Package 'scrime'

December 1, 2018

Type Package

Title Analysis of High-Dimensional Categorical Data Such as SNP Data

Version 1.3.5

Date 2018-12-01

Author Holger Schwender, with a contribution of Arno Fritsch

Maintainer Holger Schwender <holger.schw@gmx.de>

Imports methods, stats, utils

Suggests MASS, RSQLite (>= 0.4-15), oligoClasses (>= 1.1.9), DBI

Description Tools for the analysis of high-dimensional data developed/implemented
at the group ``Statistical Complexity Reduction In Molecular Epidemiology'' (SCRIME).
Main focus is on SNP data. But most of the functions can also be applied to other
types of categorical data.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-01 11:00:03 UTC

R topics documented:

---

abf   *Approximate Bayes Factor*

Description

Computes the Approximate Bayes Factor proposed by Wakefield (2009) for test statistics `theta / sqrt(V)` that under the null hypothesis are assumed to follow an asymptotic standard normal distribution.

Usage

```
abf(theta, V, W, numerator = 0, pi1 = NA)
```

Arguments

theta   a vector of numeric values, e.g., the maximum likelihood estimates for the pa-
                 rameter of a logistic regression model computed by separately applying this
                 simple logistic regression to several SNPs. It is thus assumed that under the null
                 hypothesis `theta / sqrt(V)` is asymptotically standard normal distributed.

V   a vector of the same length as `theta` containing the variances of the estimates
                 comprised by `theta`.

W	the prior variance. Must be either a positive value or a vector of the same length as theta consisting of positive values.
numerator	either 0 or 1, specifying whether the numerator of the approximate Bayes factor comprises the probability for the null hypothesis or the probability for the alternative hypothesis.
pi1	either a numeric value between 0 and 1 specifying the prior probability of association or a vector of the same length as theta specifying for each of the SNPs a prior probability that this SNP is associated with the response. If specified, prior odds, posterior odds, and depending on numerator the Bayesian False Discovery Probability (numerator = 0) or the posterior probability of association (numerator = 1) are computed. If NA, only the approximate Bayes factors are returned.

Value

If pi1 = NA, a vector of the same length as theta containing the values of the approximate Bayes factor. If pi1 is specified, a list consisting of

ABF	a numeric vector containing the values of the approximate Bayes factors,
priorOdds	either a numeric value or a numeric vector comprising the prior odds of association (if numerator = 1) or no association (if numerator = 0),
postOdds	a numeric vector containing the posterior odds of association (if numerator = 1) or no association (if numerator = 0),

and either

BFDP	a numeric vector containing the Bayesian False Discovery Probabilities for the SNPs (if numerator = 0),

or

PPA	a numeric vector comprising the posterior probabilities of association (if numerator = 1)l

Author(s)

Holger Schwender, <holger.schw@gmx.de>

References

Wakefield, J. (2007). A Bayesian Measure of Probability of False Discovery in Genetic Epidemiology Studies. *American Journal of Human Genetics*, 81, 208-227.

---

analyse.models   *Summarize MCMC sample of Bayesian logic regression models*

Description

For an MCMC sample of Bayesian logic regression models obtained with `fblr` the distribution of the model size and the most common logic predictors with up to three binaries are reported.

Usage

```
analyse.models(file, size.freq = TRUE, moco = c(20, 10), int.freq = TRUE,
                 kmax = 10, int.level = 2, bin.names = NULL)
```

Arguments

`file`	character string naming file where MCMC output of `fblr` is stored.
`size.freq`	determines whether distribution of model size is reported as frequencies (default) or proportions.
`moco`	a vector of length 2 or 3 that determines how many of the most common main effects, two-factor interactions and (possibly) three-factor interactions are reported.
`int.freq`	determines whether the number (default) or the proportion of models containing a specific interaction is reported.
`kmax`	the maximum number of allowed logic predictors used in `fblr`.
`int.level`	the maximum number of allowed binaries in a logic predictor used in `fblr`.
`bin.names`	character vector of names for the binary variables. If no names are supplied, binaries are referred to with their indices.

Details

The logic regression models visited during the MCMC run are stored by `fblr` in the rows of a matrix in the following fashion: Position 1 contains the number of logic predictors in the model. The next `kmax * (int.level + 1)` positions contain the predictors, each predictor being coded as c(number of binaries in predictor, indices of binaries), where negative indices denote the complement of a variable. It follow the log-likelihood of the model, the value of the precision of the regression parameters and the `kmax+1` regression parameters. Zeros indicate empty entries. `analyse.models` extracts some of the most interesting information, namely which logic predictors occur most often in the visited models, from the sample. The complement of a binary is indicated with a minus sign preceding its name.

Value

`size`	table of model sizes.
`ones`	table of the `moco[1]` most common single-binary predictors.
`twos`	table of the `moco[2]` most common two-binaries predictors.
`threes`	table of the `moco[3]` most common three-binaries predictors.

Author(s)

Arno Fritsch, <arno.fritsch@uni-dortmund.de>

See Also

[fblr](), [predictFBLR]()

Examples

```
## Not run:
# Use fblr on some simulated SNP data
snp <- matrix(rbinom(500*20,2,0.3),ncol=20)
bin <- snp2bin(snp)
int <- apply(bin,1,function(x) (x[1] == 1 & x[3] == 0)*1)
case.prob <- exp(-0.5+log(5)*int)/(1+exp(-0.5+log(5)*int))
y <- rbinom(nrow(snp),1,prob=case.prob)
fblr(y, bin, niter=1000, nburn=0)

analyse.models("fblr_mcmc.txt")

# with SNP names
name.snp <- LETTERS[1:20]
name.bin <- paste(rep(name.snp,each=2), c("_d","_r"),sep="")

analyse.models("fblr_mcmc.txt", bin.names=name.bin)

## End(Not run)
```

---

buildSNPannotation   *Construct Annotation for Affymetrix SNP Chips*

Description

Constructs a data frame from a metadata package containing annotations for the SNPs from the corresponding Affymetrix SNP Chip.

Usage

```
buildSNPannotation(pkg, rs = TRUE, allele = TRUE, gene = TRUE,
  chromosome = FALSE, position = FALSE, strand = FALSE, cytoband = FALSE,
  max.genes = 0, lib.loc = NULL, others = NULL, subset = NULL,
  pattern = NULL, na.rm = TRUE)
```

Arguments

pkg	the name of the metadata package from which the data frame containing the annotations of the SNPs should be generated.
rs	should the RefSNP-ID of the SNPs be added to the data frame?

allele	should the two alleles of each SNP be added to the data frame?
gene	should the genes associated with the SNPs be added to the data frame?
chromosome	should the chromosome to which the respective SNP belongs be added to the data frame?
position	should the physical positions of the SNPs be added to the data frame?
strand	should the strands be added to the data frame?
cytoband	logical indicating whether the cytoband of each SNP is added to the data frame.
max.genes	integer specifying the maximum number of genes associated with the respective SNP that should be stored in the data frame. By default, all entries are considered. The corresponding column of the data frame can also be shortened afterwards using [shortenGeneDescription](). Shortened entries are marked by ... at the end of the entries.
lib.loc	the directory in which the metadata package is stored. Needs only to be specified if it is not stored in the usual directory of the packages.
others	character string or vector naming other entries of the object featureSet saved in pkg that should be added to the data frame.
subset	character string consisting of the probe set IDs of the SNPs for which the data frame should be generated. The data frame will contain all SNPs if subset = NULL
pattern	character string specifying the pattern of the probe set IDs of the SNPs for which the data frame should be generated. For example, pattern = "SNP%" will lead to a data frame containing all SNPs whose probe set ID start with "SNP".
na.rm	should the rows of the data frame corresponding to SNPs specified by subset for which no information is available in the metadata package be removed?

Value

A data frame composed of annotations for the SNPs for which information is available in the specified metadata package.

Author(s)

Holger Schwender, <holger.schw@gmx.de>

See Also

[shortenGeneDescription]()

colEpistatic   *Cordell's Test for Epistatic Interactions*

Description

Performs the likelihood ratio test for epistatic interactions proposed by Cordell (2002) for case-control data, where colEpistatic assumes that each column represents a SNP, and rowEpistatic assumes that each row represents a SNP.

Usage

```
colEpistatic(mat.snp, cl, genes = NULL, warnError = TRUE)
rowEpistatic(mat.snp, cl, genes = NULL, warnError = TRUE)

## S3 method for class 'colEpi'
print(x, top = 5, digits = 4, ...)
```

Arguments

mat.snp	a matrix containing genotype data, where the genotypes of each SNPs need to be coded by the number of minor alleles, i.e. 0, 1, and 2. Missing values are allowed. For colEpistatic, each column of mat.snp is assumed to represent a SNP, and each row a subject. For rowEpistatic, each row of mat.snp is assumed to represent a subject, and each column a SNP.
cl	a numeric vector of ones and zeros specifying which of the subjects in mat.snp are cases (coded by 1) and which are controls (coded by 0).
genes	a character vector containing the names of the genes (or, e.g., LD-blocks or pathways) to which the SNPs belong. If specified, only the two-way interactions between SNPs from different genetic sets (e.g., genes, LD-blocks, or pathways) are tested. If NULL, all two-way interactions between all possible pairs of SNPs are tested.
warnError	logical indicating whether the statistics for the gTDT for pairs of SNPs should be returned as NA if the fitting of one of the logistic regression models with [glm](#) returns warning. If warnError = FALSE, these warnings will be ignored, and the gTDT will be performed also for these SNP pairs.
x	an object of class colEpi, i.e. the output of either colEpistatic or rowEpistatic.
top	number of interactions that should be printed. If top is set to 0 or the number of interactions is smaller than or equal to top, then the statistics for all interactions are printed in the order of their computation. Otherwise, the top interactions with the smallest p-values are printed (sorted by their p-values).
digits	number of digits that should be printed.
...	ignored.

Value

An object of class `colEpi` consisting of

`ll.main`	a numeric vector containing the values of the maximized loglikelihoods of the logistic regression models considering only main effects,
`ll.full`	a numeric vector containing the values the maximized loglikelihoods of the logistic regression models additionally containing interaction terms,
`stat`	a vector comprising the values of the test statistic,
`pval`	a vector comprising the corresponding p-values,
`genes`	if genes has been specified, a vector composed of the gene pairs to which the SNP pairs belong,
`vec.error`	if `warnError = TRUE`, a list in which an entry corresponding to a SNP pair for which the fitting of (at least) one of the logistic regression models led to a warning shows this warning message of `glm`, and all other entries are `NULL`.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Cordell, H. J. (2002). Epistasis: What it Means, what it Doesn't mean, and Statistical Methods to Detect it in Humans. Human Molecular Genetics, 11, 2463-2468.

---

computeContCells   *Pairwise Contingency Tables*

Description

Computes a contingency table for each pair of rows of a matrix, and stores all contigency table in a matrix.

Usage

```
computeContCells(data, computeExp = TRUE, justDiag = FALSE,
    check = TRUE, n.cat = NULL)
```

Arguments

`data`	a numeric matrix consisting of integers between 1 and `n.cat`. It is assumed that each row of these matrix represents a variable. Missing values and different numbers of categories a variable can take are allowed.
`computeExp`	should the numbers of observations expected under the null hypothesis that the respective two variables are independent also be computed? Required when `computeContCells` is used to compute Pearson's χ^2-statistic.

justDiag	should only the diagonal elements of the contingency tables, i.e.\ n_{ii}, $i = 1, \ldots,$ n.cat, be computed?
check	should `data` be checked more thoroughly? It is highly recommended to use `check = TRUE`.
n.cat	integer specifying the maximum number of levels a variable can take. If `NULL`, this number will be computed. It is highly recommended not to change the default.

Value

A list consisting of two matrices each consisting of $m*(m-1)/2$ rows and n.cat^2 columns, where m is the number of rows of `data`. One of these matrices called `mat.obs` contains in each row the values of the contingency table for a particular pair of rows of `data`, where the contigency table of the variables represented by the *i*th and *j*th row of `data` is shown in the $j+m*(i-1)-i*(i-1)/2$ row of `mat.obs`. The other matrix called `mat.exp` consists of the corresponding numbers of observations expected under the null hypothesis that the respective two variables are independent.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). A Note on the Simultaneous Computation of Thousands of Pearson's χ^2-Statistics. *Technical Report*, SFB 475, Deparament of Statistics, University of Dortmund.

