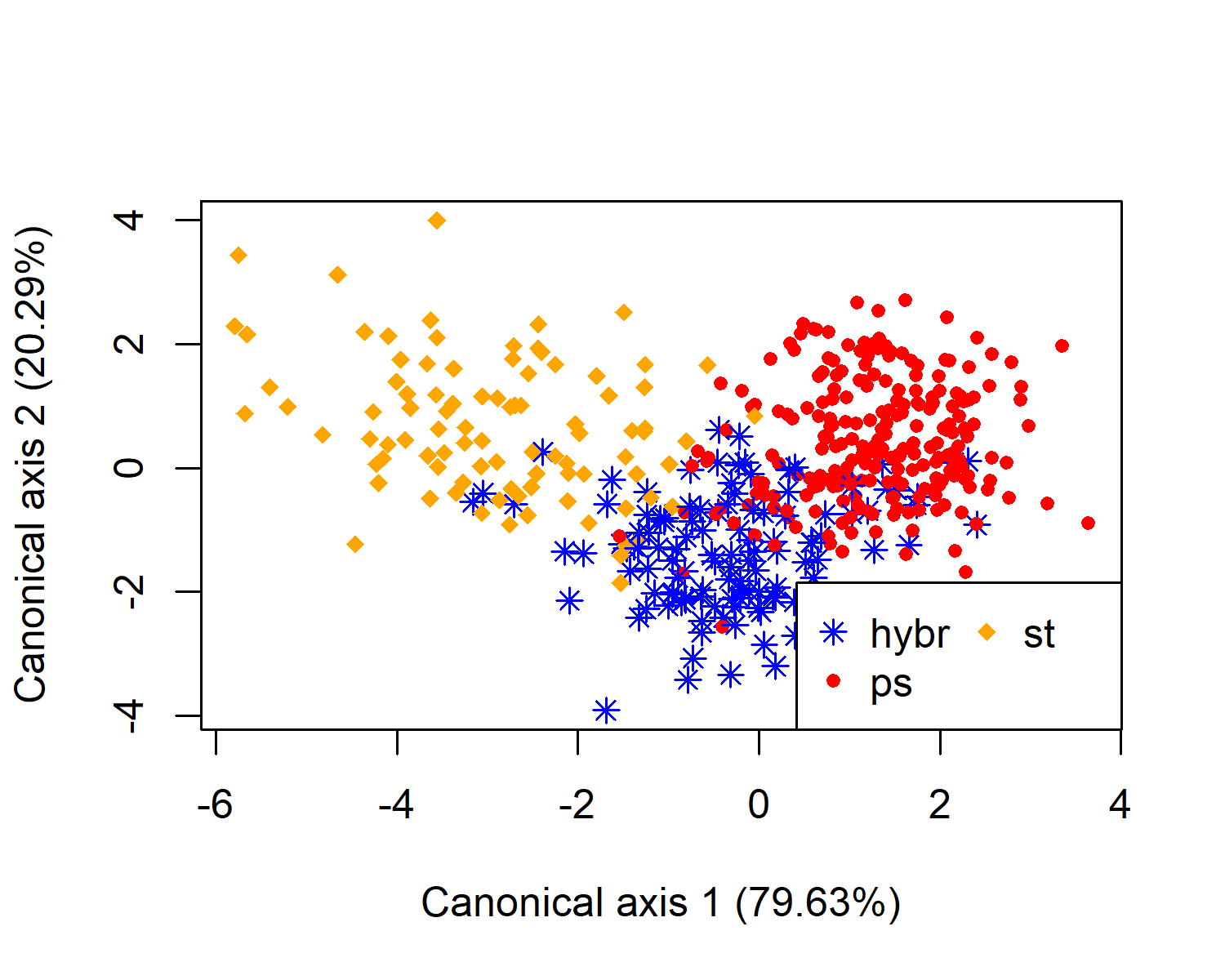
cda.centaurea.TCS = cda.centaurea$totalCanonicalStructure  
  
export.res(cda.centaurea.TCS, "clipboard")  
  
