Package ‘CNVtools’

April 22, 2016

Type Package

Title A package to test genetic association with CNV data

Version 1.64.0

Date 2009-10-06

Author Chris Barnes <christopher.barnes@imperial.ac.uk> and Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk>

Maintainer Chris Barnes <christopher.barnes@imperial.ac.uk>

Description This package is meant to facilitate the testing of Copy Number Variant data for genetic association, typically in case-control studies.

License GPL-3

Depends R (>= 2.10), survival

biocViews GeneticVariability

NeedsCompilation yes

R topics documented:

CNVtools-package 2
A112 3
apply.ldf 3
apply.pca 4
CNV.fitModel 5
cnv.plot 6
CNVtest.binary 7
CNVtest.binary.T 10
CNVtest.qt 12
CNVtest.qt.T 14
CNVtest.select.model 16
compact.data.frame 18
EM.starting.point 19
ExpandData 20
get.model.spec 21
getparams 21

getQualityScore 22
qt.plot 23
test.posterior 24

Index **25**

`CNVtools-package` *CNVtools : CNV association studies*

Description

A package to perform robust case-control and quantitative trait association testing of Copy Number Variants.

Details

Package:	CNVtools
Type:	Package
Version:	1.42.3
Date:	2009-06-23
License:	Artistic License, v2.0
URL:	http://cnv-tools.sourceforge.net/

Main functions:

CNVtest.select.model
CNVtest.binary
CNVtest.qt
apply.pca
apply.ldf
cnv.plot
qt.plot

Author(s)

Chris Barnes <christopher.barnes@imperial.ac.uk> and Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk>

Maintainer: Chris Barnes <christopher.barnes@imperial.ac.uk>

References

"A robust statistical method for case-control association testing with Copy Number Variation" Barnes, C., Plagnol, V., Fitzgerald, T., Redon, R., V., Marchini, J., Clayton, D., Hurles, M. Nature Genetics, 2009.

A112 — *Copy Number Variant intensity data*

Description

Dummmy simulated data set. Will eventually contain a CNV from WTCCC data but awaiting authorization.

Usage

```
data(A112)
```

Source

Wellcome Trust Case Control Consortium

apply.ldf — *Applies a canonical correlation transformation to the data*

Description

Applies a canonical correlation transformation to the combination of the raw signal intensities with an initial set of posterior probabilities.

Usage

```
apply.ldf(full.signal, posterior)
```

Arguments

`full.signal`	A matrix with the raw signal intensity. One row per data point or sample in the data, and one column for the probability of each call. The matrix MUST have row names.
`posterior`	A matrix of posterior distribution for the calls. This matrix must have row names that match the signal intensity. The ordering does not have to be the same as the matrix of signals but each data point in "full.signal" must have a corresponding set of posterior probabilities.

Details

Do not forget to add row names to both matrices.

Value

A one-dimensional vector with the transformed canonical corelation transformed values.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

---

apply.pca *Applies to the data a principal component analysis*

Description

A simple wrapper around the R function prcomp.

Usage

```
apply.pca(matrix.signal)
```

Arguments

matrix.signal	A matrix containing the raw calls. The rows are the samples and the columns are the SNPs.

Value

A one dimensional vector, one value per sample: this is the first principal component.

Note

The output vector is normalized to have a standard deviation of 1.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

prcomp

`CNV.fitModel` *Fits a mixture of Gaussian to a set of one dimensional points.*

Description

This is the workhorse function, essentially an R wrapper around a lot of C code. It fits GLM models to the data.

Usage

```
CNV.fitModel(ncomp,
     nind,
     hyp = "H0",
     data,
     logit.offset,
             design.matrix.mean,
     design.matrix.variance,
             design.matrix.disease,
     pi.model = 0,
     mix.model = 10,
             control = list(tol = 1e-05, max.iter = 3000, min.freq= 4))
```

Arguments

`ncomp`	integer, number of components to fit to the data
`nind`	integer, total number of data points
`hyp`	Hypothesis, can be either H0 or H1
`data`	The data frame containing the data, in an expanded form (one point per individual and copy number)
`logit.offset`	An option most users will not use. It sets an offset when fitting the logit model for the disease status. This is used to obtain a profile likelihood when the disease parameter beta varies.
`design.matrix.mean`	The design matrix that relate mean cluster locations with batch.copy numbers.
`design.matrix.variance`	The design matrix for the cluster variances.
`design.matrix.disease`	The design matrix for the disease model.
`pi.model`	0,1,2 fit disease, hetero and quantitative models respectively.
`mix.model`	Specifies model for the components.
`control`	A list of parameters that control the behavior of the fitting.