See Also

[computeContClass](#), [rowChisqStats](#)

Examples

```
## Not run:
# Generate an example data set consisting of 5 rows (variables)
# and 200 columns (observations) by randomly drawing integers
# between 1 and 3.

mat <- matrix(sample(3, 1000, TRUE), 5)

# Generate the matrix containing the contingency tables for each
# pair of rows of mat.

out <- computeContCells(mat)

# out contains both the observed numbers of observations
# summarized by contingency tables

out$mat.obs

# and the number of observations expected under the null hypothesis
# of independence.
```

```
out$mat.exp

# If, e.g., only the observed number of observations having the same
# value is of interest, call

computeContCells(mat, computeExp = FALSE, justDiag = TRUE)


## End(Not run)
```

---

computeContClass   *Rowwise Contigency Tables*

Description

Generates a matrix containing a contingency table for each row of a matrix and a vector of class labels.

Usage

```
computeContClass(data, cl, n.cat)
```

Arguments

data	a numeric matrix consisting of integers between 1 and n.cat. Each row of data is assumed to represent a variable, and each column to represent an observation. Missing values are not allowed. All variables must comprise the same number of levels.
cl	a numeric vector of length ncol(data) specifying the class labels of the observations represented by the columns of data. cl must consist of integers between 1 and n_{cl}, where n_{cl} is the number of classes.
n.cat	an integer giving the number of levels the variables can take. If not specified, n.cat will be determined automatically. It is highly recommended not to specify n.cat.

Value

A list composed of the following two matrices:

mat.obs	a matrix consisting of m rows and n.cat $* n_{cl}$ columns, where m is the number of variables, i.e.\ the number of rows of data. Each row of data shows the contingency table of cl and the corresponding row of data.
mat.exp	a matrix of the same size as mat.obs containing the numbers of observations expected under the null hypothesis of equal distribution in all classes that correspond to the respective entries in mat.obs.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). A Note on the Simultaneous Computation of Thousands of Pearson's χ^2-Statistics. *Technical Report*, SFB 475, Deparment of Statistics, University of Dortmund.

See Also

[computeContCells](), [rowChisqStats]()

Examples

```
## Not run:
# Generate an example data set consisting of 10 rows (variables)
# and 200 columns (observations) by randomly drawing integers
# between 1 and 3, and a vector of class labels of length 200
# indicating that the first 100 observation belong to class 1
# and the other 100 to class 2.

mat <- matrix(sample(3, 2000, TRUE), 10)
cl <- rep(1:2, e = 100)

# Applying computeContClass to this data set

out <- computeContClass(mat, cl)

# generates the observed numbers of observations

out$mat.obs

# and the corresponding expected numbers of observations.

out$mat.exp


## End(Not run)
```

---

fblr   *Full Bayesian Logic Regression for SNP Data*

Description

Performs full Bayesian logic regression for Single Nucleotide Polymorphism (SNP) data as described in Fritsch and Ickstadt (2007).

`fblr.weight` allows to incorporate prior pathway information by restricting search for interactions to specific groups of SNPs and/or giving them different weights. `fblr.weight` is only implemented for an interaction level of 2.

Usage

```
fblr(y, bin, niter, thin = 5, nburn = 10000, int.level = 2, kmax = 10,
  geo = 1, delta1 = 0.001, delta2 = 0.1, predict = FALSE,
  file = "fblr_mcmc.txt")

fblr.weight(y, bin, niter, thin = 5, nburn = 10000, kmax = 10, geo = 1,
  delta1 = 0.001, delta2 = 0.1, predict = FALSE, group = NULL,
  weight = NULL, file = "fblr_mcmc.txt")
```

Arguments

y	binary vector indicating case-control status.
bin	binary matrix with number of rows equal to `length(y)`. Usually the result of applying snp2bin to a matrix of SNP data.
niter	number of MCMC iterations after burn-in.
thin	after burn-in only every `thin`th iteration is kept.
nburn	number of burn-in iterations.
int.level	maximum number of binaries allowed in a logic predictor. Is fixed to 2 for `fblr.weight`.
kmax	maximum number of logic predictors allowed in the model.
geo	geometric penalty parameter for the number of binaries in a predictor. Value between 0 and 1. Default is 1, meaning no penalty.
delta1	shape parameter for hierarchical gamma prior on precision of regression parameters.
delta2	rate parameter for hierarchical gamma prior on precision of regression parameters.
predict	should predicted case probabilities be returned?
file	character string naming a file to write the MCMC output to. If `fblr` is called again, the file is overwritten.
group	list containing vectors of indices of binaries that are allowed to interact. Groups may be overlapping, but every binary has to be in at least one group. Groups have to contain at least two binaries. Defaults to NULL, meaning that all interactions are allowed.
weight	vector of length `ncol(bin)` containing different relative prior weights for binaries. Defaults to NULL, meaning equal weight for all binaries.

Details

The MCMC output in `file` can be analysed using the function `analyse.models`. In the help of this function it is also described how the models are stored in `file`.

Value

accept	acceptance rate of MCMC algorithm.
pred	vector of predicted case probabilities. Only given if `predict = TRUE`.

Author(s)

Arno Fritsch, <arno.fritsch@uni-dortmund.de>

References

Fritsch, A. and Ickstadt, K.\ (2007). Comparing logic regression based methods for identifying SNP interactions. In *Bioinformatics in Research and Development*, Hochreiter, S.\ and Wagner, R.\ (Eds.), Springer, Berlin.

See Also

[analyse.models,predictFBLR](#)

Examples

```
## Not run:
# SNP dataset with 500 persons and 20 SNPs each,
# a two-SNP interaction influences the case probability
snp <- matrix(rbinom(500*20,2,0.3),ncol=20)
bin <- snp2bin(snp)
int <- apply(bin,1,function(x) (x[1] == 1 & x[3] == 0)*1)
case.prob <- exp(-0.5+log(5)*int)/(1+exp(-0.5+log(5)*int))
y <- rbinom(nrow(snp),1,prob=case.prob)

# normally more iterations should be used
fblr(y, bin, niter=1000, nburn=0)
analyse.models("fblr_mcmc.txt")

# Prior information: SNPs 1-10 belong to genes in one pathway,
# SNPs 8-20 to another. Only interactions within a pathway are
# considered and the first pathway is deemed to be twice as
# important than the second.
fblr.weight(y,bin,niter=1000, nburn=0, group=list(1:20, 15:40),
  weight=c(rep(2,20),rep(1,20)))
analyse.models("fblr_mcmc.txt")


## End(Not run)
```

---

gknn   *Generalized k Nearest Neighbors*

Description

Predicts the classes of new observations with k Nearest Neighbors based on an user-specified distance measure.

Usage

```
gknn(data, cl, newdata, nn = 5, distance = NULL, use.weights = FALSE, ...)
```

Arguments

data	a numeric matrix in which each row represents an observation and each column a variable. If distance is "smc", "cohen" or "pcc", the values in data must be integers between 1 and n_{cat}, where n_{cat} is the maximum number of levels one of the variables can take. Missing values are allowed.
cl	a numeric vector of length nrow(data) giving the class labels of the observations represented by the rows of data. cl must consist of integers between 1 and n_{cl}, where n_{cl} is the number of groups.
newdata	a numeric matrix in which each row represents a new observation for which the class label should be predicted and each column consists of the same variable as the corresponding column of data.
nn	an integer specifying the number of nearest neighbors used to classify the new observations.
distance	character vector naming the distance measure used to identify the nn nearest neighbors. Must be one of "smc", "cohen", "pcc", "euclidean", "maximum", "manhattan", "canberra", and "minkowski". If NULL, it is determined in an ad hoc way if the data seems to be categorical. If this is the case distance is set to "smc". Otherwise, it is set to "euclidean".
use.weights	should the votes of the nearest neighbors be weighted by the reciprocal of the distances to the new observation when the class of a new observation should be predicted?
...	further arguments for the distance measure. If, e.g., distance = "minkowski", then p can also be specified, see [dist](). If distance = "pcc", then version can also be specified, see [pcc]().

Value

The predicted classes of the new observations.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

See Also

[knncatimpute](), [smc](), [pcc]()

Examples

```
## Not run:
# Using the example from the function knn.
```

```
library(class)
data(iris3)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- c(rep(2, 25), rep(1, 25), rep(1, 25))

knn.out <- knn(train, test, as.factor(cl), k = 3, use.all = FALSE)
gknn.out <- gknn(train, cl, test, nn = 3)

# Both applications lead to the same predictions.

knn.out == gknn.out

# But gknn allows to use other distance measures than the Euclidean
# distance. E.g., the Manhattan distance.

gknn(train, cl, test, nn = 3, distance = "manhattan")


## End(Not run)
```

---

identifyMonomorphism   *Identification of Constant Variables*

Description

Identifies the rows of a matrix that only show one level.

Usage

```
identifyMonomorphism(x)
```

Arguments

x   a matrix consisting of integers between 1 and n_{cat}, where n_{cat} is the number
                 of levels the variables in x can take. Each row of x is assumed to represent one
                 variable. Missing values are allowed.

Value

Numeric vector containing the rows of x showing only one level.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

Examples

```
## Not run:
# Generate a data set consisting of 10 rows and 15 columns,
# where the values are randomly drawn from the integers 1, 2, and 3,
# and row 3 and 7 consist only of one level.

mat <- matrix(sample(3, 2000, TRUE), 10)
mat[3, ] <- 1
mat[7, ] <- 2

identifyMonomorphism(mat)


## End(Not run)
```

---

knncatimpute	*Missing Value Imputation with kNN*

Description

Imputes missing values in a matrix composed of categorical variables using k Nearest Neighbors.

Usage

```
knncatimpute(x, dist = NULL, nn = 3, weights = TRUE)
```

Arguments

x	a numeric matrix containing missing values. All non-missing values must be integers between 1 and n_{cat}, where n_{cat} is the maximum number of levels the categorical variables in x can take. If the k nearest observations should be used to replace the missing values of an observation, then each row must represent one of the observations and each column one of the variables. If the k nearest variables should be used to impute the missing values of a variable, then each row must correspond to a variable and each column to an observation.
dist	either a character string naming the distance measure or a distance matrix. If the former, dist must be either "smc", "cohen", or "pcc". If the latter, dist must be a symmetric matrix having the same number of rows as x. In this case, both the upper and the lower triangle of dist must contain the distances, and the row and column names of dist must be equal to the row names of x. If NULL, dist = "smc" is used.
nn	an integer specifying k, i.e.\ the number of nearest neighbors, used in the imputation of the missing values.
weights	should weighted kNN be used to impute the missing values? If TRUE, the vote of each nearest neighbor is weighted by the reciprocal of its distance to the observation or variable when the missing values of this observation or variable, respectively, are replaced.

Value

A matrix of the same size as x in which all the missing values have been imputed.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

See Also

knncatimputeLarge, gknn, smc, pcc

Examples

```
## Not run:
# Generate a data set consisting of 200 rows and 50 columns
# in which the values are integers between 1 and 3.
# Afterwards, remove 20 of the values randomly.

mat <- matrix(sample(3, 10000, TRUE), 200)
mat[sample(10000, 20)] <- NA

# Replace the missing values.

mat2 <- knncatimpute(mat)

# Replace the missing values using the 5 nearest neighbors
# and Cohen's Kappa.

mat3 <- knncatimpute(mat, nn = 5, dist = "cohen")


## End(Not run)
```

| knncatimputeLarge | *Missing Value Imputation with kNN for High-Dimensional Data* |

Description

Imputes missing values in a high-dimensional matrix composed of categorical variables using k Nearest Neighbors.

