Package 'klaR'

July 22, 2025

Version 1.7-3

Date 2023-12-13

Title Classification and Visualization

Author Christian Roever, Nils Raabe, Karsten Luebke, Uwe Ligges, Gero Szepannek, Marc Zentgraf, David Meyer

Maintainer Uwe Ligges <ligges@statistik.tu-dortmund.de>

SystemRequirements SVMlight

Suggests scatterplot3d (>= 0.3-22), som, mlbench, rpart, e1071

Enhances clustMixType, randomForest, ClustVarLV

Depends R (>= 2.10.0), MASS

Imports combinat, questionr, grDevices, stats, utils, graphics

Description Miscellaneous functions for classification and visualization,
e.g. regularized discriminant analysis, sknn() kernel-density naive Bayes,
an interface to 'svmlight' and stepclass() wrapper variable selection
for supervised classification, partimat() visualization of classification rules
and shardsplot() of cluster results as well as kmodes() clustering for categorical data,
corclust() variable clustering, variable extraction from different variable clustering models
and weight of evidence preprocessing.

License GPL-2 | GPL-3

URL https://statistik.tu-dortmund.de

NeedsCompilation no

Repository CRAN

Date/Publication 2023-12-13 22:30:03 UTC

Contents

b.scal 3
B3 4
benchB3 5
betascale 7

calc.trans . . . 7
centerlines . . . 8
classscatter . . . 9
cond.index . . . 10
corclust . . . 11
countries . . . 12
cvtree . . . 13
dkernel . . . 14
drawparti . . . 15
e.scal . . . 16
EDAM . . . 17
errormatrix . . . 20
friedman.data . . . 21
GermanCredit . . . 23
greedy.wilks . . . 24
hmm.sop . . . 26
kmodes . . . 27
loclda . . . 28
locpvs . . . 31
meclight.default . . . 33
NaiveBayes . . . 34
nm . . . 36
partimat . . . 37
plineplot . . . 39
plot.NaiveBayes . . . 40
plot.woe . . . 41
predict.loclda . . . 42
predict.locpvs . . . 43
predict.meclight . . . 44
predict.NaiveBayes . . . 45
predict.pvs . . . 46
predict.rda . . . 47
predict.sknn . . . 48
predict.svmlight . . . 49
predict.woe . . . 50
pvs . . . 51
quadplot . . . 54
rda . . . 56
shardsplot . . . 59
sknn . . . 62
stepclass . . . 63
svmlight . . . 65
triframe . . . 68
trigrid . . . 68
triperplines . . . 70
triplot . . . 71
tripoints . . . 72
tritrafo . . . 73

ucpm 75
woe 76
xtractvars 78

Index **79**

`b.scal` *Calculation of beta scaling parameters*

Description

Calculates the scaling parameter for `betascale`.

Usage

```
b.scal(member, grouping, dis = FALSE, eps = 1e-04)
```

Arguments

`member`	Membership values of an argmax classification method. Eg. posterior probabilities of `lda`. Row-wise values must sum up to 1 and must be in the interval [0,1].
`grouping`	Class vector.
`dis`	Logical, whether to optimize the dispersion parameter in `pbeta`.
`eps`	Minimum variation of membership values. If variance is smaller than `eps`, the values are treated as one point.

Details

With `betascale` and `b.scal`, membership values of an argmax classifier are scaled in such a way, that the mean membership value of those values which are assigned to each class reflect the mean correctness rate of that values. This is done via `qbeta` and `pbeta` with the appropriate shape parameters. If `dis` is `TRUE`, it is tried that the variation of membership values is optimal for the accuracy relative to the correctness rate. If the variation of the membership values is less than `eps`, they are treated as one point and shifted towards the correctness rate.

Value

A list containing

`model`	Estimated parameters for `betascale`.
`eps`	Value of `eps` from the call.
`member`	Scaled membership values.

Author(s)

Karsten Luebke (<karsten.luebke@fom.de>), Uwe Ligges

References

Garczarek, Ursula Maria (2002): Classification rules in standardized partition spaces. Dissertation, University of Dortmund. URL http://hdl.handle.net/2003/2789

See Also

betascale, e.scal

Examples

```
library(MASS)
data(B3)
pB3 <- predict(lda(PHASEN ~ ., data = B3))$posterior
pbB3 <- b.scal(pB3, B3$PHASEN, dis = TRUE)
ucpm(pB3, B3$PHASEN)
ucpm(pbB3$member, B3$PHASEN)
```

B3 *West German Business Cycles 1955-1994*

Description

West German Business Cycles 1955-1994

Usage

```
data(B3)
```

Format

A data frame with 157 observations on the following 14 variables.

PHASEN a factor with levels 1 (upswing), 2 (upper turning points), 3 (downswing), and 4 (lower turning points).

BSP91JW GNP (y)

CP91JW Private Consumption (y)

DEFRATE Government deficit (percent of GNP)

EWAJW Wage and salary earners (y)

EXIMRATE Net exports as (percent of GNP)

GM1JW Money supply M1 (y)

IAU91JW Investment in equipment (y)

IB91JW Investment in construction (y)

LSTKJW Unit labor cost (y)

PBSPJW GNP price deflator (y)

PCPJW Consumer price index (y)

ZINSK Short term interest rate (nominal)

ZINSLR Long term interest rate (real)

where (y) stands for "yearly growth rates".

Note that years and corresponding year quarters are given in the row names of the data frame, e.g. "1988,3" for the third quarter in 1988.

Details

The West German Business Cycles data (1955-1994) is analyzed by the project *B3* of the SFB475 (Collaborative Research Centre "Reduction of Complexity for Multivariate Data Structures"), supported by the Deutsche Forschungsgemeinschaft.

Source

RWI (Rheinisch Westfälisches Institut für Wirtschaftsforschung), Essen, Germany.

References

Heilemann, U. and Münch, H.J. (1996): West German Business Cycles 1963-1994: A Multivariate Discriminant Analysis. *CIRET–Conference in Singapore, CIRET–Studien* 50.

See Also

For benchmarking on this data see also benchB3

Examples

```
data(B3)
summary(B3)
```

---

benchB3 *Benchmarking on B3 data*

Description

Evaluates the performance of a classification method on the B3 data.

Usage

```
benchB3(method, prior = rep(1/4, 4), sv = "4", scale = FALSE, ...)
```

Arguments

method	classification method to use
prior	prior probabilities of classes
sv	class of the start of a business cycle
scale	logical, whether to use scale first
...	furhter arguments passed to method

Details

The performance of classification methods on cyclic data can be measured by a special form of cross-validation: Leave-One-Cycle-Out. That means that a complete cycle is used as test data and the others are used as training data. This is repeated for all complete cycles in the data.

Value

A list with elements

MODEL	list with the model returned by method of the training data
error	vector of test error rates in cycles
l1co.error	leave-one-cycle-out error rate

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

B3

Examples

```
perLDA <- benchB3("lda")
## Not run:
## due to parameter optimization rda takes a while
perRDA <- benchB3("rda")
library(rpart)
## rpart will not work with prior argument:
perRpart <- benchB3("rpart", prior = NULL)

## End(Not run)
```

betascale *Scale membership values according to a beta scaling*

Description

Performs the scaling for beta scaling learned by `b.scal`.

Usage

```
betascale(betaobj, member)
```

Arguments

`betaobj`	A model learned by `b.scal`.
`member`	Membership values to be scaled.

Details

See `b.scal`.

Value

A matrix with the scaled membership values.

See Also

`b.scal`, `e.scal`

Examples

```
library(MASS)
data(B3)
pB3 <- predict(lda(PHASEN ~ ., data = B3))$posterior
pbB3 <- b.scal(pB3, B3$PHASEN)
betascale(pbB3)
```

calc.trans *Calculation of transition probabilities*

Description

Function to estimate the probabilities of a time series to stay or change the state.

Usage

```
calc.trans(x)
```

Arguments

x	(factor) vector of states

Details

To estimate the transition probabilities the empirical frequencies are counted.

Value

The transition probabilities matrix. `x[i,j]` is the probability to change from state `i` to state `j`.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

Examples

```
data(B3)
calc.trans(B3$PHASEN)
```

---

centerlines *Lines from classborders to the center*

Description

Function which constructs the lines from the borders between two classes to the center. To be used in connection with `triplot` and `quadplot`.

Usage

```
centerlines(n)
```

Arguments

n	number of classes. Meaningful are 3 or 4.

Value

a matrix with n-columns.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

`triplot`, `quadplot`

Examples

```
centerlines(3)
centerlines(4)
```

`classscatter` *Classification scatterplot matrix*

Description

Function to plot a scatterplot matrix with a classification result.

Usage

```
classscatter(formula, data, method, col.correct = "black",
    col.wrong = "red", gs = NULL, ...)
```

Arguments

`formula`	formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
`data`	Data frame from which variables specified in formula are preferentially to be taken.
`method`	character, name of classification function (e.g. "`lda`").
`col.correct`	color to use for correct classified objects.
`col.wrong`	color to use for missclassified objects.
`gs`	group symbol (plot character), must have the same length as the data. If `NULL`, `as.character(groups)` is the default.
`...`	further arguments passed to the underlying classification method or plot functions.

Value

The actual error rate.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

`plot`

Examples

```
data(B3)
library(MASS)
classscatter(PHASEN ~ BSP91JW + EWAJW + LSTKJW,
    data = B3, method = "lda")
```

cond.index *Calculation of Condition Indices for Linear Regression*

Description

Diagnosis of collinearity in X

Usage

```
cond.index(formula, data, ...)
```

Arguments

formula	formula of the form 'groups ~ x1 + x2 + ...'
data	data frame (or matrix) containing the explanatory variables
...	further arguments to be passed to lm

Details

Collinearities can inflate the variance of the estimated regression coefficients and numerical stability. The condition indices are calculated by the eigenvalues of the crossproduct matrix of the scaled but uncentered explanatory variables. Indices > 30 may indicate collinearity.

Value

A vector of the condition indices.

Author(s)

Andrea Preusser, Karsten Luebke (<karsten.luebke@fom.de>)

References

Belsley, D. , Kuh, E. and Welsch, R. E. (1979), *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, John Wiley (New York)

See Also

stepclass, manova

Examples

```
data(Boston)
condition_medv <- cond.index(medv ~ ., data = Boston)
condition_medv
```

`corclust` *Function to identify groups of highly correlated variables for removing correlated features from the data for further analysis.*

Description

A hierarchical clustering of variables using `hclust` is performed using 1 - the absolute correlation as a distance measure between tow variables.

Usage

```
corclust(x, cl = NULL, method = "complete")
## S3 method for class 'corclust'
plot(x, selection = "both", mincor = NULL, ...)
```

Arguments

`x`	Either a data frame or a matrix consisting of numerical attributes.
`cl`	Optional vector of ty factor indicating class levels, if class specific correlations should to be considered.
`method`	Linkage to be used for clustering. Default is `complete` linkage.
`selection`	If `"numeric"`, '1 - average absolute correlation within cluster' is plotted, if `"factor"`, '1 - minimum Cramer's V within cluster' is plotted. The default, `"both"`, generates both variations.
`mincor`	Adds a horizontal line for this correlation.
`...`	passed to underlying plot functions.

Details

Each cluster consists of a set of correlated variables according to the chosen clustering criterion. The default criterion is '`complete`'. This choice is meaningful as it represents the *minimum absolute correlation* between all variables of a cluster.
The data set is split into numerics and factors two separate clustering models are built, depending on the variable type. For factors distances are computed based on 1-Cramer's V statistic using `chisq.test`. For a large number of factor variables this might take some time. The resulting trees can be plotted using `plot`.
Further proceeding would consist in chosing one variable of each cluster to obtain a subset of rather uncorrelated variables for further analysis. An automatic variable selection can be done using `cvtree` and `xtractvars`.
If an additional class vector `cl` is given to the function for any two variables their minimum correlation over all classes is used.

Value

Object of class `corclust`.

`cor`	Correlation matrix of numeric variables.
`crv`	Matrix of Cramer's V for factor variables.
`cluster.numerics`	Resulting hierarchical `hclust` model for numeric variables.
`cluster.factors`	Resulting hierarchical `hclust` model for factor variables.
`id.numerics`	Variable IDs of numeric variables in `x`.
`id.factors`	Variable IDs of factor variables `x`.

Author(s)

Gero Szepannek

References

Roever, C. and Szepannek, G. (2005): Application of a genetic algorithm to variable selection in fuzzy clustering. In C. Weihs and W. Gaul (eds), Classification - The Ubiquitous Challenge, 674-681, Springer.

See Also

`plot.corclust` and `hclust` for details on the clustering algorithm, and `cvtree`, `xtractvars` for postprocessing.

Examples

```
data(iris)
classes <- iris$Species
variables <- iris[,1:4]
ccres <- corclust(variables, classes)
plot(ccres, mincor = 0.6)
```

---

countries *Socioeconomic data for the most populous countries.*

Description

Socioeconomic data for the most populous countries.

Usage

```
data(countries)
```

Format

A data frame with 42 observations on the following 7 variables.

Country name of the country.

Popul population.

PopDens population density.

GDPpp GDP per inhabitant.

LifeEx mean life expectation

InfMor infant mortality

Illit illiteracy rate

Source

CIA World Factbook https://www.cia.gov/the-world-factbook/

Examples

```
data(countries)
summary(countries)
```

---

cvtree *Extracts variable cluster IDs*

Description

Extracts cluster IDs for variables according to a dendrogram from object of class cvtree.

Usage

```
cvtree(object, k = 2, mincor = NULL, ...)
```

Arguments

object	Object of class corclust.
k	Number of clusters to be extracted from dendrogram.
mincor	Minimum within cluster correlation. Can be specified alternatively to k.
...	Currently not used.

Details

Like in corclust for correlation comparison numerics and factors are considered separately. For factors Cramer's V statistic is used.

Value

Object of class `cvtree` with elements:

`cluster`	Vector of cluster IDs.
`correlations`	Matrix of average within cluster correlations and average corrleation to all variables of the closest cluster as well as the ID of the closest cluster. For factor variables Cramer's V is computed.

Author(s)

Gero Szepannek

References

Roever, C. and Szepannek, G. (2005): Application of a genetic algorithm to variable selection in fuzzy clustering. In C. Weihs and W. Gaul (eds), Classification - The Ubiquitous Challenge, 674-681, Springer.

See Also

See also `corclust`, `plot.corclust` and `hclust` for details on the clustering algorithm.

