# Package 'QDNAseq'

December 2, 2025

**Version** 1.46.0

**Title** Quantitative DNA Sequencing for Chromosomal Aberrations

**Depends** R (>= 3.1.0)

**Imports** graphics, methods, stats, utils, BiocGenerics, Biobase (>= 2.18.0), CGHbase (>= 1.18.0), CGHcall (>= 2.18.0), DNAcopy (>= 1.32.0), Seqinfo, GenomicRanges (>= 1.20), IRanges (>= 2.2), matrixStats (>= 0.60.0), R.utils (>= 2.9.0), Rsamtools (>= 1.20), future.apply (>= 1.8.1)

**Suggests** BiocStyle (>= 1.8.0), BSgenome (>= 1.38.0), digest (>= 0.6.20), GenomeInfoDb (>= 1.6.0), future (>= 1.22.1), parallelly (>= 1.28.1), R.cache (>= 0.13.0), QDNAseq.hg19, QDNAseq.mm10

**Description** Quantitative DNA sequencing for chromosomal aberrations. The genome is divided into non-overlapping fixed-sized bins, number of sequence reads in each counted, adjusted with a simultaneous two-dimensional loess correction for sequence mappability and GC content, and filtered to remove spurious regions in the genome. Downstream steps of segmentation and calling are also implemented via packages DNAcopy and CGHcall, respectively.

**biocViews** CopyNumberVariation, DNASeq, Genetics, GenomeAnnotation, Preprocessing, QualityControl, Sequencing

**License** GPL

**URL** https://github.com/ccagc/QDNAseq

**BugReports** https://github.com/ccagc/QDNAseq/issues

**RoxygenNote** 7.3.2

**git_url** https://git.bioconductor.org/packages/QDNAseq

**git_branch** RELEASE_3_22

**git_last_commit** 00b06a1

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2025-12-01

**Author** Ilari Scheinin [aut],
Daoud Sie [aut, cre],
Henrik Bengtsson [aut],
Erik van Dijk [ctb]

**Maintainer** Daoud Sie <d.sie@vumc.nl>

# Contents



---

`QDNAseq-package` *Package QDNAseq*

---

## Description

Quantitative DNA sequencing for chromosomal aberrations. The genome is divided into non-overlapping fixed-sized bins, number of sequence reads in each counted, adjusted with a simultaneous two-dimensional loess correction for sequence mappability and GC content, and filtered to remove spurious regions in the genome. Downstream steps of segmentation and calling are also implemented via packages DNAcopy and CGHcall, respectively.

## Details

A package to detect chromosomal aberrations from whole-genome sequencing data. `QDNAseqReadCounts` and `QDNAseqCopyNumbers` classes are used as the main data structures.

### How to cite this package

Whenever using this package, please cite: Scheinin I, Sie D, Bengtsson H, van de Wiel MA, Olshen AB, van Thuijl HF, van Essen HF, Eijk PP, Rustenburg F, Meijer GA, Reijneveld JC, Wesseling P, Pinkel D, Albertson DG, Ylstra B (2014). "DNA copy number analysis of fresh and formalin-fixed specimens by shallow whole-genome sequencing with identification and exclusion of problematic regions in the genome assembly." _Genome Research_, *24*, 2022-2032.

### License

This package is licensed under GPL.

### Author(s)

Ilari Scheinin

---

`addPhenodata` *Adds phenotype data from a file to a QDNAseqReadCounts or a QDNAseqCopyNumbers object*

---

### Description

Adds phenotype data from a file to a QDNAseqReadCounts or a QDNAseqCopyNumbers object.

### Usage

```
addPhenodata(object, phenofile)
```

### Arguments

| | |
|---|---|
| `object` | A `QDNAseqReadCounts` or `QDNAseqCopyNumbers` object. |
| `phenofile` | A file name with phenotypic data for samples in `object`. |

### Value

Returns a `QDNAseqReadCounts` or `QDNAseqCopyNumbers` object with phenotype data added.

