fastseg

March 24, 2012

`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.

References

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

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

`fastseg` *Detection of breakpoints using a fast segmentation algorithm based on the cyber t-test.*

Description

Detection of breakpoints using a fast segmentation algorithm based on the cyber t-test.

Usage

```
fastseg(x, type = 1, alpha = 0.1, segMedianT, minSeg = 4,
eps = 0, delta = 5, maxInt = 40, squashing = 0,
cyberWeight = 10, ...)
```

Arguments

x	Values to be segmented either in the format of a sorted GRanges object, ExpressionSet object, matrix or vector.
type	Parameter that sets the type of test. If set to 1 a test of the left against the right window is performend. If set to 2 the segment is also tested against the global mean. (Default = 1).
alpha	A value between 0 and 1 is interpreted as the ratio of initial breakpoints. An integer greater than one is interpreted as number of desired breakpoints. Increasing this parameter leads to more segments. (Default = 0.05)
segMedianT	A numeric vector of length two with the thresholds of segments' median values that are considered as significant. Only segments with a median above the first or below the second value are kept in a final merging step. If missing the algorithm will try to find a reasonable value by using z-scores. (Default "missing".)
minSeg	The minimal segment length. (Default = 4).
eps	Minimal distance between consecutive values. Only consecutive values with a minimium distance of "eps" are tested. This makes the segmentation algorithm even faster. If all values should be tested "eps" can be set to zero. If missing the algorithm will try to find a reasonable value by using quantiles. (Default "missing".)
delta	Segment extension parameter. If delta consecutive extensions of the left and the right segment do not lead to a better p-value the testing is stopped. (Default = 5).
maxInt	Maximal length of the left and the right segment. (Default = 40).
squashing	The degree of squashing of the input values. If set to zero no squashing is performed. (Default = 0).
cyberWeight	The nu parameter of the cyber t-test. Can be interpreted as the weight of the global variance. The higher the value the more small segments with high variance will be significant. (Default = 10).
...	Further arguments passed to the plot function.

Value

A data frame containing the segments.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

```
library(fastseg)

#######################################################################
### the data
#######################################################################
data(coriell)
head(coriell)

samplenames <- colnames(coriell)[4:5]
data <- as.matrix(coriell[4:5])
```

```
data[is.na(data)] <- median(data, na.rm=TRUE)
chrom <- coriell$Chromosome
maploc <- coriell$Position


############################################################
## GRanges
############################################################

library("GenomicRanges")

## with both individuals
gr <- GRanges(seqnames=chrom,
ranges=IRanges(maploc, end=maploc))
elementMetadata(gr) <- data
colnames(elementMetadata(gr)) <- samplenames
res <- fastseg(gr)

segres <- toDNAcopyObj(
segData     = res,
chrom       = as.character(seqnames(gr)),
maploc      = as.numeric(start(gr)),
genomdat    = data,
sampleNames = samplenames)

## with one individual
gr2 <- gr
data2 <- as.matrix(data[, 1])
colnames(data2) <- "sample1"
elementMetadata(gr2) <- data2
res <- fastseg(gr2)

segres <- toDNAcopyObj(
segData     = res,
chrom       = as.character(seqnames(gr)),
maploc      = as.numeric(start(gr)),
genomdat    = as.matrix(data2),
sampleNames = unique(elementMetadata(res)$ID))


############################################################
## vector
############################################################
data2 <- data[, 1]
res <- fastseg(data2)
segres <- toDNAcopyObj(
segData     = res,
chrom       = rep(1, length(data2)),
maploc      = 1:length(data2),
genomdat    = as.matrix(data2),
sampleNames = "sample1")


############################################################
## matrix
############################################################
data2 <- data[1:400, ]
```

```
res <- fastseg(data2)
segres <- toDNAcopyObj(
segData     = res,
chrom       = rep(1, nrow(data2)),
maploc      = 1:nrow(data2),
genomdat    = as.matrix(data2),
sampleNames = colnames(data2))


############################################################
## Expression set object
############################################################
library(oligo)
eSet <- new("ExpressionSet")
assayData(eSet) <- list(intensity=data)

featureData(eSet) <- new("AnnotatedDataFrame",
data=data.frame(
chrom = chrom,
start = maploc,
end   = maploc))
phenoData(eSet) <- new("AnnotatedDataFrame",
data=data.frame(samples=samplenames))
sampleNames(eSet) <- samplenames
res <- fastseg(eSet)

segres <- toDNAcopyObj(
segData     = res,
chrom       = rep(1, nrow(data)),
maploc      = maploc,
genomdat    = as.matrix(data),
sampleNames = colnames(data))


#####################################################################
### plot the segments
#####################################################################

library(DNAcopy)
plot(segres, xmaploc=TRUE)
```

`fastsegData` *Example data set for fastseg*

Description

The data is a small subset of copy number calls which were produced by the cn.farms algorithm from an Affymetrix SNP microarray experiment of a HapMap sample.

Usage

```
data(fastsegData)
```

Format

A simple vector with a copy number call as produced by the cn.farms algorithm.

