MorphoTools2 version 0.9.0 tutorial

developed by **Marek Šlenker**[†‡*]
with contributions from **Petr Koutecký**[§] and **Karol Marhold**[†‡]

https://github.com/MarekSlenker/MorphoTools2

Date: Jun 2021

[†]Institute of Botany, Plant Science and Biodiversity Centre, Slovak Academy of Sciences, Bratislava, Slovakia
[‡]Department of Botany, Faculty of Science, Charles University, Prague, Czechia
[§]Department of Botany, Faculty of Science, University of South Bohemia, České Budějovice, Czechia

[*]author for correspondence: marek.slenker@savba.sk

Contents

1. Introduction **3**

2. Obtaining and installing the MorphoTools2 package **3**

3. Data import, checking and manipulation **3**

3.1 Data import . . . 3

3.2 Assessing normality of data . . . 6

3.2.1 Shapiro-Wilk test of normality . . . 6

3.2.2 Histograms . . . 7

3.2.3 Normal Q-Q plot . . . 7

3.3 Data transformation . . . 8

3.4 Box Plots . . . 10

3.5 Descriptive statistics . . . 11

3.6 Correlations of characters . . . 12

3.7 Populations as operational taxonomic units . . . 12

3.8 Missing data . . . 12

3.8.1 Removing items . . . 14

3.8.2 Replacing missing values . . . 14

4. Hierarchical clustering **15**

5. Principal component analysis (PCA) **16**

6. Principal coordinate analysis (PCoA) **23**

7. Non-metric multidimensional scaling (NMDS) **24**

8. Stepwise discriminant analysis **25**

9. Canonical discriminant analysis (CDA) **26**

Passive prediction of samples. . . . 31

10. Classificatory discriminant analysis **33**

10.1 Linear discriminant analysis (LDA) . . . 35

10.2 Quadratic discriminant analysis (LDA) . . . 36

10.3 K nearest neighbour classificatory DA . . . 37

10.4 Classification of sample individuals based on an independent training set. . . . 38

Further reading **39**

References **40**

1. Introduction

The package `MorphoTools2` is intended for multivariate analyses of morphological data. At the moment, various necessary tools are scattered across several R packages. This package wraps available statistical and graphical tools and provides a comprehensive framework for checking and manipulating input data, performing core statistical analyses and running a wide palette of functions designed to visualize results, making the workflow convenient and fast.

2. Obtaining and installing the MorphoTools2 package

The R console and base system can be downloaded from http://www.r-project.org. Once R is installed, `MorphoTools2` can be installed and loaded by typing the following commands into the R console:

```
install.packages(MorphoTools2)
library(MorphoTools2)
```

To get the latest version of the `MorphoTools2` package, install it using the `devtools::install_github()` function from the `devtools` package.

```
install.packages("devtools")
devtools::install_github("MarekSlenker/MorphoTools2")
```

After quitting or restarting R, the package needs to be loaded again (using the `library` function as shown above).

3. Data import, checking and manipulation

As with any statistical software, the first task is to import raw data. However, raw data may contain errors (e.g. typos in numbers or decimal points) and missing values which should be corrected or removed. One should also consider removing very highly correlated characters that could distort the results of some multivariate analyses. Moreover, an assessment of the normality of distribution of data is a prerequisite for some statistical tests. If this assumption is not met, the distribution of data can be improved by transformation, or non-parametric methods that do not require normality of distribution of data can be preferred. The following chapters go through these issues and end up with a cleaned-up dataset ready for exploring the morphological differentiation among taxa (or any defined groups).

3.1 Data import

Data can be imported from plain text files (tab-, comma-, or space-delimited, see below) or from spreadsheet files. The following structure of input data is required:

- the first row containing variable names;
- rows each containing values for a single individual sample or any other kind of sampling unit;

- the first three columns containing unique identifiers of individuals, populations and taxa/groups (named "ID", "Population" and "Taxon")[1]; and
- the fourth and next columns holding values of morphological characters[2].

ID	Population	Taxon	SN	SF	ST	SFT	LL	LW	LLW
RTE1	RTE	hybr	35.2	23.6	58.8	0.4	11.2	3.9	2.87
RTE2	RTE	hybr	39	11.8	50.8	0.23	7.2	2.6	2.77
RUS112	RUS	hybr	24.8	23.4	48.2	0.49	7.1	2.8	2.54
RUS113	RUS	hybr	30	25.5	55.5	0.46	10.2	3.7	2.76
OLE1272	OLE1	ps	48.6	6.3	54.9	0.11	8.6	3.8	2.26
OLE1273	OLE1	ps	58.1	10	68.1	0.15	11.2	3.7	3.03
OLE1274	OLE1	ps	30.7	26.6	57.3	0.46	7.9	3.1	2.55
STGH309	STGH	ps	77.1	15.5	92.6	0.17	11.6	3.9	2.97
STGH310	STGH	ps	35.6	19.2	54.8	0.35	9.5		*NA*

Use underscores (_) instead of spaces, and avoid special characters (e.g. punctuation marks). Missing values have to be represented as empty cells or by the text `NA` (without quotes).

In this tutorial, we will use the centaurea dataset, containing measurements of 25 morphological characters of three diploid species of the *Centaurea phrygia* complex: *C. phrygia* s.str. (abbreviated "ph"), *C. pseudophrygia* ("ps") and *C. stenolepis* ("st") and the putative hybrid of the last two species abbreviated as "hybr" (for details, see Koutecký, 2015). The centaurea dataset is included with this package. Execute the command `data(centaurea)` to load this data to the R workspace.

```
data(centaurea)
```

In general, morphological data can be imported using the `read.morphodata()` function, providing a path to the data[3]. The argument `dec` stands for the character used in the file for the decimal separator, and `sep` is the column delimiter character, usually a blank space `""`, comma `","` or tab `"\t"`. The default values are a dot and a tab (`"\t"`), respectively, and these may be omitted from the function call. To read data from clipboard (select cells in the spreadsheet, press Ctrl+C), set `file = "clipboard"`.

```
centaurea = read.morphodata(file = "<PATH TO centaurea.txt>", dec = ".", sep = "\t")
centaurea = read.morphodata(file = "clipboard")
```

[1] If the **population level is missing or inapplicable** (e.g. more than one individual only in some populations and/or very low number of individuals per population), copy the values from the "ID" column to the "Population" column. This will allow analysing such data by most methods except analyses considering the population level.

[2] **The morphological characters** can be quantitative, binary (coded as 0/1), or multi-state ordered categorical (semiquantitative, rank-ordered) characters (e.g. 1 = small, 2 = medium, 3 = large, where change from state 1 to 3 is more costly than change from 1 to 2). By contrast, unordered categorical (qualitative, nominal) characters (e.g. describing colour: 1 = red, 2 = green, 3 = blue) are not applicable in most analyses (e.g. principal component or discriminant analyses). If there is a reason to include such unordered multistate characters, these characters either have to be coded as binary characters as follows: redFlowers (0/1), greenFlowers (0/1), blueFlowers (0/1), or, in some cases, coefficients for mixed data (e.g. the Gower coefficient) should be used.

[3] **Example dataset** in txt and xlsx formats are stored in the "extdata" directory of the MorphoTools2 package installation directory. To find the path to the package location run `system.file("extdata", package = "MorphoTools2")`.

The dataset now exists as a `morphodata` object in R. The `morphodata` object, like other objects used later, is defined as a `list`. In R, lists act as containers for data. Elements stored in the `morphodata` object can be referenced by the `$` notation. Type `centaurea$` and press the tab key to see the contained elements. The command `centaurea$Taxon` prints the values to the R console. Run `?morphodata` to see the structure of a `morphodata` object. Alternatively, the following commands display basic information about the dataset or show data in the data viewer.

```
summary(centaurea)
#> Object of class 'morphodata'
#> - contains 33 populations
#> - contains 4 taxa (defined groups)
#>
#> 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
#> Taxa (defined groups): 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"
#> [61] "BRT1773"  "BRT1774"  "BRT1775"  "BRT1778"  "BRT1779"  "BRT1780"
#> [67] "BRT1781"  "BRT1782"  "BRT1783"  "BRT1784"  "BRT1785"  "BRT1788"
#> [73] "BRT1792"  "BRT1794"  "BRT1795"  "BRT1797"  "BRT1798"  "BRT1799"
#>  [ reached getOption("max.print") -- omitted 574 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"
```

```
characters(centaurea)
#>  [1] "SN"  "SF"  "ST"  "SFT" "LL"  "LW"  "LLW" "LM"  "LBA" "LBS" "LS"  "IL"
#> [13] "IW"  "ILW" "CG"  "ML"  "MW"  "MLW" "MF"  "IS"  "IV"  "AL"  "AW"  "ALW"
#> [25] "AP"
```

```
viewMorphodata(centaurea)
```

3.2 Assessing normality of data

An assessment whether the data are approximately normally distributed is a prerequisite for many statistical tests, even though many analyses are quite robust to moderate deviations from normality. Out of the analyses used here, normality of distribution is required by Pearson's correlation coefficient and discriminant analysis (both canonical and linear or quadratic classificatory analysis). The normality of distribution of data is not an inevitable assumption of hierarchical clustering, principal component analysis, principal coordinates analysis or non-metric multidimensional scaling.
There are two main approaches to assessing normality: numerical and graphical. Please note that, although all methods available in the `MorphoTools2` package are presented here, there is no need to use all these methods at once.

3.2.1 Shapiro-Wilk test of normality

The normality of distribution of each character at the level of a taxon can be tested using the Shapiro-Wilk statistic. If the calculated p-value of a certain character is below a set threshold (0.05 is the default, but this can be changed using the `p.value` argument), we can reject the null hypothesis that characters are normally distributed. The default behaviour is to print only ***`normally distributed`*** or ***`NOT normally distributed`*** as the result, but setting the `p.value` to `NA` displays the exact p-values.

```
shapiroWilkTest(centaurea)
#>                          hybr                          ph                          ps
#> SN  NOT normally distributed NOT normally distributed     normally distributed
#> SF  NOT normally distributed NOT normally distributed NOT normally distributed
#> ST  NOT normally distributed NOT normally distributed NOT normally distributed
#> SFT NOT normally distributed NOT normally distributed NOT normally distributed
#> LL      normally distributed     normally distributed NOT normally distributed
#> LW      normally distributed     normally distributed     normally distributed
#> LLW     normally distributed NOT normally distributed NOT normally distributed
#>                           st
#> SN  NOT normally distributed
#> SF  NOT normally distributed
#> ST  NOT normally distributed
#> SFT NOT normally distributed
#> LL  NOT normally distributed
#> LW  NOT normally distributed
#> LLW NOT normally distributed
#>  [ reached 'max' / getOption("max.print") -- omitted 18 rows ]
```

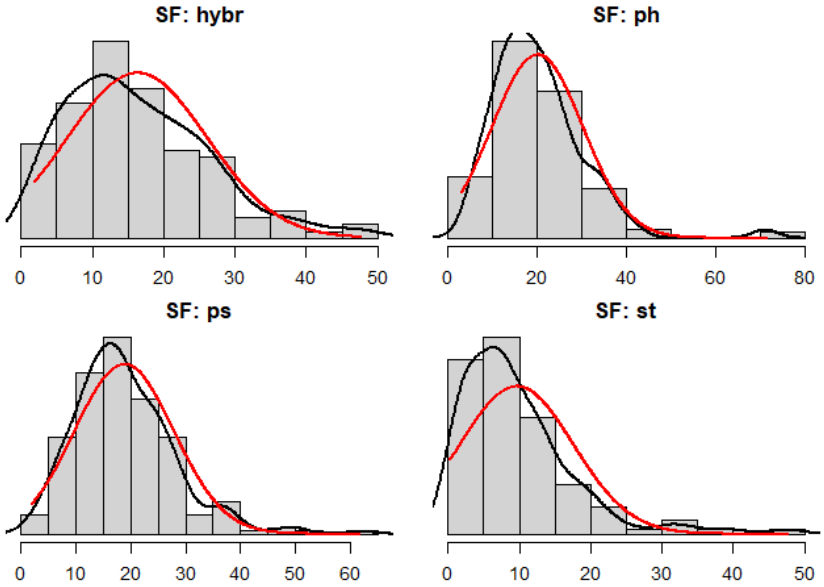
Because the results are rather extensive (depending on the number of groups and characters), they can be assigned to an object and exported to the clipboard or a file using the `exportRes()` function. This function is designed to export the results in a spreadsheet-like form. If needed, the default decimal separator (`dec`) and column delimiter character (`sep`) can be changed using the respective arguments; see the `exportRes()` function's documentation for details.

```
swTest = shapiroWilkTest(centaurea)
exportRes(swTest, file = "clipboard")
exportRes(swTest, file = "D:/Projects/Centaurea/morpho/shapiroWilkTest.txt")
```

3.2.2 Histograms

Histograms are a traditional way of displaying the shape of the distribution of data. The function `histCharacter()` displays the within-group distribution of values of a particular character for each taxon. The density curve smoothing of the histogram (black) and the normal distribution curve (red) are drawn as default but can be removed by setting the `densityLine` and `normDistLine` arguments to `FALSE`. Missing data are omitted.

```
histCharacter(centaurea, character = "SF")
```



To save histograms for all characters with default settings to a new folder (in the working directory), use the `histAll()` function.