### Author(s)

Ilari Scheinin

### Examples

```
data(LGG150)
readCounts <- LGG150
## Not run:
readCounts <- addPhenodata(readCounts, "phenodata.txt")

## End(Not run)
```

`applyFilters` *Adjusts the filtering on which bins are used*

## Description

Adjusts the filtering on which bins are used.

## Usage

```
applyFilters(object, residual=TRUE, blacklist=TRUE, mappability=NA, bases=NA,
  chromosomes=c("X", "Y"), verbose=getOption("QDNAseq::verbose", TRUE))
```

## Arguments

| | |
|---|---|
| `object` | A `QDNAseqReadCounts` object. |
| `residual` | Either a `logical` specifying whether to filter based on loess residuals of the calibration set, or if a `numeric`, the cutoff as number of standard deviations estimated with `madDiff` to use for. Default is `TRUE`, which corresponds to 4.0 standard deviations. |
| `blacklist` | Either a `logical` specifying whether to filter based on overlap with blacklisted regions, or if numeric, the maximum percentage of overlap allowed. Default is `TRUE`, which corresponds to no overlap allowed (i.e. value of 0). |
| `mappability` | A `numeric` in $[0, 100]$ to specify filtering out bins with mappabilities lower than the number specified. NA (default) or `FALSE` will not filter based on mappability. |
| `bases` | A `numeric` specifying the minimum percentage of characterized bases (not Ns) in the reference genome sequence. NA (default) or `FALSE` will not filter based on uncharacterized bases. |
| `chromosomes` | A `character` vector specifying which chromosomes to filter out. Defaults to the sex chromosomes and mitochondria, i.e. `c("X", "Y", "MT")`. |
| `verbose` | If `TRUE`, verbose messages are produced. |

## Value

Returns a `QDNAseqReadCounts` object with updated filtering.

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
```

`binReadCounts` *Calculate binned read counts from a set of BAM files*

## Description

Calculate binned read counts from a set of BAM files.

## Usage

```
binReadCounts(bins, bamfiles=NULL, path=NULL, ext="bam", bamnames=NULL, phenofile=NULL,
 chunkSize=NULL, cache=getOption("QDNAseq::cache", FALSE), force=!cache, isPaired=NA,
  isProperPair=NA, isUnmappedQuery=FALSE, hasUnmappedMate=NA, isMinusStrand=NA,
 isMateMinusStrand=NA, isFirstMateRead=NA, isSecondMateRead=NA, isSecondaryAlignment=NA,
 isNotPassingQualityControls=FALSE, isDuplicate=FALSE, minMapq=37, pairedEnds=NULL,
  verbose=getOption("QDNAseq::verbose", TRUE))
```

## Arguments

| | |
|---|---|
| `bins` | A data.frame or an `AnnotatedDataFrame` object containing bin annotations. |
| `bamfiles` | A character vector of (BAM) file names. If NULL (default), all files with extension ext, are read from directory path. |
| `path` | If bamfiles is NULL, directory path to read input files from. Defaults to the current working directory. |
| `ext` | File name extension of input files to read, default is "bam". |
| `bamnames` | An optional character vector of sample names. Defaults to file names with extension ext removed. |
| `phenofile` | An optional character(1) specifying a file name for phenotype data. |
| `chunkSize` | An optional integer specifying the chunk size (nt) by which to process the bam file. |
| `cache` | Whether to read and write intermediate cache files, which speeds up subsequent analyses of the same files. Requires packages R.cache and digest (both available on CRAN) to be installed. Defaults to getOption("QDNAseq::cache", FALSE). |
| `force` | When using the cache, whether to force reading input data from the BAM files even when an intermediate cache file is present. |
| `isPaired` | A logical(1) indicating whether unpaired (FALSE), paired (TRUE), or any (NA, default) read should be returned. |
| `isProperPair` | A logical(1) indicating whether improperly paired (FALSE), properly paired (TRUE), or any (NA, default) read should be returned. A properly paired read is defined by the alignment algorithm and might, e.g., represent reads aligning to identical reference sequences and with a specified distance. |
| `isUnmappedQuery` | A logical(1) indicating whether unmapped (TRUE), mapped (FALSE, default), or any (NA) read should be returned. |
| `hasUnmappedMate` | A logical(1) indicating whether reads with mapped (FALSE), unmapped (TRUE), or any (NA, default) mate should be returned. |
| `isMinusStrand` | A logical(1) indicating whether reads aligned to the plus (FALSE), minus (TRUE), or any (NA, default) strand should be returned. |

isMateMinusStrand
: A logical(1) indicating whether mate reads aligned to the plus (FALSE), minus (TRUE), or any (NA, default) strand should be returned.

isFirstMateRead
: A logical(1) indicating whether the first mate read should be returned (TRUE) or not (FALSE), or whether mate read number should be ignored (NA, default).

isSecondMateRead
: A logical(1) indicating whether the second mate read should be returned (TRUE) or not (FALSE), or whether mate read number should be ignored (NA, default).

isSecondaryAlignment
: A logical(1) indicating whether alignments that are primary (FALSE), are not primary (TRUE) or whose primary status does not matter (NA, default) should be returned. A non-primary alignment ("secondary alignment" in the SAM specification) might result when a read aligns to multiple locations. One alignment is designated as primary and has this flag set to FALSE; the remainder, for which this flag is TRUE, are designated by the aligner as secondary.

isNotPassingQualityControls
: A logical(1) indicating whether reads passing quality controls (FALSE, default), reads not passing quality controls (TRUE), or any (NA) read should be returned.

isDuplicate
: A logical(1) indicating that un-duplicated (FALSE, default), duplicated (TRUE), or any (NA) reads should be returned. 'Duplicated' reads may represent PCR or optical duplicates.

minMapq
: If quality scores exists, the minimum quality score required in order to keep a read, otherwise all reads are kept.

pairedEnds
: A boolean value or vector specifying whether the BAM files contain paired-end data or not. Only affects the calculation of the expected variance.

verbose
: If TRUE, verbose messages are produced.

## Value

Returns a QDNAseqReadCounts object with assay data element counts containing the binned read counts as non-negative integers.

## Author(s)

Ilari Scheinin, Daoud Sie

## Examples

