Package ‘candisc’

November 25, 2025

Type Package

Title Visualizing Generalized Canonical Discriminant and Canonical Correlation Analysis

Version 1.1.0

Date 2025-11-18

Depends R (>= 4.1.0), heplots (>= 0.8-6)

Imports car, graphics, stats, insight, dplyr, ggplot2, MASS

Suggests rgl, carData, corrplot, knitr, rmarkdown, rpart, rpart.plot, klaR, lifecycle, tidyr, testthat (>= 3.0.0), RColorBrewer, tinytable

LazyLoad yes

LazyData yes

Maintainer Michael Friendly `<friendly@yorku.ca>`

Language en-US

Description Functions for computing and visualizing
generalized canonical discriminant analyses and canonical correlation analysis
for a multivariate linear model.
Traditional canonical discriminant analysis is restricted to a one-way 'MANOVA'
design and is equivalent to canonical correlation analysis between a set of quantitative
response variables and a set of dummy variables coded from the factor variable.
The 'candisc' package generalizes this to higher-way 'MANOVA' designs
for all factors in a multivariate linear model,
computing canonical scores and vectors for each term. The graphic functions provide low-rank (1D, 2D, 3D)
visualizations of terms in an 'mlm' via the 'plot.candisc' and 'heplot.candisc' methods. Related plots are
now provided for canonical correlation analysis when all predictors are quantitative. Methods for
linear discriminant analysis are now included.

License GPL (>= 2)

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/friendly/candisc/,
https://friendly.github.io/candisc/

BugReports https://github.com/friendly/candisc/issues

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Michael Friendly [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3237-0941>),
John Fox [aut] (ORCID: <https://orcid.org/0000-0002-1196-8012>)

Repository CRAN

Date/Publication 2025-11-25 06:10:54 UTC

Contents

`candisc-package` *Visualizing Generalized Canonical Discriminant and Canonical Correlation Analysis*

Description

This package includes functions for computing and visualizing generalized canonical discriminant analyses and canonical correlation analysis for a multivariate linear model. The goal is to provide ways of visualizing such models in a low-dimensional space corresponding to dimensions (linear combinations of the response variables) of maximal relationship to the predictor variables.

Details

Traditional canonical discriminant analysis is restricted to a one-way MANOVA design and is equivalent to canonical correlation analysis between a set of quantitative response variables and a set of dummy variables coded from the factor variable. The `candisc` package generalizes this to multi-way MANOVA designs for all terms in a multivariate linear model (i.e., an `"mlm"` object), computing canonical scores and vectors for each term (giving a `"candiscList"` object).

The graphic functions are designed to provide low-rank (1D, 2D, 3D) visualizations of terms in a `mlm` via the `plot.candisc()` method, and the HE plot `heplot.candisc()` and `heplot3d.candisc()` methods. For `mlm`s with more than a few response variables, these methods often provide a much simpler interpretation of the nature of effects in canonical space than heplots for pairs of responses or an HE plot matrix of all responses in variable space.

Analogously, a multivariate linear (regression) model with quantitative predictors can also be represented in a reduced-rank space by means of a canonical correlation transformation of the Y and X variables to uncorrelated canonical variates, Ycan and Xcan. Computation for this analysis is provided by `cancor()` and related methods. Visualization of these results in canonical space are provided by the `plot.cancor()`, `heplot.cancor()` and `heplot3d.cancor()` methods.

These relations among response variables in linear models can also be useful for “effect ordering” (Friendly & Kwan (2003) for *variables* in other multivariate data displays to make the displayed relationships more coherent. The function `varOrder()` implements a collection of these methods.

A new vignette, `vignette("diabetes", package="candisc")`, illustrates some of these methods. A more comprehensive collection of examples is contained in the vignette for the **heplots** package,

`vignette("HE-examples", package="heplots")`.

The organization of functions in this package and the **heplots** package may change in a later version.

Author(s)

Michael Friendly and John Fox

Maintainer: Michael Friendly friendly@yorku.ca

References

Friendly, M. (2007). HE plots for Multivariate General Linear Models. *Journal of Computational and Graphical Statistics*, **16**(2) 421–444. `http://datavis.ca/papers/jcgs-heplots.pdf`, doi:10.1198/106186007X208407.

Friendly, M. & Kwan, E. (2003). Effect Ordering for Data Displays, *Computational Statistics and Data Analysis*, **43**, 509-539. doi:10.1016/S01679473(02)002906

Friendly, M. & Sigal, M. (2014). Recent Advances in Visualizing Multivariate Linear Models. *Revista Colombiana de Estadistica* , **37**(2), 261-283. doi:10.15446/rce.v37n2spe.47934.

Friendly, M. & Sigal, M. (2017). Graphical Methods for Multivariate Linear Models in Psychological Research: An R Tutorial, *The Quantitative Methods for Psychology*, 13 (1), 20-45. doi:10.20982/tqmp.13.1.p020.

Gittins, R. (1985). *Canonical Analysis: A Review with Applications in Ecology*, Berlin: Springer.

See Also

`heplots::heplot()` for details about HE plots.

`candisc()`, `cancor()` for details about canonical discriminant analysis and canonical correlation analysis.

`cancor` *Canonical Correlation Analysis*

Description

The function `cancor` generalizes and regularizes computation for canonical correlation analysis in a way conducive to visualization using methods in the **heplots** package.

The package provides the following display, extractor and plotting methods for `"cancor"` objects

`print(), summary()` Print and summarise the CCA

`coef()` Extract coefficients for X, Y, or both

`scores()` Extract observation scores on the canonical variables

`redundancy()` Redundancy analysis: proportion of variances of the variables in each set (X and Y) accounted for by the variables in the other set through the canonical variates

`plot()` Plot pairs of canonical scores with a data ellipse and regression line

`heplot()` HE plot of the Y canonical variables showing effects of the X variables and projections of the Y variables in this space.

As well, the function provides for observation weights, which may be useful in some situations, as well as providing a basis for robust methods in which potential outliers can be down-weighted.

Usage

```
cancor(x, ...)

## S3 method for class 'formula'
cancor(formula, data, subset, weights, na.rm = TRUE, method = "gensvd", ...)

## Default S3 method:
cancor(
  x,
  y,
  weights,
  X.names = colnames(x),
  Y.names = colnames(y),
  row.names = rownames(x),
  xcenter = TRUE,
  ycenter = TRUE,
  xscale = FALSE,
  yscale = FALSE,
  ndim = min(p, q),
  set.names = c("X", "Y"),
  prefix = c("Xcan", "Ycan"),
  na.rm = TRUE,
  use = if (na.rm) "complete" else "pairwise",
  method = "gensvd",
  ...
)

## S3 method for class 'cancor'
print(x, digits = max(getOption("digits") - 2, 3), ...)

## S3 method for class 'cancor'
summary(object, digits = max(getOption("digits") - 2, 3), ...)

## S3 method for class 'cancor'
scores(x, type = c("x", "y", "both", "list", "data.frame"), ...)

## S3 method for class 'cancor'
coef(object, type = c("x", "y", "both", "list"), standardize = FALSE, ...)
```

Arguments

`x`	Varies depending on method. For the `cancor.default` method, this should be a matrix or data.frame whose columns contain the X variables
`...`	Other arguments, passed to methods
`formula`	A two-sided formula of the form `cbind(y1, y2, y3, \dots) ~ x1 + x2 + x3 + \dots`
`data`	The data.frame within which the formula is evaluated

`subset`	an optional vector specifying a subset of observations to be used in the calculations.
`weights`	Observation weights. If supplied, this must be a vector of length equal to the number of observations in X and Y, typically within (0,1). In that case, the variance-covariance matrices are computed using stats::cov.wt, and the number of observations is taken as the number of non-zero weights.
`na.rm`	logical, determining whether observations with missing cases are excluded in the computation of the variance matrix of (X,Y). See Notes for details on missing data.
`method`	the method to be used for calculation; currently only `method = "gensvd"` is supported;
`y`	For the `cancor.default` method, a matrix or data.frame whose columns contain the Y variables
`X.names`, `Y.names`	Character vectors of names for the X and Y variables.
`row.names`	Observation names in `x`, `y`
`xcenter`, `ycenter`	logical. Center the X, Y variables? (not yet implemented)
`xscale`, `yscale`	logical. Scale the X, Y variables to unit variance? (not yet implemented)
`ndim`	Number of canonical dimensions to retain in the result, for scores, coefficients, etc.
`set.names`	A vector of two character strings, giving names for the collections of the X, Y variables.
`prefix`	A vector of two character strings, giving prefixes used to name the X and Y canonical variables, respectively.
`use`	argument passed to `var` determining how missing data are handled. Only the default, `use="complete"` is allowed when observation weights are supplied.
`digits`	Number of digits passed to `print` and `summary` methods
`object`	A `cancor` object for related methods.
`type`	For the `coef` method, the type of coefficients returned, one of `"x"`, `"y"`, `"both"`. For the `scores` method, the same list, or `"data.frame"`, which returns a data.frame containing the X and Y canonical scores.
`standardize`	For the `coef` method, whether coefficients should be standardized by dividing by the standard deviations of the X and Y variables.

Details

Canonical correlation analysis (CCA), as traditionally presented is used to identify and measure the associations between two sets of quantitative variables, X and Y. It is often used in the same situations for which a multivariate multiple regression analysis (MMRA) would be used.

However, CCA is is “symmetric” in that the sets X and Y have equivalent status, and the goal is to find orthogonal linear combinations of each having maximal (canonical) correlations. On the other hand, MMRA is “asymmetric”, in that the Y set is considered as responses, *each one* to be explained by *separate* linear combinations of the Xs.

Let $\mathbf{Y}_{n \times p}$ and $\mathbf{X}_{n \times q}$ be two sets of variables over which CCA is computed. We find canonical coefficients $\mathbf{A}_{p \times k}$ and $\mathbf{B}_{q \times k}, k = \min(p, q)$ such that the canonical variables

$$\mathbf{U}_{n \times k} = \mathbf{YA} \quad \text{and} \quad \mathbf{V}_{n \times k} = \mathbf{XB}$$

have maximal, diagonal correlation structure. That is, the coefficients $\mathbf{A}$ and $\mathbf{B}$ are chosen such that the (canonical) correlations between each pair $r_i = \text{cor}(\mathbf{u}_i, \mathbf{v}_i), i = 1, 2, \ldots, k$ are maximized and all other pairs are uncorrelated, $r_{ij} = \text{cor}(\mathbf{u}_i, \mathbf{v}_j) = 0, i \neq j$. Thus, all correlations between the X and Y variables are channeled through the correlations between the pairs of canonical variates.

For visualization using HE plots, it is most natural to consider plots representing the relations among the canonical variables for the Y variables in terms of a multivariate linear model predicting the Y canonical scores, using either the X variables or the X canonical scores as predictors. Such plots, using `heplot.cancor()` provide a low-rank (1D, 2D, 3D) visualization of the relations between the two sets, and so are useful in cases when there are more than 2 or 3 variables in each of X and Y.

The connection between CCA and HE plots for MMRA models can be developed as follows. CCA can also be viewed as a principal component transformation of the predicted values of one set of variables from a regression on the other set of variables, in the metric of the error covariance matrix.

For example, regress the Y variables on the X variables, giving predicted values $\hat{Y} = X(X'X)^{-1}X'Y$ and residuals $R = Y - \hat{Y}$. The error covariance matrix is $E = R'R/(n-1)$. Choose a transformation Q that orthogonalizes the error covariance matrix to an identity, that is, $(RQ)'(RQ) = Q'R'RQ = (n-1)I$, and apply the same transformation to the predicted values to yield, say, $Z = \hat{Y}Q$. Then, a principal component analysis on the covariance matrix of Z gives eigenvalues of $E^{-1}H$, and so is equivalent to the MMRA analysis of `lm(Y ~ X)` statistically, but visualized here in canonical space.

