Package ‘cn.mops’

March 22, 2017

Maintainer Guenter Klambauer <cn.mops@bioinf.jku.at>
Author Guenter Klambauer
License LGPL (>= 2.0)
Type Package
Title cn.mops - Mixture of Poissons for CNV detection in NGS data
Description cn.mops (Copy Number estimation by a Mixture Of PoissonS) is a data processing pipeline for copy number variations and aberrations (CNVs and CNAs) from next generation sequencing (NGS) data. The package supplies functions to convert BAM files into read count matrices or genomic ranges objects, which are the input objects for cn.mops. cn.mops models the depths of coverage across samples at each genomic position. Therefore, it does not suffer from read count biases along chromosomes. Using a Bayesian approach, cn.mops decomposes read variations across samples into integer copy numbers and noise by its mixture components and Poisson distributions, respectively. cn.mops guarantees a low FDR because wrong detections are indicated by high noise and filtered out. cn.mops is very fast and written in C++.

Version 1.20.1
Date 2016-03-03

URL http://www.bioinf.jku.at/software/cnmops/cnmops.html

Depends R (>= 2.12), methods, utils, stats, graphics, parallel, GenomicRanges
Imports BiocGenerics, Biobase, IRanges, Rsamtools, GenomeInfoDb, S4Vectors, exomeCopy
Suggests DNAcopy

LazyLoad yes
biocViews Sequencing, CopyNumberVariation, Homo_sapiens, CellBiology, HapMap, Genetics

RoxygenNote 6.0.1

NeedsCompilation yes
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>calcFractionalCopyNumbers</td>
<td>3</td>
</tr>
<tr>
<td>calcFractionalCopyNumbers,CNVDetectionResult-method</td>
<td>3</td>
</tr>
<tr>
<td>calcIntegerCopyNumbers</td>
<td>4</td>
</tr>
<tr>
<td>calcIntegerCopyNumbers,CNVDetectionResult-method</td>
<td>5</td>
</tr>
<tr>
<td>cn.mops</td>
<td>5</td>
</tr>
<tr>
<td>CNVDetectionResult-class</td>
<td>7</td>
</tr>
<tr>
<td>cnrv,CNVDetectionResult-method</td>
<td>8</td>
</tr>
<tr>
<td>cnvr</td>
<td>9</td>
</tr>
<tr>
<td>cnvs</td>
<td>10</td>
</tr>
<tr>
<td>cnvs,CNVDetectionResult-method</td>
<td>11</td>
</tr>
<tr>
<td>exomecn.mops</td>
<td>11</td>
</tr>
<tr>
<td>exomeCount</td>
<td>13</td>
</tr>
<tr>
<td>getReadCountsFromBAM</td>
<td>14</td>
</tr>
<tr>
<td>getSegmentReadCountsFromBAM</td>
<td>15</td>
</tr>
<tr>
<td>gr,CNVDetectionResult-method</td>
<td>16</td>
</tr>
<tr>
<td>haplocn.mops</td>
<td>17</td>
</tr>
<tr>
<td>individualCall</td>
<td>19</td>
</tr>
<tr>
<td>individualCall,CNVDetectionResult-method</td>
<td>19</td>
</tr>
<tr>
<td>iniCall</td>
<td>20</td>
</tr>
<tr>
<td>iniCall,CNVDetectionResult-method</td>
<td>20</td>
</tr>
<tr>
<td>integerCopyNumber</td>
<td>21</td>
</tr>
<tr>
<td>integerCopyNumber,CNVDetectionResult-method</td>
<td>21</td>
</tr>
<tr>
<td>localAssessments</td>
<td>22</td>
</tr>
<tr>
<td>localAssessments,CNVDetectionResult-method</td>
<td>23</td>
</tr>
<tr>
<td>makeRobustCNVR</td>
<td>23</td>
</tr>
<tr>
<td>normalizeChromosomes</td>
<td>24</td>
</tr>
<tr>
<td>normalizedData</td>
<td>25</td>
</tr>
<tr>
<td>normalizedData,CNVDetectionResult-method</td>
<td>26</td>
</tr>
<tr>
<td>normalizeGenome</td>
<td>27</td>
</tr>
<tr>
<td>params</td>
<td>28</td>
</tr>
<tr>
<td>params,CNVDetectionResult-method</td>
<td>29</td>
</tr>
<tr>
<td>plot,CNVDetectionResult-method</td>
<td>29</td>
</tr>
<tr>
<td>posteriorProbs</td>
<td>30</td>
</tr>
<tr>
<td>posteriorProbs,CNVDetectionResult-method</td>
<td>31</td>
</tr>
<tr>
<td>referencecn.mops</td>
<td>31</td>
</tr>
<tr>
<td>sampleNames</td>
<td>33</td>
</tr>
<tr>
<td>sampleNames,CNVDetectionResult-method</td>
<td>34</td>
</tr>
<tr>
<td>segment</td>
<td>35</td>
</tr>
<tr>
<td>segmentation</td>
<td>36</td>
</tr>
<tr>
<td>segmentation,CNVDetectionResult-method</td>
<td>36</td>
</tr>
<tr>
<td>segplot</td>
<td>37</td>
</tr>
<tr>
<td>segplot,CNVDetectionResult-method</td>
<td>38</td>
</tr>
<tr>
<td>show</td>
<td>38</td>
</tr>
<tr>
<td>singlecn.mops</td>
<td>40</td>
</tr>
<tr>
<td>X</td>
<td>40</td>
</tr>
<tr>
<td>XRanges</td>
<td>43</td>
</tr>
</tbody>
</table>

Index 44
calcFractionalCopyNumbers

Calculation of fractional copy numbers for the CNVs and CNV regions.

Description
This generic function calculates the fractional copy numbers of a CNV detection method stored in an instance of `CNVDetectionResult-class`. Must be a result of "referencecn.mops".

Arguments

- `object`: An instance of "CNVDetectionResult"
- `segStat`: Which statistic per segment should be used. Can be either "mean" or "median". (Default="mean").

Value
`calcFractionalCopyNumbers` returns an instance of "CNVDetectionResult".

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
```r
data(cn.mops)
r <- referencecn.mops(X[,1:2],apply(X,1,median))
calcFractionalCopyNumbers(r)
```

Usage
```r
## S4 method for signature 'CNVDetectionResult'
calcFractionalCopyNumbers(object, segStat = "mean")
```

Arguments

- `object`: An instance of "CNVDetectionResult"
- `segStat`: Which statistic per segment should be used. Can be either "mean" or "median". (Default="mean").
calcIntegerCopyNumbers

Value

calcFractionalCopyNumbers returns an instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- referencecn.mops(X[,1:2],apply(X,1,median))
calcFractionalCopyNumbers(r)

calcIntegerCopyNumbers

Calculation of integer copy numbers for the CNVs and CNV regions.

Description

This generic function calculates the integer copy numbers of a CNV detection method stored in an instance of CNVDetectionResult-class.

Arguments

object An instance of "CNVDetectionResult"

Value

calcIntegerCopyNumbers returns an instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
calcIntegerCopyNumbers(r)
calcIntegerCopyNumbers,CNVDetectionResult-method

Calculation of integer copy numbers for the CNVs and CNV regions.