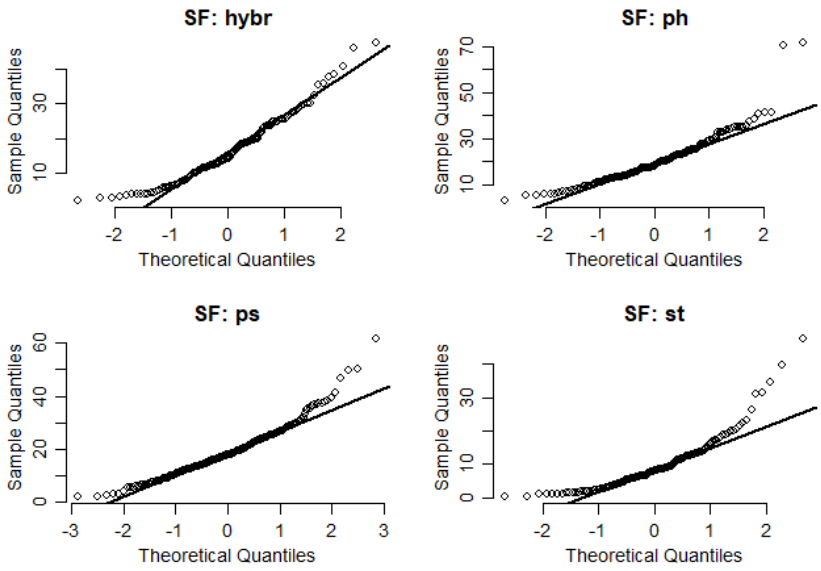
```
histAll(centaurea, folderName = "histograms")
```

3.2.3 Normal Q-Q plot

The normal Q-Q plot is another graphical method of assessing normality. The points should lie as close to the line as possible, with no obvious pattern of deviation from the line. Deviations from this line correspond to various types of non-normality.

The function `qqnormCharacter()` draws a Q-Q plot for each taxon and a particular character. The function `qqnormAll()` does the same for all characters (and save images to a new folder). Missing data are omitted.

```
qqnormCharacter(centaurea, character = "SF")
```




```
qqnormAll(centaurea, folderName = "qqnormPlots")
```

Most of the characters in the centaurea dataset do not have normal distribution. In general, there are two options: The distribution of data can be improved by transformation to make it more like normal, or non-parametric methods that do not require normality of distribution (Spearman's correlation coefficient instead of Pearson's and k nearest neighbours classificatory discriminant analysis instead of linear or quadratic DA) may be preferred.

The transformation of data is addressed in the following section. However, in all following analyses, the original data are used and non-parametric methods preferred.

3.3 Data transformation

The characters that deviate the most from the normal distribution can be transformed to improve their distribution (to make them normally distributed or at least to achieve lower deviation from normality). From the wide palette of applicable transformations (e.g. logarithmic, square root, cube root, arcsine), the one which improves the distribution of a particular character the most should be chosen. Note that, when using a log transformation, a constant should be added to all values to make them all positive before transformation if there are zero values in the data, because the argument of the logarithm can only take positive numbers. The arcsine transformation is often used for proportions and percentages (for values ranging from 0 to 1).

Transformation can be done using the `transformCharacter()` function, which, in addition to the data object, the name of the character to be transformed (`character`) and a new name for the transformed character (`newName`; not required), takes as argument an anonymous function (`FUN`), also known as a lambda expression. Without long explanation, this is where to place the function that will transform the data. Transformed values will replace original values of the character, under the old or new name, if the `newName` argument is set.

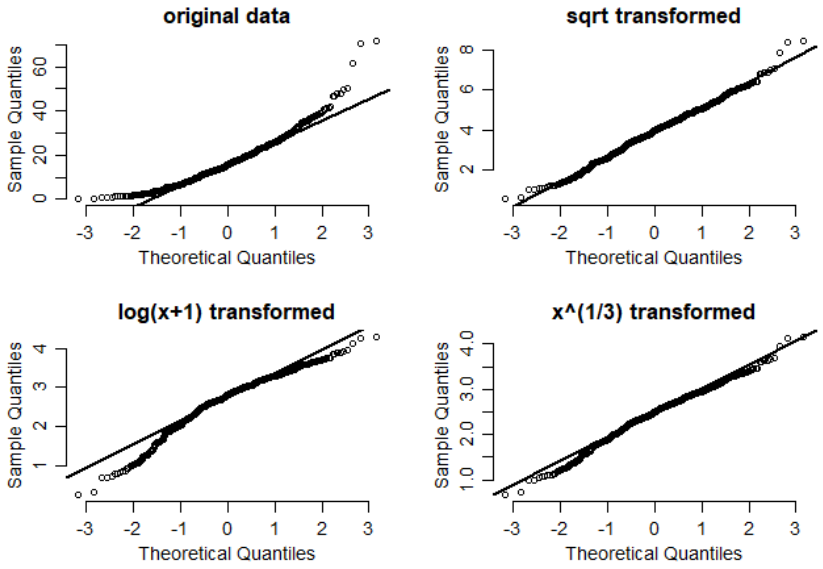
As the `transformCharacter()` takes another function as an argument (`FUN`), there is an inexhaustible amount of potential transformations:

For a right-skewed (positive) distribution, the following can be used:
* logarithmic transformation (natural): `FUN = function(x) log(100*x+1)`
* logarithmic transformation (common): `FUN = function(x) log10(100*x+1)`
* square root transformation: `FUN = function(x) sqrt(x)`
* cube root transformation: `FUN = function(x) x^(1/3)`
* arcsine transformation: `FUN = function(x) asin(sqrt(x))`

For a left-skewed (negative) distribution, the following can be used:
* logarithmic transformation (natural): `FUN = function(x) log((100*max(x)+1)-x)`
* logarithmic transformation (common): `FUN = function(x) log10((100*max(x)+1)-x)`
* square root transformation: `FUN = function(x) sqrt((max(x)+1)-x)`
* cube root transformation: `FUN = function(x) ((max(x)+1)-x)^(1/3)`
* arcsine transformation: `FUN = function(x) asin(sqrt((max(x))-x))`

As stated above, when applying a log transformation, a constant should be added to all values to make them all positive before transformation. However, log transformation (besides changing the shape of the distribution) also changes multiplication to sum (the values differ x-times vs differ by x). For small values of x, adding 1 significantly alters the original ratios, so when log-transforming small numbers, it is recommended to first multiply x by some constant (e.g. 100) and then add 1, as is shown in the examples above.

The following figure depicts the effects of different types of transformation on the same data.



So finally, to apply a square root transformation to character `SF`, the following code can be used.

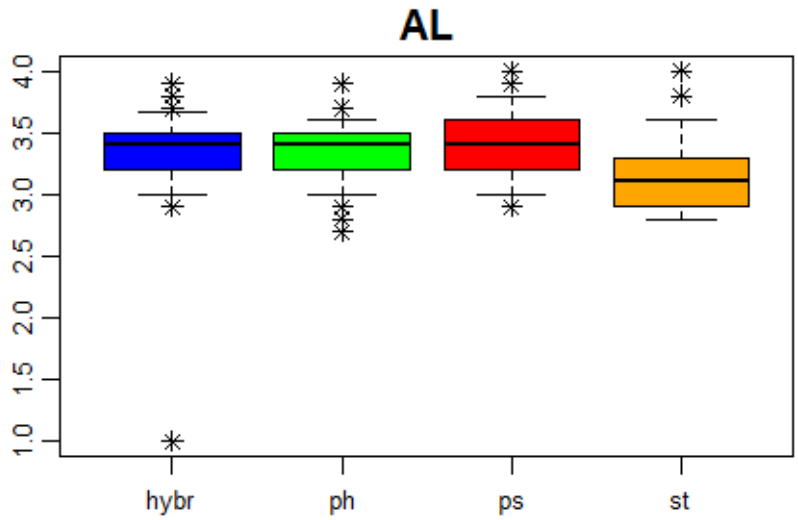
```
centaurea = transformCharacter(centaurea, character = "SF", newName = "SF.sqrt",
                               FUN = function(x) sqrt(x))
```

3.4 Box Plots

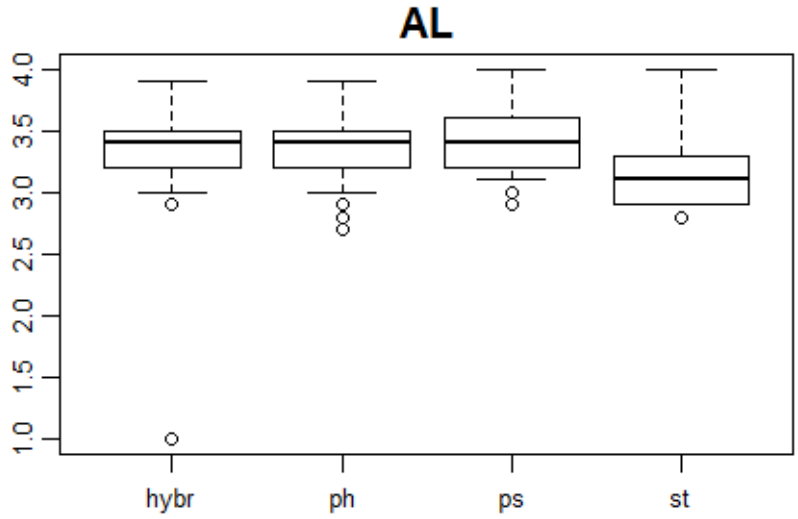
Boxplots are a handy tool for detecting outlier values (potential typos, missing decimal points, etc.), between-species dissimilarities and critical morphological values discriminating among species.

Boxplots can be produced for a particular character using the `boxplotCharacter()` function or for all characters at once by invoking the `boxplotAll()` function, which saves all boxplots to a new folder in the working directory or other location. A box is drawn from the first to the third quartile (25th-75th percentiles), a horizontal line drawn inside denotes the median (50th percentile). The whiskers can be extended to the desired percentiles using the arguments `lowerWhisker` and `upperWhisker`. Missing data are omitted. Many graphic parameters can be set; run `?boxplotCharacter` for details.

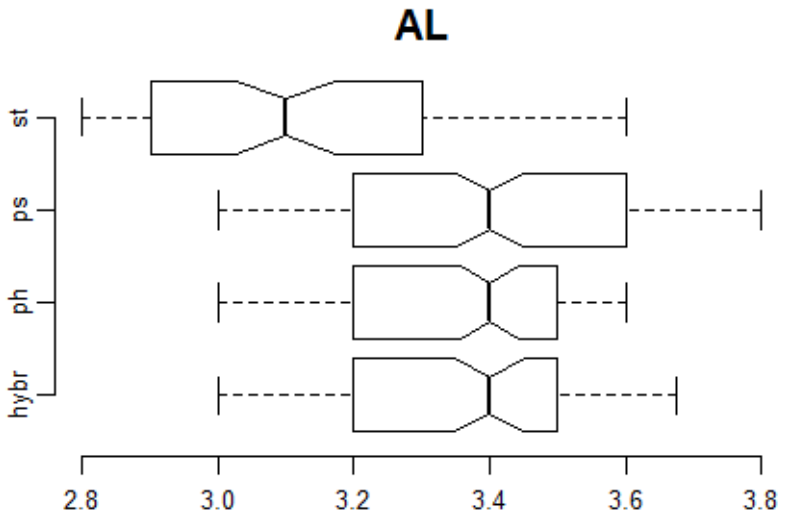
```
boxplotCharacter(centaurea, character="AL", col=c("blue","green","red","orange"))
```




```
boxplotCharacter(centaurea, character = "AL", pch = 1,
                 lowerWhisker = 0.1, upperWhisker = 1)
```



```
boxplotCharacter(centaurea, character = "AL", outliers = FALSE,
                 frame = FALSE, horizontal = T, notch = TRUE)
```



The default behaviour is to plot outliers (as asterisks be default, but this can be changed in the `pch` argument; `outliers = FALSE` will suppress the plotting of outliers) and to show the trimmed range (omitting 10% of the most extreme values) using whiskers.

Boxplots for all characters with the default settings can be saved to a new folder (in the working directory) using the following command:

```
boxplotAll(centaurea, folderName = "boxplots")
```

3.5 Descriptive statistics

The table of descriptive statistics is a less comfortable way of detecting outlier values. However, it can be used for reporting descriptive statistics for morphological characters. These statistics can be calculated at the levels of populations, taxa/groups or for the whole dataset, using the functions `descrPopulation()`, `descrTaxon()` or `descrAll()`, respectively.

Using the argument `format`, the desired output format can be specified. The keywords `$MEAN`, `$SD`, `$MIN`, `$5%`, `$25%`, `$MEDIAN`, `$75%`, `$95%` and `$MAX` are then replaced by actual values. The default behaviour (`format = NULL`) is to produce a table with all values. Run `?descrTaxon` for more details.

```
descrTaxon(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)
#>  [ reached 'max' / getOption("max.print") -- omitted 20 rows ]
```

The results can be assigned to an object and copied to the clipboard (this fails with extensive datasets) or exported to file, both using the `exportRes()` function.

```
descrTax = descrTaxon(centaurea, format = "($MEAN ± $SD)", decimalPlaces = 2)

exportRes(descrTax, file = "clipboard")
exportRes(descrTax, file = "descrTax.txt")
```

Please note that some of the following statistical analyses require that no character is invariant in any taxon or group. If it is, a more common practice is to add a small constant (e.g. 0.000001) to some value instead of removing the whole character.

