

Package ‘CNVassoc’

April 12, 2016

Type Package

Title Association Analysis of CNV Data and Imputed SNPs

Version 2.2

Date 2016-04-12

Depends R (>= 2.15.0), CNVassocData, mixdist, mclust, survival

Suggests CGHcall, CGHregions, CNVtools, xtable, MASS, Biobase, CGHbase

Author Juan R González, Isaac Subirana

Maintainer Isaac Subirana <isubirana@imim.es>

Description Carries out analysis of common

Copy Number Variants (CNVs) and imputed Single Nucleotide Polymorphisms (SNPs) in population-based studies.

It includes tools for estimating association under a series of study designs (case-control, cohort, etc), using several dependent variables (class status, censored data, counts) as response, adjusting for covariates and considering various inheritance models. Moreover, it is possible to perform epistasis studies with pairs of CNVs or imputed SNPs. It has been optimized in order to make feasible the analyses of Genome Wide Association studies (GWAs) with hundreds of thousands of genetic variants (CNVs / imputed SNPs). Also, it incorporates functions for inferring copy number (CNV genotype calling). Various classes and methods for generic functions (print, summary, plot, anova, ...) have been created to facilitate the analysis.

License GPL (>= 2)

LazyLoad yes

URL <http://www.creal.cat/jrgonzalez/software.htm>

Encoding latin1

BuildVignettes True

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-04-12 17:01:26

R topics documented:

cnv	2
CNVassoc	4
CNVtest	6
fastCNVassoc	7
fastCNVinter	10
getProbs	12
getProbsRegions	13
getPvalBH	14
getQualityScore	14
multiCNVassoc	16
plotSignal	17
simCNVdataBinary	18
simCNVdataCaseCon	19
simCNVdataNorm	20
simCNVdataPois	22
simCNVdataWeibull	23
Index	25

cnv	<i>CNV object</i>
-----	-------------------

Description

cnv creates a 'cnv' object is returns TRUE if x is of class 'cnv' print gives a summary for an object of class 'cnv' including ... plot plots an object of class 'cnv' ...

Usage

```
cnv(x, batches, ...)
cnvDefault(x, num.copies, num.class, cnv.tol = 0.001, mix.method = "mixdist",
  check.probs = TRUE, threshold.0, threshold.k, mu.ini, sigma.ini, pi.ini,
  cutoffs = NULL, check.alpha = 0.05, check.cnv = TRUE, var.equal)
cnvBatches(intensities, batches, threshold.0, threshold.k, common.pi = TRUE, ...)
is.cnv(obj)
## S3 method for class 'cnv'
plot(x, ...)
## S3 method for class 'cnv'
print(x, digits = 4, ...)
```

Arguments

x	a vector of CNV intensity signal for each individual, or a matrix with CNV calling probabilities per row
num.copies	vector with copy number status values, i.e, number of copies or a vector of characters indicating loss ('l'), normal ('n') or gain ('g') for example

<code>num.class</code>	integer indicating how many classes CNV contains
<code>cnv.tol</code>	error tolerance when <code>x</code> is a probability matrix and row sums are not identical to one
<code>mix.method</code>	normal mixture fitting method when <code>x</code> is a vector of univariate CNV signal intensities. Current methods are "mixdist" that uses the function <code>mix</code> from the package <code>mixdist</code> , "mclust" that uses the function <code>Mclust</code> from the package <code>mclust</code> and "EMmixt" that uses an internal function <code>EMmixt</code> from the <code>CNVassoc</code> package. The last two are based on Expectation-Maximization procedure and the first one is based on quasi-Newton-Raphson procedure
<code>check.probs</code>	logical. If TRUE it checks whether row sums are equal to one +/- <code>cnv.tol</code> when <code>x</code> is a probability matrix
<code>threshold.0</code>	assigns zero copies (or first copy number status) to all individuals whose CNV signal intensity is lower than <code>threshold.0</code>
<code>threshold.k</code>	assigns <code>k</code> copies (or last copy number status) to all individuals whose CNV signal intensity is bigger than <code>threshold.k</code>
<code>mu.ini</code>	an optional vector to specify the initial values of means when fitting a normal mixture to CNV intensity signal data
<code>sigma.ini</code>	an optional vector to specify the initial values of standard deviations when fitting a normal mixture to CNV intensity signal data
<code>pi.ini</code>	an optional vector to specify the initial values of copy number status probabilities when fitting a normal mixture to CNV intensity signal data
<code>cutoffs</code>	a vector indicating the cut-off points to assign the copy number status assign individuals to the individuals according to the categories defined by these cut-off points on CNV intensity signal data
<code>check.alpha</code>	significance level to goodness-of-fit test indicating whether the normal mixture model to CNV intensity data has been fitted appropriately
<code>check.cnv</code>	logical. If TRUE, <code>cnv</code> functions returns an error when normal mixture model does not fit well to the univariate CNV intensity signal data
<code>var.equal</code>	logical. If TRUE, standard deviation are supposed to be the same for all copy number status when fitting univariate CNV intensity signal data
<code>intensities</code>	a vector with the univariate CNV intensity signal data
<code>batches</code>	a vector indicating the batch (leave it missing if no batch effect is present)
<code>common.pi</code>	logical. If TRUE, copy number status probabilities for each individual are computed estimating specific means and standard deviations separately for every batch, but the same population copy number status probabilities for all batches. It is suggested to leave it as TRUE
<code>obj</code>	an object of any class
<code>digits</code>	number of digits when printing a <code>cnv</code> object
<code>...</code>	other arguments passed to <code>cnvDefault</code> , <code>print.default</code> or <code>plot.cnv</code> . The arguments passed to <code>plot.cnv</code> are the same as the ones for the <code>plotSignal</code> function

