Package ‘MorphoTools2’

February 3, 2022

Type Package
Title Multivariate Morphometric Analysis
Version 0.9.1.2
Date 2022-01-28
Author Marek Šlenker [aut, cre] (https://orcid.org/0000-0002-5919-890X)
Petr Koutecký [ctb] (https://orcid.org/0000-0002-3455-850X)
Karol Marhold [ctb] (https://orcid.org/0000-0002-7658-0844)
Maintainer Marek Šlenker <marek.slenker@savba.sk>
Description Tools for multivariate analyses of morphological data, wrapped in one package, to make the workflow convenient and fast. Statistical and graphical tools provide a comprehensive framework for checking and manipulating input data, statistical analyses, and visualization of results. Several methods are provided for the analysis of raw data, to make the dataset ready for downstream analyses. Integrated statistical methods include hierarchical classification, principal component analysis, principal coordinates analysis, non-metric multidimensional scaling, and multiple discriminant analyses: canonical, stepwise, and classificatory (linear, quadratic, and the non-parametric k nearest neighbours). The philosophy of the package will be described in Šlenker et al. (in prep).

URL https://github.com/MarekSlenker/MorphoTools2
BugReports https://github.com/MarekSlenker/MorphoTools2/issues
Depends R (>= 2.10)
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Imports ade4, candisc, car, class, ellipse, heplots, MASS, methods, plot3D, StatMatch, utils, vegan, stats
Suggests testthat, knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2022-02-03 12:40:06 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxMTest</td>
<td>3</td>
</tr>
<tr>
<td>boxplotCharacter</td>
<td>4</td>
</tr>
<tr>
<td>cda.calc</td>
<td>5</td>
</tr>
<tr>
<td>cda.data</td>
<td>7</td>
</tr>
<tr>
<td>centaurea</td>
<td>7</td>
</tr>
<tr>
<td>characters</td>
<td>8</td>
</tr>
<tr>
<td>classif.lda</td>
<td>9</td>
</tr>
<tr>
<td>classif.matrix</td>
<td>11</td>
</tr>
<tr>
<td>classif.data</td>
<td>12</td>
</tr>
<tr>
<td>classifSample.lda</td>
<td>12</td>
</tr>
<tr>
<td>clust</td>
<td>14</td>
</tr>
<tr>
<td>cormat</td>
<td>16</td>
</tr>
<tr>
<td>descrTaxon</td>
<td>17</td>
</tr>
<tr>
<td>exportRes</td>
<td>18</td>
</tr>
<tr>
<td>head.morphodata</td>
<td>19</td>
</tr>
<tr>
<td>histCharacter</td>
<td>20</td>
</tr>
<tr>
<td>keepTaxon</td>
<td>21</td>
</tr>
<tr>
<td>knn.select</td>
<td>22</td>
</tr>
<tr>
<td>missingCharactersTable</td>
<td>23</td>
</tr>
<tr>
<td>missingSamplesTable</td>
<td>24</td>
</tr>
<tr>
<td>morphodata</td>
<td>24</td>
</tr>
<tr>
<td>naMeanSubst</td>
<td>25</td>
</tr>
<tr>
<td>nmds.calc</td>
<td>26</td>
</tr>
<tr>
<td>nmdsdata</td>
<td>28</td>
</tr>
<tr>
<td>pca.calc</td>
<td>28</td>
</tr>
<tr>
<td>pcad.data</td>
<td>29</td>
</tr>
<tr>
<td>pcoa.calc</td>
<td>30</td>
</tr>
<tr>
<td>pcoadata</td>
<td>32</td>
</tr>
<tr>
<td>plot3Dpoints</td>
<td>32</td>
</tr>
<tr>
<td>plotAddEllipses</td>
<td>34</td>
</tr>
<tr>
<td>plotAddLabels.characters</td>
<td>35</td>
</tr>
<tr>
<td>plotAddLabels.points</td>
<td>36</td>
</tr>
<tr>
<td>plotAddLegend</td>
<td>37</td>
</tr>
<tr>
<td>plotAddSpiders</td>
<td>38</td>
</tr>
<tr>
<td>plotCharacters</td>
<td>39</td>
</tr>
<tr>
<td>plotPoints</td>
<td>41</td>
</tr>
<tr>
<td>populOTU</td>
<td>42</td>
</tr>
<tr>
<td>qqnormCharacter</td>
<td>43</td>
</tr>
<tr>
<td>read.morphodata</td>
<td>44</td>
</tr>
<tr>
<td>removeTaxon</td>
<td>45</td>
</tr>
<tr>
<td>shapiroWilkTest</td>
<td>47</td>
</tr>
<tr>
<td>stepdisc.calc</td>
<td>47</td>
</tr>
<tr>
<td>summary</td>
<td>48</td>
</tr>
<tr>
<td>transformCharacter</td>
<td>49</td>
</tr>
<tr>
<td>viewMorphodata</td>
<td>50</td>
</tr>
</tbody>
</table>

**Index** 52
The boxMTest function performs Box’s (1949) M-test for homogeneity of covariance matrices. The null hypothesis for this test is that the observed covariance matrices for the dependent variables are equal across groups.

**Usage**

```r
boxMTest(object)
```

**Arguments**

- `object` an object of class `morphodata`.

**Value**

None. Used for its side effect.

**References**


**Examples**

```r
data(centaurea)

# remove NAs and linearly dependent characters (characters with unique contributions can be identified by stepwise discriminant analysis.)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

# add a small constant to characters which are invariant within taxa
centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] = centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] = centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "st", "LBS" ][1] = centaurea$data[ centaurea$Taxon == "st", "LBS" ][1] + 0.000001

boxMTest(centaurea)
```
**boxplotCharacter**

## Box Plots

**Description**

These functions produce a box-and-whisker plot(s) of the given morphological character(s).

**Usage**

```r
boxplotCharacter(object, character, outliers = TRUE, lowerWhisker = 0.05, upperWhisker = 0.95, col = "white", border = "black", main = character, cex.main = 1.5, xlab = NULL, ylab = NULL, frame = TRUE, pch = 8, horizontal = FALSE, varwidth = FALSE, ...)

boxplotAll(object, folderName = "boxplots", outliers = TRUE, lowerWhisker = 0.05, upperWhisker = 0.95, col = "white", border = "black", main = character, cex.main = 1.5, xlab = NULL, ylab = NULL, frame = TRUE, pch = 8, horizontal = FALSE, varwidth = FALSE, width = 480, height = 480, units = "px", ...)
```

**Arguments**

- **object**: an object of class `morphodata`.
- **character**: a morphological character used to plot boxplot.
- **folderName**: folder to save produced boxplots.
- **outliers**: logical, if TRUE, the outliers are drawn.
- **lowerWhisker**: percentile to which the lower whisker is extended.
- **upperWhisker**: percentile to which the upper whisker is extended.
- **col**: background colour for the boxes.
- **border**: colour of outliers and the lines.
- **frame**: logical, if TRUE, a 'frame' (box around the plot) is drawn.
- **main**: main title for the plot.
- **cex.main**: magnification to be used for the main title.
- **pch**: plotting symbol of the outliers.
- **xlab, ylab**: title of the respective axes.
- **horizontal**: logical, indicating if the boxplot should be horizontal.
- **varwidth**: logical, if TRUE, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.
- **width**: the width of the figure.
- **height**: the height of the figure.
- **units**: the units in which height and width are given. Can be "px" (pixels, the default), "in" (inches), "cm" or "mm".
- **...**: further arguments to be passed to `boxplot` or `bxp`. 
Details

These functions modify the classical boxplot function to allow whiskers to be extended to the desired percentiles. By default, the whiskers are extended to the 5th and 95th percentiles, because of the trimmed range (without the most extreme 10% of values) used in taxa descriptions, determination keys, etc. Box defines 25th and 75th percentiles, bold horizontal line shows median (50th percentile). Missing values are ignored.

The boxplotAll function produces boxplots for each morphological character and saves them to a folder defined by the folderName argument. If it does not exist, a new folder is created.

Value

None. Used for its side effect of producing a plot(s).

Examples

data(centaurea)

boxplotCharacter(centaurea, character = "ST", col = "orange", border = "red")

boxplotCharacter(centaurea, character = "ST", outliers = FALSE,
    lowerWhisker = 0.1, upperWhisker = 0.9)

boxplotCharacter(centaurea, "ST", varwidth = TRUE, notch = TRUE,
    boxwex = 0.4, staplewex = 1.3, horizontal = TRUE)

boxplotCharacter(centaurea, "ST", boxlty = 1, medlwd = 5,
    whisklty = 2, whiskcol = "red", staplecol = "red",
    outcol = "grey30", pch = "-")

## Not run: boxplotAll(centaurea, folderName = ".\boxplots")

cda.calc

Canonical Discriminant Analysis

Description

This function performs canonical discriminant analysis.

Usage

cda.calc(object, passiveSamples = NULL)

Arguments

object an object of class morphodata.

passiveSamples taxa or populations, which will be only predicted, see Details.
Details

The `cda.calc` function performs canonical discriminant analysis using the `candisc` method from the `candisc` package. Canonical discriminant analysis finds linear combination of the quantitative variables that maximize the difference in the mean discriminant score between groups. This function allows exclude subset of samples (`passiveSamples`) from computing the discriminant function, and only passively predict them in multidimensional space. This approach is advantageous for testing the positions of “atypical” populations (e.g., putative hybrids) or for assessing positions of selected individuals (e.g., type herbarium specimens).