Value

An object of class `cancorr`, a list with the following components:

`cancor`	Canonical correlations, i.e., the correlations between each canonical variate for the Y variables with the corresponding canonical variate for the X variables.
`names`	Names for various items, a list of 4 components: `X`, `Y`, `row.names`, `set.names`
`ndim`	Number of canonical dimensions extracted, <= `min(p,q)`
`dim`	Problem dimensions, a list of 3 components: `p` (number of X variables), `q` (number of Y variables), `n` (sample size)
`coef`	Canonical coefficients, a list of 2 components: `X`, `Y`
`scores`	Canonical variate scores, a list of 2 components: `X`, `Y`
`scores`	Canonical variate scores, a list of 2 components: `X` Canonical variate scores for the X variables `Y` Canonical variate scores for the Y variables
`X`	The matrix X
`Y`	The matrix Y
`weights`	Observation weights, if supplied, else `NULL`
`structure`	Structure correlations, a list of 4 components: `X.xscores`, `Y.xscores`, `X.yscores`, `Y.yscores`

`structure`	Structure correlations ("loadings"), a list of 4 components: **X.xscores** Structure correlations of the X variables with the Xcan canonical scores **Y.xscores** Structure correlations of the Y variables with the Xcan canonical scores **X.yscores** Structure correlations of the X variables with the Ycan canonical scores **Y.yscores** Structure correlations of the Y variables with the Ycan canonical scores The formula method also returns components `call` and `terms`

Methods (by class)

- `cancor(formula)`: `"formula"` method.
- `cancor(default)`: `"default"` method.

Methods (by generic)

- `print(cancor)`: `print()` method for `"cancor"` objects.
- `summary(cancor)`: `summary()` method for `"cancor"` objects.
- `scores(cancor)`: `scores()` method for `"cancor"` objects.
- `coef(cancor)`: `coef()` method for `"cancor"` objects.

Note

Not all features of CCA are presently implemented: standardized vs. raw scores, more flexible handling of missing data, other plot methods, ...

Author(s)

Michael Friendly

References

Gittins, R. (1985). *Canonical Analysis: A Review with Applications in Ecology*, Berlin: Springer.

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

See Also

Other implementations of CCA: `stats::cancor()` (very basic), `yacca::cca()` in the **yacca** (fairly complete, but very messy return structure), `CCA::cc()` in **CCA** (fairly complete, very messy return structure, no longer maintained).

`redundancy()`, for redundancy analysis; `plot.cancor()`, for enhanced scatterplots of the canonical variates.

`heplot.cancor()` for CCA HE plots and `heplots::heplot()` for generic heplot methods.

`candisc()` for related methods focused on multivariate linear models with one or more factors among the X variables.

Examples

```
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])  # the PA tests
Y <- as.matrix(Rohwer[,3:5])   # the aptitude/ability variables

# visualize the correlation matrix using corrplot()
if (require(corrplot)) {
M <- cor(cbind(X,Y))
corrplot(M, method="ellipse", order="hclust", addrect=2, addCoef.col="black")
}


(cc <- cancor(X, Y, set.names=c("PA", "Ability")))

## Canonical correlation analysis of:
##       5   PA  variables:  n, s, ns, na, ss
##   with        3   Ability  variables:  SAT, PPVT, Raven
##
##     CanR  CanRSQ   Eigen percent    cum                          scree
## 1 0.6703 0.44934 0.81599   77.30  77.30 ******************************
## 2 0.3837 0.14719 0.17260   16.35  93.65 ******
## 3 0.2506 0.06282 0.06704    6.35 100.00 **
##
## Test of H0: The canonical correlations in the
## current row and all that follow are zero
##
##      CanR  WilksL      F df1   df2  p.value
## 1 0.67033 0.44011 3.8961  15 168.8 0.000006
## 2 0.38366 0.79923 1.8379   8 124.0 0.076076
## 3 0.25065 0.93718 1.4078   3  63.0 0.248814


# formula method
cc <- cancor(cbind(SAT, PPVT, Raven) ~  n + s + ns + na + ss, data=Rohwer,
    set.names=c("PA", "Ability"))

# using observation weights
set.seed(12345)
wts <- sample(0:1, size=nrow(Rohwer), replace=TRUE, prob=c(.05, .95))
(ccw <- cancor(X, Y, set.names=c("PA", "Ability"), weights=wts) )

# show correlations of the canonical scores
zapsmall(cor(scores(cc, type="x"), scores(cc, type="y")))

# standardized coefficients
coef(cc, type="both", standardize=TRUE)

# plot canonical scores
plot(cc,
     smooth=TRUE, pch=16, id.n = 3)
text(-2, 1.5, paste("Can R =", round(cc$cancor[1], 3)), pos = 4)
plot(cc, which = 2,
```

```
     smooth=TRUE, pch=16, id.n = 3)
text(-2.2, 2.5, paste("Can R =", round(cc$cancor[2], 3)), pos = 4)

##################
data(schooldata)
##################

#fit the MMreg model
school.mod <- lm(cbind(reading, mathematics, selfesteem) ~
education + occupation + visit + counseling + teacher, data=schooldata)
car::Anova(school.mod)
pairs(school.mod)

# canonical correlation analysis
school.cc <- cancor(cbind(reading, mathematics, selfesteem) ~
education + occupation + visit + counseling + teacher, data=schooldata)
school.cc
heplot(school.cc, xpd=TRUE, scale=0.3)
```

`candisc` *Canonical discriminant analysis*

Description

`candisc` performs a generalized canonical discriminant analysis for one term in a multivariate linear model (i.e., an `mlm` object), computing canonical scores and vectors. It represents a transformation of the original variables into a canonical space of maximal differences for the term, controlling for other model terms.

Usage

```
candisc(mod, ...)

## S3 method for class 'mlm'
candisc(mod, term, type = "2", manova, ndim = rank, ...)

## S3 method for class 'candisc'
print(x, digits = max(getOption("digits") - 2, 3), LRtests = TRUE, ...)

## S3 method for class 'candisc'
summary(
  object,
  means = TRUE,
  scores = FALSE,
  coef = c("std"),
  ndim,
  digits = max(getOption("digits") - 2, 4),
```

```
  ...
)

## S3 method for class 'candisc'
coef(object, type = c("std", "raw", "structure"), ...)

## S3 method for class 'candisc'
scores(x, ...)

## S3 method for class 'candisc'
plot(
  x,
  which = 1:2,
  conf = 0.95,
  col,
  pch,
  scale,
  asp = 1,
  var.col = "blue",
  var.lwd = par("lwd"),
  var.labels,
  var.cex = 1,
  var.pos,
  rev.axes = c(FALSE, FALSE),
  ellipse = FALSE,
  ellipse.prob = 0.68,
  fill.alpha = 0.1,
  prefix = "Can",
  suffix = TRUE,
  titles.1d = c("Canonical scores", "Structure"),
  points.1d = FALSE,
  ...
)
```

Arguments

`mod`	An mlm object, such as computed by `lm()` with a multivariate response
`...`	arguments to be passed down. In particular, `type="n"` can be used with the `plot` method to suppress the display of canonical scores.
`term`	the name of one term from `mod` for which the canonical analysis is performed.
`type`	type of test for the model `term`, one of: "II", "III", "2", or "3"
`manova`	the `Anova.mlm` object corresponding to `mod`. Normally, this is computed internally by `Anova(mod)`
`ndim`	Number of dimensions to store in (or retrieve from, for the `summary` method) the `means`, `structure`, `scores` and `coeffs.*` components. The default is the rank of the H matrix for the hypothesis term.
`digits`	significant digits to print.

`LRtests`	logical; should likelihood ratio tests for the canonical dimensions be printed?
`object, x`	A candisc object
`means`	Logical value used to determine if canonical means are printed
`scores`	Logical value used to determine if canonical scores are printed
`coef`	Type of coefficients printed by the summary method. Any one or more of `"std"`, `"raw"`, or `"structure"`
`which`	A vector of one or two integers, selecting the canonical dimension(s) to plot. If the canonical structure for a `term` has `ndim==1`, or `length(which)==1`, a 1D representation of canonical scores and structure coefficients is produced by the `plot` method. Otherwise, a 2D plot is produced.
`conf`	Confidence coefficient for the confidence circles around canonical means plotted in the `plot` method
`col`	A vector of the unique colors to be used for the levels of the term in the `plot` method, one for each level of the `term`. In this version, you should assign colors and point symbols explicitly, rather than relying on the somewhat arbitrary defaults, based on `grDevices::palette()`.
`pch`	A vector of the unique point symbols to be used for the levels of the term in the `plot` method
`scale`	Scale factor for the variable vectors in canonical space. If not specified, a scale factor is calculated to make the variable vectors approximately fill the plot space.
`asp`	Aspect ratio for the `plot` method. The `asp=1` (the default) assures that the units on the horizontal and vertical axes are the same, so that lengths and angles of the variable vectors are interpretable.
`var.col`	Color used to plot variable vectors
`var.lwd`	Line width used to plot variable vectors
`var.labels`	Optional vector of variable labels to replace variable names in the plots
`var.cex`	Character expansion size for variable labels in the plots
`var.pos`	Position(s) of variable vector labels wrt. the end point. If not specified, the labels are out-justified left and right with respect to the end points.
`rev.axes`	Logical, a vector of `length(which)`. `TRUE` causes the orientation of the canonical scores and structure coefficients to be reversed along a given axis.
`ellipse`	Draw data ellipses for canonical scores?
`ellipse.prob`	Coverage probability for the data ellipses
`fill.alpha`	Transparency value for the color used to fill the ellipses. Use `fill.alpha` to draw the ellipses unfilled.
`prefix`	Prefix used to label the canonical dimensions plotted
`suffix`	Suffix for labels of canonical dimensions. If `suffix=TRUE` the percent of hypothesis (H) variance accounted for by each canonical dimension is added to the axis label.
`titles.1d`	A character vector of length 2, containing titles for the panels used to plot the canonical scores and structure vectors, for the case in which there is only one canonical dimension.
`points.1d`	Logical value for `plot.candisc` when only one canonical dimension.

Details

In typical usage, the `term` should be a factor or interaction corresponding to a multivariate test with 2 or more degrees of freedom for the null hypothesis.

Canonical discriminant analysis is typically carried out in conjunction with a one-way MANOVA design. It represents a linear transformation of the response variables into a canonical space in which (a) each successive canonical variate produces maximal separation among the groups (e.g., maximum univariate F statistics), and (b) all canonical variates are mutually uncorrelated. For a one-way MANOVA with g groups and p responses, there are `dfh` = min(g-1, p) such canonical dimensions, and tests, initially stated by Bartlett (1938) allow one to determine the number of significant canonical dimensions.

Computational details for the one-way case are described in Cooley & Lohnes (1971), and in the *SAS/STAT User's Guide*, "The CANDISC procedure: Computational Details," `http://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_candisc_sect012.htm`.

A generalized canonical discriminant analysis extends this idea to a general multivariate linear model. Analysis of each term in the `mlm` produces a rank df_h H matrix sum of squares and crossproducts matrix that is tested against the rank df_e E matrix by the standard multivariate tests (Wilks' Lambda, Hotelling-Lawley trace, Pillai trace, Roy's maximum root test). For any given term in the `mlm`, the generalized canonical discriminant analysis amounts to a standard discriminant analysis based on the H matrix for that term in relation to the full-model E matrix.