3.6 Correlations of characters

Highly correlated characters ($r > |0.95|$) should not be used in discriminant analysis, as this can distort the results. The function `cormat()` calculates the correlation coefficients of the characters, be it Pearson's (default) or Spearman's (does not require normally distributed data). The results can be exported with the `exportRes()` function. One of the pair of highly correlated characters can be removed from the dataset using the `deleteCharacter()` function, see below.

```
correlations.s = cormat(centaurea, method = "spearman")
exportRes(correlations.s, file = "correlations.spearman.txt")
```

Significance tests are usually unnecessary for morphometric analysis. Anyway, if tests are needed, they can be performed using the `cormatSignifTest()` function.

```
correlations.s.signifTest = cormatSignifTest(centaurea, method = "spearman")
```

3.7 Populations as operational taxonomic units

To simplify the overall structure, especially with large datasets, using populations instead of individuals can be considered. This means that each population will be represented by averages of the individuals' values. Missing values will be ignored.

```
pops = populOTU(centaurea)
#> Warning: Unable to calculate the means of characters AL AW ALW AP in
#> populations LIP PREL. Values are NA.
```

There is a warning that the values of some characters are NA. How to deal with missing data is discussed in the following section.

3.8 Missing data

Missing values are not accepted in the majority of morphological analyses. The decision what to do with missing values is on the user. There are two options: remove or replace. However, before doing anything else, let us have a look at the descriptive statistic about missing data, using the `missingCharactersTable()` and `missingSamplesTable()` functions. The amount of missing data can be summarized at various levels, namely `"taxon"`, populations (`"pop"`), or individuals (`"indiv"`).

```
# For demonstration only. Not all populations are displayed.
missingCharactersTable(centaurea, level = "pop")
#>    Population Taxon  N missing.percentage missing.values
#> 1        BABU  hybr 20               0.03             16
#> 2         BOL    ps 20               0.12             60
#> 3         BUK    ph 20               0.00              0
#> 4        CZLE    ps 20               0.08             40
#> 5         DEB  hybr 20               0.00              0
#> 6         DOM    st 12               0.07             20
#> 7         DUB    st 20               0.08             40
#> 8         KOT    ph 20               0.00              0
#> 9        KOZH    ps 20               0.15             76
#> 10        LES    st 20               0.00              0
#> 11        LIP    st 20               0.16             80
#> 12        NEJ    ph 20               0.01              4
#> 13       PREL    st 20               0.16             80
#> 14        RTE  hybr 20               0.09             47
#> 15        RUS  hybr 20               0.00              0
#> 16       STCV    ph 20               0.11             56
#> 17        VIT  hybr 20               0.05             24
#> 18        VOL    st 20               0.02              8
```

```
# For demonstration purposes only. Only a subset of data is displayed.
missingSamplesTable(centaurea, level = "pop")
#>    Population  N SN LS CG ML MF IS IV   AP missing.percentage missing.values
#> 1        BABU 20  0  0  0  0  0  0  0 0.20               0.03              4
#> 2         BOL 20  0  0  0  0  0  0  0 0.75               0.09             15
#> 3         BUK 20  0  0  0  0  0  0  0 0.00               0.00              0
#> 4        CZLE 20  0  0  0  0  0  0  0 0.50               0.06             10
#> 5         DEB 20  0  0  0  0  0  0  0 0.00               0.00              0
#> 6         DOM 12  0  0  0  0  0  0  0 0.42               0.05              5
#> 7         DUB 20  0  0  0  0  0  0  0 0.50               0.06             10
#> 8         KOT 20  0  0  0  0  0  0  0 0.00               0.00              0
#> 9        KOZH 20  0  0  0  0  0  0  0 0.95               0.12             19
#> 10        LES 20  0  0  0  0  0  0  0 0.00               0.00              0
#> 11        LIP 20  0  0  0  0  0  0  0 1.00               0.12             20
#> 12        NEJ 20  0  0  0  0  0  0  0 0.05               0.01              1
#> 13       PREL 20  0  0  0  0  0  0  0 1.00               0.12             20
#> 14        RTE 20  0  0  0  0  0  0  0 0.60               0.07             12
#> 15        RUS 20  0  0  0  0  0  0  0 0.00               0.00              0
#> 16       STCV 20  0  0  0  0  0  0  0 0.70               0.09             14
#> 17        VIT 20  0  0  0  0  0  0  0 0.30               0.04              6
#> 18        VOL 20  0  0  0  0  0  0  0 0.10               0.01              2
```

As indicated by the warnings above, populations LIP and PREL have the highest percentages of missing values in morphological characters (16%; 80 values per population are missing). The latter table shows that the characters AL, AW, ALW and AP are completely missing in these populations and 95% samples of population KOZH lack values of these characters.

3.8.1 Removing items

The descriptive tables above show that four characters in two populations are completely missing. The user should decide between deleting the characters, using the `deleteCharacter()` function, or the populations, using the `deletePopulation()` function. As the character AP looks promising for the delimitation of *C. pseudophrygia* and *C. stenolepis*, characters will be retained and the populations removed.

```
centaurea = deletePopulation(centaurea, populationName = c("LIP", "PREL"))
pops = deletePopulation(pops, populationName = c("LIP", "PREL"))
```

Another available option is to delete samples with a high portion of missing data using the `deleteSample()` function. The command `deleteSample(centaurea, missingPercentage = 0.1)` returns a new `morphodata` object (dataset), retaining only samples having no more than 10% of missing data. To remove specific samples, enumerate them in the 'sampleName' argument in these functions.

Here is the right place to mention also `deleteTaxon()` and another four functions with reversed logic, which will return only mentioned samples, populations, taxa or characters: `keepSample()`, `keepPopulation()`, `keepTaxon()` and `keepCharacter()`.

3.8.2 Replacing missing values

Missing values can be substituted by the average value of the respective character in the respective population. However, substitution by the mean introduces values that are not present in the original dataset. This approach is acceptable only if the following conditions are met: There are relatively few missing values; these missing values are scattered across characters (each character including only a few missing values); and removing all individuals or all characters with missing data would unacceptably reduce the dataset. To substitute remaining missing values by an average value, use the function `naMeanSubst()`.

```
centaurea = naMeanSubst(centaurea)
```

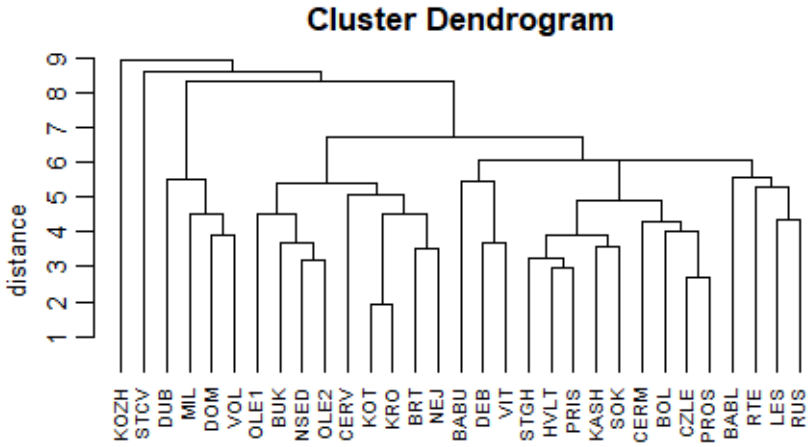
After examining the normality of distribution of each character, confirming that the data do not contain highly correlated characters, replacing missing values by average values and removing remaining NAs, the centaurea dataset is prepared for further analyses. It is not a bad idea to save a copy of it using the `exportRes()` function.

4. Hierarchical clustering

Hierarchical classification is one of the methods that do not require *a priori* specification of the membership of samples in taxa (groups). Therefore, this method is recommended to be used first in order to gain insight into the existence of a (hierarchical) group structure in the data. Both individuals and populations can be used, but with large datasets (of hundreds of specimens or more) dendrograms for individuals may be somewhat messy and populations are a better choice. Various measures of distance between observations (rows) are applicable: (1) coefficients of distance for quantitative and binary characters: Euclidean (default), Manhattan, Minkovski; (2) similarity coefficients for binary characters: Jaccard and simple matching; and (3) coefficient for mixed data: Gower. The clustering methods available with this package, using the above coefficients are: UPGMA (default), Ward's method, single linkage, complete linkage, WPGMA, WPGMC and UPGMC. However, note that for morphometric analysis, Euclidean distance and UPGMA or Ward's method are the most commonly used. The function includes the standardization of characters to a zero mean and a unit standard deviation. For further details, run `?clust`.

The dendrogram is displayed using the `plot()` function with usual graphical parameters. The parameter `hang` controls the distance of the labels from the plot; negative values cause labels to be aligned at zero.

```
pops_hierClust = clust(pops, distMethod = "Euclidean", clustMethod = "UPGMA")
plot(pops_hierClust, hang = -1, sub = "", xlab = "", ylab = "distance")
```



Several main clusters were formed in the dendrogram above; however, some populations (BABL, LES, OLE1, OLE2 and PROS) were clustered "incorrectly", requiring further inspection.

5. Principal component analysis (PCA)

Principal component analysis (PCA) is another method without the requirement for the *a priori* specification of the samples' membership in taxa (groups). PCA transforms the measured variables into principal components (artificial variables). The first few of them extract most of the variance in the measured variables. Standardized PCA based on a correlation matrix is calculated by the `pca.calc()` function (based on the package `stats`; R Core Team, 2020); the result is an object of the class `pcadata`. Run `?pcadata` for the help page about the elements of this object. Note the limitation of PCA with regard to the number of analysed characters. It should be lower than the number of objects analysed.

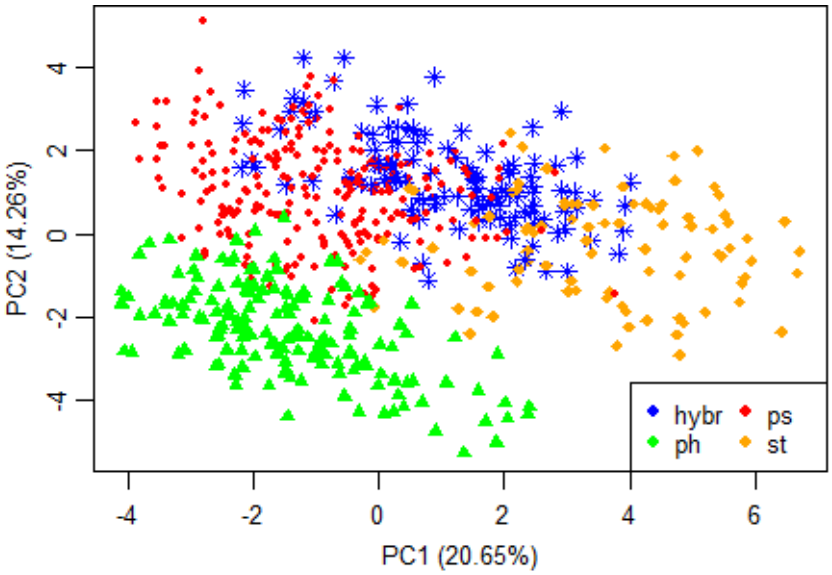
```
pca.centaurea = pca.calc(centaurea)
```

The summary statistics of the data are available through the function `summary()`. Eigenvalues indicate the proportion of variation of the original dataset expressed by individual axes. They are usually presented as a percentage of their total sum (eigenvalues as percentages). Eigenvectors express the direction of vectors characterizing the influence of the original characters on the principal component axes. The output of the `summary()` function is usually truncated. To get a full listing, execute the following commands: `pca.centaurea$eigenvalues`, `pca.centaurea$eigenvaluesAsPercentages`, `pca.centaurea$cumulativePercentageOfEigenvalues` and `pca.centaurea$eigenvectors` (values will be printed on the console).

```
summary(pca.centaurea)
#> Object of class 'pcadata'; storing results of principal component analysis
#>
#> Variation explained by individual axes (listing of axes is truncated):
#>                                        PC1    PC2    PC3    PC4
#> Eigenvalues                         5.1628 3.5660 2.6634 1.7359
#> Eigenvalues as percentages          0.2065 0.1426 0.1065 0.0694
#> Cumulative percentage of eigenvalues 0.2065 0.3492 0.4557 0.5251
#>
#> Eigenvectors (listing of axes is truncated):
#>              PC1         PC2          PC3          PC4
#> SN    0.10771859  0.09373031  0.533139126  0.173265222
#> SF   -0.26900856  0.08099386  0.059673923 -0.462177220
#> ST   -0.02956541  0.12948462  0.538957839 -0.062077155
#> SFT  -0.27145559  0.02321041 -0.244805653 -0.411793361
#> LL   -0.07372490  0.19206329  0.383723626 -0.232003396
#> LW   -0.20896559  0.21922418  0.152923297 -0.218305082
#> LLW   0.20543544 -0.11181134  0.188443414  0.037332596
#> LM   -0.07506076 -0.13142341  0.065759045 -0.102072769
#> LBA   0.27729432 -0.03899012 -0.004432443 -0.089885703
#> LBS  -0.16210200  0.23562860 -0.130667743  0.042490929
#> LS    0.27249981 -0.04295413  0.111132394  0.048352730
#> IL   -0.19760433  0.22449890 -0.018527821  0.225092141
#> IW   -0.36011332  0.06964662  0.053343011  0.247319193
#> ILW   0.34334276  0.04623376 -0.072130258 -0.164859081
#> CG   -0.06830949  0.13736296  0.084854346 -0.328948291
#> ML    0.05725511  0.43511355 -0.160955641  0.110889828
#>  [ reached getOption("max.print") -- omitted 9 rows ]
```

The result can be plotted using the `plotPoints()` function. The parameter `axes` define which principal components to plot (the 1st and 2nd being default), and the parameters `col` and `pch`[4] control the colour and type of plotting character, respectively (the same for each point or specific for each taxon). The usual graphical parameters affect the axes, data point symbol size, etc., and several parameters define the appearance and position of the legend; see the documentation for `plotPoints()` for details.

```
plotPoints(pca.centaurea, col = c("blue","green","red","orange"), axes=c(1,2),
           pch = c(8,17,20,18), legend = T, ncol = 2, legend.pos="bottomright")
```

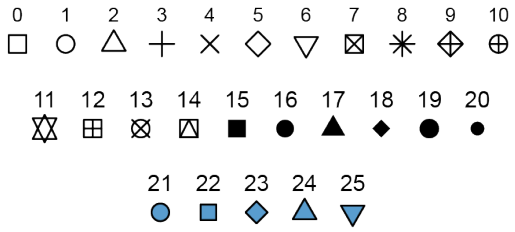


The coordinates of the individuals (populations) in the principal component space (sample scores) are stored in `pca.centaurea$objects$scores` and can be exported using the `exportRes()` function. The dollar sign (`$`) enables one to extract items from an object, as above.

