Package 'CGHcall'

October 24, 2025

Type Package

Title Calling aberrations for array CGH tumor profiles.

Version 2.71.0

Date 2016-09-12

Author Mark van de Wiel, Sjoerd Vosse

Maintainer Mark van de Wiel <mark.vdwiel@vumc.nl>

Depends R (>= 2.0.0), impute(>= 1.8.0), DNAcopy (>= 1.6.0), methods, Biobase, CGHbase (>= 1.15.1), snowfall

Description
Calls aberrations for array CGH data using a six state mixture model as well as several biological concepts that are ignored by existing algorithms. Visualization of profiles is also provided.

License GPL (http://www.gnu.org/copyleft/gpl.html)

biocViews Microarray,Preprocessing,Visualization

git_url https://git.bioconductor.org/packages/CGHcall

git_branch devel

git_last_commit c5cb51d

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-10-24

Contents

CGHcall-package 2
CGHcall 2
ExpandCGHcall 4
normalize 6
postsegnormalize 7
preprocess 8
segmentData 9
Wilting 10

Index **11**

CGHcall-package *Calling aberrations for array CGH tumor profiles.*

Description

Calls aberrations for array CGH data using a six state mixture model as well as several biological concepts that are ignored by existing algorithms. Visualization of profiles is also provided.

Details

Package:	CGHcall
Type:	Package
Version:	2.34.1
Date:	2016-09-12
License:	GPL

Author(s)

Sjoerd Vosse and Mark van de Wiel

Maintainer: Mark van de Wiel <mark.vdwiel@vumc.nl>

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. *Bioinformatics, 23*, 892-894.

CGHcall *Calling aberrations for array CGH tumor profiles.*

Description

Calls aberrations for array CGH data using a six state mixture model.

Usage

```
CGHcall(inputSegmented, prior = "auto", nclass = 5, organism = "human", cellularity=1, robustsig="ye
```

Arguments

inputSegmented	An object of class cghSeg
prior	Options are all, not all, or auto. See details for more information.
nclass	The number of levels to be used for calling. Either 3 (loss, normal, gain), 4 (including amplifications), 5 (including double deletions).
organism	Either human or other. This is only used for chromosome arm information when prior is set to all or auto (and samplesize > 20).
cellularity	A vector of cellularities ranging from 0 to 1 to define the contamination of your sample with healthy cells (1 = no contamination). See details for more information.
robustsig	Options are yes or no. yes enforces a lower bound on the standard deviation of the normal segments
nsegfit	Maximum number of segments used for fitting the mixture model. Posterior probabilities are computed for all segments
maxnumseg	Maximum number of segments per profile used for fitting the model
minlsforfit	Minimum length of the segment (in Mb) to be used for fitting the model
build	Build of Humane Genome. Either GRCh37, GRCh36, GRCh35 or GRCh34.
ncpus	Number of cpus used for parallel calling. Has a large effect on computing time. ncpus larger than 1 requires package snowfall.

Details

Please read the article and the supplementary information for detailed information on the algorithm. The parameter prior states how the data is used to determine the prior probabilities. When set to all, the probabilities are determined using the entire genome of each sample. When set to not all probabilites are determined per chromosome for each sample when organism is set to other or per chromosome arm when organism is human. The chromosome arm information is taken from the March 2006 version of the UCSC database. When prior is set to auto, the way probabilities are determined depends on the sample size. The entire genome is used when the sample size is smaller than 20, otherwise chromosome (arm) information is used. Please note that CGHcall uses information from all input data to determine the aberration probabilities. When for example triploid or tetraploid tumors are observed, we advise to run CGHcall separately on those (groups of) samples. Note that robustsig = yes enforces the sd corresponding to the normal segments to be at least half times the pooled gain/loss sd. Use of nsegfit significantly lower computing time with respect to previous CGHcall versions without much accuracy loss. Moreover, maxnumseg decreases the impact on the results of profiles with inferior segmentation results. Finally, minlsforfit decreases the impact of very small aberations (potentially CNVs rather than CNAs) on the fit of the model. Note that always a result for all segments is produced. IN MOST CASES, CGHcall SHOULD BE FOLLOWED BY FUNCTION ExpandCGHcall.

Value

This function return a list with six components:

posteriorfin2	Matrix containing call probabilities for each segment. First column denotes profile number, followed by k columns with aberration probabilities for each sample, where k is the number of levels used for calling (nclass).
nclone	Number of clone or probes
nc	Number of samples

nclass	Number of classes used
regionsprof	Matrix containing information about the segments, 4 colums: profile, start probe, end probe, segmented value
params	Vector containing the parameter values of the mixture model

Author(s)

Sjoerd Vosse, Mark van de Wiel, Ilari Scheinin

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. *Bioinformatics, 23*, 892-894.