The plot method for candisc objects is typically a 2D plot, similar to a biplot. It shows the canonical scores for the groups defined by the `term` as points and the canonical structure coefficients as vectors from the origin.

If the canonical structure for a `term` has `ndim==1`, or `length(which)==1`, the 1D representation consists of a boxplot of canonical scores and a vector diagram showing the magnitudes of the structure coefficients.

Value

An object of class `candisc` with the following components:

`dfh`	hypothesis degrees of freedom for `term`
`dfe`	error degrees of freedom for the `mlm`
`rank`	number of non-zero eigenvalues of HE^{-1}
`eigenvalues`	eigenvalues of HE^{-1}
`canrsq`	squared canonical correlations
`pct`	A vector containing the percentages of the `canrsq` of their total.
`ndim`	Number of canonical dimensions stored in the `means`, `structure` and `coeffs.*` components
`means`	A data.frame containing the class means for the levels of the factor(s) in the term
`factors`	A data frame containing the levels of the factor(s) in the `term`
`term`	name of the `term`
`terms`	A character vector containing the names of the terms in the `mlm` object
`coeffs.raw`	A matrix containing the raw canonical coefficients

`coeffs.std`	A matrix containing the standardized canonical coefficients
`structure`	A matrix containing the canonical structure coefficients on `ndim` dimensions, i.e., the correlations between the original variates and the canonical scores. These are sometimes referred to as Total Structure Coefficients.
`scores`	A data frame containing the predictors in the `mlm` model and the canonical scores on `ndim` dimensions. These are calculated as `Y \%*\% coeffs.raw`, where `Y` contains the standardized response variables.

Methods (by class)

- `candisc(mlm)`: `"mlm"` method.

Methods (by generic)

- `print(candisc)`: `print()` method for `"candisc"` objects.
- `summary(candisc)`: `summary()` method for `"candisc"` objects.
- `coef(candisc)`: `coef()` method for `"candisc"` objects.
- `scores(candisc)`: `scores()` method for `"candisc"` objects.
- `plot(candisc)`: `"plot"` method.

Author(s)

Michael Friendly and John Fox

References

Bartlett, M. S. (1938). Further aspects of the theory of multiple regression. Proc. Cambridge Philosophical Society **34**, 33-34.

Cooley, W.W. & Lohnes, P.R. (1971). Multivariate Data Analysis, New York: Wiley.

Gittins, R. (1985). Canonical Analysis: A Review with Applications in Ecology, Berlin: Springer.

See Also

`candiscList()`, `heplots::heplot()`, `heplots::heplot3d()`

Examples

```
grass.mod <- lm(cbind(N1,N9,N27,N81,N243) ~ Block + Species, data=Grass)
car::Anova(grass.mod, test="Wilks")

grass.can1 <-candisc(grass.mod, term="Species")
print(grass.can1)

plot(grass.can1, var.lwd = 2)

# library(heplots)
heplot(grass.can1, scale=6, fill=TRUE)
```

```
# iris data
iris.mod <- lm(cbind(Petal.Length, Sepal.Length, Petal.Width, Sepal.Width) ~ Species, data=iris)
iris.can <- candisc(iris.mod, data=iris)
#-- assign colors and symbols corresponding to species
col <- c("red", "brown", "green3")
pch <- 1:3
plot(iris.can, col=col, pch=pch)

heplot(iris.can)

# 1-dim plot
iris.can1 <- candisc(iris.mod, data=iris, ndim=1)
plot(iris.can1)
```

`candiscList` *Canonical discriminant analyses*

Description

`candiscList` performs a generalized canonical discriminant analysis for all terms in a multivariate linear model (i.e., an `mlm` object), computing canonical scores and vectors.

Usage

```
candiscList(mod, ...)

## S3 method for class 'mlm'
candiscList(mod, type = "2", manova, ndim, ...)

## S3 method for class 'candiscList'
print(x, ...)

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

## S3 method for class 'candiscList'
plot(x, term, ask = interactive(), graphics = TRUE, ...)
```

Arguments

`mod`	An mlm object, such as computed by lm() with a multivariate response
`...`	arguments to be passed down.
`type`	type of test for the model `term`, one of: "II", "III", "2", or "3"
`manova`	the `Anova.mlm` object corresponding to `mod`. Normally, this is computed internally by `Anova(mod)`

`ndim`	Number of dimensions to store in the `means`, `structure`, `scores` and `coeffs.*` components. The default is the rank of the H matrix for the hypothesis term.
`object, x`	A candiscList object
`term`	The name of one term to be plotted for the `plot` method. If not specified, one candisc plot is produced for each term in the `mlm` object.
`ask`	If `TRUE` (the default, when running interactively), a menu of terms is presented; if ask is FALSE, canonical plots for all terms are produced.
`graphics`	if `TRUE` (the default, when running interactively), then the menu of terms to plot is presented in a dialog box rather than as a text menu.

Value

An object of class `candiscList` which is a list of `"candisc"` objects for the terms in the mlm.

Methods (by class)

- `candiscList(mlm)`: `"mlm"` method.

Methods (by generic)

- `print(candiscList)`: `print()` method for `"candiscList"` objects.
- `summary(candiscList)`: `summary()` method for `"candiscList"` objects.
- `plot(candiscList)`: `plot()` method for `"candiscList"` objects.

Author(s)

Michael Friendly and John Fox

See Also

`candisc()`, `heplots::heplot()`, `heplots::heplot3d()`

Examples

```
grass.mod <- lm(cbind(N1,N9,N27,N81,N243) ~ Block + Species, data=Grass)

grass.canL <-candiscList(grass.mod)
names(grass.canL)
names(grass.canL$Species)

## Not run:
print(grass.canL)

## End(Not run)
plot(grass.canL, type="n", ask=FALSE)
heplot(grass.canL$Species, scale=6)
heplot(grass.canL$Block, scale=2)
```

`can_lm` *Transform a Multivariate Linear model* `mlm` *to a Canonical Representation*

Description

This function uses `candisc()` to transform the responses in a multivariate linear model to scores on canonical variables for a given term and then uses those scores as responses in a linear (lm) or multivariate linear model (mlm).

The function constructs a model formula of the form `Can ~ terms` where Can is the canonical score(s) and terms are the terms in the original mlm, then runs lm() with that formula.

Usage

```
can_lm(mod, term, ...)
```

Arguments

`mod`	A `mlm` object
`term`	One term in that model
`...`	Arguments passed to `candisc()`

Value

A `lm` object if `term` is a rank 1 hypothesis, otherwise a `mlm` object

Author(s)

Michael Friendly

See Also

`candisc()`, `cancor()`

Examples

```
iris.mod <- lm(cbind(Petal.Length, Sepal.Length, Petal.Width, Sepal.Width) ~ Species, data=iris)
iris.can <- can_lm(iris.mod, "Species")
iris.can
car::Anova(iris.mod)
car::Anova(iris.can)
```

`cereal` *Breakfast Cereal Dataset*

Description

A multivariate dataset describing seventy-seven commonly available breakfast cereals, based on the information now available on the FDA food label. The variable `rating` is a likely response variable in statistical models.

Usage

```
data("cereal")
```

Format

A data frame with 77 observations on the following 16 variables.

- `name` cereal name, a character vector
- `mfr` manufacturer (A G K N P Q R), a character vector
- `type` type (cold/hot), a character vector
- `calories` calories (number), a numeric vector
- `protein` protein(g), a numeric vector
- `fat` fat(g), a numeric vector
- `sodium` sodium(mg), a numeric vector
- `fiber` dietary fiber(g), a numeric vector
- `carbo` complex carbohydrates(g), a numeric vector
- `sugars` sugars(g), a numeric vector
- `potass` potassium(mg), a numeric vector
- `vitamins` vitamins & minerals (0, 25, or 100, respectively indicating "none added"; "enriched"; "FDA recommended"), a numeric vector
- `shelf` display shelf (1, 2, or 3, counting from the floor), a numeric vector
- `weight` weight (in ounces) of one serving (serving size), a numeric vector
- `cups` cups per serving, a numeric vector
- `rating` health rating of the cereal (unknown calculation method), a numeric vector

Details

This dataset was used in the poster competition for the American Statistical association 1993 Statistical Graphics Exposition, titled *Serial Correlation or Cereal Correlation ??*.

The call for participation reads: "A multivariate dataset describing seventy-seven commonly available breakfast cereals, based on the information now available on the newly-mandated F&DA food label. What are you getting when you eat a bowl of cereal? Can you get a lot of fiber without a lot of calories? Can you describe what cereals are displayed on high, low, and middle shelves? The good news is that none of the cereals for which we collected data had any cholesterol, and manufacturers rarely use artificial sweeteners and colors, nowadays. However, there is still a lot of data for the consumer to understand while choosing a good breakfast cereal."

Further details on the variables and suggested analyses are available at `https://community.amstat.org/jointscsg-section/dataexpo/dataexpo1993`

The abbreviations for manufacturer, `mfr`, stand for:

`A`	American Home Food Products
`G`	General Mills
`K`	Kellog
`N`	Nabisco
`P`	Post
`Q`	Quaker Oats
`R`	Ralston Purina

Source

From the American Statistical Association 1993 Statistical Graphics Exposition, 'Serial Correlation or Cereal Correlation ??', `https://community.amstat.org/jointscsg-section/dataexpo/dataexpo1993`.

References

Jean Dos Santos, Breakfast Cereals: Data Analysis and Clustering, (Kaggle link doesn't work) Does a bunch of data cleaning and exploratory data analysis in R.

See Also

MASS::UScereal has a similar dataset with fewer observations and variables, but with the variables normalized to a portion of one US cup.

`https://www.kaggle.com/datasets/crawford/80-cereals` Essentially the same dataset

Examples

```
library(dplyr)
data(cereal)
str(cereal)

# Add explicit name of manufacturer
# names for manufacturers
mfr_names <- c(
  "A" = "American Home Foods",
  "G" = "General Mills",
  "K" = "Kellog",
  "N" = "Nabisco",
  "P" = "Post",
  "Q" = "Quaker Oats",
  "R" = "Ralston Purina"
```

```
)

# recode `mfr` as `mfr_name`
cereal <- cereal |>
  mutate(mfr_name = recode(mfr, !!!mfr_names))

# density plot of ratings
library(ggplot2)
ggplot(data = cereal,
       aes(x = rating, fill = mfr_name, color = mfr_name)) +
  geom_density(alpha = 0.1) +
  theme_classic(base_size = 14) +
  theme(legend.position = "bottom")
```

`cor_lda` *Calculate Structure Correlations from Discriminant Analysis*

Description

`cor_lda()` calculates the "structure" correlations between the observed variables and the discriminant dimension scores from a linear discriminant analysis provided by `MASS::lda()`. These more directly assess the direction and strength of the relations between the two sets than do the `scaling` weights returned by `lda()`. They are useful for plotting the discriminant scores, showing the contributions of the variables by vectors.