```
## Not run: # read all files from the current directory with names ending in .bam
bins <- getBinAnnotations(15)
readCounts <- binReadCounts(bins)

## End(Not run)
```

`callBins` *Call aberrations from segmented copy number data*

## Description

Call aberrations from segmented copy number data.

## Usage

```
callBins(object, organism=c("human", "other"), method=c("CGHcall", "cutoff"),
 cutoffs=log2(c(deletion = 0.5, loss = 1.5, gain = 2.5, amplification = 10)/2), ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class QDNAseqCopyNumbers |
| `organism` | Either "human" or "other", see manual page for `CGHcall` for more details. This is only used for chromosome arm information when "prior" is set to "all" or "auto" (and samplesize > 20). Ignored when `method` is not "CGHcall". |
| `method` | Calling method to use. Options currently implemented are: "CGHcall" or "cutoff". |
| `cutoffs` | When method="cutoff", a numeric vector of (log2-transformed) thresholds to use for calling. At least one positive and one negative value must be provided. The smallest positive value is used as the cutoff for calling gains, and the negative value closest to zero is used as the cutoff for losses. If a second positive value is provided, it is used as the cutoff for amplifications. And if a second negative value is provided, it is used as the cutoff for homozygous deletions. |
| `...` | Additional arguments passed to `CGHcall`. |

## Details

By default, chromosomal aberrations are called with **CGHcall**. It has been developed for the analysis of series of cancer samples, and uses a model that contains both gains and losses. If used on a single sample, or especially only on a subset of chromosomes, or especially on a single non-cancer sample, it may fail, but method "cutoff" can be used instead.

When using method "cutoff", the default values assume a uniform cell population and correspond to thresholds of (assuming a diploid genome) 0.5, 1.5, 2.5, and 10 copies to distinguish between homozygous deletions, (hemizygous) losses, normal copy number, gains, and amplifications, respectively. When using with cancer samples, these values might require adjustments to account for tumor cell percentage.

## Value

Returns an object of class `QDNAseqCopyNumbers` with calling results added.

## Author(s)

Ilari Scheinin

## See Also

Internally, `CGHcall` and `ExpandCGHcall` of the **CGHcall** package are used when method="CGHcall".

