Package ‘fastseg’

December 21, 2016

Maintainer Guenter Klambauer <fastseg@bioinf.jku.at>
Author Guenter Klambauer
License LGPL (>= 2.0)
Type Package
Title fastseg - a fast segmentation algorithm
Description fastseg implements a very fast and efficient segmentation algorithm. It has similar functionality as DNACopy (Olshen and Venkatraman 2004), but is considerably faster and more flexible. fastseg can segment data from DNA microarrays and data from next generation sequencing for example to detect copy number segments. Further it can segment data from RNA microarrays like tiling arrays to identify transcripts. Most generally, it can segment data given as a matrix or as a vector. Various data formats can be used as input to fastseg like expression set objects for microarrays or GRanges for sequencing data. The segmentation criterion of fastseg is based on a statistical test in a Bayesian framework, namely the cyber t-test (Baldi 2001). The speed-up arises from the facts, that sampling is not necessary in for fastseg and that a dynamic programming approach is used for calculation of the segments' first and higher order moments.

Version 1.20.0
URL http://www.bioinf.jku.at/software/fastseg/fastseg.html
Depends R (>= 2.13), GenomicRanges, Biobase
Imports methods, graphics, stats, BiocGenerics, S4Vectors, IRanges
Suggests DNAcopy, oligo
Collate 'roxygen.R' 'toDnaCopy.R' 'fastseg.R' 'segPlot.R'
biocViews Classification, CopyNumberVariation
NeedsCompilation yes

R topics documented:

  coriell ......................................................... 2
  fastseg ........................................................ 2
  fastsegData .................................................... 4
  segPlot ........................................................ 4
  toDNAcopyObj .................................................. 6
Index

fastseg

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.

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

```r
fastseg(x, type = 1, alpha = 0.05, 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 performed. 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 = 0.1).
- **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
minimum 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 = 0.)
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 vari-
ance will be significant. (Default = 10).

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))
mcols(gr) <- data
colnames(mcols(gr)) <- samplenames
res <- fastseg(gr)

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

###########################################################
## vector
###########################################################
data2 <- data[, 1]
res <- fastseg(data2)

###########################################################
## matrix
###########################################################
data2 <- data[1:400, ]
res <- fastseg(data2)
```

**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.

**Format**

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

**References**


**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, ...)
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)
```r
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))
mcols(gr) <- data
colnames(mcols(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"
mcols(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(mcols(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")
```
```r
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))

library(DNAcopy)
plot(segres)
```

## matrix

<table>
<thead>
<tr>
<th>data2 &lt;- data[1:400, ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>res &lt;- fastseg(data2)</td>
</tr>
</tbody>
</table>
| segres <- toDNAcopyObj(
  segData = res,
  chrom = rep(1, nrow(data2)),
  maploc = 1:nrow(data2),
  genomdat = as.matrix(data2),
  sampleNames = colnames(data2)) |

### plot the segments

```r
library(DNAcopy)
plot(segres)
```
Index

*Topic datasets
  coriell, 2
  fastsegData, 4
*Topic data
  coriell, 2
  fastsegData, 4

coriell, 2
fastseg, 2
fastsegData, 4
segPlot, 4
toDNAcopyObj, 6