Value

an object of class `cdadata` with the following elements:

- **objects**
  - ID: IDs of each row of scores object.
  - Population: population membership of each row of scores object.
  - Taxon: taxon membership of each row of scores object.
  - scores: ordination scores of cases (objects, OTUs).

- **eigenValues**: eigenvalues, i.e., proportion of variation of the original dataset expressed by individual axes.
- **eigenvaluesAsPercent**: eigenvalues as percent, percentage of their total sum.
- **cumulativePercentageOfEigenvalues**: cumulative percentage of eigenvalues.
- **groupMeans**: data.frame containing the means for the taxa.
- **rank**: number of non-zero eigenvalues.
- **coeffs.raw**: matrix containing the raw canonical coefficients.
- **coeffs.std**: matrix containing the standardized canonical coefficients.
- **totalCanonicalStructure**: matrix containing the total canonical structure coefficients, i.e., total-sample correlations between the original variables and the canonical variables.
- **canrsq**: squared canonical correlations.

Examples

```r
data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))
cdaRes = cda.calc(centaurea)
summary(cdaRes)
plotPoints(cdaRes, col = c("red", "green", "blue", "red"),
          pch = c(20, 17, 8, 21), pt.bg = "orange", legend = TRUE)
```
cdadata

Class Cdadata

Description

The cdadata class is designed for storing results of canonical discriminant analysis.

Format

Class cdadata.

- objects ID IDs of each row of scores object.
- Population population membership of each row of scores object.
- Taxon taxon membership of each row of scores object.
- scores ordination scores of cases (objects, OTUs).

- eigenValues eigenvalues, i.e., proportion of variation of the original dataset expressed by individual axes.
- eigenvaluesAsPercent eigenvalues as percent, percentage of their total sum.
- cumulativePercentageOfEigenvalues cumulative percentage of eigenvalues.
- groupMeans data.frame containing the means for the taxa.
- rank number of non-zero eigenvalues.
- coeffs.raw matrix containing the raw canonical coefficients.
- coeffs.std matrix containing the standardized canonical coefficients.
- totalCanonicalStructure matrix containing the total canonical structure coefficients, i.e., total-sample correlations between the original variables and the canonical variables.
- canrsq squared canonical correlations.

centaurea

25 Morphological Characters of Three Species of the Centaurea phrygia Complex

Description

The sample data include part of data sets from previously published studies by Koutecky (2007) and Koutecky et al. (2012): 25 morphological characters (see the cited studies for details) of the vegetative (stems and leaves) and reproductive structures (capitula and achenes) of three diploid species of the Centaurea phrygia complex: C. phrygia L. s.str. (abbreviated “ph”), C. pseudophrygia C.A.Mey. (“ps”) and C. stenolepis A.Kern. (“st”). Moreover, a fourth group includes the putative hybrid of the C. pseudophrygia and C. stenolepis (“hybr”). The data represent 8, 12, 7 and 6 populations for each group, respectively, and 20 individuals per population, with one exception in which only 12 individuals were available. All morphological characters are either quantitative (sizes, counts, or ratios) or binary (two characters states or presence/absence). In four characters of achenes (AL, AW, ALW, AP), there are missing data because fruits were not available in all individuals. In two populations of C. stenolepis (LIP, PREL) fruits were completely missing. In total, the data set includes 652 individuals (453 complete) from 33 populations (31 complete).
Usage

data(centaurea)

Format

an object of class *morphodata* with the following elements:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td>IDs of each row of data object.</td>
</tr>
<tr>
<td><strong>Population</strong></td>
<td>population membership of each row of data object.</td>
</tr>
<tr>
<td><strong>Taxon</strong></td>
<td>taxon membership of each row of data object.</td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>data.frame of individuals (rows) and values of morphological characters (columns).</td>
</tr>
</tbody>
</table>

References


### characters

*List Morphological Characters*

**Description**

Returns list morphological characters of object.

**Usage**

characters(object)

**Arguments**

- **object**: an object of class *morphodata, pcadata* or *cdadata*.

**Value**

A character vector containing names of morphological characters of object.

**Examples**

data(centaurea)

characters(centaurea)
**classif.lda**

**Classificatory Discriminant Analysis**

**Description**

These functions computes discriminant function for classifying observations. Linear discriminant function (classif.lda), quadratic discriminant function (classif.qda), or nonparametric k-nearest neighbours classification method (classif.knn) can be used.

**Usage**

```r
classif.lda(object, crossval = "indiv")
classif.qda(object, crossval = "indiv")
classif.knn(object, k, crossval = "indiv")
```

**Arguments**

- `object` an object of class `morphodata`.
- `crossval` crossvalidation mode, sets individual ("indiv"; default, one-leave-out method) or whole populations ("pop") as leave-out unit.
- `k` number of neighbours considered for the k-nearest neighbours method.

**Details**

The `classif.lda` and `classif.qda` performs classification using linear and quadratic discriminant functions with cross-validation using the `lda` and `qda` functions from the package `MASS`. The prior probabilities of group memberships are equal.

LDA and QDA analyses have some requirements: (1) no character can be a linear combination of any other character; (2) no pair of characters can be highly correlated; (3) no character can be invariant in any taxon; (4) for the number of taxa (g), characters (p) and total number of samples (n) 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 some of these assumptions may result in warnings or error messages (rank deficiency).

Nonparametric classification method k-nearest neighbours is performed using the `knn` and `knn.cv` functions from the package `class`.

The mode of crossvalidation is set by the parameter `crossval`. The default "indiv" uses the standard one-leave-out method. However, as some hierarchical structure is usually present in the data (individuals from a population are not completely independent observations, as they are morphologically closer to each other than to individuals from other populations), the value "pop" sets whole populations as leave-out units. The latter method does not allow classification if there is only one population for a taxon and is more sensitive to "atypical" populations, which usually leads to a somewhat lower classification success rate.
Value

an object of class classifdata with the following elements:

ID IDs of each row.
Population population membership of each row.
Taxon taxon membership of each row.
classif classification from discriminant analysis.
prob posterior probabilities of classification into each taxon (if calculated by classif.lda or classif.qda), or proportion of the votes for the winning class (calculated by classif.knn)
correct logical, correctness of classification.

See Also
classifSample.lda, classif.matrix, knn.select

Examples

data(centaurea)

# remove NAs and linearly dependent characters (characters with unique contributions
# can be identified by stepwise discriminant analysis.)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))
centaurea = keepCharacter(centaurea, c("MLW", "ML", "IW", "LS", "IV", "MW", "MF",
"AP", "IS", "LBA", "LW", "AL", "ILW", "LBS",
"SFT", "CG", "IL", "LM", "ALW", "AW", "SF") )

# add a small constant to characters witch are invariant within taxa
centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] =
    centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] =
    centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "st", "LBS" ][1] =
    centaurea$data[ centaurea$Taxon == "st", "LBS" ][1] + 0.000001

# classification by linear discriminant function
classifRes.lda = classif.lda(centaurea, crossval = "indiv")

# classification by quadratic discriminant function
classifRes.qda = classif.qda(centaurea, crossval = "indiv")

# classification by nonparametric k-nearest neighbour method
# use knn.select to find the optimal K.
knn.select(centaurea, crossval = "pop")
classifRes.knn = classif.knn(centaurea, k = 12, crossval = "pop")

# exporting results
classif.matrix(classifRes.lda, level = "taxon")
classif.matrix

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

---

classif.matrix

Format the Classifdata to Summary Table

Description

The *classif.matrix* method formats the results stored in *classifdata* class to a summary classification table of taxa, populations, or individuals.

Usage

```r
classif.matrix(result, level = "taxon")
```

Arguments

- **result**: an object of class *classifdata*.
- **level**: level of grouping of classification matrix, "taxon" (default), populations ("pop"), or individuals ("indiv")

Value

A *data.frame*, summary classification table.

Examples

data(centaurea)

centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

# classification by linear discriminant function
classifRes.lda = classif.lda(centaurea, crossval = "indiv")

# exporting results
classif.matrix(classifRes.lda, level = "taxon")
classif.matrix(classifRes.lda, level = "pop")
classifdata  

Class classifdata

Description
The classifdata class is designed for storing results of classificatory discriminant analysis.

Format
Class classifdata.

- **ID** IDs of each row.
- **Population** population membership of each row.
- **Taxon** taxon membership of each row.
- **classif** classification from discriminant analysis.
- **prob** posterior probabilities of classification into each taxon (if calculated by classif.lda or classif.qda), or proportion of the votes for the winning class (calculated by classif.knn)
- **correct** logical, correctness of classification.

classifSample.lda  

Classificatory Discriminant Analysis

Description
These functions compute discriminant function based on an independent training set and classify observations in sample set. Linear discriminant function (classifSample.lda), quadratic discriminant function (classifSample.qda), or nonparametric k-nearest neighbour classification method (classifSample.knn) can be used.

Usage

classifSample.lda(sampleData, trainingData)

classifSample.qda(sampleData, trainingData)

classifSample.knn(sampleData, trainingData, k)

Arguments

- **sampleData** observations which should be classified. An object of class morphodata.
- **trainingData** observations for computing discriminant function. An object of class morphodata.
- **k** number of neighbours considered.
classifSample.lda

Details

The classifSample.lda and classifSample.qda functions perform classification using linear and quadratic discriminant functions from the package MASS. Nonparametric classification method classifSample.knn (k-nearest neighbours) is performed using the knn functions from the package class. The classifSample functions are designed to classify hybrid populations, type herbarium specimens, atypical samples, entirely new data, etc. Discriminant criterion is developed from the original (training) dataset and applied to the specific sample (set).

