DNAcopy

October 5, 2010

CNA *Create ‘Copy Number Array’ data object*

Description

Creates a ‘copy number array’ data object used for DNA copy number analyses by programs such as circular binary segmentation (CBS).

Usage

```
  CNA(genomdat, chrom, maploc, data.type=c("logratio","binary"),
                 sampleid=NULL)
  ## S3 method for class 'CNA':
print(x, ...)
```

Arguments

`genomdat` a vector or matrix of data from array-CGH, ROMA, or other copy number experiments. If it is a matrix the rows correspond to the markers and the columns to the samples.

`chrom` the chromosomes (or other group identifier) from which the markers came. Vector of length same as the number of rows of genomdat. If one wants the chromosomes to be ordered in the natural order, this variable should be numeric or ordered category.

`maploc` the locations of marker on the genome. Vector of length same as the number of rows of genomdat. This has to be numeric.

`data.type` logratio (aCGH, ROMA, etc.) or binary (LOH).

`sampleid` sample identifier. If missing the samples are named by prefixing "Sample" to consecutive integers.

`x` object returned by CNA

`...` arguments to be passed onto print command called within.

Details

Data that are NA, Inf, NaN will be removed on a per sample basis for "genomdat" and all samples for "chrom" and "maploc".

Value

An object of class CNA. There is a print method that gives the number of samples and probes and the type of data.

Examples

```
data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))
```

DNAcopy *Results of segmenting a CNA data object*

Description

The results of segmenting data from copy number array experiments from programs such as circular binary segmentation (CBS).

Usage

```
## S3 method for class 'DNAcopy':
print(x, ...)
```

Arguments

x	an object of class DNAcopy – output of segment.
...	arguments to be passed onto print command called within.

Details

An object of class DNAcopy. There is a print method that prints the results in a tabular format. Each row gives the sample, the chromosome, the start and end map locations, the number of markers and the mean of each segment.

Value

data	The original CNA object which was the input for segment
ID	sample identifier.
chrom	the chromosome within the sample.
loc.start	the starting map location of the segment
loc.end	the ending map location of the segment
num.mark	the number of markers in the segment
data.type	the segment mean.
call	the call that produced the object.

`coriell` *Array CGH data set of Coriell cell lines*

Description

These are two data array CGH studies sets of Corriel cell lines taken from the reference below.

Usage

```
data(coriell)
```

Format

A data frame containing five variables: first is clone name, second is clone chromosome, third is clone position, fourth and fifth are log2ratio for two cell lines.

Source

http://www.nature.com/ng/journal/v29/n3/suppinfo/ng754_S1.html

References

Snijders et al., Assembly of microarrays for genome-wide measurement of DNA copy number, *Nature Genetics*, 2001

`cytoBand` *Cytogenic band data*

Description

Cytogenic band data from the godenPath repository

Usage

```
data(cytoBand)
```

Format

A data frame containing five variables: chromosome, start and end positions, band name and giesma stain.

Source

http://genome.cse.ucsc.edu/goldenPath/hg16/database/cytoBand.txt.gz

exon.segment *Binary segmentation of exon data.*

Description

Compute the binary segmentation statistic, location and approximate p-value.

Usage

```
exon.segment(gene, eloc, edat, ngrid=100, tol=1e-6)
```

Arguments

`gene`	gene names in the exon data
`eloc`	exon locations within gene
`edat`	exon expressions within gene
`ngrid`	number grid points for the integral
`tol`	tolerance level for calculating nu

Details

The p-values are obtained by applying Siegmund's approximation for the maximal statistic from binary segmenting consecutive segments within a chromosome. These are one-sided test for an increase in expression.

Value

a matrix with three columns. The maximal statistic from binary segmentation, its location and the p-values for each gene.

Author(s)

Venkatraman E. Seshan <ves2111@columbia.edu>

Examples

```
# test code on an easy data set
set.seed(25)
gene <- rep(c("A", "B"), c(30,20))
eloc <- c(1:30, 1:20)
edat <- matrix(rnorm(500), 50, 10)
# changes for gene1 in samples 3 & 7
edat[1:30, 3] <- edat[1:30, 3] + rep(0.9*0:1, c(17, 13))
edat[1:30, 7] <- edat[1:30, 7] + rep(1.1*0:1, c(21, 9))
# changes for gene2 in samples 4 & 7
edat[31:50, 4] <- edat[31:50, 4] + rep(1.1*0:1, c(8, 12))
edat[31:50, 7] <- edat[31:50, 7] + rep(1.2*0:1, c(13, 7))
exon.segment(gene, eloc, edat)
```

`getbdry` *Sequential stopping boundary*

Description

Function to compute the sequential boundary for early stopping.

