# Package 'cn.mops'

March 26, 2013

**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++.

**biocViews** HighThroughputSequencing, Sequencing, Bioinformatics,CopyNumberVariants

**Version** 1.4.12

**Date** 2013-03-08

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

**Depends** R (>= 2.12), BiocGenerics, Biobase, IRanges, GenomicRanges

**Imports** methods, graphics, BiocGenerics, IRanges, Rsamtools,

**Suggests** snow, DNAcopy

**LazyLoad** yes

**Collate**
'AllClasses.R' 'cn.mops.R' 'getReadCountsFromBAM.R' 'normalizeChromosomes.R' 'normalizeGenome.R' 'plot-
methods.R' 'show-
methods.R' 'segment.R' 'zzz.R' 'AllGenerics.R' 'methodsAccess.R' 'getSegmentReadCountsFromBAM.R' 'haplocn.n

# R topics documented:

---

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)
```

---

cn.mops                    *Copy number detection in NGS data.*

---

## Description

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

## Usage

```
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,
    normType = "poisson", normQu = 0.25, norm = TRUE,
    upperThreshold = 0.5, 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 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. The package "snow" has to be installed for this option. Default = 0. |
| 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". |

| | |
|---|---|
| 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. |
| norm | Logical that indicates whether normalization should be applied or not. Default = TRUE. |
| 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)
cn.mops(XRanges)
```

---

CNVDetectionResult-class
                        *Class "CNVDetectionResult"*

---

## 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"): ...

**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 eturns a "GRanges" object containing the CNV regions.

### Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

### Examples

```
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

Guenter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arne Clevert, Andreas Mitterecker, Ulrich Bodenhofer, Sepp Hochreiter. *cn.MOPS: mixture of Poissons for discovering copy number variations in next generation sequencing data with a low false discovery rate.* Nucleic Acids Research 2012 40(9); doi:10.1093/nar/gks003.

---

| | |
|---|---|
| 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 eturns 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)
```

---

| | |
|---|---|
| exomecn.mops | *Copy number detection in exome sequencing data.* |

---

## Description

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

## Usage

```
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,
    normType = "poisson", normQu = 0.25, norm = TRUE,
    upperThreshold = 0.5, lowerThreshold = -0.8,
    minWidth = 5, segAlgorithm = "fast", minReadCount = 1,
    ...)
```

## 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. The package "snow" has to be installed for this option. Default = 0. |
| 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". |
| 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. |
| norm | Logical that indicates whether normalization should be applied or not. Default = TRUE. |
| 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. |
| ... | 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)
```

---

exomeCounts                 *Read counts from exome sequencing for CNV detection*

---

## 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

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

## References

Guenter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arne Clevert, Andreas Mitterecker, Ulrich Bodenhofer, Sepp Hochreiter. *cn.MOPS: mixture of Poissons for discovering copy number variations in next generation sequencing data with a low false discovery rate.* Nucleic Acids Research 2012 40(9); doi:10.1093/nar/gks003.

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

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. Note that the function is much faster, if the BAM files have an index file. The index file is assumed to be in the same folder and have an identical file name except that ".bai" is appended.

This function can also be run in a parallel version.

**Usage**

```
getReadCountsFromBAM(BAMFiles, sampleNames, refSeqName,
    WL, mode, parallel = 0)
```

**Arguments**

| | |
|---|---|
| BAMFiles | BAMFiles |
| sampleNames | The corresponding sample names to the BAM Files. |
| refSeqName | Name 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. |
| WL | 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. If not given, cn.mops will try to find an appropiate window length. |
| mode | Possible values are "paired" and "unpaired", whether the mapping algorithm was using a "paired" or "unpaired" strategy. |
| parallel | The number of parallel processes to be used for this function. Default=0. |

**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**

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

---

getSegmentReadCountsFromBAM

*Calculation of read counts from BAM files for predefined segments.*

---

**Description**

Generates the read counts from BAM Files for predefined segments. This is the appropiate choice for exome sequencing data, where the bait regions, target regions or exons are the predefined segments. These counts are necessary for CNV detection methods based on depth of coverage information. Note that the function is much faster, if the BAM files have an index file. The index file is assumed to be in the same folder and have an identical file name except that ".bai" is appended.

This function can also be run in a parallel version.