Description

This generic function calculates the integer copy numbers of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Usage

```r
## S4 method for signature 'CNVDetectionResult'
calcIntegerCopyNumbers(object)
```

Arguments

- `object` An instance of "CNVDetectionResult"

Value

`calcIntegerCopyNumbers` returns an instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
calcIntegerCopyNumbers(r)
```

cn.mops

Copy number detection in NGS data.

Description

This function performs the cn.mops algorithm for copy number detection in NGS data.

Usage

```r
cn.mops(input, I = c(0.025, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4),
classes = c("CN0", "CN1", "CN2", "CN3", "CN4", "CN5", "CN6", "CN7", "CN8"),
priorImpact = 1, cyc = 20, parallel = 0, norm = 1,
normType = "poisson", sizeFactor = "mean", normQu = 0.25,
quSizeFactor = 0.75, upperThreshold = 0.5, lowerThreshold = -0.9,
minWidth = 3, segAlgorithm = "fast", minReadCount = 5,
useMedian = FALSE, returnPosterior = FALSE, ...)
```

Arguments

**input**
Either an instance of "GRanges" or a raw data matrix, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region.

**I**
Vector positive real values that contain the expected fold change of the copy number classes. Length of this vector must be equal to the length of the "classes" parameter vector. For human copy number polymorphisms we suggest to use the default \( I = c(0.025,0.5,1.5,2,2.5,3,3.5,4) \).

**classes**
Vector of characters of the same length as the parameter vector "I". One vector element must be named "CN2". The names reflect the labels of the copy number classes. Default = c("CN0","CN1","CN2","CN3","CN4","CN5","CN6","CN7","CN8").

**priorImpact**
Positive real value that reflects how strong the prior assumption affects the result. The higher the value the more samples will be assumed to have copy number 2. Default = 1.

**cyc**
Positive integer that sets the number of cycles for the algorithm. Usually after less than 15 cycles convergence is reached. Default = 20.

**parallel**
How many cores are used for the computation. If set to zero than no parallelization is applied. Default = 0.

**norm**
The normalization strategy to be used. If set to 0 the read counts are not normalized and cn.mops does not model different coverages. If set to 1 the read counts are normalized. If set to 2 the read counts are not normalized and cn.mops models different coverages. (Default=1).

**normType**
Mode of the normalization technique. Possible values are "mean", "min", "median", "quant", "poisson" and "mode". Read counts will be scaled sample-wise. Default = "poisson".

**sizeFactor**
By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

**normQu**
Real value between 0 and 1. If the "normType" parameter is set to "quant" then this parameter sets the quantile that is used for the normalization. Default = 0.25.

**quSizeFactor**
Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

**upperThreshold**
Positive real value that sets the cut-off for copy number gains. All CNV calling values above this value will be called as "gain". The value should be set close to the \( \log_2 \) of the expected foldchange for copy number 3 or 4. Default = 0.5.

**lowerThreshold**
Negative real value that sets the cut-off for copy number losses. All CNV calling values below this value will be called as "loss". The value should be set close to the \( \log_2 \) of the expected foldchange for copy number 1 or 0. Default = -0.9.

**minWidth**
Positive integer that is exactly the parameter "min.width" of the "segment" function of "DNAcopy". minWidth is the minimum number of segments a CNV should span. Default = 3.

**segAlgorithm**
Which segmentation algorithm should be used. If set to "DNAcopy" circular binary segmentation is performed. Any other value will initiate the use of our fast segmentation algorithm. Default = "fast".

**minReadCount**
If all samples are below this value the algorithm will return the prior knowledge. This prevents that the algorithm from being applied to segments with very low coverage. Default=5.
useMedian     Whether "median" instead of "mean" of a segment should be used for the CNV call. Default=FALSE.

returnPosterior  Flag that decides whether the posterior probabilities should be returned. The posterior probabilities have a dimension of samples times copy number states times genomic regions and therefore consume a lot of memory. Default=FALSE.

... Additional parameters will be passed to the "DNAcopy" or the standard segmentation algorithm.

Value
An instance of "CNVDetectionResult".

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
```r
data(cn.mops)
cn.mops(XRanges)
cn.mops(XRanges,parallel=2)
```

Description
S4 class for storing results of a CNV detection method.

Slots
The following slots are defined for `CNVDetectionResult` objects:

- `gr`  The segments in which the reads are counted. GRanges object.
- `normalizedData` The normalized data.
- `localAssessments` The data to which the segmentation algorithm is applied. These can be z-Scores, ratios, log-ratios or I/NI calls.
- `individualCall` The CNV call that the method provides for a specific sample
- `iniCall` The CNV call that the method provides a specific segment.
- `posteriorProbs` The posterior probabilities for different copy numbers.
- `cnvs` The detected CNVs.
- `cnvr` The detected CNV regions.
- `segmentation` The segmentation of the reference sequence (sample-wise).
- `integerCopyNumber` The most probable integer copy number.
- `sampleNames` The sample names.
- `params` The parameters with which the method was run.
Methods

- `gr` signature(object = "CNVDetectionResult"): ...
- `cnvr` signature(object = "CNVDetectionResult"): ...
- `cnvs` signature(object = "CNVDetectionResult"): ...
- `individualCall` signature(object = "CNVDetectionResult"): ...
- `iniCall` signature(object = "CNVDetectionResult"): ...
- `integerCopyNumber` signature(object = "CNVDetectionResult"): ...
- `localAssessments` signature(object = "CNVDetectionResult"): ...
- `normalizedData` signature(object = "CNVDetectionResult"): ...
- `params` signature(object = "CNVDetectionResult"): ...
- `plot` signature(x = "CNVDetectionResult", y = "missing"): ...
- `posteriorProbs` signature(object = "CNVDetectionResult"): ...
- `sampleNames` signature(object = "CNVDetectionResult"): ...
- `segmentation` signature(object = "CNVDetectionResult"): ...
- `segplot` signature(object = "CNVDetectionResult"): ...
- `show` signature(object = "CNVDetectionResult"): ...
- `calcIntegerCopyNumbers` signature(object = "CNVDetectionResult"): ...
- `makeRobustCNVR` signature(object = "CNVDetectionResult"): ...

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

`showClass("CNVDetectionResult")`

| cnvr | This generic function returns CNV regions of a CNV detection method stored in an instance of `CNVDetectionResult`-class. |

Description

This generic function returns CNV regions of a CNV detection method stored in an instance of `CNVDetectionResult`-class.

Arguments

- `object` An instance of "CNVDetectionResult"

Value

`cnvr` returns a "GRanges" object containing the CNV regions.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>
**Examples**

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
cnvr(r)
```

---

**Description**

This generic function returns CNV regions of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

**Usage**

```r
## S4 method for signature 'CNVDetectionResult'
cnvr(object)
```

**Arguments**

- `object` An instance of "CNVDetectionResult"

**Value**

cnvr returns a `GRanges` object containing the CNV regions.