Details

The user is very unlikely to actually use that function which is meant as an internal routine, a wrapper around the C code of the package. This function is called by the more user friendly function CNVtest.binary.

Value

`data`	The input expanded data frame, but with the posterior probabilities estimated.
`status`	A marker of convergence

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

CNVtest.binary

---

cnv.plot *Plots posterior probabilty distributions*

Description

Makes formatted density plots from the posterior data frame(s) returned by CNVtest.binary

Usage

```
cnv.plot(posterior, hist.or.dens='histogram', batch = NULL, freq = NULL, ...)
```

Arguments

`posterior`	The posterior distribution obtained from the CNVtools fitting algorithm, for example using CNVtest.binary
`hist.or.dens`	Either 'histogram' or 'density' to plot the data as an histogram or using a kernel density estimator
`batch`	character vector (usually of length 1, but not always), designing the batches one wants to plot.
`freq`	This argument is only relevant when hist.or.dens='histogram' (the default). It matches the argument freq of the hist function. With freq = FALSE frequencies, and not raw counts, are shown in the histogram.
`...`	Usual arguments passed to the hist function, including main or breaks for example.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

Examples

```
#Load data for CNV for two control cohorts
data(A112)
raw.signal <- as.matrix(A112[, -c(1,2)])
dimnames(raw.signal)[[1]] <- A112$subject

#Extract CNV signal using principal components
pca.signal <- apply.pca(raw.signal)

#Extract batch, sample and trait information
  batches <- factor(A112$cohort)
sample <- factor(A112$subject)
trait <- ifelse( A112$cohort == '58C', 0, 1)

#Fit the CNV with a three component model
fit.pca <- CNVtest.binary(signal = pca.signal, sample = sample, batch = batches,
              disease.status = trait, ncomp = 3, n.H0=3, n.H1=3,
  model.disease = "~ cn")

cnv.plot(fit.pca[['posterior.H0']], batch = '58C', breaks = 30)
```

CNVtest.binary *Fits a mixture of Gaussian to CNV data*

Description

This function fits a mixture of Gaussians to Copy Number Variant data, both under the null hypothesis of no association and under the alternate hypothesis that the CNV frequencies differ between cases and controls.

Usage

```
CNVtest.binary(signal, batch, sample = NULL, disease.status = NULL, ncomp,
       n.H0 = 5, n.H1 = 0,
               output = 'compact',
       model.mean = "~ strata(batch, cn)",
       model.var = "~ strata(batch, cn)",
               model.disease ="~ cn",
               association.test.strata = NULL,
       beta.estimated = NULL,
       start.mean = NULL,
               start.var = NULL,
       control = list(tol = 1e-05, max.iter = 3000,min.freq = 4))
```

Arguments

`signal` The vector of intensity values, meant to be a proxy for the number of copies.

`batch` Factor, that describes how the data points should be separated in batches, corresponding to different tehnologies to measure the number of DNA copies, or maybe different cohorts in a case control framework.

`sample` Optional (but recommended). A character vector containing a name for each data point, typically the name of the individuals.

`disease.status` In the case control situation a vector of 0 and 1 indicating which individuals are controls or cases.

`ncomp` Number of components one wants to fit to the data.

`n.H0` Number of times the EM should be used to maximize the likelihood under the null hypothesis of no association, each time with a different random starting point. The run that maximizes the likelihood is stored.

`n.H1` Number of times the EM should be used to maximize the likelihood under the alternate hypothesis of association present, each time with a different random starting point. The run that maximizes the likelihood is stored.

`output` The default value, "compact", returns a data frame with one line per sample. Any other setting witll return a much bigger data frame with one line per individual and copy number. This long format is the one used by the underlying fitting algorithm and is only useful if one attempts to use CNVtools in a non standard manner.