LDA and QDA analyses have some requirements: (1) no character can be a linear combination of any other character; (2) no pair of characters can be highly correlated; (3) no character can be invariant in any taxon (group); (4) for the number of taxa (g), characters (p) and total number of samples (n) 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 some of these assumptions may result in warnings or error messages (rank deficiency).

Value

an object of class classifdata with the following elements:

- **ID**
  IDs of each row.

- **Population**
  population membership of each row.

- **Taxon**
  taxon membership of each row.

- **classif**
  classification from discriminant analysis.

- **prob**
  posterior probabilities of classification into each taxon (if calculated by classif.lda or classif.qda), or proportion of the votes for the winning class (calculated by classif.knn)

- **correct**
  logical, correctness of classification.

See Also

classif.lda, classif.matrix, knn.select

Examples

data(centaurea)

# remove NAs and linearly dependent characters (characters with unique contributions # can be identified by stepwise discriminant analysis.)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

# add a small constant to characters which are invariant within taxa
centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] =
  centaurea$data[ centaurea$Taxon == "hybr", "LM" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] =
  centaurea$data[ centaurea$Taxon == "ph", "IV" ][1] + 0.000001
centaurea$data[ centaurea$Taxon == "st", "LBS" ][1] =

centaurea$data[ centaurea$Taxon == "st", "LBS"][1] + 0.000001

trainingSet = removePopulation(centaurea, populationName = "LES")
LES = keepPopulation(centaurea, populationName = "LES")

# classification by linear discriminant function
classifSample.lda(LES, trainingSet)
# classification by quadratic discriminant function
classifSample.qda(LES, trainingSet)
# classification by nonparametric k-nearest neighbour method
# use knn.select to find the optimal K.
knn.select(trainingSet)
classifSample.knn(LES, trainingSet, k = 12)

---

clust

Hierarchical Clustering

Description

Hierarchical cluster analysis of objects.

Usage

clust(object, distMethod = "Euclidean", clustMethod = "UPGMA", binaryChs = NULL, nominalChs = NULL, ordinalChs = NULL)

Arguments

object an object of class morphodata.
distMethod the distance measure to be used. This must be one of: "Euclidean" (default), "Manhattan", "Minkowski", "Jaccard", "simpleMatching", or "Gower". See details.
clustMethod the agglomeration method to be used: "average" (= "UPGMA"; default), "complete", "ward.D" (= "Ward"), "ward.D2", "single", "Mcquitty" (= "WPGMA"), "median" (= "WPGMC") or "centroid" (= "UPGMC"). See hclust for details.
binaryChs, nominalChs, ordinalChs names of categorical ordinal, categorical nominal (multistate), and binary characters. Needed for Gower’s dissimilarity coefficient only, see details.
Details

This function performs agglomerative hierarchical clustering. Typically, populations are used as OTUs (operational taxonomic units). Characters are standardised to a zero mean and a unit standard deviation.

Various measures of distance between the observations (rows) are applicable: (1) coefficients of distance for quantitative and binary characters: "Euclidean", "Manhattan", "Minkowski"; (2) similarity coefficients for binary characters: "Jaccard" and simple matching ("simpleMatching"); (3) coefficient for mixed data: "Gower". Note that the other than default methods for clustering and distance measurement are rarely used in morphometric analyses.

The Gower's dissimilarity coefficient can handle different types of variables. Characters have to be divided into four categories: (1) quantitative characters, (2) categorical ordinal characters, (3) categorical nominal (multistate) characters, and (4) binary characters. All characters are considered to be quantitative characters unless otherwise specified. Other types of characters have to be explicitly specified. To mark characters as ordinal, nominal, or binary, enumerate them by names using ordinalChs, nominalChs, and binaryChs arguments, respectively.

Value

An object of class 'hclust'. It encodes a stepwise dendrogram.

Examples

data(centaurea)

clustering.UPGMA = clust(centaurea)

plot(clustering.UPGMA, cex = 0.6, frame.plot = TRUE, hang = -1,
main = "", sub = "", xlab = "", ylab = "distance")

# using Gower's method
data = list(
  ID = as.factor(c("id1","id2","id3","id4","id5","id6")),
  Population = as.factor(c("Pop1", "Pop1", "Pop2", "Pop2", "Pop3", "Pop3")),
  data = data.frame(
    stemBranching = c(1, 1, 1, 0, 0, 0), # binaryChs
    petalColour = c(1, 1, 2, 3, 3, 3), # nominalChs; 1=white, 2=red, 3=blue
    leaves = c(1,1,1,2,2,3), # nominalChs; 1=simple, 2=palmately compound, 3=pinnately compound
    taste = c(2, 2, 2, 3, 1, 1), # ordinal; 1=hot, 2=hotter, 3=hottest
    stemHeight = c(10, 11, 14, 22, 23, 21), # quantitative
    leafLength = c(8, 7.1, 9.4, 1.2, 2.3, 2.1) ) # quantitative
)
attr(data, "class") = "morphodata"

clustering.GOWER = clust(data, distMethod = "Gower", clustMethod = "UPGMA",
  binaryChs = c("stemBranching"),
  nominalChs = c("petalColour", "leaves"),
  ordinalChs = c("taste"))
cormat

Correlations of Characters

Description

The cormat function calculates the matrix of the correlation coefficients of the characters.

Usage

cormat(object, method = "Pearson")
cormatSignifTest(object, method = "Pearson", alternative = "two.sided")

Arguments

object an object of class morphodata.
method a character string indicating which correlation coefficient is to be used for the test. One of "Pearson" (default), or "Ppearman" can be applied.
alternative indicates the alternative hypothesis and must be one of "two.sided" (default), "greater" (positive association) or "less" (negative association).

Details

This function returns table with pairwise correlation coefficients for each pair of morphological characters. The result is formatted as a data.frame to allow export with the exportRes function. Significance tests are usually unnecessary for morphometric analysis. Anyway, if tests are needed, they can be computed using the cormatSignifTest function.

Value

A data.frame, storing correlation coefficients for each pair of morphological characters.

Examples

data(centaurea)
correlations.p = cormat(centaurea, method = "Pearson")
correlations.s = cormat(centaurea, method = "Spearman")

## Not run: exportRes(correlations.p, file = "correlations.pearson.txt")
## Not run: exportRes(correlations.s, file = "correlations.spearman.txt")
correlations.p = cormatSignifTest(centaurea, method = "Pearson")
descrTaxon

**Description**

These functions calculate the descriptive statistics of each character in the whole dataset, each taxon and each population.

**Usage**

```r
descrTaxon(object, format = NULL, decimalPlaces = 3)
descrPopulation(object, format = NULL, decimalPlaces = 3)
descrAll(object, format = NULL, decimalPlaces = 3)
```

**Arguments**

- `object` an object of class `morphodata`.
- `format` form to which will be formatted descriptive characters. See Details.
- `decimalPlaces` the number of a digit to the right of a decimal point.

**Details**

The following statistics are computed: number of observations, mean, standard deviation, and the percentiles: 0% (minimum), 5%, 25% (lower quartile), 50% (median), 75% (upper quartile), 95% and 100% (maximum).

The `format` argument brings a handy way how to receive only what is wanted and in format what is desired. Otherways, if format remains `NULL`, output table contains all calculated descriptors. The format argument is a single string, where keywords will be replaced by particular values.

Keywords: "$\text{MEAN}$" = mean; "$\text{SD}$" = standard deviation; "$\text{MIN}$" = minimum; "$5\%$" = 5th percentile; "$25\%$" = 25th percentile (lower quartile); "$\text{MEDIAN}$" = median (50th percentile); "$75\%$" = 75th percentile (upper quartile); "$95\%$" = 95th percentile; "$\text{MAX}$" = maximum.

**Value**

A data.frame with calculated statistical descriptors.

**Examples**

```r
data(centaurea, decimalPlaces = 3)
descrTaxon(centaurea)
descrTaxon(centaurea, format = "($\text{MEAN} \pm \text{SD})")
descrPopulation(centaurea, format = "$\text{MEAN} (\text{MIN} - \text{MAX})$")
```
descrAll(centaurea, format = "\$MEAN \pm \$SD (\$5\% - \$95\%)")

exportRes

Export Data

Description

This function is designed for exporting results, stored in objects of MorphoTools2 package.

Usage

exportRes(object, file = "", dec = ".", sep = "\t", row.names = FALSE, col.names = TRUE)

Arguments

object an object to be exported.
file either a character string naming a file or a connection opened for writing (e.g., "clipboard"). "" indicates output to the console.
dec the character used for decimal points.
sep the column separator character.
row.names logical, if TRUE, row names of the object are to be written.
col.names logical, if TRUE, column names of the object are to be written.

Value

None. Used for its side effect.

Examples

data(centaurea)
descr = descrTaxon(centaurea, format = "($MEAN \pm $SD)")
## Not run: exportRes(descr, file = "centaurea_descrTax.txt")
Return the First or Last Parts of an Object

Description

Returns the first or last parts of an object.