Usage

```
knncatimputeLarge(data, mat.na = NULL, fac = NULL, fac.na = NULL,
    nn = 3, distance = c("smc", "cohen", "snp1norm", "pcc"),
    n.num = 100, use.weights = TRUE, verbose = FALSE)
```

Arguments

data	a numeric matrix consisting of integers between 1 and n_{cat}, where n_{cat} is maximum number of levels the categorical variables can take. If mat.na is specified, data is assumed to contain only non-missing data, and the rows of data are used to impute the missing values in mat.na. Otherwise, data is also allowed to contain missing values, and the missing values in the rows of data are imputed by employing the rows of data showing no missing values.
	Each row of data represents one of the objects that should be used to identify the k nearest neighbors, i.e.\ if the k nearest variables should be used to replace the missing values, then each row must represent one of the variables. If the k nearest observations should be used to impute the missing values, then each row must correspond to one of the observations.
mat.na	a numeric matrix containing missing values. Must have the same number of columns as data. All non-missing values must be integers between 1 and n_{cat}. If NULL, data is assumed to also contain the rows with missing values.
fac	a numeric or character vector of length nrow{data} specifying the values of a factor used to split data into subsets. If, e.g., the values of fac are given by the chromosomes to which the SNPs represented by the rows of data belong, then k nearest neighbors is applied chromosomewise to the missing values in mat.na (or data). If NULL, no such splitting is done. Must be specified, if fac.na is specified.
fac.na	a numeric or character vector of length nrow{mat.na} specifying the values of a factor by which mat.na is split into subsets. Each possible value of fac.na must be at least nn times in fac. Must be specified, if fac and mat.na is specified. If both fac and fac.na are NULL, then no splitting is done.
nn	an integer specifying k, i.e.\ the number of nearest neighbors, used to impute the missing values.
distance	character string naming the distance measure used in k Nearest Neighbors. Must be either "smc" (default), "cohen", "snp1norm" (which denotes the Manhattan distance for SNPs), or "pcc".
n.num	an integer giving the number of rows of mat.na considered simultaneously when replacing the missing values in mat.na.
use.weights	should weighted k nearest neighbors be used to impute the missing values? If TRUE, the votes of the nearest neighbors are weighted by the reciprocal of their distances to the variable (or observation) whose missing values are imputed.
verbose	should more information about the progress of the imputation be printed?

Value

If `mat.na` = `NULL`, then a matrix of the same size as `data` in which the missing values have been replaced. If `mat.na` has been specified, then a matrix of the same size as `mat.na` in which the missing values have been replaced.

Note

While in `knncatimpute` all variable/rows are considered when replacing missing values, `knncatimputeLarge` only considers the rows with no missing values when searching for the k nearest neighbors.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H. and Ickstadt, K.\ (2008). Imputing Missing Genotypes with k Nearest Neighbors. *Technical Report*, SFB 475, Department of Statistics, University of Dortmund. Appears soon.

See Also

knncatimpute, gknn, smc, pcc

Examples

```
## Not run:
# Generate a data set consisting of 100 columns and 2000 rows (actually,
# knncatimputeLarge is made for much larger data sets), where the values
# are randomly drawn from the integers 1, 2, and 3.
# Afterwards, remove 200 of the observations randomly.

mat <- matrix(sample(3, 200000, TRUE), 2000)
mat[sample(200000, 20)] <- NA

# Apply knncatimputeLarge to mat to remove the missing values.

mat2 <- knncatimputeLarge(mat)
sum(is.na(mat))
sum(is.na(mat2))

# Now assume that the first 100 rows belong to SNPs from chromosome 1,
# the second 100 rows to SNPs from chromosome 2, and so on.

chromosome <- rep(1:20, e = 100)

# Apply knncatimputeLarge to mat chromosomewise, i.e. only consider
# the SNPs that belong to the same chromosome when replacing missing
# genotypes.

mat4 <- knncatimputeLarge(mat, fac = chromosome)
```

```
## End(Not run)
```

---

pamCat   *Prediction Analysis of Categorical Data*

Description

Performs a Prediction Analysis of Categorical Data.

Usage

```
pamCat(data, cl, theta = NULL, n.theta = 10, newdata = NULL, newcl = NULL)
```

Arguments

data	a numeric matrix composed of the integers between 1 and n_{cat}, where n_{cat} is the number of levels each of the variables represented by the rows of data must take. No missing values allowed.
cl	a numeric vector of length ncol(data) comprising the class labels of the observations represented by the columns of data. cl must consist of the integers between 1 and n_{cl}, where n_{cl} is the number of classes.
theta	a numeric vector consisting of the strictly positive values of the shrinkage parameter used in the Prediction Analysis. If NULL, a vector consisting of n.theta values for the shrinkage parameter are determined automatically.
n.theta	an integer specifying the number of values for the shrinkage parameter of the Prediction Analysis. Ignored if theta is specified.
newdata	a numeric matrix composed of the integers between 1 and n_{cat}. Must have the same number of rows as data, and each row of newdata must contain the same variable as the corresponding row of data. newdata is employed to compute the misclassification rates of the Prediction Analysis for the given values of the shrinkage parameter. If NULL, data is used to determine the misclassification rates.
newcl	a numeric vector of length ncol(newdata) that consists of integers between 1 and n_{cl}, and specifies the class labels of the observations in newdata. Must be specified, if newdata is specified.

Value

An object of class pamCat composed of

mat.chisq	a matrix with m rows and n_{cl} columns consisting of the classwise values of Pearson's χ^2 statistic for each of the m variables.
mat.obs	a matrix with m rows and $n_{cat} * n_{cl}$ columns in which each row shows a contingency table between the corresponding variable and cl.

mat.exp	a matrix of the same size as mat.obs containing the numbers of observations expected under the null hypothesis of an association between the respective variable and cl.
mat.theta	a data frame consisting of the numbers of variables used in the classification of the observations in newdata and the corresponding misclassification rates for a set of values of the shrinkage parameter θ.
tab.cl	a table summarizing the values of the response, i.e.\ the class labels.
n.cat	n_{cat}.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

See Also

predict.pamCat

Examples

```
## Not run:
# Generate a data set consisting of 2000 rows (variables) and 50 columns.
# Assume that the first 25 observations belong to class 1, and the other
# 50 observations to class 2.

mat <- matrix(sample(3, 100000, TRUE), 2000)
rownames(mat) <- paste("SNP", 1:2000, sep = "")
cl <- rep(1:2, e = 25)

# Apply PAM for categorical data to this matrix, and compute the
# misclassification rate on the training set, i.e. on mat.

pam.out <- pamCat(mat, cl)
pam.out

# Now generate a new data set consisting of 20 observations,
# and predict the classes of these observations using the
# value of theta that has led to the smallest misclassification
# rate in pam.out.

mat2 <- matrix(sample(3, 40000, TRUE), 2000)
rownames(mat2) <- paste("SNP", 1:2000, sep = "")
predict(pam.out, mat2)

# Let's assume that the predicted classes are the real classes
# of the observations. Then, mat2 can also be used in pamCat
# to compute the misclassification rate.
```

```
cl2 <- predict(pam.out, mat2)
pamCat(mat, cl, newdata = mat2, newcl = cl2)


## End(Not run)
```

| pcc | *Pearson's Contingency Coefficient* |

Description

Computes the values of (the corrected) Pearson's contingency coefficient for all pairs of rows of a matrix.

Usage

```
pcc(x, dist = FALSE, corrected = TRUE, version = 1)
```

Arguments

x	a numeric matrix consisting of integers between 1 and n_{cat}, where n_{cat} is the maximum number of levels a variable in x can take.
dist	should the distance based on Pearson's contingency coefficient be computed? For how this distance is computed, see version.
corrected	should Pearson's contingency coefficient be corrected such that it can take values between 0 and 1? If not corrected, it takes values between and 0 and $\sqrt{(a-1)/a}$, where a is the minimum of the numbers of levels that the respective two variables can take. Must be set to TRUE, if dist = TRUE.
version	a numeric value – either 1, 2, or 3 – specifying how the distance is computed. Ignored if dist = FALSE. If 1, $\sqrt{1 - Cont^2}$ is computed, where $Cont$ denotes Pearson's contigency coefficient. If 2, $1 - Cont$ is determined, and if 3, $1 - Cont^2$ is returned.

Value

A matrix with nrow(x) columns and rows containing the values of (or distances based on) the (corrected) Pearson's contigency coefficient for all pairs of rows of x.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[smc](#)

Examples

```
## Not run:
# Generate a data set consisting of 10 rows and 200 columns,
# where the values are randomly drawn from the integers 1, 2, and 3.

mat <- matrix(sample(3, 2000, TRUE), 10)

# For each pair of rows of mat, the value of the corrected Pearson's
# contingency coefficient is then obtained by

out1 <- pcc(mat)
out1

# and the distances based on this coefficient by

out2 <- pcc(mat, dist = TRUE)
out2

# Note that if version is set to 1 (default) in pcc, then

all.equal(sqrt(1 - out1^2), out2)


## End(Not run)
```

---

predict.pamCat	*Predict Method for pamCat Objects*

Description

Predicts the classes of new observations based on a Prediction Analysis of Categorical Data.

Usage

```
## S3 method for class 'pamCat'
predict(object, newdata, theta = NULL, add.nvar = FALSE,
    type = c("class", "prob"), ...)
```

Arguments

object   an object of class pamCat.

newdata   a numeric matrix consisting of the integers between 1 and n_{cat}, where n_{cat} is
               the number of levels each of the variables in newdata must take. Each row of
               newdata must represent the same variable as the corresponding row in the matrix
               data used to produce object. Each column corresponds to an observation for
               which the class should be predicted.

theta	a strictly positive numeric value specifying the value of the shrinkage parameter of the Prediction Analysis that should be used in the class prediction. If NULL, then the value of theta will be used that has led to the smallest misclassification rate in the initial Prediction Analysis generating object.
add.nvar	should the number of variables used in the class prediction be added to the output?
type	either ″class″ or ″prob″. If ″class″, then the predicted classes will be returned. Otherwise, the probabilities for the classes are returned.
...	Ignored.

Value

If add.nvar = FALSE, the predicted classes or the class probabilities (depending on type). Otherwise, a list consisting of

pred	a vector or matrix containing the predicted classes or the class probabilities, respectively.
n.var	the number of variables used in the prediction.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

See Also

[pamCat](#)

Examples

```
## Not run:
# Generate a data set consisting of 2000 rows (variables) and 50 columns.
# Assume that the first 25 observations belong to class 1, and the other
# 50 observations to class 2.

mat <- matrix(sample(3, 100000, TRUE), 2000)
rownames(mat) <- paste("SNP", 1:2000, sep = "")
cl <- rep(1:2, e = 25)

# Apply PAM for categorical data to this matrix, and compute the
# misclassification rate on the training set, i.e. on mat.

pam.out <- pamCat(mat, cl)
pam.out

# Now generate a new data set consisting of 20 observations,
```

```
# and predict the classes of these observations using the
# value of theta that has led to the smallest misclassification
# rate in pam.out.

mat2 <- matrix(sample(3, 40000, TRUE), 2000)
rownames(mat2) <- paste("SNP", 1:2000, sep = "")
predict(pam.out, mat2)

# Another theta, say theta = 4, can also be specified.

predict(pam.out, mat2, theta = 4)

# The class probabilities for each observation can be obtained by

predict(pam.out, mat2, theta = 4, type = "prob")


## End(Not run)
```

| predictFBLR | *Predict Case Probabilities with Full Bayesian Logic Regression* |

Description

Predicts case probabilities for binary data (usually SNP data dichotomized with snp2bin) based on an MCMC sample of Bayesian logic regression models obtained with fblr.

Usage

```
predictFBLR(file, bin, kmax = 10, int.level = 2)
```

Arguments

file	character string naming file where MCMC sample is stored.
bin	matrix of binary variables to make predictions for. One row is one observation. The number of binary variables has to be the same as used in fblr.
kmax	the maximum number of allowed logic predictors used in fblr.
int.level	the maximum number of allowed binaries in a logic predictor used in fblr.

Value

Vector of length nrow(bin) with predicted case probabilities.

Author(s)

Arno Fritsch, <arno.fritsch@uni-dortmund.de>

See Also

[fblr](#)

Examples

```
## Not run:
# Use fblr on some simulated SNP data
snp <- matrix(rbinom(500 * 20, 2, 0.3), ncol = 20)
bin <- snp2bin(snp)
int <- apply(bin,1,function(x) (x[1] == 1 & x[3] == 0)*1)
case.prob <- exp(-0.5+log(5)*int)/(1+exp(-0.5+log(5)*int))
y <- rbinom(nrow(snp),1,prob=case.prob)
fblr(y, bin, niter=1000, nburn=0)

# Prediction for some new observations
newbin <- snp2bin(matrix(rbinom(100 * 20, 2, 0.3), ncol = 20))
predictFBLR("fblr_mcmc.txt",newbin)


## End(Not run)
```

---

recodeAffySNP   *Recoding of Affymetrix SNP Values*

Description

Recodes the values used on Affymetrix SNP chips to code the genotypes to other values – required, e.g., by other functions of this package.