Usage

```
cor_lda(
  object,
  prior = object$prior,
  dimen,
  method = c("pearson", "kendall", "spearman"),
  ...
)
```

Arguments

`object`	An object of class `"lda"` such as results from `MASS::lda()`
`prior`	The prior probabilities of the classes. By default, taken to be the proportions in what was set in the call to `MASS::lda()`
`dimen`	The dimension of the space to be used. If this is less than the number of available dimensions, $\min(p, ng - 1)$, only the first `dimen` discriminant components are used.
`method`	a character string indicating which correlation coefficient is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`: can be abbreviated. See `stats::cor()` for details
`...`	other arguments (presently ignored)

Value

`a numeric matrix of correlations, of size `nv` = number of predictor variables * `dimen``

Author(s)

Michael Friendly

See Also

`predict_discrim()`, `MASS::lda()`, `stats::cor()`

Examples

```
library(candisc)
library(MASS)   # for lda()

iris.lda <- lda(Species ~ ., iris)
cor_lda(iris.lda)
```

`dataIndex` *Indices of observations in a model data frame*

Description

Find sequential indices for observations in a data frame corresponding to the unique combinations of the levels of a given model term from a model object or a data frame

Usage

```
dataIndex(x, term)
```

Arguments

`x`	Either a data frame or a model object
`term`	The name of one term in the model, consisting only of factors

Value

A vector of indices.

Author(s)

Michael Friendly

Examples

```
factors <- expand.grid(A=factor(1:3),B=factor(1:2),C=factor(1:2))
n <- nrow(factors)
responses <-data.frame(Y1=10+round(10*rnorm(n)),Y2=10+round(10*rnorm(n)))

test <- data.frame(factors, responses)
mod <- lm(cbind(Y1,Y2) ~ A*B, data=test)

dataIndex(mod, "A")
dataIndex(mod, "A:B")
```

`Grass` *Yields from Nitrogen nutrition of grass species*

Description

The data frame `Grass` gives the yield (10 * log10 dry-weight (g)) of eight grass Species in five replicates (Block) grown in sand culture at five levels of nitrogen.

Format

A data frame with 40 observations on the following 7 variables.

`Species` a factor with levels `B.media` `D.glomerata` `F.ovina` `F.rubra` `H.pubesens` `K.cristata` `L.perenne` `P.bertolonii`

`Block` a factor with levels `1` `2` `3` `4` `5`

`N1` species yield at 1 ppm Nitrogen

`N9` species yield at 9 ppm Nitrogen

`N27` species yield at 27 ppm Nitrogen

`N81` species yield at 81 ppm Nitrogen

`N243` species yield at 243 ppm Nitrogen

Details

Nitrogen (NaNO3) levels were chosen to vary from what was expected to be from critically low to almost toxic. The amount of Nitrogen can be considered on a log3 scale, with levels 0, 2, 3, 4, 5. Gittins (1985, Ch. 11) treats these as equally spaced for the purpose of testing polynomial trends in Nitrogen level.

The data are also not truly multivariate, but rather a split-plot experimental design. For the purpose of exposition, he regards Species as the experimental unit, so that correlations among the responses refer to a composite representative of a species rather than to an individual exemplar.

Source

Gittins, R. (1985), Canonical Analysis: A Review with Applications in Ecology, Berlin: Springer-Verlag, Table A-5.

Examples

```
str(Grass)
grass.mod <- lm(cbind(N1,N9,N27,N81,N243) ~ Block + Species, data=Grass)
car::Anova(grass.mod)

grass.canL <-candiscList(grass.mod)
names(grass.canL)
names(grass.canL$Species)
```

`heplot.cancor` *Canonical Correlation HE plots*

Description

Hypothesis - Error (HE) plots for canonical correlation analysis provide a new graphical method for understanding the relations between two sets of variables, $\mathbf{X}$ and $\mathbf{Y}$. They are similar to HE plots for multivariate multiple regression (MMRA) problems, except that ...

These functions plot ellipses (or ellipsoids in 3D) in canonical space representing the hypothesis and error sums-of-squares-and-products matrices for terms in a multivariate linear model representing the result of a canonical correlation analysis. They provide a low-rank 2D (or 3D) view of the effects in the space of maximum canonical correlations, together with variable vectors representing the correlations of Y variables with the canonical dimensions.

For consistency with `heplot.candisc()`, the plots show effects in the space of the canonical Y variables selected by `which`.

Usage

```
## S3 method for class 'cancor'
heplot(
  mod,
  which = 1:2,
  scale,
  asp = 1,
  var.vectors = "Y",
  var.col = c("blue", "darkgreen"),
  var.lwd = par("lwd"),
  var.cex = par("cex"),
  var.xpd = TRUE,
  prefix = "Ycan",
  suffix = TRUE,
```

```
  terms = TRUE,
  ...
)
```

Arguments

`mod`	A `"cancor"` object, as computed by `cancor()`
`which`	A numeric vector containing the indices of the Y canonical dimensions to plot.
`scale`	Scale factor for the variable vectors in canonical space. If not specified, the function calculates one to make the variable vectors approximately fill the plot window.
`asp`	aspect ratio setting. Use `asp=1` in 2D plots and `asp="iso"` in 3D plots to ensure equal units on the axes. Use `asp=NA` in 2D plots and `asp=NULL` in 3D plots to allow separate scaling for the axes. See Details below.
`var.vectors`	Which variable vectors to plot? A character vector containing one or more of `"X"` and `"Y"`.
`var.col`	Color(s) for variable vectors and labels, a vector of length 1 or 2. The first color is used for Y vectors and the second for X vectors, if these are plotted.
`var.lwd`	Line width for variable vectors
`var.cex`	Text size for variable vector labels
`var.xpd`	logical. Allow variable labels outside the plot box? Does not apply to 3D plots.
`prefix`	Prefix for labels of the Y canonical dimensions.
`suffix`	Suffix for labels of canonical dimensions. If `suffix=TRUE` the percent of hypothesis (H) variance accounted for by each canonical dimension is added to the axis label.
`terms`	Terms for the X variables to be plotted in canonical space. The default, `terms=TRUE` or `terms="X"` plots H ellipses for all of the X variables. `terms="Xcan"` plots H ellipses for all of the X canonical variables, `Xcan1`, `Xcan2`, `...`.
`...`	Other arguments passed to `link[heplots]{heplot}`. In particular, you can pass linear hypotheses among the term variables via `hypotheses`.

Details

The interpretation of variable vectors in these plots is different from that of the `terms` plotted as H "ellipses," which appear as degenerate lines in the plot (because they correspond to 1 df tests of rank(H)=1).

In canonical space, the interpretation of the H ellipses for the `terms` is the same as in ordinary HE plots: a term is significant *iff* its H ellipse projects outside the (orthogonalized) E ellipsoid somewhere in the space of the Y canonical dimensions. The orientation of each H ellipse with respect to the Y canonical dimensions indicates which dimensions that X variate contributes to.

On the other hand, the variable vectors shown in these plots are intended only to show the correlations of Y variables with the canonical dimensions. Only their relative lengths and angles with respect to the Y canonical dimensions have meaning. Relative lengths correspond to proportions of variance accounted for in the Y canonical dimensions plotted; angles between the variable vectors and the canonical axes correspond to the structure correlations. The absolute lengths of these vectors are typically manipulated by the `scale` argument to provide better visual resolution and labeling for the variables.

Setting the aspect ratio of these plots is important for the proper interpretation of angles between the variable vectors and the coordinate axes. However, this then makes it impossible to change the aspect ratio of the plot by re-sizing manually. You can override this using `asp=NA` in 2D plots

Value

Returns invisibly an object of class `"heplot"`, with coordinates for the various hypothesis ellipses and the error ellipse, and the limits of the horizontal and vertical axes.

Author(s)

Michael Friendly

References

Gittins, R. (1985). *Canonical Analysis: A Review with Applications in Ecology*, Berlin: Springer.

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

See Also

```
[cancor()] for details on canonical correlation as implemented here;
[plot.cancor()] for scatterplots of canonical  variable scores.
[heplot.candisc()], [heplots::heplot()], [car::linearHypothesis()]
```

Examples

```
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])
Y <- as.matrix(Rohwer[,3:5])
cc <- cancor(X, Y, set.names=c("PA", "Ability"))

# basic plot
heplot(cc)

# note relationship of joint hypothesis to individual ones
heplot(cc, scale=1.25, hypotheses=list("na+ns"=c("na", "ns")))

# more options
heplot(cc, hypotheses=list("All X"=colnames(X)),
fill=c(TRUE,FALSE), fill.alpha=0.2,
var.cex=1.5, var.col="red", var.lwd=3,
prefix="Y canonical dimension"
)

# 3D version
## Not run:
heplot3d(cc, var.lwd=3, var.col="red")

## End(Not run)
```

`heplot.candisc` *Canonical Discriminant HE plots*

Description

These functions plot ellipses (or ellipsoids in 3D) in canonical discriminant space representing the hypothesis and error sums-of-squares-and-products matrices for terms in a multivariate linear model. They provide a low-rank 2D (or 3D) view of the effects for that term in the space of maximum discrimination.

Usage

```
## S3 method for class 'candisc'
heplot(
  mod,
  which = 1:2,
  scale,
  asp = 1,
  var.col = "blue",
  var.labels,
  var.lwd = par("lwd"),
  var.cex = par("cex"),
  var.pos,
  rev.axes = c(FALSE, FALSE),
  prefix = "Can",
  suffix = TRUE,
  terms = mod$term,
  ...
)
```

Arguments

`mod`	A `candisc` object for one term in a `mlm`
`which`	A numeric vector containing the indices of the canonical dimensions to plot.
`scale`	Scale factor for the variable vectors in canonical space. If not specified, the function calculates one to make the variable vectors approximately fill the plot window.
`asp`	Aspect ratio for the horizontal and vertical dimensions. The defaults, `asp=1` for `heplot.candisc` and `asp="iso"` for `heplot3d.candisc` ensure equal units on all axes, so that angles and lengths of variable vectors are interpretable. As well, the standardized canonical scores are uncorrelated, so the Error ellipse (ellipsoid) should plot as a circle (sphere) in canonical space. For `heplot3d.candisc`, use `asp=NULL` to suppress this transformation to iso-scaled axes.

`var.col`	Color for variable vectors and labels
`var.labels`	Labels for the variable vectors. The default is the rownames of the canonical structure coefficients.
`var.lwd`	Line width for variable vectors
`var.cex`	Text size for variable vector labels
`var.pos`	Position(s) of variable vector labels wrt. the end point. If not specified, the labels are out-justified left and right with respect to the end points.
`rev.axes`	Logical, a vector of `length(which)`. `TRUE` causes the orientation of the canonical scores and structure coefficients to be reversed along a given axis.
`prefix`	Prefix for labels of canonical dimensions.
`suffix`	Suffix for labels of canonical dimensions. If `suffix=TRUE` the percent of hypothesis (H) variance accounted for by each canonical dimension is added to the axis label.
`terms`	Terms from the original `mlm` whose H ellipses are to be plotted in canonical space. The default is the one term for which the canonical scores were computed. If `terms=TRUE`, all terms are plotted.
`...`	Arguments to be passed down to `heplot` or `heplot3d`

Details

The generalized canonical discriminant analysis for one term in a `mlm` is based on the eigenvalues, λ_i, and eigenvectors, V, of the H and E matrices for that term. This produces uncorrelated canonical scores which give the maximum univariate F statistics. The canonical HE plot is then just the HE plot of the canonical scores for the given term.

For `heplot3d.candisc`, the default `asp="iso"` now gives a geometrically correct plot, but the third dimension, CAN3, is often small. Passing an expanded range in `zlim` to `heplot3d` usually helps.

Value

`heplot.candisc` returns invisibly an object of class `"heplot"`, with coordinates for the various hypothesis ellipses and the error ellipse, and the limits of the horizontal and vertical axes.

Similarly, `heploted.candisc` returns an object of class `"heplot3d"`.