Usage

```r
## S3 method for class 'classifdata'
head(x, n = 6, ...)
```

```r
## S3 method for class 'classifdata'
tail(x, n = 6, ...)
```

```r
## S3 method for class 'morphodata'
head(x, n = 6, ...)
```

```r
## S3 method for class 'morphodata'
tail(x, n = 6, ...)
```

Arguments

- **x**: an object of class `morphodata` or `classifdata`.
- **n**: number of rows to print.
- **...**: arguments to be passed to or from other methods.

Details

Object passed as parameter is formatted to `data.frame`. A `head()` (`tail()`) returns the first (last) `n` rows when `n >= 0` or all but the last (first) `n` rows when `n < 0`.

Value

A `data.frame`, containing the first or last `n` individuals of the passed object.

Examples

```r
data(centaurea)

head(centaurea)
tail(centaurea)
```
Histograms of Characters

Description

Histograms are produced for the level of taxa/groups, to display a within-group distribution of each taxon for a particular character, and its deviation from the normal distribution (red line).

Usage

```r
histCharacter(object, character, taxon = levels(object$Taxon), histogram = TRUE, col = "lightgray", main = NULL, densityLine = TRUE, normDistLine = TRUE, ...)
```

```r
histAll(object, folderName = "histograms", taxon = levels(object$Taxon), histogram = TRUE, col = "lightgray", main = NULL, densityLine = TRUE, normDistLine = TRUE, width = 480, height = 480, units = "px", ...)
```

Arguments

- **object**: an object of class `morphodata`.
- **character**: a morphological character used to plot histogram.
- **folderName**: folder to save produced histograms.
- **col**: colour to be used to fill the bars.
- **taxon**: taxa which should be plotted, default is to plot all of the taxa.
- **main**: a main title for the plot.
- **histogram**: logical, if TRUE, the histograms will be drawn.
- **densityLine**: logical, if TRUE, the density line smoothing out the histogram will be drawn.
- **normDistLine**: logical, if TRUE, the normal distribution curve will be drawn.
- **width**: the width of the figure.
- **height**: the height of the figure.
- **units**: the units in which `height` and `width` are given. Can be "px" (pixels, the default), "in" (inches), "cm" or "mm".
- **...**: further arguments to be passed to `hist` or graphical parameters `par`.

Value

None. Used for its side effect of producing a plot(s).

Examples

```r
data(centaurea)

histCharacter(centaurea, character = "IW", breaks = seq(0.5, 2.5, 0.1))
```

```
## Not run: histAll(centaurea, folderName = "./histograms")
```
keepTaxon

Keep Items (Taxa, Populations, Samples, Morphological Characters) in an Morphodata Object (and Remove Others)

Description
These functions keep only selected taxa, populations, samples or morphological characters in morphodata object. The samples can be kept by names using sampleName argument, or by the threshold. Each sample holding less or equal portion of missing data than the desired threshold (missingPercentage) will be kept. Only one parameter can be specified in one run.

Usage

call the functions:

keepTaxon(object, taxonName)
keepPopulation(object, populationName)
keepSample(object, sampleName = NULL, missingPercentage = NA)
keepCharacter(object, characterName)

Arguments

- object: an object of class morphodata.
- taxonName: vector of taxa to be kept.
- populationName: vector of populations to be kept.
- sampleName: vector of samples to be kept.
- missingPercentage: a numeric, samples holding less or equal portion of missing data than specified by missingPercentage will be kept.
- characterName: vector of characters to be kept.

Value

an object of class morphodata with the following elements:

- ID: IDs of each row of data object.
- Population: population membership of each row of data object.
- Taxon: taxon membership of each row of data object.
- data: data.frame of individuals (rows) and values of measured morphological characters (columns).
Examples

```r
data(centaurea)

centaurea.hybr = keepTaxon(centaurea, "hybr")
centaurea.PHybr = keepTaxon(centaurea, c("ph", "hybr"))
centaurea.PREL = keepPopulation(centaurea, "PREL")
centaurea.NA_0.1 = keepSample(centaurea, missingPercentage = 0.1)
centaurea.stem = keepCharacter(centaurea, c("SN", "SF", "ST"))
```

---

**knn.select**

Search for the Optimal K-nearest Neighbours

**Description**

This function search for the optimal number of neighbours for the given data set for k-nearest neighbour cross-validatory classification.

**Usage**

```r
knn.select(object, crossval = "indiv")
```

**Arguments**

- `object` an object of class `morphodata`.
- `crossval` crossvalidation mode, sets individual ("indiv"; default, one-leave-out method) or whole populations ("pop") as leave-out unit.

**Details**

The `knn.select` function compute number of correctly classified individuals for k values ranging from 1 to 30 and highlight the value with the highest success rate. Ties (i.e., when there are the same numbers of votes for two or more groups) are broken at random, and thus several iterations may yield different results. Therefore, the functions compute 10 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.

The mode of crossvalidation is set by the parameter `crossval`. The default "indiv" uses the standard one-leave-out method. However, as some hierarchical structure is usually present in the data (individuals from a population are not completely independent observations, as they are morphologically closer to each other than to individuals from other populations), the value "pop" sets whole populations as leave-out units. The latter method does not allow classification if there is only one population for a taxon and is more sensitive to "atypical" populations, which usually leads to a somewhat lower classification success rate.
Value

Optimal number of neighbours is written to the console, and plot displaying all Ks is produced.

See Also
classif.lda, classifSample.lda, classif.qda, classifSample.qda, classif.knn, classifSample.knn

Examples
data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

# classification by nonparametric k-nearest neighbour method
knn.select(centaurea, crossval = "indiv")
classifRes.knn = classif.knn(centaurea, k = 12, crossval = "indiv")

missingCharactersTable

Summarize Missing Data

Description

Summarize percentage and number of missing values on the desired grouping level.

Usage

missingCharactersTable(object, level)

Arguments

object   an object of class morphodata.
level    level of grouping, one of the following: "taxon", populations ("pop"), or individuals ("indiv")

Value

A data.frame summarizing a number of missing values.

Examples

data(centaurea)

missingCharactersTable(centaurea, level = "pop")
missingSamplesTable  
*Summarize Missing Data*

**Description**
Summarize number of missing values for each character on the desired grouping level.

**Usage**

```r
missingSamplesTable(object, level)
```

**Arguments**

- `object` an object of class `morphodata`.
- `level` level of grouping, one of the following: "taxon", populations ("pop"), or individuals ("indiv").

**Value**
A `data.frame` summarizing a number of missing values.

**Examples**

```r
data(centaurea)

missingSamplesTable(centaurea, level = "pop")
```

---

**morphodata**  
*Class morphodata*

**Description**

The `morphodata` class is designed for storing morphological data of individuals, their IDs and it’s appertaining to population and taxon.

**Format**

Class `morphodata`.

- **ID** IDs of each row of data object.
- **Population** population membership of each row of data object.
- **Taxon** taxon membership of each row of data object.
- **data** `data.frame` of individuals (rows) and values of measured morphological characters (columns).
**naMeanSubst**

*Replace Missing Data by Population Average*

**Description**

This function substitutes missing data using the average value of the respective character in the respective population.

**Usage**

`naMeanSubst(object)`

**Arguments**

- `object`: an object of class `morphodata`.

**Details**

Generally, most of the multivariate analyses require a full data matrix. The preferred approach is to reduce the data set to complete observations only (i.e., perform the casewise deletion of missing data) or to remove characters for which there are missing values. The use of mean substitution, which introduces values that are not present in the original data, is justified only if (1) there are relatively few missing values, (2) these missing values are scattered throughout many characters (each character includes only a few missing values) and (3) removing all individuals or all characters with missing data would unacceptably reduce the data set.

**Value**

an object of class `morphodata` with the following elements:

- **ID**: IDs of each row of `data` object.
- **Population**: population membership of each row of `data` object.
- **Taxon**: taxon membership of each row of `data` object.
- **data**: data frame of individuals (rows) and values of measured morphological characters (columns).

**Examples**

```r
data(centaurea)

centaurea = naMeanSubst(centaurea)
```
Non-metric Multidimensional Scaling (NMDS)

Description

This function performs Non-metric multidimensional scaling.

Usage

nmds.calc(object, distMethod = "Euclidean", k = 3, binaryChs = NULL, nominalChs = NULL, ordinalChs = NULL)

Arguments

- object: an object of class morphodata.
- distMethod: the distance measure to be used. This must be one of: "Euclidean", "Manhattan", "Minkowski", "Jaccard", "simpleMatching", or "Gower". See details.
- k: number of dimensions.
- binaryChs, nominalChs, ordinalChs: names of categorical ordinal, categorical nominal (multistate), and binary characters. Needed for Gower’s dissimilarity coefficient only, see details.

Details

The nmds.calc function performs non-metric multidimensional scaling using the monoMDS function from package vegan. The main threat of NMDS is, that this method doesn’t preserve distances among objects in the original character space and approximates only the order of the dissimilarities among objects, based on any coefficient of similarity or distance.

Further, multiple runs of the NMDS analysis are needed to ensure that the stable ordination has been reached, as anyone run may get “trapped” in local optima which are not representative of true similarities.

The stress value reflects how well the ordination summarizes the observed relationship among the samples. A rule of thumb, 0.1-0.2 is considered fairly good, but there is no general rule since the stress is greatly influenced by the number of points. Since stress decreases as dimensionality increases, the optimal solution is when the decrease in stress is small after decreasing the number of dimensions.

Various measures of distance between the observations (rows) are applicable: (1) coefficients of distance for quantitative and binary characters: "Euclidean", "Manhattan", "Minkowski"; (2) similarity coefficients for binary characters: "Jaccard" and simple matching ("simpleMatching"); (3) coefficient for mixed data: ("Gower").