Usage

```
getbdry(eta, nperm, max.ones, tol= 1e-2)
```

Arguments

`eta`	Type I error rate of the boundary.
`nperm`	Number of permutations for the reference distribution.
`max.ones`	maximum number of ones given by "floor(nperm*alpha)+1".
`tol`	tolerance level for the iterations.

Value

A vector integer values of length max.ones*(max.ones+1)/2 corresponding to the boundary for the number of ones from 1 to max.ones. The default boundary for nperm=10000, eta=0.05, alpha=0.01 is stored in the data object "default.DNAcopy.bdry". Use this function to get the boundary for your favorite values for the parameters "nperm, eta, alpha" and use it for the argument "sbdry" in the function "segment."

`glFrequency` *Additional summary measured for the segments*

Description

This program computes the frequency of gains and losses for each probe as a function of level of mad.

Usage

```
glFrequency(xout, threshold=1)
```

Arguments

`xout`	an object of class DNAcopy
`threshold`	threshold value to call gain or loss

Value

A segment is called a gain or loss if the segment mean is at least the threshold* mad distance away from the median copy number level. The output is a data frame with five columns which give the chromosome (chrom), genomic position (maploc), the number of samples with available data (pfreq), and the gain (gain) and loss (loss).

Author(s)

Venkatraman E. Seshan <ves2111@columbia.edu>

plot.DNAcopy *Plot the data and results from segment of a CNA object*

Description

Plots the data from a copy number array experiment (aCGH, ROMA etc.) along with the results of segmenting it into regions of equal copy numbers.

Usage

```
   ## S3 method for class 'DNAcopy':
plot(x, plot.type=c("whole", "plateau", "samplebychrom",
               "chrombysample"), xmaploc=FALSE, altcol=TRUE, sbyc.layout=
                NULL, cbys.nchrom=1, cbys.layout=NULL, include.means=TRUE,
                zeroline=TRUE, pt.pch=NULL, pt.cex=NULL, pt.cols=NULL,
                segcol= NULL, zlcol=NULL, ylim=NULL, lwd=NULL, ...)
```

Arguments

`x` an object of class `DNAcopy` resulting from analyzing data from copy number array experiments.

`plot.type` the type of plot.

`xmaploc` logical flag to indicate that the X axis is the maploc position rather than the index. Since the segments are rearranged the plateau plot does not use maploc position.

`altcol` logical flag to indicate if chromosomes should be plotted in alternating colors in the whole genome plot.

`sbyc.layout` `layout` settings for the multifigure grid layout for the ‘samplebychrom’ type. It should be specified as a vector of two integers which are the number of rows and columns. The default values are chosen based on the number of chromosomes to produce a near square graph. For normal genome it is 4x6 (24 chromosomes) plotted by rows.

`cbys.layout` `layout` settings for the multifigure grid layout for the ‘chrombysample’ type. As above it should be specified as number of rows and columns and the default chosen based on the number of samples.

`cbys.nchrom` the number of chromosomes per page in the layout. The default is 1.

`include.means` logical flag to indicate whether segment means are to be drawn.

`zeroline` logical flag to indicate whether a horizontal line at y=0 is to be drawn.

`pt.pch` the plotting character used for plotting the log-ratio values (default is ".").

`pt.cex` the size of plotting character used for the log-ratio values (default is 3).

`pt.cols` the color list for the points. The colors alternate between chromosomes. If missing the point colors are black and green.

`segcol` the color of the lines indicating the segment means. If missing the line color is set to be red.

`zlcol` the color of the zeroline. If missing it is set to be grey.

`ylim` this argument is present to override the default limits which is the range of symmetrized log-ratios.

`lwd` line weight of lines for segment mean and zeroline. If missing it is set to 3.

`...` other arguments which will be passed to `plot` commands.