`model.mean` Formula that describes the linear model for the location of the mean signal intensity. The default is "~ strata(cn, batch)", which means that the mean intensity can take any value for any combination of the variables "cn" (for copy number) and "batch". More traditional model description such as ' ~ as.factor(cn)' for example are also possible, but are likely to be slower to fit and less numerically stable than the "strata" notation, which should be preferred.

`model.var` A formula as above, but to model the variances. Whenever possible and to maximise speed and stability the model should be specified using the strata command, for example "strata(batch, cn)" (the default), meaning that variances are free to take any value for each combination of the variables "batch" and "copy number". Alternatives such as " ~ cn", i.e. variance proportional to the number of copies are allowed but slower to fit, and less stable numerically.

`model.disease` A formula that links the number of copies with the case/control status. The default is a logit linear trend model "~ cn". Note that this formula will only matter under the alternate hypothesis and has no effect under the null (model descriptions using the "strata" command are not allowed for this model).

`association.test.strata` Optional factor providing the strata when using a stratified test of association (typically, but not always, these are geographic regions of origins of the samples).

`beta.estimated` Optional. It is used if one wants to fit the model for a particular value of the log odds parameter beta (essentially if one is interested in the profile likelihood). In this case the disease model should be set to ' ~ 1' and the model to 'H1'. It will then provide the best model assuming the value of beta (the log odds ratio parameter) provided by the user.

`start.mean` Optional. A set of starting values for the means. Must be numeric and the size must match ncomp. This argument can also be a matrix if one wants to specify multiple starting points. When passing a matrix as argument, the number of columns should equal the number of components, and the number of rows must be greater than max(n.H0, n.H1). When in a row some numbers are missing, CNVtools will pick the starting points randomly (the default).

`start.var` Optional. A set of starting values for the variances. Must be numeric and the size must match ncomp. Can also be a matrix (see start.mean for details).

`control` A list of parameters that control the behavior of the fitting. min.freq is the minimum number of data points in a copy number class before the algorithm sets the frequency of this class to zero.In the presence of a very rare genotype group it might be useful to lower this threshold. Note, however, that estimating the variance if there are very few individuals in a class may not be possible, so setting options such as constant variances (i.e. model.var = ' ~1') might be sensible.

Value

`model.H0` The parameters for the best fit under H0.

`posterior.H0` The output dataframe with the estimate posterior distribution under H0 as well as the most likely call.

`status.H0` A character that describes the status of the fit under H0. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

`model.H1` The parameters for the best fit under H1.

`posterior.H1` The output dataframe with the estimate posterior distribution under H1

`status.H1` A character that describes the status of the fit under H1. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

apply.pca apply.lda

Examples

```
#Load data for CNV for two control cohorts
data(A112)
raw.signal <- as.matrix(A112[, -c(1,2)])
dimnames(raw.signal)[[1]] <- A112$subject

#Extract CNV signal using principal components
pca.signal <- apply.pca(raw.signal)

#Extract batch, sample and trait information
```

```
  batches <- factor(A112$cohort)
sample <- factor(A112$subject)
trait <- ifelse( A112$cohort == '58C', 0, 1)

#Fit the CNV with a three component model
fit.pca <- CNVtest.binary(signal = pca.signal, sample = sample, batch = batches,
              disease.status = trait, ncomp = 3, n.H0=3, n.H1=3,
  model.disease = "~ cn")

if(fit.pca[['status.H0']] == 'C' && fit.pca[['status.H1']] == 'C'){
   #Calculate the likelihood ratio
   LR <- -2*(fit.pca$model.H0$lnL - fit.pca$model.H1$lnL)

   #Calculate the pvalue. Has 1 dof since we fit a trend model
   pvalue <- 1 - pchisq(LR,1)
}
```

`CNVtest.binary.T` *CNV association testing using T distributions*

Description

Test for CNV association with binary trait (typically case control) using a mixture of T distributions.