The Gower’s dissimilarity coefficient can handle different types of variables. Characters have to be divided into four categories: (1) quantitative characters, (2) categorical ordinal characters, (3) categorical nominal (multistate) characters, and (4) binary characters. All characters are considered to be quantitative characters unless otherwise specified. Other types of characters have to be explicitly specified. To mark characters as ordinal, nominal, or binary, enumerate them by names using ordinalChs, nominalChs, and binaryChs arguments, respectively.
Value

an object of class `nmdsdata` with the following elements:

- **objects**
  - **ID**: IDs of each row of scores object.
  - **Population**: population membership of each row of scores object.
  - **Taxon**: taxon membership of each row of scores object.
  - **scores**: ordination scores of cases (objects, OTUs).

- **stress**: stress value, e.i., goodness of fit.

- **groupMeans**: data.frame containing the means for the taxa.

- **distMethod**: used distance measure.

- **rank**: number of positive eigenvalues.

Examples

data(centaurea)
nmdsRes = nmds.calc(centaurea, distMethod = "Euclidean", k = 3)
summary(nmdsRes)
plotPoints(nmdsRes, axes = c(1,2), col = c("red", "green", "blue", "black"),
pch = c(20,17,8,21), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")

# using Gower's method
data = list(
  ID = as.factor(c("id1","id2","id3","id4","id5","id6")),
  Population = as.factor(c("Pop1", "Pop1", "Pop2", "Pop2", "Pop3", "Pop3")),
  data = data.frame(
    stemBranching = c(1, 1, 1, 0, 0, 0), # binaryChs
    petalColour = c(1, 1, 2, 3, 3, 3), # nominalChs; 1=white, 2=red, 3=blue
    leaves = c(1,1,1,2,2,3), # nominalChs; 1=simple, 2=palmately compound, 3=pinnately compound
    taste = c(2, 2, 2, 3, 1, 1), # ordinal; 1=hot, 2=hotter, 3=hottest
    stemHeight = c(10, 11, 14, 22, 23, 21), # quantitative
    leafLength = c(8, 7.1, 9.4, 1.2, 2.3, 2.1) # quantitative
  )
attr(data, "class") = "morphodata"
nmdsGower = nmds.calc(data, distMethod = "Gower", k = 2, binaryChs = c("stemBranching"),
nominalChs = c("petalColour", "leaves"), ordinalChs = c("taste"))
plotPoints(nmdsGower, axes = c(1,2), col = c("red","green"),
pch = c(20,17), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")
nmdsdata  

*Class nmdsdata*

**Description**

The nmdsdata class is designed for storing results of non-metric multidimensional scaling (NMDS).

**Format**

Class nmdsdata.

- **objects ID**: IDs of each row of scores object.
- **Population**: population membership of each row of scores object.
- **Taxon**: taxon membership of each row of scores object.
- **scores**: ordination scores of cases (objects, OTUs).
- **stress**: stress value, e.i., goodness of fit.
- **groupMeans**: data.frame containing the means for the taxa.
- **distMethod**: used distance measure.
- **rank**: number of positive eigenvalues.

pca.calc  

*Principal Component Analysis*

**Description**

This function performs principal component analysis.

**Usage**

```
pca.calc(object)
```

**Arguments**

- **object**: an object of class *morphodata*.

**Details**

The pca.calc function performs an R type principal component analysis using the R base *princomp* function. Principal component analysis is a variable reduction procedure. It reduces original variables into a smaller number of principal components (artificial variables) that will account for most of the variance in the observed variables.

**Value**

an object of class *pcadata* with the following elements:

- **objects**
The \texttt{pcadata} class is designed for storing results of principal component analysis (PCA).

### Format

Class \texttt{pcadata}.

- **objects** \texttt{ID} IDs of each row of scores object.
  - **Population** population membership of each row of scores object.
  - **Taxon** taxon membership of each row of scores object.
  - **scores** ordination scores of cases (objects, OTUs).
- **eigenVectors** matrix of eigenvectors (i.e., a matrix of characters loadings).
- **eigenValues** eigenvalues of principal components, i.e., proportion of variation of the original dataset expressed by individual axes.
- **eigenvaluesAsPercent** eigenvalues as percent, percentage of their total sum.
- **cumulativePercentageOfEigenvalues** cumulative percentage of eigenvalues.
- **groupMeans** data.frame containing the means for the taxa.
- **rank** number of principal components.
- **center, scale** the centring and scaling of the input data.

### Examples

```r
data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))
pcaRes = pca.calc(centaurea)
summary(pcaRes)
plotPoints(pcaRes, axes = c(1,2), col = c("red", "green", "blue", "black"),
pch = c(20,17,8,21), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")
```

---

### Description

The \texttt{pcadata} class is designed for storing results of principal component analysis (PCA).

### Class pcadata
**pcoa.calc**  
*Principal Coordinates Analysis (PCoA)*

**Description**

This function performs principal coordinates analysis.

**Usage**

```r
pcoa.calc(object, distMethod = "Euclidean", binaryChs = NULL,
nominalChs = NULL, ordinalChs = NULL)
```

**Arguments**

- `object`:
  - an object of class `morphodata`.
- `distMethod`:
  - the distance measure to be used. This must be one of: "Euclidean", "Manhattan", "Minkowski", "Jaccard", "simpleMatching", or "Gower". See details.
- `binaryChs`, `nominalChs`, `ordinalChs`:
  - names of categorical ordinal, categorical nominal (multistate), and binary characters. Needed for Gower's dissimilarity coefficient only, see details.

**Details**

The `pcoa.calc` function performs principal coordinates analysis using the `cmdscale` function from package `stats`. Principal coordinates analysis estimates coordinates for a set of objects in a space. Distances among objects is approximation of the dissimilarities, based on any similarity or distance coefficient.

Various measures of distance between the observations (rows) are applicable: (1) coefficients of distance for quantitative and binary characters: "Euclidean", "Manhattan", "Minkowski"; (2) similarity coefficients for binary characters: "Jaccard" and simple matching ("simpleMatching"); (3) coefficient for mixed data: ("Gower").

The Gower's dissimilarity coefficient can handle different types of variables. Characters have to be divided into four categories: (1) quantitative characters, (2) categorical ordinal characters, (3) categorical nominal (multistate) characters, and (4) binary characters. All characters are considered to be quantitative characters unless otherwise specified. Other types of characters have to be explicitly specified. To mark characters as ordinal, nominal, or binary, enumerate them by names using `ordinalChs`, `nominalChs`, and `binaryChs` arguments, respectively.
Value

an object of class `pcoadata` with the following elements:

- **objects**
  - **ID**
    - IDs of each row of scores object.
  - **Population**
    - population membership of each row of scores object.
  - **Taxon**
    - taxon membership of each row of scores object.
  - **scores**
    - ordination scores of cases (objects, OTUs).

- **eigenValues**
  - eigenvalues of principal coordinates.

- **eigenvaluesAsPercent**
  - eigenvalues as percent, percentage of their total sum.

- **cumulativePercentageOfEigenvalues**
  - cumulative percentage of eigenvalues.

- **groupMeans**
  - data.frame containing the means for the taxa.

- **distMethod**
  - used distance measure.

- **rank**
  - number of positive eigenvalues.

Examples

data(centaurea)
pcoRes = pcoa.calc(centaurea, distMethod = "Manhattan")
summary(pcoRes)
plotPoints(pcoRes, axes = c(1,2), col = c("red", "green", "blue", "black"),
pch = c(20,17,8,21), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")

# using Gower's method
data = list(
  ID = as.factor(c("id1","id2","id3","id4","id5","id6")),
  Population = as.factor(c("Pop1", "Pop1", "Pop2", "Pop2", "Pop3", "Pop3")),
  data = data.frame(
    stemBranching = c(1, 1, 1, 1, 0, 0), # binaryChs
    petalColour = c(1, 1, 2, 3, 3, 3), # nominalChs; 1=white, 2=red, 3=blue
    leaves = c(1,1,1,2,2,3), # nominalChs; 1=simple, 2=palmately compound, 3=pinnately compound
    taste = c(2, 2, 2, 3, 1, 1), # ordinal; 1=hot, 2=hotter, 3=hottest
    stemHeight = c(10, 11, 14, 22, 23, 21), # quantitative
    leafLength = c(8, 7.1, 9.4, 1.2, 2.3, 2.1) ) # quantitative
) attr(data, "class") = "morphodata"
pcoaGower = pcoa.calc(data, distMethod = "Gower", binaryChs = c("stemBranching"),
nominalChs = c("petalColour", "leaves"), ordinalChs = c("taste"))
plot3Dpoints(pcoaGower, axes = c(1,2), col = c("red","green"),
    pch = c(20,17), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")

__pcoadata__  

Class pcoadata

**Description**

The pcoadata class is designed for storing results of principal coordinates analysis (PCoA).

**Format**

Class pcoadata.

- **objects ID** IDs of each row of scores object.
- **Population** population membership of each row of scores object.
- **Taxon** taxon membership of each row of scores object.
- **scores** ordination scores of cases (objects, OTUs).
- **eigenValues** eigenvalues of principal coordinates.
- **eigenvaluesAsPercent** eigenvalues as percent, percentage of their total sum.
- **cumulativePercentageOfEigenvalues** cumulative percentage of eigenvalues.
- **groupMeans** data.frame containing the means for the taxa.
- **distMethod** used distance measure.
- **rank** number of positive eigenvalues.