Examples

```
data(B3)
ccres <- corclust(B3)
plot(ccres)
cvtree(ccres, k = 3)
```

---

dkernel *Estimate density of a given kernel*

Description

Given an estimated kernel density this function estimates the density of a new vector.

Usage

```
dkernel(x, kernel = density(x), interpolate = FALSE, ...)
```

Arguments

`x`	vector of which the density should be estimated
`kernel`	object of `class` `density`
`interpolate`	Interpolate or use `density` of nearest point?
`...`	currently not used.

Value

Denstiy of x in kernel.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

density, NaiveBayes

Examples

```
kern <- density(rnorm(50))
x <- seq(-3, 3, len = 100)
y <- dkernel(x, kern)
plot(x, y, type = "l")
```

---

drawparti *Plotting the 2-d partitions of classification methods*

Description

Plot showing the classification of observations based on classification methods (e.g. lda, qda) for two variables. Moreover, the classification borders are displayed and the apparent error rates are given in each title.

Usage

```
drawparti(grouping, x, y, method = "lda", prec = 100, xlab = NULL,
    ylab = NULL, col.correct = "black", col.wrong = "red",
    col.mean = "black", col.contour = "darkgrey",
    gs = as.character(grouping), pch.mean = 19, cex.mean = 1.3,
    print.err = 0.7, legend.err = FALSE, legend.bg = "white",
    imageplot = TRUE, image.colors = cm.colors(nc),
    plot.control = list(), ...)
```

Arguments

grouping	factor specifying the class for each observation.
x	first explanatory vector.
y	second explanatory vector.
method	the method the classification is based on, currently supported are: lda, qda, rpart, naiveBayes, rda, sknn and svmlight
.	

prec	precision used to draw the classification borders (the higher the more precise; default: 100).
xlab	a title for the x axis.
ylab	a title for the y axis.
col.correct	color for correct classified objects.
col.wrong	color for wrong classified objects.
col.mean	color for class means (only for methods lda and qda).
col.contour	color of the contour lines (if imageplot = FALSE).
gs	group symbol (plot character), must have the same length as grouping.
pch.mean	plot character for class means (only for methods lda and qda).
cex.mean	character expansion for class means (only for methods lda and qda).
print.err	character expansion for text specifying the apparent error rate. If print.err = 0, nothing is printed.
legend.err	logical; whether to plot the apparent error rate above the plot (if FALSE), or into a legend into the upper right corner of the plot (if TRUE). This argument is ignored, if print.err = 0, i.e. if no error rate is printed.
legend.bg	Backgound colour to use for the legend.
imageplot	logical; whether to use an image plot or contour lines.
image.colors	colors used for the imageplot, if TRUE.
plot.control	A list containing further arguments passed to the underlying plot functions.
...	Further arguments passed to the classification method.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>, Uwe Ligges, Irina Czogiel

See Also

partimat

---

e.scal *Function to calculate e- or softmax scaled membership values*

Description

Calculates the e- or softmax scaled membership values of an argmax based classification rule.

Usage

```
e.scal(x, k = 1, tc = NULL)
```

Arguments

x	matrix of membership values
k	parameter for e-scaling (1 for softmax)
tc	vector of true classes (required if k has to be optimized)

Details

For any membership vector y $\exp(y \cdot k)/\sum \exp(y \cdot k)$ is calculated. If k=1, the classical softmax scaling is used. If the true classes are given, k is optimized so that the apparent error rate is minimized.

Value

A list containing elements

sv	Scaled values
k	Optimal k

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

References

Garczarek, Ursula Maria (2002): Classification rules in standardized partition spaces. Dissertation, University of Dortmund. URL http://hdl.handle.net/2003/2789

Examples

```
library(MASS)
data(iris)
ldaobj <- lda(Species ~ ., data = iris)
ldapred <- predict(ldaobj)$posterior
e.scal(ldapred)
e.scal(ldapred, tc = iris$Species)
```

EDAM *Computation of an Eight Direction Arranged Map*

Description

Produces an object of class EDAM which is a two dimensional representation of data in a rectangular, equally spaced grid as known from Self-Organizing Maps.

Usage

```
EDAM(EV0, nzx = 0, iter.max = 10, random = TRUE, standardize = FALSE,
    wghts = 0, classes = 0, sa = TRUE, temp.in = 0.5, temp.fin = 1e-07,
    temp.gamma = 0)
```

Arguments

EV0	either a symmetric dissimilarity matrix or a matrix of arbitrary dimensions whose n rows correspond to cases and whose k columns correspond to variables.
nzx	an integer specifying the number of vertical bars in the grid. By default, nzx is chosen automatically, so that the grid gets closest do a square. If n is no multiple of nzx, all surplus objects are skipped.
iter.max	an integer giving the maxmimum number of iterations to perform for the same neighborhood size.
random	logical. If TRUE, the initital order is drawn from a uniform distribution.
standardize	logical. If TRUE, the measurements in EV0 are standardized before calculating Euclidean distances. Measurements are standardized for each variable by dividing by the variable's standard deviation. Meaningless if EV0 is a dissimilarity matrix.
wghts	an optional vector of length k giving relative weights of the variables in computing Euclidean distances. Meaningless if EV0 is a dissimilarity matrix.
classes	an optional vector of length n specifying the membership to classes for all objects.
sa	logical. If TRUE, the optimization is obtained by Simulated Annealing.
temp.in	numeric giving the initial temperature, if sa is set to TRUE.
temp.fin	numeric giving the final temperature, if sa is set to TRUE. Meaningless if temp.gamma is greater than 0.
temp.gamma	numeric giving the relative change of the temperature from one iteration to the other, if sa is set to TRUE.

Details

The data given by EV0 is visualized by the EDAM-algorithm. This method approximates the best visualization where goodness is measured by S, a transformation of the criterion stress as i.e. known from sammon. The target space of the visualization is restricted to a grid so the problem has a discrete solution space. Originally this restriction was made to make the results comparable to those of Kohonen Self-Organizing Maps. But it turns out that also for reasons of a clear arrangement the representation in a grid can be more favorable than in the hole plane.

During the computation of EDAM 3 values indicating its progress are given online. The first is the number of the actual iteration, the second the maximum number of overall performed iterations. The latter may reduce during computation, since the neighborhood reduces in case of convergence before the last iteration. The last number gives the actual criterion S. The default plot method plot.edam for objects of class EDAM is shardsplot.

Value

EDAM returns an object of class EDAM, which is a list containing the following components:

preimages	the re-ordered data; the position of the i-th object is where Z equals i.
Z	a matrix representing the positions of the preimages in the grid by their numbers.

`Z.old.terms`	a matrix representing the positions of the data in original order in the grid by their numbers.
`cl.ord`	a vector giving the re-ordered classes. All elements equal 1 if argument `classes` is undefined.
`S`	the criterion of the map

Author(s)

Nils Raabe

References

Raabe, N. (2003). *Vergleich von Kohonen Self-Organizing-Maps mit einem nichtsimultanen Klassifikations- und Visualisierungsverfahren*. Diploma Thesis, Department of Statistics, University of Dortmund.

See Also

`shardsplot`, `TopoS`

Examples

```
# Compute an Eight Directions Arranged Map for a random sample
# of the iris data.
data(iris)
set.seed(1234)
iris.sample <- sample(150, 42)
irisEDAM <- EDAM(iris[iris.sample, 1:4], classes = iris[iris.sample, 5],
    standardize = TRUE, iter.max = 3)
plot(irisEDAM, vertices = FALSE)
legend(3, 5, col = rainbow(3), legend = levels(iris[,5]), pch = 16)
print(irisEDAM)

# Construct clusters within the phases of the german business data
# and visualize the centroids by EDAM.
data(B3)
phasemat <- lapply(1:4, function(x) B3[B3[,1] == x, 2:14])
subclasses <- lapply(phasemat,
    function(x) cutree(hclust(dist(x)), k = round(nrow(x) / 4.47)))
centroids <- lapply(1:4,
    function(y) apply(phasemat[[y]], 2,
        function(x) by(x, subclasses[[y]], mean)))
centmat <- matrix(unlist(sapply(centroids, t)), ncol = 13,
    byrow = TRUE, dimnames = list(NULL, colnames(centroids[[1]])))
centclasses <- unlist(lapply(1:4,
    function(x) rep(x, unlist(lapply(centroids, nrow))[x])))
B3EDAM <- EDAM(centmat, classes = centclasses, standardize = TRUE,
    iter.max = 6, rand = FALSE)
plot(B3EDAM, standardize = TRUE)
opar <- par(xpd = NA)
legend(4, 5.1, col = rainbow(4), pch = 16, xjust = 0.5, yjust = 0,
    ncol = 2, legend = c("upswing", "upper turning point",
```

```
                          "downswing", "lower turning point"))
print(B3EDAM)
par(opar)
```

`errormatrix` *Tabulation of prediction errors by classes*

Description

Cross-tabulates true and predicted classes with the option to show relative frequencies.

Usage

```
errormatrix(true, predicted, relative = FALSE)
```

Arguments

`true`	Vector of true classes.
`predicted`	Vector of predicted classes.
`relative`	Logical. If `TRUE` rows are normalized to show relative frequencies (see below).

Details

Given vectors of true and predicted classes, a (symmetric) table of misclassifications is constructed.

Element [i,j] shows the number of objects of class i that were classified as class j; so the main diagonal shows the correct classifications. The last row and column show the corresponding sums of misclassifications, the lower right element is the total sum of misclassifications.

If '`relative`' is `TRUE`, the *rows* are normalized so they show relative frequencies instead. The lower right element now shows the total error rate, and the remaining last row sums up to one, so it shows "where the misclassifications went".

Value

A (named) matrix.

Note

Concerning the case that '`relative`' is `TRUE`:

If a prior distribution over the classes is given, the misclassification rate that is returned as the lower right element (which is only the fraction of misclassified *data*) is not an estimator for the expected misclassification rate.

In that case you have to multiply the individual error rates for each class (returned in the last column) with the corresponding prior probabilities and sum these up (see example below).

Both error rate estimates are equal, if the fractions of classes in the data are equal to the prior probabilities.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

table

Examples

```
data(iris)
library(MASS)
x <- lda(Species ~ Sepal.Length + Sepal.Width, data=iris)
y <- predict(x, iris)

# absolute numbers:
errormatrix(iris$Species, y$class)

# relative frequencies:
errormatrix(iris$Species, y$class, relative = TRUE)

# percentages:
round(100 * errormatrix(iris$Species, y$class, relative = TRUE), 0)

# expected error rate in case of class prior:
indiv.rates <- errormatrix(iris$Species, y$class, relative = TRUE)[1:3, 4]
prior <- c("setosa" = 0.2, "versicolor" = 0.3, "virginica" = 0.5)
total.rate <- t(indiv.rates) %*% prior
total.rate
```

---

friedman.data *Friedman's classification benchmark data*

Description

Function to generate 3-class classification benchmarking data as introduced by J.H. Friedman (1989)

Usage

```
friedman.data(setting = 1, p = 6, samplesize = 40, asmatrix = FALSE)
```

Arguments

setting	the problem setting (integer 1,2,...,6).
p	number of variables (6, 10, 20 or 40).
samplesize	sample size (number of observations, >=6).
asmatrix	if TRUE, results are returned as a matrix, otherwise as a data frame (default).

Details

When J.H. Friedman introduced the Regularized Discriminant Analysis (rda) in 1989, he used artificially generated data to test the procedure and to examine its performance in comparison to Linear and Quadratic Discriminant Analysis (see also lda and qda).

6 different settings were considered to demonstrate potential strengths and weaknesses of the new method:

1. equal spherical covariance matrices,
2. unequal spherical covariance matrices,
3. equal, highly ellipsoidal covariance matrices with mean differences in low-variance subspace,
4. equal, highly ellipsoidal covariance matrices with mean differences in high-variance subspace,
5. unequal, highly ellipsoidal covariance matrices with zero mean differences and
6. unequal, highly ellipsoidal covariance matrices with nonzero mean differences.

For each of the 6 settings data was generated with 6, 10, 20 and 40 variables.

Classification performance was then measured by repeatedly creating training-datasets of 40 observations and estimating the misclassification rates by test sets of 100 observations.

The number of classes is always 3, class labels are assigned randomly (with equal probabilities) to observations, so the contributions of classes to the data differs from dataset to dataset. To make sure covariances can be estimated at all, there are always at least two observations from each class in a dataset.

Value

Depending on `asmatrix` either a data frame or a matrix with `samplesize` rows and `p+1` columns, the first column containing the class labels, the remaining columns being the variables.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

References

Friedman, J.H. (1989): Regularized Discriminant Analysis. In: *Journal of the American Statistical Association* 84, 165-175.

See Also

rda

Examples

```
# Reproduce the 1st setting with 6 variables.
# Error rate should be somewhat near 9 percent.
training <- friedman.data(1, 6, 40)
x <- rda(class ~ ., data = training, gamma = 0.74, lambda = 0.77)
test <- friedman.data(1, 6, 100)
y <- predict(x, test[,-1])
errormatrix(test[,1], y$class)
```

GermanCredit *Statlog German Credit*

Description

The dataset contains data of past credit applicants. The applicants are rated as *good* or *bad*. Models of this data can be used to determine if new applicants present a *good* or *bad* credit risk.

Usage

```
data("GermanCredit")
```

Format

A data frame containing 1,000 observations on 21 variables.

status factor variable indicating the status of the existing checking account, with levels `... < 100 DM`, `0 <= ... < 200 DM`, `... >= 200 DM/salary for at least 1 year` and `no checking account`.

duration duration in months.

credit_history factor variable indicating credit history, with levels `no credits taken/all credits paid back duly`, `all credits at this bank paid back duly`, `existing credits paid back duly till now`, `delay in paying off in the past` and `critical account/other credits existing`.

purpose factor variable indicating the credit's purpose, with levels `car (new)`, `car (used)`, `furniture/equipment`, `radio/television`, `domestic appliances`, `repairs`, `education`, `retraining`, `business` and `others`.

amount credit amount.

savings factor. savings account/bonds, with levels `... < 100 DM`, `100 <= ... < 500 DM`, `500 <= ... < 1000 DM`, `... >= 1000 DM` and `unknown/no savings account`.

employment_duration ordered factor indicating the duration of the current employment, with levels `unemployed`, `... < 1 year`, `1 <= ... < 4 years`, `4 <= ... < 7 years` and `... >= 7 years`.

installment_rate installment rate in percentage of disposable income.

personal_status_sex factor variable indicating personal status and sex, with levels `male:divorced/separated`, `female:divorced/separated/married`, `male:single`, `male:married/widowed` and `female:single`.

other_debtors factor. Other debtors, with levels `none`, `co-applicant` and `guarantor`.

present_residence present residence since?

property factor variable indicating the client's highest valued property, with levels `real estate`, `building society savings agreement/life insurance`, `car` or `other` and `unknown/no property`.

age client's age.

other_installment_plans factor variable indicating other installment plans, with levels `bank`, `stores` and `none`.

housing factor variable indicating housing, with levels `rent`, `own` and `for free`.

number_credits number of existing credits at this bank.

job factor indicating employment status, with levels `unemployed/unskilled - non-resident`, `unskilled - resident`, `skilled employee/official` and `management/self-employed/highly qualified employee/officer`.

people_liable Number of people being liable to provide maintenance.

telephone binary variable indicating if the customer has a registered telephone number.

foreign_worker binary variable indicating if the customer is a foreign worker.

credit_risk binary variable indicating credit risk, with levels `good` and `bad`.