Usage

```
recodeAffySNP(mat, refAA = FALSE, geno = 1:3)
```

Arguments

mat   a matrix or data frame consisting of the character strings "AA", "AB", "BB".
               Missing values can be coded by either "NN" or NA. Each row is assumed to
               correspond to a variable, and each column to a microarray.

refAA   codes "AA" always for the homozygous reference genotype? If TRUE, "AA" is
               always replaced by geno[1], and "BB" by geno[3]. If FALSE, it is evaluated
               rowwise whether "AA" or "BB" occurs more often, and the more frequently oc-
               curing value is set to geno[1].

geno   a numeric or character vector of length 3 giving the three values that should be
               used to recode the genotypes. By default, geno = 1:3 which is the coding, e.g.,
               required by [rowChisqStats](#) or [pamCat](#).

Value

A matrix of the same size as mat containing the recoded genotypes. (Missing values are coded by NA.)

See Also

recodeSNPs, snp2bin

Examples

```
## Not run:
# Generate a sample data set consisting of 10 rows and 12 columns,
# and randomly replace 5 of the values by "NN".

mat <- matrix("", 10, 12)
mat[1:5,] <- sample(c("AA", "AB", "BB"), 60, TRUE,
   prob = c(0.49, 0.42, 0.09))
mat[6:10,] <- sample(c("AA", "AB", "BB"), 60, TRUE,
   prob = c(0.09, 0.42, 0.49))
mat[sample(120, 5)] <- "NN"
mat

# Recode the SNPs.

recodeAffySNP(mat)

# Recode the SNPs assuming that "A" is always the major allele.

recodeAffySNP(mat, refAA = TRUE)


## End(Not run)
```

---

recodeSNPs   *Recoding of SNP Values*

Description

Recodes the values used to specify the genotypes of the SNPs to other values. Such a recoding might be required to use other functions contained in this package.

Usage

```
recodeSNPs(mat, first.ref = FALSE, geno = 1:3, snp.in.col = FALSE)
```

Arguments

mat	a matrix or data frame consisting of character strings of length 2 that specify the genotypes of the SNPs. Each of these character strings must be a combination of the letters A, T, C, and G. Missing values can be specified by "NN" or NA. Depending on snp.in.col it is assumed that each row of mat represents a SNP and each column a variable (snp.in.col = FALSE), or vice versa.
first.ref	does the first letter in the string coding the heterozygous genotype always stands for the more frequent allele? E.g., codes "CC" for the homozygous reference genotype if the genotypes of a SNP are coded by "CC", "CG" and "GG"? If TRUE, the value made up only of this first letter is set to geno[1], and the value made up only of the second letter is set to geno[3]. If FALSE, it is evaluated rowwise which of the homozygous genotypes has the higher frequency and the more often occuring value is set to geno[1], and the other to geno[3].
geno	a numeric or character vector of length 3 giving the three values that should be used to recode the genotypes. By default, geno = 1:3 which is the coding, e.g., required by rowChisqStats or pamCat.
snp.in.col	does each column of mat correspond to a SNP (and each row to an array)? If FALSE, it is assumed that each row represents a SNP, and each column an array.

Value

A matrix of the same size as mat containing the recoded genotypes. (Missing values are coded by NA).

See Also

recodeAffySNP, snp2bin

Examples

```
## Not run:
# Generate an example data set consisting of 5 rows and 12 columns,
# where it is assumed that each row corresponds to a SNP.

mat <- matrix("", 10, 12)
mat[c(1, 4, 6),] <- sample(c("AA", "AT", "TT"), 18, TRUE)
mat[c(2, 3, 10),] <- sample(c("CC", "CG", "GG"), 18, TRUE)
mat[c(5, 8),] <- sample(c("GG", "GT", "TT"), 12, TRUE)
mat[c(7, 9),] <- sample(c("AA", "AC", "CC"), 12, TRUE)
mat

# Recode the SNPs

recodeSNPs(mat)

# Recode the SNPs by assuming that the first letter in
# the heterogyzous genotype refers to the major allele.

recodeSNPs(mat, first.ref = TRUE)
```

```
## End(Not run)
```

---

rowCATTs   *Rowwise Cochran-Armitage Trend Test Based on Tables*

Description

Given two matrices, each representing one group of subjects (e.g., cases and controls in a case-control study), that summarize the numbers of subjects showing the different (ordered) levels of the ordinal variables represented by the rows of the matrices, the value of the Cochran-Armitage Trend Test statistic is computed for each variable.

Using this function instead of rowTrendStats is in particular recommended when the total number of observations is very large.

Usage

```
rowCATTs(cases, controls, scores = NULL, add.pval = TRUE)
```

Arguments

cases
: a numeric matrix in which each row represents one ordinal variable and each column one of the ordered levels that the variables exhibit. The entries of this matrix are the numbers of observations from one group (e.g., the cases in a case-control study) showing a particular level at the different variables. Such a matrix can, e.g., be generated by [rowTables](). The rowwise sums of cases are allowed to differ between variables (which might happen when some of the observations are missing for some of the variables).

controls
: a numeric matrix of the same dimensions as cases comprising the numbers of observations from the second group (e.g., the controls in a case-control study) that show the respective level at the different ordinal variables. The rows of controls must represent the same variables in the same order as cases, and the columns must represent the same levels in the same order. This matrix can also be generated by employing [rowTables](). The rowwise sums of controls are allowed to differ between variables (which might happen when some of the observations are missing for some of the variables).

scores
: a numeric vector of length ncol(cases) containing the scores for the different levels. If not specified, i.e.\ NULL, the column names of cases are interpreted as scores.

add.pval
: should p-values be added to the output? If FALSE, only the rowwise values of the Cochran-Armitage trend test statistic will be returned. If TRUE, additionally the (raw) p-values based on an approximation to the ChiSquare-distribution with 1 degree of freedom are returned.

Value

Either a vector containing the rowwise values of the Cochran-Armitage trend test statistic (if add.pval = FALSE), or a list containing these values (stats), and the (raw) p-values (rawp) not adjusted for multiple comparisons (if add.pval = TRUE).

Note

The usual contingency table for a variable can be obtained from the matrices by forming a variable-specific matrix in which each row consists of the row of one of these matrices.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Agresti, A.\ (2002). *Categorical Data Analysis*. Wiley, Hoboken, NJ. 2nd Edition.

Armitage, P.\ (1955). Tests for Linear Trends in Proportions and Frequencies. *Biometrics*, 11, 375-386.

Cochran, W.~G.\ (1954). Some Methods for Strengthening the Common ChiSquare Tests. *Biometrics*, 10, 417-451.

See Also

[rowTrendStats](#), [rowMsquares](#), [rowChisq2Class](#)

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

# Now assume that the first 25 columns correspond to
# cases and the remaining 25 columns to cases. Then
# a vector containing the class labels is given by

cl <- rep(1:2, e=25)

# and the matrices summarizing the numbers of subjects
# showing the respective levels at the different variables
# are computed by

cases <- rowTables(mat[, cl==1])
controls <- rowTables(mat[,cl==2])

# The values of the rowwise Cochran-Armitage trend test
# are computed by
```

```
rowCATTs(cases, controls)

# which leads to the same results as

rowTrendStats(mat, cl)

# or as

out <- rowMsquares(cases, controls)
n <- ncol(mat)
out$stats * n / (n-1)


## End(Not run)
```

---

rowChisq2Class   *Rowwise Pearson's ChiSquare Test Based on Tables*

Description

Given a set of matrices, each of which represents one group of subjects, and summarizes rowwise the numbers of these observations showing the levels of the categorical variables represented by the rows of the matrices, the value of Pearson's ChiSquare statistic for testing whether the distribution of the variable differs between the different groups is computed for each variable.

Using this function instead of rowChisqStats is recommended when the total number of observations is very large.

Usage

```
rowChisq2Class(cases, controls, add.pval = TRUE, sameNull = FALSE)

rowChisqMultiClass(..., listTables = NULL, add.pval = TRUE,
    sameNull = FALSE)
```

Arguments

cases   a numeric matrix in which each row represents one categorical variable and each column one of the levels that the variables exhibit. The entries of this matrix are the numbers of observations from one group (e.g., the cases in a case-control study) showing a particular level at the different variables. Such a matrix can, e.g., be generated by [rowTables](#).The rowwise sums of cases are allowed to differ between variables (which might happen when some of the observations are missing for some of the variables).

controls   a numeric matrix of the same dimensions as cases comprising the numbers of observations from the second group (e.g., the controls in a case-control study) that show the respective level at the different variables. The rows of controls must represent the same variables in the same order as cases, and the columns

must represent the same levels in the same order. This matrix can also be generated by employing [rowTables](). The rowwise sums of controls are allowed to differ between variables (which might happen when some of the observations are missing for some of the variables).

...   numeric matrices (such as cases and controls) each of which comprises the numbers of observations showing the respective levels at the different variables. The dimensions of all matrices must be the same, and the rows and columns must represent the same variables and levels, respectively, in the same order in all matrices.

listTables   instead of inputting the matrices directly into rowChisqMultiClass a list can generated, where each entry of this list is one of matrices, and this list can be used in rowChisqMultiClass by specifying listTables.

add.pval   should p-values be added to the output? If FALSE, only the rowwise values of Pearson's ChiSquare statistic will be returned. If TRUE, additionally the degrees of freedom and the (raw) p-values are computed by assuming approximation to the ChiSquare-distribution, and added to the output.

sameNull   should all variables follow the same null distribution? If TRUE, then all variables must show the same number of levels such that their null distribution is approximated by the same ChiSquare-distribution.

Value

Either a vector containing the rowwise values of Pearson's ChiSquare statistic (if add.pval = FALSE) or a list containing these values (stats), the degrees of freedom (df), and the p-values (rawp) not adjusted for multiple comparisons (if add.pval = TRUE).

Note

In the case of a 2 x 2 table, no continuity correction is applied. In this case, the results of rowChisq2Class and rowChisMultiClass are only equal to the results of chisq.test if in the latter correct = FALSE is used.

The usual contingency table for a variable can be obtained from the matrices by forming a variable-specific matrix in which each row consists of the row of one of these matrices.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[rowChisqStats](), [rowTables](), [rowCATTs](), [rowMsquares]()

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.
```

```
mat <- matrix(sample(3, 500, TRUE), 10)

# Now assume that the first 25 columns correspond to
# cases and the remaining 25 columns to cases. Then
# a vector containing the class labels is given by

cl <- rep(1:2, e=25)

# and the matrices summarizing the numbers of subjects
# showing the respective levels at the different variables
# are computed by

cases <- rowTables(mat[, cl==1])
controls <- rowTables(mat[,cl==2])

# To obtain the ChiSquare values call

rowChisq2Class(cases, controls)

# This leads to the same results as

rowChisqStats(mat, cl)

# or

rowChisqMultiClass(cases, controls)

# or

listTab <- list(cases, controls)
rowChisqMultiClass(listTables = listTab)


## End(Not run)
```

---

rowChisqStats   *Rowwise Pearson's ChiSquare Statistic*

Description

Computes for each row of a matrix the value of Pearson's ChiSquare statistic for testing if the corresponding categorical variable is associated with a (categorical) response, or determines for each pair of rows of a matrix the value of Pearson's ChiSquare statistic for testing if the two corresponding variables are independent.

Usage

```
rowChisqStats(data, cl, compPval = TRUE, asMatrix = TRUE)
```

Arguments

data   a numeric matrix consisting of the integers between 1 and n_{cat}, where n_{cat} is
                the maximum number of levels the categorical variables can take. Each row of
                data must correspond to a variable, each row to an observation. Missing values
                and different numbers of levels a variable might take are allowed.

cl   a numeric vector of length ncol(data) containing the class labels for the obser-
                vations represented by the columns of data. The class labels must be coded by
                the integers between 1 and n_{cl}, where n_{cl} is the number of classes. If missing,
                the value of the statistic for Pearson's χ^2-test of independence will be computed
                for each pair of rows of data. Otherwise, the value of Pearson's χ^2-statistic for
                testing if the distribution of the variable differs between the groups specified by
                cl will be determined for each row of data.

compPval   should also the p-value (based on the approximation to a χ^2-distribution) be
                computed?

asMatrix   should the pairwise test scores be returned as matrix? Ignored if cl is specified.
                If TRUE, a matrix with m rows and columns is returned that contains the values of
                Pearson's χ^2-statistic in its lower triangle, where m is the number of variables.
                If FALSE, a vector of length $m*(m-1)/2$ is returned, where the value for testing
                the ith and jth variable is given by the $j + m * (i-1) - i * (i-1)/2$ element
                of this vector.