Details

When argument `batches` is not specified, then `cnvDefault` is used, otherwise `cnvBatch` is called. If univariate CNV intensity signal data is used to create the `cnv` class object, then one can introduce the batch effect if it necessary. But, if other algorithms have been used previously and the `cnv` class object is created directly from the CNV calling probabilities matrix, then it is not possible to specify the batch argument. The batch effect is important when cases and controls have been genotyped in different platforms for example. In this situations, the platform should be introduced in the batch argument as a vector indicating which platform every CNV intensity signal data comes from. Generic plot function applied on a `'cnv'` class object performs two types of plots whether `'cnv'` class object has been created from univariate CNV intensity signal data or whether it has been created directly from a probability matrix provided by any CNV calling algorithm. The first type is a plot similar to the one created by `plotSignal` function, and the second type is a barplot.

Value

`cnv` return an object of class `'cnv'` with generic function such as `print` or `plot` implemented for this kind of objects. `is.cnv` is a function that returns TRUE or FALSE whether `obj` is of class `'cnv'` or not.

References

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[CNVassoc](#), [plotSignal](#)

Examples

```
data(dataMLPA)
CNV <- cnv(x = dataMLPA$Gene2, threshold.0 = 0.01, mix.method = "mixdist")
CNV
plot(CNV)
```

CNVassoc

Association analysis between a CNV and phenotype

Description

This function performs an association analysis between a CNV and a dependent variable (phenotype) using a latent class model that incorporates the uncertainty arising from calling procedure. The phenotype may be quantitative or categorical. In the second case (e.g. case-control studies) this variable must be coded as 1 (for cases) and 0 (for controls). The association can be adjusted for other covariates (e.g. clinical covariates, stratification, ...)

Usage

```

CNVassoc(formula, data, subset, na.action, model = "multiplicative",
          family = "binomial", tol = 1e-06, max.iter = 30, emsteps = 0,
          verbose = FALSE, coef.start, sigma.start, alpha.start=1)

```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Right side of ~ should have an object of class 'cnv'.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)'.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain 'NA's. The default is set by the 'na.action' setting of 'options', and is 'na.fail' if that is unset. The 'factory-fresh' default is 'na.omit'. Another possible value is 'NULL', no action. Value 'na.exclude' can be useful.
model	Genetic model to be tested. Possible values are "multiplicative" (model free, e.g. co-dominant) or "additive", partial matching allowed. Default value is "multiplicative".
family	a description of the error distribution and link function to be used in the model. This must be a character string naming a family function. Possible values are "binomial", "gaussian", "poisson" or "weibull". Default value is "binomial"
tol	Tolerance for convergence in fitting model. Default value is 1e-06.
max.iter	Maximum number of iterations in fitting model. Default value is 30.
emsteps	Number of iterations using Expectation Maximization (EM) algorithm to set initial values before using Newton-Rapson (NR) in fitting model. Default value is zero, that means that EM step is not performed
verbose	logical. If TRUE parameter values for each iteration are shown in the console. Default value is FALSE
coef.start	initial values for coefficients in NR procedure
sigma.start	initial values for scale parameter (only for "gaussian") in NR procedure
alpha.start	initial values for shape parameter (only for "weibull") in NR procedure

Value

An object of class 'CNVassoc'.

'print' returns model parameter estimates

'summary' returns a summary table similar to summary.glm

'anova' performs a Likelihood Ratio Test comparing two nested models fitted using CNVassoc

'logLik' returns the log-likelihood of a model fitted using CNVassoc

See examples for further illustration about all previous issues.

References

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[cnv](#), [CNVtest](#)

Examples

```
data(dataMLPA)
CNV <- cnv(x = dataMLPA$Gene2, threshold.0 = 0.01, mix.method = "mixdist")
modmul <- CNVassoc(casco ~ CNV, data = dataMLPA, model = "mul")
modmul
summary(modmul)
anova(modmul, update(modmul, model="add"))
logLik(modmul)
```

CNVtest

Testing association between a CNV and phenotype

Description

This function perform a Wald test or a Likelihood Ratio Test (LRT) to determine whether a CNV is associated with the phenotype.