Usage

```
CNVtest.binary.T(signal,batch, sample = NULL, disease.status = NULL,
ncomp, n.H0 = 5, n.H1 = 0, output = "compact",
model.mean = "~ strata(batch, cn)",
model.var = "~ strata(batch, cn)",
model.disease = "~ cn",
beta.estimated = NULL,
start.mean = NULL,
start.var = NULL,
control = list(tol = 1e-05, max.iter = 3000, min.freq = 4))
```

Arguments

`signal`	The vector of intensity values, meant to be a proxy for the number of copies.
`batch`	Factor, that describes how the data points should be separated in batches, corresponding to different tehnologies to measure the number of DNA copies, or maybe different cohorts in a case control framework.
`sample`	Optional (but recommended). A character vector containing a name for each data point, typically the name of the individuals.
`disease.status`	In the case control situation a vector of 0 and 1 indicating which individuals are controls or cases.
`ncomp`	Number of components one wants to fit to the data.

`n.H0` Number of times the EM should be used to maximize the likelihood under the null hypothesis of no association, each time with a different random starting point. The run that maximizes the likelihood is stored.

`n.H1` Number of times the EM should be used to maximize the likelihood under the alternate hypothesis of association present, each time with a different random starting point. The run that maximizes the likelihood is stored.

`output` The default value, "compact", returns a data frame with one line per sample. Any other setting witll return a much bigger data frame with one line per individual and copy number. This long format is the one used by the underlying fitting algorithm and is only useful if one attempts to use CNVtools in a non standard manner.

`model.mean` Formula that relates the location of the means for the clusters with the number of copies and the different batches if there are multiple batches. Should be on the following: "~strata(cn)" or "strata(batch, cn)".

`model.var` A formula describing the variance model, as above. The default is the free variance model "~ strata(cn, batch)" but could also be "~ 1", "~ strata(cn)" or "~ strata(batch)".

`model.disease` A formula that relates the number of copies with the case/control status. The default is a linear trend model "~ cn". Note that this formula will only matter under the alternate hypothesis and has no effect under the null.

`beta.estimated` Optional. It is used if one wants to fit the model for a particular value of the log odds parameter beta (essentially if one is interested in the profile likelihood). In this case the disease model should be set to ' ~ 1' and the model to 'H1'. It will then provide the best model assuming the value of beta (the log odds ratio parameter) provided by the user.

`start.mean` Optional. A set of starting values for the means. Must be numeric and the size must match ncomp. This argument can also be a matrix if one wants to specify multiple starting points. When passing a matrix as argument, the number of columns should equal the number of components, and the number of rows must be greater than max(n.H0, n.H1). When in a row some numbers are missing, CNVtools will pick the starting points randomly (the default).

`start.var` Optional. A set of starting values for the variances. Must be numeric and the size must match ncomp. Can also be a matrix (see start.mean for details).

`control` A list of parameters that control the behavior of the fitting. min.freq is the minimum number of data points in a copy number class before the algorithm sets the frequency of this class to zero. In the presence of a very rare genotype group it might be useful to lower this threshold. Note, however, that estimating the variance if there are very few individuals in a class may not be possible, so setting options such as constant variances (i.e. model.var = ' ~1') might be sensible.

Value

`model.H0` The parameters for the best fit under H0.

`posterior.H0` The output dataframe with the estimate posterior distribution under H0 as well as the most likely call.

status.H0	A character that describes the status of the fit under H0. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.
model.H1	The parameters for the best fit under H1.
posterior.H1	The output dataframe with the estimate posterior distribution under H1
status.H1	A character that describes the status of the fit under H1. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

Author(s)

Vincent Plagnol and Chris Barnes

References

Finite Mixture Models (Wiley Series in Probability and Statistics), G. Mc Lachlan and David Peel

See Also

CNVtest.binary

CNVtest.qt *Fits a mixture of Gaussian to CNV data*

Description

This function fits a mixture of Gaussians to Copy Number Variant data to explore potential correlations between the copy number and a quantitative trait.

Usage

```
CNVtest.qt(signal, batch, sample = NULL, qt = NULL, ncomp, n.H0=5, n.H1=0,
   model.mean = '~ strata(cn)',
           model.var = '~ strata(cn)',
   model.qt = '~ cn',
           beta.estimated = NULL,
           start.mean = NULL,
           start.var = NULL,
   control=list(tol=1e-5, max.iter = 3000, min.freq=4) )
```

Arguments

signal	The vector of intensity values, meant to be a proxy for the number of copies.
batch	Factor, that describes how the data points should be separated in batches, corresponding to different tehnologies to measure the number of DNA copies, or maybe different cohorts in a case control framework.

`sample` Optional (but recommended). A character vector containing a name for each data point, typically the name of the individuals.