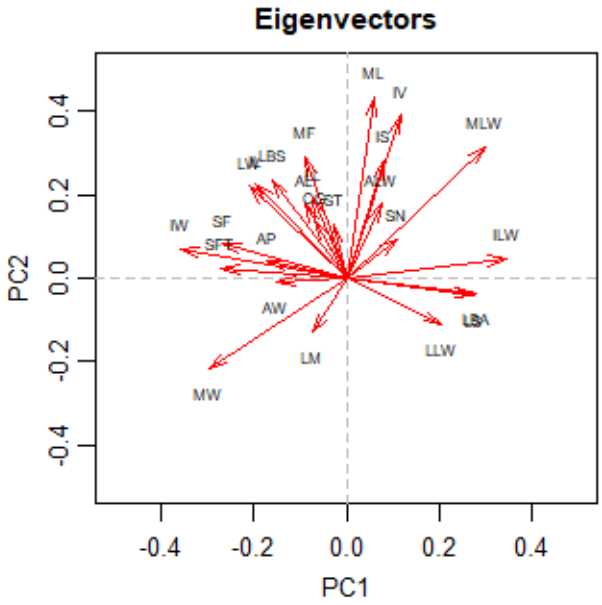
```
exportRes(pca.centaurea$objects$scores, file="scoresPCA.centaurea.txt")
```

The character loadings (eigenvectors) express the influence of the original characters on the main components. Eigenvectors are stored in `pca.centaurea$eigenvectors` and can be exported using the `exportRes()` function. The function `plotCharacters()` draws character loadings as arrows.

[4] **Plotting symbols commonly used in R**



```
plotCharacters(pca.centaurea)
```




```
exportRes(pca.centaurea$eigenvectors, file="eigenvectors.centaurea.txt")
```

Ordination diagrams of PCA resulted in relatively compact groupings corresponding to taxa with partial overlaps. The first two components (axes) extracted 20.65% and 14.26% of the overall variability in the data. The characters ILW and IW are strongly correlated with the direction of the separation of the taxa *C. pseudophrygia* and *C. stenolepis* ("ps", "st") and their putative hybrid "hybr". *Centaurea phrygia* s.str. ("ph") is separated in the diagonal direction, being highly correlated with the characters MW, ML, IV and MLW.

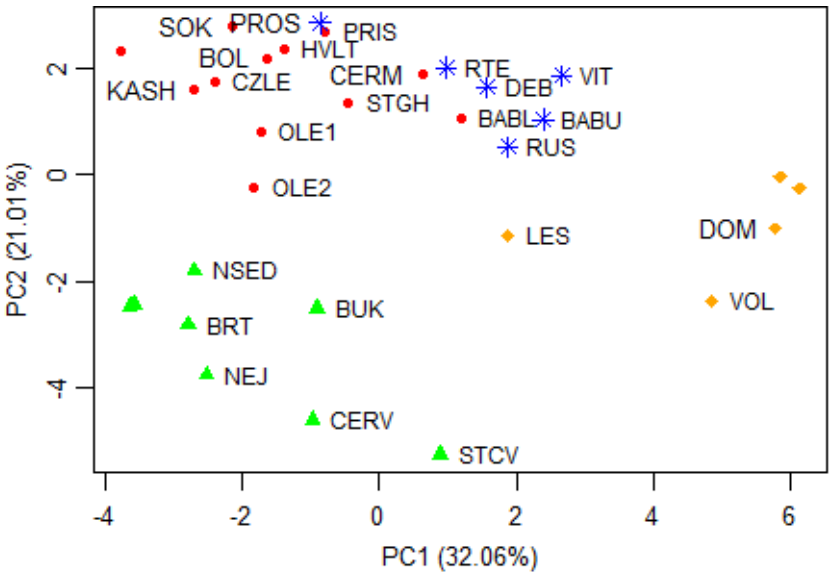
The `plotPoints()` and `plotCharacters()` are default plotting functions. Simple data point labels and a legend can be added using the arguments `labels = TRUE` and `legend = TRUE`, respectively.

For more precise control of labels and the legend, or adding elements to the plot, the following functions can be used:

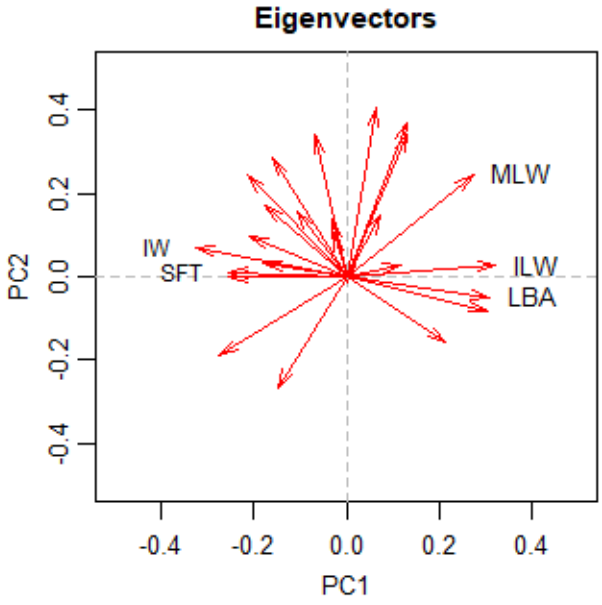
- `plotAddLabels.points()`, `plotAddLabels.characters()` allows to include or exclude specified labels (`include`), specify the label's position (`pos`), offset (`offset`), colours (`col`), magnification (`cex`), etc.
- `plotAddLegend()` allows to specify the position using a keyword (`x`[5]), the number of columns (`ncol`), expansion and interspacing factors (`cex`, `pt.cex`, `x.intersp`, `y.intersp`), line width (`lwd`), borders parameters (e.g., `box.type`, `box.lty`, `box.lwd`), etc.
- `plotAddEllipses()` draws prediction ellipses around taxa. Ellipses with a given probability (`probability`) define regions where any new independent observation belonging to the respective taxa will fall.
- `plotAddSpiders()` connects points with their group centroid, thus forming a "spider" diagram.

[5] **legend position**: `"topleft"`, `"topright"`, `"bottomleft"`, `"bottomright"`, `"top"`, `"left"`, `"bottom"`, `"right"` and `"center"`.

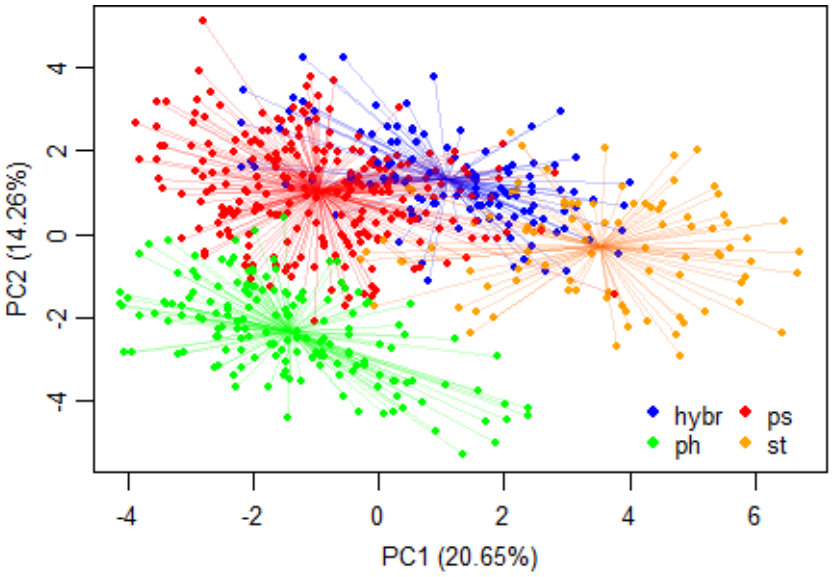
```
pca.pops = pca.calc(pops)
plotPoints(pca.pops, col = c("blue","green","red","orange"), pch=c(8,17,20,18),
           legend = FALSE, labels = FALSE)
plotAddLabels.points(pca.pops, labels=c("PROS","SOK","KASH","BOL","KRO","DUB",
         "MIL","CERM","DOM","KOZH","KOT"), include=FALSE, pos=4, cex=0.7)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




```
plotCharacters(pca.pops, labels = FALSE)
plotAddLabels.characters(pca.pops,labels=c("ILW","MLW","LBA"),pos=4,cex=0.75)
plotAddLabels.characters(pca.pops,labels=c("IW","SFT"),pos=2,offset=0.7)
```

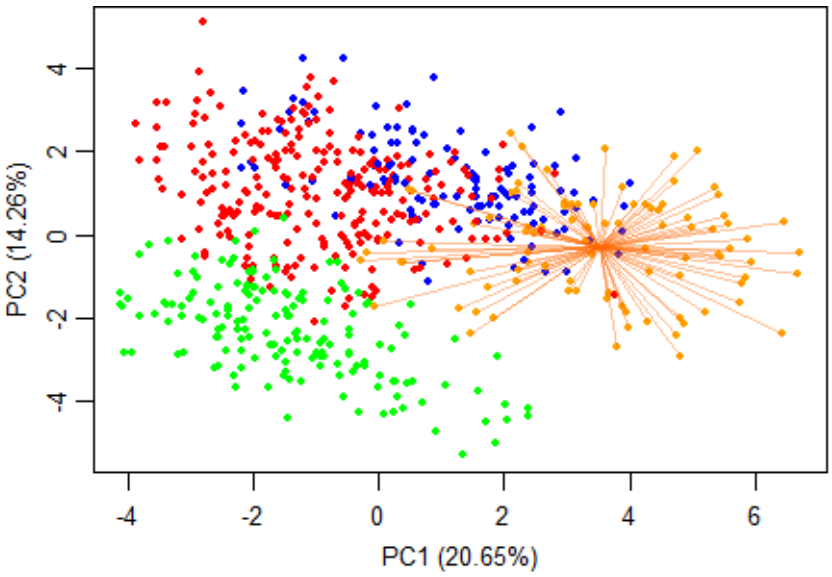


```
plotPoints(pca.centaurea, col = c("blue","green","red","orange"), cex = 0.5)
plotAddLegend(pca.centaurea, col = c("blue","green","red","orange"),
              x = "bottomright", cex = 0.8, box.type = "n", ncol = 2)

# Semi-transparent spiders
plotAddSpiders(pca.centaurea, col=c(rgb(0,0,255, max=255, alpha=50), # blue
                                    rgb(0,255,0, max=255, alpha=50), # green
                                    rgb(255,0,0, max=255, alpha=50), # red
                                    # orange
                                    rgb(255,102,0, max=255, alpha=50)))
```

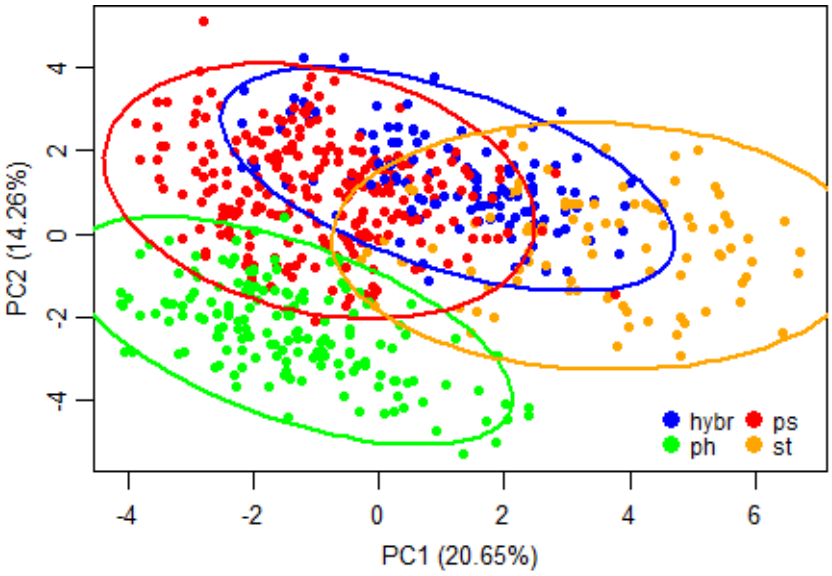


To highlight only some groups in colour, set the colour values of groups not to be highlighted to `NA`.

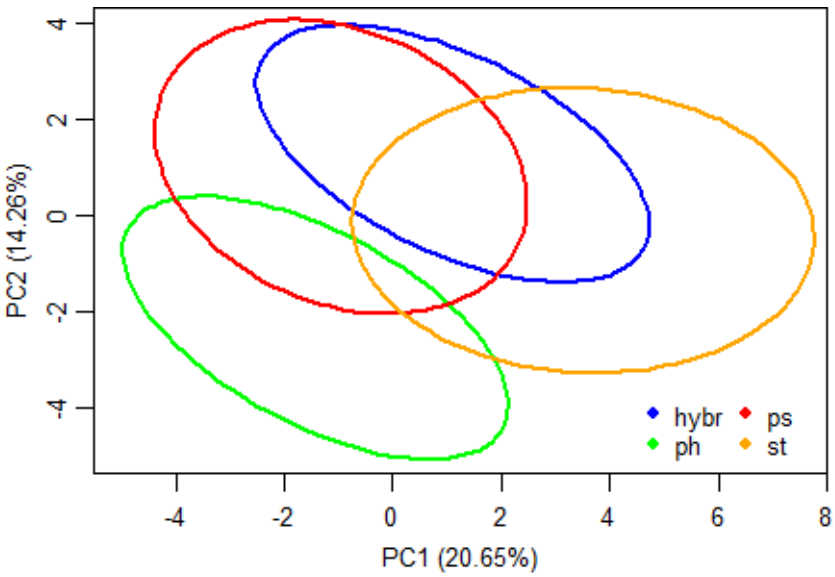
```
plotPoints(pca.centaurea, col = c("blue","green","red","orange"), cex = 0.5)

plotAddSpiders(pca.centaurea, col=c(NA,NA,NA,rgb(255,102,0,max=255,alpha=100)))
```