Author(s)

Michael Friendly and John Fox

References

Friendly, M. (2006). Data Ellipses, HE Plots and Reduced-Rank Displays for Multivariate Linear Models: SAS Software and Examples *Journal of Statistical Software*, 17(6), 1-42. % `https://www.jstatsoft.org/v17/i06/` doi:10.18637/jss.v017.i06

Friendly, M. (2007). HE plots for Multivariate General Linear Models. *Journal of Computational and Graphical Statistics*, **16**(2) 421–444. `http://datavis.ca/papers/jcgs-heplots.pdf`, doi:10.1198/106186007X208407.

See Also

candisc, candiscList, heplot, heplot3d, aspect3d

Examples

```
## Pottery data, from car package
data(Pottery, package = "carData")
pottery.mod <- lm(cbind(Al, Fe, Mg, Ca, Na) ~ Site, data=Pottery)
pottery.can <-candisc(pottery.mod)

heplot(pottery.can, var.lwd=3)
if(requireNamespace("rgl")){
heplot3d(pottery.can, var.lwd=3, scale=10, zlim=c(-3,3), wire=FALSE)
}


# reduce example for CRAN checks time

grass.mod <- lm(cbind(N1,N9,N27,N81,N243) ~ Block + Species, data=Grass)

grass.can1 <-candisc(grass.mod,term="Species")
grass.canL <-candiscList(grass.mod)

heplot(grass.can1, scale=6)
heplot(grass.can1, scale=6, terms=TRUE)
heplot(grass.canL, terms=TRUE, ask=FALSE)

heplot3d(grass.can1, wire=FALSE)
# compare with non-iso scaling
rgl::aspect3d(x=1,y=1,z=1)
# or,
# heplot3d(grass.can1, asp=NULL)


## Can't run this in example
# rgl::play3d(rgl::spin3d(axis = c(1, 0, 0), rpm = 5), duration=12)

# reduce example for CRAN checks time

## FootHead data, from heplots package
library(heplots)
data(FootHead)

# use Helmert contrasts for group
contrasts(FootHead$group) <- contr.helmert

foot.mod <- lm(cbind(width, circum,front.back,eye.top,ear.top,jaw)~group, data=FootHead)
foot.can <- candisc(foot.mod)
heplot(foot.can, main="Candisc HE plot",
 hypotheses=list("group.1"="group1","group.2"="group2"),
 col=c("red", "blue", "green3", "green3" ), var.col="red")
```

`heplot.candiscList` *Canonical Discriminant HE plots*

Description

These functions plot ellipses (or ellipsoids in 3D) in canonical discriminant space representing the hypothesis and error sums-of-squares-and-products matrices for terms in a multivariate linear model. They provide a low-rank 2D (or 3D) view of the effects for that term in the space of maximum discrimination.

Usage

```
## S3 method for class 'candiscList'
heplot(mod, term, ask = interactive(), graphics = TRUE, ...)
```

Arguments

`mod`	A `candiscList` object for terms in a `mlm`
`term`	The name of one term to be plotted for the `heplot` and `heplot3d` methods. If not specified, one plot is produced for each term in the `mlm` object.
`ask`	If `TRUE` (the default), a menu of terms is presented; if ask is FALSE, canonical HE plots for all terms are produced.
`graphics`	if `TRUE` (the default, when running interactively), then the menu of terms to plot is presented in a dialog box rather than as a text menu.
`...`	Arguments to be passed down

Value

No useful value; used for the side-effect of producing canonical HE plots.

Author(s)

Michael Friendly and John Fox

References

Friendly, M. (2006). Data Ellipses, HE Plots and Reduced-Rank Displays for Multivariate Linear Models: SAS Software and Examples *Journal of Statistical Software*, 17(6), 1-42. `https://www.jstatsoft.org/v17/i06/` doi:10.18637/jss.v017.i06.

Friendly, M. (2007). HE plots for Multivariate General Linear Models. *Journal of Computational and Graphical Statistics*, **16**(2) 421–444. `http://datavis.ca/papers/jcgs-heplots.pdf`, doi:10.1198/106186007X208407.

See Also

`candisc`, `candiscList`, `heplot`, `heplot3d`

`HSB` *High School and Beyond Data*

Description

The High School and Beyond Project was a longitudinal study of students in the U.S. carried out in 1980 by the National Center for Education Statistics. Data were collected from 58,270 high school students (28,240 seniors and 30,030 sophomores) and 1,015 secondary schools. The HSB data frame is sample of 600 observations, of unknown characteristics, originally taken from Tatsuoka (1988).

Format

A data frame with 600 observations on the following 15 variables. There is no missing data.

`id` Observation id: a numeric vector

`gender` a factor with levels `male` `female`

`race` Race or ethnicity: a factor with levels `hispanic` `asian` `african-amer` `white`

`ses` Socioeconomic status: a factor with levels `low` `middle` `high`

`sch` School type: a factor with levels `public` `private`

`prog` High school program: a factor with levels `general` `academic` `vocation`

`locus` Locus of control: a numeric vector

`concept` Self-concept: a numeric vector

`mot` Motivation: a numeric vector

`career` Career plan: a factor with levels `clerical` `craftsman` `farmer` `homemaker` `laborer` `manager` `military` `operative` `prof1` `prof2` `proprietor` `protective` `sales` `school` `service` `technical` `not working`

`read` Standardized reading score: a numeric vector

`write` Standardized writing score: a numeric vector

`math` Standardized math score: a numeric vector

`sci` Standardized science score: a numeric vector

`ss` Standardized social science (civics) score: a numeric vector

Source

Tatsuoka, M. M. (1988). Multivariate Analysis: Techniques for Educational and Psychological Research (2nd ed.). New York: Macmillan, Appendix F, 430-442.

% Originally retrieved from: http://www.gseis.ucla.edu/courses/data/hbs6.dta

References

High School and Beyond data files: http://www.icpsr.umich.edu/icpsrweb/ICPSR/studies/7896

Examples

```
str(HSB)
# main effects model
hsb.mod <- lm( cbind(read, write, math, sci, ss) ~
gender + race + ses + sch + prog, data=HSB)
car::Anova(hsb.mod)

# Add some interactions
hsb.mod1 <- update(hsb.mod, . ~ . + gender:race + ses:prog)
heplot(hsb.mod1, col=palette()[c(2,1,3:6)], variables=c("read","math"))

hsb.can1 <- candisc(hsb.mod1, term="race")
heplot(hsb.can1, col=c("red", "black"))

# show canonical results for all terms
## Not run:
hsb.can <- candiscList(hsb.mod)
hsb.can

## End(Not run)
```

`painters2` *Painters Data with Historical Art Variables*

Description

The original `painters` dataset from the **MASS** package contains the subjective assessment, on a 0 to 20 integer scale, of 54 classical painters. The painters were rated on four characteristics: Composition, Drawing, Colour and Expression, and grouped according to "School" based on nationality, era, and style. The data is due to the Eighteenth century art critic, Roger de Piles (1743).

This extended version adds categorical variables that capture art historical distinctions among schools of painting based on their chronological period, artistic emphasis, style, and treatment of light.

Usage

```
painters2
```

Format

A data frame with 54 observations on 10 variables:

Composition Composition score (0-20) assigned by de Piles

Drawing Drawing score (0-20) assigned by de Piles

Colour Colour score (0-20) assigned by de Piles

Expression Expression score (0-20) assigned by de Piles

School School of painter: "Renaissance", "Mannerist", "Sciento", "Venetian", "Lombard", "16th C", "17th C", "French"

Sch Letter code for School of painter: "A" through "H"

Period An ordered factor. Historical period: "Early" (1400-1520: Renaissance, Venetian, Lombard), "Transition" (1500-1600: 16th C, Mannerist), "Baroque" (1600-1750: Sciento, 17th C, French)

Emphasis A factor. Primary artistic focus: "Form" (emphasis on drawing, composition, classical ideals), "Color" (emphasis on color and light effects), "Drama" (emphasis on dramatic realism and emotional intensity)

Style A factor. Aesthetic approach: "Classical" (balanced, harmonious, adherence to classical ideals), "Expressive" (emotional intensity, dramatic effects), "Regional" (distinctive regional characteristics)

Light A factor. Treatment of light and shadow: "Balanced" (even, harmonious lighting), "Luminous" (rich color and atmospheric light effects), "Dramatic" (strong chiaroscuro, dramatic contrasts)

Details

Names of the painters are given as `rownames(painters2)`.

The original version `painters` used letters, "A" through "H" to identify the schools of painters. This is kept here as the variable `Sch`, and the variable `School` now gives the actual label of the school.

The four new categorical variables (Period, Emphasis, Style, Light) were constructed to reflect art historical distinctions among the schools:

- **Period** groups schools by broad historical era
- **Emphasis** captures the primary artistic focus of each school
- **Style** reflects the aesthetic approach and philosophical orientation
- **Light** distinguishes approaches to light and shadow treatment

These variables can be useful for exploring how art historical categories relate to de Piles' quantitative ratings, and for demonstrating MANOVA and multivariate discriminant analysis techniques.

Source

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

References

De Piles, R. (1743). The Principles of Painting. London, n.p.

See Also

`painters`

Examples

```
data(painters2)

# Compare original School with new Period grouping
with(painters2, table(School, Period))

# Compare original School with new Period grouping
with(painters2, table(School, Emphasis))

# Summary of de Piles ratings by Period
aggregate(cbind(Composition, Drawing, Colour, Expression) ~ Period,
          data = painters2, FUN = mean)
```

`plot.cancor` *Canonical Correlation Plots*

Description

This function produces plots to help visualize X, Y data in canonical space.

The present implementation plots the canonical scores for the Y variables against those for the X variables on given dimensions. We treat this as a view of the data in canonical space, and so offer additional annotations to a standard scatterplot.

Canonical correlation analysis assumes that the all correlations between the X and Y variables can be expressed in terms of correlations the canonical variate pairs, (Xcan1, Ycan1), (Xcan2, Ycan2), . . . , and that the relations between these pairs are indeed linear.

Data ellipses, and smoothed (loess) curves, together with the linear regression line for each canonical dimension help to assess whether there are peculiarities in the data that might threaten the validity of CCA. Point identification methods can be useful to determine influential cases.

Usage

```
## S3 method for class 'cancor'
plot(
  x,
  which = 1,
  xlim,
  ylim,
  xlab,
```

```
  ylab,
  points = TRUE,
  add = FALSE,
  col = palette()[1],
  ellipse = TRUE,
  ellipse.args = list(),
  smooth = FALSE,
  smoother.args = list(),
  col.smooth = palette()[3],
  abline = TRUE,
  col.lines = palette()[2],
  lwd = 2,
  labels = rownames(xy),
  id.method = "mahal",
  id.n = 0,
  id.cex = 1,
  id.col = palette()[1],
  ...
)
```

Arguments

`x`	A `"cancor"` object
`which`	Which dimension to plot? An integer in `1:x$ndim`.
`xlim, ylim`	Limits for x and y axes
`xlab, ylab`	Labels for x and y axes. If not specified, these are constructed from the `set.names` component of `x`.
`points`	logical. Display the points?
`add`	logical. Add to an existing plot?
`col`	Color for points.
`ellipse`	logical. Draw a data ellipse for the canonical scores?
`ellipse.args`	A list of arguments passed to `dataEllipse`. Internally, the function sets the default value for `levels` to 0.68.
`smooth`	logical. Draw a (loess) smoothed curve?
`smoother.args`	Arguments passed to `loessLine`, which should be consulted for details and defaults.
`col.smooth`	Color for the smoothed curve.
`abline`	logical. Draw the linear regression line for `Ycan[, which]` on `Xcan[, which]`?
`col.lines`	Color for the linear regression line
`lwd`	Line widths
`labels`	Point labels for point identification via the `id.method` argument.
`id.method`	Method used to identify individual points. See `showLabels` for details. The default, `id.method = "mahal"` identifies the `id.n` points furthest from the centroid.