Value

If compPval = FALSE, a vector (or matrix if cl is not specified and as.matrix = TRUE) composed
of the values of Pearson's χ^2-statistic. Otherwise, a list consisting of

stats   a vector (or matrix) containing the values of Pearson's χ^2-statistic.

df   a vector (or matrix) comprising the degrees of freedom of the asymptotic χ^2-
                distribution.

rawp   a vector (or matrix) containing the (unadjusted) p-values.

Note

Contrary to chisq.test, currently no continuity correction is done for 2 x 2 tables.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Schwender, H.\ (2007). A Note on the Simultaneous Computation of Thousands of Pearson's χ^2-
Statistics. *Technical Report*, SFB 475, Deparment of Statistics, University of Dortmund.

See Also

computeContCells, computeContClass

Examples

```
## Not run:
# Generate an example data set consisting of 5 rows (variables)
# and 200 columns (observations) by randomly drawing integers
# between 1 and 3.

mat <- matrix(sample(3, 1000, TRUE), 5)
rownames(mat) <- paste("SNP", 1:5, sep = "")

# For each pair of rows of mat, test if they are independent.

r1 <- rowChisqStats(mat)

# The values of Pearson's ChiSquare statistic as matrix.

r1$stats

# And the corresponding (unadjusted) p-values.

r1$rawp

# Obtain only the values of the test statistic as vector

rowChisqStats(mat, compPval = FALSE, asMatrix =FALSE)


# Generate an example data set consisting of 10 rows (variables)
# and 200 columns (observations) by randomly drawing integers
# between 1 and 3, and a vector of class labels of length 200
# indicating that the first 100 observation belong to class 1
# and the other 100 to class 2.

mat2 <- matrix(sample(3, 2000, TRUE), 10)
cl <- rep(1:2, e = 100)

# For each row of mat2, test if they are associated with cl.

r2 <- rowChisqStats(mat2, cl)
r2$stats

# And the results are identical to the one of chisq.test
pv <- stat <- numeric(10)
for(i in 1:10){
    tmp <- chisq.test(mat2[i,], cl)
    pv[i] <- tmp$p.value
    stat[i] <- tmp$stat
}

all.equal(r2$stats, stat)
all.equal(r2$rawp, pv)
```

```
## End(Not run)
```

---

rowCors   *Rowwise Correlation with a Vector*

Description

Computes Pearson's correlation coefficient of a vector with each row of a matrix.

Usage

```
rowCors(X, y, trendStat = FALSE, use.n = NULL)
```

Arguments

X	a numeric matrix in which each row represents a variable and each column an observation.
y	a numeric vector of length `ncol(X)`.
trendStat	instead of the correlation coefficients should the values of the statistic for a test of linear trend based on this coefficient be returned? If TRUE, then it is assumed that all variables in X and the variable represented by y are ordinal, and the values in X and y represent scores for the different levels.
use.n	should the squared values of the correlation coefficient be multiplied by `ncol(X)`? Ignored if `trendStat = FALSE`. If FALSE, the squared values are multiplied by `ncol(X) - 1`. By default, the squared values are multiplied by `ncol(X)` if y shows two levels, leading to the Cochran-Armitage test of trend. Otherwise, they are multiplied by `ncol(X) - 1`.

Value

A vector containing the rowwise values of Pearson's correlation coefficient (if `trendStat = FALSE` or the rowwise values of the trend statistics (if `trendStat = TRUE`.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Agresti, A.\ (2002). *Categorical Data Analysis*. Wiley, Hoboken, NJ. 2nd Edition.

See Also

[rowTrendStats](), [rowCATTs](), [rowMsquares]()

Examples

```
## Not run:
# Generate a random matrix containing 10 continuous variables
# and a vector representing a continuous variable.

mat <- matrix(runif(200, 0, 20), 10)
y <- sample(runif(20, 0, 20))

# The correlations between y and each of row of mat are
# computed by

rowCors(mat, y)

# Generate a random binary vector and a matrix consisting
# of 10 ordinal variables with levels 0, 1, 2, where these
# values can be interpreted as scores for the differ
# categories.

mat <- matrix(sample(0:2, 500, TRUE), 10)
y <- sample(0:1, 50, TRUE)

# The values of the Cochran-Armitage trend statistic are
# computed by

rowCors(mat, y, trendStat = TRUE)

# If the values of the general test of linear trend described
# on page 87 of Agresti (2002) should be computed, then call

rowCors(mat, y, trendStat = TRUE, use.n = FALSE)


## End(Not run)
```

---

rowFreqs   *Rowwise Frequencies*

Description

Computes the frequencies of the levels that the categorical variables in a matrix show.

Usage

```
rowFreqs(x, levels = 1:3, divide.by.n = FALSE, affy = FALSE,
    includeNA = FALSE, useNN = c("not", "only", "also"), check = TRUE)
```

Arguments

x	a matrix in which each row represents a categorical variable (e.g., a SNP) and each column an observation, where the variables are assumed to show the levels specified by levels. Missing values are allowed in x.
levels	vector specifying the levels that the categorical variables in x show. Ignored if affy = TRUE.
divide.by.n	should the numbers of observations showing the respective levels be divided by the total number of observations, i.e.\ by ncol(x)? If FALSE, these numbers are divided by the number of non-missing values of the respective variable. Ignored if includeNA = TRUE.
affy	logical specifying whether the SNPs in x are coded in the Affymetrix standard way. If TRUE, levels = c("AA", "AB", "BB") and useNN = "also" will be used (the latter only when includeNA = TRUE).
includeNA	should a column be added to the output matrix containing the number of missing values for each variable?
useNN	character specifying whether missing values can also be coded by "NN". If useNN = "not" (default), missing values are assumed to be coded only by NA. If "only", then missing values are assumed to be coded only by "NN" (and not by NA. If "both", both "NN" and NA are considered. Ignored if affy = TRUE.
check	should it be checked whether some of the variables show other levels than the one specified by levels?

Value

A matrix with the same number of rows as x containing for each variable the numbers of observations showing the levels specified by levels.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

rowTables

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

rowFreqs(mat)

# leads to the same results as

rowTables(mat) / ncol(mat)
```

```
# If mat contains missing values

mat[sample(500, 20)] <- NA

# then

rowFreqs(mat)

# leads to the same result as

rowTables(mat) / rowSums(!is.na(mat))


## End(Not run)
```

---

rowHWEs   *Rowwise Test for Hardy-Weinberg Equilibrium*

Description

Tests for each row of a matrix whether the Hardy-Weinberg Equilibrium holds for the SNP represented by the row.

Usage

```
rowHWEs(x, levels = 1:3, affy = FALSE, check = TRUE)
```

Arguments

x	a matrix in which each row represents a SNP and each column a subject, where the SNPs can take the values specified by levels. NAs are allowed.
levels	a vector of length three specifying the values with which the three genotypes of each SNP are represented. It is assumed that the second element of levels represents the heterozygous genotype, whereas the first and the third element represent the homozygous genotypes. Ignored if affy = TRUE.
affy	logical specifying whether the SNPs in x are coded as in the Affymetrix standard output. If TRUE, levels = c("AA", "AB", "BB") will be used.
check	should some checks be done if, e.g., other than the specified levels are used in x? It is highly recommended to leave check = TRUE. Setting check = FALSE will reduce the computation time slightly.

Value

A list containing the values of the ChiSquare statistic for testing for deviation from HWE (stats) and the raw p-values (rawp) computed by employing the ChiSquare distribution with 1 degree of freedom.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

---

rowMAFs   *Rowwise Minor Allele Frequency*

Description

Computes for each SNP represented by a row of a matrix the frequency of the minor allele.

Usage

```
rowMAFs(x, check = TRUE)
```

Arguments

x	a matrix in which each row represents a SNP and each column a subject, where the genotypes of each SNP are coded by 1 (for the homozygous reference genotype), 2 (heterozygous), and 3 (homozygous variant). NAs are also allowed.
check	should it be checked if the matrix contains values differing from 1, 2, and 3? It is highly recommended to leave check = TRUE. Setting check = FALSE reduces the computation time only slightly.

Value

a vector containing the minor allele frequency of the SNPs represented by x.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

---

rowMsquares   *Rowwise Linear Trend Test Based on Tables*

Description

Given a set of matrices, each representing one group of subjects (e.g., cases and controls in a case-control study), that summarize the numbers of subjects showing the different levels of the categorical variables represented by the rows of the matrices, the value of the linear trend statistic based on Pearson's correlation coefficient and described on page 87 of Agresti (2002) is computed for each variable.

Using this function instead of rowTrendStats is in particular recommended when the total number of observations is very large.

Usage

```
rowMsquares(..., listTables = NULL, clScores = NULL, levScores = NULL,
    add.pval = TRUE)
```

Arguments

`...`	numeric matrices in each of which each row corresponds to a ordinal variable and each column to one of the ordered levels of these variables. Each of these matrices represents one of the groups of interest and comprises the numbers of observations showing the respective levels at the different variables. These matrices can, e.g., generated by employing [rowTables](#). The dimensions of all matrices must be the same, and the rows and columns must represent the same variables and levels, respectively, in the same order in all matrices. The row-wise sums in a matrix are allowed to differ (which might happen if some of the observations are missing for some of the variables.)
`listTables`	instead of inputting the matrices directly, a list consisting of these matrices can be generated and then be used in `rowMsquares` by specifying `listTables`.
`clScores`	a numeric vector with one entry for each matrix specifying the score that should be assigned to the corresponding group. If `NULL`, `clScores` is set to `1:m`, where m is the number of groups/matrices, such that the first input matrix (or the first entry in `listTables`) gets a score of 1, the second a score of 2, and so on.
`levScores`	a numeric vector with one score for each level of the variables.If not specified, i.e.\ `NULL`, the column names of the matrices are interpreted as scores.
`add.pval`	should p-values be added to the output? If `FALSE`, only the rowwise values of the linear trend test statistic will be returned. If `TRUE`, additionally the (raw) p-values based on an approximation to the ChiSquare-distribution with 1 degree of freedom are returned.

Details

This is an extension of the Cochran-Armitage trend test from two to several classes. The statistic of the Cochran-Armitage trend test can be obtained by multiplying the statistic of this general linear trend test with $n/(n-1)$, where n is the number of observations.

Value

Either a vector containing the rowwise values of the linear trend test statistic (if `add.pval = FALSE`), or a list containing these values (`stats`), and the (raw) p-values (`rawp`) not adjusted for multiple comparisons (if `add.pval = TRUE`).

Note

The usual contingency table for a variable can be obtained from the matrices by forming a variable-specific matrix in which each row consists of the row of one of these matrices.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Agresti, A.\ (2002). *Categorical Data Analysis*. Wiley, Hoboken, NJ. 2nd Edition.

Mantel, N.\ (1963). Chi-Square Test with one Degree of Freedom: Extensions of the Mantel-Haenszel Procedure. *Journal of the American Statistical Association*, 58, 690-700.

See Also

rowTrendStats, rowCATTs, rowChisqMultiClass

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

# Now assume that we consider a case-control study,
# i.e. two groups, and that the first 25 columns
# of mat correspond to cases and the remaining 25
# columns to cases. Then a vector containing the
# class labels is given by

cl <- rep(1:2, e=25)

# and the matrices summarizing the numbers of subjects
# showing the respective levels at the different variables
# are computed by

cases <- rowTables(mat[, cl==1])
controls <- rowTables(mat[,cl==2])

# The values of the rowwise liner trend test are
# computed by

rowMsquares(cases, controls)

# which leads to the same results as

listTab <- list(cases, controls)
rowMsquares(listTables = listTab)

# or as

rowTrendStats(mat, cl, use.n = FALSE)

# or as

out <- rowCATTs(cases, controls)
n <- ncol(mat)
out$stats * (n - 1) / n
```

```
## End(Not run)
```

---

rowScales   *Rowwise Scaling*

Description

Scales each row of a matrix such that the values in this row have zero mean and a standard deviation of 1.

Usage

```
rowScales(X, add.stats = FALSE)
```

Arguments

X   a numeric matrix whose rows should be scaled. Missing values are allowed.

add.stats   should the rowwise means and standard deviations of X be returned?