```
plotPoints(pca.centaurea, col = c("blue","green","red","orange"), cex = 0.7)
plotAddLegend(pca.centaurea, col = c("blue","green","red","orange"),
              x = "bottomright", pt.cex = 1.3, box.type = "n", ncol = 2)

plotAddEllipses(pca.centaurea, col = c("blue","green","red","orange"), lwd = 2)
```

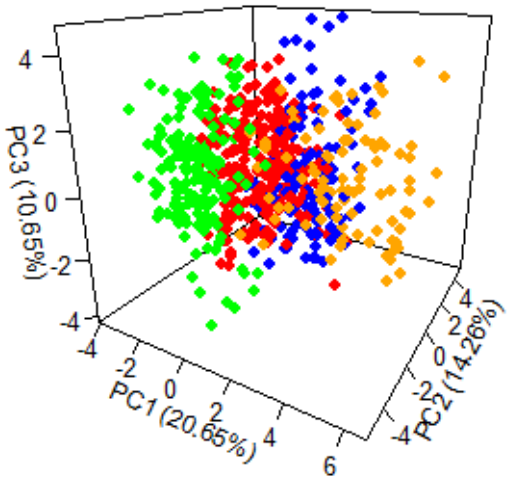



```
plotPoints(pca.centaurea, type = "n", xlim = c(-5,7.5), ylim = c(-5,4))

plotAddEllipses(pca.centaurea, col = c("blue","green","red","orange"), lwd = 2)

plotAddLegend(pca.centaurea, col = c("blue","green","red","orange"),
                       x = "bottomright", box.type = "n", ncol = 2)
```

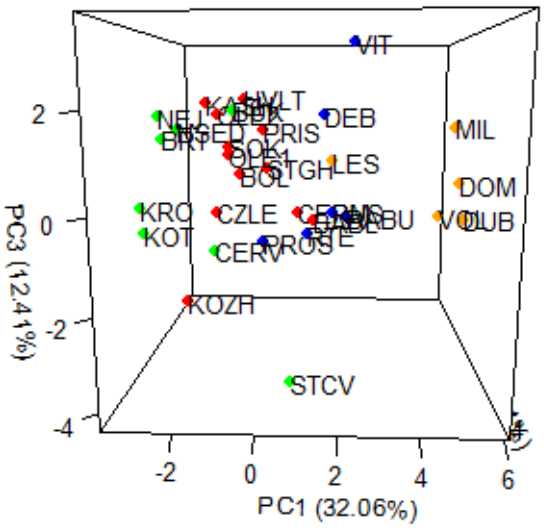


To draw a 3D scatterplot, the `plot3Dpoints()` function can be used. The `theta` and `phi` arguments define the viewing direction (azimuthal direction and co-latitude, respectively).

```
plot3Dpoints(pca.centaurea, col = c("blue","green","red","orange"),
             phi = 20, theta = 30)
```




```
plot3Dpoints(pca.pops, col = c("blue","green","red","orange"), labels = T)
```



6. Principal coordinate analysis (PCoA)

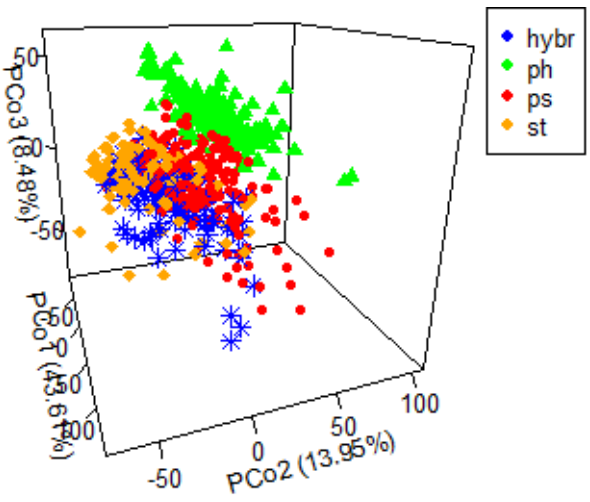
Principal coordinate analysis (PCoA) is another method for exploring and visualizing similarities or dissimilarities within the data. This method is especially useful in analyses of non-quantitative characters, when Euclidean distance and the corresponding measures of correlation do not provide acceptable model, so PCA is not adequate for ordination. PCoA estimates coordinates for a set of objects (rows) in a space, whose relationships are measured by any coefficient of similarity or distance (Euclidean, Manhattan, Minkovski, Jaccard, simple matching, or Gower). PCoA might be used also when there are more characters than objects in the analysis. As PCoA is computed from distances among objects, so there is no direct information on the influence of the original characters on the coordinate axes.

PCoA is performed by the `pcoa.calc()` function (based on the `stats` package; R Core Team, 2020); the result is an object of the class `pcoadata`.

```
pcoa.res = pcoa.calc(centaurea, distMethod = "Manhattan")
summary(pcoa.res)
#> Object of class 'pcoadata'; storing results of principal coordinates analysis
#> Resemblance coefficient:  Manhattan
#>
#> Variation explained by individual axes (listing of axes is truncated):
#>                                          PCo1        PCo2        PCo3
#> Eigenvalues                       1357127.8750 434014.9211 263932.2874
#> Eigenvalues as percentages              0.4361      0.1395      0.0848
#> Cumulative percentage of eigenvalues    0.4361      0.5755      0.6604
```

The result can be plotted by either the `plotPoints()` or `plot3Dpoints()` function as per usual. The extending functions `plotAddEllipses()`, `plotAddSpiders()`, `plotAddLabels.points()` and `plotAddLegend()` are available, too. Only the function `plotCharacters()` cannot be used, as there is no information on the influence of the original characters on the coordinate axes.

```
plot3Dpoints(pcoa.res, col = c("blue","green","red","orange"), pch = c(8,17,20,18),
             phi = 20, theta = 70, legend = T)
```



7. Non-metric multidimensional scaling (NMDS)

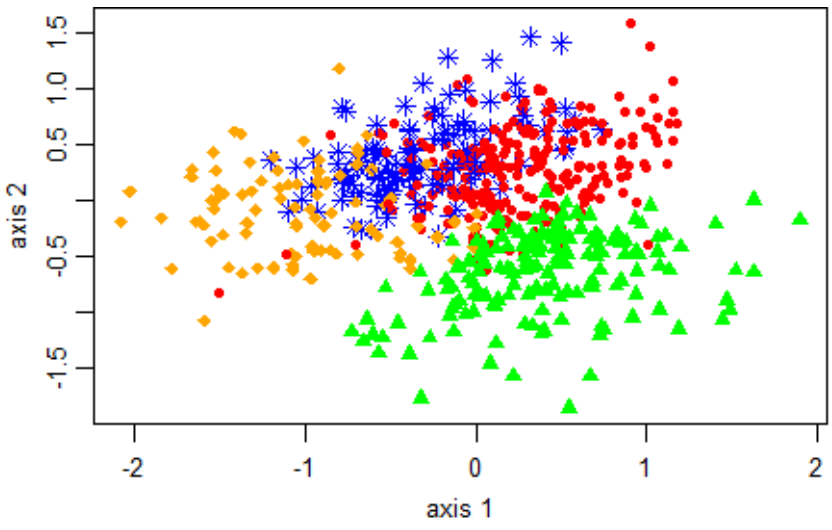
All of the clustering and ordination analyses mentioned above attempt to preserve the distance relationships among the objects as much as possible. The main difference with non-metric multidimensional scaling (NMDS) is that the preservation of distances is not of primary importance. This analysis attempts to represent the objects in a small number of dimensions (usually two or three) specified by the argument `k`, preserving the order of distances among objects (similar objects are plotted closer to one another and dissimilar objects far apart). Like principal coordinate analysis (PCoA), NMDS is not limited to Euclidean distance; it can produce ordinations using any coefficient of similarity or distance.

Because NMDS compresses the relationships among objects into two or three dimensions, this compression creates "stress". This stress value can be interpreted as the "goodness" of the solution, lower values being better. Since stress decreases as dimensionality increases, the optimal solution is when the decrease in stress is small after decreasing the number of dimensions. Further, multiple runs of the NMDS analysis are needed to ensure that a stable ordination has been reached, as any single run may get "trapped" in local optima which are not representative of true similarities. Similarly to PCoA, the influence of the original characters on new axes cannot be derived directly. Moreover, the ordination axes are arbitrary, so the variation explained by individual axes is unknown.

The NMDS is calculated by the `nmds.calc()` function, using the `monoMDS()` function from the package `vegan` (R Core Team, 2020); the result is an object of class `pcoadata`. The result can be plotted with the functions `plotPoints()` or `plot3Dpoints()`; the functions `plotAddEllipses()`, `plotAddSpiders()`, `plotAddLabels.points()` and `plotAddLegend()` work as usual. Only the `plotCharacters()` function is not applicable, as there is no information on the influence of the original characters on the coordinate axes.

```
nmds.res = nmds.calc(centaurea, distMethod = "Euclidean", k = 3)
summary(nmds.res)
#> Object of class 'nmdsdata'; storing results of non-metric multidimensional scaling
#> Resemblance coefficient:  Euclidean
#>
#> Dimensions:  3
#> Stress:  0.1590489
#> Scores scaled to unit root mean square, rotated to principal components
```

```
plotPoints(nmds.res, col = c("blue","green","red","orange"), pch=c(8,17,20,18))
```



8. Stepwise discriminant analysis

In some analyses, the number of characters must not exceed the number of samples. If it does, in many cases only a subset of the "best" characters contributing the most to the differentiation of taxa (predefined groups) have to be selected. Another common requirement is the linear independence of characters. No character should not be a linear combination of any other character(s). The way to eliminate such unnecessary or redundant characters is to use stepwise discriminant analysis.

The most useful characters are identified and added to the selection step by step. After adding a new character, the significance of all the characters in the model is tested, and those whose unique contribution is no more significant (bellow the `FToStay` threshold) are excluded before the addition of the next most useful character. When none of the unselected variables meets the entry criterion (the significance of their unique contribution is bellow the `FToEnter` threshold) or the maximum number of characters (depending on the number of individuals and defined groups) is reached, the selection process stops. After the final step, the selected characters are printed to the console, ordered according their importance for the separation of the predefined groups.

Stepwise discriminant analysis is calculated by the `stepdisc.calc()` function.