Source

The original data was provided by:

Professor Dr. Hans Hofmann, Institut fuer Statistik und Oekonometrie, Universitaet Hamburg, FB Wirtschaftswissenschaften, Von-Melle-Park 5, 2000 Hamburg 13

The dataset has been taken from the UCI Repository Of Machine Learning Databases at http://archive.ics.uci.edu/ml/.

It was published this way in CRAN package evtree (maintainer: Thomas Grubinger) that has been archived from CRAN on May 31, 2014. Afterwards the exactly same data object has been copied from the evtree package to klaR.

`greedy.wilks` *Stepwise forward variable selection for classification*

Description

Performs a stepwise forward variable/model selection using the Wilk's Lambda criterion.

Usage

```
greedy.wilks(X, ...)
## Default S3 method:
greedy.wilks(X, grouping, niveau = 0.2, ...)
## S3 method for class 'formula'
greedy.wilks(formula, data = NULL, ...)
```

Arguments

`X`	matrix or data frame (rows=cases, columns=variables)
`grouping`	class indicator vector
`formula`	formula of the form 'groups ~ x1 + x2 + ...'
`data`	data frame (or matrix) containing the explanatory variables
`niveau`	level for the approximate F-test decision
`...`	further arguments to be passed to the default method, e.g. `niveau`

Details

A stepwise forward variable selection is performed. The initial model is defined by starting with the variable which separates the groups most. The model is then extended by including further variables depending on the Wilk's lambda criterion: Select the one which minimizes the Wilk's lambda of the model including the variable if its p-value still shows statistical significance.

Value

A list of two components, a `formula` of the form '`response ~ list + of + selected + variables`', and a data.frame `results` containing the following variables:

`vars` the names of the variables in the final model in the order of selection.

`Wilks.lambda` the appropriate Wilks' lambda for the selected variables.

`F.statistics.overall`
the approximated F-statistic for the so far selected model.

`p.value.overall`
the appropriate p-value of the F-statistic.

`F.statistics.diff`
the approximated F-statistic of the partial Wilks's lambda (for comparing the model including the new variable with the model not including it).

`p.value.diff` the appropriate p-value of the F-statistic of the partial Wilk's lambda.

Author(s)

Andrea Preusser, Karsten Luebke (<karsten.luebke@fom.de>)

References

Mardia, K. V. , Kent, J. T. and Bibby, J. M. (1979), *Multivariate analysis*, Academic Press (New York; London)

See Also

`stepclass`, `manova`

Examples

```
data(B3)
gw_obj <- greedy.wilks(PHASEN ~ ., data = B3, niveau = 0.1)
gw_obj
## now you can say stuff like
## lda(gw_obj$formula, data = B3)
```

hmm.sop *Calculation of HMM Sum of Path*

Description

A Hidden Markov Model for the classification of states in a time series. Based on the transition probabilities and the so called emission probabilities ($p(class|x)$) the 'prior probablilities' of states (classes) in time period *t* given all past information in time period *t* are calculated.

Usage

```
hmm.sop(sv, trans.matrix, prob.matrix)
```

Arguments

sv	state at time 0
trans.matrix	matrix of transition probabilities
prob.matrix	matrix of $p(class|x)$

Value

Returns the 'prior probablilities' of states.

Author(s)

Daniel Fischer, Reinald Oetsch

References

Garczarek, Ursula Maria (2002): Classification rules in standardized partition spaces. Dissertation, University of Dortmund. URL http://hdl.handle.net/2003/2789

See Also

calc.trans

Examples

```
library(MASS)
data(B3)
trans.matrix <- calc.trans(B3$PHASEN)

# Calculate posterior prob. for the classes via lda
prob.matrix <- predict(lda(PHASEN ~ ., data = B3))$post
errormatrix(B3$PHASEN, apply(prob.matrix, 1, which.max))
prior.prob <- hmm.sop("2", trans.matrix, prob.matrix)
errormatrix(B3$PHASEN, apply(prior.prob, 1, which.max))
```

kmodes *K-Modes Clustering*

Description

Perform k-modes clustering on categorical data.

Usage

```
kmodes(data, modes, iter.max = 10, weighted = FALSE, fast = TRUE)
```

Arguments

data	A matrix or data frame of categorical data. Objects have to be in rows, variables in columns.
modes	Either the number of modes or a set of initial (distinct) cluster modes. If a number, a random set of (distinct) rows in data is chosen as the initial modes.
iter.max	The maximum number of iterations allowed.
weighted	Whether usual simple-matching distance between objects is used, or a weighted version of this distance.
fast	Logical Whether a fast version of the algorithm should be applied.

Details

The k-modes algorithm (Huang, 1997) an extension of the k-means algorithm by MacQueen (1967).

The data given by data is clustered by the k-modes method (Huang, 1997) which aims to partition the objects into k groups such that the distance from objects to the assigned cluster modes is minimized.

By default simple-matching distance is used to determine the dissimilarity of two objects. It is computed by counting the number of mismatches in all variables. Alternative this distance is weighted by the frequencies of the categories in data (see Huang, 1997, for details).

If an initial matrix of modes is supplied, it is possible that no object will be closest to one or more modes. In this case less cluster than supplied modes will be returned and a warning is given.

If called using fast = TRUE the reassignment of the data to clusters is done for the entire data set before recomputation of the modes is done. For computational reasons this option should be chosen unless moderate data sizes.

For clustering mixed type data it is referred to kproto.

Value

An object of class "kmodes" which is a list with components:

cluster	A vector of integers indicating the cluster to which each object is allocated.
size	The number of objects in each cluster.

modes	A matrix of cluster modes.
withindiff	The within-cluster simple-matching distance for each cluster.
iterations	The number of iterations the algorithm has run.
weighted	Whether weighted distances were used or not.

Author(s)

Christian Neumann, <christian2.neumann@tu-dortmund.de>, Gero Szepannek, <gero.szepannek@web.de>

References

Huang, Z. (1997) A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. in *KDD: Techniques and Applications* (H. Lu, H. Motoda and H. Luu, Eds.), pp. 21-34, World Scientific, Singapore.

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, eds L. M. Le Cam & J. Neyman, **1**, pp. 281-297. Berkeley, CA: University of California Press.

Examples

```
### a 5-dimensional toy-example:

## generate data set with two groups of data:
set.seed(1)
x <- rbind(matrix(rbinom(250, 2, 0.25), ncol = 5),
           matrix(rbinom(250, 2, 0.75), ncol = 5))
colnames(x) <- c("a", "b", "c", "d", "e")

## run algorithm on x:
(cl <- kmodes(x, 2))

## and visualize with some jitter:
plot(jitter(x), col = cl$cluster)
points(cl$modes, col = 1:5, pch = 8)
```

---

loclda *Localized Linear Discriminant Analysis (LocLDA)*

Description

A localized version of Linear Discriminant Analysis.

Usage

```
loclda(x, ...)

## S3 method for class 'formula'
loclda(formula, data, ..., subset, na.action)

## Default S3 method:
loclda(x, grouping, weight.func = function(x) 1/exp(x),
    k = nrow(x), weighted.apriori = TRUE, ...)

## S3 method for class 'data.frame'
 loclda(x, ...)

## S3 method for class 'matrix'
loclda(x, grouping, ..., subset, na.action)
```

Arguments

formula	Formula of the form 'groups ~ x1 + x2 + ...'.
data	Data frame from which variables specified in formula are to be taken.
x	Matrix or data frame containing the explanatory variables (required, if formula is not given).
grouping	(required if no formula principal argument is given.) A factor specifying the class for each observation.
weight.func	Function used to compute local weights. Must be finite over the interval [0,1]. See Details below.
k	Number of nearest neighbours used to construct localized classification rules. See Details below.
weighted.apriori	Logical: if TRUE, class prior probabilities are computed using local weights (see Details below). If FALSE, equal priors for all classes actually occurring in the train data are used.
subset	An index vector specifying the cases to be used in the training sample.
na.action	A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit which leads to rejection of cases with missing values on any required variable.
...	Further arguments to be passed to loclda.default.

Details

This is an approach to apply the concept of localization described by Tutz and Binder (2005) to Linear Discriminant Analysis. The function loclda generates an object of class loclda (see Value below). As localization makes it necessary to build an individual decision rule for each test observation, this rule construction has to be handled by predict.loclda. For convenience, the rule building procedure is still described here.

To classify a test observation x_s, only the k nearest neighbours of x_s within the train data are used. Each of these k train observations $x_i, i = 1, \ldots, k$, is assigned a weight w_i according to

$$w_i = K\left(\frac{||x_i - x_s||}{d_k}\right), i = 1, \ldots, k$$

where K is the weighting function given by `weight.func`, $||x_i - x_s||$ is the euclidian distance of x_i and x_s and d_k is the euclidian distance of x_s to its k-th nearest neighbour. With these weights for each class $A_g, g = 1, \ldots, G$, its weighted empirical mean $\hat{\mu}_g$ and weighted empirical covariance matrix are computed. The estimated pooled (weighted) covariance matrix $\hat{\Sigma}$ is then calculated from the individual weighted empirical class covariance matrices. If `weighted.apriori` is `TRUE` (the default), prior class probabilities are estimated according to:

$$prior_g := \frac{\sum_{i=1}^{k}(w_i \cdot I(x_i \in A_g))}{\sum_{i=1}^{k}(w_i)}$$

where I is the indicator function. If `FALSE`, equal priors for all classes are used. In analogy to Linear Discriminant Analysis, the decision rule for x_s is

$$\hat{A} := argmax_{g \in 1,\ldots,G}(posterior_g)$$

where

$$posterior_g := prior_g \cdot \exp\left((-\frac{1}{2})t(x_s - \hat{\mu}_g)\hat{\Sigma}^{-1}(x_s - \hat{\mu}_g)\right)$$

If $posterior_g < 10^{(-150)} \forall g \in \{1, \ldots, G\}$, $posterior_g$ is set to $\frac{1}{G}$ for all $g \in 1, \ldots, G$ and the test observation x_s is simply assigned to the class whose weighted mean has the lowest euclidian distance to x_s.

Value

A list of class `loclda` containing the following components:

`call`	The (matched) function call.
`learn`	Matrix containing the values of the explanatory variables for all train observations.
`grouping`	Factor specifying the class for each train observation.
`weight.func`	Value of the argument `weight.func`.
`k`	Value of the argument `k`.
`weighted.apriori`	Value of the argument `weighted.apriori`.

Author(s)

Marc Zentgraf (<marc-zentgraf@gmx.de>) and Karsten Luebke (<karsten.luebke@fom.de>)

References

Tutz, G. and Binder, H. (2005): Localized classification. *Statistics and Computing* 15, 155-166.

See Also

predict.loclda, lda

Examples

```
benchB3("lda")$l1co.error
benchB3("loclda")$l1co.error
```

---

locpvs *Pairwise variable selection for classification in local models*

Description

Performs pairwise variable selection on subclasses.

Usage

```
locpvs(x, subclasses, subclass.labels, prior=NULL, method="lda",
    vs.method = c("ks.test", "stepclass", "greedy.wilks"),
    niveau=0.05, fold=10, impr=0.1, direct="backward", out=FALSE, ...)
```

Arguments

x	matrix or data frame containing the explanatory variables. x must consist of numerical data only.
subclasses	vector indicating the subclasses (a factor)
subclass.labels	must be a matrix with 2 coloumns, where the first coloumn specifies the subclass and the second coloumn the according upper class
prior	prior probabilites for the classes. If not specified the prior probabilities will be set according to proportion in "subclasses". If specified the order of prior probabilities must be the same as in "subclasses".
method	character, name of classification function (e.g. "lda" (default)).
vs.method	character, name of variable selection method. Must be one of "ks.test" (default), "stepclass" or "greedy.wilks".
niveau	used niveau for "ks.test"
fold	parameter for cross-validation, if "stepclass" is chosen 'vs.method'
impr	least improvement of performance measure desired to include or exclude any variable (<=1), if "stepclass" is chosen 'vs.method'
direct	direction of variable selection, if "stepclass" is chosen 'vs.method'. Must be one if "forward", "backward" (default) or "both".
out	indicator (logical) for textoutput during computation (slows down computation!), if "stepclass" is chosen 'vs.method'
...	further parameters passed to classification function ('method') or variable selection method ('vs.method')

Details

A call on pvs is performed using "subclasses" as grouping variable. See pvs for further details.

Value

An object of class 'locpvs' containing the following components:

`pvs.result` the complete output of the call to pvs (see pvs for further details

`subclass.labels`
the subclass.labels as specified in function call

Author(s)

Gero Szepannek, <szepannek@statistik.tu-dortmund.de>, Christian Neumann

References

Szepannek, G. and Weihs, C. (2006) Local Modelling in Classification on Different Feature Subspaces. In *Advances in Data Mining.*, ed Perner, P., LNAI 4065, pp. 226-234. Springer, Heidelberg.

See Also

predict.locpvs for predicting 'locpvs' models and pvs

Examples

```
## this example might be a bit artificial, but it sufficiently shows how locpvs has to be used

## learn a locpvs-model on the Vehicle dataset

library("mlbench")
data("Vehicle")

subclass <- Vehicle$Class # use four car-types in dataset as subclasses
## aggregate "bus" and "van" to upper-class "big" and "saab" and "opel" to upper-class "small"
subclass_class <- matrix(c("bus","van","saab","opel","big","big","small","small"),ncol=2)

## learn now a locpvs-model for the subclasses:
model <- locpvs(Vehicle[,1:18], subclass, subclass_class)
model # short summary, showing the class-pairs of the submodels
# together with the selected variables and the relation of sub- to upperclasses

## predict:
pred <- predict(model, Vehicle[,1:18])

## now you can look at the predicted classes:
pred$class
## or at the posterior probabilities:
pred$posterior
## or at the posterior probabilities for the subclasses:
pred$subclass.posteriors
```

meclight.default *Minimal Error Classification*

Description

Computer intensive method for linear dimension reduction that minimizes the classification error directly.