References

http://nar.oxfordjournals.org/content/early/2011/04/12/nar.gkr197.abstract Clevert et al., cn.FARMS: a latent variable model to detect copy number variations in microarray data with a low false discovery rate, NAR, 2011

`segPlot` *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.*

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

```
segPlot(x, res, plot.type = "chrombysample",
  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`	The object that was segmented by fastseg.
`res`	The result of fastseg.
`plot.type`	the type of plot. (Default = "s").
`altcol`	logical flag to indicate if chromosomes should be plotted in alternating colors in the whole genome plot. (Default = TRUE).
`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. (Default = NULL).
`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. (Default = NULL).
`cbys.nchrom`	the number of chromosomes per page in the layout. (Default = 1).
`include.means`	logical flag to indicate whether segment means are to be drawn. (Default = TRUE).
`zeroline`	logical flag to indicate whether a horizontal line at y=0 is to be drawn. (Default = TRUE).

pt.pch	the plotting character used for plotting the log-ratio values. (Default = ".")
pt.cex	the size of plotting character used for the log-ratio values (Default = 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. (Default = "red").
zlcol	the color of the zeroline. (Default = "grey").
ylim	this argument is present to override the default limits which is the range of symmetrized log-ratios. (Default = NULL).
lwd	line weight of lines for segment mean and zeroline. (Default = 3).
...	other arguments which will be passed to plot commands.

Value

A plot of the values and segments.

Author(s)

klambaue

Examples

```
data(coriell)
head(coriell)
samplenames <- colnames(coriell)[4:5]
data <- as.matrix(coriell[4:5])
chrom <- coriell$Chromosome
maploc <- coriell$Position
library("GenomicRanges")
gr <- GRanges(seqnames=chrom,
ranges=IRanges(maploc, end=maploc))
elementMetadata(gr) <- data
colnames(elementMetadata(gr)) <- samplenames
res <- fastseg(gr)
segPlot(gr,res)
```

---

toDNAcopyObj *Function to create a DNAcopy object for plot functions.*

Description

Function to create a DNAcopy object for plot functions.

Usage

```
toDNAcopyObj(segData, chrom, maploc, genomdat,
sampleNames)
```

Arguments

segData	The results of the segmentation.
chrom	The vector of the chromosomes from the original data.
maploc	A vector with the physical positions of the original data.
genomdat	A matrix with the original data.
sampleNames	The sample names of the original data.

Value

An DNAcopy equivalent object.

Author(s)

Andreas Mitterecker

Examples

```
library(fastseg)

#########################################################################
### the data
#########################################################################
data(coriell)
head(coriell)

samplenames <- colnames(coriell)[4:5]
data <- as.matrix(coriell[4:5])
data[is.na(data)] <- median(data, na.rm=TRUE)
chrom <- coriell$Chromosome
maploc <- coriell$Position


############################################################
## GRanges
############################################################

library("GenomicRanges")

## with both individuals
gr <- GRanges(seqnames=chrom,
ranges=IRanges(maploc, end=maploc))
elementMetadata(gr) <- data
colnames(elementMetadata(gr)) <- samplenames
res <- fastseg(gr)

segres <- toDNAcopyObj(
segData     = res,
chrom       = as.character(seqnames(gr)),
maploc      = as.numeric(start(gr)),
genomdat    = data,
sampleNames = samplenames)

## with one individual
gr2 <- gr
data2 <- as.matrix(data[, 1])
```

```
colnames(data2) <- "sample1"
elementMetadata(gr2) <- data2
res <- fastseg(gr2)

segres <- toDNAcopyObj(
segData     = res,
chrom       = as.character(seqnames(gr)),
maploc      = as.numeric(start(gr)),
genomdat    = as.matrix(data2),
sampleNames = unique(elementMetadata(res)$ID))


############################################################
## vector
############################################################
data2 <- data[, 1]
res <- fastseg(data2)
segres <- toDNAcopyObj(
segData     = res,
chrom       = rep(1, length(data2)),
maploc      = 1:length(data2),
genomdat    = as.matrix(data2),
sampleNames = "sample1")


############################################################
## matrix
############################################################
data2 <- data[1:400, ]
res <- fastseg(data2)
segres <- toDNAcopyObj(
segData     = res,
chrom       = rep(1, nrow(data2)),
maploc      = 1:nrow(data2),
genomdat    = as.matrix(data2),
sampleNames = colnames(data2))


############################################################
## Expression set object
############################################################
library(oligo)
eSet <- new("ExpressionSet")
assayData(eSet) <- list(intensity=data)

featureData(eSet) <- new("AnnotatedDataFrame",
data=data.frame(
chrom = chrom,
start = maploc,
end   = maploc))
phenoData(eSet) <- new("AnnotatedDataFrame",
data=data.frame(samples=samplenames))
sampleNames(eSet) <- samplenames
res <- fastseg(eSet)

segres <- toDNAcopyObj(
segData     = res,
```

```
    chrom       = rep(1, nrow(data)),
    maploc      = maploc,
    genomdat    = as.matrix(data),
    sampleNames = colnames(data))


    ######################################################################
    ### plot the segments
    ######################################################################

    library(DNAcopy)
    plot(segres)
```

Index