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
cnvr(r)
```

---

**CNVRanges**

Genomic locations and indices of the simulated CNVs.

---

**Description**

This data set gives the starts, ends, and the integer copy number of the simulated CNVs in the data set `XRanges` object.

**Usage**

`CNVRanges`
Format

A GRanges object with 20 rows and 40 value columns across 1 space.

Source

http://www.bioinf.jku.at/cnmops/cnmops.html.

References


cnvs

This generic function returns CNVs of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns CNVs of a CNV detection method stored in an instance of CNVDetectionResult-class.

Arguments

object

An instance of "CNVDetectionResult"

Value

cnvs returns a GRanges object containing the CNVs.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
cnvs(r)
This generic function returns CNVs of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

### Usage

```r
## S4 method for signature 'CNVDetectionResult'
cnvs(object)
```

### Arguments

- `object` : An instance of "CNVDetectionResult"

### Value

`cnvs` returns a "GRanges" object containing the CNVs.

### Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

### Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
cnvs(r)
```

---

### Description

Performs the `cn.mops` algorithm for copy number detection in NGS data with parameters adjusted to exome sequencing data.

### Usage

```r
exomecn.mops(input, I = c(0.025, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4),
classes = c("CN0", "CN1", "CN2", "CN3", "CN4", "CN5", "CN6", "CN7", "CN8"),
priorImpact = 10, cyc = 20, parallel = 0, norm = 1,
normType = "poisson", sizeFactor = "mean", normQu = 0.25,
quSizeFactor = 0.75, upperThreshold = 0.5, lowerThreshold = -0.8,
minWidth = 5, segAlgorithm = "fast", minReadCount = 1,
useMedian = FALSE, returnPosterior = FALSE, ...)
```
Arguments

input Either an instance of "GRanges" or a raw data matrix, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region.

I Vector positive real values that contain the expected fold change of the copy number classes. Length of this vector must be equal to the length of the "classes" parameter vector. For human copy number polymorphisms we suggest to use the default $I = c(0.025,0.5,1,1.5,2,2.5,3,3.5,4)$.

classes Vector of characters of the same length as the parameter vector "I". One vector element must be named "CN2". The names reflect the labels of the copy number classes. Default = c("CN0","CN1","CN2","CN3","CN4","CN5","CN6","CN7","CN8").

priorImpact Positive real value that reflects how strong the prior assumption affects the result. The higher the value the more samples will be assumed to have copy number 2. Default = 10.

cyc Positive integer that sets the number of cycles for the algorithm. Usually after less than 15 cycles convergence is reached. Default = 20.

parallel How many cores are used for the computation. If set to zero than no parallelization is applied. Default = 0.

norm The normalization strategy to be used. If set to 0 the read counts are not normalized and cn.mops does not model different coverages. If set to 1 the read counts are normalized. If set to 2 the read counts are not normalized and cn.mops models different coverages. (Default=1).

normType Mode of the normalization technique. Possible values are "mean", "min", "median", "quant", "poisson" and "mode". Read counts will be scaled sample-wise. Default = "poisson".

sizeFactor By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

normQu Real value between 0 and 1. If the "normType" parameter is set to "quant" then this parameter sets the quantile that is used for the normalization. Default = 0.25.

quSizeFactor Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

upperThreshold Positive real value that sets the cut-off for copy number gains. All CNV calling values above this value will be called as "gain". The value should be set close to the log2 of the expected foldchange for copy number 3 or 4. Default = 0.55.

lowerThreshold Negative real value that sets the cut-off for copy number losses. All CNV calling values below this value will be called as "loss". The value should be set close to the log2 of the expected foldchange for copy number 1 or 0. Default = -0.8.

minWidth Positive integer that is exactly the parameter "min.width" of the "segment" function of "DNAcopy". minWidth is the minimum number of segments a CNV should span. Default = 5.

segAlgorithm Which segmentation algorithm should be used. If set to "DNAcopy" circular binary segmentation is performed. Any other value will initiate the use of our fast segmentation algorithm. Default = "fast".

minReadCount If all samples are below this value the algorithm will return the prior knowledge. This prevents that the algorithm from being applied to segments with very low coverage. Default=1.
exomeCounts

useMedian  Whether "median" instead of "mean" of a segment should be used for the CNV call. Default=FALSE.

returnPosterior  Flag that decides whether the posterior probabilities should be returned. The posterior probabilities have a dimension of samples times copy number states times genomic regions and therefore consume a lot of memory. Default=FALSE.

Additional parameters will be passed to the "DNAcopy" or the standard segmentation algorithm.

Value

An instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
exomecn.mops(exomeCounts)

Description

This data set gives the read counts on chromosome 22 (hg19) of 22 samples in 3785 exons. The rows correspond to targeted regions or exons and columns to samples. An entry is the number of reads that map to the specific segment, i.e. targeted region or exon, of the sample. The GRanges object contains the information of the genomic location. The read counts were generated from freely available exome sequencing data of the 1000Genomes Project.

Usage

exomeCounts

Format

A GRanges object of 3785 rows and 22 columns.

Source

getReadCountsFromBAM

Calculation of read counts from BAM files.

Description

Generates the read counts from BAM Files. These counts are necessary for CNV detection methods based on depth of coverage information.

This function can also be run in a parallel version.

Usage

getReadCountsFromBAM(BAMFiles, sampleNames, refSeqNames, WL = 25000, parallel = 0, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAMFiles</td>
<td>BAMFiles</td>
</tr>
<tr>
<td>sampleNames</td>
<td>The corresponding sample names to the BAM Files.</td>
</tr>
<tr>
<td>refSeqNames</td>
<td>Names of the reference sequence that should be analyzed. The name must appear in the header of the BAM file. If it is not given the function will select the first reference sequence that appears in the header of the BAM files. Can be set to analyze multiple chromosomes at once, e.g. refSeqNames=c(&quot;chr1&quot;,&quot;chr2&quot;)</td>
</tr>
<tr>
<td>WL</td>
<td>Windowlength. Length of the initial segmentation of the genome in basepairs. Should be chosen such that on the average 100 reads are contained in each segment.</td>
</tr>
<tr>
<td>parallel</td>
<td>The number of parallel processes to be used for this function. Default=0.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters passed to the function &quot;countBamInGRanges&quot; of the Bioconductor package &quot;exomeCopy&quot;. Quality filters for read counts can be adjusted there. Please see &quot;??countBamInGRanges&quot; for more information.</td>
</tr>
</tbody>
</table>

Value

An instance of "GRanges", that contains the breakpoints of the initial segments and the raw read counts that were extracted from the BAM files. This object can be used as input for cn.mops and other CNV detection methods.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

References


The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. Nature 2010 467(1061-1073); doi:10.1038/nature09534.
**getSegmentReadCountsFromBAM**

**Examples**

```r
BAMFiles <- list.files(system.file("extdata", package="cn.mops"), pattern=".bam\$", full.names=TRUE)
bamDataRanges <- getReadCountsFromBAM(BAMFiles, sampleNames=paste("Sample",1:3), WL=5000)
X <- getReadCountsFromBAM(BAMFiles, sampleNames=paste("Sample",1:3), WL=5000, parallel=2)
```

---

**Description**

Calculates read counts from BAM files for predefined segments. This is the appropriate choice for exome sequencing data, where the bait regions, target regions or exons are predefined segments. These counts are necessary for CNV detection methods based on depth of coverage information.