`id.n`	Number of points to identify
`id.cex`, `id.col`	Character size and color for labeled points
`...`	Other arguments passed down to `plot(\dots)` and `points(\dots)`

Value

None. Used for its side effect of producing a plot. %% ~Describe the value returned

Author(s)

Michael Friendly

References

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

See Also

`cancor`,

`dataEllipse`, `loessLine`, `showLabels`

Examples

```
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])  # the PA tests
Y <- as.matrix(Rohwer[,3:5])   # the aptitude/ability variables

cc <- cancor(X, Y, set.names=c("PA", "Ability"))

plot(cc)
# exercise some options
plot(cc, which=1,
     smooth=TRUE,
     pch = 16,
     id.n=3, ellipse.args=list(fill=TRUE, fill.alpha = 0.2))
plot(cc, which=2, smooth=TRUE)
plot(cc, which=3, smooth=TRUE)


# plot vectors showing structure correlations of Xcan and Ycan with their own variables
plot(cc)
struc <- cc$structure
Xstruc <- struc$X.xscores[,1]
Ystruc <- struc$Y.yscores[,1]
scale <- 2

# place vectors in the margins of the plot
usr <- matrix(par("usr"), nrow=2, dimnames=list(c("min", "max"), c("x", "y")))
ypos <- usr[2,2] - (1:5)/10
arrows(0, ypos, scale*Xstruc, ypos, angle=10, len=0.1, col="blue")
text(scale*Xstruc, ypos, names(Xstruc), pos=2, col="blue")
```

```
xpos <- usr[2,1] - ( 1 + 1:3)/10
arrows(xpos, 0, xpos, scale*Ystruc, angle=10, len=0.1, col="darkgreen")
text(xpos, scale*Ystruc, names(Ystruc), pos=1, col="darkgreen")
```

`plot_discrim` *Discriminant Analysis Decision Plot using ggplot.*

Description

Discriminant analysis can be more easily understood from plots of the data variables showing how observations are classified. `plot_discrim()` uses the ideas behind **effect plots** (Fox, 1987): Visualize predicted classes of the observations for two focal variables over a grid of their values, with other variables in a model held fixed. This differs from the usual effect plots in that the predicted values to be visualized are discrete categories rather than quantitative.

In the case of discriminant analysis, the predicted values are class membership, so this can be visualized by mapping the categorical predicted class to discrete colors used as the background for the plot, or plotting the **contours** of predicted class membership as lines (for `[MASS::lda()]`) or qauadratic curves (for `[MASS::qda()]`) in the plot. The predicted class of any observation in the space of the variables displayed can also be rendered as colored **tiles** or **points** in the background of the plot.

Usage

```
plot_discrim(
  model,
  vars,
  data = insight::get_data(model),
  resolution = 100,
  point.size = 3,
  showgrid = c("tile", "point", "none"),
  contour = TRUE,
  contour.color = "black",
  tile.alpha = 0.2,
  ellipse = FALSE,
  ellipse.args = list(level = 0.68, linewidth = 1.2),
  labels = FALSE,
  labels.args = list(geom = "text", size = 5),
  rev.axes = c(FALSE, FALSE),
  xlim = NULL,
  ylim = NULL,
  ...,
  other.levels
)
```

Arguments

`model`	a discriminant analysis model object from `MASS::lda()` or `MASS::qda()`
`vars`	either a character vector of length 2 of the names of the `x` and `y` variables, or a formula of form `y ~ x` specifying the axes in the plot. Can include discriminant dimensions like `LD1`, `LD2`, etc.
`data`	data to use for visualization. Should contain all the data needed to use the `model` for prediction. The default is to use the data used to fit the `model`.
`resolution`	number of points in x, y variables to use for visualizing the predicted class boundaries and regions.
`point.size`	size of the plot symbols use to show the data observations
`showgrid`	a character string; how to display predicted class regions: `"tile"` for `ggplot2::geom_tile()`, `"point"` for `ggplot2::geom_point()`, or `"none"` for no grid display.
`contour`	logical (default: `TRUE`); should the plot display the boundaries of the classes by contours?
`contour.color`	color of the lines for the contour boundaries (default: `"black"`)
`tile.alpha`	transparency value for the background tiles of predicted class.
`ellipse`	logical; if `TRUE`, 68 percent data ellipses for the groups are added to the plot.
`ellipse.args`	a named list of arguments passed to `ggplot2::stat_ellipse()`. Common arguments include `level` (confidence level, default: 0.68), `linewidth` (line thickness, default: 1.2), `geom` (either `"path"` for unfilled ellipses or `"polygon"` for filled ellipses), and `alpha` (transparency for filled ellipses). Any valid argument to `stat_ellipse()` can be used.
`labels`	logical; if `TRUE`, class labels are added to the plot at the group means (default: `FALSE`).
`labels.args`	a named list of arguments passed to `ggplot2::geom_text()` or `ggplot2::geom_label()`. Common arguments include `geom` (either `"text"` or `"label"`, default: `"text"`), `size` (text size, default: 5), `fontface` (e.g., `"bold"` or `"italic"`), `nudge_x` and `nudge_y` (position offsets), and `alpha` (transparency for label backgrounds). Any valid argument to `geom_text()` or `geom_label()` can be used.
`rev.axes`	a logical vector of length 2 controlling axis reversal for discriminant dimensions. `rev.axes[1] = TRUE` reverses the horizontal (x) axis; `rev.axes[2] = TRUE` reverses the vertical (y) axis. Only applies when plotting discriminant dimensions (e.g., `LD2 ~ LD1`). Default: `c(FALSE, FALSE)`.
`xlim`, `ylim`	numeric vectors of length 2 giving the axis limits. If `NULL` (default), uses the range of the variable in the data.
`...`	further parameters passed to `predict()`
`other.levels`	a named list specifying the fixed values to use for variables in the model that are **not** included in `vars` (the non-focal variables). These values are held constant across the prediction grid. If not specified, the function uses sensible defaults: means for quantitative variables, and the first level for factors or character variables.

Details

Since `plot_discrim()` returns a `"ggplot"` object, you can easily customize colors and shapes by adding scale layers after the function call. You can also add other graphic layers, such as annotations, and control the overall appearance of plots using `ggplot2::theme()` components.

Customizing colors and shapes

- Use `scale_color_manual()` **and** `scale_fill_manual()` to control the colors used when using `showgrid = "tile"`, because that maps both **both** `color` and `fill` to the group variable.
- Use `scale_shape_manual()` to control the symbols used for `geom_points()`

Customizing ellipses

The `ellipse.args` parameter provides fine control over the appearance of data ellipses. Common arguments include:

- `level`: the confidence level for the ellipse (default: 0.68)
- `linewidth`: thickness of the ellipse line (default: 1.2)
- `geom`: either `"path"` for unfilled ellipses (default) or `"polygon"` for filled ellipses
- `alpha`: transparency when using `geom = "polygon"`

See `ggplot2::stat_ellipse()` for additional parameters.

Adding class labels

The `labels` and `labels.args` parameters allow you to add text labels for each class, positioned at the group means. Common arguments for `labels.args` include:

- `geom`: either `"text"` (default) for simple text or `"label"` for text with a background box
- `size`: text size (default: 5)
- `fontface`: font style such as `"bold"` or `"italic"`
- `nudge_x`, `nudge_y`: offsets for label positioning
- `alpha`: transparency for label backgrounds when using `geom = "label"`

See `ggplot2::geom_text()` and `ggplot2::geom_label()` for additional parameters.

Plotting in discriminant space

When `vars` specifies discriminant dimensions (e.g., `LD2 ~ LD1`), the function automatically:

1. Calculates discriminant scores using `predict_discrim()`
2. Creates a new LDA model in the discriminant space
3. Plots the observations and decision boundaries in this transformed space

This is particularly useful for visualizing how well the discriminant dimensions separate the groups, since by construction the groups are maximally separated in discriminant space.

Reversing discriminant axes

The orientation of discriminant axes (LD1, LD2, etc.) is arbitrary in the sense that multiplying any discriminant dimension by -1 does not change the discriminant solution or model fit. The `rev.axes` parameter allows you to reverse the direction of one or both axes when plotting in discriminant space. This can be useful for:

- Aligning the discriminant plot with conventional interpretations (e.g., having "positive" on the right)
- Making the orientation consistent across different analyses or visualizations
- Improving the interpretability of the axes in relation to the original variables

The `rev.axes` parameter **only affects plots of discriminant dimensions** (e.g., `LD2 ~ LD1`) and has no effect when plotting original observed variables. To reverse the horizontal axis (x-axis), set `rev.axes[1] = TRUE`; to reverse the vertical axis (y-axis), set `rev.axes[2] = TRUE`. Both axes can be reversed simultaneously with `rev.axes = c(TRUE, TRUE)`.

Author(s)

Original code by Oliver on SO `https://stackoverflow.com/questions/63782598/quadratic-discriminant-analysi`

Generalized by Michael Friendly

References

Fox, J. (1987). Effect Displays for Generalized Linear Models. In C. C. Clogg (Ed.), *Sociological Methodology*, 1987 (pp. 347–361). Jossey-Bass

See Also

`klaR::partimat()` for pairwise discriminant plots, but with little control of plot details

Examples

```
library(MASS)
library(ggplot2)
library(dplyr)

iris.lda <- lda(Species ~ ., iris)
# formula call: y ~ x
plot_discrim(iris.lda, Petal.Length ~ Petal.Width)

# add data ellipses
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
             ellipse = TRUE)

# add filled ellipses with transparency
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
             ellipse = TRUE,
             ellipse.args = list(geom = "polygon", alpha = 0.2))

# customize ellipse level and line thickness
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
             ellipse = TRUE,
             ellipse.args = list(level = 0.95, linewidth = 2))

# without contours
# data ellipses
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
```

```
             contour = FALSE)

# specifying `vars` as character names for x, y
plot_discrim(iris.lda, c("Petal.Width", "Petal.Length"))

# Define custom colors and shapes, modify theme() and legend.position
iris.colors <- c("red", "darkgreen", "blue")
iris.pch <- 15:17
plot_discrim(iris.lda, Petal.Length ~ Petal.Width) +
  scale_color_manual(values = iris.colors) +
  scale_fill_manual(values = iris.colors) +
  scale_shape_manual(values = iris.pch) +
  theme_bw(base_size = 14) +
  theme(legend.position = "inside",
        legend.position.inside = c(.8, .25))

# Quadratic discriminant analysis gives quite a different result
iris.qda <- qda(Species ~ ., iris)
plot_discrim(iris.qda, Petal.Length ~ Petal.Width)

# Add class labels, with custom styling
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
             labels = TRUE,
             labels.args = list(geom = "label", size = 6, fontface = "bold"))

# Add labels with position adjustments
plot_discrim(iris.lda, Petal.Length ~ Petal.Width,
             labels = TRUE,
             labels.args = list(nudge_y = 0.1, size = 5))

# Plot in discriminant space
plot_discrim(iris.lda, LD2 ~ LD1)

# Reverse the horizontal axis in discriminant space
plot_discrim(iris.lda, LD2 ~ LD1, rev.axes = c(TRUE, FALSE))

# Control axis limits
plot_discrim(iris.lda, LD2 ~ LD1,
             xlim = c(-10, 10), ylim = c(-8, 8))
```

`predictor.names` *Get predictor names from a* `lm`*-like model*

Description

Get predictor names from a `lm`-like model

Usage

```
predictor.names(model, ...)

## Default S3 method:
predictor.names(model, ...)
```

Arguments

`model`	Model object
`...`	other arguments (ignored)

Value

A character vector of variable names

Methods (by class)

- `predictor.names(default)`: `"default"` method.