__plot3Dpoints__  

The Default Scatterplot 3D Function

**Description**

A generic function for plotting ordination scores stored in padata, pcoadata, nmdsdata, and cdadata objects.

**Usage**

plot3Dpoints(result, axes = c(1,2,3), xlab = NULL, ylab = NULL, zlab = NULL,
    pch = 16, col = "black", pt.bg = "white", phi = 10, theta = 2,
    ticktype = "detailed", bty = "u", type = "p", labels = FALSE,
    legend = FALSE, legend.pos = "topright", ncol = 1, ...)
Arguments

result an object of class `pcadata`, `pcoadata`, `nmmdsdata`, or `cdadata`.

axes x, y, z axes of plot.

xlab, ylab, zlab a title of the respective axes.

pch a vector of plotting characters or symbols, see `points`.

col the colours for points. Multiple colours can be specified so that each taxon can be given its own colour. If there are fewer colours than taxa, they are recycled in the standard fashion.

pt.bg the background colours for points. Multiple colours can be specified, as above.

theta, phi the angles defining the viewing direction. Theta gives the azimuthal direction and phi the colatitude, see `persp`.

ticktype character: "simple" draws just an arrow parallel to the axis to indicate direction of increase; "detailed" draws normal ticks as per 2D plots.

bty the type of the box. One of "g", "b2", "b1", "f", "u" can be specified.

type the type of plot points, "p" for points, or "h" for vertical spikes.

labels logical, if TRUE, point’s labels are displayed.

legend logical, if TRUE, legend is displayed. Only restricted number of legend parameters are supported. For more precise legend plotting, use `plotAddLegend` directly.

legend.pos a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", and "center", to be used to position the legend.

ncol the number of columns in which to set the legend items.

... further arguments to be passed to `plot3D`, `persp`, `par`.

Value

None. Used for its side effect of producing a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plot3Dpoints(pcaRes, col = c("red", "green", "blue", "black"), pch = c(20, 17, 8, 21),
pt.bg = "orange")
plotAddEllipses

Add Prediction Ellipses to a Plot

Description

This function draws prediction ellipses around taxa.

Usage

plotAddEllipses(result, axes = c(1,2), probability = 0.95, col = "black",
    type = "l", lty = 1, lwd = 1, ...)

Arguments

result
result of pca.calc, pcoa.calc, nmds.calc, or cda.calc, has to be plotted at first.

axes
x, y axes of plot.

probability
probability, that a new independent observation from the same population will fall in that ellipse.

col
the colours for labels.

type
character indicating the type of plotting, for details, see plot: "p" for points, "l" for lines, "b" for both points and lines, "c" for empty points joined by lines, "o" for overplotted points and lines, "s" and "S" for stair steps and "h" for histogram-like vertical lines.

lty
the line type. Line types can either be specified as one of following types: 0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash.

lwd
the line width.

...

Further arguments to be passed to lines or other graphical parameters in par.

Details

Prediction ellipses with given probability define the regions where will fall any new independent observation from the respective taxa. The prediction ellipses are quantified using covariance matrices of taxa scores and chi-squared distribution with two degrees of freedom (Friendly et al. 2013).

Value

None. Used for its side effect of adding elements to a plot.

References

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotPoints(pcaRes, col = c(rgb(255, 0, 0, max = 255, alpha = 150), # red
rgb(0, 255, 0, max = 255, alpha = 150), # green
rgb(0, 0, 255, max = 255, alpha = 150), # blue
rgb(0, 0, 0, max = 255, alpha = 150)), # black
legend = FALSE, xlim = c(-5, 7.5), ylim = c(-5, 5.5))

plotAddLegend(pcaRes, col = c("red", "green", "blue", "black"), ncol = 2)

plotAddEllipses(pcaRes, col = c("red", "green", "blue", "black"), lwd = 3)

plotAddLabels.characters

Add Labels to a Plot

Description

This is a generic function for drawing labels to the character arrows of \texttt{pcadata} and \texttt{cdadata} objects.

Usage

\begin{verbatim}
plotAddLabels.characters(result, labels = characters(result), include = TRUE,
axes = c(1,2), pos = NULL, offset = 0.5, cex = 0.7, col = NULL, ...)
\end{verbatim}

Arguments

- \texttt{result}: an object of class \texttt{pcadata} or \texttt{cdadata}.
- \texttt{labels}: a vector of label names, which should be included / excluded from plotting, see \texttt{include}.
- \texttt{include}: logical, specify if labels in \texttt{labels} argument should be plotted or excluded from plotting.
- \texttt{axes}: x, y axes of plot.
- \texttt{pos}: a position specifier for the text. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the point.
- \texttt{offset}: when \texttt{pos} is specified, this value controls the distance (offset) of the text label from the point in fractions of a character width.
- \texttt{cex}: character expansion factor for text.
- \texttt{col}: the colours for labels.
- \texttt{...}: further arguments to be passed to \texttt{text} or other graphical parameters in \texttt{par}.  


Value

None. Used for its side effect of adding elements to a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotCharacters(pcaRes, labels = FALSE)
plotAddLabels.characters(pcaRes, labels = c("MW", "IW", "SFT", "SF", "LW"), pos = 2, cex = 1)
plotAddLabels.characters(pcaRes, labels = c("LLW", "ILW", "LBA"), pos = 4, cex = 1)
plotAddLabels.characters(pcaRes, labels = c("ML", "IV", "MLW"), pos = 1, cex = 1)

plotAddLabels.points  Add Labels to a Plot

Description

This is a generic function for drawing labels to the data points of pcdatadata, pcoadata, nmdsdata, and cdadata objects.

Usage

plotAddLabels.points(result, labels = result$objects$ID, include = TRUE,
                      axes = c(1,2), pos = NULL, offset = 0.5, cex = 1, col = NULL, ...)

Arguments

result  result of pca.calc, pcoa.calc, nmds.calc, or cda.calc, has to by plotted at first.
labels  a vector of label names, which should be included / excluded from plotting, see include.
include logical, specify if labels in labels argument should be plotted or excluded from plotting.
axes    x, y axes of plot.
pos      a position specifier for the text. Values of 1, 2, 3 and 4, respectively indicate
          positions below, to the left of, above and to the right of the point.
offset   when pos is specified, this value controls the distance (offset) of the text label
          from the point in fractions of a character width.
cex      character expansion factor for text.
col      the colours for labels.
...      further arguments to be passed to text or other graphical parameters in par.
plotAddLegend

Add Legend to a Plot

Value

None. Used for its side effect of adding elements to a plot.

Examples

data(centaurea)
  centaurea = naMeanSubst(centaurea)
  centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))
  pops = populOTU(centaurea)

  pcaRes = pca.calc(pops)
  plotPoints(pcaRes, col = c("red", "green", "blue", "red"),
             pch = c(20, 17, 8, 21), pt.bg = "orange", legend = FALSE)
  plotAddLabels.points(pcaRes, labels = c("LES", "BUK", "VOL", "OLE1"), include = TRUE)

  plotPoints(pcaRes, col = c("red", "green", "blue", "red"),
             pch = c(20, 17, 8, 21), pt.bg = "orange", legend = FALSE)
  plotAddLabels.points(pcaRes, labels = c("LES", "BUK", "VOL", "OLE1"), include = FALSE)

plotAddLegend

Description

This function can be used to add legend to plot.

Usage

plotAddLegend(result, x = "topright", y = NULL, pch = 16, col = "black",
               pt.bg = "white", pt.cex = cex, pt.lwd = 1, x.intersp = 1,
               y.intersp = 1, box.type = "o", box.lty = "solid", box.lwd = 1,
               box.col = "black", box.bg = "white", cex = 1, ncol = 1, horiz = FALSE, ...)

Arguments

result  result of `pca.calc`, `pcoa.calc`, `nmds.calc`, or `cda.calc`, has to by plotted at first.

x, y    the x and y coordinates or a single keyword from the list "bottomright",
        "bottom", "bottomleft", "left", "topleft", "top", "topright", "right",
        and "center", to be used to position the legend.

pch     the plotting symbols of points appearing in the legend.

col     the colours of points appearing in the legend.

pt.bg   the background colour for the `points`, corresponding to its argument bg.

pt.cex  character expansion factor for the points.
plotAddSpiders

Add Spiders to a Plot

Description

This function connects taxa’s points with its centroids, thus forms a “spider” diagram.

Usage

plotAddSpiders(result, axes = c(1,2), col = "black", lty = 1, lwd = 1, ...)

pt.lwd
the line width for the points.

x.intersp, y.intersp
character interspacing factor for horizontal (x) and vertical (y) line distances.

box.type
the type of box to be drawn around the legend. The applicable values are “o” (the default) and “n”.

box.lty, box.lwd, box.col, box.bg
the line type, width colour and background colour for the legend box (if box.type = "o").

cex
character expansion factor for text.

ncol
the number of columns in which to set the legend item.

horiz
logical; if TRUE, set the legend horizontally rather than vertically (specifying horiz overrides the ncol specification).

...
进一步 arguments to be passed to legend or other graphical parameters in par.

Value

None. Used for its side effect of adding elements to a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotPoints(pcaRes, col = c("red", "green", "blue", "red"),
pch = c(20, 17, 8, 21), pt.bg = "orange", legend = FALSE)

plotAddLegend(pcaRes, x = "bottomright", col = c("red", "green", "blue", "red"),
pch = c(20, 17, 8, 21), pt.bg = "orange", ncol = 2)
Arguments

result
axes
col
lty
lwd
...