Value

If add.stats = FALSE, a matrix of the same dimensions as X with a rowwise mean of zero and a rowwise standard deviation of 1. If add.stats = TRUE, a list containing this matrix and the rowwise means and standard deviations of the input matrix.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[rowCors](#)

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

rowScales(mat)


## End(Not run)
```

---

rowTables   *Rowwise Tables*

Description

Computes a one-dimensional table for each row of a matrix that summarizes the values of the categorical variables represented by the rows of the matrix.

Usage

```
rowTables(x, levels = 1:3, affy = FALSE, includeNA = FALSE,
   useNN = c("not", "only", "also"), check = TRUE)
```

Arguments

x	a matrix in which each row represents a categorical variable (e.g., a SNP) and each column an observation, where the variables are assumed to show the levels specified by levels. Missing values are allowed in x.
levels	vector specifying the levels that the categorical variables in x show. Ignored if affy = TRUE.
affy	logical specifying whether the SNPs in x are coded in the Affymetrix standard way. If TRUE, levels = c("AA", "AB", "BB") and useNN = "also" will be used (the latter only when includeNA = TRUE).
includeNA	should a column be added to the output matrix containing the number of missing values for each variable?
useNN	character specifying whether missing values can also be coded by "NN". If useNN = "not" (default), missing values are assumed to be coded only by NA. If "only", then missing values are assumed to be coded only by "NN" (and not by NA. If "both", both "NN" and NA are considered. Ignored if affy = TRUE.
check	should it be checked whether some of the variables show other levels than the one specified by levels?

Value

A matrix with the same number of rows as x containing for each variable the numbers of observations showing the levels specified by levels.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[rowFreqs](), [rowScales]()

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

rowTables(mat)


## End(Not run)
```

---

rowTrendFuzzy   *Trend Test for Fuzzy Genotype Calls*

Description

rowTrendFuzzy performs the trend test proposed by Louis et al. (2010) based on fuzzy genotype calls, i.e. the weighted sums over the confidences for the three genotypes as they are determined by preprocessing algorithms (e.g., CRLMM) or imputation procedures.

Given the confidences and scores for all three genotypes, getMatFuzzy constructs a matrix containing the fuzzy genotype calls.

Usage

```
rowTrendFuzzy(score, probs, y, mat.fuzzy = NULL,
    alternative = c("two.sided", "less", "greater"),
    check = TRUE)

getMatFuzzy(score, probs, check = TRUE)
```

Arguments

score   either a numeric vector of length 2 or 3, or a character string.

                     If the latter, score must be either "additive", "dominant", "recessive", or an abbreviation of these terms. If score = "additive", then score is set to c(0, 1, 2). If score = "dominant", then score is set to c(0, 1, 1). And if score = "recessive", score is set to c(0, 0, 1).

                     If score is a numeric vector of length 2, then the first value must be the score for the heterozygous genotype, and the second value the score for the homozygous variant genotype. If score is of length, the first entry of this vector must be a zero (the score for the homozygous reference genotype), followed by the scores for the heterozygous and homozygous variant genotype.

probs   a list of length 2 or 3 consisting of matrices of the same size. Each matrix must contain the confidences for one of the three genotypes, where each row in the matrix represents a SNP and each column a sample (which must be in

<div style="margin-left: 2em;">

the same order in all matrices). The matrices in `probs` correspond to the scores in `score`. Thus, if `probs` has length 2, then the first matrix must contain the confidences for the heterozygous genotype, and the second matrix the confidences for the homozygous variant genotype. All elementwise sums of these two matrices must be smaller than or equal to 1. If `probs` has length 3, the first object must be a matrix containing the confidences for the homozygous reference genotype, followed by the two other matrices comprising the confidences for the heterozygous and homozygous variant genotype. All elementwise sums of the three matrices must be equal to 1.

</div>

y	a vector of zeros and ones specifying which of the samples in the matrices in `probs` are cases (1) and which are controls (0).
mat.fuzzy	a matrix containing the fuzzy genotype calls. If specified, `score` and `probs` are not allowed to be specified in `rowTrendFuzzy`. If NULL, `mat.fuzzy` is determined by employing `getMatFuzzy`, i.e.\ by multiplying the confidences in `probs` with the corresponding scores in `score` and computing the elementwise sums over the resulting matrices with the weighted confidences.
alternative	a character string specifying the alternative hypothesis. Must be one of `"two.sided"` (default), `"greater"`, or `"less"`. Abbreviations (e.g.\ the initial letter) for these choices are also allowed.
check	logical specifying whether the specified objects should be extensively checked. If FALSE, only some basic checks are done. It is highly recommended to use `check = TRUE`, although the checking takes much more computing time than the determination of the values of the trend test statistic.

Value

For `getMatFuzzy`, a matrix containing the fuzzy genotype calls. For `rowTrendFuzzy`, a list consisting of

stat	a vector containing the values of the trend test statistic for all SNPs comprised by `probs`,
rawp	a vector containing the unadjusted p-values computed for the values in `stat` based on a standard normal distribution and the specification of `alternative`.
theta	a vector containing estimates for the log odds ratios for risk corresponding to `stat`,
varTheta	a vector containing the variance estimates for `theta`.

References

Louis, T.A., Carvalho, B.S., Fallin, M.D., Irizarry, R.A., Li, Q., and Ruczinski, I. (2010). Association Tests that Accommodate Genotyping Errors. In Bernardo, J.M., Bayarri, M.J., Berger, J.O., Dawid, A.P., Heckerman, D., Smith, A.F.M., and West, M. (eds.), *Bayesian Statistics 9*, 393-420. Oxford University Press, Oxford, UK. With Discussion.

See Also

[rowTrendStats](), [rowCATTs]()

---

rowTrendStats   *Rowwise Linear Trend Tests*

Description

Computes for each row of a matrix the value of the statistic of a linear trend test for testing whether the ordinal variable corresponding to the row of the matrix is associated with an ordinal response.

In the two-class case, the statistic of the Cochran-Armitage trend test is computed by default.

Usage

```
rowTrendStats(X, y, use.n = NULL, add.pval = TRUE)
```

Arguments

X	a numeric matrix in which each row represents an ordinal variable and each column corresponds to an observation. The entries of this matrix are interpreted as scores for the different (ordered) levels of the respective variables.
y	a numeric vector of length `ncol(X)` containing the class labels of the observations represented by the columns of X, where these labels are interpreted as scores for the different classes.
use.n	should the squared values of Pearson's correlation coefficient be multiplied by `ncol(X)` to generate the values of the test statistic? If `FALSE`, the squared values are multiplied by `ncol(X) - 1`. By default, the squared values are multiplied by `ncol(X)` if y shows two levels, leading to the Cochran-Armitage test of trend. Otherwise, they are multiplied by `ncol(X) - 1` leading to the linear trend test statistic of Mantel (1963) described, e.g., on page 87 of Agresti (2002).
add.pval	should p-values be added to the output? If `FALSE`, only the rowwise values of the linear trend test statistic will be returned. If `TRUE`, additionally the (raw) p-values based on an approximation to the ChiSquare-distribution with 1 degree of freedom are returned.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Agresti, A.\ (2002). *Categorical Data Analysis*. Wiley, Hoboken, NJ. 2nd Edition.

Armitage, P.\ (1955). Tests for Linear Trends in Proportions and Frequencies. *Biometrics*, 11, 375-386.

Cochran, W.~G.\ (1954). Some Methods for Strengthening the Common ChiSquare Tests. *Biometrics*, 10, 417-451.

Mantel, N.\ (1963). Chi-Square Test with one Degree of Freedom: Extensions of the Mantel-Haenszel Procedure. *Journal of the American Statistical Association*, 58, 690-700.

See Also

rowMsquares, rowCATTs, rowChisqMultiClass

Examples

```
## Not run:
# Generate a matrix containing data for 10 categorical
# variables with levels 1, 2, 3.

mat <- matrix(sample(3, 500, TRUE), 10)

# Now assume that the first 25 columns correspond to
# cases and the remaining 25 columns to cases. Then
# a vector containing the class labels is given by

cl <- rep(0:1, e=25)

# The values of the Cochran-Armitage trend test can
# then be computed by

rowTrendStats(mat, cl)

# This leads to the same results as

cases <- rowTables(mat[, cl==1])
controls <- rowTables(mat[,cl==0])

rowCATTs(cases, controls)

# or as

out <- rowMsquares(cases, controls)
n <- ncol(mat)
out$stats * n / (n - 1)


## End(Not run)
```

---

shortenGeneDescription

Shorten the Gene Description

Description

Shortens the entries of the column of a data frame containing the genes associated with the SNPs for which the data frame comprises annotations. Typically used in combination with, i.e.\ either within or after an application of, buildSNPannotation.

Usage

```
    shortenGeneDescription(dat, colname = "Gene", max.length = 2,
        sep = "///", add.ldots = TRUE)
```

Arguments

dat	a data frame. Typically, the output of [buildSNPannotation](#).
colname	character string comprising the name of the column of dat containing the gene description.
max.length	integer specifying the maximum number of genes associated with the respective SNP that should be stored in the data frame. By default, the first two genes are retained. Shortened entries are marked by ... at the end of the entries, when add.ldots = TRUE.
sep	character string specifying the separation symbol between the different genes.
add.ldots	should ... be added at the entries which are shortened?

Value

The same data frame as dat with shortened entries in the column of dat named colname.

Author(s)

Holger Schwender, <holger.schw@gmx.de>

See Also

[buildSNPannotation](#)

---

showChanges	*Displaying Changes*

Description

Shows the content of the file CHANGES stored in a package.

Usage

```
    showChanges(pkg = "scrime")
```

Arguments

pkg	character string naming the package.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

---

simulateSNPcatResponse

Simulation of SNP Data with Categorical Response

Description

Simulates SNP data. Interactions of some of the simulated SNPs are then used to specify a categorical response by level-wise or multinomial logistic regression.

Usage

```
simulateSNPcatResponse(n.obs = 1000, n.snp = 50, list.ia = NULL,
    list.snp = NULL, withRef = FALSE, beta0 = -0.5, beta = 1.5,
    maf = 0.25, sample.y = TRUE, rand = NA)

## S3 method for class 'simSNPcatResponse'
print(x, justify = c("left", "right"), spaces = 2, ...)
```

Arguments

n.obs   number of observations that should be generated.

n.snp   number of SNPs that should be generated.

list.ia   a list consisting of n_{cat} objects, where n_{cat} is the number of levels the response
                 should have. If one interaction of SNPs should be explanatory for a specific level
                 of the response, then the corresponding object in list.ia must be a numeric
                 vector specifying the genotypes of the interacting SNPs by the integers -3, -
                 2, -1, 1, 2, or 3, where 1 codes for the homozygous reference genotype, 2 for
                 the heterozygous genotype, and 3 for the homozygous variant genotype, and a
                 minus before these numbers means that the corresponding SNP should be not of
                 this genotype. If more than one interaction should be explanatory for a specific
                 category, then the corresponding object of list.ia must be a list containing one
                 numeric vector composed of the integers -3, -2, -1, 1, 2, and 3 for each of the
                 interactions.

                 If, e.g., one of the vectors is given by c(1, -1, -3) and the corresponding vec-
                 tor in list.snp is c(5, 7, 8), then the corresponding interaction explanatory
                 for a level of the response is given by

                 (SNP5 == 1) & (SNP7 != 1) & (SNP8 != 3).

                 For more details, see Details. Must be specified if list.snp is specified. If both
                 list.ia and list.snp are NULL, then the interactions shown in the Details
                 section are used.

list.snp   a list consisting of numeric vectors (if one interaction should be explanatory
                 for a level of the response) or lists of numeric vectors (if there should be more
                 than one explanatory interaction) specifying the SNPs that compose the inter-
                 actions. list.snp must have the same structure as list.ia, and each entry of
                 list.snp must be an integer between 1 and n.snp. If list.ia is specified but

	not `list.snp`, then the first n SNPs are used to generate the interactions, where n is the total number of values in `list.ia`. For the case that both `list.ia` and `list.snp` are not specified, see Details.
withRef	should there be an additional reference group (i.e.\ a control group) denoted by a zero? If `TRUE`, a multinomial logistic regression is used to specify the class labels. If `FALSE`, level-wise logistic regressions are employed to generate the class labels. For details, see Details.
beta0	a numeric value or vector of `length(list.ia)` specifying the intercept of the logistic regression models.
beta	either a non-negative numeric value or a list of non-negative numeric values specifying the parameters in the logistic regression models. If a numeric value, all parameters (except for the intercept) in all logistic regression models will be equal to this value. If a list, then this list must have the same length as `list.ia`, and each object must consist of as many numeric values as interactions are specified by the corresponding object in `list.ia`.
maf	either an integer, or a vector of length 2 or `n.snp` specifying the minor allele frequency. If an integer, all the SNPs will have the same minor allele frequency. If a vector of length `n.snp`, each SNP will have the minor allele frequency specified in the corresponding entry of `maf`. If length 2, then `maf` is interpreted as the range of the minor allele frequencies, and for each SNP, a minor allele frequency will be randomly drawn from a uniform distribution with the range given by `maf`.
sample.y	should the values of the response be randomly drawn using the probabilities determined by the logistic regression models? If `FALSE`, then for each of the `n.obs` observations, the value of the response is given by the level exhibiting the largest probability at this observation.
rand	a numeric value for setting the random number generator in a reproducible state.
x	the output of `simulateSNPcatResponse`
justify	a character string specifying whether the column of the summarizing table that names the explanatory interactions should be `"left"`- or `"right"`-adjusted.
spaces	integer specifying the distance from the left end of the column mentioned in `justify` to the position at which the column name is presented.
...	ignored.