Usage

```
meclight(x, ...)

## Default S3 method:
meclight(x, grouping, r = 1, fold = 10, ...)
## S3 method for class 'formula'
meclight(formula, data = NULL, ..., subset, na.action = na.fail)
## S3 method for class 'data.frame'
meclight(x, ...)
## S3 method for class 'matrix'
meclight(x, grouping, ..., subset, na.action = na.fail)
```

Arguments

`x`	(required if no formula is given as the principal argument.) A matrix or data frame containing the explanatory variables.
`grouping`	(required if no formula principal argument is given.) A factor specifying the class for each observation.
`r`	Dimension of projected subspace.
`fold`	Number of Bootstrap samples.
`formula`	A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
`data`	Data frame from which variables specified in formula are preferentially to be taken.
`subset`	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
`na.action`	A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is `na.omit`, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)
`...`	Further arguments passed to `lda`.

Details

Computer intensive method for linear dimension reduction that minimizes the classification error in the projected subspace directly. Classification is done by `lda`. In contrast to the reference function minimization is done by Nelder-Mead in `optim`.

Value

method.model	An object of class 'lda'.
Proj.matrix	Projection matrix.
B.error	Estimated bootstrap error rate.
B.impro	Improvement in lda error rate.

Author(s)

Maria Eveslage, Karsten Luebke, <karsten.luebke@fom.de>

References

Roehl, M.C., Weihs, C., and Theis, W. (2002): Direct Minimization in Multivariate Classification. *Computational Statistics*, 17, 29-46.

See Also

predict.meclight

Examples

```
data(iris)
meclight.obj <- meclight(Species ~ ., data = iris)
meclight.obj
```

NaiveBayes *Naive Bayes Classifier*

Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Usage

```
## S3 method for class 'formula'
NaiveBayes(formula, data, ..., subset, na.action = na.pass)
## Default S3 method:
NaiveBayes(x, grouping, prior, usekernel = FALSE, fL = 0, ...)
```

Arguments

x	a numeric matrix, or a data frame of categorical and/or numeric variables.
grouping	class vector (a factor).
formula	a formula of the form class ~ x1 + x2 + ... Interactions are not allowed.
data	a data frame of predictors (categorical and/or numeric).
prior	the prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
usekernel	if TRUE a kernel density estimate (density) is used for density estimation. If FALSE a normal density is estimated.
fL	Factor for Laplace correction, default factor is 0, i.e. no correction.
...	arguments passed to density.
subset	for data given in a data frame, an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
na.action	a function to specify the action to be taken if NAs are found. The default action is not to count them for the computation of the probability factors. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)

Details

This implementation of Naive Bayes as well as this help is based on the code by David Meyer in the package e1071 but extended for kernel estimated densities and user specified prior probabilities. The standard naive Bayes classifier (at least this implementation) assumes independence of the predictor variables.

Value

An object of class "NaiveBayes" including components:

apriori	Class distribution for the dependent variable.
tables	A list of tables, one for each predictor variable. For each categorical variable a table giving, for each attribute level, the conditional probabilities given the target class. For each numeric variable, a table giving, for each target class, mean and standard deviation of the (sub-)variable or a object of class density.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

predict.NaiveBayes,plot.NaiveBayes,naiveBayes,qda

Examples

```
data(iris)
m <- NaiveBayes(Species ~ ., data = iris)
```

nm *Nearest Mean Classification*

Description

Function for nearest mean classification.

Usage

```
nm(x, ...)

## Default S3 method:
nm(x, grouping, gamma = 0, ...)
## S3 method for class 'data.frame'
nm(x, ...)
## S3 method for class 'matrix'
nm(x, grouping, ..., subset, na.action = na.fail)
## S3 method for class 'formula'
nm(formula, data = NULL, ..., subset, na.action = na.fail)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given)
grouping	factor specifying the class for each observation (required, if formula is not given)
formula	formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators
data	Data frame from which variables specified in formula are preferentially to be taken
gamma	gamma parameter for rbf weight of the distance to mean. If gamma=0 the posterior is 1 for the nearest class (mean) and 0 else.
subset	An index vector specifying the cases to be used in the training sample. (Note: If given, this argument must be named!)
na.action	specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (Note: If given, this argument must be named.)
...	further arguments passed to the underlying sknn function

Details

nm is calling sknn with the class means as observations. If gamma>0 a gaussian like density is used to weight the distance to the class means weight=exp(-gamma*distance). This is similar to an rbf kernel. If the distances are large it may be useful to scale the data first.

Value

A list containing the function call and the class means (learn)).

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

sknn, rda, knn

Examples

```
data(B3)
x <- nm(PHASEN ~ ., data = B3)
x$learn
x <- nm(PHASEN ~ ., data = B3, gamma = 0.1)
predict(x)$post
```

---

partimat *Plotting the 2-d partitions of classification methods*

Description

Provides a multiple figure array which shows the classification of observations based on classification methods (e.g. lda, qda) for every combination of two variables. Moreover, the classification borders are displayed and the apparent error rates are given in each title.

Usage

```
partimat(x,...)

## Default S3 method:
partimat(x, grouping, method = "lda", prec = 100,
    nplots.vert, nplots.hor, main = "Partition Plot", name, mar,
    plot.matrix = FALSE, plot.control = list(), ...)
## S3 method for class 'data.frame'
partimat(x, ...)
## S3 method for class 'matrix'
partimat(x, grouping, ..., subset, na.action = na.fail)
## S3 method for class 'formula'
partimat(formula, data = NULL, ..., subset, na.action = na.fail)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given).
grouping	factor specifying the class for each observation (required, if formula is not given).
formula	formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
method	the method the classification is based on, currently supported are: lda, qda, rpart, naiveBayes, rda, sknn and svmlight
.	
prec	precision used to draw the classification borders (the higher the more precise; default: 100).
data	Data frame from which variables specified in formula are preferentially to be taken.
nplots.vert	number of rows in the multiple figure array
nplots.hor	number of columns in the multiple figure array
subset	index vector specifying the cases to be used in the training sample. (Note: If given, this argument must be named.)
na.action	specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (Note: If given, this argument must be named.)
main	title
name	Variable names to be printed at the axis / into the diagonal.
mar	numerical vector of the form c(bottom, left, top, right) which gives the lines of margin to be specified on the four sides of the plot. Defaults are rep(0, 4) if plot.matrix = TRUE, c(5, 4, 2, 1) + 0.1 otherwise.
plot.matrix	logical; if TRUE, like a scatterplot matrix; if FALSE (default) uses less space and arranges the plots "optimal" (using a fuzzy algorithm) in an array by plotting each pair of variables once.
plot.control	A list containing further arguments passed to the underlying plot functions (and to drawparti).
...	Further arguments passed to the classification method (through drawparti).

Note

Warnings such as 'parameter "xyz" couldn't be set in high-level plot function' are expected, if making use of ....

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>, Uwe Ligges, Irina Czogiel

See Also

for much more fine tuning see drawparti

Examples

```
library(MASS)
data(iris)
partimat(Species ~ ., data = iris, method = "lda")
## Not run:
partimat(Species ~ ., data = iris, method = "lda",
    plot.matrix = TRUE, imageplot = FALSE) # takes some time ...

## End(Not run)
```

---

plineplot *Plotting marginal posterior class probabilities*

Description

For a given variable the posteriori probabilities of the classes given by a classification method are plotted. The variable need not be used for the actual classifcation.

Usage

```
plineplot(formula, data, method, x, col.wrong = "red",
          ylim = c(0, 1), loo = FALSE, mfrow, ...)
```

Arguments

formula	formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
data	Data frame from which variables specified in formula are preferentially to be taken.
method	character, name of classification function (e.g. "lda").
x	variable that should be plotted. See examples.
col.wrong	color to use for missclassified objects.
ylim	ylim for the plot.
loo	logical, whether leave-one-out estimate is used for prediction
mfrow	number of rows and columns in the graphics device, see par. If missing, number of rows equals number of classes, and 1 column.
...	further arguments passed to the underlying classification method or plot functions.

Value

The actual error rate.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

partimat

Examples

```
library(MASS)

# The name of the variable can be used for x
data(B3)
plineplot(PHASEN ~ ., data = B3, method = "lda",
    x = "EWAJW", xlab = "EWAJW")

# The plotted variable need not be in the data
data(iris)
iris2 <- iris[ , c(1,3,5)]
plineplot(Species ~ ., data = iris2, method = "lda",
    x = iris[ , 4], xlab = "Petal.Width")
```

---

plot.NaiveBayes *Naive Bayes Plot*

Description

Visualizes the marginal probabilities of predictor variables given the class.

Usage

```
## S3 method for class 'NaiveBayes'
plot(x, vars, n = 1000, legendplot = TRUE, lty, col,
    ylab = "Density", main = "Naive Bayes Plot", ...)
```

Arguments

x	an object of class NaiveBayes
vars	variables to be plotted. If missing, all predictor variables are plotted.
n	number of points used to plot the density line.
legendplot	logical, whether to print a legend
lty	line type for different classes, defaults to the first length(x$apriori) colors of the current palette in use.
col	color for different classes, defaults to rainbow(length(x$apriori)).
ylab	label for y-axis.
main	title of the plots.
...	furhter arguments passed to the underlying plot functions.

Details

For metric variables the estimated density is plotted. For categorial variables mosaicplot is called.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

NaiveBayes

Examples

```
data(iris)
mN <- NaiveBayes(Species ~ ., data = iris)
plot(mN)

mK <- NaiveBayes(Species ~ ., data = iris, usekernel = TRUE)
plot(mK)
```

---

plot.woe *Plot information values*

Description

Barplot of information values to compare dicriminator of the transformed variables.

Usage

```
## S3 method for class 'woe'
plot(x, type = c("IV", "woes"), ...)
```

Arguments

x	An object of class woe.
type	Character to specify the plot type, see below. Either "IV" (default) or "woes".
...	Further arguments to be passed to the barplot function.

Details

For type=="IV" a barplot of information values for all transformed variables. A thumb rule of interpretation is that Values above 0.3 are considered as strongly discrimative where values below 0.02 are considered to characterize unpredictive variables. For type=="woes" for each variable the relative frequencies of all transformed levels are plotted.

Value

No value is returned.

Author(s)

Gero Szepannek

References

Good, I. (1950): *Probability and the Weighting of Evidences.* Charles Griffin, London.

Kullback, S. (1959): *Information Theory and Statistics.* Wiley, New York.

See Also

woe, predict.woe

Examples

```
# see examples in ?woe
```

`predict.loclda` *Localized Linear Discriminant Analysis (LocLDA)*

Description

Classifies new observations using parameters determined by the `loclda`-function.

Usage

```
## S3 method for class 'loclda'
predict(object, newdata, ...)
```

Arguments

`object`	Object of class `loclda`.
`newdata`	Data frame of cases to be classified.
`...`	Further arguments are ignored.

Value

A list with components:

`class`	Vector (of class `factor`) of classifications.
`posterior`	Posterior probabilities for the classes. For details of computation see `loclda` (+ normalization so posterior-values add up to 1 for each observation).
`all.zero`	Vector (of class `integer`) indicating for which rows of `newdata` all corresponding posterior-values are $< 10^{-150}$ before normalization. Those observations are assigned to the class to whose (locally weighted) centroid they have the lowest euclidian distance.

Author(s)

Marc Zentgraf (<marc-zentgraf@gmx.de>) and Karsten Luebke (<karsten.luebke@fom.de>)

See Also

loclda

Examples

```
data(B3)
x <- loclda(PHASEN ~ ., data = B3, subset = 1:80)
predict(x, B3[-(1:80),])
```

---

predict.locpvs *predict method for locpvs objects*

Description

Prediction of class membership and posterior probabilities in local models using pairwise variable selection.

Usage

```
## S3 method for class 'locpvs'
predict(object,newdata, quick = FALSE, return.subclass.prediction = TRUE, ...)
```

Arguments

object	an object of class 'locpvs', as that created by the function "locpvs"
newdata	a data frame or matrix containing new data. If not given the same datas as used for training the 'pvs'-model are used.
quick	indicator (logical), whether a quick, but less accurate computation of posterior probabalities should be used or not.
return.subclass.prediction	indicator (logical), whether the returned object includes posterior probabilities for each date in each subclass
...	Further arguments are passed to underlying predict calls.

Details

Posterior probabilities are predicted as if object is a standard 'pvs'-model with the subclasses as classes. Then the posterior probabalities are summed over all subclasses for each class. The class with the highest value becomes the prediction.

If "quick=FALSE" the posterior probabilites for each case are computed using the pairwise coupling algorithm presented by Hastie, Tibshirani (1998). If "quick=FALSE" a much quicker solution is used, which leads to less accurate posterior probabalities. In almost all cases it doesn't has a negative effect on the classification result.

Value

a list with components:

class	the predicted (upper) classes
posterior	posterior probabilities for the (upper) classes
subclass.posteriors	(only if "return.subclass.prediction=TRUE". A matrix containing posterior probabalities for the subclasses.

Author(s)

Gero Szepannek, <szepannek@statistik.tu-dortmund.de>, Christian Neumann

References

Szepannek, G. and Weihs, C. (2006) Local Modelling in Classification on Different Feature Subspaces. In *Advances in Data Mining.*, ed Perner, P., LNAI 4065, pp. 226-234. Springer, Heidelberg.

See Also

locpvs for learning 'locpvs'-models and examples for applying this predict method, pvs for pairwise variable selection without modeling subclasses, predict.pvs for predicting 'pvs'-models

---

predict.meclight *Prediction of Minimal Error Classification*

Description

Classify multivariate observations in conjunction with meclight and lda.

Usage

```
## S3 method for class 'meclight'
predict(object, newdata,...)
```

Arguments

object	Object of class meclight.
newdata	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.
...	currently ignored

Details

Classify multivariate observations in conjunction with meclight and lda.

Value

class	The estimated class (factor).
posterior	Posterior probabilities for the classes.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

References

Roehl, M.C., Weihs, C., and Theis, W. (2002): Direct Minimization in Multivariate Classification. *Computational Statistics*, 17, 29-46.