Value

None. Used for its side effect of adding elements to a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotPoints(pcaRes, col = c(rgb(255, 0, 0, max = 255, alpha = 150), # red
                           rgb(0, 255, 0, max = 255, alpha = 150), # green
                           rgb(0, 0, 255, max = 255, alpha = 150), # blue
                           rgb(0, 0, 0, max = 255, alpha = 150)), # black
           legend = FALSE, xlim = c(-5, 7.5), ylim = c(-5, 5.5))

plotAddLegend(pcaRes, col = c("red", "green", "blue", "black"), ncol = 2)
plotAddSpiders(pcaRes, col = c("red", "green", "blue", "black"))

plotPoints(pcaRes, col = c("red", "green", "blue", "black"), legend = TRUE, cex = 0.4)
plotAddSpiders(pcaRes, col = c(rgb(255, 0, 0, max = 255, alpha = 150), # red
                                rgb(0, 255, 0, max = 255, alpha = 150), # green
                                rgb(0, 0, 255, max = 255, alpha = 150), # blue
                                rgb(0, 0, 0, max = 255, alpha = 150))) # black

plotCharacters

Draws Character’s Contribution as Arrows

Description

The character’s contribution to ordination axes are visualised as arrows.
plotCharacters

Usage

plotCharacters(result, axes = c(1, 2), xlab = NULL, ylab = NULL, 
main = NULL, xlim = NULL, ylim = NULL, col = "red", length = 0.1, 
angle = 15, labels = TRUE, cex = 0.7, ...)

Arguments

result an object of class pcadata or cdadata.
axes x, y axes of plot.
xlab, ylab a title of the respective axes.
xlim, ylim numeric vectors of length 2, giving the x and y coordinates ranges.
main a main title for the plot.
col the colour for arrows.
length length of the edges of the arrow head (in inches).
angle angle from the shaft of the arrow to the edge of the arrow head.
labels logical, if TRUE, labels are displayed. Only restricted number of parameters are supported. For more precise labels plotting, use plotAddLabels.characters directly.
cex character expansion factor for labels.
... further arguments to be passed to arrows or other graphical parameters in par.

Details

The distribution of samples in ordination space is driven by morphological characters. Each character has its own contribution to ordination axes. These contributions are visualised as arrows. The direction and length of the arrows characterize the impact of the morphological characters on the separation of objects along a given axis. This information is stored in eigenvectors or total canonical structure coefficients for principal component analysis of canonical discriminant analysis, respectively.

The plotCharacters method is not applicable to results of the principal coordinates analysis (pcoa.calc) and non-metric multidimensional scaling (nmds.calc) analyses, as the influence of original characters on new axes can not be directly derived, and variation explained by individual axes is unknown.

Value

None. Used for its side effect of producing a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotCharacters(pcaRes)
plotPoints

The Default Scatterplot Function

Description

A generic function for plotting ordination scores stored in `pcadata`, `pcoadata`, `nmdsdata`, and `cdadata` objects.

Usage

```r
plotPoints(result, axes = c(1,2), xlab = NULL, ylab = NULL,
            pch = 16, col = "black", pt.bg = "white", breaks = 1,
            ylim = NULL, xlim = NULL, labels = FALSE, legend = FALSE,
            legend.pos = "topright", ncol = 1, ...)
```

Arguments

- `result`: an object of class `pcadata`, `pcoadata`, `nmdsdata`, or `cdadata`.
- `axes`: x, y axes of plot.
- `xlab`, `ylab`: a title of the respective axes.
- `pch`: a vector of plotting characters or symbols: see `points`.
- `col`: the colours for points. Multiple colours can be specified so that each taxon can be given its own colour. If there are fewer colours than taxa, they are recycled in the standard fashion.
- `pt.bg`: the background colours for points. Multiple colours can be specified, as above.
- `breaks`: a numeric, giving the width of one histogram bar.
- `xlim`, `ylim`: the range of x and y axes.
- `labels`: logical, if `TRUE`, labels are displayed. Only restricted number of parameters are supported. For more precise labels plotting, use `plotAddLabels.points` directly.
- `legend`: logical, if `TRUE`, legend is displayed. Only restricted number of legend parameters are supported. For more precise legend plotting, use `plotAddLegend` directly.
- `legend.pos`: a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", and "center", to be used to position the legend.
- `ncol`: the number of columns in which to set the legend items.
- `...`: further arguments to be passed to `plot.default` or other graphical parameters in `par`.

Details

This generic method holds separate implementations of plotting points for `pcadata`, `pcoadata`, `nmdsdata`, and `cdadata` objects. If only one axis exists, sample scores are displayed as a histogram.
Value

None. Used for its side effect of producing a plot.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))

pcaRes = pca.calc(centaurea)

plotPoints(pcaRes, axes = c(1,2), col = c("red", "green", "blue", "red"),
          pch = c(20, 17, 8, 21), pt.bg = "orange", legend = TRUE, legend.pos = "bottomright")

plotPoints(pcaRes, main = "My PCA plot", frame.plot = FALSE, cex = 0.8)

cdaRes = cda.calc(centaurea)

plotPoints(cdaRes, col = c("red", "green", "blue", "red"),
          pch = c(20, 17, 8, 21), pt.bg = "orange", legend = TRUE)

populOTU

Description

This function calculates the average value for each character in each population, with the pairwise deletion of missing data.

Usage

populOTU(object)

Arguments

object an object of class morphodata.

Details

This function returns morphodata object, where each population is used as the operational taxonomic unit (OTUs), thus is represented by single “individual” (row) with average values for each character. Note that when using populations as OTUs, they are handled with the same weight in all analyses (disregarding population size, within-population variation, etc.)
**Value**

an object of class `morphodata` with the following elements:

- **ID**: IDs of each row of data object.
- **Population**: population membership of each row of data object.
- **Taxon**: taxon membership of each row of data object.
- **data**: data frame of individuals (rows) and values of measured morphological characters (columns).

**Examples**

```r
data(centaurea)
pops = populOTU(centaurea)
```

---

**qqnormCharacter**  
*Quantile-Quantile Plots*

**Description**

Q-Q plots are produced for the level of taxa/groups, to displays a deviation of morphological characters of each taxon from the normal distribution (line).

**Usage**

```r
qqnormCharacter(object, character, taxon = levels(object$Taxon), main = NULL, ...)
qqnormAll(object, folderName = "qqnormPlots", taxon = levels(object$Taxon),
          main = NULL, width = 480, height = 480, units = "px", ...)
```

**Arguments**

- **object**: an object of class `morphodata`.
- **character**: a morphological character used to plot Q-Q plot.
- **folderName**: folder to save produced Q-Q plots.
- **taxon**: taxa which should be plotted, default is to plot all of the taxa.
- **main**: main title for the plot.
- **width**: the width of the figure.
- **height**: the height of the figure.
- **units**: the units in which height and width are given. Can be "px" (pixels, the default), "in" (inches), "cm" or "mm".
- **...**: further arguments to be passed to `qqnorm` or graphical parameters `par`.
Value

None. Used for its side effect of producing a plot(s).

Examples

```r
data(centaurea)

qqnormCharacter(centaurea, character = "SF")

## Not run: qqnormAll(centaurea, folderName = "/qqnormPlots")
```

---

read.morphodata  Data Input and Description

Description

This function imports data and produces a `morphodata` object from it.

Usage

```r
read.morphodata(file, dec = ".", sep = "\t", ...)

## S3 method for class 'morphodata'
samples(object)

populations(object)

 taxa(object)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>the file which the data are to be read from or a <code>connection</code> for reading (e.g., &quot;clipboard&quot;).</td>
</tr>
<tr>
<td>dec</td>
<td>the character used for decimal points.</td>
</tr>
<tr>
<td>sep</td>
<td>the column separator character.</td>
</tr>
<tr>
<td>object</td>
<td>an object of class <code>morphodata</code>.</td>
</tr>
<tr>
<td>...</td>
<td>further arguments to be passed to <code>read.table</code> function.</td>
</tr>
</tbody>
</table>

Details

The function expects the following data structure:

1. the first row contains variable names;
2. the following rows contains individuals, single individual per row;
3. the first three columns include unique identifiers for individuals, populations and taxa/groups, respectively. Columns have to be named as “ID”, “Population” and “Taxon”;
starting from the fourth column, any number of quantitative or binary morphological characters may be recorded. Any variable names can be used (avoiding spaces and special characters);

If there are missing values in the data, they must be represented as empty cells or by the text NA, not zero, space or any other character. 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").

Value

an object of class `morphodata` with the following elements:

- **ID**: IDs of each row of data object.
- **Population**: population membership of each row of data object.
- **Taxon**: taxon membership of each row of data object.
- **data**: data.frame of individuals (rows) and values of measured morphological characters (columns).

See Also

`characters`

Examples

data = read.morphodata(file = system.file("extdata", "centaurea.txt", package = "MorphoTools2"), dec = ".", sep = "\t")

# Not run: data = read.morphodata(file = "morphodata.txt", dec = ".", sep = "\t")

# Not run: data = read.morphodata("clipboard")

summary(data)
samples(data)
populations(data)
taxa(data)
Usage

removeTaxon(object, taxonName)
removePopulation(object, populationName)
removeSample(object, sampleName = NULL, missingPercentage = NA)
removeCharacter(object, characterName)

Arguments

object object of class \texttt{morphodata}.
taxonName vector of taxa to be removed.
populationName vector of populations to be removed.
sampleName vector of samples to be removed.
missingPercentage a numeric, samples holding more missing data than specified by missingPercentage will be removed.
characterName vector of characters to be removed.

Value

an object of class \texttt{morphodata} with the following elements:

\begin{itemize}
\item \textbf{ID} IDs of each row of data object.
\item \textbf{Population} population membership of each row of data object.
\item \textbf{Taxon} taxon membership of each row of data object.
\item \textbf{data} \texttt{data.frame} of individuals (rows) and values of measured morphological characters (columns).
\end{itemize}

Examples

data(centaurea)

centaurea.3tax = removeTaxon(centaurea, "hybr")
centaurea.PsSt = removeTaxon(centaurea, c("ph", "hybr"))

centaurea.short = removePopulation(centaurea, c("LIP", "PREL"))

centaurea.NA_0.1 = removeSample(centaurea, missingPercentage = 0.1)

centaurea.short = removeCharacter(centaurea, "LL")
**shapiroWilkTest**

**Shapiro-Wilk Normality Test**

**Description**
Calculates the Shapiro-Wilk normality test of characters for taxa.

**Usage**