This function can also be run in a parallel version.

**Usage**

```r
getSegmentReadCountsFromBAM(BAMFiles, GR, sampleNames, parallel = 0, ...)
```

**Arguments**

- **BAMFiles**: BAMFiles
- **GR**: A genomic ranges object that contains the genomic coordinates of the segments.
- **sampleNames**: The corresponding sample names to the BAM Files.
- **parallel**: The number of parallel processes to be used for this function. Default=0.
- **...**: Additional parameters passed to the function `countBamInGRanges` of the Bioconductor package `exomeCopy`. Quality filters for read counts can be adjusted there. Please see `?countBamInGRanges` for more information.

**Value**

An instance of "GRanges", that contains the breakpoints of the initial segments and the raw read counts that were extracted from the BAM files. This object can be used as input for cn.mops and other CNV detection methods.

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
BAMFiles <- list.files(system.file("extdata", package="cn.mops"), pattern=".bam\$", full.names=TRUE)
gr <- GRanges(c("20","20"), IRanges(c(60000,70000),c(70000,80000)))
bamDataRanges <- getSegmentReadCountsFromBAM(BAMFiles, GR=gr)
bamDataRanges <- getSegmentReadCountsFromBAM(BAMFiles, GR=gr, parallel=2)
```
This generic function returns the genomic ranges of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

**Arguments**

- `object`: An instance of "CNVDetectionResult".

**Value**

- `normalizedData`: returns a "GRanges" object containing the normalized data.

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
data(cn.mops)
r <- cn.mops(X[[1:100,1:5]])
gr(r)
```

---

This generic function returns the genomic ranges of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

**Usage**

```r
## S4 method for signature 'CNVDetectionResult'
gr(object)
```

**Arguments**

- `object`: An instance of "CNVDetectionResult".

**Value**

- `normalizedData`: returns a "GRanges" object containing the normalized data.
**haplocn.mops**

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
gr(r)
```

**Description**

Copy number detection in NGS data of haploid samples. It is assumed that the normal state is copy number 1. This is an experimental method at the moment.

**Usage**

```r
haplocn.mops(input, I = c(0.025, 1, 2, 3, 4, 5, 6, 7, 8), classes = c("CN0", "CN1", "CN2", "CN3", "CN4", "CN5", "CN6", "CN7", "CN8"), priorImpact = 1, cyc = 20, parallel = 0, norm = 1, normType = "poisson", sizeFactor = "mean", normQu = 0.25, quSizeFactor = 0.75, upperThreshold = 0.6, lowerThreshold = -0.9, minWidth = 3, segAlgorithm = "fast", minReadCount = 1, returnPosterior = FALSE, ...)
```

**Arguments**

- **input**: Either an instance of "GRanges" or a raw data matrix, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region.
- **I**: Vector positive real values that contain the expected fold change of the copy number classes. Length of this vector must be equal to the length of the "classes" parameter vector. For copy number polymorphisms in haploid organisms we suggest to use the default I = c(0.025,1,2,3,4,5,6,7,8).
- **classes**: Vector of characters of the same length as the parameter vector "I". One vector element must be named "CN1". The names reflect the labels of the copy number classes. Default = c("CN0", "CN1", "CN2", "CN3", "CN4", "CN5", "CN6", "CN7", "CN8").
- **priorImpact**: Positive real value that reflects how strong the prior assumption affects the result. The higher the value the more samples will be assumed to have copy number 1. Default = 1.
- **cyc**: Positive integer that sets the number of cycles for the algorithm. Usually after less than 15 cycles convergence is reached. Default = 20.
- **parallel**: How many cores are used for the computation. If set to zero than no parallelization is applied. Default = 0.
- **norm**: The normalization strategy to be used. If set to 0 the read counts are not normalized and cn.mops does not model different coverages. If set to 1 the read counts are normalized. If set to 2 the read counts are not normalized and cn.mops models different coverages. (Default=1).
normType: Mode of the normalization technique. Possible values are "mean", "min", "median", "quant", "poisson" and "mode". Read counts will be scaled sample-wise. Default = "poisson".

sizeFactor: By this parameter one can decide to how the size factors are calculated. Possible choices are the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

normQu: Real value between 0 and 1. If the "normType" parameter is set to "quant" then this parameter sets the quantile that is used for the normalization. Default = 0.25.

quSizeFactor: Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

upperThreshold: Positive real value that sets the cut-off for copy number gains. All CNV calling values above this value will be called as "gain". The value should be set close to the log2 of the expected foldchange for copy number 3 or 4. Default = 0.5.

lowerThreshold: Negative real value that sets the cut-off for copy number losses. All CNV calling values below this value will be called as "loss". The value should be set close to the log2 of the expected foldchange for copy number 1 or 0. Default = -0.9.

minWidth: Positive integer that is exactly the parameter "min.width" of the "segment" function of "DNAcopy". minWidth is the minimum number of segments a CNV should span. Default = 4.

segAlgorithm: Which segmentation algorithm should be used. If set to "DNAcopy" circular binary segmentation is performed. Any other value will initiate the use of our fast segmentation algorithm. Default = "fast".

minReadCount: If all samples are below this value the algorithm will return the prior knowledge. This prevents that the algorithm from being applied to segments with very low coverage.

returnPosterior: Flag that decides whether the posterior probabilities should be returned. The posterior probabilities have a dimension of samples times copy number states times genomic regions and therefore consume a lot of memory. Default=FALSE.

Value
An instance of "CNVDetectionResult".

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
```r
data(cn.mops)
ahaplocn.mops(XRanges[1:200, ])
ahaplocn.mops(XRanges[1:200, ],parallel=2)
```
individualCall

This generic function returns the individual calls of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns the individual calls of a CNV detection method stored in an instance of CNVDetectionResult-class.

Arguments

- object
  - An instance of "CNVDetectionResult"

Value

- individualCalls returns a "GRanges" object containing the individual calls.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
individualCall(r)

individualCall,CNVDetectionResult-method

This generic function returns the individual calls of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns the individual calls of a CNV detection method stored in an instance of CNVDetectionResult-class.

Usage

## S4 method for signature 'CNVDetectionResult'
individualCall(object)

Arguments

- object
  - An instance of "CNVDetectionResult"

Value

- individualCalls returns a "GRanges" object containing the individual calls.
iniCall

This generic function returns the informative/non-informative call of a CNV detection method stored in an instance of \texttt{CNVDetectionResult-class}. The I/NI call is a measure for a genomic segment across all samples, whether this segment is a CNV region (informative) or a normal genomic region (non-informative).

Description

This generic function returns the informative/non-informative call of a CNV detection method stored in an instance of \texttt{CNVDetectionResult-class}. The I/NI call is a measure for a genomic segment across all samples, whether this segment is a CNV region (informative) or a normal genomic region (non-informative).

Arguments

\begin{itemize}
  \item \textbf{object} \hspace{1cm} An instance of \texttt{"CNVDetectionResult"}
\end{itemize}

Value

\texttt{iniCall} returns a \texttt{"GRanges"} object containing the individual calls.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

\begin{verbatim}
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
iniCall(r)
\end{verbatim}
iniCall(CNVDetectionResult-method)

This generic function returns the informative/non-informative call of a CNV detection method stored in an instance of CNVDetectionResult-class. The I/NI call is a measure for a genomic segment across all samples, whether this segment is a CNV region (informative) or a normal genomic region (non-informative).

Description

This generic function returns the informative/non-informative call of a CNV detection method stored in an instance of CNVDetectionResult-class. The I/NI call is a measure for a genomic segment across all samples, whether this segment is a CNV region (informative) or a normal genomic region (non-informative).

Usage

## S4 method for signature 'CNVDetectionResult'
iDomainCall(object)

Arguments

object An instance of "CNVDetectionResult"

Value

iniCall returns a "GRanges" object containing the individual calls.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
iDomainCall(r)

integerCopyNumber

This generic function returns the integer copy numbers of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns the integer copy numbers of a CNV detection method stored in an instance of CNVDetectionResult-class.

Arguments

object An instance of "CNVDetectionResult"
integerCopyNumber,CNVDetectionResult-method

Value

integerCopyNumber returns a "GRanges" object containing the integer copy numbers.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
integerCopyNumber(r)

integerCopyNumber,CNVDetectionResult-method

This generic function returns the integer copy numbers of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns the integer copy numbers of a CNV detection method stored in an instance of CNVDetectionResult-class.

Usage

## S4 method for signature 'CNVDetectionResult'
integerCopyNumber(object)

Arguments

object An instance of "CNVDetectionResult"

Value

integerCopyNumber returns a "GRanges" object containing the integer copy numbers.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
integerCopyNumber(r)
localAssessments

This generic function returns the local assessments, i.e. signed individual informative/non-informative calls, of a CNV detection method stored in an instance of `CNVDetectionResult-class`. For other CNV detection methods this can be (log-) ratios or z-scores.

Description

This generic function returns the local assessments, i.e. signed individual informative/non-informative calls, of a CNV detection method stored in an instance of `CNVDetectionResult-class`. For other CNV detection methods this can be (log-) ratios or z-scores.

Arguments

object

An instance of "CNVDetectionResult"

Value

localAssessments returns a "GRanges" object containing the local assessments.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
localAssessments(r)
```