cda.centaurea.TCS  
#> Can1 Can2 Can3  
#> SN -0.145905291 -0.143820071 0.187446717  
#> SF 0.226719447 0.288948819 -0.088309973  
#> ST -0.027765120 0.004883579 0.135653239  
#> SFT 0.233159338 0.331672879 -0.213795111  
#> LL -0.067843561 0.057468347 -0.252875809  
#> LW 0.007404962 0.296609347 -0.354584342  
#> LLW -0.051697329 -0.350145429 0.279507573  
#> LM 0.297463454 -0.045009362 -0.047078365  
#> LBA -0.318778094 -0.539202777 -0.294264249  
#> LBS -0.163214379 0.469973590 0.277373927  
#> LS -0.251889023 -0.640679116 0.197855575  
#> IL -0.058484540 0.488340257 0.343069292  
#> IW 0.389239321 0.685339935 0.239991480  
#> ILW -0.525181687 -0.611784805 -0.002541299  
#> CG -0.157798961 0.163326602 0.015718224  
#> ML -0.831038839 0.379514738 0.079785332  
#> MW 0.735479254 0.267972743 -0.012135822  
#> MLW -0.867777425 -0.282448838 0.009070526  
#> MF -0.347280596 0.504543863 0.175527742  
#> IS -0.508744545 0.113697036 0.012459746  
#> IV -0.735244321 0.270554318 -0.229699765  
#> AL -0.014053761 0.326312832 -0.008420114  
#> AW 0.124084168 0.210514371 0.067441011  
#> ALW -0.183216081 0.102614196 -0.134701510  
#> AP 0.117451381 0.528786582 -0.006020908

As you can see, the above mentioned characters received highest scores at the firs canonical axis (regardless of plus or minus sign).

To gain better insight into the differentiation of , and their putative hybrid, we will analyse them separately. The new dataset will be created by removing from the original dataset.

stPsHybr = delete.taxon(centaurea, taxonName = "ph")  
  
cda.stPsHybr = cda.calc(stPsHybr)  
  
plot.points(cda.stPsHybr, col = c("blue","red","orange"), pch = c(8,20,18),   
 legend = T, ncol = 2, legend.pos="bottomright")



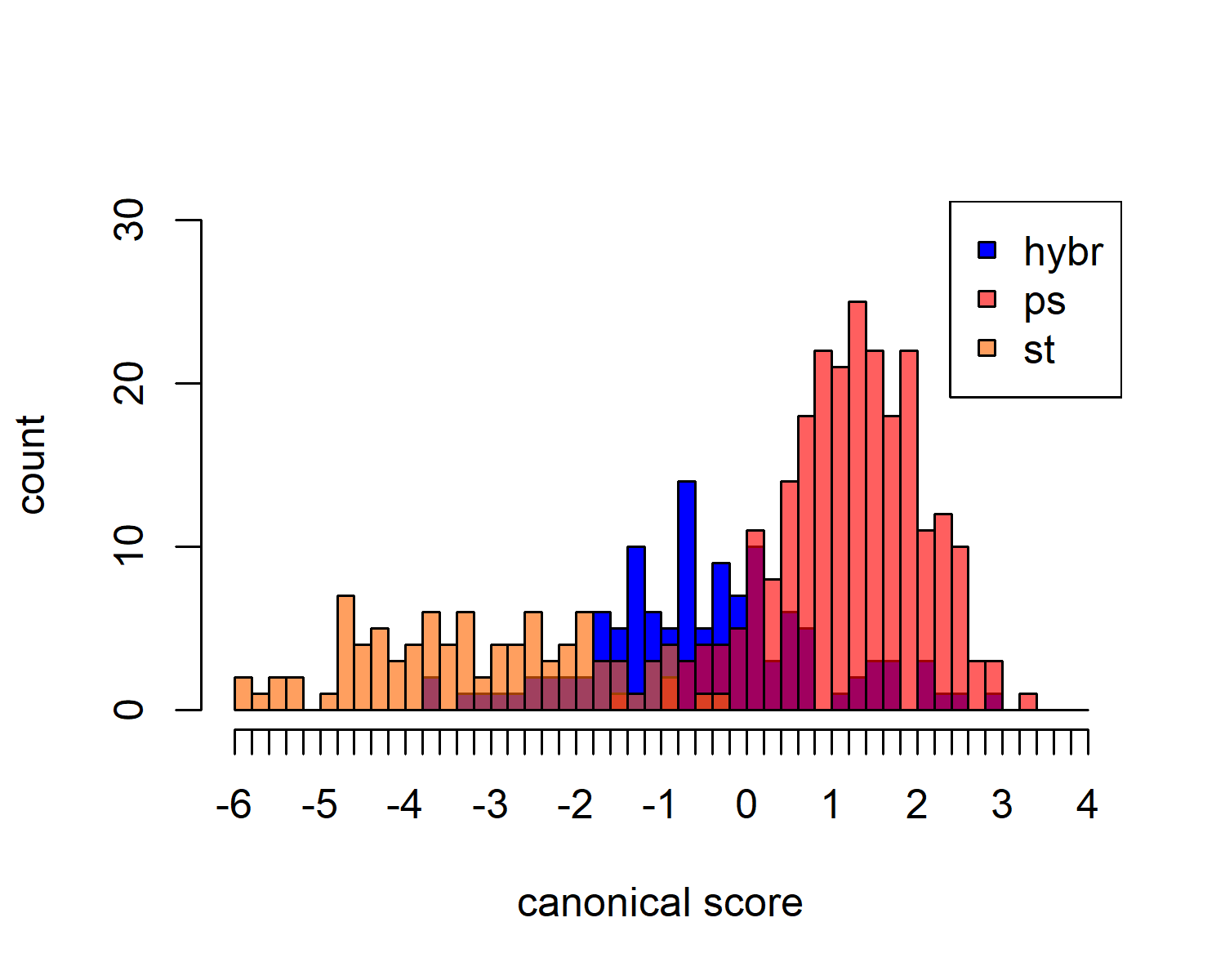
cda.stPsHybr$totalCanonicalStructure  
#> Can1 Can2  
#> SN -0.184060229 0.140729352  
#> SF 0.380685926 -0.045311134  
#> ST 0.001661422 0.112106806  
#> SFT 0.413332646 -0.162634321  
#> LL 0.038953631 -0.265634161  
#> LW 0.332488605 -0.380540862  
#> LLW -0.379824050 0.278389296  
#> LM 0.048811155 0.103057725  
#> LBA -0.644983671 -0.296127557  
#> LBS 0.437841099 0.138703090  
#> LS -0.705219959 0.166523413  
#> IL 0.515834335 0.238410156  
#> IW 0.797627723 0.248437287  
#> ILW -0.759008837 -0.091588297  
#> CG 0.129352318 -0.050925559  
#> ML 0.311426180 -0.358590686  
#> MW 0.699934603 0.275940062  
#> MLW -0.659822961 -0.289017875  
#> MF 0.499559997 -0.005854128  
#> IS -0.007715440 -0.188181984  
#> IV 0.113766105 -0.544282887  
#> AL 0.340820227 -0.048742382  
#> AW 0.272345410 0.079949564  
#> ALW 0.058836906 -0.187449402  
#> AP 0.604687608 -0.030108718