Examples

```
#none
```

`predict_discrim` *Predicted values for discriminant analysis*

Description

`predict_discrim` calculates predicted class membership values for a linear or quadratic discriminant analysis, returning a `data.frame` suitable for graphing or other analysis.

Usage

```
predict_discrim(
  object,
  newdata,
  prior = object$prior,
  dimen,
  scores = FALSE,
  posterior = FALSE,
  ...
)
```

Arguments

`object`	An object of class `"lda"` or `"qda"` such as results from `MASS::lda()` or `MASS::qda()`
`newdata`	A data frame of cases to be classified or, if `object` has a formula, a data frame with columns of the same names as the variables used. A vector will be interpreted as a row vector. If `newdata` is missing, an attempt will be made to retrieve the data used to fit the `lda` object.
`prior`	The prior probabilities of the classes. By default, taken to be the proportions in what was set in the call to `MASS::lda()` or `MASS::qda()`
`dimen`	The dimension of the space to be used. If this is less than the number of available dimensions, $\min(p, ng - 1)$, only the first `dimen` discriminant components are used. (This argument is not yet implemented because `MASS::qda()` does not support this.)
`scores`	A logical. If `TRUE`, the discriminant scores of the cases in `newdata` are appended as additional columns in the the result, with names `LD1`, `LD2`, ...
`posterior`	Either a logical or the character string `"max"`. If `TRUE`, the posterior probabilities for all classes are included as columns named for the classes. If `FALSE`, these are omitted. If `"max"`, the maximum value of the probabilities across the classes are included, with the variable name `"maxp"`.
`...`	arguments based from or to other methods, not yet used here

Details

The `predict()` methods provided for `MASS::lda()` and `MASS::qda()` are a mess, because they return their results as a list, with components `class`, `posterior` and `x`. This function is designed as a wrapper on those to return results in a more consistent and flexible way.

For use in graphs, where you want to show the classification boundaries or regions, you should supply a `newdata` data frame consisting of two focal variables which are varied over their ranges, with the remaining variables used in the discriminant analysis held fixed at typical values.

Using the `scores` argument, the function also returns the scores on the discriminant functions. This is only available for linear discriminant analysis with `MASS::lda()`.

Value

A `data.frame`, containing the the predicted class of the observations (named for the class in the model) and values of the `newdata` variables. Other variables included are determined by the `scores` and `posterior` arguments. `rownames()` in the result are inherited from those in `newdata`.

Examples

```
library(candisc)
library(MASS)   # for lda()

iris.lda <- lda(Species ~ ., iris)
pred_iris <- predict_discrim(iris.lda)
names(pred_iris)

# include scores, exclude posterior
```

```
pred_iris <- predict_discrim(iris.lda, scores = TRUE, posterior = FALSE)
names(pred_iris)

data(peng, package="heplots")
peng.lda <- lda(species ~ bill_length + bill_depth + flipper_length + body_mass,
                data = peng)
peng_pred <- predict_discrim(peng.lda, scores = TRUE)
str(peng_pred)
```

`PsyAcad` *Psychological Measures and Academic Achievement*

Description

A researcher collected data on three psychological variables, four academic variables (standardized test scores) and gender for 600 college freshman. She is interested in how the set of psychological variables relates to the academic variables and gender. In particular, the researcher is interested in how many dimensions (canonical variables) are necessary to understand the association between the two sets of variables.

Usage

```
data("PsyAcad")
```

Format

A data frame with 600 observations on the following 8 variables.

- `LocControl` locus of control, a numeric vector
- `SelfConcept` self concept, a numeric vector
- `Motivation` motivation, a numeric vector
- `Read` reading score, a numeric vector
- `Write` writing score, a numeric vector
- `Math` mathematics score, a numeric vector
- `Science` science score, a numeric vector
- `Sex` a factor with levels `M`, `F`

Source

Taken from `https://stats.oarc.ucla.edu/r/dae/canonical-correlation-analysis/`

Examples

```
data(PsyAcad)
PsyAcad$Sex <- as.numeric(PsyAcad$Sex)
PsyAcad.can <- cancor(cbind(LocControl, SelfConcept, Motivation) ~
       Read + Write + Math + Science + Sex, data = PsyAcad)

PsyAcad.can

# redundancy analysis
redundancy(PsyAcad.can)

# Plots
canR <- PsyAcad.can$cancor
plot(PsyAcad.can, pch=16, id.n = 3)
text(-2, 3, paste("Can R =", round(canR[1], 3)), pos = 3)

plot(PsyAcad.can, which = 2, pch=16, id.n = 3)
text(-2, 3.5, paste("Can R =", round(canR[2], 3)), pos = 3)
```

`redundancy` *Canonical Redundancy Analysis*

Description

Calculates indices of redundancy (Stewart & Love, 1968) from a canonical correlation analysis. These give the proportion of variances of the variables in each set (X and Y) which are accounted for by the variables in the other set through the canonical variates.

Usage

```
redundancy(object, ...)

## S3 method for class 'cancor.redundancy'
print(x, digits = max(getOption("digits") - 3, 3), ...)
```

Arguments

`object`	A "cancor" object
`...`	Other arguments
`x`	A "`cancor.redundancy`" for the `print` method.
`digits`	Number of digits to print

Details

The term "redundancy analysis" has a different interpretation and implementation in the environmental ecology literature, such as the **vegan**. In that context, each Y_i variable is regressed separately on the predictors in X, to give fitted values $\widehat{Y} = [\widehat{Y}_1, \widehat{Y}_2, \dots$. Then a PCA of $\widehat{Y}$ is carried out to determine a reduced-rank structure of the predictions.

Value

An object of class `"cancor.redundancy"`, a list with the following 5 components:

`Xcan.redun`	Canonical redundancies for the X variables, i.e., the total fraction of X variance accounted for by the Y variables through each canonical variate.
`Ycan.redun`	Canonical redundancies for the Y variables
`X.redun`	Total canonical redundancy for the X variables, i.e., the sum of `Xcan.redun`.
`Y.redun`	Total canonical redundancy for the Y variables
`set.names`	names for the X and Y sets of variables

Functions

- `print(cancor.redundancy)`: `print()` method for `"cancor.redundancy"` objects.

Author(s)

Michael Friendly

References

Muller K. E. (1981). Relationships between redundancy analysis, canonical correlation, and multivariate regression. *Psychometrika*, **46**(2), 139-42.

Stewart, D. and Love, W. (1968). A general canonical correlation index. *Psychological Bulletin*, 70, 160-163.

Brainder, "Redundancy in canonical correlation analysis", `https://brainder.org/2019/12/27/redundancy-in-canonical-correlation-analysis/`

See Also

\ `cancor()`

Examples

```
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])  # the PA tests
Y <- as.matrix(Rohwer[,3:5])   # the aptitude/ability variables

cc <- cancor(X, Y, set.names=c("PA", "Ability"))

redundancy(cc)
##
## Redundancies for the PA variables & total X canonical redundancy
##
##     Xcan1     Xcan2     Xcan3 total X|Y
##   0.17342   0.04211   0.00797   0.22350
##
## Redundancies for the Ability variables & total Y canonical redundancy
##
##     Ycan1     Ycan2     Ycan3 total Y|X
```

```
##    0.2249    0.0369    0.0156    0.2774
```

`reflect` *Reflect Columns in an Object, reversing the sign of all elements*

Description

`candisc` and `cancor` objects have coefficients for the X and Y and weighted scores for these, whose signs are arbitrary, in the sense that a given column can be reflected (multiplied by -1) without changing the fit. But, often you will want to reverse the direction of one or more dimensions for ease of interpretation.

This function allows you to reflect any columns of the variable coefficients (and corresponding observation scores). This is often useful for interpreting a biplot, for example when a component (often the first) has all negative signs.

Usage

```
reflect(object, columns = 1:2, ...)

## S3 method for class 'data.frame'
reflect(object, columns = 1:2, ...)

## S3 method for class 'cancor'
reflect(object, columns = 1:2, ...)

## S3 method for class 'candisc'
reflect(object, columns = 1:2, ...)
```

Arguments

`object`	An object whose columns are to be reflected
`columns`	a vector of indices of the columns to reflect
`...`	Unused

Details

`reflect` methods are available for:

- `data.frames`, for numeric columns
- `"cancor"` objects, for the coefficients and scores of the `X` and `Y` variables
- `"candisc"` objects, for the coefficients, structure correlations and scores

Note that `plot.candisc()` and `plot.candisc()` can handle this internally using the argument `rev.axes`.

Value

The object with specified columns of the appropriate components (variable coefficients, observation scores, ...) multiplied by -1.

Methods (by class)

- `reflect(data.frame)`: `"data.frame"` method.
- `reflect(cancor)`: `"cancor"` method.
- `reflect(candisc)`: `"candisc"` method.

Author(s)

Michael Friendly

See Also

ggbiplot::reflect has similar methods for PCA-like objects

Examples

```
# reflect cols in a data.frame
X <- data.frame(x1 = 1:4, x2 = 5:8)
reflect(X)
reflect(X, 1)
reflect(X, 2)
cbind (X, letters[1:4]) |> reflect(1)

# reflect a candisc
iris.mod <- lm(cbind(Petal.Length, Sepal.Length, Petal.Width, Sepal.Width) ~ Species, data=iris)
iris.can <- candisc(iris.mod, data=iris)
coef(iris.can)
# reflect Can1
iris.can |> reflect(1) |> coef()

# reflect a cancor
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])  # the PA tests
Y <- as.matrix(Rohwer[,3:5])   # the aptitude/ability variables
Rohwer.can <- cancor(X, Y, set.names=c("PA", "Ability"))
coef(Rohwer)
Rohwer.can |> reflect() |> coef()
```

`scores.lda` *Extract Observation Discriminant Scores for Linear Discriminant Analysis*

Description

This is a thin wrapper for `predict_discrim()` to provide a `scores()` method for discriminant analysis from `MASS::lda()`.

Usage

```
## S3 method for class 'lda'
scores(x, prior = x$prior, dimen, ...)
```

Arguments

`x`	An object of class `"lda"` such as results from `MASS::lda()`
`prior`	The prior probabilities of the classes. By default, taken to be the proportions in what was set in the call to `MASS::lda()`
`dimen`	The dimension of the space to be used. If this is less than the number of available dimensions, $min(p, ng - 1)$, only the first `dimen` discriminant components are used.
`...`	Unused; for compatibility with the generic

Value

a data frame for the observations with columns `LD1`, `LD2`, ... for the discriminant dimensions

Author(s)

Michael Friendly

See Also

`predict_discrim()`, `MASS::lda()`

Examples

```
library(MASS)   # for lda()

iris.lda <- lda(Species ~ ., iris)
scores(iris.lda) |>
   str()
```

`varOrder` *Order variables according to canonical structure or other criteria*

Description

The `varOrder` function implements some features of “effect ordering” (Friendly & Kwan (2003) for *variables* in a multivariate data display to make the displayed relationships more coherent.

This can be used in pairwise HE plots, scatterplot matrices, parallel coordinate plots, plots of multivariate means, and so forth.

For a numeric data frame, the most useful displays often order variables according to the angles of variable vectors in a 2D principal component analysis or biplot. For a multivariate linear model, the analog is to use the angles of the variable vectors in a 2D canonical discriminant biplot.