`qt` Quantitative trait values.

`ncomp` Number of components one wants to fit to the data.

`n.H0` Number of times the EM should be used to maximize the likelihood under the null hypothesis of no association, each time with a different random starting point. The run that maximizes the likelihood is stored.

`n.H1` Number of times the EM should be used to maximize the likelihood under the alternate hypothesis of association present, each time with a different random starting point. The run that maximizes the likelihood is stored.

`model.mean` Formula that relates the location of the means for the clusters with the number of copies and the different batches if there are multiple batches. The default is "~ strata(cn)" that assumes a free model for the cluster locations for each copy number." ~ strata(cn, batch)" assumes free variances for each combination of copy number and batch. More traditional model specifications such as ' ~ cn' are also possible, but will converge more slowly and might have numerical stability issues.

`model.var` A formula as above, but to model the variances. The default is the free variance model for each copy number "~ strata(cn)" and the same model specifications as model.means can be used.

`model.qt` A formula that relates the number of copies with the case/control status. The default is a linear trend model "~ cn". Note that this formula will only matter under the alternate hypothesis and has no effect under the null.

`beta.estimated` Optional. It is used if one wants to fit the model for a particular value of the log odds parameter beta (essentially if one is interested in the profile likelihood). In this case the disease model should be set to ' ~ 1' and the model to 'H1'. It will then provide the best model assuming the value of beta (the log odds ratio parameter) provided by the user.

`start.mean` Optional. A set of starting values for the means. Must be numeric and the size must match ncomp.

`start.var` Optional. A set of starting values for the variances. Must be numeric and the size must match ncomp.

`control` A list of parameters that control the behavior of the fitting.

Value

`model.H0` The parameters for the best fit under H0.

`posterior.H0` The output dataframe with the estimate posterior distribution under H0 as well as the most likely call.

`status.H0` A character that describes the status of the fit under H0. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

`model.H1` The parameters for the best fit under H1.

`posterior.H1` The output dataframe with the estimate posterior distribution under H1

status.H1 A character that describes the status of the fit under H1. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

apply.pca apply.ldf

Examples

```
#Load data for CNV for two control cohorts
data(A112)
raw.signal <- as.matrix(A112[, -c(1,2)])
dimnames(raw.signal)[[1]] <- A112$subject

#Extract CNV signal using principal components
pca.signal <- apply.pca(raw.signal)

#Extract batch, sample
sample <- factor(A112$subject)
batches <- rep("ALL",length(sample))

#Create a fake quantitative trait
trait <- rnorm(length(sample),mean=9.0,sd=1.0)

#Fit the CNV with a three component model
fit.pca <- CNVtest.qt(signal = pca.signal, sample = sample, batch = batches,
          qt = trait, ncomp = 3, n.H0=3, n.H1=3,
      model.qt = "~ cn")

if(fit.pca[['status.H0']] == 'C' && fit.pca[['status.H1']] == 'C'){
   #Calculate the likelihood ratio
   LR <- -2*(fit.pca$model.H0$lnL - fit.pca$model.H1$lnL)

   #Calculate the pvalue. Has 1 dof since we fit a trend model
   pvalue <- 1 - pchisq(LR,1)
}
```

CNVtest.qt.T *Fits a mixture of Gaussian to CNV data*

Description

This function fits a mixture of T distributions to Copy Number Variant data to explore potential correlations between the copy number and a quantitative trait.

Usage

```
CNVtest.qt.T(signal, batch, sample = NULL, qt = NULL, ncomp, n.H0=5, n.H1=0,
   model.mean = '~ strata(cn)',
           model.var = '~ strata(cn)',
   model.qt = '~ cn',
           beta.estimated = NULL,
           start.mean = NULL,
           start.var = NULL,
   control=list(tol=1e-5, max.iter = 3000, min.freq=4) )
```