Usage

```
CNVtest(x, type = "Wald")
## S3 method for class 'CNVtest'
print(x, ...)
```

Arguments

x	An object of class 'CNVassoc'
type	The statistical test used. Possible values are "Wald" for Wald test or "LRT" for Likelihood Ratio Test
...	Further arguments passed to or from other methods

Value

An object of class 'CNVtest' with methods for the generic function print, returning the Chi-squared value, its degrees of freedom and the corresponding p-value of CNV significance in the associated test fitted by CNVassoc function.

References

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[CNVassoc](#), [cnv](#)

Examples

```
data(dataMLPA)
CNV<- cnv(x = dataMLPA$Gene2, threshold.0 = 0.01, mix.method = "mixdist")
mod<-CNVassoc(formula = casco ~ CNV, data = dataMLPA, model = "mul")
CNVtest(mod, type = "LRT")
CNVtest(mod, type = "Wald")
```

fastCNVassoc	<i>Fast association analysis between a CNV or imputed SNPs and phenotype for case-control and cohort GWA studies.</i>
--------------	---

Description

This function performs an association analyses with several genetic variants with uncertainty (CNVs or imputed SNPs) and a response, maybe adjusting for covariates (e.g. clinical covariates, stratification, ...). It uses the Newthorn-Raphson procedure with analytic likelihood derivatives and it has written in C language to speed up the process making feasible to analyse hundreds of thousands of variants. It also incorporates the possibility to use several cores to make calculations even faster.

Usage

```
fastCNVassoc(probs, formula, data, model = "additive",
  family = "binomial", nclass = 3, colskip = 5, tol = 1e-06, max.iter = 30,
  verbose = FALSE, multicores=0)
```

Arguments

probs	either a matrix containing the probabilities of genetic variants in IMPUTE format (i.e. each rows represent a variant and every 'nclass' columns an individual. The first 'colskip' columns refers to the variant info (e.g. position, rs name, alleles, etc.).
formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. In the right side of ~ covariates must be included. If no covariates are present in the model just type 1.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)'.

model	Genetic model to be tested. Possible values are "multiplicative" (model free, e.g. co-dominant) or "additive", partial matching allowed. Only "additive" model is implemented.
family	a description of the error distribution and link function to be used in the model. This must be a character string naming a family function. Possible values are "binomial" and "weibull". Default value is "binomial"
nclass	integer specifying the number of possible alleles or genotypes. Default value is 3 (typically for SNPs).
colskip	integer specifying the number of columns to be skipped in order to read the probabilities. This columns may contain the SNPs info (such as rs name, position, alleles and chromosome). Default value is 5 for IMPUTE format files.
tol	Tolerance for convergence in fitting model. Default value is 1e-06.
max.iter	Maximum number of iterations in fitting model. Default value is 30.
verbose	logical. If TRUE the number of current analysed variants is shown in the console. Default value is FALSE
multicores	integer indicating the number of cores to be used. It uses 'parallel'. Default value is 0 indicating that only one core is used and 'parallel' package is not required. For Windows OS, 'multicores'>1 is not supported.

Value

A data.frame with the following variables:

- variant: consecutive integer from one to the number of analyzed variants (CNV or imputed SNPs) in the same order as in the probs matrix.
- beta coefficient: log-Odds Ratio for binary response or log-Hazard Ratio for time-to-event response.
- se: standard error of beta coefficient.
- zscore: ratio between beta and se
- pvalue: p-value of association between each genetic variant and response, maybe adjusted by covariates.
- iter: number of iterations necessary achieve convergence during the Newton-Raphson algorithm used to fit the model.

See examples for further illustration about all previous issues.

Note

The order of individuals from probabilities ('probs' argument) matrix must be the same as in the response and covariates variables.

The 'subset' and 'na.action' is not implemented. Therefore, no missings are allowed in probabilities, response or covariates.

It is important be aware whether the number of iterations has achieved the maximum (30 by default). In this case, the results may be not reliable.

The probability matrix ('probs') must have 'nclass' * N + 'colskip' columns, where N is the number of individuals.

References

Subirana I, González JR. Genetic Association Analysis and Meta-Analysis of Imputed SNPs in Longitudinal Studies. *Genet Epidemiol*, 2013 Jul;37(5):465-77.

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[CNVassoc](#), [multiCNVassoc](#), [CNVtest](#)

Examples

```
## Not run:

require(CNVassoc)
require(parallel)

# read imputed SNP probabilities from a file-.
# Example from SNPTEST software of 500 cases and 500 controls on 200 imputed SNPS.
fileprobs <- system.file("exdata/SNPTEST.probs",package="CNVassocData")

# build response (500 controls and 500 cases).
resp<-rep(0:1,each=500)

# generate two covariates randomly
N<-1000
covar1<-rnorm(N) # continuous covariate
covar2<-factor(sample(1:3,N,replace=TRUE),labels=c("A","B","C")) # categorical covariate

# run with 6 cores. Under Windows OS, multicore must be <=1.
system.time(
  res<-fastCNVassoc(fileprobs,resp~covar1+covar2,family="binomial",multicore=6)
)
res

# build a time-to-event response randomly
set.seed(123456)
times <- rexp(N,1)
cens <- rbinom(N,1,0.8)

system.time(
  res<-fastCNVassoc(fileprobs,Surv(times,cens)~covar1+covar2,family="weibull",multicore=6)
)
res

## End(Not run)
```

fastCNVinter

Fast epistasis (pairwise interaction) analysis between two CNV or imputed SNPs for case-control and cohort GWA studies.