```
stepdisc.calc(centaurea)
#>    Entered Removed Partial R-square    F-value        Pr > F
#> 1      MLW              0.667335064 406.554939 7.890234e-145
#> 2       ML              0.506437108 207.611046  1.192547e-92
#> 3       IW              0.200252478  50.579714  3.483830e-29
#> 4       LS              0.157685396  37.752982  2.249101e-22
#> 5       IV              0.146472637  34.550532  1.281334e-20
#> 6       MW              0.115246110  26.181821  6.249414e-16
#> 7       MF              0.091473201  20.203721  1.736040e-12
#> 8       AP              0.070455039  15.184304  1.550026e-09
#> 9       IS              0.056738406  12.030259  1.174018e-07
#> 10     LBA              0.056209421  11.891566  1.422384e-07
#> 11      LW              0.054420588  11.472159  2.538006e-07
#> 12      AL              0.047197814   9.857623  2.364358e-06
#> 13     ILW              0.038600447   7.976519  3.205143e-05
#> 14     LBS              0.033553832   6.885891  1.454076e-04
#> 15     SFT              0.026130580   5.312678  1.281299e-03
#> 16      CG              0.028448848   5.788052  6.647989e-04
#> 17      IL              0.020554879   4.141286  6.406217e-03
#> 18      LM              0.012787149   2.551697  5.474060e-02
#> 19     ALW              0.011851686   2.358787  7.062704e-02
#> 20      AW              0.013462436   2.679193  4.621888e-02
#> 21              AL      0.005503766   1.086553  3.541773e-01
#> 22      SF              0.011227184   2.229299  8.371473e-02
#>
#> Selected characters:
#> MLW, ML, IW, LS, IV, MW, MF, AP, IS, LBA, LW, AL, ILW, LBS, SFT, CG, IL, LM, ALW, AW, SF
```

9. Canonical discriminant analysis (CDA)

The canonical discriminant analysis finds linear combinations of the original variables that provide maximum separation among *a priori* defined groups (by the `Taxon` column in the input data). Group membership should be defined using some independent, non-morphological data (e.g. on ploidy levels, geographic origin or genetic groups) to avoid circular reasoning.

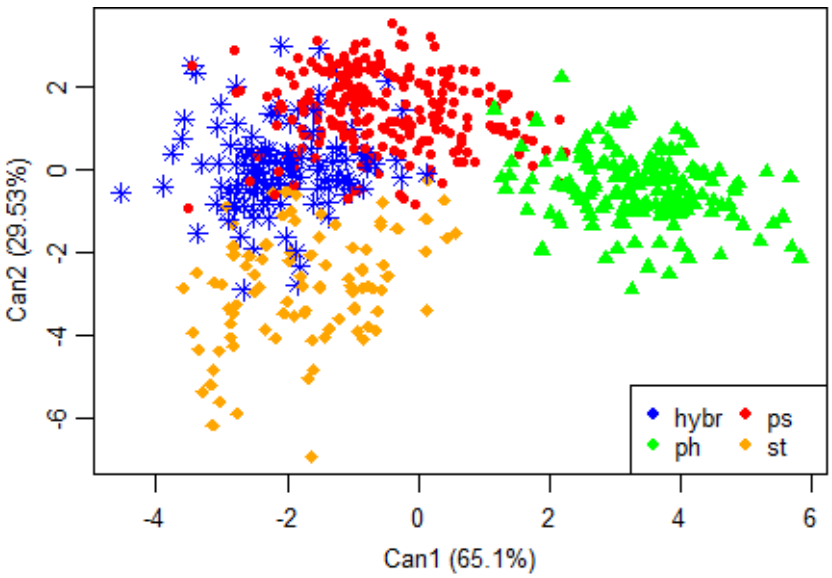
Discriminant analyses (in general) have requirements concerning the number, correlation and variability of characters: (1) No character may be a linear combination of any other character; (2) No pair of characters may be highly correlated; (3) No character may be invariant in any taxon (group); (4) For the number of taxa (g), characters (p) and the total number of samples (n), the following should hold: $0 < p < (n - g)$; and (5) There must be at least two groups (taxa) and in each group there must be at least two objects.

The CDA analysis is used to ascertain the extent to which the predefined groups of objects can be distinguished based on available characters and to identify characters which contribute the most to this differentiation. Note that canonical discriminant analysis finds n-1 meaningful canonical axes, where n is the number of groups (taxa). If two groups are analysed, only a single axis is computed and the sample scores are displayed as a histogram.

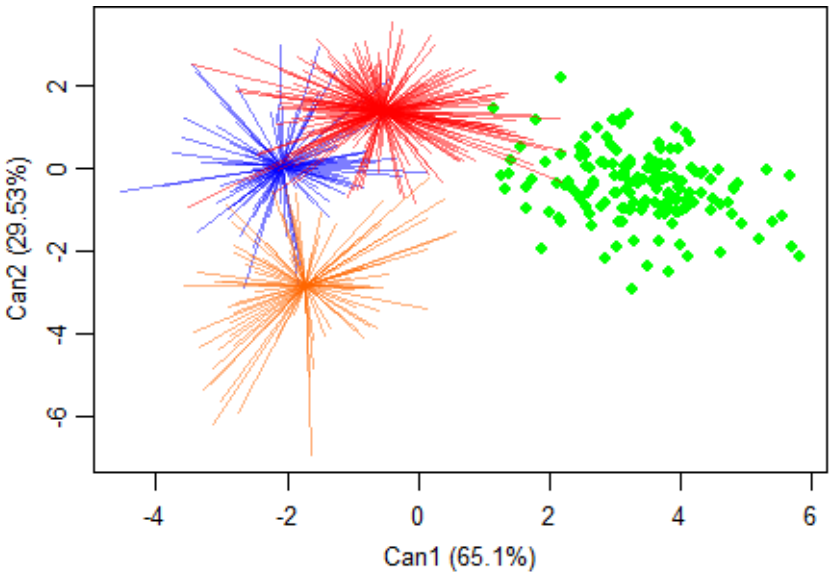
CDA can be applied by invoking the `cda.calc()` function (using the `candisc` package; Friendly & Fox, 2020). The result is an object of the class `cdadata`, and among other elements (run `?cdadata` for details) it stores total canonical structure coefficients, specifically total-sample correlations between the original variables and the canonical variates. Thus, these coefficients are used to interpret the contribution of different characters to the separation of groups. The function `summary()` prints summaries of the results of the `cda.calc()` function (variation explained by individual axes, total canonical structure coefficients).

```
cda.centaurea = cda.calc(centaurea)

summary(cda.centaurea)
#> Object of class 'cdadata'; storing results of canonical discriminant analysis
#>
#> Variation explained by individual axes:
#>                                       Can1   Can2   Can3
#> Eigenvalues                         4.4194 2.0049 0.3647
#> Eigenvalues as percentages          0.6510 0.2953 0.0537
#> Cumulative percentage of eigenvalues 0.6510 0.9463 1.0000
#>
#> Total canonical structure coefficients:
#>              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
#>  [ reached getOption("max.print") -- omitted 14 rows ]
```

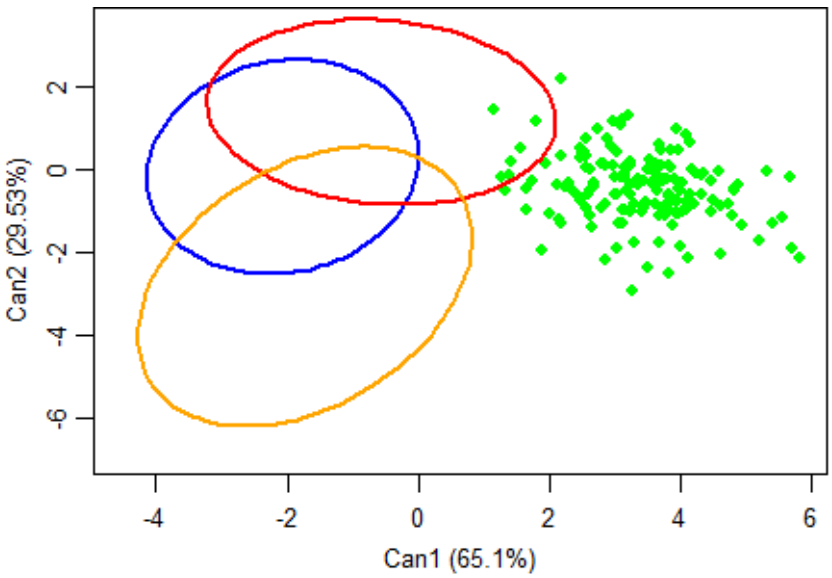
```
plotPoints(cda.centaurea, col = c("blue","green","red","orange"), axes = c(1, 2),
           pch = c(8,17,20,18), legend = T, ncol = 2, legend.pos="bottomright")
```



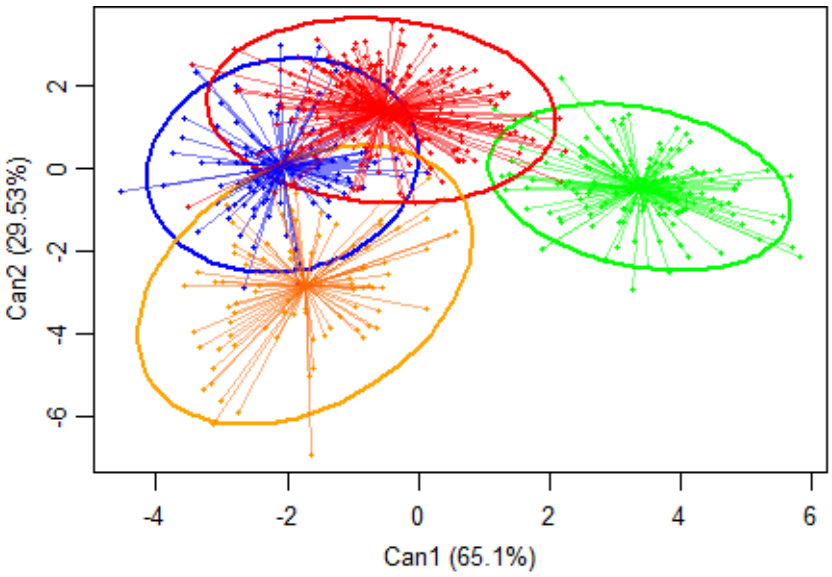

```
plotPoints(cda.centaurea, col = c(NA, "green", NA, NA), cex = 0.8)
plotAddSpiders(cda.centaurea, col = c(rgb(0,0,255,max=255,alpha=130), # blue
                                      NA, # green
                                      rgb(255,0,0,max=255,alpha=130), # red
                                      rgb(255,102,0,max=255,alpha=130))) # orange
```



```
plotPoints(cda.centaurea, col = c(NA, "green", NA, NA), cex = 0.8)
plotAddEllipses(cda.centaurea, col = c("blue", NA, "red", "orange"), lwd = 2)
```

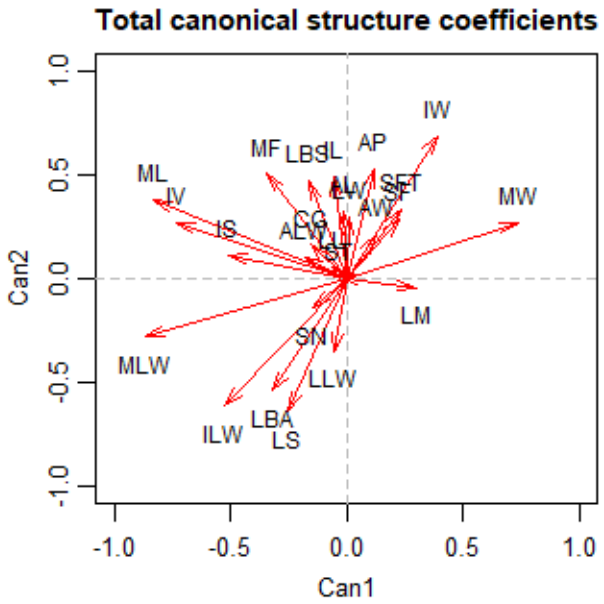



```
plotPoints(cda.centaurea, col = c("blue","green","red","orange"), cex = 0.4)
plotAddEllipses(cda.centaurea, col = c("blue","green","red","orange"), lwd = 2)
plotAddSpiders(cda.centaurea, col = c(rgb(0,0,255,max=255,alpha=130), # blue
                                      rgb(0,255,0,max=255,alpha=130), # green
                                      rgb(255,0,0,max=255,alpha=130), # red
                                      rgb(255,102,0,max=255,alpha=130))) # orange
```



This CDA ordination diagram unequivocally supports the morphological differentiation of *C. phrygia* s.str. ("ph") along the first axis.

```
plotCharacters(cda.centaurea, cex = 1.2)
```



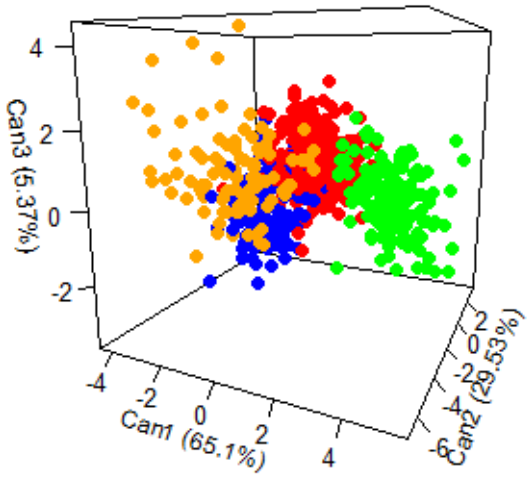
The function `plotCharacters()`, called with an object of the class `cdadata` as an argument, visualizes total canonical structure coefficients as arrows. The direction and length of the arrows indicate the characters' contribution to the separation of groups. The characters ML, MLW, IV and MW are oriented in the direction of the separation of *C. phrygia* s.str. ("ph"), so they contributed the most significantly, which is in accordance with the results of PCA.

The total canonical structure coefficients are elements of an object of the class `cdadata` and can be accessed using the `$` notation. The above-mentioned characters (ML, MLW, IV and MW) received the highest absolute scores along the first canonical axis (regardless of sign). The results can be exported using the `exportRes()` function.