Arguments

signal	The vector of intensity values, meant to be a proxy for the number of copies.
batch	Factor, that describes how the data points should be separated in batches, corresponding to different tehnologies to measure the number of DNA copies, or maybe different cohorts in a case control framework.
sample	Optional (but recommended). A character vector containing a name for each data point, typically the name of the individuals.
qt	Quantitative trait values.
ncomp	Number of components one wants to fit to the data.
n.H0	Number of times the EM should be used to maximize the likelihood under the null hypothesis of no association, each time with a different random starting point. The run that maximizes the likelihood is stored.
n.H1	Number of times the EM should be used to maximize the likelihood under the alternate hypothesis of association present, each time with a different random starting point. The run that maximizes the likelihood is stored.
model.mean	Formula that relates the location of the means for the clusters with the number of copies and the different batches if there are multiple batches. The default is "~ strata(cn)" that assumes a free model for the cluster locations for each copy number. For this T distribution model there is only one alternative: " ~ strata(cn, batch)" assumes free variances for each combination of copy number and batch.
model.var	A formula as above, but to model the variances. The default is the free variance model for each copy number "~ strata(cn)". There are three alternative variance models for this T distribution model: "~ strata(cn,batch)", "~ strata(batch)" or even " ~ 1" (constant variances for all batches and components).
model.qt	A formula that relates the number of copies with the case/control status. The default is a linear trend model "~ cn". Note that this formula will only matter under the alternate hypothesis and has no effect under the null.
beta.estimated	Optional. It is used if one wants to fit the model for a particular value of the log odds parameter beta (essentially if one is interested in the profile likelihood). In this case the disease model should be set to ' ~ 1' and the model to 'H1'. It will then provide the best model assuming the value of beta (the log odds ratio parameter) provided by the user.
start.mean	Optional. A set of starting values for the means. Must be numeric and the size must match ncomp.

`start.var`	Optional. A set of starting values for the variances. Must be numeric and the size must match ncomp.
`control`	A list of parameters that control the behavior of the fitting.

Value

`model.H0`	The parameters for the best fit under H0.
`posterior.H0`	The output dataframe with the estimate posterior distribution under H0 as well as the most likely call.
`status.H0`	A character that describes the status of the fit under H0. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.
`model.H1`	The parameters for the best fit under H1.
`posterior.H1`	The output dataframe with the estimate posterior distribution under H1
`status.H1`	A character that describes the status of the fit under H1. The possible values are 'C' (converged), 'M' (maximum iterations reached), 'P' (posterior distribution problem). Fits that don't return 'C' should be excluded.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

apply.pca apply.ldf

`CNVtest.select.model` *Select number of components in a CNV*

Description

This function fits mixtures of Gaussians to Copy Number Variant data, and uses the Bayesian Information Criteria (BIC) to select the most likely number of components.

Usage

```
CNVtest.select.model(signal, batch, sample = NULL, n.H0 = 3,
     method="BIC",
     v.ncomp = 1:6,
                    v.model.component = rep('gaussian',6),
                    v.model.mean = rep("~ strata(cn)",6),
                    v.model.var = rep("~1", 6),
     control=list(tol=1e-5, max.iter = 500, min.freq=4) )
```

Arguments

`signal`	The vector of intensity values, meant to be a proxy for the number of copies.
`batch`	Factor, that describes how the data points should be separated in batches, corresponding to different tehnologies to measure the number of DNA copies, or maybe different cohorts in a case control framework.
`sample`	Character vector containing a name for each data point, typically the name of the individuals.
`n.H0`	Number of times the EM should be used to maximize the likelihood and calculate the BIC for each different model.
`v.ncomp`	Model specification. Numeric vector specifying number of components to attempt. See discussion.
`v.model.component`	Character vector defining the mixture model. Can either be 'T' or 'gaussian'.
`v.model.mean`	Model specification. Character vector specifying different models for the component means. See discussion.
`v.model.var`	Model specification. Character vector specifying different models for the component means. See discussion.
`method`	Either BIC or AIC
`control`	A list of parameters that control the behavior of the fitting.

Details

The function fits the different models, specified by the vectors v.ncomp, v.model.mean, v.model.var, to the data contained in signal. The lengths of v.ncomp, v.model.mean, v.model.var must be equal. The function iterates through the length of these vectors and fits the models n.H0 times, keeping the fit with the highest likelihood. The BIC and AIC is calculated for each model, the lowest BIC/AIC indicates the 'best' model.

In the default model specification, first the data is fit with 1 component, mean model = "~ 1" and variance model = "~ 1". Next the data is fit with 2 components, mean model = "~ as.factor(cn)" and variance model "~ 1" etc.