See Also

meclight

Examples

```
data(iris)
meclight.obj <- meclight(Species ~ ., data = iris)
predict(meclight.obj, iris)
```

---

predict.NaiveBayes *Naive Bayes Classifier*

Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Usage

```
## S3 method for class 'NaiveBayes'
predict(object, newdata, threshold = 0.001, ...)
```

Arguments

object	An object of class "naiveBayes".
newdata	A dataframe with new predictors.
threshold	Value replacing cells with 0 probabilities.
...	passed to dkernel function if neccessary.

Details

This implementation of Naive Bayes as well as this help is based on the code by David Meyer in the package e1071 but extended for kernel estimated densities. The standard naive Bayes classifier (at least this implementation) assumes independence of the predictor variables. For attributes with missing values, the corresponding table entries are omitted for prediction.

Value

A list with the conditional a-posterior probabilities for each class and the estimated class are returned.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

NaiveBayes,dkernelnaiveBayes,qda

Examples

```
data(iris)
m <- NaiveBayes(Species ~ ., data = iris)
predict(m)
```

---

predict.pvs *predict method for pvs objects*

Description

Prediction of class membership and posterior probabilities using pairwise variable selection.

Usage

```
## S3 method for class 'pvs'
predict(object, newdata, quick = FALSE, detail = FALSE, ...)
```

Arguments

object	an object of class 'pvs', as that created by the function "pvs"
newdata	a data frame or matrix containing new data. If not given the same datas as used for training the 'pvs'-model are used.
quick	indicator (logical), whether a quick, but less accurate computation of posterior probabalities should be used or not.
detail	indicator (logical), whether the returned object includes additional information about the posterior probabilities for each date in each submodel.
...	Further arguments are passed to underlying predict calls.

Details

If "quick=FALSE" the posterior probabilites for each case are computed using the pairwise coupling algorithm presented by Hastie, Tibshirani (1998). If "quick=FALSE" a much quicker solution is used, which leads to less accurate posterior probabalities. In almost all cases it doesn't has a negative effect on the classification result.

Value

a list with components:

class	the predicted classes
posterior	posterior probabilities for the classes
details	(only if "details=TRUE". A list containing matrices of posterior probabalities computated by the classification method for each case and classpair.

Author(s)

Gero Szepannek, <szepannek@statistik.tu-dortmund.de>, Christian Neumann

References

Szepannek, G. and Weihs, C. (2006) Variable Selection for Classification of More than Two Classes Where the Data are Sparse. In *From Data and Information Analysis to Kwnowledge Engineering.*, eds Spiliopolou, M., Kruse, R., Borgelt, C., Nuernberger, A. and Gaul, W. pp. 700-708. Springer, Heidelberg.

See Also

For more details and examples how to use this predict method, see pvs.

---

predict.rda *Regularized Discriminant Analysis (RDA)*

Description

Classifies new observations using parameters determined by the rda-function.

Usage

```
## S3 method for class 'rda'
predict(object, newdata, posterior = TRUE,
    aslist = TRUE, ...)
```

Arguments

object	Object of class rda.
newdata	Data frame (or matrix) of cases to be classified.
posterior	Logical; indicates whether a matrix of posterior probabilites over all classes for each observation shall be returned in addition to classifications.
aslist	Logical; if TRUE, a list containing classifications and posterior probabilities is returned, otherwise a vector with an attribute 'posterior'.
...	currently unused

Value

Depends on the value of argument 'aslist':

Either a vector (of class factor) of classifications that (optionally) has an attribute 'posterior' containing the posterior probability matrix, or

A list with elements 'class' and 'posterior'.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

rda

Examples

```
data(iris)
x <- rda(Species ~ ., data = iris, gamma = 0.05, lambda = 0.2)
predict(x, iris[, 1:4])
```

---

predict.sknn *Simple k Nearest Neighbours Classification*

Description

Classifies new observations using the sknn learned by the sknn-function.

Usage

```
## S3 method for class 'sknn'
predict(object, newdata,...)
```

Arguments

object	Object of class sknn.
newdata	Data frame (or matrix) of cases to be classified.
...	...

Value

A list with elements 'class' and 'posterior'.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

sknn, knn

Examples

```
data(iris)
x <- sknn(Species ~ ., data = iris)
predict(x, iris)
x <- sknn(Species ~ ., gamma = 10, kn = 10, data = iris)
predict(x, iris)
```

---

predict.svmlight *Interface to SVMlight*

Description

Predicts new observations using the SVM learned by the svmlight-function.

Usage

```
## S3 method for class 'svmlight'
predict(object, newdata, scal = TRUE, ...)
```

Arguments

object	Object of class svmlight.
newdata	Data frame (or matrix) of cases to be predicted.
scal	Logical, whether to scale membership values via e.scal.
...	...

Value

If a classification is learned (type="C") in svmlight a list with elements 'class' and 'posterior' (scaled, if scal = TRUE).

If a Regression is learned (type="R") in svmlight the predicted values.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

svmlight, svm

Examples

```
## Not run:
data(iris)
x <- svmlight(Species ~ ., data = iris)
predict(x, iris)

## End(Not run)
```

---

predict.woe *Weights of evidence*

Description

Applies weight of evidence transform of factor variables for binary classification based on a model of class woe.

Usage

```
## S3 method for class 'woe'
predict(object, newdata, replace = TRUE, ...)
```

Arguments

object	Object resulting from a call of woe.
newdata	A matrix or data frame where WOE transform should be applied of the same dimension as the data used for training the woe object.
replace	Logical flag specifying whether the original factor variables should be kept in the output.
...	Currently not used.

Value

Data frame including the transformed numeric woe variables.

Author(s)

Gero Szepannek

References

Good, I. (1950): *Probability and the Weighting of Evidences.* Charles Griffin, London.

See Also

woe, plot.woe

Examples

```
# see examples in ?woe
```

---

pvs *Pairwise variable selection for classification*

Description

Pairwise variable selection for numerical data, allowing the use of different classifiers and different variable selection methods.

Usage

```
pvs(x, ...)

## Default S3 method:
pvs(x, grouping, prior=NULL, method="lda",
    vs.method=c("ks.test","stepclass","greedy.wilks"), niveau=0.05,
    fold=10, impr=0.1, direct="backward", out=FALSE, ...)

## S3 method for class 'formula'
pvs(formula, data = NULL, ...)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given). x must consist of numerical data only.
formula	A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor (the classes) and the right hand side specifies the (numerical) discriminators. Interaction terms are not supported.
data	data matrix (rows=cases, columns=variables)
grouping	class indicator vector (a factor)
prior	prior probabilites for the classes. If not specified the prior probabilities will be set according to proportion in "grouping". If specified the order of prior probabilities must be the same as in "grouping".
method	character, name of classification function (e.g. "lda" (default)).
vs.method	character, name of variable selection method. Must be one of "ks.test" (default), "stepclass" or "greedy.wilks".
niveau	used niveau for "ks.test"
fold	parameter for cross-validation, if "stepclass" is chosen 'vs.method'

impr	least improvement of performance measure desired to include or exclude any variable (<=1), if "stepclass" is chosen 'vs.method'
direct	direction of variable selection, if "stepclass" is chosen 'vs.method'. Must be one if "forward", "backward" (default) or "both".
out	indicator (logical) for textoutput during computation (slows down computation!), if "stepclass" is chosen 'vs.method'
...	further parameters passed to classification function ('method') or variable selection method ('vs.method')

Details

The classification "method" (e.g. 'lda') must have its own 'predict' method (like 'predict.lda' for 'lda') returns a list with an element 'posterior' containing the posterior probabilties. It must be able to deal with matrices as in method(x, grouping, ...). Examples of such classification methods are 'lda', 'qda', 'rda', 'NaiveBayes' or 'sknn'.\ For the classification methods "svm" and "randomForest" there are special routines implemented, to make them work with 'pvs' method even though their 'predict' methods don't provide the demanded posteriors. However those two classfiers can not be used together with variable selection method "stepclass".

'pvs' performs a variable selection using the selection method chosen in 'vs.method' for each pair of classes in 'x'. Then for each pair of classes a submodel using 'method' is trained (using only the earlier selected variables for this class-pair).

If 'method' is "ks.test", then for each variable the empirical distribution functions of the cases of both classes are compared via "ks.test". Only variables with a p-values below 'niveau' are used for training the submodel for this pair of classes.

If 'method' is "stepclass" the variable selection is performed using the "stepclass" method.

If 'method' is "greedy.wilks" the variable selection is performed using Wilk's lambda criterion.

Value

An object of class 'pvs' containing the following components:

classes	the classes in grouping
prior	used prior probabilities
method	name of used classification function
vs.method	name of used function for variable selection
submodels	containing a list of submodels. For each pair of classes there is a list element being another list of 3 containing the class-pair of this submodel, the selected variables for the subspace of classes and the result of the trained classification function.
call	the (matched) function call

Author(s)

Gero Szepannek, <szepannek@statistik.tu-dortmund.de>, Christian Neumann

References

- Szepannek, G. and Weihs, C. (2006) Variable Selection for Classification of More than Two Classes Where the Data are Sparse. In *From Data and Information Analysis to Kwnowledge Engineering.*, eds Spiliopolou, M., Kruse, R., Borgelt, C., Nuernberger, A. and Gaul, W. pp. 700-708. Springer, Heidelberg.
- Szepannek, G. (2008): Different Subspace Classification - Datenanalyse, -interpretation, -visualisierung und Vorhersage in hochdimensionalen Raeumen, ISBN 978-3-8364-6302-7, vdm, Saarbruecken.

See Also

`predict.pvs` for predicting 'pvs' models and `locpvs` for pairwisevariable selection in local models of several subclasses

Examples

```
## Example 1: learn an "lda" model on the waveform data using pairwise variable
## selection (pvs) using "ks.test" and compare it to using lda without pvs

library("mlbench")
trainset <- mlbench.waveform(300)
pvsmodel <- pvs(trainset$x, trainset$classes, niveau=0.05) # default: using method="lda"
## short summary, showing the class-pairs of the submodels and the selected variables
pvsmodel

testset <-  mlbench.waveform(500)
## prediction of the test data set:
prediction <- predict(pvsmodel, testset$x)

## calculating the test error rate
1-sum(testset$classes==prediction$class)/length(testset$classes)
## Bayes error is 0.149

## comparison to performance of simple lda
ldamodel <- lda(trainset$x, trainset$classes)
LDAprediction <- predict(ldamodel, testset$x)

## test error rate
1-sum(testset$classes==LDAprediction$class)/length(testset$classes)


## Example 2: learn a "qda" model with pvs on half of the Satellite dataset,
## using "ks.test"


library("mlbench")
data("Satellite")

## takes few seconds as exact KS tests are calculated here:
model <- pvs(classes ~ ., Satellite[1:3218,], method="qda", vs.method="ks.test")
## short summary, showing the class-pairs of the submodels and the selected variables
```

```
model

## now predict on the rest of the data set:
## pred <- predict(model,Satellite[3219:6435,]) # takes some time
pred <- predict(model,Satellite[3219:6435,], quick=TRUE) # that's much quicker

## now you can look at the predicted classes:
pred$class
## or the posterior probabilities:
pred$posterior
```

---

quadplot *Plotting of 4 dimensional membership representation simplex*

Description

For a 4 class discrimination problem the membership values of each class are visualized in a 3 dimensional barycentric coordinate system.

Usage

```
quadplot(e = NULL, f = NULL, g = NULL, h = NULL, angle = 75,
    scale.y = 0.6, label = 1:4, labelcol = rainbow(4),
    labelpch = 19, labelcex = 1.5, main = "", s3d.control = list(),
    simplex.control = list(), legend.control = list(), ...)
```

Arguments

e	either a matrix with 4 columns represanting the membership values or a vector with the membership values of the first class
f	vector with the membership values of the second class
g	vector with the membership values of the third class
h	vector with the membership values of the forth class
angle	angle between x and y axis
scale.y	scale of y axis related to x- and z axis
label	label for the classes
labelcol	colors to use for the labels
labelpch	pch for the labels
labelcex	cex for the labels
main	main title of the plot
s3d.control	a *list* with further arguments passed to the underlying scatterplot3d function call that sets up the plot

`simplex.control` a *list* with further arguments passed to the underlying function call that draws the barycentric coordinate system

`legend.control` a *list* with further arguments passed to the underlying function call that adds the `legend`

`...` further arguments passed to the underlying `plot` function that draws the data points

Details

The membership values are calculated with `quadtrafo` and plotted with `scatterplot3d`.

Value

A `scatterplot3d` object.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>, and Uwe Ligges

References

Garczarek, Ursula Maria (2002): Classification rules in standardized partition spaces. Dissertation, University of Dortmund. URL http://hdl.handle.net/2003/2789

See Also

`triplot`, `scatterplot3d`

Examples

```
library("MASS")
data(B3)
opar <- par(mfrow = c(1, 2), pty = "s")
posterior <- predict(lda(PHASEN ~ ., data = B3))$post
s3d <- quadplot(posterior, col = rainbow(4)[B3$PHASEN],
        labelpch = 22:25, labelcex = 0.8,
        pch = (22:25)[apply(posterior, 1, which.max)],
        main = "LDA posterior assignments")
quadlines(centerlines(4), sp = s3d, lty = "dashed")

posterior <- predict(qda(PHASEN ~ ., data = B3))$post
s3d <- quadplot(posterior, col = rainbow(4)[B3$PHASEN],
        labelpch = 22:25, labelcex = 0.8,
        pch = (22:25)[apply(posterior, 1, which.max)],
        main = "QDA posterior assignments")
quadlines(centerlines(4), sp = s3d, lty = "dashed")
par(opar)
```

rda *Regularized Discriminant Analysis (RDA)*

Description

Builds a classification rule using regularized group covariance matrices that are supposed to be more robust against multicollinearity in the data.

Usage

```
rda(x, ...)