See Also

ExpandCGHcall

Examples

```
data(Wilting)
## Convert to \code{\link{cghRaw}} object
cgh <- make_cghRaw(Wilting)
print(cgh)
## First preprocess the data
raw.data <- preprocess(cgh)
## Simple global median normalization for samples with 75% tumor cells
normalized.data <- normalize(raw.data)
## Segmentation with slightly relaxed significance level to accept change-points.
## Note that segmentation can take a long time.
## Not run: segmented.data <- segmentData(normalized.data, alpha=0.02)
## Not run: postsegnormalized.data <- postsegnormalize(segmented.data)
## Call aberrations
perc.tumor <- rep(0.75, 3)
## Not run: result <- CGHcall(postsegnormalized.data,cellularity=perc.tumor)

## Expand to CGHcall object
## Not run: result <- ExpandCGHcall(result,postsegnormalized.data)
```

ExpandCGHcall *Expands result fron CGHcall to CGHcall object.*

Description

Expands result from CGHcall function to CGHcall object.

Usage

```
ExpandCGHcall(listcall,inputSegmented, digits=3, divide=4, memeff = FALSE, fileoutpre="Callobj_",C
```

Arguments

listcall	List object; output of function CGHcall
inputSegmented	An object of class cghSeg
digits	Number of decimal digits to be saved in the resulting call object. Allows for saving storage space
divide	Number of batches to divide the work load in. Larger values saves memory, but requires more computing time
memeff	When set to TRUE, memory efficient mode is used: results are written in batches to multiple external files. If FALSE, one output object is provided.
fileoutpre	Only relevant when memeff=TRUE. Define prefix for output file names
CellularityCorrectSeg	If TRUE, corrects segmented and normalized values for cellularity as well

Details

This function is new in version 2.7.0. It allows more memory efficient handling of large data objects. If R crashes because of memory problem, we advise to set memeff = TRUE and increase the value of divide. When multiple files are output (in case of memeff=TRUE) the function combine may be used to combine CGHcall objects.

Value

An object of class cghCall-class either as one object (when memeff = FALSE) or as multiple objects stored in .Rdata files in the working directory (when memeff = FALSE)

Author(s)

Sjoerd Vosse & Mark van de Wiel

References

Mark A. van de Wiel, Kyung In Kim, Sjoerd J. Vosse, Wessel N. van Wieringen, Saskia M. Wilting and Bauke Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. *Bioinformatics, 23*, 892-894.

See Also

CGHcall, cghCall-class

Examples

```
data(Wilting)
## Convert to \code{\link{cghRaw}} object
cgh <- make_cghRaw(Wilting)
print(cgh)
## First preprocess the data
raw.data <- preprocess(cgh)
## Simple global median normalization for samples with 75% tumor cells
perc.tumor <- rep(0.75, 3)
normalized.data <- normalize(raw.data)
## Segmentation with slightly relaxed significance level to accept change-points.
## Note that segmentation can take a long time.
```

```
## Not run: segmented.data <- segmentData(normalized.data, alpha=0.02)
## Not run: postsegnormalized.data <- postsegnormalize(segmented.data)
## Call aberrations
## Not run: result <- CGHcall(postsegnormalized.data, cellularity=perc.tumor)
## Not run: result <- ExpandCGHcall(result,postsegnormalized.data)
```

`normalize` *Normalization and cellularity adjustment for arrayCGH data.*

Description

This function normalizes arrayCGH data using the global mode or median. It can also adjust for the cellularity of your data.

Usage

```
normalize(input, method = "median", smoothOutliers = TRUE, ...)
```

Arguments

`input`	Object of class `cghRaw`.
`method`	Normalization method, either `median`, `mode`, or `none`.
`smoothOutliers`	Logical. Indicates whether outliers should be smoothed using the `smooth.CNA` function.
`...`	Arguments for `smooth.CNA`.

Details

The cellularity parameter should be a vector of length n where n is the number of samples in your dataset. The vector is recycled if there are not enough values in it, or truncated if there are too many. For more information on the correction we refer to section 1.6 of the supplementary information for van de Wiel et al. 2006.

Value

This function returns a dataframe in the same format as the input with normalized and/or cellularity adjusted log2 ratios.

Author(s)

Sjoerd Vosse & Mark van de Wiel

Examples

```
data(Wilting)
## Convert to 'cghRaw' object
cgh <- make_cghRaw(Wilting)
## First preprocess the data
raw.data <- preprocess(cgh)
## Simple global median normalization for samples with 75% tumor cells
normalized.data <- normalize(raw.data)
```

postsegnormalize *Post-segmentation normalization*

Description

This function normalizes arrayCGH data after segmentation in order to find a better 0-level.

Usage

```
postsegnormalize(segmentData, inter=c(-0.1,0.1))
```

Arguments

segmentData	Object of class cghSeg.
inter	Interval in which the function should search for the normal level.

Details

This function recursively searches for the interval containing the most segmented data, decreasing the interval length in each recursion. The recursive search makes the post-segmentation normalization robust against local maxima. This function is particularly useful for profiles for which, after segmentation, the 0-level does not coincide with many segments. It is more or less harmless to other profiles. We advise to keep the search interval (inter) small, in particular at the positive (gain) side to avoid that the 0-level is set to a common gain level.

Value

This function returns a cghSeg object in the same format as the input with post-segmentation-normalized adjusted log2 ratios and segmented values.