Usage

```r
## S4 method for signature 'CNVDetectionResult'
localAssessments(object)
```

Arguments

object

An instance of "CNVDetectionResult"
makeRobustCNVR

Value

localAssessments returns a "GRanges" object containing the local assessments.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
localAssessments(r)

makeRobustCNVR **Calculates robust CNV regions.**

Description

This generic function calculates robust CNV regions by segmenting the I/NI call per genomic region of an object `CNVDetectionResult-class`.

Usage

```r
## S4 method for signature 'CNVDetectionResult'
makeRobustCNVR(object, robust = 0.5,
    minWidth = 4, ...)
```

Arguments

- `object`: An instance of "CNVDetectionResult"
- `robust`: Robustness parameter. The higher the value, the more samples are required to have a CNV that confirms the CNV region. Setting this parameter to 0 restores the original CNV regions. (Default=0.5)
- `minWidth`: The minimum length measured in genomic regions a CNV region has to span in order to be called. A parameter of the segmentation algorithm. (Default=4).
- `...`: Additional parameters passed to the segmentation algorithm.

Details

This generic function calculates robust CNV regions by segmenting the I/NI call per genomic region of an object `CNVDetectionResult-class`.

cn.mops usually reports a CNV region if at least one individual has a CNV in this region. For some applications it is useful to find more common CNV regions, i.e., regions in which more than one sample has a CNV. The I/NI call measures both signal strength and how many sample show an abnormal copy number, therefore segmentation of the I/NI call can provide robust CNV regions.

Value

`makeRobustCNVR` returns a "CNVDetectionResult" object containing new values in the slot "cnvr".
### Description
Normalize quantitative NGS data in order to make counts comparable over samples, i.e., correcting for different library sizes or coverages. Scales each samples’ reads such that the coverage is even for all samples after normalization.

### Usage
```r
normalizeChromosomes(X, chr, normType = "poisson", sizeFactor = "mean", qu = 0.25, quSizeFactor = 0.75, ploidy)
```

### Arguments
- **X**: Matrix of positive real values, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region. Alternatively this can be a GRanges object containing the read counts as values.
- **chr**: Character vector that has as many elements as "X" has rows. The vector assigns each genomic segment to a reference sequence (chromosome).
- **normType**: Type of the normalization technique. Each samples’ read counts are scaled such that the total number of reads are comparable across samples. If this parameter is set to the value "mode", the read counts are scaled such that each samples’ most frequent value (the "mode") is equal after normalization. Accordingly for the other options are "mean", "median", "poisson", "quant", and "mode". Default = "poisson".
- **sizeFactor**: By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").
- **qu**: Quantile of the normType if normType is set to "quant". Real value between 0 and 1. Default = 0.25.
- **quSizeFactor**: Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.
- **ploidy**: An integer value for each sample or each column in the read count matrix. At least two samples must have a ploidy of 2. Default = "missing".

### Value
A data matrix of normalized read counts with the same dimensions as the input matrix X.
Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
```
data(cn.mops)
X.norm <- normalizeChromosomes(X)
```

Description
This generic function returns the normalized data of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Arguments
- `object` An instance of "CNVDetectionResult".

Value
normalizedData returns a "GRanges" object containing the normalized data.

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
```
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
normalizedData(r)
```

Description
This generic function returns the normalized data of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Usage
```
## S4 method for signature 'CNVDetectionResult'
normalizedData(object)
```
**Arguments**

- object: An instance of "CNVDetectionResult".

**Value**

- normalizedData returns a "GRanges" object containing the normalized data.

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
normalizedData(r)
```

**Description**

Normalize quantitative NGS data in order to make counts comparable over samples. Scales each samples’ reads such that the coverage is even for all samples after normalization.

**Usage**

```r
normalizeGenome(X, normType = "poisson", sizeFactor = "mean", qu = 0.25, quSizeFactor = 0.75, ploidy)
```

**Arguments**

- **X**: Matrix of positive real values, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region. Alternatively this can be a GRanges object containing the read counts as values.

- **normType**: Type of the normalization technique. Each samples’ read counts are scaled such that the total number of reads are comparable across samples. If this parameter is set to the value "mode", the read counts are scaled such that each samples’ most frequent value (the "mode") is equal after normalization. Accordingly for the other options are "mean","median","poisson", "quant", and "mode". Default = "poisson".

- **sizeFactor**: By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

- **qu**: Quantile of the normType if normType is set to "quant". Real value between 0 and 1. Default = 0.25.

- **quSizeFactor**: Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

- **ploidy**: An integer value for each sample or each column in the read count matrix. At least two samples must have a ploidy of 2. Default = "missing".
Value

A data matrix of normalized read counts with the same dimensions as the input matrix X.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
X.norm <- normalizeGenome(X)

params

This generic function returns the parameters of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Description

This generic function returns the parameters of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Arguments

object

An instance of "CNVDetectionResult"

Value

params returns a Granges object containing the parameters.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
params(r)
This generic function returns the parameters of a CNV detection method stored in an instance of "CNVDetectionResult" class.

### Description

This generic function returns the parameters of a CNV detection method stored in an instance of "CNVDetectionResult" class.

### Usage

```r
## S4 method for signature 'CNVDetectionResult'
params(object)
```

### Arguments

- `object`:
  - An instance of "CNVDetectionResult"

### Value

- `params` returns a "GRanges" object containing the parameters.

### Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

### Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
params(r)
```

---

**plot**

Plots a CNVDetectionResult

### Description

Plots read counts, call values and CNV calls in an identified CNV region.