Details

There are four possible plot types. For the type ‘whole’ the data are plotted for the entire genome. For the ‘samplebychrom’ type a graph with each chromosome (of a given sample) is drawn in a separate figure on a multi-figure grid. For the ‘plateau’ type the graph is drawn with the chromosome segments re-ordered by the segment means. For the ‘chrombysample’ type the samples for a given chromosome are drawn in a 4x6 multi-figure grid in multiples of 24. By default the segments means are drawn. For multisample data each sample or chromosome is drawn on a separate sheet. When invoked interactively the user is prompted before advancing to the next sample.

Examples

```
#Read in two examples from Snijders et al.

data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))

#We generally recommend smoothing single point outliers before analysis
#Make sure to check that the smoothing is proper

smoothed.CNA.object <- smooth.CNA(CNA.object)

#Segmentation at default parameters

segment.smoothed.CNA.object <- segment(smoothed.CNA.object, verbose=1)

#Plot whole studies

plot(segment.smoothed.CNA.object, plot.type="w")

#Plot each study by chromosome

plot(segment.smoothed.CNA.object, plot.type="s")

#Plot each chromosome across studies (6 per page)

plot(segment.smoothed.CNA.object, plot.type="c", cbys.layout=c(2,1), cbys.nchrom=6)

#Plot by plateaus

plot(segment.smoothed.CNA.object, plot.type="p")
```

`segment` *Genome Segmentation Program*

Description

This program segments DNA copy number data into regions of estimated equal copy number using circular binary segmentation (CBS).

Usage

```
segment(x, weights = NULL, alpha = 0.01, nperm = 10000, p.method =
                    c("hybrid", "perm"), min.width=2, kmax=25, nmin=200,
                    eta=0.05, sbdry=NULL, trim = 0.025, undo.splits =
                    c("none", "prune", "sdundo"), undo.prune=0.05,
                    undo.SD=3, verbose=1)
```

Arguments

`x` an object of class CNA

`weights` a vector of weights for the probes. The weights should be inversely proportional to their variances. Currently all weights should be positive i.e. remove probes with zero weight prior to segmentation.

`alpha` significance levels for the test to accept change-points.

`nperm` number of permutations used for p-value computation.

`p.method` method used for p-value computation. For the "perm" method the p-value is based on full permutation. For the "hybrid" method the maximum over the entire region is split into maximum of max over small segments and max over the rest. Approximation is used for the larger segment max. Default is hybrid.

`min.width` the minimum number of markers for a changed segment. The default is 2 but can be made larger. Maximum possible value is set at 5 since arbitrary widths can have the undesirable effect of incorrect change-points when a true signal of narrow widths exists.

`kmax` the maximum width of smaller segment for permutation in the hybrid method.

`nmin` the minimum length of data for which the approximation of maximum statistic is used under the hybrid method. should be larger than 4*kmax

`eta` the probability to declare a change conditioned on the permuted statistic exceeding the observed statistic exactly j (= 1,...,nperm*alpha) times.

`sbdry` the sequential boundary used to stop and declare a change. This boundary is a function of nperm, alpha and eta. It can be obtained using the function "getbdry" and used instead of having the "segment" function compute it every time it is called.

`trim` proportion of data to be trimmed for variance calculation for smoothing outliers and undoing splits based on SD.

`undo.splits` A character string specifying how change-points are to be undone, if at all. Default is "none". Other choices are "prune", which uses a sum of squares criterion, and "sdundo", which undoes splits that are not at least this many SDs apart.

`undo.prune`	the proportional increase in sum of squares allowed when eliminating splits if undo.splits="prune".
`undo.SD`	the number of SDs between means to keep a split if undo.splits="sdundo".
`verbose`	level of verbosity for monitoring the program's progress where 0 produces no printout, 1 prints the current sample, 2 the current chromosome and 3 the current segment. The default level is 1.

Details

This function implements the cicular binary segmentation (CBS) algorithm of Olshen and Venkatraman (2004). Given a set of genomic data, either continuous or binary, the algorithm recursively splits chromosomes into either two or three subsegments based on a maximum t-statistic. A reference distribution, used to decided whether or not to split, is estimated by permutation. Options are given to eliminate splits when the means of adjacent segments are not sufficiently far apart. Note that after the first split the α-levels of the tests for splitting are not unconditional.