Author(s)

Mark van de Wiel

Examples

```
data(Wilting)
## Convert to \code{\link{cghRaw}} object
cgh <- make_cghRaw(Wilting)
## First preprocess the data
raw.data <- preprocess(cgh)
## Simple global median normalization for samples with 75% tumor cells
normalized.data <- normalize(raw.data)
## Segmentation with slightly relaxed significance level to accept change-points.
## Note that segmentation can take a long time.
## Not run: segmented.data <- segmentData(normalized.data, alpha=0.02)
## Not run: postsegnormalized.data <- postsegnormalize(segmented.data, inter=c(-0.1,0.1))
```

`preprocess` *Preprocess arrayCGH data*

Description

This function preprocesses your aCGH data so it can be processed by other functions without errors.

Usage

```
preprocess(input, maxmiss = 30, nchrom = 23, ...)
```

Arguments

`input`	Object of class `cghRaw`.
`maxmiss`	Maximum percentage of missing values per row.
`nchrom`	Number of chromosomes.
`...`	Arguments for `impute.knn` from the impute package.

Details

This function performs the following actions on arrayCGH data:

- Filter out data with missing position information.
- Remove data on chromosomes larger than nchrom.
- Remove rows with more than maxmiss percentage missing values.
- Imputes missing values using the `impute.knn` function from the impute package.

Value

This function returns a dataframe in the same format as the input with missing values imputed.

Author(s)

Sjoerd Vosse & Mark van de Wiel

References

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics, 17*, 520-525.

Examples

```
data(WiltingRaw)
preprocessed <- preprocess(WiltingRaw, nchrom = 22)
```

segmentData *Breakpoint detection for arrayCGH data.*

Description

A wrapper function to run existing breakpoint detection algorithms on arrayCGH data. Currently only DNAcopy is implemented.

Usage

```
segmentData(input, clen=10, relSDlong=3, method = "DNAcopy", ...)
```

Arguments

input	Object of class cghRaw.
clen	Boundary for short vs long segments, in number of features
relSDlong	Relative undo sd for long segments. See details.
method	The method to be used for breakpoint detection. Currently only DNAcopy is supported, which will run the segment function.
...	Arguments for segment.

Details

See segment for details on the algorithm. About clen and relSDlong: these are only relevant when segment option undo.splits=sdundo is set, in combination with segment option undo.SD. relSDlong provides the undo sd for long segments, which equals undo.SD/relSDlong. undo.SD is then used for short segments. In the example below, short segments are considered to contain less or equal to clen=10 features. The example below undoes splits for two consecutive short segments if these are less than undo.SD=3 sd apart, while it undoes splits for two long segments if these are less than undo.SD/relSDlong=3/3=1 sd apart. If, for two consecutive segements, one is short and one is long, splits are undone in the same way as for two short segments.

Value

This function returns a dataframe in the same format as the input with segmented arrayCGH data.

Author(s)

Sjoerd Vosse & Mark van de Wiel

References

Venkatraman, A.S., Olshen, A.B. (2007). A faster circulary binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics, 23*, 657-663.

Examples

```
 data(WiltingNorm)
## Not run: segmented.data <- segmentData(WiltingNorm, alpha=0.02,clen=10,relSDlong=3,undo.SD=3,undo.splits
```

`Wilting` *Cervical cancer arrayCGH data*

Description

A dataframe containing 4709 rows and 8 columns with arrayCGH data.

Usage

```
Wilting
```

Format

A dataframe containing the following 8 columns:

Name The unique identifiers of array elements.

Chromosome Chromosome number of each array element.

Position Chromosomal position in bp of each array element.

AdCA10 Raw log2 ratios for cervical cancer sample AdCA10.

SCC27 Raw log2 ratios for cervical cancer sample SCC27.

SCC32 Raw log2 ratios for cervical cancer sample SCC32.

SCC36 Raw log2 ratios for cervical cancer sample SCC36.

SCC39 Raw log2 ratios for cervical cancer sample SCC39.

Source

Wilting, S.M., Snijders, P.J., Meijer, G.A., Ylstra, B., van den IJssel, P.R., Snijders, A.M., Albertson, D.G., Coffa, J., Schouten, J.P., van de Wiel, M.A., Meijer, C.J., & Steenbergen, R.D. (2006). Increased gene copy numbers at chromosome 20q are frequent in both squamous cell carcinomas and adenocarcinomas of the cervix. *Journal of Pathology, 210*, 258-259.

Index

∗ **datasets**
    Wilting, 10
∗ **misc**
    CGHcall, 2
    ExpandCGHcall, 4
    normalize, 6
    postsegnormalize, 7
    preprocess, 8
    segmentData, 9
∗ **package**
    CGHcall-package, 2

CGHcall, 2, *4, 5*
CGHcall-package, 2
cghRaw, *6*, *8, 9*
cghSeg, *3*, *5*, *7*

ExpandCGHcall, *4*, 4

impute.knn, *8*

normalize, 6

postsegnormalize, 7
preprocess, 8

segment, *9*
segmentData, 9
smooth.CNA, *6*

Wilting, 10