### Usage

```r
## S4 method for signature 'CNVDetectionResult,missing'
plot(x,
     which=NA,
     margin=c(10,10),toFile=FALSE)
```
posteriorProbs

Arguments

x                     An instance of "CNVDetectionResult"
which                 The index of the CNV region to be plotted.
margin               Vector of two positive integers that states how many segments left and right of the CNV region should be included in the plot. Default = c(10,10).
toFile                Logical value whether the output should be plotted to a file. Default = FALSE.

Value

Generates a CNV calling plot.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
posteriorProbs(r)
This generic function returns the posterior probabilities of a CNV detection method stored in an instance of `CNVDetectionResult-class`. The posterior probabilities are represented as a three dimensional array, where the three dimensions are segment, copy number and individual.

### Usage

```r
## S4 method for signature 'CNVDetectionResult'
posteriorProbs(object)
```

### Arguments

- `object` An instance of "CNVDetectionResult"

### Value

`posteriorProbs` returns a three dimensional array.

### Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
posteriorProbs(r)
```

### Description

This function performs the an alternative version of the cn.mops algorithm adapted to a setting of control versus cases setting.
Usage

`referencecn.mops(cases, controls, I = c(0.025, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 8, 16, 32, 64), classes = paste("CN", c(0:8, 16, 32, 64, 128), sep = ""), priorImpact = 1, cyc = 20, parallel = 0, norm = 1, normType = "poisson", sizeFactor = "mean", normQu = 0.25, quSizeFactor = 0.75, upperThreshold = 0.5, lowerThreshold = -0.9, minWidth = 4, segAlgorithm = "DNAcopy", minReadCount = 1, verbose = 1, returnPosterior = FALSE, ...)`

Arguments

cases Either an instance of "GRanges" or a raw data matrix, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region.

controls Either an instance of "GRanges" or a raw data matrix, where columns are interpreted as samples and rows as genomic regions. An entry is the read count of a sample in the genomic region.

I Vector positive real values that contain the expected fold change of the copy number classes. Length of this vector must be equal to the length of the "classes" parameter vector. For human copy number polymorphisms we suggest to use the default I = c(0.025,0.5,1,1.5,2,2.5,3,3.5,4,8,16,32,64).

classes Vector of characters of the same length as the parameter vector "I". One vector element must be named "CN2". The names reflect the labels of the copy number classes. Default = paste("CN",c(0:8,16,32,64,128),sep="").

priorImpact Positive real value that reflects how strong the prior assumption affects the result. The higher the value the more samples will be assumed to have copy number 2. Default = 1.

cyc Positive integer that sets the number of cycles for the algorithm. Usually after less than 15 cycles convergence is reached. Default = 20.

parallel How many cores are used for the computation. If set to zero than no parallelization is applied. Default = 0.

norm The normalization strategy to be used. If set to 0 the read counts are not normalized and cn.mops does not model different coverages. If set to 1 the read counts are normalized. If set to 2 the read counts are not normalized and cn.mops models different coverages. (Default=1).

normType Mode of the normalization technique. Possible values are "mean","min","median","quant", "poisson" and "mode". Read counts will be scaled sample-wise. Default = "poisson".

sizeFactor By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

normQu Real value between 0 and 1. If the "normType" parameter is set to "quant" then this parameter sets the quantile that is used for the normalization. Default = 0.25.

quSizeFactor Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

upperThreshold Positive real value that sets the cut-off for copy number gains. All CNV calling values above this value will be called as "gain". The value should be set close to the log2 of the expected foldchange for copy number 3 or 4. Default = 0.5.
lowerThreshold

Negative real value that sets the cut-off for copy number losses. All CNV calling values below this value will be called as "loss". The value should be set close to the log2 of the expected foldchange for copy number 1 or 0. Default = -0.9.

minWidth

Positive integer that is exactly the parameter "min.width" of the "segment" function of "DNAcopy". minWidth is the minimum number of segments a CNV should span. Default = 3.

segAlgorithm

Which segmentation algorithm should be used. If set to "DNAcopy" circular binary segmentation is performed. Any other value will initiate the use of our fast segmentation algorithm. Default = "DNAcopy".

minReadCount

If all samples are below this value the algorithm will return the prior knowledge. This prevents that the algorithm from being applied to segments with very low coverage. Default=1.

verbose

Flag that decides whether referencecn.mops gives status if (verbose>0) messages. Default=1.

returnPosterior

Flag that decides whether the posterior probabilities should be returned. The posterior probabilities have a dimension of samples times copy number states times genomic regions and therefore consume a lot of memory. Default=FALSE.

... Additional parameters will be passed to the "DNAcopy" or the standard segmentation algorithm.

Value

An instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
referencecn.mops(X[1:200, ],apply(X[1:200, ],1, median))
referencecn.mops(X[1:200, ],apply(X[1:200, ],1, median),parallel=2)

sampleNames

This generic function returns the sample names of a CNV detection method stored in an instance of CNVDetectionResult-class.

Description

This generic function returns the sample names of a CNV detection method stored in an instance of CNVDetectionResult-class.

Arguments

object An instance of "CNVDetectionResult"

Value

sampleNames returns a eturns a "GRanges" object containing the parameters.
Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
sampleNames(r)

Description
This generic function returns the sample names of a CNV detection method stored in an instance of CNVDetectionResult-class.

Usage

```r
## S4 method for signature 'CNVDetectionResult'
sampleNames(object)
```

Arguments

- `object` 
  An instance of "CNVDetectionResult"

Value

`sampleNames` returns a `GRanges` object containing the parameters.

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
sampleNames(r)
**segment**

**Fast segmentation of CNV calls.**

**Description**

Performs a fast segmentation algorithm based on the cyber t test and the t statistics. This is a special version for log-ratios or I/NI calls that are assumed to be centered around 0. For segmentation of data with different characteristics you can a) substract the mean/median/mode from your data or b) use the more general version of this algorithm in the R Bioconductor package "fastseg".

**Usage**