Description

This function assess the association of the interaction of two genetic variants measured with uncertainty (CNVs or imputed SNPs) and a response, maybe adjusting for covariates (e.g. clinical covariates, stratification, ...). It uses the Newthron-Raphson procedure with analytic likelihood derivatives and it has written in C language to speed up the process making feasible to analyse hundreds of thousands of variants. It also incorporates the possibility to use several cores to make calculations even faster.

Usage

```
fastCNVinter(probs, formula, data, model = "additive",
             family = "binomial", nclass = 3, colskip = 5, tol = 1e-06, max.iter = 30,
             verbose = FALSE, multicores=0)
```

Arguments

probs	either a matrix containing the probabilities of genetic variants in IMPUTE format (i.e. each rows represent a variant and every 'nclass' columns an individual. The first 'colskip' columns refers to the variant info (e.g. position, rs name, alleles, etc.).
formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. In the right side of ~ covariates must be included. If no covariates are present in the model just type 1.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)'.
model	Genetic model to be tested. Possible values are "multiplicative" (model free, e.g. co-dominant) or "additive", partial matching allowed. Only "additive" model is implemented.
family	a description of the error distribution and link function to be used in the model. This must be a character string naming a family function. Possible values are "binomial" and "weibull". Default value is "binomial"
nclass	integer specifying the number of possibles alleles or genotypes. Default value is 3 (tipycally for SNPs).
colskip	integer specifying the number of columns to be skipped in order to read the probabilities. This columns may contain the SNPs info (such as rs name, position, alleles and chromosome). Default value is 5 for IMPUTE format files.

tol	Tolerance for convergence in fitting model. Default value is 1e-06.
max.iter	Maximum number of iterations in fitting model. Default value is 30.
verbose	logical. If TRUE the number of current analysed variants is shown in the console. Default value is FALSE
multicores	integer indicating the number of cores to be used. It uses 'parallel'. Default value is 0 indicating that only one core is used and 'parallel' package is not required. For Windows OS, 'multicores'>1 is not supported.

Value

A data.frame with the following variables:

- pair: indexes corresponding to the pair of genetic variants (CNV or imputed SNPs) in the same order as in the probs matrix.
- beta coefficient: log-Odds Ratio for binary response or log-Hazard Ratio for time-to-event response.
- se: standard error of beta coefficient.
- zscore: ratio between beta and se
- pvalue: p-value of association between each genetic variant and response, maybe adjusted by covariates.
- iter: number of iterations necessary achieve convergence during the Newton-Raphson algorithm used to fit the model.

See examples for further illustration about all previous issues.

Note

The order of individuals from probabilities ('probs' argument) matrix must be the same as in the response and covariates variables.

The 'subset' and 'na.action' is not implemented. Therefore, no missings are allowed in probabilities, response or covariates.

It is important be aware whether the number of iterations has achieved the maximum (30 by default). In this case, the results may be not reliable.

The probability matrix ('probs') must have 'nclass' * N + 'colskip' columns, where N is the number of individuals.

References

Subirana I, González JR. Interaction association analysis of imputed SNPs in case control and cohort studies. *Genet Epidemiol*, 2015 Jan 22; doi: 10.1002/gepi.21883. [Epub ahead of print]

Subirana I, González JR. Genetic Association Analysis and Meta-Analysis of Imputed SNPs in Longitudinal Studies. *Genet Epidemiol*, 2013 Jul;37(5):465-77.

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also[fastCNVassoc](#)**Examples**

```
## Not run:

require(CNVassoc)
require(parallel)

# read imputed SNP probabilities from a file-.
# Example from SNPTTEST software of 500 cases and 500 controls on 200 imputed SNPS.
fileprobs <- system.file("exdata/SNPTEST.probs",package="CNVassocData")

# build response (500 controls and 500 cases).
resp<-rep(0:1,each=500)

# generate two covariates randomly
N<-1000
covar1<-rnorm(N) # continuous covariate
covar2<-factor(sample(1:3,N,replace=TRUE),labels=c("A","B","C")) # categorical covariate

# run with 6 cores. Under Windows OS, multicore must be <=1.
system.time(
res<-fastCNVinter(fileprobs,resp~covar1+covar2,family="binomial",multicore=6)
)
res

# build a time-to-event response randomly
set.seed(123456)
times <- rexp(N,1)
cens <- rbinom(N,1,0.8)

system.time(
res2<-fastCNVinter(fileprobs,Surv(times,cens)~covar1+covar2,family="weibull",multicore=6)
)
res2

## End(Not run)
```

Description

This function creates a list where each component correspond to a given probe. Any component contains the posterior probabilities obtained from CGHcall algorithm.