We recommend using one of the undoing options to remove change-points detected due to local trends (see the manuscript below for examples of local trends).

Since the segmentation procedure uses a permutation reference distribution, R commands for setting and saving seeds should be used if the user wishes to reproduce the results.

Data that are NA, Inf, NaN will be removed on a per sample basis for "genomdat" and all samples for "chrom" and "maploc".

Value

An object of class `DNAcopy`. It has three elements:

`data`	The original CNA object which was the input for segment
`out`	a data frame with six columns. Each row of the data frame contains a segment for which there are six variables: the sample id, the chromosome number, the map position of the start of the segment, the map position of the end of the segment, the number of markers in the segment, and the average value in the segment.
`call`	the call that produced the output object.

Author(s)

E. S. Venkatraman and Adam Olshen `<olshena@mskcc.org>`

References

Olshen, A. B., Venkatraman, E. S., Lucito, R., Wigler, M. (2004). Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics* 5: 557-572. `http://www.mskcc.org/biostat/~olshena/research`.

Examples

```
# test code on an easy data set
set.seed(25)
genomdat <- rnorm(500, sd=0.1) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
plot(genomdat)
```

```
chrom <- rep(1:2,c(290,210))
maploc <- c(1:290,1:210)
test1 <- segment(CNA(genomdat, chrom, maploc))

# test code on a noisier and hence more difficult data set
set.seed(51)
genomdat <- rnorm(500, sd=0.2) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
plot(genomdat)
chrom <- rep(1:2,c(290,210))
maploc <- c(1:290,1:210)
test2 <- segment(CNA(genomdat, chrom, maploc))

# test code for weighted CBS
set.seed(97)
wts <- sample(1:3, 500, replace=TRUE)
genomdat <- rnorm(500, sd=0.3)/sqrt(wts) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
plot(genomdat)
chrom <- rep(1:2,c(290,210))
maploc <- c(1:290,1:210)
test3 <- segment(CNA(genomdat, chrom, maploc), weights=wts)

#A real analyis

data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))

#We generally recommend smoothing single point outliers before analysis
#Make sure to check that the smoothing is proper

smoothed.CNA.object <- smooth.CNA(CNA.object)

#Segmentation at default parameters

segment.smoothed.CNA.object <- segment(smoothed.CNA.object, verbose=1)
data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))

#We generally recommend smoothing single point outliers before analysis
#Make sure to check that the smoothing is proper

smoothed.CNA.object <- smooth.CNA(CNA.object)

#Segmentation at default parameters

segment.smoothed.CNA.object <- segment(smoothed.CNA.object, verbose=1)
```

segments.p *p-values for the change-points*

Description

This program computes pseudo p-values and confidence intervals for the change-points found by the circular binary segmentation (CBS) algorithm.

Usage

```
segments.p(x, ngrid=100, tol=1e-6, alpha=0.05, search.range=100, nperm=1000)
```

Arguments

`x`	an object of class DNAcopy
`ngrid`	number grid points for the integral
`tol`	tolerance level for calculating nu
`alpha`	Confidence level is 1-alpha
`search.range`	statistic is maximized over nu +/- search.range
`nperm`	number of permutations for confidence interval

Details

The p-values are obtained by applying Siegmund's approximation for the maximal statistic from binary segmenting consecutive segments within a chromosome. This p-value is only to give the relative importance of the change-points as the CBS is different from the algorithm used here.

The confidence intervals are obtained by a permutation algorithm. The data are permuted to the left and right of the identified change-point and the location of the maximal binary segmentation statistic computed. The confidence interval is given by the quantiles of the permutation distribution of the locations.

The statistical properties of this confidence interval is unknown. It is used to give an idea of the uncertainity on the location of the change-points as the CBS is different from the algorithm used here.

Value

a data frame with ten columns. The maximal statistic from binary segmentation, the p-values and lower and upper alpha/2 confidence limits (as genomic positions) are added to the six columns from the segment command.

NOTE: THE p VALUES ARE APPROXIMATE TAIL PROBABILITIES. ANY VALUE GREATER THAN 0.1 CAN HAVE LARGE ERROR. p > 1 ARE REPLACED WITH 1.