Taxa are separated along the first axis which extracts the majority of variation. Characters IW, ILW, LS, MW and MLW shows the highest correlation with the direction of separation of groups, thus these characters are the most suitable for taxonomic delimitation and identification key. On the other hand, the second axis represents the variation within taxa and to overall separation contribute only marginally. Thus, characters with a high score on the second axis are variable within, not among taxa (eg., IV).

#### Passive prediction of samples.

Sometimes is desirable to passively display (or predict) the position of some samples in the canonical space formed by other samples. This approach is applicable for displaying the position of hybrids, type specimens, “atypical” populations, etc. Passive samples can be specified using passiveSamples argument (accepting both populations and taxa). These samples will be excluded from computing discriminant function, and only passively predicted in multidimensional space. If there are only two groups (except passive samples), there is only one canonical axis, and canonical scores are plotted as a histogram.

cda.stPs\_passiveHybr = cda.calc(stPsHybr, passiveSamples = "hybr")  
  
plot.points(cda.stPs\_passiveHybr, legend = T, breaks = 0.2,  
 col = c(rgb(0,0,255,max=255,alpha=255), # blue  
 rgb(255,0,0,max=255,alpha=160), # red  
 rgb(255,102,0,max=255,alpha=160))) # orange,



# 3D

# 3D

## How to cite MorphoTools

s.str. (abbreviated “ph”), (“ps”) and (“st”)

sssssssssssssssssssssssssssssssssssssss

Feel free to email me at [marek.slenker@savba.sk](mailto:marek.slenker@savba.sk) with any questions, comments, or bug reports!

## References

**Klecka WR. 1980**. *Discriminant analysis (No. 19)*. Sage University Paper Series on Quantitative Applications in the Social Sciences 07-019.

**Koutecký, P. (2007)**. Morphological and ploidy level variation of Centaurea phrygia agg.(Asteraceae) in the Czech Republic, Slovakia and Ukraine. *Folia Geobotanica*, **42**: 77-102.

**Koutecký, P., Štěpánek, J. & Baďurová, T. (2012)**. Differentiation between diploid and tetraploid Centaurea phrygia: mating barriers, morphology and geographic distribution. *Preslia*, **84**: 1-32.

**Krzanowski W. 1990**. *Principles of multivariate analysis*. Oxford: Oxford University Press.

**Thorpe RS. 1976**. Biometric analysis of geographic variation and racial affinities. *Biological Reviews of the Cambridge Philosophical Society* **51**: 407–425.