```r
shapiroWilkTest(object, p.value = 0.05)
```

**Arguments**

- `object`: an object of class `morphodata`.
- `p.value`: a number or `NA` are acceptable. In the case of number, the output will be formatted as "normally distributed" or "NOT normally distributed". In the case of `NA`, exact p-values will be returned.

**Value**

A `data.frame`, storing results of Shapiro-Wilk normality test.

**Examples**

```r
data(centaurea)

sW = shapiroWilkTest(centaurea)

## Not run: exportRes(sW, file = "sW_test.txt")

sW = shapiroWilkTest(centaurea, p.value = NA)

## Not run: exportRes(sW, file = "sW_test.txt")
```

---

**stepdisc.calc**

**Stepwise Discriminant Analysis**

**Description**

This function perform stepwise discriminant analysis.

**Usage**

```r
stepdisc.calc(object, FToEnter = 0.15, FToStay = 0.15)
```
Arguments

object an object of class `morphodata`.
FToEnter significance levels for a variable to enter the subset.
FToStay significance levels for a variable to stay in the subset.

Details

The `stepdisc.calc` function performs a stepwise discriminant analysis to select the “best” subset of the quantitative variables for use in discriminating among the groups (taxa).

Value

None. Used for its side effect.

Examples

data(centaurea)
centaurea = naMeanSubst(centaurea)
centaurea = removePopulation(centaurea, populationName = c("LIP", "PREL"))
stepdisc.calc(centaurea)

summary Object Summaries

Description

summary methods for classes `morphodata, pcdatal, pcoadata, nmdsdata, cdadata`, and `classifdata`.

Usage

```r
## S3 method for class 'morphodata'
summary(object, ...)

## S3 method for class 'pcadata'
summary(object, ...)

## S3 method for class 'pcoadata'
summary(object, ...)

## S3 method for class 'nmdsdata'
summary(object, ...)

## S3 method for class 'cdadata'
summary(object, ...)

## S3 method for class 'classifdata'
summary(object, ...)
```
transformCharacter

Arguments

- **object**: an object of class `morphodata`, `pcadata`, `pcoadata`, `nmdsdata`, `cdadata`, or `classifdata`.
- ... additional arguments affecting the summary produced.

Value

None. Used for its side effect.

____________________________

**transformCharacter**  
*Transformation of Character*

Description

This function transforms morphological characters by applying another function passed in the argument.

Usage

```
transformCharacter(object, character, FUN, newName = NULL)
```

Arguments

- **object**: an object of class `morphodata`.
- **character**: a morphological character that should be transformed.
- **FUN**: the transforming function to be applied to character.
- **newName**: a name to rename the original character. If NULL, the name of the transformed character remains the same.

Details

Transformation is applied to characters to improve their distribution (to become normally distributed or at least to achieve lesser deviation from normality). The **FUN** argument takes any function, able to accept as input any value of the character specified by **character** argument.

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 be only positive numbers. The arcsine transformation is applicable for proportions and percentages (for values ranging from 0 to 1).

Value

an object of class `morphodata` with the following elements:

- **ID**: IDs of each row of data object.
- **Population**: population membership of each row of data object.
- **Taxon**: taxon membership of each row of data object.
- **data**: data.frame of individuals (rows) and values of measured morphological characters (columns).
Examples

data(centaurea)

# For a right-skewed (positive) distribution can be used:
# Logarithmic transformation
cTransf = transformCharacter(centaurea, character = "SF", FUN = function(x) log(x+1))
cTransf = transformCharacter(centaurea, character = "SF", FUN = function(x) log10(x+1))
# Square root transformation
cTransf = transformCharacter(centaurea, character = "SF", FUN = function(x) sqrt(x))
# Cube root transformation
cTransf = transformCharacter(centaurea, character = "SF", FUN = function(x) x^(1/3))
# Arcsine transformation
cTransf = transformCharacter(centaurea, character = "SF", FUN = function(x) asin(sqrt(x)))

# For a left-skewed (negative) distribution can be used:
# Logarithmic transformation
cTransf = transformCharacter(centaurea, character="SF", FUN=function(x) log((max(x)+1)-x))
cTransf = transformCharacter(centaurea, character="SF", FUN=function(x) log10((max(x)+1)-x))
# Square root transformation
cTransf = transformCharacter(centaurea, character="SF", FUN=function(x) sqrt((max(x)+1)-x))
# Cube root transformation
cTransf = transformCharacter(centaurea, character="SF", FUN=function(x) ((max(x)+1)-x)^(1/3))
# Arcsine transformation
cTransf = transformCharacter(centaurea, character="SF", FUN=function(x) asin(sqrt((max(x))-x)))

viewMorphodata

Invoke a Data Viewer

Description

Invoke a spreadsheet-style data viewer on a data stored in morphodata class.

Usage

viewMorphodata(object)

Arguments

object an object of class morphodata.

Value

None. Used for its side effect.
Examples

```
data(centaurea)

## Not run: viewMorphodata(centaurea)
```
Index

* datasets
  centaurea, 7
arrows, 40
boxMTest, 3
boxplot, 4, 5
boxplotAll (boxplotCharacter), 4
boxplotCharacter, 4
bxp, 4
cda.calc, 5, 34, 36, 37, 39
cdadat, 6, 7, 8, 32, 33, 35, 36, 40, 41, 48, 49
centaurea, 7
characters, 8, 45
classif.knn, 10, 12, 13, 23
classif.knn (classif.lda), 9
classif.lda, 9, 10, 12, 13, 23
classif.matrix, 10, 11, 13
classif.qda, 10, 12, 13, 23
classif.qda (classif.lda), 9
classifdata, 10, 11, 12, 13, 19, 48, 49
classifSample.knn, 23
classifSample.knn (classifSample.lda), 12
classifSample.lda, 10, 12, 23
classifSample.qda, 23
classifSample.qda (classifSample.lda), 12
clust, 14
cmdscale, 30
connection, 18, 44
cormat, 16
cormatSignifTest, 16
cormatSignifTest (cormat), 16
descrAll (descrTaxon), 17
descrPopulation (descrTaxon), 17
descrTaxon, 17
exportRes, 16, 18
hclust, 14
head.classifdata (head.morphodata), 19
head.morphodata, 19
hist, 20
histAll (histCharacter), 20
histCharacter, 20
keepCharacter (keepTaxon), 21
keepPopulation (keepTaxon), 21
keepSample (keepTaxon), 21
keepTaxon, 21
knn.select, 10, 13, 22
legend, 38
lines, 34, 39
missingCharactersTable, 23
missingSamplesTable, 24
monoMDS, 26
morphodata, 3–5, 8, 9, 12, 14, 16, 17, 19–24, 24, 25, 26, 28, 30, 42–50
naMeanSubst, 25
nmds.calc, 26, 34, 36, 37, 39, 40
nmdsdata, 27, 28, 32, 33, 36, 41, 48, 49
par, 20, 33–36, 38–41, 43
pca.calc, 28, 34, 36, 37, 39
pcadata, 8, 28, 29, 32, 33, 35, 36, 40, 41, 48, 49
pcoa.calc, 30, 34, 36, 37, 39, 40
pcoadat, 31, 32, 32, 33, 36, 41, 48, 49
persp, 33
plot, 34
plot.default, 41
plot3D, 33
plot3Dpoints, 32
plotAddEllipses, 34
plotAddLabels.characters, 35, 40
plotAddLabels.points, 36, 41
plotAddLegend, 33, 37, 41
INDEX

plotAddSpiders, 38
plotCharacters, 39
plotPoints, 41
points, 33, 37, 41
populations (read.morphodata), 44
populOTU, 42
princomp, 28

qqnorm, 43
qqnormAll (qqnormCharacter), 43
qqnormCharacter, 43

read.morphodata, 44
read.table, 44
removeCharacter (removeTaxon), 45
removePopulation (removeTaxon), 45
removeSample (removeTaxon), 45
removeTaxon, 45

samples (read.morphodata), 44
shapiroWilkTest, 47
stats, 30
stepdisc.calc, 47
summary, 48

tail.classifdata (head.morphodata), 19
tail.morphodata (head.morphodata), 19
taxa (read.morphodata), 44
text, 35, 36
transformCharacter, 49

vegan, 26
viewMorphodata, 50