## Default S3 method:
rda(x, grouping = NULL, prior = NULL, gamma = NA,
    lambda = NA, regularization = c(gamma = gamma, lambda = lambda),
    crossval = TRUE, fold = 10, train.fraction = 0.5,
    estimate.error = TRUE, output = FALSE, startsimplex = NULL,
    max.iter = 100, trafo = TRUE, simAnn = FALSE, schedule = 2,
    T.start = 0.1, halflife = 50, zero.temp = 0.01, alpha = 2,
    K = 100, ...)
## S3 method for class 'formula'
rda(formula, data, ...)
```

Arguments

x	Matrix or data frame containing the explanatory variables (required, if formula is not given).
formula	Formula of the form 'groups ~ x1 + x2 + ...'.
data	A data frame (or matrix) containing the explanatory variables.
grouping	(Optional) a vector specifying the class for each observation; if not specified, the first column of 'data' is taken.
prior	(Optional) prior probabilities for the classes. Default: proportional to training sample sizes. "prior=1" indicates equally likely classes.
gamma, lambda, regularization	One or both of the rda-parameters may be fixed manually. Unspecified parameters are determined by minimizing the estimated error rate (see below).
crossval	Logical. If TRUE, in the optimization step the error rate is estimated by Cross-Validation, otherwise by drawing several training- and test-samples.
fold	The number of Cross-Validation- or Bootstrap-samples to be drawn.
train.fraction	In case of Bootstrapping: the fraction of the data to be used for training in each Bootstrap-sample; the remainder is used to estimate the misclassification rate.
estimate.error	Logical. If TRUE, the apparent error rate for the final parameter set is estimated.
output	Logical flag to indicate whether text output during computation is desired.
startsimplex	(Optional) a starting simplex for the Nelder-Mead-minimization.

max.iter	Maximum number of iterations for Nelder-Mead.
trafo	Logical; indicates whether minimization is carrried out using transformed parameters.
simAnn	Logical; indicates whether Simulated Annealing shall be used.
schedule	Annealing schedule 1 or 2 (exponential or polynomial).
T.start	Starting temperature for Simulated Annealing.
halflife	Number of iterations until temperature is reduced to a half (schedule 1).
zero.temp	Temperature at which it is set to zero (schedule 1).
alpha	Power of temperature reduction (linear, quadratic, cubic,...) (schedule 2).
K	Number of iterations until temperature = 0 (schedule 2).
...	currently unused

Details

J.H. Friedman (see references below) suggested a method to fix almost singular covariance matrices in discriminant analysis. Basically, individual covariances as in QDA are used, but depending on two parameters (γ and λ), these can be shifted towards a diagonal matrix and/or the pooled covariance matrix. For ($\gamma = 0$, $\lambda = 0$) it equals QDA, for ($\gamma = 0$, $\lambda = 1$) it equals LDA.

You may fix these parameters at certain values or leave it to the function to try to find "optimal" values. If one parameter is given, the other one is determined using the R-function '`optimize`'. If no parameter is given, both are determined numerically by a Nelder-Mead-(Simplex-)algorithm with the option of using Simulated Annealing. The goal function to be minimized is the (estimated) misclassification rate; the misclassification rate is estimated either by Cross-Validation or by repeatedly dividing the data into training- and test-sets (Boostrapping).

Warning: If these sets are small, optimization is expected to produce almost random results. We recommend to adjust the parameters manually in such a case. In all other cases it is recommended to run the optimization several times in order to see whether stable results are gained.

Since the Nelder-Mead-algorithm is actually intended for *continuous* functions while the observed error rate by its nature is *discrete*, a greater number of Boostrap-samples might improve the optimization by increasing the smoothness of the response surface (and, of course, by reducing variance and bias). If a set of parameters leads to singular covariance matrices, a penalty term is added to the misclassification rate which will hopefully help to maneuver back out of singularity (so do not worry about error rates greater than one during optimization).

Value

A list of class `rda` containing the following components:

call	The (matched) function call.
regularization	vector containing the two regularization parameters (gamma, lambda)
classes	the names of the classes
prior	the prior probabilities for the classes
error.rate	apparent error rate (if computation was not suppressed), and, if any optimization took place, the final (cross-validated or bootstrapped) error rate estimate as well.

means	Group means.
covariances	Array of group covariances.
covpooled	Pooled covariance.
converged	(Logical) indicator of convergence (only for Nelder-Mead).
iter	Number of iterations actually performed (only for Nelder-Mead).

More details

The explicit defintion of γ, λ and the resulting covariance estimates is as follows:

The pooled covariance estimate $\hat{\Sigma}$ is given as well as the individual covariance estimates $\hat{\Sigma}_k$ for each group.

First, using λ, a convex combination of these two is computed:

$$\hat{\Sigma}_k(\lambda) := (1-\lambda)\hat{\Sigma}_k + \lambda\hat{\Sigma}.$$

Then, another convex combination is constructed using the above estimate and a (scaled) identity matrix:

$$\hat{\Sigma}_k(\lambda,\gamma) = (1-\gamma)\hat{\Sigma}_k(\lambda) + \gamma\frac{1}{d}\mathrm{tr}[\hat{\Sigma}_k(\lambda)]\mathrm{I}.$$

The factor $\frac{1}{d}\mathrm{tr}[\hat{\Sigma}_k(\lambda)]$ in front of the identity matrix I is the mean of the diagonal elements of $\hat{\Sigma}_k(\lambda)$, so it is the mean variance of all d variables assuming the group covariance $\hat{\Sigma}_k(\lambda)$.

For the four extremes of (γ,λ) the covariance structure reduces to special cases:

- ($\gamma = 0$, $\lambda = 0$): QDA - individual covariance for each group.
- ($\gamma = 0$, $\lambda = 1$): LDA - a common covariance matrix.
- ($\gamma = 1$, $\lambda = 0$): Conditional independent variables - similar to Naive Bayes, but variable variances within group (main diagonal elements) are equal.
- ($\gamma = 1$, $\lambda = 1$): Classification using euclidean distance - as in previous case, but variances are the same for all groups. Objects are assigned to group with nearest mean.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

References

Friedman, J.H. (1989): Regularized Discriminant Analysis. In: *Journal of the American Statistical Association* 84, 165-175.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. (1992): *Numerical Recipes in C*. Cambridge: Cambridge University Press.

See Also

predict.rda, lda, qda

Examples

```
data(iris)
x <- rda(Species ~ ., data = iris, gamma = 0.05, lambda = 0.2)
predict(x, iris)
```

---

shardsplot *Plotting Eight Direction Arranged Maps or Self-Organizing Maps*

Description

Plotting method for objects of class EDAM or som.

Usage

```
shardsplot(object, plot.type = c("eight", "four", "points", "n"),
    expand = 1, stck = TRUE, grd = FALSE, standardize = FALSE,
    data.or = NA, label = FALSE, plot = TRUE, classes = 0,
    vertices = TRUE, classcolors = "rainbow", wghts = 0,
    xlab = "Dimension 1", ylab = "Dimension 2", xaxs = "i",
    yaxs = "i", plot.data.column = NA,
    log.classes = FALSE, revert.colors = FALSE, ...)

level_shardsplot(object, par.names, rows = 1:NCOL(object$data),
    centers = rep(NA, length(par.names)), class.labels = NA,
    revert.colors = rep(FALSE, length(par.names)),
    log.classes = rep(FALSE, length(par.names)),
    centeredcolors = colorRamp(c("red", "white", "blue")),
    mfrow = c(2, 2), plot.type = c("eight", "four", "points", "n"),
    expand = 1, stck = TRUE, grd = FALSE, standardize = FALSE,
    label = FALSE, plot = TRUE, vertices = TRUE, classcolors = "topo",
    wghts = 0, xlab = "Dimension 1", ylab = "Dimension 2",
    xaxs = "i", yaxs = "i", ...)

## S3 method for class 'EDAM'
plot(...)
```

Arguments

object	an object of class EDAM or som.
par.names	names used to lable the data columns
rows	vector with indices of colomns to be plotted
centers	vector of type numeric defining the class centers for the data. NA if data does not have a center.
class.labels	matrix of type text and dimension(3, NROW(object$data)) defining the lables to be used for maximum, minimum and central value.

centeredcolors	colors to represent the classes with a central value
mfrow	parameter defining number of plots on a page. see par
plot.type	a character giving the shape of the shards. Available are "eight" and "four" for octagons resp. rectangles, and "points" for points. If plot.type is "n", no shards are plotted at all.
expand	a numeric giving the relative expansion of the axes. A value greater than one implies smaller shards. Varying expand can be sensible for visual reasons.
stck	logical. If TRUE the cells are varied continously corresponding to the differences of direct neighbors in the origin space. Within this variation the relative order of the cells is always preserved.
grd	logical. If TRUE (which automatically sets stck to TRUE), the variation of cells is restricted to their original discrete values.
standardize	logical. If TRUE, then the measurements in object$preimages are standardized before calculating Euclidean distances. Measurements are standardized for each variable by dividing by the variable's standard deviation. Meaningless if object$preimages is a dissimilarity matrix.
data.or	original data and classes where the first k columns are variables and the (k+1)-th column are the classes. If defined and class of object is som, data.or is used to assign a class to each codebook. There a codebook receives the class, from which the majority of its assigned objects origins.
label	logical. If TRUE, the shards are labeled by the rownames of the preimages.
plot	logical. If FALSE, all graphical output is suppressed.
classes	a vector giving alternative classes for objects of class EDAM; classes have to be given in the original order of the data to which EDAM was applied.
vertices	logical. If TRUE the grid is drawn.
classcolors	colors to represent the classes, or a character giving the *colorscale* for the classes. Since now available scales are rainbow, topo and gray.
wghts	an optional vector of length k giving relative weights of the variables in computing Euclidean distances. Meaningless if object$preimages is a dissimilarity matrix.
xaxs	see par
yaxs	see par
xlab	see par
ylab	see par
...	further plotting parameters.
plot.data.column	column index defining from data.or providing the data used to calculate the coloring of the cells.
log.classes	boolean indicating that the data should be transformed with the logarithmic function before calculating the cell coloring
revert.colors	boolean indicating that the colorscale should be reverted.

Details

level_shardsplot uses multiple shardsplot representations of a SOM in order to depict how the data used to calculate the SOM is distribution across the map. Two representations are possible for the data, first with a single color ramp from the minimum value to the maximum value. The second representation is usefull for data for which a basic value exists some where between minimum and maximum for which a special color representation should be used (e.g. 0 is indicated with white).

If plot.type is "four" or "eight", the shape of each shard depends on the relative distances of the actual object or codebook to its up to eight neighbours. If plot.type is "eight", shardsplot corresponds to the representation method suggested by Cottrell and de Bodt (1996) for Kohonen Self-Organizing Maps. If plot.type is "points", shardsplot reduces to a usual scatter plot.

Value

The following list is (invisibly) returned:

Cells.ex	the images of the visualized data
S	the criterion of the visualization

Author(s)

Nils Raabe, level_shardsplot function from Dominik Reusser

References

Cottrell, M., and de Bodt, E. (1996). A Kohonen Map Representation to Avoid Misleading Interpretations. *Proceedings of the European Symposium on Atrificial Neural Networks*, D-Facto, pp. 103–110.

See Also

EDAM, TopoS, som

Examples

```
# Compute clusters and an Eight Directions Arranged Map for the
# country data. Plotting the result.
data(countries)
logcount <- log(countries[,2:7])
sdlogcount <- apply(logcount, 2, sd)
logstand <- t((t(logcount) / sdlogcount) * c(1,2,6,5,5,3))
cclasses <- cutree(hclust(dist(logstand)), k = 6)
countryEDAM <- EDAM(logstand, classes = cclasses, sa = FALSE,
    iter.max = 10, random = FALSE)
plot(countryEDAM, vertices = FALSE, label = TRUE, stck = FALSE)

# Compute and plot a Self-Organizing Map for the iris data
data(iris)
library(som)
irissom <- som(iris[,1:4], xdim = 6, ydim = 14)
shardsplot(irissom, data.or = iris, vertices = FALSE)
```

```
opar <- par(xpd = NA)
legend(7.5, 6.1, col = rainbow(3), xjust = 0.5, yjust = 0,
    legend = levels(iris[, 5]), pch = 16, horiz = TRUE)
par(opar)

level_shardsplot(irissom, par.names = names(iris),
    class.labels = NA, mfrow = c(2,2))
```

---

sknn *Simple k nearest Neighbours*

Description

Function for simple knn classification.

Usage

```
sknn(x, ...)

## Default S3 method:
sknn(x, grouping, kn = 3, gamma=0, ...)
## S3 method for class 'data.frame'
sknn(x, ...)
## S3 method for class 'matrix'
sknn(x, grouping, ..., subset, na.action = na.fail)
## S3 method for class 'formula'
sknn(formula, data = NULL, ..., subset, na.action = na.fail)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given).
grouping	factor specifying the class for each observation (required, if formula is not given).
formula	formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
data	Data frame from which variables specified in formula are preferentially to be taken.
kn	Number of nearest neighbours to use.
gamma	gamma parameter for rbf in knn. If gamma=0 ordinary knn classification is used.
subset	An index vector specifying the cases to be used in the training sample. (Note: If given, this argument must be named.)
na.action	specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (Note: If given, this argument must be named.)
...	currently unused

Details

If gamma>0 an gaussian like density is used to weight the classes of the kn nearest neighbors. weight=exp(-gamma*distance). This is similar to an rbf kernel. If the distances are large it may be useful to scale the data first.

Value

A list containing the function call.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

See Also

predict.sknn, knn

Examples

```
data(iris)
x <- sknn(Species ~ ., data = iris)
x <- sknn(Species ~ ., gamma = 4, data = iris)
```

---

stepclass *Stepwise variable selection for classification*

Description

Forward/backward variable selection for classification using any specified classification function and selecting by estimated classification performance measure from ucpm.