Usage

```
getProbs(x)
## S3 method for class 'cghCall'
getProbs(x)
## S3 method for class 'cnv'
getProbs(x)
```

Arguments

x an object of class 'CGHcall' or 'cnv'

Value

A list where each component correspond to a given probe. Any component contains the posterior probabilities obtained from CGHcall algorithm

Note

See vignette for an example.

Author(s)

This function was created using a script kindly provided by Mark van de Wiel

getProbsRegions	<i>Get posterior probabilities for blocks/regions</i>
-----------------	---

Description

See vignette for further details.

Usage

```
getProbsRegions(probs,regions,intensities,nclass=3)
```

Arguments

probs	probabilities from CGH calling algorithms
regions	probs that define a segment
intensities	mean probe intensities for each region
nclass	See 'nclass' argument of function CGHcall

Note

See vignette for an example.

getPvalBH

Corrected p values using Benjamini & Hochberg approach

Description

This functions corrects the association p-values using the Benjamini & Hochberg approachby for multiple testing.

Usage

```
getPvalBH(x)
```

Arguments

x a list containing p values

Details

This function calls 'p.adjust' to compute 'BH' correction

Value

A data frame with the blocks and corrected p-values

See Also

[p.adjust](#)

getQualityScore

Computes a quality score for a CNV fit

Description

This function provides different types of measurements of uncertainty after CNV calling

Usage

```
getQualityScore(x, ...)
## Default S3 method:
getQualityScore(x, sds, w, type, iter = 10000, threshold = 0.1, ...)
## S3 method for class 'cnv'
getQualityScore(x, type = "class", iter = 10000, threshold = 0.1, ...)
```

Arguments

<code>x</code>	and object of class <code>cnv</code> or means vector of intensity signal for each copy number status
<code>...</code>	further arguments passed to or from <code>getQualityScore</code> methods
<code>type</code>	the type of quality score measurement computed. Possible values are "class", "CNVtools" or "CANARY" (see Details)
<code>iter</code>	number of iterations when <code>type = 'class'</code> or <code>type = 'CANARY'</code> is specified
<code>threshold</code>	a value to compute the proportion of sample individuals with confidence score bigger than it (see Details)
<code>sds</code>	standard deviations vector of intensity signal for each copy number status
<code>w</code>	copy number status proportions vector

Details

The quality scores measures how well the clusters are separated. It compares the locations of the means with the standard error for each pair of adjacent cluster. Obviously, except for probability of good classification (`type="class"`), the lower quality score the highest uncertainty. There are 3 possible types of quality score measurements: "class": probability of good classification), "CNVtools": the score defined in 'CNVtools' package) and "CANARY": proportions of sample individuals with confidence score bigger than threshold. The confidence score is defined as the ratio between the second biggest copy number call probability divided by the biggest one.

Value

An object of class `getQualityScore` with a single number of quality score.

Note

For `cnv` objects created directly from probabilities and not from fitting a univariate intensity signal, only "class" quality score type can be calculated.

Examples

```
data(dataMLPA)
CNV<-cnv(x = dataMLPA$Gene2, threshold.0 = 0.01, mix.method = "mixdist")
getQualityScore(CNV,type="class")
getQualityScore(CNV,type="CNVtools")
getQualityScore(CNV,type="CANARY")
```

multiCNVassoc

Association between several CNVs and disease

Description

This function repeatedly calls CNVassoc function

Usage

```
multiCNVassoc(x, formula, num.copies = 0:2, cnv.tol = 0.01, ...)
```

Arguments

x	a list of calling probabilities matrix for each CNV
formula	see 'formula' argument of CNVassoc function.
num.copies	See argument of cnv function.
cnv.tol	See argument of cnv function.
...	other arguments passed through 'CNVassoc' function.

Details

See vignette for an example

Value

A list of p-values for each CNV

References

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[CNVassoc](#)

plotSignal	<i>plots the intensities of a CNV univariate signal data</i>
------------	--

Description

This function creates a plot with probe intensity

Usage

```
plotSignal(x, my.colors = c("black", "red", "blue"), ylab = "Peak Intensity",  
           xlab = c("individuals", "Phenotype"), case.control = NULL, cex.legend = 0.6,  
           dens.bw = "nrd0", dens.adjust = 1, n = 0, ...)
```

Arguments

x	A vector with probe intensities
my.colors	Colours for each copy number status.
ylab	Label of y-axis
xlab	Label of x-axis
case.control	Vector indicating case-control status
cex.legend	Size of legend
dens.bw	Adjustment for intensity signal density curve. See argument 'bw' of density function for more details
dens.adjust	Adjustment for intensity signal density curve. See argument 'adjust' of density function for more details
n	integer indicating the number of points to be placed on the plot interactively (using locator)) to define the thresholds that separate the different copy number status, colouring the points differently according to the assigned copy number status. If it zero or negative it makes no possible to place the threshold points.
...	Other arguments passed to plot.default

Details

See vignette for further description

References

Gonzalez JR, Subirana I, Escaramis G, Peraza S, Caceres A, Estivill X and Armengol L. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009;10:172.