Details

`simulateSNPcatResponse` first simulates a matrix consisting of `n.obs` observations and `n.snp` SNPs, where the minor allele frequencies of these SNPs are given by `maf`.

Note that all SNPs are currently simulated independently of each other such that they are unlinked. Moreover, an observation is currently not allowed to have genotypes/interactions that are explanatory for more than one of the levels of the response. If, e.g., the response has three categories, then an observation can either exhibit one (or more) of the genotypes explaining the first level, or one (or more) of the genotypes explanatory for the second level, or one (or more) of the genotypes explaining the third level, or none of these genotypes.

Afterwards, the response is generated by employing the specifications of `list.ia`, `list.snp`, `beta0` and `beta`.

By default, i.e.\ if both `list.ia` and `list.snp` are NULL, `list.ia` is set to

`list(c(-1, 1), c(1, 1, 1), list(c(-1, 1), c(1, 1, 1)))`,

and `list.snp` is set to

`list(c(6, 7), c(3, 9, 10), list(c(2, 5), c(1, 4, 8)))`

such that the interaction

`(SNP6 != 1) & (SNP7 == 1)`

is assumed to be explanatory for the first level of the three-categorical response, the interaction

`(SNP3 == 1) & (SNP9 == 1) & (SNP10 == 1)`

is assumed to be explanatory for the second level, and the interactions

`(SNP2 != 1) & (SNP5 == 1)\\\` and

`(SNP1 == 1) & (SNP4 == 1) & (SNP8 == 1)`,

are assumed to be explanatory for the third level.

If `withRef = FALSE`, then for each of the levels, the probability of having this level given that an observation exhibits one, two, ... of the interactions intended to be explanatory for that level is determined using the corresponding logistic regression model. Afterwards, the value of the response for an observation showing one, two, ... of the interactions explanatory for a particular level is randomly drawn using the above probability p for the particular level and $(1 - p)/(n_{cat} - 1)$ as probabilities for the other $(n_{cat} - 1)$ levels. If an observation exhibits none of the explanatory interactions, its response value is randomly drawn using the probabilities $\exp beta0/(1 + \exp beta0)$.

If `withRef = TRUE`, a multinomial logistic regression is used to specify the class labels. In this case the probabilities p_j, $j = 1, ..., n.cat$, are given by $p_j = \exp(q_j) * p_0$, where q_j are the probabilities on the logit-scale (i.e.\ the probabilities on the scale of the linear predictors) and $p_0^{-1} = 1 + p_1 + ... + p_{n.cat}$ is the reciprocal of the probability for the control/reference group.

Value

An object of class `simSNPcatResponse` consisting of

x	a matrix with `n.obs` rows and `n.snp` columns containing the simulated SNP values.
y	a vector of length `n.obs` composed of the values of the response.
models	a character vector naming the level-wise logistic regression models.
maf	a vector of length `n.snp` composed of the minor allele frequencies.
tab.explain	a data frame summarizing the results of the simulation.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[simulateSNPs](#), [simulateSNPglm](#)

Examples

```
## Not run:
# The simulated data set described in Details.

sim1 <- simulateSNPcatResponse()
sim1

# Specifying the values of the response by the levels with
# the largest probability.

sim2 <- simulateSNPcatResponse(sample.y = FALSE)
sim2

# If ((SNP4 != 2) & (SNP3 == 1)), (SNP5 ==3), and
# ((SNP12 !=1) & (SNP9 == 3)) should be the three interactions
# (or variables) that are explanatory for the three levels
# of the response, list.ia and list.snp are specified as follows.

list.ia <- list(c(-2, 1), 3, c(-1,3))
list.snp <- list(c(4, 3), 5, c(12,9))

# The categorical response and a data set consisting of
# 800 observations and 25 SNPs, where the minor allele
# frequency of each SNP is randomly drawn from a
# uniform distribution with minimum 0.1 and maximum 0.4,
# is then generated by

sim3 <- simulateSNPcatResponse(n.obs = 800, n.snp = 25,
  list.ia = list.ia, list.snp = list.snp, maf = c(0.1, 0.4))
sim3


## End(Not run)
```

---

simulateSNPglm   *Simulation of SNP data*

Description

Simulates SNP data. Interactions of some of the simulated SNPs are then used to specify either a binary or a quantitative response by a logistic or linear regression model, respectively.

Usage

```
simulateSNPglm(n.obs = 1000, n.snp = 50, list.ia = NULL, list.snp = NULL,
   beta0 = -0.5, beta = 1.5, maf = 0.25, sample.y = TRUE, p.cutoff = 0.5,
   err.fun = NULL, rand = NA, ...)
```

Arguments

`n.obs`	number of observations that should be generated.
`n.snp`	number of SNPs that should be generated.
`list.ia`	a list consisting of numeric vectors (or values) specifying the genotypes of the SNPs that should be explanatory for the response. Each of these vectors must be composed of some of the numbers -3, -2, -1, 1, 2, 3, where 1 denotes the homozygous reference genotype, 2 the heterozygous genotype, and 3 the homozygous variant genotype, and a minus before these numbers means that the corresponding SNP should be not of this genotype. If, e.g., one of the vectors is given by `c(1, -1, -3)` and the corresponding vector in `list.snp` is `c(5, 7, 8)`, then the corresponding interaction used in the regression model to specify the response is
	`(SNP5 == 1) & (SNP7 != 1) & (SNP8 != 3)`.
	For more details, see Details. Must be specified if `list.snp` is specified. If both `list.ia` and `list.snp` are NULL, then the interactions shown in the Details section are used.
`list.snp`	a list consisting of numeric vectors specifying the SNPs that compose the interactions used in the regression model. Each of these vectors must have the same length as the corresponding vector in `list.ia`, and must consist of integers between 1 and `n.snp`, where the integer i corresponds to the ith column of the simulated SNP matrix. If `list.ia` is specified but not `list.snp`, then the first n SNPs are used to generate the interactions, where n is the total number of values in `list.ia`. For the case that both `list.ia` and `list.snp` are not specified, see Details.
`beta0`	a numeric value specifying the intercept of the regression model.
`beta`	a non-negative numeric value or vector of the same length as `list.ia` (i.e.\ one numeric value for each interaction) specifying the parameters in the regression model.
`maf`	either an integer, or a vector of length 2 or `n.snp` specifying the minor allele frequency. If an integer, all the SNPs will have the same minor allele frequency. If a vector of length `n.snp`, each SNP will have the minor allele frequency specified in the corresponding entry of `maf`. If length 2, then `maf` is interpreted as the range of the minor allele frequencies, and for each SNP, a minor allele frequency will be randomly drawn from a uniform distribution with the range given by `maf`.
`sample.y`	should the values of the response in the logistic regression model be randomly drawn using the probabilities of the respective observations for being a case? If FALSE, then the response value of an observation is 1 if its probability for being a case is larger than `p.cutoff`, and otherwise the observation is classified as a control (i.e.\ the value of the response is 0). Ignored if `err.fun` is specified.
`p.cutoff`	a probability, i.e.\ a numeric value between 0 and 1, naming the cutoff for an observation to be called a case if `sample.y = FALSE`. For details, see `sample.y`. Ignored if `sample.y = TRUE` or `err.fun` is specified.
`err.fun`	a function for generating the error that is added to the linear model to determine the value of the (quantitative) response. If NULL, a logistic regression model is

fitted. If specified, a linear model is fitted. Therefore, this argument is used to differ between the two types of models. The specified function must have as first argument the number of values that should be generated and as output a vector consisting of these values. Further arguments can also be specified because of `...` in simulateSNPglm. If, e.g., err.fun = rnorm, then rnorm(n.obs) will be called to generate n.obs observations from a standard normal function.

rand   a numeric value for setting the random number generator in a reproducible state.

...   further arguments of the function specified by err.fun.

Details

simulateSNPglm first simulates a matrix consisting of n.obs observations and n.snp SNPs, where the minor allele frequencies of these SNPs are given by maf.

Note that all SNPs are currently simulated independently of each other such that they are unlinked.

Afterwards, the response is determined by a regression model using the specifications of list.ia, list.snp, beta0 and beta. Depending on whether err.fun is specified or not, a linear or a logistic regression model is used, respectively, i.e.\ the response Y is continuous or binary.

By default, a logistic regression model

$\text{logit}(\text{Prob}(Y = 1)) = $ beta0 + beta[1] * L_1 + beta[2] * L_2 + ...

is fitted, since err.fun = NULL.

If both list.ia and list.snp are NULL, then interactions similar to the one considered, e.g., in Nunkesser et al.\ (2007) or Schwender et al.\ (2007) are used, i.e.\

$L_1 = $ (SNP6 != 1) & (SNP7 == 1)\\\ and

$L_2 = $ (SNP3 == 1) & (SNP9 == 1) & (SNP10 == 1),

by setting list.ia = list(c(-1, 1), c(1, 1, 1)) and list.snp = list(c(6, 7), c(3, 9, 10)).

Using the above model $\text{Prob}(Y = 1)$ is computed for each observation, and its value of the response is determined either by a random draw from a Bernoulli distribution using this probability (if sample.y = TRUE), or by evaluating if $\text{Prob}(Y = 1) > $ p.cutoff (if sample.y = FALSE).

If err.fun is specified, then the linear model

$Y = $ beta0 + beta[1] * L_1 + beta[2] * L_2 + ... + error

is used to determine the values of the response Y, where the values for error are given by the output of a call of err.fun.

Value

An object of class simSNPglm consisting of

x   a matrix with n.obs rows and n.snp columns containing the simulated SNP values.

y   a vector of length n.obs composed of the values of the response.

beta0   the value of the intercept.

beta   the vector of parameters.

ia   a character vector naming the explanatory interactions.

maf	a vector of length n.snp composed of the minor allele frequencies.
prob	a vector of length n.obs consisting of the values of $\text{Prob}(Y = 1)$ (if err.fun = NULL).
err	a vector of length n.obs composed of the values of the error (if err.fun is specified).
p.cutoff	the value of p.cutoff (if err.fun = NULL and sample.y = FALSE).
err.call	a character string naming the call of the error function (if err.fun is specified).

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

References

Nunkesser, R., Bernholt, T., Schwender, H., Ickstadt, K. and Wegener, I.\ (2007). Detecting High-Order Interactions of Single Nucleotide Polymorphisms Using Genetic Programming. *Bioinformatics*, 23, 3280-3288.