Usage

```
stepclass(x, ...)

## Default S3 method:
stepclass(x, grouping, method, improvement = 0.05, maxvar = Inf,
    start.vars = NULL, direction = c("both", "forward", "backward"),
    criterion = "CR",  fold = 10, cv.groups = NULL, output = TRUE,
    min1var = TRUE, ...)
## S3 method for class 'formula'
stepclass(formula, data, method, ...)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given).
formula	A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators. Interaction terms are not supported.
data	data matrix (rows=cases, columns=variables)
grouping	class indicator vector (a factor)
method	character, name of classification function (e.g. "lda").
improvement	least improvement of performance measure desired to include or exclude any variable (<=1)
maxvar	maximum number of variables in model
start.vars	set variables to start with (indices or names). Default is no variables if 'direction' is "forward" or "both", and all variables if 'direction' is "backward".
direction	"forward", "backward" or "both" (default)
criterion	performance measure taken from ucpm.
fold	parameter for cross-validation; omitted if 'cv.groups' is specified.
cv.groups	vector of group indicators for cross-validation. By default assigned automatically.
output	indicator (logical) for textoutput during computation (slows down computation!)
min1var	logical, whether to include at least one variable in the model, even if the prior itself already is a reasonable model.
...	further parameters passed to classification function ('method'), e.g. priors etc.

Details

The classification "method" (e.g. 'lda') must have its own 'predict' method (like 'predict.lda' for 'lda') that either returns a matrix of posterior probabilities or a list with an element 'posterior' containing that matrix instead. It must be able to deal with matrices as in method(x, grouping, ...)

Then a stepwise variable selection is performed. The initial model is defined by the provided starting variables; in every step new models are generated by including every single variable that is not in the model, and by excluding every single variable that is in the model. The resulting performance measure for these models are estimated (by cross-validation), and if the maximum value of the chosen criterion is better than 'improvement' plus the value so far, the corresponding variable is in- or excluded. The procedure stops, if the new best value is not good enough, or if the specified maximum number of variables is reached.

If 'direction' is "forward", the model is only extended (by including further variables), if 'direction' is "backward", the model is only reduced (by excluding variables from the model).

Value

An object of class 'stepclass' containing the following components:

call	the (matched) function call.
method	name of classification function used (e.g. "lda").
start.variables	vector of starting variables.
process	data frame showing selection process (included/excluded variables and performance measure).
model	the final model: data frame with 2 columns; indices and names of variables.
perfomance.measure	value of the criterion used by ucpm
formula	formula of the form 'response ~ list + of + selected + variables'

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>, Irina Czogiel

See Also

step, stepAIC, and greedy.wilks for stepwise variable selection according to Wilk's lambda

Examples

```
data(iris)
library(MASS)
iris.d <- iris[,1:4]  # the data
iris.c <- iris[,5]    # the classes
sc_obj <- stepclass(iris.d, iris.c, "lda", start.vars = "Sepal.Width")
sc_obj
plot(sc_obj)

## or using formulas:
sc_obj <- stepclass(Species ~ ., data = iris, method = "qda",
    start.vars = "Sepal.Width", criterion = "AS")  # same as above
sc_obj
## now you can say stuff like
## qda(sc_obj$formula, data = B3)
```

---

svmlight *Interface to SVMlight*

Description

Function to call SVMlight from R for classification. Multiple group classification is done with the one-against-rest partition of data.

Usage

```
svmlight(x, ...)

## Default S3 method:
svmlight(x, grouping, temp.dir = NULL, pathsvm = NULL,
    del = TRUE, type = "C", class.type = "oaa", svm.options = NULL,
    prior = NULL, out = FALSE, ...)
## S3 method for class 'data.frame'
svmlight(x, ...)
## S3 method for class 'matrix'
svmlight(x, grouping, ..., subset, na.action = na.fail)
## S3 method for class 'formula'
svmlight(formula, data = NULL, ..., subset,
    na.action = na.fail)
```

Arguments

x	matrix or data frame containing the explanatory variables (required, if formula is not given).
grouping	factor specifying the class for each observation (required, if formula is not given).
formula	formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
data	Data frame from which variables specified in formula are preferentially to be taken.
temp.dir	directory for temporary files.
pathsvm	Path to SVMlight binaries (required, if path is unknown by the OS).
del	Logical: whether to delete temporary files
type	Perform "C"=Classification or "R"=Regression
class.type	Multiclass scheme to use. See details.
svm.options	Optional parameters to SVMlight. For further details see: "How to use" on http://svmlight.joachims.org/.
prior	A Priori probabilities of classes.
out	Logical: whether SVMlight output ahouild be printed on console (only for Windows OS.)
subset	An index vector specifying the cases to be used in the training sample. (Note: If given, this argument must be named.)
na.action	specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (Note: If given, this argument must be named.)
...	currently unused

Details

Function to call SVMlight from R for classification (`type="C"`). SVMlight is an implementation of Vapnik's Support Vector Machine. It is written in C by Thorsten Joachims. On the homepage (see below) the source-code and several binaries for SVMlight are available. If more then two classes are given the SVM is learned by the one-against-all scheme (`class.type="oaa"`). That means that each class is trained against the other K-1 classes. The class with the highest decision function in the SVM wins. So K SVMs have to be learned. If `class.type="oao"` each class is tested against every other and the final class is elected by a majority vote.

If `type="R"` a SVM Regression is performed.

Value

A list containing the function call and the result of SVMlight.

Requirements

SVMlight (http://svmlight.joachims.org/) must be installed before using this interface.

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>, Andrea Preusser

References

http://svmlight.joachims.org/

See Also

predict.svmlight,svm,

Examples

```
## Not run:
## Only works if the svmlight binaries are in the path.
data(iris)
x <- svmlight(Species ~ ., data = iris)
## Using RBF-Kernel with gamma=0.1:
data(B3)
x <- svmlight(PHASEN ~ ., data = B3, svm.options = "-t 2 -g 0.1")

## End(Not run)
```

triframe *Barycentric plots*

Description

Function to add a frame to an existing (barycentric) plot.

Usage

```
triframe(label = 1:3, label.col = 1, cex = 1, ...)
```

Arguments

label	labels for the three corners of the plot.
label.col	text color for labels.
cex	Magnification factor for label text relative to the default.
...	Further graphical parameters passed to trilines.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

triplot, trilines, trigrid, centerlines

Examples

```
triplot(grid = TRUE, frame = FALSE)     # plot without frame
some.triangle <- rbind(c(0, 0.65, 0.35), c(0.53, 0.47, 0),
                       c(0.72, 0, 0.28))[c(1:3, 1), ]
trilines(some.triangle, col = "red", pch = 16, type = "b")
triframe(label = c("left", "top", "right"), col = "blue",
         label.col = "green3")          # frame on top of points
```

trigrid *Barycentric plots*

Description

Function to add a grid to an existing (barycentric) plot.

Usage

```
trigrid(x = seq(0.1, 0.9, by = 0.1), y = NULL, z = NULL,
    lty = "dashed", col = "grey", ...)
```

Arguments

x	Values along which to draw grid lines for first dimension (or all dimensions if y and z omitted). For NO grid lines in some dimensions just supply an NA.
y	Grid lines for second dimension.
z	Grid lines for third dimension.
lty	Line type (see par).
col	Line colour (see par).
...	Further graphical parameters passed to trilines.

Details

Grid lines illustrate the set of points for which one of the dimensions is held constant; e.g. horizontal lines contain all points with a certain value y for the second dimension, connecting the two extreme points (0,y,1-y) and (1-y,y,0).

Grids may be designed more flexible than with triplot's grid option.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

triplot, trilines, triframe, centerlines

Examples

```
triplot(grid = FALSE)
trigrid(c(1/3, 0.5)) # same grid for all 3 dimensions

triplot(grid = c(1/3, 0.5))  # (same effect)

triplot(grid = FALSE)
# different grids for all dimensions:
trigrid(x = 1/3, y = 0.5, z = seq(0.2, 0.8, by=0.2))

triplot(grid = FALSE)
# grid for third dimension only:
trigrid(x = NA, y = NA, z = c(0.1, 0.2, 0.4, 0.8))
```

`triperplines` *Barycentric plots*

Description

Function to add a point and the corresponding perpendicular lines to all three sides to an existing (barycentric) plot.

Usage

```
triperplines(x, y = NULL, z = NULL, lcol = "red", pch = 17, ...)
```

Arguments

`x`	fraction of first component OR 3-element vector (for all three components, omitting `y` and `z`).
`y`	(optional) fraction of second component.
`z`	(optional) fraction of third component.
`lcol`	line color
`pch`	plotting character. `pch = 0` for no point
`...`	Further graphical parameters (see `points`, `lines` and `par`).

Details

Adds a (single!) point and lines to an existing plot (generated by `triplot`). The lines originate from the point and run (perpendicular) towards all three sides. The lengths (and proportions) of these lines are identical to those of `x`, `y` and `z`.

Value

a 2-column-matrix containing plot coordinates.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

`triplot`, `tripoints`, `trilines`, `tritrafo`

Examples

```
triplot()  # empty plot
triperplines(1/2, 1/3, 1/6)
```

`triplot` *Barycentric plots*

Description

Function to produce triangular (barycentric) plots illustrating proportions of 3 components, e.g. discrete 3D-distributions or mixture fractions that sum up to 1.

Usage

```
triplot(x = NULL, y = NULL, z = NULL, main = "", frame = TRUE,
    label = 1:3, grid = seq(0.1, 0.9, by = 0.1), center = FALSE,
    set.par = TRUE, ...)
```

Arguments

`x`	Vector of fractions of first component OR 3-column matrix containing all three components (omitting `y` and `z`) OR 3-element vector (for all three components, omitting `y` and `z`).
`y`	(Optional) vector of fractions of second component.
`z`	(Optional) vector of fractions of third component.
`main`	Main title
`frame`	Controls whether a frame (triangle) and labels are drawn.
`label`	(Character) vector of labels for the three corners.
`grid`	Values along which grid lines are to be drawn (or `FALSE` for no grid at all). Default is steps of 10 percent.
`center`	Controls whether or not to draw centerlines at which there is a 'tie' between any two dimensions (see also `centerlines`).
`set.par`	Controls whether graphical parameter `mar` is set so the plot fills the window (see `par`).
`...`	Further graphical parameters passed to `trilines`.

Details

The barycentric plot illustrates the set of points (x,y,z) with x,y,z between 0 and 1 and x+y+z=1; that is, the triangle spanned by (1,0,0), (0,1,0) and (0,0,1) in 3-dimensional space. The three dimensions x, y and z correspond to lower left, upper and lower right corner of the plot. The greater the share of x in the proportion, the closer the point is to the lower left corner; Points on the opposite (upper right) side have a zero x-fraction. The grid lines show the points at which one dimension is held constant, horizontal lines for example contain points with a constant second dimension.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

tripoints, trilines, triperplines, trigrid, triframe for points, lines and layout, tritrafo for placing labels, and quadplot for the same in 4 dimensions.

Examples

```
# illustrating probabilities:
triplot(label = c("1, 2 or 3", "4 or 5", "6"),
    main = "die rolls: probabilities", pch = 17)
triperplines(1/2, 1/3, 1/6)

# expected...
triplot(1/2, 1/3, 1/6, label = c("1, 2 or 3", "4 or 5", "6"),
    main = "die rolls: expected and observed frequencies", pch = 17)
# ... and observed frequencies.
dierolls <- matrix(sample(1:3, size = 50*20, prob = c(1/2, 1/3, 1/6),
                          replace = TRUE), ncol = 50)
frequencies <- t(apply(dierolls, 1,
    function(x)(summary(factor(x, levels = 1:3)))) / 50)
tripoints(frequencies)

# LDA classification posterior:
data(iris)
require(MASS)
pred <- predict(lda(Species ~ ., data = iris),iris)
plotchar <- rep(1,150)
plotchar[pred$class != iris$Species] <- 19
triplot(pred$posterior, label = colnames(pred$posterior),
        main = "LDA posterior assignments", center = TRUE,
        pch = plotchar, col = rep(c("blue", "green3", "red"), rep(50, 3)),
        grid = TRUE)
legend(x = -0.6, y = 0.7, col = c("blue", "green3", "red"),
    pch = 15, legend = colnames(pred$posterior))
```

---

tripoints *Barycentric plots*

Description

Function to add points or lines to an existing (barycentric) plot.

Usage

```
tripoints(x, y = NULL, z = NULL, ...)
trilines(x, y = NULL, z = NULL, ...)
```

Arguments

x	Vector of fractions of first component OR 3-column matrix containing all three components (omitting y and z) OR 3-element vector (for all three components, omitting y and z).
y	(optional) vector of fractions of second component.
z	(optional) vector of fractions of third component.
...	Further graphical parameters (see points and par).

Details

Adds points or lines to an existing plot (generated by triplot).

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

points, lines, triplot, tritrafo, centerlines

Examples

```
triplot()  # empty plot
tripoints(0.1, 0.2, 0.7)                          # a point
tripoints(c(0.2, 0.6), c(0.3, 0.3), c(0.5, 0.1),
    pch = c(2, 6))                                # two points
trilines(c(0.1, 0.6), c(0.2, 0.3), c(0.7, 0.1),
    col = "blue", lty = "dotted")                 # a line

trilines(centerlines(3))
```

---

tritrafo *Barycentric plots*

Description

Function to carry out the transformation into 2D space for triplot, trilines etc.

Usage

```
tritrafo(x, y = NULL, z = NULL, check = TRUE, tolerance = 0.0001)
```

Arguments

x	Vector of fractions of first component OR 3-column matrix containing all three components (omitting y and z) OR 3-element vector (for all three components, omitting y and z).
y	(optional) vector of fractions of second component.
z	(optional) vector of fractions of third component.
check	if TRUE, it is checked whether x+y+z=1 and x,y,z>=0 for all cases.
tolerance	tolerance for above sum check.

Details

Projects the mixture given by x, y, and z with x, y, z between one and zero and x+y+z=1 into a two-dimensional space.

For further details see triplot.

Value

A matrix with two columns corresponding to the two dimensions.

Author(s)

Christian Röver, <roever@statistik.tu-dortmund.de>

See Also

triplot, tripoints, trilines, trigrid

Examples

```
tritrafo(0.1, 0.2, 0.7)
tritrafo(0.1, 0.2, 0.6) # warning

triplot()
points(tritrafo(0.1, 0.2, 0.7), col="red")
tripoints(0.1, 0.2, 0.7, col="green")  # the same

tritrafo(c(0.1,0.2), c(0.3,0.4), c(0.6,0.4))
tritrafo(diag(3))

point <- c(0.25,0.6,0.15)
triplot(point, pch=16)
text(tritrafo(point), "(0.25, 0.60, 0.15)", adj=c(0.5,2)) # add a label
```

ucpm *Uschi's classification performance measures*

Description

Function to calculate the Correctness Rate, the Accuracy, the Ability to Seperate and the Confidence of a classification rule.

Usage

```
ucpm(m, tc, ec = NULL)
```

Arguments

m	matrix of (scaled) membership values
tc	vector of true classes
ec	vector of estimated classes (only required if scaled membership values are used)

Details

- The *correctness rate* is the estimator for the correctness of a classification rule (1-error rate).
- The *accuracy* is based on the euclidean distances between (scaled) membership vectors and the vectors representing the true class corner. These distances are standardized so that a measure of 1 is achieved if all vectors lie in the correct corners and 0 if they all lie in the center.
- Analougously, the *ability to seperate* is based on the distances between (scaled) membership vectors and the vector representing the corresponding assigned class corner.
- The *confidence* is the mean of the membership values of the assigned classes.

Value

A list with elements:

CR	Correctness Rate
AC	Accuracy
AS	Ability to Seperate
CF	Confidence
CFvec	Confidence for each (true) class

Author(s)

Karsten Luebke, <karsten.luebke@fom.de>

References

Garczarek, Ursula Maria (2002): Classification rules in standardized partition spaces. Dissertation, University of Dortmund. URL http://hdl.handle.net/2003/2789

Examples