```
exportRes(cda.centaurea$totalCanonicalStructure, file = "centaurea_TCS.txt")
# For demonstration purposes only. Only a subset of data is displayed.
cda.centaurea$totalCanonicalStructure
#>             Can1         Can2         Can3
#> SN  -0.145905291 -0.143820071  0.187446717
#> SF   0.226719447  0.288948819 -0.088309973
#> ST  -0.027765120  0.004883579  0.135653239
#> LW   0.007404962  0.296609347 -0.354584342
#> LS  -0.251889023 -0.640679116  0.197855575
#> IW   0.389239321  0.685339935  0.239991480
#> ILW -0.525181687 -0.611784805 -0.002541299
#> ML  -0.831038839  0.379514738  0.079785332
#> MW   0.735479254  0.267972743 -0.012135822
#> MLW -0.867777425 -0.282448838  0.009070526
#> IV  -0.735244321  0.270554318 -0.229699765
#> ALW -0.183216081  0.102614196 -0.134701510
#> AP   0.117451381  0.528786582 -0.006020908
```

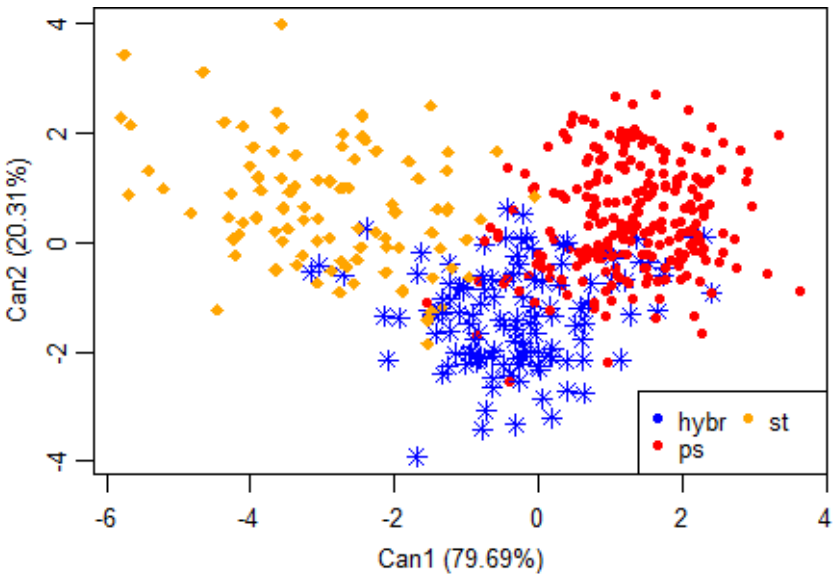
A 3D scatterplot can be produced using the `plot3Dpoints()` function. The viewing direction and slope (co-latitude) can be set using the `theta` and `phi` arguments, respectively.

```
plot3Dpoints(cda.centaurea, col = c("blue","green","red","orange"),
                            phi = 12, theta = 25)
```



To gain better insight into the differentiation among the remaining taxa (*C. pseudophrygia*, *C. stenolepis* and their putative hybrid), these taxa will be analysed separately. A new subdataset (stPsHybr) will be created by removing *C. phrygia* from the original dataset.

```
stPsHybr = deleteTaxon(centaurea, taxonName = "ph")
cda.stPsHvbr = cda.calc(stPsHvbr)
plotPoints(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
```

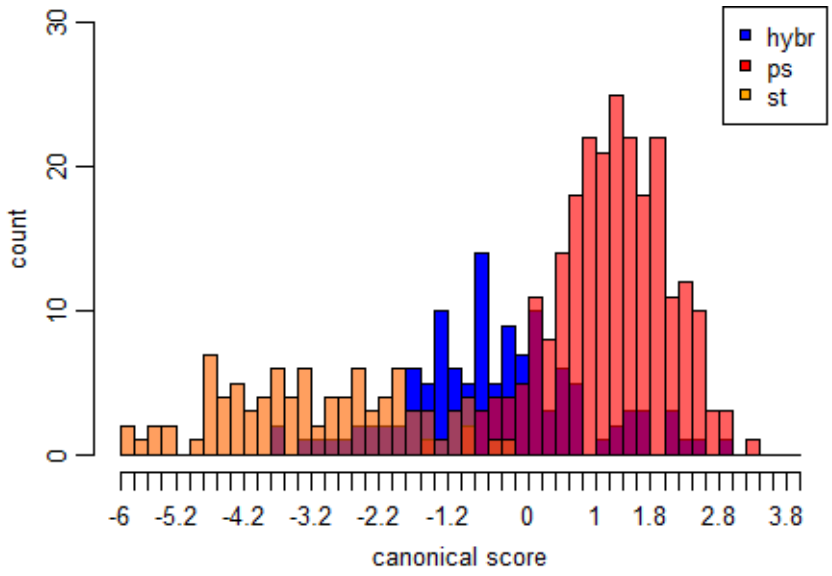
The first axis captures most of the variation. Characters correlated with this axis (IW, ILW, LS, MW and MLW) are the most suitable for the taxonomic delimitation of *C. pseudophrygia* ("ps") and *C. stenolepis* ("st"), while their hybrid exhibits intermediate values of these particular characters. The characters correlated with the second axis (IV, ML and LW) contribute to the separation of the hybrid from the parental species.

Passive prediction of samples.

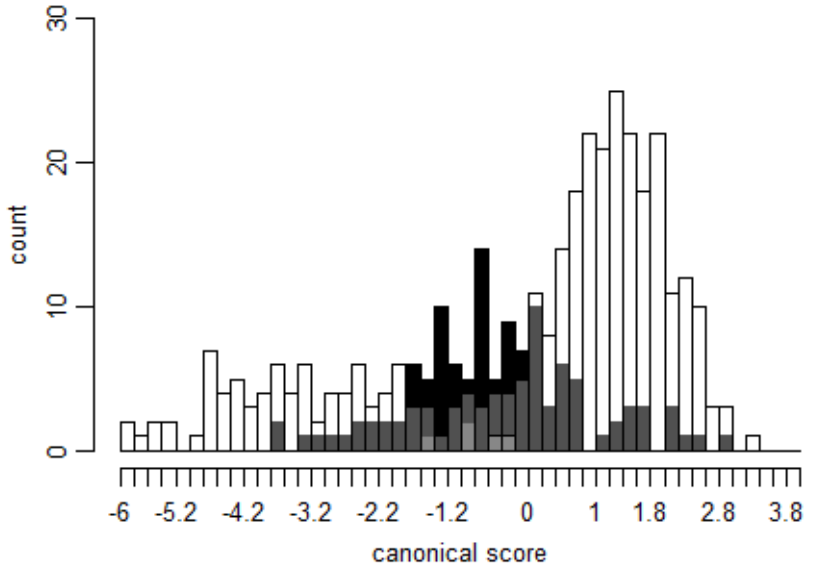
Sometimes it is desirable to passively display (predict) the position of certain samples in the canonical space formed by other samples. This approach is useful for displaying the position of hybrids, type specimens, "atypical" populations (that could not be assigned reliably to any of the predefined groups), etc. Passive samples can be specified using the `passiveSamples` argument (accepting both populations and taxa). These samples are excluded from the computation of the discriminant function and only passively predicted in the multidimensional space.

Note that in the following example, in which hybrids are passively projected into the analysis of parental species, only two "active" groups are present, "ps" and "st". In such a case, there is only one canonical axis and the sample scores are displayed as a histogram instead of a scatterplot by the `plotPoints()` function. The additional parameter `breaks` allows to define the width of intervals (histogram columns). Also note the use of semi-transparent colours to show the overlap of the groups. The colours are defined using the `rgb()` function, in which the argument `alpha` defines opacity, in the example below on a scale 0 (fully transparent) to 255 (solid).

```
cda.stPs_passiveHybr = cda.calc(stPsHybr, passiveSamples = "hybr")

plotPoints(cda.stPs_passiveHybr, legend = T, breaks = 0.2,
                col = c(rgb(0,0,255, alpha=255, max=255), # blue
                        rgb(255,0,0, alpha=160, max=255), # red
                        rgb(255,102,0, alpha=160, max=255))) # orange,
```




```
plotPoints(cda.stPs_passiveHybr, breaks = 0.2,
                col = c(rgb(0,0,0, alpha=255, max=255), # hybr - black
                        rgb(255,255,255, alpha=80, max=255), # ps - white
                        rgb(255,255,255, alpha=80, max=255))) # st - white
```



10. Classificatory discriminant analysis

Classificatory discriminant analysis (based on the packages `MASS` and `class`; Venables & Ripley, 2002) is used to classify observations (sample dataset) into known groups, using criteria (discriminant functions) based on other observations with known group membership (training dataset). Group membership should be defined using independent, non-morphological data of some sort, for example on ploidy levels, geographic origin or genetic groups, to avoid circular reasoning. If a morphological character has to be used to define groups, it should be excluded from further analyses. The results then indicate whether the retained characters are able to separate the groups in addition to the defining character. Often, however, we do not have two independent datasets (training and sample). We then opt for a cross validation procedure, in which part of the dataset is used to compute the criteria that are applied to the remaining observations, and the procedure is repeated until all observations are classified. The default is leave-one-out cross validation (the criterion is based on n – 1 individuals and applied to the individual left out), but it is also possible to use whole populations as leave-out units (as individuals from a population are not completely independent observations and may be morphologically closer to each other than to individuals from other populations). The cross-validation mode can be specified by the argument `crossval` as `"indiv"` or `"pop"` (the former, which is the default option, is applied in the example below). The resulting classification is then compared with the original (*a priori*) classification of individuals into groups. The result of classificatory DA is stored in an object of the class `classifdata`. The `classif.matrix()` function formats the results of the above functions as a summary classification table of taxa, populations or individuals. The results (classification tables) can be exported using the `exportRes()` function.

As stated above, discriminant analyses have certain requirements: (1) No character may be a linear combination of any other character; (2) No pair of characters may be highly correlated; (3) No character may be invariant in any taxon (group); (4) For the number of taxa (g), characters (p) and total number of samples (n) the following should hold: $0 < p < (n - g)$; and (5) There must be at least two groups (taxa) and in each group there must be at least two objects. Violation of these requirements is reflected by warning or error messages (rank deficiency).

To fulfil these requirements, only a subset of characters is used. Linearly independent characters can be identified by stepwise discriminant analysis and invariant characters within a taxon are revealed by the `descrTaxon()` function. If invariant characters are present, a more common practice is to add a small constant (e.g. 0.000001) to some value than to remove the character as a whole. The subset includes all characters identified as most suitable for taxonomic delimitation by PCA and CDA analyses.

```
stepdisc.calc(centaurea)
#>    Entered Removed Partial R-square    F-value        Pr > F
#> 1      MLW              0.667335064 406.554939 7.890234e-145
#> 2       ML              0.506437108 207.611046  1.192547e-92
#> 3       IW              0.200252478  50.579714  3.483830e-29
#> 4       LS              0.157685396  37.752982  2.249101e-22
#> 5       IV              0.146472637  34.550532  1.281334e-20
#> 6       MW              0.115246110  26.181821  6.249414e-16
#> 7       MF              0.091473201  20.203721  1.736040e-12
#> 8       AP              0.070455039  15.184304  1.550026e-09
#> 9       IS              0.056738406  12.030259  1.174018e-07
#> 10     LBA              0.056209421  11.891566  1.422384e-07
#> 11      LW              0.054420588  11.472159  2.538006e-07
#> 12      AL              0.047197814   9.857623  2.364358e-06
```

```
#> 13     ILW              0.038600447   7.976519  3.205143e-05
#> 14     LBS              0.033553832   6.885891  1.454076e-04
#> 15     SFT              0.026130580   5.312678  1.281299e-03
#> 16      CG              0.028448848   5.788052  6.647989e-04
#> 17      IL              0.020554879   4.141286  6.406217e-03
#> 18      LM              0.012787149   2.551697  5.474060e-02
#> 19     ALW              0.011851686   2.358787  7.062704e-02
#> 20      AW              0.013462436   2.679193  4.621888e-02
#> 21              AL      0.005503766   1.086553  3.541773e-01
#> 22      SF              0.011227184   2.229299  8.371473e-02
#>
#> Selected characters:
#> MLW, ML, IW, LS, IV, MW, MF, AP, IS, LBA, LW, AL, ILW, LBS, SFT, CG, IL, LM, ALW, AW, SF

partialCent = keepCharacter(centaurea, c("MLW", "ML", "IW", "LS", "IV",
                          "MW", "MF", "AP", "IS", "LBA", "LW", "AL", "ILW",
                          "LBS","SFT", "CG", "IL", "LM", "ALW", "AW", "SF"))

descrTaxon(partialCent, format = "$SD")
#>     group  hybr     ph    ps    st
#> 1  format    SD     SD    SD    SD
#> 2       N   120    160   240    92
#> 3     MLW 3.912  1.962 4.522 5.531
#> 4      ML  1.66  1.255 1.548 1.269
#> 5      IW 0.234  0.175 0.192 0.208
#> 6      LS 0.359  0.243 0.111 0.491
#> 7      IV 0.129      0 0.459 0.501
#> 8      MW 0.123  0.394 0.251  0.11
#> 9      MF 2.069  2.066 2.054 1.801
#> 10     AP 0.194  0.256 0.243 0.204
#> 11     IS 0.341  0.454 0.429 0.435
#> 12    LBA 0.501  0.332   0.2 0.479
#> 13     LW  0.78  1.049  0.86 0.783
#> 14     AL 0.295  0.237  0.24 0.281
#> 15    ILW 0.147  0.078 0.117 0.217
#> 16    LBS 0.374  0.191 0.493     0
#> 17    SFT 0.164  0.139 0.142 0.106
#> 18     CG 0.435  0.283 0.443  0.35
#> 19     IL 0.152  0.133 0.135 0.119
#> 20     LM     0  0.352 0.143 0.104
#> 21    ALW 0.195  0.176 0.228 0.246
#> 22     AW 0.098  0.115 0.104 0.094
#> 23     SF 9.827 10.211 9.084 8.944

partialCent$data[ partialCent$Taxon == "hybr", "LM" ][1] =
             partialCent$data[partialCent$Taxon=="hybr","LM"][1] + 0.000001
partialCent$data[ partialCent$Taxon == "ph", "IV" ][1] =
             partialCent$data[partialCent$Taxon=="ph","IV"][1] + 0.000001
partialCent$data[ partialCent$Taxon == "st", "LBS"][1] =
             partialCent$data[partialCent$Taxon=="st","LBS"][1] + 0.000001
```

If the data have an approximately normal distribution, linear (LDA; `classif.lda()`) or quadratic (QDA; `classif.qda()`) discriminant function can be used. The decision what analysis to use should be based on the homogeneity of the within-group covariance matrices (tested by Box's M-test; `BoxMTest()`). LDA assumes equality of covariances among the characters (the predictor variables). This assumption is relaxed with QDA.
The non-parametric k nearest neighbours method (`classif.knn()`) can be used without making any assumptions about data distributions.