Value

A data structure containing information from the fitting of the different models specified.

`model`	A list (length = number of model fit) containing the models specified by the user.
`BIC`	A vector (length = number of model fit) containing the BIC values for each model.
`AIC`	A vector (length = number of model fit) containing the AIC values for each model.
`status`	A vector (length = number of model fit) containing the status of every fit of every model.
`np`	A vector (length = number of model fit) containing the number of parameters of each model.

posteriors	A list (length = number of model fit) containing the best posterior distribution number of each model.
selected	The number of the best model. This will be the model that has the lowest BIC or AIC depending on which method was specified.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

References

Schwarz, G. (1978) "Estimating the Dimension of a Model", Annals of Statistics, 6, 461-464.

Examples

```
#Load data for CNV for two control cohorts
data(A112)
raw.signal <- as.matrix(A112[, -c(1,2)])
dimnames(raw.signal)[[1]] <- A112$subject

#Extract CNV signal using principal components
pca.signal <- apply.pca(raw.signal)

#Extract batch, sample and trait information
  batches <- factor(A112$cohort)
sample <- factor(A112$subject)

results <- CNVtest.select.model(signal = pca.signal, batch = batches,
                 sample = sample, n.H0 = 3)

        # Best model - with the default model setting this is also
# the number of components
best_model <- results$selected

# Look at the fit
cnv.plot( results[['posteriors']][[best_model]] )
```

---

compact.data.frame *Compacts the expanded data frame format needed by our fitting procedure into more compact and user friendly version*

Description

Small internal routine returning a more compact and user friendly version of the output of the fitting algorithm.

Usage

```
compact.data.frame(full.frame)
```

Arguments

`full.frame`	An expanded data frame (one point per data point and per component in the fit, ie. 1,000 individuals fitted on three components would have 3,000 rows.

Details

This function should be invisible to most users and is part of the EM fitting procedure.

Value

A data frame in a compact version, with one row per data point and one column for each component: P1, P2, P3 in the three component case for the probabilities for the calls to be equal to 1,2 or 3.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

`EM.starting.point` *Randomly assigns a starting point for the EM algorithm*

Description

This function should be invisible to most users, and is part of our the fitting routine using the EM algorithm. Our maximum likelihood procedure uses an iterative algorithm called Expectation-Maximization. This requires a starting point, chosen at random. EM.starting point randomly assigns this starting point.

Usage

```
EM.starting.point(d, trait = "binary")
```

Arguments

`d`	The dataframe that needs to be initialized
`trait`	Can be either “binary” or ”eQTL”

Value

Returns the input data frame with reasonable random starting values.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

References

A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J. Royal Stat. Soc., vol. 39, pp. 1–38, 1977.

`ExpandData` *Expands a CNV input data frame for the maximum likelihood routines*

Description

This function should be invisible to most users. The methods within CNV.fitModel require that the CNV data is expanded N times where N is the number of copies. This allows the use of Generalized Linear Models (GLM) in constraining the Gaussian mixture component locations and spreads to be functions of the copy number.

Usage

```
ExpandData(batch, trait, names, signal, ncomp, association.strata = NULL)
```

Arguments

`batch`	List of vectors, one vector per batch in the data. Because each element in the list corresponds to a batch, each element should be a vector with a unique values repeated as many times as the number of data point in the batch.
`trait`	List of vectors, one vector per batch in the data. Each element of the list can be either a vector of quantitative traits or a vector of 0 and 1 in a case/control framework
`names`	List of vectors, one vector per batch in the data containing names for each data point, typically individual IDs.
`signal`	List, one vector per batch in the data.
`ncomp`	Integer, number of components one wants to fit to the data
`association.strata`	Optional, a factor vector containing the strata when using a stratified test of association.

Value

An expanded data frame needed for CNVfit.binary.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

See Also

CNVtest.binary

get.model.spec — *Get model specifications (internal function)*

Description

Internal function to parse the formulas and extract codes for the model, as well as number of parameters

Usage

```
get.model.spec(model.component, model.mean, model.var, model.nu, design.matrix.mean, design.matrix.v
```

Arguments

`model.component`

`model.mean`

`model.var`

`model.nu`

`design.matrix.mean`

`design.matrix.variance`

`ncomp`

`nbatch`

Value

A list with two components: model code and number of parameters of the model.