```
library(MASS)
data(iris)
ucpm(predict(lda(Species ~ ., data = iris))$posterior, iris$Species)
```

---

woe *Weights of evidence*

Description

Computes weight of evidence transform of factor variables for binary classification.

Usage

```
woe(x, ...)
## Default S3 method:
woe(x, grouping, weights = NULL, zeroadj = 0, ids = NULL,
                      appont = TRUE, ...)
## S3 method for class 'formula'
woe(formula, data = NULL, weights = NULL, ...)
```

Arguments

x	A matrix or data frame containing the explanatory variables.
grouping	A factor specifying the binary class for each observation.
formula	A formula of the form grouping ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the discriminators.
data	Data frame from which variables specified in formula are to be taken.
weights	Vector with observation weights. For call woe(formula, data, weights) also a a character is possible that specifies the name of the weight coloumn in data.
zeroadj	Additive constant to be added for a level with 0 observations in a class.
ids	Vector of either indices or variable names that specifies the variables to be transformed.
appont	Application on training data: logical indicating whether the transformed values for the training data should be returned by recursive calling of predict.woe.
...	For woe.formula: Further arguments passed to function woe.default such as ids. For woe.default: replace = FALSE can be passed to recursive call of predict.woe if appont.

Details

To each factor level x a numeric value $WOE(x) = ln(f(x|1)/f(x|2))$ is assigned where 1 and 2 denote the class labels. The WOE transform is motivated for subsequent modelling by logistic regression. Note that the frequencies of the classes should be investigated before. Information values heuristically quantify the discriminatory power of a variable by $IV = (f(x|1) - f(x|2))ln(f(x|1)/f(x|2))$.

Value

Returns an object of class *woe* that can be applied to new data.

woe	WOE coefficients for factor2numeric transformation of each (specified) variable.
IV	Vector of information values of all transformed variables.
newx	Data frame of transformed data if appont.

Author(s)

Gero Szepannek

References

Good, I. (1950): *Probability and the Weighting of Evidences.* Charles Griffin, London.

Kullback, S. (1959): *Information Theory and Statistics.* Wiley, New York.

See Also

predict.woe, plot.woe

Examples

```
## load German credit data
data("GermanCredit")

## training/validation split
train <- sample(nrow(GermanCredit), round(0.6*nrow(GermanCredit)))

woemodel <- woe(credit_risk~., data = GermanCredit[train,], zeroadj=0.5, applyontrain = TRUE)
woemodel

## plot variable information values and woes
plot(woemodel)
plot(woemodel, type = "woes")

## apply woes
traindata <- predict(woemodel, GermanCredit[train,], replace = TRUE)
str(traindata)

## fit logistic regression model
glmodel     <- glm(credit_risk~., traindata, family=binomial)
summary(glmodel)
pred.trn <- predict(glmodel, traindata, type = "response")

## predict validation data
validata <- predict(woemodel, GermanCredit[-train,], replace = TRUE)
pred.val <- predict(glmodel, validata, type = "response")
```

`xtractvars` *Variable clustering based variable selection*

Description

Applies variable selection to data based on variable clusterings as resulting from `corclust` or `CLV`.

Usage

```
xtractvars(object, data, thres = 0.5)
```

Arguments

`object`	Object of class `cvtree` applied to a `corclust` object or the `summary()` of a `clv` object as created by `CLV`.
`data`	Data where variables are to be selected. Coloumn names must be identical to those used in corclust model.
`thres`	Maximum accepted average within cluster correlation for selection of a variable.

Details

Of each cluster the first variable is selected as well as all other variables with an average within cluster correlation below `thres`.

Value

The data is returned where unselected coloumns are removed.

Author(s)

Gero Szepannek

References

Roever, C. and Szepannek, G. (2005): Application of a genetic algorithm to variable selection in fuzzy clustering. In C. Weihs and W. Gaul (eds), Classification - The Ubiquitous Challenge, 674-681, Springer.

See Also

See also `corclust`, `cvtree` and `CLV`.

Examples

```
data(B3)
ccres <- corclust(B3)
plot(ccres)
cvtres <- cvtree(ccres, k = 3)
newdata <- xtractvars(cvtres, B3, thres = 0.5)
```

Index

∗ **Ability to Seperate**
ucpm, 75
∗ **Accuracy**
ucpm, 75
∗ **Barycentric plots**
centerlines, 8
quadplot, 54
triframe, 68
trigrid, 68
triperplines, 70
triplot, 71
tripoints, 72
tritrafo, 73
∗ **Benchmark**
benchB3, 5
∗ **Beta scaling**
b.scal, 3
betascale, 7
∗ **Classification Performance Measures**
ucpm, 75
∗ **Classification in time series**
hmm.sop, 26
∗ **Classification**
b.scal, 3
betascale, 7
e.scal, 16
meclight.default, 33
predict.meclight, 44
predict.svmlight, 49
svmlight, 65
∗ **Cluster analysis**
EDAM, 17
shardsplot, 59
∗ **Collinearity diagnosis in regression**
cond.index, 10
∗ **Confidence**
ucpm, 75
∗ **Confusion matrix**
errormatrix, 20
∗ **Correctness Rate**
ucpm, 75
∗ **E scaling**
e.scal, 16
∗ **EDAM**
EDAM, 17
shardsplot, 59
∗ **Eight Direction Arranged Maps**
EDAM, 17
∗ **Friedman's classification benchmark data**
friedman.data, 21
∗ **HMM**
hmm.sop, 26
∗ **Hidden Markov Model**
hmm.sop, 26
∗ **KNN**
predict.sknn, 48
sknn, 62
∗ **Kernel density estimation**
dkernel, 14
∗ **Kernel estimated densities**
NaiveBayes, 34
predict.NaiveBayes, 45
∗ **Linear Dimension Reduction**
meclight.default, 33
predict.meclight, 44
∗ **Linear Discriminant Analysis**
loclda, 28
meclight.default, 33
predict.loclda, 42
predict.meclight, 44
predict.rda, 47
rda, 56
∗ **Localization**
loclda, 28
predict.loclda, 42
∗ **Localized Linear Discriminant Analysis**
loclda, 28
predict.loclda, 42

∗ **Membership values**
e.scal, 16
∗ **Naive Bayes Classification**
NaiveBayes, 34
predict.NaiveBayes, 45
∗ **Nearest Mean Classification**
nm, 36
∗ **Pairwise variable selection for classification**
locpvs, 31
predict.locpvs, 43
predict.pvs, 46
pvs, 51
∗ **Posterior probabilities**
b.scal, 3
betascale, 7
∗ **Quadratic Discriminant Analysis**
predict.rda, 47
rda, 56
∗ **Regularized Discriminant Analysis**
friedman.data, 21
predict.rda, 47
rda, 56
∗ **SOM**
shardsplot, 59
∗ **SVMlight**
predict.svmlight, 49
svmlight, 65
∗ **SVM**
predict.svmlight, 49
svmlight, 65
∗ **Softmax scaling**
e.scal, 16
∗ **Stepwise variable selection in classification**
greedy.wilks, 24
stepclass, 63
∗ **Support vector machines**
predict.svmlight, 49
svmlight, 65
∗ **Visualizing Classification Performance Measures**
centerlines, 8
quadplot, 54
triframe, 68
trigrid, 68
triperplines, 70
triplot, 71
tripoints, 72
tritrafo, 73
∗ **Visualizing Naive Bayes Classification**
plot.NaiveBayes, 40
∗ **Visualizing classification results**
classscatter, 9
errormatrix, 20
∗ **Vizualizing Eight Direction Arranged Maps**
shardsplot, 59
∗ **Vizualizing classification results**
drawparti, 15
partimat, 37
plineplot, 39
∗ **Wilk's lambda**
greedy.wilks, 24
∗ **aplot**
triframe, 68
trigrid, 68
triperplines, 70
triplot, 71
tripoints, 72
∗ **attribute**
corclust, 11
cvtree, 13
xtractvars, 78
∗ **category**
kmodes, 27
NaiveBayes, 34
predict.NaiveBayes, 45
∗ **classif**
b.scal, 3
benchB3, 5
betascale, 7
centerlines, 8
classscatter, 9
corclust, 11
cvtree, 13
drawparti, 15
e.scal, 16
hmm.sop, 26
locpvs, 31
meclight.default, 33
NaiveBayes, 34
nm, 36
partimat, 37
plineplot, 39
plot.NaiveBayes, 40
plot.woe, 41

predict.locpvs, 43
predict.meclight, 44
predict.NaiveBayes, 45
predict.pvs, 46
predict.sknn, 48
predict.svmlight, 49
predict.woe, 50
pvs, 51
quadplot, 54
sknn, 62
svmlight, 65
ucpm, 75
woe, 76
xtractvars, 78
∗ **cluster**
corclust, 11
cvtree, 13
kmodes, 27
xtractvars, 78
∗ **datasets**
B3, 4
countries, 12
GermanCredit, 23
∗ **distribution**
dkernel, 14
∗ **dplot**
centerlines, 8
classscatter, 9
drawparti, 15
partimat, 37
plineplot, 39
plot.NaiveBayes, 40
quadplot, 54
tritrafo, 73
∗ **hplot**
shardsplot, 59
∗ **k nearest neighbour classification**
predict.sknn, 48
sknn, 62
∗ **manip**
corclust, 11
cvtree, 13
xtractvars, 78
∗ **multivariate**
cond.index, 10
corclust, 11
cvtree, 13
EDAM, 17
errormatrix, 20
friedman.data, 21
greedy.wilks, 24
kmodes, 27
loclda, 28
plot.woe, 41
predict.loclda, 42
predict.rda, 47
predict.woe, 50
pvs, 51
rda, 56
stepclass, 63
woe, 76
xtractvars, 78
∗ **nonparametric**
dkernel, 14
∗ **robust against multicollinearity**
predict.rda, 47
rda, 56
∗ **ts**
calc.trans, 7
hmm.sop, 26

b.scal, 3, *7*
B3, 4, *5, 6*
benchB3, *5*, 5
betascale, *3, 4*, 7

calc.trans, 7, *26*
centerlines, 8, *68, 69*, *71*, *73*
chisq.test, *11*
classscatter, 9
CLV, *78*
cond.index, 10
contour, *16*
corclust, 11, *13, 14*, *78*
countries, 12
cvtree, *11–13*, 13, *78*

density, *14, 15*, *35*
dkernel, 14, *45, 46*
drawparti, 15, *38, 39*

e.scal, *4*, *7*, 16, *49*
EDAM, 17, *59–61*
errormatrix, 20

friedman.data, 21

GermanCredit, 23

greedy.wilks, 24, *31*, *51, 52*, *65*

hclust, *12*, *14*
hmm.sop, 26

image, *16*

kmodes, 27
knn, *37*, *49*, *63*
kproto, *27*
ks.test, *31*, *51, 52*

lda, *3*, *9*, *15*, *22*, *31*, *33, 34*, *38, 39*, *44*, *51, 52*, *58*, *64*
legend, *40*, *55*
level_shardsplot *(*shardsplot*)*, 59
lines, *70*, *73*
loclda, 28, *42, 43*
locpvs, 31, *43, 44*, *53*

manova, *10*, *25*
meclight, *44, 45*
meclight *(*meclight.default*)*, 33
meclight.default, 33
mosaicplot, *41*

na.omit, *36*, *38*, *62*, *66*
NaiveBayes, *15*, 34, *40, 41*, *46*, *52*
naiveBayes, *15*, *35*, *38*, *46*
nm, 36

optim, *33*
optimize, *57*

par, *39*, *60*, *69–71*, *73*
partimat, *16*, 37, *40*
pbeta, *3*
plineplot, 39
plot, *9*
plot.corclust, *12*, *14*
plot.corclust *(*corclust*)*, 11
plot.EDAM *(*shardsplot*)*, 59
plot.NaiveBayes, *35*, 40
plot.rda *(*rda*)*, 56
plot.stepclass *(*stepclass*)*, 63
plot.woe, 41, *51*, *77*
points, *70*, *73*
predict.lda, *52*, *64*
predict.loclda, *29*, *31*, 42
predict.locpvs, *32*, 43
predict.meclight, *34*, 44
predict.NaiveBayes, *35*, 45
predict.pvs, *44*, 46, *53*
predict.rda, 47, *58*
predict.sknn, 48, *63*
predict.svmlight, 49, *67*
predict.woe, *42*, 50, *77*
print.greedy.wilks *(*greedy.wilks*)*, 24
print.kmodes *(*kmodes*)*, 27
print.loclda *(*loclda*)*, 28
print.meclight *(*meclight.default*)*, 33
print.pvs *(*pvs*)*, 51
print.rda *(*rda*)*, 56
print.stepclass *(*stepclass*)*, 63
print.woe *(*woe*)*, 76
pvs, *32*, *44*, *46, 47*, 51

qbeta, *3*
qda, *15*, *22*, *35*, *38*, *46*, *52*, *58*
quadplot, *8*, 54, *72*
quadtrafo, *55*

randomForest, *52*
rda, *15*, *22*, *37, 38*, *48*, *52*, 56
rpart, *15*, *38*

sammon, *18*
scale, *6*, *36*, *63*
scatterplot3d, *54, 55*
shardsplot, *18, 19*, 59
sknn, *15*, *36–38*, *48, 49*, *52*, 62
som, *59*, *61*
step, *65*
stepAIC, *65*
stepclass, *10*, *25*, *31*, *51, 52*, 63
svm, *50*, *52*, *67*
svmlight, *15*, *38*, *49, 50*, 65

table, *21*
TopoS, *19*, *61*
triframe, 68, *69*, *72*
trigrid, *68*, 68, *72*, *74*
trilines, *68–74*
trilines *(*tripoints*)*, 72
triperplines, 70, *72*
triplot, *8*, *55*, *68–70*, 71, *73, 74*
tripoints, *70*, *72*, 72, *74*
tritrafo, *70*, *72, 73*, 73

ucpm, *63–65*, 75

woe, *42*, *50*, *51*, 76

xtractvars, *11*, *12*, 78