```
boxMTest(partialCent)
#> Box's M-test for homogeneity of covariance matrices
#>  Chi-Sq (approx.) = 14628, df = 693, p-value < 2.2e-16
```

The Box's M test indicates that there is a significant difference in the covariance matrices among taxa (p-value < 0.001), so quadratic DA is recommended.

10.1 Linear discriminant analysis (LDA)

Linear discriminant analysis can be performed by invoking the `classif.lda()` function. If collinear variables are present, a warning message is returned. The problem of multicollinearity is not covered in this tutorial, but note that if the classification itself is the only thing that matters, the warning can be ignored.

```
classifRes.lda = classif.lda(partialCent)
```

```
classif.matrix(classifRes.lda, level = "taxon")
#>   Taxon   N as.hybr as.ph as.ps as.st correct correct[%]
#> 1  hybr 120      92     0    23     5      92      76.67
#> 2    ph 160       0   154     6     0     154      96.25
#> 3    ps 240      36     8   195     1     195      81.25
#> 4    st  92      12     0     3    77      77      83.70
#> 5 Total 612     140   162   227    83     518      84.64

classif.matrix(classifRes.lda, level = "pop")
#>    Population Taxon   N as.hybr as.ph as.ps as.st correct correct[%]
#> 1        BABL    ps  20      10     0     9     1       9      45.00
#> 2        BABU  hybr  20      18     0     0     2      18      90.00
#> 3         BOL    ps  20       3     0    17     0      17      85.00
#> 4         BRT    ph  20       0    20     0     0      20     100.00
#> 5         BUK    ph  20       0    15     5     0      15      75.00
#> 6        CERM    ps  20      14     0     6     0       6      30.00
#> 7        CERV    ph  20       0    20     0     0      20     100.00
#> 8        CZLE    ps  20       0     1    19     0      19      95.00
#> 9         DEB  hybr  20      15     0     4     1      15      75.00
#> 10        DOM    st  12       0     0     0    12      12     100.00
#> 11        DUB    st  20       4     0     0    16      16      80.00
#> 12       HVLT    ps  20       1     0    19     0      19      95.00
#> 13       KASH    ps  20       0     1    19     0      19      95.00
#> 14        KOT    ph  20       0    20     0     0      20     100.00
#> 15       KOZH    ps  20       0     0    20     0      20     100.00
#> 16        KRO    ph  20       0    20     0     0      20     100.00
```

```
#> 17        LES    st  20       5     0     3    12       12       60.00
#> 18        MIL    st  20       2     0     0    18       18       90.00
#> 19        NEJ    ph  20       0    20     0     0       20      100.00
#> 20       NSED    ph  20       0    19     1     0       19       95.00
#> 21       OLE1    ps  20       1     0    19     0       19       95.00
#> 22       OLE2    ps  20       1     6    13     0       13       65.00
#> 23       PRIS    ps  20       3     0    17     0       17       85.00
#> 24       PROS  hybr  20       8     0    12     0        8       40.00
#> 25        RTE  hybr  20      20     0     0     0       20      100.00
#> 26        RUS  hybr  20      17     0     2     1       17       85.00
#> 27        SOK    ps  20       1     0    19     0       19       95.00
#> 28       STCV    ph  20       0    20     0     0       20      100.00
#> 29       STGH    ps  20       2     0    18     0       18       90.00
#> 30        VIT  hybr  20      14     0     5     1       14       70.00
#> 31        VOL    st  20       1     0     0    19       19       95.00
#> 32      Total       612     140   162   227    83      518       84.64
```

A detailed classification of populations is very useful, as it can reveal atypical or incorrectly assigned populations. Most of the populations from the sample data were classified successfully (generally over 70% of correct classifications and often 100%), but some populations (BABL, CERM, LES, OLE2, PROS) yielded a maximum of 65% of correct classifications. Moreover, almost all of them were "incorrectly" clustered by hierarchical classification, and the positions of these populations in the PCA ordination plot are within clusters of different taxa. Such populations require further attention. For example, these results may indicate previously unrecognized hybridization.

The results can be exported to the clipboard or a file using the `exportRes()` function. In the example below, posterior probabilities of classification of an individual into each group are exported.

```
classif_lda = classif.matrix(classifRes.lda, level = "indiv")
exportRes(object = classif_lda, file = "lda_classifMatrix.txt")
```

10.2 Quadratic discriminant analysis (LDA)

Quadratic DA does not assume equal covariance matrices among groups and can be calculated by the `classif.qda()` function.

```
classifRes.qda = classif.qda(partialCent)


classif.matrix(classifRes.qda, level = "taxon")
#>   Taxon   N as.hybr as.ph as.ps as.st correct correct[%]
#> 1  hybr 120     105     0     3    12     105      87.50
#> 2    ph 160       0   158     2     0     158      98.75
#> 3    ps 240     118    31    74    17      74      30.83
#> 4    st  92       8     0     2    82      82      89.13
#> 5 Total 612     231   189    81   111     419      68.46
```

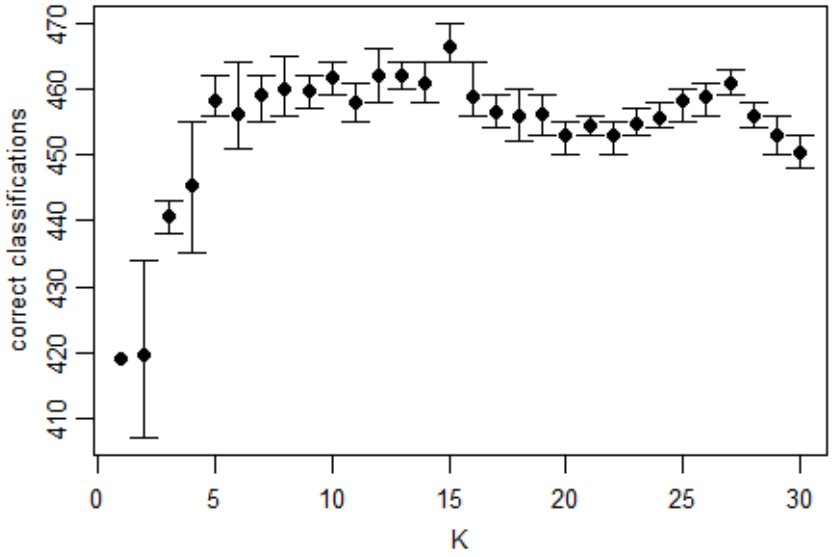
10.3 K nearest neighbour classificatory DA

The **k nearest neighbour** method classifies each individual according to the *a priori* classification of its k neighbours (by Euclidean distance) using a majority vote. As this method uses Euclidean distance, the characters are standardized to a zero mean and a unit variance. The cross-validation mode can be set as `"indiv"` or `"pop"`, in the same manner as above.

The optimum number of neighbours to consider is unknown, but it is estimated empirically from the data. The `knn.select()` function searches for the optimal k for the given dataset. The function can use two cross-validation methods (by individuals and populations) in a similar way as with classificatory discriminant analysis functions (the latter is used in the example below). The function computes the number of correctly classified individuals for k values from 1 to 30 and highlights the value with the greatest success rate. Ties (i.e. when there are the same numbers of votes for two or more groups) are broken at random, so subsequent iterations may yield different results. Therefore, the functions compute ten iterations, and the average success rates for each k are used; the minimum and maximum success rates for each k are also displayed as error bars. Note that several k values may have nearly the same success rates; if this is the case, the similarity of iterations may also be considered.

```
knn.select(partialCent, crossval = "pop")
```

```
#> Tested  10 % of Ks
#> Tested  20 % of Ks
#> Tested  30 % of Ks
#> Tested  40 % of Ks
#> Tested  50 % of Ks
#> Tested  60 % of Ks
#> Tested  70 % of Ks
#> Tested  80 % of Ks
#> Tested  90 % of Ks
#> Tested  100 % of Ks
```




```
#>
#> The highest number of correct classifications is at k = 15.
```

The highest number of correct classifications is at k = 15, so we set `k` to equal 15 in the `classif.knn()` function (to classify each individual according its 15 nearest neighbours).

```
classifRes.knn = classif.knn(partialCent, crossval = "pop", k = 15)

classif.matrix(classifRes.knn, level = "taxon")
#>   Taxon   N as.hybr as.ph as.ps as.st correct correct[%]
#> 1  hybr 120      54     0    53    13      54      45.00
#> 2    ph 160       0   152     5     3     152      95.00
#> 3    ps 240      14    14   211     1     211      87.92
#> 4    st  92      30     3    11    48      48      52.17
#> 5 Total 612      98   169   280    65     465      75.98
```

The results can be exported with the `exportRes()` function.

```
popClassifMatrix = classif.matrix(classifRes.knn, level = "taxon")

exportRes(popClassifMatrix, file = "clipboard")
```

10.4 Classification of sample individuals based on an independent training set.

The functions `classifSample.lda()`, `classifSample.qda()` and `classifSample.knn()` are designed to classify hybrid populations, type herbarium specimens, atypical samples, entirely new data, etc. A discriminant criterion is devised from the original (training) dataset and applied to the specific sample(s).

Let us remove population LES, and pretend that sample LES1116 is a type herbarium specimen of *Centaurea stenolepis* and that we want to quantify its similarity with other populations of the species. To classify LES1116, run the following code:

```
trainingSet = deletePopulation(partialCent, populationName = "LES")
typeSpecimen = keepSample(partialCent, "LES1116")

classifSample.lda(typeSpecimen, trainingSet)
#>        ID Population Taxon classification as.hybr  as.ph  as.ps  as.st
#> 1 LES1116        LES    st             st  0.0105 0.0143 0.0029 0.9723

classifSample.qda(typeSpecimen, trainingSet)
#>        ID Population Taxon classification as.hybr as.ph as.ps as.st
#> 1 LES1116        LES    st             st       0     0     0     1

classifSample.knn(typeSpecimen, trainingSet, k = 4)
#>        ID Population Taxon classification
#> 1 LES1116        LES    st             st
#>   Proportion.of.the.votes.for.the.winning.class
#> 1                                             1
```

The probability that sample LES1116 is of *C. stenolepis* is over 90%.

Further reading

A brief account of the multivariate methods used in taxonomy is given in:

Marhold K. (2011). Multivariate morphometrics and its application to monography at specific and infraspecific levels. In: Stuessy TF, Lack HW, eds. *Monographic plant systematics: fundamental assessment of plant biodiversity.* Ruggell: A.R.G. Gantner Verlag K. G., 73–99.

For a detailed description of the methods used, see:

Legendre P. & Legendre L. (2012). *Numerical Ecology*, 3rd English edn. Elsevier Science BV, Amsterdam.

Selection of papers employing the methods of multivariate morphometrics:

Lihová J., Kudoh H., Marhold K. (2010). Morphometric studies of polyploid *Cardamine* species (Brassicaceae) from Japan: solving a long-standing taxonomic and nomenclatural controversy. *Australian Systematic Botany* 23, 94-111. doi: 10.1071/sb09038

Melichárková A., Španiel S., Marhold K., Hurdu B.I., Drescher A., Zozomová-Lihová J. (2019). Diversification and independent polyploid origins in the disjunct species *Alyssum repens* from the Southeastern Alps and the Carpathians. *American Journal of Botany* 106, 1499-1518. doi: 10.1002/ajb2.1370

Skokanová K., Hodálová I., Mereďa P., Slovák M., Kučera, J. (2019). The *Cyanus tuberosus* group (Asteraceae) in the Balkans: biological entities require correct names. *Plant Systematics and Evolution* 305, 569–596. doi: 10.1007/s00606-019-01576-4

Šlenker M., Zozomová-Lihová J., Mandáková T., Kudoh H., Zhao Y., Soejima A., et al. (2018). Morphology and genome size of the widespread weed *Cardamine occulta*: how it differs from cleistogamic *C. kokaiensis* and other closely related taxa in Europe and Asia. *Botanical Journal of the Linnean Society* 187, 456-482. doi: 10.1093/botlinnean/boy030

Španiel S., Marhold K., Passalacqua N.G., Zozomová-Lihová J. (2011). Intricate variation patterns in the diploid-polyploid complex of *Alyssum montanum-A. repens* (Brassicaceae) in the Apennine Peninsula: Evidence for long-term persistence and diversification. *American Journal of Botany* 98, 1887-1904. doi: 10.3732/ajb.1100147

Šrámková G., Kolář F., Záveská E., Lučanová M., Španiel S., Kolník M., et al. (2019). Phylogeography and taxonomic reassessment of *Arabidopsis halleri* - a montane species from Central Europe. *Plant Systematics and Evolution* 305, 885-898. doi: 10.1007/s00606-019-01625-y

References

Friendly M., Fox J. (2020). candisc: Visualizing Generalized Canonical Discriminant and Canonical Correlation Analysis. R package version 0.8-3. https://CRAN.R-project.org/package=candisc

Klecka W.R. (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. (2015). MorphoTools: a set of R functions for morphometric analysis. *Plant Systematics and Evolution* 301, 1115-1121.

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

Shlens, J. (2014). A tutorial on principal component analysis. *ArXiv* preprint arXiv:1404.1100

Thorpe R.S. (1976). Biometric analysis of geographic variation and racial affinities. *Biological Reviews of the Cambridge Philosophical Society* 51, 407–425.

Venables W.N., Ripley B.D. (2002). *Modern Applied Statistics with S, Fourth edition.* New York: Springer.