See Also

[cnv](#)

Examples

```
data(dataMLPA)
plotSignal(dataMLPA$Gene2)
```

simCNVdataBinary	<i>Simulation of CNV and discrete traits</i>
------------------	--

Description

This function simulates intensity for a CNV and a binary trait response for different scenarios

Usage

```
simCNVdataBinary(n, mu.surrog, sd.surrog, w, p0, or, cnv.random = FALSE)
```

Arguments

n	number of simulated individuals
mu.surrog	a vector of intensity signal means for every copy number status
sd.surrog	a vector of intensity signal standard deviations for every copy number status
w	a vector of copy number status proportions
p0	prevalence of disease (trait) for populations with zero copies (reference category)
or	a vector of odds ratio for one, two,... copies respect to zero copies
cnv.random	A logical value. TRUE means that copy number status is drawn under a multinomial distribution with proportions indicated by 'w'. FALSE means that the real simulated frequency is always the same and is rounded to the most similar integer to the frequencies indicated by 'w'. Default value is FALSE

Details

This function is useful to calculate the power of association models with binary traits under different scenarios, e.g. setting different degrees of association (odds ratios), considering different degrees of uncertainty controlled by the distribution of intensity signal data, i.e. mean `mu.surrog`, standard deviation `sd.surrog` and proportion `w`, etc.

Value

Data frame with individual simulated data per row and with the following variables:

resp	Trait (response) variable following a Bernoulli distribution given the CNV status
surrog	Signal intensity following a mixture of normals with means, standard deviations and proportions specified by <code>mu.surrog</code> , <code>sd.surrog</code> and <code>w</code> respectively.
cnv	True copy number status

See Also

[simCNVdataCaseCon](#), [simCNVdataNorm](#), [simCNVdataPois](#), [simCNVdataWeibull](#), [cnv](#), [CNVassoc](#)

Examples

```
maf<-0.3
set.seed(123)
simData<-simCNVdataBinary(n=1000, mu.surrog=c(0,0.5,1), sd.surrog=rep(0.15,3),
  w=c((1-maf)^2,2*maf*(1-maf),maf^2), p0=0.1, or=c(1.3,1.3^2), cnv.random = FALSE)
CNV<-cnv(simData$surrog,mix.method="EMmixt")
getQualityScore(CNV,type="CNVtools")
mod<-CNVassoc(resp~CNV,data=simData,family="binomial")
CNVtest(mod)
summary(mod)
```

simCNVdataCaseCon	<i>Simulation of CNV in a case-control study design</i>
-------------------	---

Description

This function simulates intensity for a CNV within cases and control groups for different scenarios

Usage

```
simCNVdataCaseCon(n0, n1, w0, or, mu.surrog0, sd.surrog0, mu.surrog1 = mu.surrog0,
  sd.surrog1 = sd.surrog0, random = TRUE)
```

Arguments

n0	number of controls simulated
n1	number of cases simulated
w0	vector of proportions of copy number status in controls
or	a vector of odds ratio for one, two,... copies respect to zero copies
mu.surrog0	vector of means of CNV intensity signal, per copy number status, in control group
sd.surrog0	vector of standard deviations of CNV intensity signal, per copy number status, in control group
mu.surrog1	vector of means of CNV intensity signal, per copy number status, in control group
sd.surrog1	vector of standard deviations of CNV intensity signal, per copy number status, in control group
random	A logical value. TRUE means that individuals (rows) are randomly permuted, and FALSE means that simulated 'data.frame' contains controls first and then cases. Default value is TRUE

Details

This function is useful to calculate the power of association models in a case control study design under different scenarios ,e.g. setting different degrees of association (odds ratios), considering different degrees of uncertainty controlled by the distribution of intensity signal data, i.e. mean `mu.surrog`, standard deviation `sd.surrog` and proportion `w`, etc.

Value

Data frame with individual simulated data per row and with the following variables:

<code>resp</code>	Trait (response) variable with 0 or 1 if the individual is a control or a case respectively
<code>surrog</code>	Signal intensity following a mixture of normals with means, standard deviations and proportions specified by <code>mu.surrog</code> , <code>sd.surrog</code> and <code>w</code> respectively, within cases and controls
<code>cnv</code>	True copy number status

See Also

[simCNVdataBinary](#), [simCNVdataNorm](#), [simCNVdataPois](#), [simCNVdataWeibull](#), [cnv](#), [CNVassoc](#)

Examples

```
maf<-0.3
set.seed(123)
simData<-simCNVdataCaseCon(n0=1000, n1=1000, mu.surrog0=c(0,0.5,1), sd.surrog0=rep(0.15,3),
  mu.surrog1=c(0,0.5,1), sd.surrog1=rep(0.15,3),
  w0=c((1-maf)^2,2*maf*(1-maf), maf^2), or=c(1.3,1.3^2),
  random = FALSE)
CNV<-cnv(simData$surrog,mix.method="EMmixt")
getQualityScore(CNV,type="CNVtools")
mod<-CNVassoc(resp~CNV,data=simData,family="binomial")
CNVtest(mod)
summary(mod)
```

simCNVdataNorm

Simulation of CNV and quantitative traits

Description

This function simulates intensity for a CNV and a quantitative trait response for different scenarios