**Usage**

```
getSegmentReadCountsFromBAM(BAMFiles, GR, sampleNames,
  mode, parallel = 0)
```

**Arguments**

| | |
|---|---|
| BAMFiles | BAMFiles |
| sampleNames | The corresponding sample names to the BAM Files. |
| GR | A genomic ranges object that contains the genomic coordinates of the segments. |
| mode | Possible values are "paired" and "unpaired", whether the mapping algorithm was using a "paired" or "unpaired" strategy. |
| parallel | The number of parallel processes to be used for this function. Default=0. |

**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**

```
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,mode="unpaired")
```

---

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

---

### Description

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

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

---

| haplocn.mops | *Copy number detection in NGS data of haploid samples.* |
| --- | --- |

---

### Description

Performs the cn.mops algorithm for copy number detection in NGS data adjusted to haploid genomes. It is assumed that the normal state is copy number 1. This is an experimental method at the moment.

### Usage

```
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,
  normType = "poisson", normQu = 0.25, norm = TRUE,
  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. The package "snow" has to be installed for this option. Default = 0. |
| 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". |
| 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. |
| norm | Logical that indicates whether normalization should be applied or not. Default = TRUE. |
| 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. |
| ... | Additional parameters will be passed to the "DNAcopy" or the standard segmentation algorithm. |
| 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**

```
data(cn.mops)
haplocn.mops(XRanges[1:200, ])
```

---

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)
```

iniCall | *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).

## 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])
iniCall(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"

## Value

integerCopyNumber returns a eturns 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**

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

| normalizeChromosomes | *Normalization of NGS data. 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.* |
|---|---|

### Description

Normalization of NGS data.

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

```
normalizeChromosomes(X, chr, normType = "poisson",
  qu = 0.25, 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 is equal after normalization. By this parameter one can decide to which coverage (i.e. total reads) the read counts should be normalized. Possible choices are the minimal coverage ("min"), the mean or median coverage ("mean", "median") or any quantile ("quant"). 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. If the parameter is set to "poisson" the values are scaled such that the distribution is (rowwise) close to a Poisson distribution. Possible values are "mean","min","median","quant","poisson, and "mode". Default = "poisson". |
| qu | Real value between 0 and 1. Default = 0.25. |
| 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)
```

---

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

---

**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)
```

---

normalizeGenome            *Normalization of NGS data*

---

**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**

```
normalizeGenome(X, normType = "poisson", qu = 0.25,
  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 | normType Type of the normalization technique. Each samples' read counts are scaled such that the total number of reads is equal after normalization. By this parameter one can decide to which coverage (i.e. total reads) the read counts should be normalized. Possible choices are the minimal coverage ("min"), the mean or median coverage ("mean", "median") or any quantile ("quant"). 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. If the parameter is set to "poisson" the values are scaled such that the distribution is (rowwise) close to a Poisson distribution. Possible values are "mean","min","median","quant","poisson" and "mode". Default = "poisson". |
| qu | Real value between 0 and 1. Default = 0.25. |
| 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 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])
params(r)
```

---

plot                    *Plots a CNVDetectionResult*

---

**Description**

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

**Usage**

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

**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>

| posteriorProbs | *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.* |
|---|---|

### Description

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.

### Arguments

object          An instance of "CNVDetectionResult"

### Value

posteriorProbs returns a three dimensional array.

### Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

### Examples

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

| 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)
```

---

| segment | *Fast segmentation of CNV calls.* |
|---|---|

---

## Description

Performs a fast segmentation algorithm based on the cyber t test and the t statistics.

## Usage

```
segment(x, alpha = 0.05, segMedianT = 0, minSeg = 3,
    eps = 0, delta = 20, maxInt = 40, squashing = 0,
    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. |
| 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. |
| squashing | An experimental parameter that squashes the values "x" before segmentation. Should be left to zero, which means that squashing is not performed. Default = 0. |
| 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**

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

---

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

---

**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 eturns 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 | *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) |
| 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").) |
| 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 CNVDetectionResult

## Usage

```
## S4 method for signature 'CNVDetectionResult'
show(object)
```

## Arguments

object          An instance of a "CNVDetectionResult".

## Value

Displays the result object of a CNV detection method.

## Author(s)

Guenter Klambauer <klambauer@bioinf.jku.at>

| X | *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 rownames contain the information of the genomic location - they are in the format refseqname_startposition_endposition. 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

```
X
```

## Format

A data matrix of 5000 rows and 40 columns.

## Source

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

**References**

Guenter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arne Clevert, Andreas Mitterecker, Ulrich Bodenhofer, Sepp Hochreiter. *cn.MOPS: mixture of Poissons for discovering copy number variations in next generation sequencing data with a low false discovery rate.* Nucleic Acids Research 2012 40(9); doi:10.1093/nar/gks003.

---

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**

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

**References**

Guenter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arne Clevert, Andreas Mitterecker, Ulrich Bodenhofer, Sepp Hochreiter. *cn.MOPS: mixture of Poissons for discovering copy number variations in next generation sequencing data with a low false discovery rate.* Nucleic Acids Research 2012 40(9); doi:10.1093/nar/gks003.

# Index