```r
segment(x, alpha = 0.05, segMedianT = NULL, minSeg = 3, eps = 0,
        delta = 20, maxInt = 40, cyberWeight = 50)
```

**Arguments**

- `x` Values to be segmented.
- `alpha` Real value between 0 and 1 is interpreted as the percentage of total points that are considered as initial breakpoints. An integer greater than 1 is interpreted as number of initial breakpoints. Default = 0.05.
- `segMedianT` Vector of length 2. Thresholds on the segment's median. Segments’ medians above the first element are considered as gains and below the second value as losses. If set to NULL the segmentation algorithm tries to determine the thresholds itself. If set to 0 the gain and loss segments are not merged. (Default = NULL).
- `minSeg` Minimum length of segments. Default = 3.
- `eps` Real value greater or equal zero. A breakpoint is only possible between to consecutive values of x that have a distance of at least "eps". Default = 0.
- `delta` Positive integer. A parameter to make the segmentation more efficient. If the statistics of a breakpoint lowers while extending the window, the algorithm extends the windows by "delta" more points until it stops. Default = 20.
- `maxInt` The maximum length of a segment left of the breakpoint and right of the breakpoint that is considered. Default = 40.
- `cyberWeight` The "nu" parameter of the cyber t-test. Default = 50.

**Value**

A data frame containing the segments.

**Author(s)**

Guenter Klambauer <klambauer@bioinf.jku.at>

**Examples**

```r
x <- rnorm(n=500, sd=0.5)
x[150:200] <- rnorm(n=51, mean=3, sd=0.5)
segment(x)
```
Description

This generic function returns segmentation of a CNV detection method stored in an instance of `CNVDetectionResult-class`.

Arguments

- object: An instance of "CNVDetectionResult"

Value

segmentation returns a "GRanges" object containing the segmentation.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:100,1:5])
segmentation(r)
segplot

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