Usage

```
simCNVdataNorm(n, mu.surrog, sd.surrog, w, mu.y, sd.y, cnv.random = FALSE)
```

Arguments

<code>n</code>	An integer indicating the desired number of individuals to be simulated
<code>mu.surrog</code>	A vector containing the signal (surrogate variable) means for every copy number status (latent classes). Its length must be equal to the number of latent classes
<code>sd.surrog</code>	A vector containing the signal standard deviation for every copy number status. Its length must be equal to <code>mu.surrog</code> .
<code>w</code>	A vector containing the frequencies for every copy number status. Its length must be equal to <code>mu.surrog</code> and its components must sum up one.
<code>mu.y</code>	A vector containing the means of the response variable for every copy number status. Its length must be equal to <code>mu.surrog</code> .
<code>sd.y</code>	A single number indicating the residual standard deviation
<code>cnv.random</code>	A logical value. TRUE means that copy number status is drawn under a multinomial distribution with proportions indicated by 'w'. FALSE means that the real simulated frequency is always the same and is rounded to the most similar integer to the frequencies indicated by 'w'. Default value is FALSE

Details

This function is useful to calculate the power of association models for a continuous (normal-distributed) trait under different scenarios ,e.g. setting different degrees of association (effects), considering different degrees of uncertainty controlled by the distribution of intensity signal data, i.e. mean `mu.surrog`, standard deviation `sd.surrog` and proportion `w`, etc.

Value

Data frame with individual simulated data per row and with the following variables:

<code>resp</code>	Continuous trait variable (response)
<code>surrog</code>	Signal intensity following a mixture of normals with means, standard deviations and proportions specified by <code>mu.surrog</code> , <code>sd.surrog</code> and <code>w</code> respectively
<code>cnv</code>	True copy number status

See Also

[simCNVdataBinary](#), [simCNVdataCaseCon](#), [simCNVdataPois](#), [simCNVdataWeibull](#), [cnv](#), [CNVassoc](#)

Examples

```
set.seed(123)
maf<-0.3
effect<-3
simData<-simCNVdataNorm(n=1000, mu.surrog=c(0,0.5,1), sd.surrog=rep(0.15,3),
  w=c((1-maf)^2,2*maf*(1-maf), maf^2), mu.y=100+c(0,effect,2*effect),
  sd.y=rep(20,3), cnv.random = FALSE)
CNV<-cnv(simData$surrog,mix.method="EMmixt")
getQualityScore(CNV,type="CNVtools")
mod<-CNVassoc(resp~CNV,data=simData,family="gaussian",emsteps=10)
CNVtest(mod)
summary(mod)
```

simCNVdataPois	<i>Simulate Poisson data</i>
----------------	------------------------------

Description

This function simulates intensity for a CNV and a discrete counting trait response for different scenarios

Usage

```
simCNVdataPois(n, mu.surrog, sd.surrog, w, lambda, cnv.random = FALSE)
```

Arguments

n	An integer indicating the desired number of individuals to be simulated
mu.surrog	A vector containing the signal (surrogate variable) means for every copy number status (latent classes). Its length must be equal to the number of latent classes
sd.surrog	A vector containing the signal standard deviation for every copy number status. Its length must be equal to mu.surrog.
w	A vector containing the frequencies for every copy number status. Its length must be equal to mu.surrog and its components must sum up one.
lambda	A vector containing the means of the response variable for every copy number status. Its length must be equal to mu.surrog.
cnv.random	A logical value. TRUE means that copy number status is drawn under a multinomial distribution with proportions indicated by 'w'. FALSE means that the real simulated frequency is always the same and is rounded to the most similar integer to the frequencies indicated by 'w'. Default value is FALSE

Details

This function is useful to calculate the power of association models for discrete counting trait under different scenarios ,e.g. setting different degrees of association (risk ratios), considering different degrees of uncertainty controlled by the distribution of intensity signal data, i.e. mean mu.surrog, standard deviation sd.surrog and proportion w, etc.

Value

Data frame with individual simulated data per row and with the following variables:

resp	Discrete variable with simulated counts (response)
surrog	Signal intensity following a mixture of normals with means, standard deviations and proportions specified by mu.surrog, sd.surrog and w respectively
cnv	True copy number status

See Also

[simCNVdataBinary](#), [simCNVdataCaseCon](#), [simCNVdataNorm](#), [simCNVdataWeibull](#), [cnv](#), [CNVassoc](#)

Examples

```

set.seed(123)
rr<-1.5
maf<-0.3
simData<-simCNVdataPois(n=1000, mu.surrog=c(0,0.5,1), sd.surrog=rep(0.15,3),
  w=c((1-maf)^2,2*maf*(1-maf), maf^2), lambda=3*c(1,rr,rr^2), cnv.random = FALSE)
CNV<-cnv(simData$surrog,mix.method="EMmixt")
getQualityScore(CNV,type="CNVtools")
mod<-CNVassoc(resp~CNV,data=simData,family="poisson",emsteps=10)
CNVtest(mod)
summary(mod)