Usage

```
varOrder(x, ...)

## S3 method for class 'mlm'
varOrder(
  x,
  term,
  variables,
  type = c("can", "pc"),
  method = c("angles", "dim1", "dim2", "alphabet", "data", "colmean"),
  names = FALSE,
  descending = FALSE,
  ...
)

## S3 method for class 'data.frame'
varOrder(
  x,
  variables,
  method = c("angles", "dim1", "dim2", "alphabet", "data", "colmean"),
  names = FALSE,
  descending = FALSE,
  ...
)

## Default S3 method:
varOrder(x, ...)
```

Arguments

`x`	A multivariate linear model or a numeric data frame
`...`	Arguments passed to methods

`term`	For the `mlm` method, one term in the model for which the canonical structure coefficients are found.
`variables`	indices or names of the variables to be ordered; defaults to all response variables an MLM or all numeric variables in a data frame.
`type`	For an MLM, `type="can"` uses the canonical structure coefficients for the given `term`; `type="pc"` uses the principal component variable eigenvectors.
`method`	One of `c("angles", "dim1", "dim2", "alphabet", "data", "colmean")` giving the effect ordering method. **"angles"** Orders variables according to the angles their vectors make with dimensions 1 and 2, counter-clockwise starting from the lower-left quadrant in a 2D biplot or candisc display. **"dim1"** Orders variables in increasing order of their coordinates on dimension 1 **"dim2"** Orders variables in increasing order of their coordinates on dimension 2 **"alphabet"** Orders variables alphabetically **"data"** Uses the order of the variables in the data frame or the list of responses in the MLM **"colmean"** Uses the order of the column means of the variables in the data frame or the list of responses in the MLM
`names`	logical; if `TRUE` the effect ordered names of the variables are returned; otherwise, their indices in `variables` are returned.
`descending`	If `TRUE`, the ordered result is reversed to a descending order.

Value

A vector of integer indices of the variables or a character vector of their names.

Methods (by class)

- `varOrder(mlm)`: `"mlm"` method.
- `varOrder(data.frame)`: `"data.frame"` method.
- `varOrder(default)`: `"default"` method.

Author(s)

Michael Friendly

References

Friendly, M. & Kwan, E. (2003). Effect Ordering for Data Displays, *Computational Statistics and Data Analysis*, **43**, 509-539. doi:10.1016/S01679473(02)002906

Examples

```
data(Wine, package="candisc")
Wine.mod <- lm(as.matrix(Wine[, -1]) ~ Cultivar, data=Wine)
Wine.can <- candisc(Wine.mod)
plot(Wine.can, ellipse=TRUE)

# pairs.mlm HE plot, variables in given order
pairs(Wine.mod, fill=TRUE, fill.alpha=.1, var.cex=1.5)

order <- varOrder(Wine.mod)
pairs(Wine.mod, variables=order, fill=TRUE, fill.alpha=.1, var.cex=1.5)
```

`vecscale` *Scale vectors to fill the current plot*

Description

Calculates a scale factor so that a collection of vectors nearly fills the current plot, that is, the longest vector does not extend beyond the plot region.

Usage

```
vecscale(
  vectors,
  bbox = matrix(par("usr"), 2, 2),
  origin = c(0, 0),
  factor = 0.95
)
```

Arguments

`vectors`	a two-column matrix giving the end points of a collection of vectors
`bbox`	the bounding box of the containing plot region within which the vectors are to be plotted. The default is the bounding box of the current plot window, obtained from `par("usr")`.
`origin`	origin of the vectors. Defaults to (0, 0).
`factor`	maximum length of the rescaled vectors relative to the maximum possible

Details

This function is used in, e.g., `vectors()` to draw labeled vectors in a dimension-reduction plot. The scaling calculated here doesn't directly calculate space for the labels to fit within the plot regions. The `factor` argument can provide for that, shrinking the vectors by that factor.

Value

scale factor, the numeric multiplier of the vectors

Author(s)

Michael Friendly

See Also

`vectors()`, `plot.candisc()`, `heplot.candisc()`

Examples

```
bbox <- matrix(c(-3, 3, -2, 2), 2, 2)
colnames(bbox) <- c("x","y")
rownames(bbox) <- c("min", "max")
bbox

vecs <- matrix( runif(10, -1, 1), 5, 2)

plot(bbox)
arrows(0, 0, vecs[,1], vecs[,2], angle=10, col="red")
(s <- vecscale(vecs))
arrows(0, 0, s*vecs[,1], s*vecs[,2], angle=10)
```

`vectors` *Draw Labeled Vectors in 2D or 3D*

Description

Graphics utility functions to draw vectors from an origin to a collection of points (using `graphics::arrows()` in 2D or `rgl::lines3d()` in 3D) with labels for each (using `graphics::text()` or `rgl::texts3d()`

Usage

```
vectors(
  x,
  origin = c(0, 0),
  labels = rownames(x),
  scale = 1,
  col = "blue",
  lwd = 1,
  cex = 1,
  length = 0.1,
  angle = 13,
  pos = NULL,
  ...
)
```

Arguments

`x`	A two-column matrix or a three-column matrix containing the end points of the vectors
`origin`	Starting point(s) for the vectors
`labels`	Labels for the vectors
`scale`	A multiplier for the length of each vector
`col`	color(s) for the vectors.
`lwd`	line width(s) for the vectors.
`cex`	color(s) for the vectors.
`length`	For `vectors`, length of the edges of the arrow head (in inches).
`angle`	For `vectors`, angle from the shaft of the arrow to the edge of the arrow head.
`pos`	For `vectors`, position of the text label relative to the vector head. If `pos==NULL` (the default), labels are positioned labels outside, relative to arrow ends
`...`	other graphical parameters, such as `lty`, `xpd`, ...

Details

The graphical parameters `col`, `lty` and `lwd` can be vectors of length > 1 and will be recycled if necessary across the rows of `x` which define the vectors.

For use in high-level plots, `vecscale()` can be used to find a value for the `scale` argument to automatically re-scale the vectors to approximately fill the plot region.

The option `xpd = TRUE` can be passed to `vectors()` via the `...` argument to avoid labels being clipped.

Value

None

Author(s)

Michael Friendly

See Also

`graphics::arrows()`, `graphics::text()`, `graphics::segments()`

```
    [rgl::lines3d()], [rgl::texts3d()]
```

Examples

```
set.seed(1234)
plot(c(-3, 3), c(-3,3), type="n",
     xlab = "X", ylab = "Y")
X <- matrix(rnorm(10), ncol=2)
rownames(X) <- LETTERS[1:5]
vectors(X, scale=2, col=palette(), xpd = TRUE)
```

`Wilks` *Wilks Lambda Tests for Canonical Correlations*

Description

Tests the sequential hypotheses that the ith canonical correlation and all that follow it are zero,

$$\rho_i = \rho_{i+1} = \cdots = 0$$

Usage

```
Wilks(object, ...)

## S3 method for class 'cancor'
Wilks(object, ...)

## S3 method for class 'candisc'
Wilks(object, ...)
```

Arguments

`object`	An object of class `"cancor""` or `"candisc""`
`...`	Other arguments passed to methods (not used)

Details

Wilks' Lambda values are calculated from the eigenvalues and converted to F statistics using Rao's approximation.

Value

A data.frame (of class `"anova"`) containing the test statistics

Methods (by class)

- `Wilks(cancor)`: `"cancor"` method.
- `Wilks(candisc)`: `print()` method for `"candisc"` objects.

Author(s)

Michael Friendly

References

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

See Also

`cancor()`

Examples

```
data(Rohwer, package="heplots")
X <- as.matrix(Rohwer[,6:10])  # the PA tests
Y <- as.matrix(Rohwer[,3:5])   # the aptitude/ability variables

cc <- cancor(X, Y, set.names=c("PA", "Ability"))
Wilks(cc)

iris.mod <- lm(cbind(Petal.Length, Sepal.Length, Petal.Width, Sepal.Width) ~ Species, data=iris)
iris.can <- candisc(iris.mod, data=iris)
Wilks(iris.can)
```

`Wine` *Chemical composition of three cultivars of wine*

Description

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

Format

A data frame with 178 observations on the following 14 variables.

`Cultivar`	a factor with levels `barolo` `grignolino` `barbera`
`Alcohol`	a numeric vector
`MalicAcid`	a numeric vector
`Ash`	a numeric vector
`AlcAsh`	a numeric vector, Alkalinity of ash
`Mg`	a numeric vector, Magnesium
`Phenols`	a numeric vector, Total phenols
`Flav`	a numeric vector, Flavanoids
`NonFlavPhenols`	a numeric vector
`Proa`	a numeric vector, Proanthocyanins
`Color`	a numeric vector, color intensity
`Hue`	a numeric vector
`OD`	a numeric vector, OD280/OD315 of diluted wines
`Proline`	a numeric vector

Details

This data set is a classic in the machine learning literature as an easy high-D classification problem, but is also of interest for examples of MANOVA and discriminant analysis.

The precise definitions of these variables is unknown: units, how they were measured, etc.

Source

This data set was obtained from the UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/datasets/Wine`. This page references a large number of papers that use this data set to compare different methods.

References

In R, a comparable data set is contained in the **ggbiplot** package.

Examples

```
data(Wine)
str(Wine)
#summary(Wine)

Wine.mlm <- lm(as.matrix(Wine[, -1]) ~ Cultivar, data=Wine)
Wine.can <- candisc(Wine.mlm)
Wine.can


plot(Wine.can, ellipse=TRUE)
plot(Wine.can, which=1)
```

`Wolves` *Wolf skulls*

Description

Skull morphometric data on Rocky Mountain and Arctic wolves (Canis Lupus L.) taken from Morrison (1990), originally from Jolicoeur (1959).

Format

A data frame with 25 observations on the following 11 variables.

`group` a factor with levels `ar:f` `ar:m` `rm:f` `rm:m`, comprising the combinations of `location` and `sex`

`location` a factor with levels `ar`=Arctic, `rm`=Rocky Mountain

`sex` a factor with levels `f`=female, `m`=male

`x1`	palatal length, a numeric vector
`x2`	postpalatal length, a numeric vector
`x3`	zygomatic width, a numeric vector
`x4`	palatal width outside first upper molars, a numeric vector
`x5`	palatal width inside second upper molars, a numeric vector
`x6`	postglenoid foramina width, a numeric vector
`x7`	interorbital width, a numeric vector
`x8`	braincase width, a numeric vector
`x9`	crown length, a numeric vector

Details

All variables are expressed in millimeters.

The goal was to determine how geographic and sex differences among the wolf populations are determined by these skull measurements. For MANOVA or (canonical) discriminant analysis, the factors `group` or `location` and `sex` provide alternative parameterizations.

Source

Morrison, D. F. *Multivariate Statistical Methods*, (3rd ed.), 1990. New York: McGraw-Hill, p. 288-289.

% ~~ reference to a publication or URL from which the data were obtained ~~

References

Jolicoeur, P. “Multivariate geographical variation in the wolf *Canis lupis L.*”, *Evolution*, XIII, 283–299.

Examples

```
data(Wolves)

# using group
wolf.mod <-lm(cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9) ~ group, data=Wolves)
car::Anova(wolf.mod)

wolf.can <-candisc(wolf.mod)
plot(wolf.can)
heplot(wolf.can)

# using location, sex
wolf.mod2 <-lm(cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9) ~ location*sex, data=Wolves)
car::Anova(wolf.mod2)

wolf.can2 <-candiscList(wolf.mod2)
plot(wolf.can2)
```

Index