Author(s)

Venkatraman E. Seshan <`ves2111@columbia.edu`>

Examples

```
# test code on an easy data set
set.seed(25)
genomdat <- rnorm(500, sd=0.1) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
plot(genomdat)
chrom <- rep(1:2,c(290,210))
maploc <- c(1:290,1:210)
test1 <- segment(CNA(genomdat, chrom, maploc))
segments.p(test1)
```

segments.summary *Additional summary measured for the segments*

Description

This program computes the standard deviation, median and the mad of the data for each segment found by the CBS algorithm.

Usage

```
segments.summary(x)
```

Arguments

x	an object of class DNAcopy

Value

a data frame with nine columns. The sd, median and mad of each segment is added to the six columns from the segment command.

Author(s)

Venkatraman E. Seshan <ves2111@columbia.edu>

Examples

```
# test code on an easy data set
set.seed(25)
genomdat1 <- rnorm(500, sd=0.1) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
genomdat2 <- rnorm(500, sd=0.1) +
rep(c(-0.2,0.1,1,-0.5,0.2,-0.5,0.1,-0.2),c(137,87,17,49,29,52,87,42))
genomdat1[sample(1:500,5)] <- NA
chrom <- rep(1:2,c(290,210))
maploc <- c(1:290,1:210)
test1 <- segment(CNA(cbind(genomdat1,genomdat2), chrom, maploc))
segments.summary(test1)
```

smooth.CNA *Smooth a 'Copy Number Array' data object*

Description

Detect outliers and smooth the data prior to analysis by programs such as circular binary segmentation (CBS).

Usage

```
smooth.CNA(x, smooth.region=2, outlier.SD.scale=4, smooth.SD.scale=2,
                trim=0.025)
```

Arguments

`x`	Copy number array data object
`smooth.region`	number of points to consider on the left and the right of a point to detect it as an outlier.
`outlier.SD.scale`	the number of SDs away from the nearest point in the smoothing region to call a point an outlier.
`smooth.SD.scale`	the number of SDs from the median in the smoothing region where a smoothed point is positioned.
`trim`	proportion of data to be trimmed for variance calculation for smoothing outliers and undoing splits based on SD.

Value

An object of class `CNA` with outliers smoothed

Examples

```
data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))

#We generally recommend smoothing single point outliers before analysis
#Make sure to check that the smoothing is proper

smoothed.CNA.object <- smooth.CNA(CNA.object)
```

subset.CNA *Subset a 'Copy Number Array' data object*

Description

Function to return a subset of a copy number array data object by a list of chromosomes and sample.

Usage

```
  ## S3 method for class 'CNA':
subset(x, chromlist=NULL, samplelist=NULL, ...)
```

Arguments

`x`	Copy number array data object
`chromlist`	chromosomes of interest. Should be a subset of the valid chromosome names in the original data.
`samplelist`	samples of interest. Can be integers denoting the samples of interest or a vector of valid sample names.
`...`	other arguments which may be passed to `subset`.

Value

An object of class `CNA` with the data for the list of chromosomes and samples of interest.

Examples

```

data(coriell)

#Combine into one CNA object to prepare for analysis on Chromosomes 1-23

CNA.object <- CNA(cbind(coriell$Coriell.05296,coriell$Coriell.13330),
                  coriell$Chromosome,coriell$Position,
                  data.type="logratio",sampleid=c("c05296","c13330"))

#Take the first ten chromosomes of the first sample

#subset.CNA.object <- subset.CNA(CNA.object,chromlist=1:10,samplelist="c05296")
subset.CNA.object <- subset(CNA.object,chromlist=1:10,samplelist="c05296")
```

subset.DNAcopy *Subset a DNAcopy data object*

Description

Function to return a subset of a copy number array data object by a list of chromosomes and sample.

Usage

```
  ## S3 method for class 'DNAcopy':
subset(x, chromlist=NULL, samplelist=NULL, ...)
```

Arguments

`x`	DNAcopy object
`chromlist`	chromosomes of interest. Should be a subset of the valid chromosome names in the original data.
`samplelist`	samples of interest. Can be integers denoting the samples of interest or a vector of valid sample names.
`...`	other arguments which may be passed to `subset`.

Value

An object of class `DNAcopy` with the input data and the results of segmenting them only for the chromosomes and samples of interest.

Index