```

simCNVdataWeibull

*Simulate of CNV and a right censored Weibull distributed trait***Description**

This function simulates intensity for a CNV and a time to event response (followed-up cohort study design) for different scenarios

Usage

```

simCNVdataWeibull(n, mu.surrog, sd.surrog, w, lambda, shape, time.cens = Inf,
  cnv.random = FALSE)

```

Arguments

n	An integer indicating the desired number of individuals to be simulated
mu.surrog	A vector containing the signal (surrogate variable) means for every copy number status (latent classes). Its length must be equal to the number of latent classes
sd.surrog	A vector containing the signal standard deviation for every copy number status. Its length must be equal to mu.surrog.
w	A vector containing the frequencies for every copy number status. Its length must be equal to mu.surrog and its components must sum up one.
lambda	A vector containing the means of the response variable for every copy number status
shape	A vector containing the shape of the response variable for every copy number status
time.cens	Censoring time, e.g. end of follow-up
cnv.random	A logical value. TRUE means that copy number status is drawn under a multinomial distribution with proportions indicated by 'w'. FALSE means that the real simulated frequency is always the same and is rounded to the most similar integer to the frequencies indicated by 'w'. Default value is FALSE

Value

Data frame with individual simulated data per row and with the following variables:

resp	Time to event or censoring variable (response)
cens	Censoring indicator
surrog	Signal intensity following a mixture of normals with means, standard deviations and proportions specified by mu.surrog, sd.surrog and w respectively
cnv	True copy number status

See Also

[simCNVdataBinary](#), [simCNVdataCaseCon](#), [simCNVdataPois](#), [simCNVdataNorm](#), [cnv](#), [CNVassoc](#)

Examples

```
library(survival)
maf<-0.3
hr<-1.5
set.seed(123)
simData<-simCNVdataWeibull(n=3000, mu.surrog=c(0,0.5,1), sd.surrog=rep(0.15,3),
  w=c((1-maf)^2,2*maf*(1-maf), maf^2), lambda=0.05*c(1,hr,hr^2), shape=rep(1,3),
  time.cens=1.5, cnv.random = FALSE)
CNV<-cnv(simData$surrog,mix.method="EMmixt")
getQualityScore(CNV,type="CNVtools")
mod<-CNVassoc(Surv(resp, cens)~CNV,data=simData,family="weibull")
CNVtest(mod)
summary(mod)
```


Index

*Topic **misc**

- cnv, [2](#)
- CNVassoc, [4](#)
- CNVtest, [6](#)
- fastCNVassoc, [7](#)
- fastCNVinter, [10](#)
- getProbsRegions, [13](#)
- getPvalBH, [14](#)
- getQualityScore, [14](#)
- multiCNVassoc, [16](#)
- simCNVdataNorm, [20](#)

*Topic **utilities**

- getProbs, [12](#)
- plotSignal, [17](#)
- simCNVdataBinary, [18](#)
- simCNVdataCaseCon, [19](#)
- simCNVdataPois, [22](#)
- simCNVdataWeibull, [23](#)

anova.CNVassoc (CNVassoc), [4](#)

CGHcall, [13](#)

cnv, [2](#), [6](#), [7](#), [16](#), [17](#), [19–22](#), [24](#)

CNVassoc, [4](#), [4](#), [7](#), [9](#), [16](#), [19–22](#), [24](#)

cnvBatches (cnv), [2](#)

cnvDefault (cnv), [2](#)

CNVtest, [6](#), [6](#), [9](#)

density, [17](#)

fastCNVassoc, [7](#), [12](#)

fastCNVinter, [10](#)

getProbs, [12](#)

getProbsRegions, [13](#)

getPvalBH, [14](#)

getQualityScore, [14](#)

is.cnv (cnv), [2](#)

logLik.CNVassoc (CNVassoc), [4](#)

multiCNVassoc, [9](#), [16](#)

p.adjust, [14](#)

plot.cnv (cnv), [2](#)

plotSignal, [3](#), [4](#), [17](#)

print.anova.CNVassoc (CNVassoc), [4](#)

print.cnv (cnv), [2](#)

print.CNVassoc (CNVassoc), [4](#)

print.CNVtest (CNVtest), [6](#)

print.QualityScore (getQualityScore), [14](#)

print.summary.CNVassoc (CNVassoc), [4](#)

simCNVdataBinary, [18](#), [20–22](#), [24](#)

simCNVdataCaseCon, [19](#), [19](#), [21](#), [22](#), [24](#)

simCNVdataNorm, [19](#), [20](#), [20](#), [22](#), [24](#)

simCNVdataPois, [19–21](#), [22](#), [24](#)

simCNVdataWeibull, [19–22](#), [23](#)

summary.CNVassoc (CNVassoc), [4](#)