```r
data(cn.mops)
r <- cn.mops(X[1:100,1:5])
segmentation(r)
```

segplot

Visualization of a CNV detection result.

Description

Plots the log normalized read counts and the detected segments as a segmentation plot.

Arguments

- `r` An instance of "CNVDetectionResult"
- `mainCN` The name of the main copy number. That is "CN2" for diploid individuals. For haplocn.mops this should be set to "CN1".
- `sampleIdx` The index of the samples to be plotted. (Default = missing)
- `seqnames` The names of the reference sequence (chromosomes) to be plotted. (Default = missing)
- `segStat` Whether the segment line should display the mean or the median of a segments calls. (Default = "mean").
- `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. (Default = c("green","black").)
segplot method

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
Generates a segmentation plot.

Author(s)
Guenter Klambauer <klambauer@bioinf.jku.at>

Examples
data(cn.mops)
r <- cn.mops(X[1:200, ])
segplot(r,sampleIdx=1)

Description
Visualization of a CNV detection result.

Usage
## S4 method for signature 'CNVDetectionResult'
segplot(r, mainCN = "CN2", sampleIdx, seqnames,
segStat = "mean", plot.type = "s", altcol = TRUE, sbyc.layout,
cbys.nchrom = 1, cbys.layout, include.means = TRUE, zeroline = TRUE,
pt.pch = ".", pt.cex = 3, pt.cols = c("green", "black"),
segcol = "red", zlcol = "grey", ylim, lwd = 3, ...)

Arguments
r
An instance of "CNVDetectionResult"

mainCN
The name of the main copy number. That is "CN2" for diploid individuals. For
haplo.cn.mops this should be set to "CN1".

sampleIdx
The index of the samples to be plotted. (Default = missing)

seqnames
The names of the reference sequence (chromosomes) to be plotted. (Default =
missing)

segStat
Whether the segment line should display the mean or the median of a segments
calls. (Default = "mean").

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.nchrom the number of chromosomes per page in the layout. (Default = 1).

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).

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. (Default = c("green","black").)

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

Generates a segmentation plot.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
r <- cn.mops(X[1:200, ])
segplot(r,sampleIdx=1)
show

Displays the result object of a copy number detection method.

Description

Displays method for S4 class \texttt{CNVDetectionResult}

Usage

\begin{verbatim}
## S4 method for signature 'CNVDetectionResult'
show(object)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{object} An instance of a "CNVDetectionResult".
\end{itemize}

Value

Displays the result object of a CNV detection method.

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

---

\texttt{singlecn.mops}

Copy number detection in NGS data with in a setting in which only one sample is available

Description

This function performs the an alternative version of the \texttt{cn.mops} algorithm adapted to a setting of a single sample.

Usage

\begin{verbatim}
singlecn.mops(x, I = c(0.025, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4),
classes = c("CN0", "CN1", "CN2", "CN3", "CN4", "CN5", "CN6", "CN7", "CN8"),
priorImpact = 1, cyc = 20, parallel = 0, norm = 1,
normType = "poisson", sizeFactor = "mean", normQu = 0.25,
quSizeFactor = 0.75, upperThreshold = 0.5, lowerThreshold = -0.9,
minWidth = 3, segAlgorithm = "fast", minReadCount = 1,
returnPosterior = FALSE, ...)
\end{verbatim}
Arguments

x Either an instance of "GRanges" or a raw data matrix with one column or a vector of read counts. An entry is the read count of the sample in the genomic region.

I Vector positive real values that contain the expected fold change of the copy number classes. Length of this vector must be equal to the length of the "classes" parameter vector. For human copy number polymorphisms we suggest to use the default I = c(0.025,0.5,1,1.5,2,2.5,3,3.5,4).

classes Vector of characters of the same length as the parameter vector "I". One vector element must be named "CN2". The names reflect the labels of the copy number classes. Default = c("CN0","CN1","CN2","CN3","CN4","CN5","CN6","CN7","CN8").

priorImpact Positive real value that reflects how strong the prior assumption affects the result. The higher the value the more samples will be assumed to have copy number 2. Default = 1.

cyc Positive integer that sets the number of cycles for the algorithm. Usually after less than 15 cycles convergence is reached. Default = 20.

parallel How many cores are used for the computation. If set to zero than no parallelization is applied. Default = 0.

norm The normalization strategy to be used. If set to 0 the read counts are not normalized and cn.mops does not model different coverages. If set to 1 the read counts are normalized. If set to 2 the read counts are not normalized and cn.mops models different coverages. (Default=1).

normType Mode of the normalization technique. Possible values are "mean","min","median","quant", "poisson" and "mode". Read counts will be scaled sample-wise. Default = "poisson".

sizeFactor By this parameter one can decide to how the size factors are calculated. Possible choices are the the mean, median or mode coverage ("mean", "median", "mode") or any quantile ("quant").

normQu Real value between 0 and 1. If the "normType" parameter is set to "quant" then this parameter sets the quantile that is used for the normalization. Default = 0.25.

quSizeFactor Quantile of the sizeFactor if sizeFactor is set to "quant". 0.75 corresponds to "upper quartile normalization". Real value between 0 and 1. Default = 0.75.

upperThreshold Positive real value that sets the cut-off for copy number gains. All CNV calling values above this value will be called as "gain". The value should be set close to the log2 of the expected foldchange for copy number 3 or 4. Default = 0.5.

lowerThreshold Negative real value that sets the cut-off for copy number losses. All CNV calling values below this value will be called as "loss". The value should be set close to the log2 of the expected foldchange for copy number 1 or 0. Default = -0.9.

minWidth Positive integer that is exactly the parameter "min.width" of the "segment" function of "DNAcopy". minWidth is the minimum number of segments a CNV should span. Default = 3.

segAlgorithm Which segmentation algorithm should be used. If set to "DNAcopy" circular binary segmentation is performed. Any other value will initiate the use of our fast segmentation algorithm. Default = "fast".

minReadCount If all samples are below this value the algorithm will return the prior knowledge. This prevents that the algorithm from being applied to segments with very low coverage. Default=1.
returnPosterior

Flag that decides whether the posterior probabilities should be returned. The posterior probabilities have a dimension of samples times copy number states times genomic regions and therefore consume a lot of memory. Default=FALSE.

Additional parameters will be passed to the "DNAcopy" or the standard segmentation algorithm.

Value

An instance of "CNVDetectionResult".

Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

Examples

data(cn.mops)
singlecn.mops(XRanges[,1])
**XRanges**

_A simulated data set for CNV detection from NGS data._

**Description**

This data set gives the read counts of 40 samples in 5000 genomic locations. The rows correspond to genomic segments of 25kbp length and the columns to samples. An entry is the number of reads that map to the specific segment of the sample. The "GRanges" object contains the name of the reference sequence, start and end position of the genomic segments. The simulated data contains CNVs given in the **CNVRanges** object. It was generated using distributions of read counts as they appear in real sequencing experiments. CNVs were implanted under the assumption that the expected read count is linear dependent on the copy number (e.g. in a certain genomic we expect \( \lambda \) reads for copy number 2, then we expect \( 2 \cdot \lambda \) reads for copy number 4).

**Usage**

`XRanges`

**Format**

A GRanges object with 5000 rows and 40 value columns across 1 space.

**Source**


**References**

### Index

**Topic classes**
- CNVDetectionResult-class, 7

**Topic datasets**
- CNVRanges, 9
- exomeCounts, 13
- X, 42
- XRanges, 43

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calcFractionalCopyNumbers</td>
<td>3</td>
</tr>
<tr>
<td>calcFractionalCopyNumbers,CNVDetectionResult-method</td>
<td>5</td>
</tr>
<tr>
<td>calcIntegerCopyNumbers</td>
<td>4</td>
</tr>
<tr>
<td>calcIntegerCopyNumbers,CNVDetectionResult-method</td>
<td>6</td>
</tr>
<tr>
<td>cn.mops</td>
<td>5</td>
</tr>
<tr>
<td>CNVDetectionResult, 7, 40</td>
<td></td>
</tr>
<tr>
<td>CNVDetectionResult/(CNVDetectionResult-class)</td>
<td>7</td>
</tr>
<tr>
<td>CnvDetectionResult</td>
<td>8</td>
</tr>
<tr>
<td>CnvDetectionResult/(CNVDetectionResult-class)</td>
<td>7</td>
</tr>
<tr>
<td>cnvDetectionResult</td>
<td>8</td>
</tr>
<tr>
<td>CNVDetectionResult-class, 7, 8-11, 16, 19-23, 26, 28-31, 33, 34, 36</td>
<td></td>
</tr>
<tr>
<td>cnvr</td>
<td>8</td>
</tr>
<tr>
<td>cnvr,CNVDetectionResult-method, 9</td>
<td></td>
</tr>
<tr>
<td>CNVRanges, 9, 42, 43</td>
<td></td>
</tr>
<tr>
<td>cnvs</td>
<td>10</td>
</tr>
<tr>
<td>cnvs,CNVDetectionResult-method, 11</td>
<td></td>
</tr>
<tr>
<td>exomecn.mops</td>
<td>11</td>
</tr>
<tr>
<td>exomeCounts</td>
<td>13</td>
</tr>
<tr>
<td>getReadCountsFromBAM</td>
<td>14</td>
</tr>
<tr>
<td>getSegmentReadCountsFromBAM</td>
<td>15</td>
</tr>
<tr>
<td>gr</td>
<td>16</td>
</tr>
<tr>
<td>gr,CNVDetectionResult-method</td>
<td>17</td>
</tr>
<tr>
<td>haplocn.mops</td>
<td>17</td>
</tr>
<tr>
<td>individualCall</td>
<td>19</td>
</tr>
<tr>
<td>individualCall,CNVDetectionResult-method</td>
<td>19</td>
</tr>
<tr>
<td>iniCall</td>
<td>20</td>
</tr>
<tr>
<td>iniCall,CNVDetectionResult-method</td>
<td>21</td>
</tr>
<tr>
<td>integerCopyNumber</td>
<td>21</td>
</tr>
<tr>
<td>integerCopyNumber,CNVDetectionResult-method</td>
<td>22</td>
</tr>
<tr>
<td>localAssessments</td>
<td>23</td>
</tr>
<tr>
<td>localAssessments,CNVDetectionResult-method</td>
<td>23</td>
</tr>
<tr>
<td>makeRobustCNVR</td>
<td>24</td>
</tr>
<tr>
<td>makeRobustCNVR,CNVDetectionResult-method</td>
<td>25</td>
</tr>
<tr>
<td>normalizeChromosomes</td>
<td>25</td>
</tr>
<tr>
<td>normalizedData</td>
<td>26</td>
</tr>
<tr>
<td>normalizedData,CNVDetectionResult-method</td>
<td>26</td>
</tr>
<tr>
<td>normalizeGenome</td>
<td>27</td>
</tr>
<tr>
<td>params</td>
<td>28</td>
</tr>
<tr>
<td>params,CNVDetectionResult-method</td>
<td>29</td>
</tr>
<tr>
<td>plot</td>
<td>29</td>
</tr>
<tr>
<td>plot,CNVDetectionResult,missing-method</td>
<td>29</td>
</tr>
<tr>
<td>plot-methods (plot)</td>
<td>29</td>
</tr>
<tr>
<td>posteriorProbs</td>
<td>30</td>
</tr>
<tr>
<td>posteriorProbs,CNVDetectionResult-method</td>
<td>31</td>
</tr>
<tr>
<td>referencecn.mops</td>
<td>31</td>
</tr>
<tr>
<td>sampleNames</td>
<td>33</td>
</tr>
<tr>
<td>sampleNames,CNVDetectionResult-method</td>
<td>34</td>
</tr>
<tr>
<td>segment</td>
<td>35</td>
</tr>
<tr>
<td>segmentation</td>
<td>36</td>
</tr>
<tr>
<td>segmentation,CNVDetectionResult-method</td>
<td>36</td>
</tr>
<tr>
<td>segplot</td>
<td>37</td>
</tr>
<tr>
<td>segplot,CNVDetectionResult-method</td>
<td>38</td>
</tr>
<tr>
<td>show</td>
<td>40</td>
</tr>
<tr>
<td>show,CNVDetectionResult-method</td>
<td>40</td>
</tr>
<tr>
<td>show-methods (show)</td>
<td>40</td>
</tr>
<tr>
<td>singlecn.mops</td>
<td>40</td>
</tr>
</tbody>
</table>
INDEX

X, 42
XRanges, 9, 43