Author(s)

Vincent Plagnol and Chris Barnes

getparams — *Return mixture parameters*

Description

This function should be invisible to most users. Given the full expanded data frame, getparams returns the number of components, the copy number, the mixture model parameters for each batch, the likelihood of the model and the p(disease|c).

Usage

```
getparams(d)
```

Arguments

d	Full expanded data frame.

Value

ns	Number of batches
nc	Copy number of this model
nind	Number of individuals
lnL	log likelihood
alpha	Matrix. The mixture proportions
mean	Matrix. The mixture means
var	Matrix. The mixture variances
pdc	Matrix. The p(disease|c)

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

---

getQualityScore *Computes a quality score for a CNV fit*

Description

The quality scores measures how well the clusters are separated. It ocmpares the locations of the means with the standard error for each pair of adjacent cluster. A quality score greater than 4 is usually good enough for association studies.

Usage

```
getQualityScore(posterior)
```

Arguments

posterior	A data frame generated by CNVtest.binary

Value

One number, the quality score.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

qt.plot *Makes signal vs trait plots and posterior probabilty distributions*

Description

Makes signal vs trait and formatted density plots from the data frame returned by CNVtest.qt

Usage

```
qt.plot(DataFrame.list, main='', hist.or.dens='histogram')
```

Arguments

DataFrame.list	The output obtained from the CNVtools fitting algorithm CNVtest.qt
main	Potential title for the graph
hist.or.dens	Either 'histogram' or 'density' to plot the data as an histogram or using a kernel density estimator

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

Examples

```
#Load data for CNV for two control cohorts
data(A112)
raw.signal <- as.matrix(A112[, -c(1,2)])
dimnames(raw.signal)[[1]] <- A112$subject

#Extract CNV signal using principal components
pca.signal <- apply.pca(raw.signal)

#Extract batch, sample
sample <- factor(A112$subject)
batches <- rep("ALL",length(sample))

#Create a fake quantitative trait
trait <- rnorm(length(sample),mean=9.0,sd=1.0)

#Fit the CNV with a three component model
fit.pca <- CNVtest.qt(signal = pca.signal, sample = sample, batch = batches,
          qt = trait, ncomp = 3, n.H0=3, n.H1=3,
      model.qt = "~ cn")

qt.plot(fit.pca)
```

`test.posterior` *Checks posterior probabilities are monotonic.*

Description

The posterior probability of belonging to a particular component should fall to zero monotonically as the signal increases or decreases away from the component mean. This function checks for posterior distributions that do not have this property.

Usage

```
test.posterior(frame, ncomp, samples.by.disease = NULL)
```

Arguments

`frame`	Posterior data frame.
`ncomp`	Number of components.
`samples.by.disease`	List containing samples split on disease status.

Value

Returns TRUE is the posterior is not monotonic.

Author(s)

Vincent Plagnol <vincent.plagnol@cimr.cam.ac.uk> and Chris Barnes <christopher.barnes@imperial.ac.uk>

Index

∗Topic **cluster**
- apply.ldf, 3
- apply.pca, 4
- CNV.fitModel, 5
- CNVtest.binary, 7
- CNVtest.binary.T, 10
- CNVtest.qt, 12
- CNVtest.qt.T, 14
- CNVtest.select.model, 16
- CNVtools-package, 2
- getQualityScore, 22

∗Topic **htest**
- CNVtest.binary, 7
- CNVtest.binary.T, 10
- CNVtest.qt, 12
- CNVtest.qt.T, 14
- CNVtest.select.model, 16
- CNVtools-package, 2

∗Topic **regression**
- CNVtest.binary, 7
- CNVtest.binary.T, 10

A112, 3
apply.ldf, 3
apply.pca, 4

CNV.fitModel, 5
cnv.plot, 6
CNVtest.binary, 7
CNVtest.binary.T, 10
CNVtest.qt, 12
CNVtest.qt.T, 14
CNVtest.select.model, 16
CNVtools *(*CNVtools-package*)*, 2
CNVtools-package, 2
compact.data.frame, 18

EM.starting.point, 19
ExpandData, 20

get.model.spec, 21
getparams, 21
getQualityScore, 22

qt.plot, 23

test.posterior, 24