Schwender, H.\ (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

See Also

simulateSNPs, summary.simSNPglm, simulateSNPcatResponse

Examples

```
## Not run:
# The simulated data set described in Details.

sim1 <- simulateSNPglm()
sim1

# A bit more information: Table of probabilities of being a case
# vs. numbers of cases and controls.

summary(sim1)

# Calling an observation a case if its probability of being
# a case is larger than 0.5 (the default for p.cutoff).

sim2 <- simulateSNPglm(sample.y = FALSE)
summary(sim2)

# If ((SNP4 != 2) & (SNP3 == 1)), (SNP5 ==3) and
# ((SNP12 !=1) & (SNP9 == 3)) should be the three interactions
# (or variables) that are explanatory for the response,
# list.ia and list.snp are specified as follows.

list.ia <- list(c(-2, 1), 3, c(-1,3))
list.snp <- list(c(4, 3), 5, c(12,9))
```

```
# The binary response and the data set consisting of
# 600 observations and 25 SNPs, where the minor allele
# frequency of each SNP is randomly drawn from a
# uniform distribution with minimum 0.1 and maximum 0.4,
# is then generated by

sim3 <- simulateSNPglm(n.obs = 600, n.snp = 25,
  list.ia = list.ia, list.snp = list.snp, maf = c(0.1, 0.4))
sim3

summary(sim3)

# If the response should be quantitative, err.fun has
# to be specified. To use a normal distribution with mean 0
# (default in rnorm) and a standard deviation of 2
# as the distribution of the error, call

simulateSNPglm(err.fun = rnorm, sd = 2)



## End(Not run)
```

---

simulateSNPs   *Simulation of SNP data*

Description

Simulates SNP data, where a specified proportion of cases and controls is explained by specified set of SNP interactions. Can also be used to simulate a data set with a multi-categorical response, i.e.\ a data set in which the cases are divided into several classes (e.g., different diseases or subtypes of a disease).

Usage

```
simulateSNPs(n.obs, n.snp, vec.ia, prop.explain = 1,
  list.ia.val = NULL, vec.ia.num = NULL, vec.cat = NULL,
  maf = c(0.1, 0.4), prob.val = rep(1/3, 3), list.equal = NULL,
  prob.equal = 0.8, rm.redundancy = TRUE, shuffle = FALSE,
  shuffle.obs = FALSE, rand = NA)
```

Arguments

n.obs   either an integer specifying the total number of observations, or a vector of length 2 specifying the number of cases and the number of controls. If vec.cat is specified, then the partitioning of the number of cases to the different classes can be governed by vec.ia.num. If n.obs is an integer, then $1/c$ of the observations will be controls and the remaining observations will be cases, where c is the total number of groups (including the controls).

n.snp	integer specifying the number of SNPs.
vec.ia	a vector of integers specifying the orders of the interactions that explain the cases. c(3,1,2,3), e.g., means that a three-way, a one-way (i.e. just a SNP), a two-way, and a three-way interaction explain the cases.
prop.explain	either an integer or a vector of length(vec.ia) specifying the proportions of cases explained by the interactions of interest among all observation having the interaction of interest. Must be larger than 0.5. E.g., prop.explain = 1 means that only cases have the interactions of interest specified by vec.ia (and list.ia.val). E.g., vec.ia = c(3, 2) and prop.explain = c(1, 0.8) means that only cases have the three-way interaction of interest, while 80% of the observations having the two-way interaction of interest are cases, and 20% are controls.
list.ia.val	a list of length(vec.ia) specifying the exact interactions. The objects in this list must be vectors of length vec.ia[i], and consist of the values 0 (for homozygous reference), 1 (heterozygous variant), or 2 (homozygous variant). E.g., vec.ia = c(3, 2) and list.ia.val = list(c(2, 0, 1), c(0, 2)) and prob.equal = 1 (see also list.equal) means that ((SNP1 == 2) \& (SNP2 == 0) \& (SNP3 == 1)) and ((SNP4 == 0) \& (SNP5 == 2)) are the explanatory interactions (if additionally prob.equal = 1; see also list.equal). If NULL, the genotypes are randomly drawn using the probabilities given by prob.val.
vec.ia.num	a vector of length(vec.ia) specifying the number of *cases* (not observations) explained by the interactions in vec.ia. If NULL, all the cases are divided into length(vec.ia) groups of about the same size. sum(vec.ia.num) must be smaller than or equal to the total number of cases. Each entry of vec.ia.num must currently be >= 10.
vec.cat	a vector of the same length of vec.ia specifying the subclasses of the cases that are explained by the corresponding interaction in vec.ia. If NULL, no subclasses will be considered. This feature is currently not fully tested. So be careful if specifying vec.cat.
maf	either an integer, or a vector of length 2 or n.snp specifying the minor allele frequencies. If an integer, all SNPs will have the same minor allele frequency. If a vector of length n.snp, each SNP will have the minor allele frequency specified in the corresponding entry of maf. If length 2, then maf is interpreted as the range of the minor allele frequencies, and for each SNP, a minor allele frequency will be randomly drawn from a uniform distribution with the range given by maf. Note: If a SNP belongs to an explanatory interaction, then only the set of observations not explained by this interaction will have the minor allele frequency specified by maf.
prob.val	a vector consisting of the probabilities for drawing a 0, 1, or 2, if list.ia.val = NULL, i.e.\ if the genotypes of the SNPs explaining the case-control status should be randomly drawn. Ignored if list.ia.val is specified. By default, each genotype has the same probability of being drawn.
list.equal	list of same structure as list.ia.val containing only ones and zeros, where a 1 specifies the equality to the corresponding value in list.ia.val, and a 0 specifies the non-equality. Thus, the entries of list.equal specify if the corresponding SNP should be of a particular genotype (when the entry is 1)

or should be not of this genotype (when entry is 0). If NULL, this list will be generated automatically using prob.equal. If, e.g., vec.ia = c(3, 2), list.ia.val = list(c(2, 0, 1), c(0, 2)), and list.equal = list(c(1, -1, 1), c(1, -1)), then the explanatory interactions are given by ((SNP1 == 2) \& (SNP2 != 0) \& (SNP3 == 1)) and ((SNP4 == 0) \& (SNP5 != 2))

prob.equal   a numeric value specifying the probability that a 1 is drawn when generating list.equal. prob.equal is thus the probability for an equal sign.

rm.redundancy   should redundant SNPs be removed from the explaining interactions? It is possible that one specify an explaining i-way interaction, but an interaction between $(i - 1)$ of the variables contained in the i-way interaction already explains all the cases (and controls) that the i-way interaction should explain. In this case, the redundant SNP is removed if rm.redundancy = TRUE.

shuffle   logical. By default, the first sum(vec.ia) columns of the generated data set contain the explanatory SNPs in the same order as they appear in this data set. If TRUE, this order will be shuffled.

shuffle.obs   should the observations be shuffled?

rand   integer. Sets the random number generator in a reproducible state.

Value

An object of class simulatedSNPs composed of

data   a matrix with n.obs rows and n.snp columns containing the SNP data.

cl   a vector of length n.obs comprising the case-control status of the observations.

tab.explain   a table naming the explanatory interactions and the numbers of cases and controls explained by them.

ia   character vector naming the interactions.

maf   vector of length n.snp containing the minor allele frequencies.

Note

Currently, the genotypes of all SNPs are simulated independently from each other (except for the SNPs that belong to the same explanatory interaction).

Author(s)

Holger Schwender <holger.schwender@udo.edu>

See Also

[simulateSNPglm](), [simulateSNPcatResponse]()

Examples

```
## Not run:
# Simulate a data set containing 2000 observations (1000 cases
# and 1000 controls) and 50 SNPs, where one three-way and two
# two-way interactions are chosen randomly to be explanatory
# for the case-control status.

sim1 <- simulateSNPs(2000, 50, c(3, 2, 2))
sim1

# Simulate data of 1200 cases and 800 controls for 50 SNPs,
# where 90% of the observations showing a randomly chosen
# three-way interaction are cases, and 95% of the observations
# showing a randomly chosen two-way interactions are cases.

sim2 <- simulateSNPs(c(1200, 800), 50, c(3, 2),
   prop.explain = c(0.9, 0.95))
sim2

# Simulate a data set consisting of 1000 observations and 50 SNPs,
# where the minor allele frequency of each SNP is 0.25, and
# the interactions
# ((SNP1 == 2) & (SNP2 != 0) & (SNP3 == 1))   and
# ((SNP4 == 0) & (SNP5 != 2))
# are explanatory for 200 and 250 of the 500 cases, respectively,
# and for none of the 500 controls.

list1 <- list(c(2, 0, 1), c(0, 2))
list2 <- list(c(1, 0, 1), c(1, 0))
sim3 <- simulateSNPs(1000, 50, c(3, 2), list.ia.val = list1,
    list.equal = list2, vec.ia.num = c(200, 250), maf = 0.25)


## End(Not run)
```

---

smc   *Simple Matching Coefficient and Cohen's Kappa*

Description

Computes the values of (or the distance based on) the simple matching coefficient or Cohen's Kappa, respectively, for each pair of rows of a matrix.

Usage

```
smc(x, dist = FALSE)
cohen(x, dist = FALSE)
```

Arguments

x	a matrix consisting of integers between 1 and n_{cat}, where n_{cat} is the number of levels the variables in x can take. Missing values are allowed.
dist	should the distance based on the simple matching coefficient or Cohen's Kappa, respectively, be computed? Note that, e.g., smc(x, dist = TRUE) is equal to 1 - smc(x, dist = FALSE).

Value

A matrix with nrow(x) columns and rows containing the distances or similarities.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[pcc](#)

Examples

```
## Not run:
# Generate a data set consisting of 10 rows and 200 columns,
# where the values are randomly drawn from the integers 1, 2, and 3.

mat <- matrix(sample(3, 2000, TRUE), 10)

# For each pair of row, the value of the simple matching coefficient
# can be obtained by

smc(mat)

# and the distance based on the SMC by

smc(mat, dist = TRUE)


## End(Not run)
```

---

snp2bin   *Transformation of SNPs to Binary Variables*

Description

Transforms SNPs to binary variables.

Usage

```
snp2bin(mat, domrec = TRUE, refAA = FALSE, snp.in.col = TRUE,
    monomorph = 0)
```

Arguments

mat   a matrix or data frame in which the genotypes of all SNPs are coded either by
                    0, 1 and 2, or by 1, 2 and 3, or by "AA", "AB" and "BB". Missing values are
                    allowed. In the latter coding not only NA, but also "NN" is allowed for specifying
                    missing values. Using the former two codings it is assumed that the smallest
                    value codes the homozygous reference genotype, the second value the heterozy-
                    gous genotype, and the largest value the homozygous variant genotype. For the
                    third coding, see refAA.

domrec   should each SNP be coded by two dummy variables from which one codes for
                    a recessive, and the other for a dominant effect? If TRUE, then the first binary
                    variable is set to 1 if the SNP is of the heterozygous or the homozygous variant
                    genotype, and the second dummy variable is set to 1 if the SNP is of the ho-
                    mozygous variant genotype. If FALSE, three dummy variables are used and each
                    of the three genotypes of a SNP is coded by one of these binary variables.

refAA   codes "AA" always for the homozygous reference gentoype? Only considered if
                    the SNPs are coded by "AA", "AB" and "BB". If FALSE, it is evaluated SNPwise
                    whether "AA" or "BB" occurs more often, and the more frequently occuring value
                    is assumed to be the homozygous reference genotype.

snp.in.col   does each column of mat correspond to a SNP (and each row to an observation)?
                    If FALSE, it is assumed that each row represents a SNP, and each column an
                    observation.

monomorph   a non-negative number. If a dummy variable contains monomorph or less values
                    that differ from the more frequent value of this variable, then the variable is
                    removed from the data set.

Value

A matrix containing the binary dummy variables.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

recodeSNPs, recodeAffySNP

Examples

```
## Not run:
# Generate an example data set consisting of 10 rows (observations)
# and 5 columns (SNPs).
```

```
mat <- matrix(sample(3, 50, TRUE), 10)
colnames(mat) <- paste("SNP", 1:5, sep = "")

# Transform each SNP into two dummy variables, one that codes for
# a recessive effect and one that codes for a dominant effect.

snp2bin(mat)

# Transform each SNP into three dummy variables, where each of
# these variables codes for one of the three genotypes.

snp2bin(mat, domrec = FALSE)

## End(Not run)
```

---

summary.simSNPglm   *Summarizing a simSNPglm object*

Description

Summarizes an object of class simSNPglm.

Usage

```
## S3 method for class 'simSNPglm'
summary(object, digits = 3, ...)
```

Arguments

object	an object of class simSNPglm, i.e.\ the output of [simulateSNPglm](#).
digits	number of digits used in the output.
...	Ignored.

Value

Shows the model used in [simulateSNPglm](#) to generate the values of the response. If the response is binary, then it additionally shows and returns a contingency table of the numbers of cases and controls and the probabilities for being a case.

Author(s)

Holger Schwender, <holger.schwender@udo.edu>

See Also

[simulateSNPglm](#)

Examples

```
## Not run:
# The default simulated data set.

sim1 <- simulateSNPglm()
sim1

# A bit more information: Table of probability of being a case
# vs. number of cases and controls.

summary(sim1)


## End(Not run)
```

Index