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
copyNumbersSegmented <- segmentBins(copyNumbersSmooth)
copyNumbersSegmented <- normalizeSegmentedBins(copyNumbersSegmented)
copyNumbersCalled <- callBins(copyNumbersSegmented)
```

---

# compareToReference *Divide binned read counts with those of reference samples*

---

## Description

Divide binned read counts with those of reference samples.

## Usage

```
compareToReference(object, references, force=FALSE)
```

## Arguments

| | |
|---|---|
| object | An object of class QDNAseqCopyNumbers. |
| references | A numeric vector of indexes of the reference sample. Must be the same length as there are samples in object. When NA, the sample will be kept as is. When FALSE, the sample will be removed from the output. As an example, object contains three samples: tumor1, tumor2, and normal2. There is no reference for tumor1, but normal2 is a matched normal sample from the same patient as tumor2. The keep tumor1 as is, but to divide tumor2 with normal2, argument references should be c(NA, 3, FALSE). |
| force | Whether to force the operation even when downstream data will be lost. |

## Value

Returns a QDNAseqCopyNumbers object with the desired samples divided by the signal of their reference samples.

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
# Note: the following command will compare the sample to itself, which
# does not really make sense:
tumorVsNormal <- compareToReference(copyNumbersSmooth, c(1))
```

---

`correctBins` *Correct binned read counts for GC content and mappability*

---

## Description

Correct binned read counts for GC content and mappability.

## Usage

```
correctBins(object, fit=NULL, method="ratio", adjustIncompletes=TRUE, ...)
```

## Arguments

| | |
|---|---|
| `object` | An `QDNAseqReadCounts` object with `counts` data. |
| `fit` | An optional matrix of values to use for the correction. If NULL (default), assay data `fit` from object is used. If it is missing, it is generated with a call to `estimateCorrection`(). |
| `method` | A character(1) string specifying the correction method. `ratio` (default) divides `counts` with `fit`. `median` calculates the median `fit`, and defines the correction for bins with GC content `gc` and mappability `map` as `median(fit) - fit(gc,map)`, which is added to `counts`. Method `none` leaves `counts` untouched. |
| `adjustIncompletes` | A boolean(1) specifying whether `counts` for bins with uncharacterized nucleotides (N's) in their reference genome sequence should be adjusted by dividing them with the percentage of characterized (A, C, G, T) nucleotides. Defaults to `TRUE`. |
| `...` | Additional arguments passed to `estimateCorrection`(). |

## Value

Returns a `QDNAseqCopyNumbers` object with assay data element `copynumber`.

## Author(s)

Ilari Scheinin

## See Also

Internally, `loess` is used to fit the regression model.

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
```

---

`createBins` *Builds bin annotation data for a particular bin size*

---

## Description

Builds bin annotation data for a particular bin size.

## Usage

```
createBins(bsgenome, binSize, ignoreMitochondria=TRUE, excludeSeqnames=NULL,
  verbose=getOption("QDNAseq::verbose", TRUE))
```

## Arguments

| | |
|---|---|
| `bsgenome` | A BSgenome package. |
| `binSize` | A `numeric` scalar specifying the width of the bins in units of kbp (1000 base pairs), e.g. `binSize=15` corresponds to 15 kbp bins. |
| `ignoreMitochondria` | Whether to ignore the mitochondria, defined as chromosomes named 'chrM', 'chrMT', 'M', or 'MT'. |
| `excludeSeqnames` | Character vector of seqnames which should be ignored. |
| `verbose` | If `TRUE`, verbose messages are produced. |

## Value

Returns a `data.frame` with columns `chromosome`, `start`, `end`, `bases`, and `gc`, which correspond to the chromosome name, positions of the first and last base pair in the bin, the percentage of characterized nucleotides (A, C, G, or T, i.e. non-N), and GC content (percentage of C and G nucleotides of non-N nucleotides).

## Parallel processing

The **future** is used parallelize the following functions:

- `createBins()` - parallelizes binned GC content across chromosomes
- `calculateBlacklist()` - parallelizes overlap counts across bins)

## Author(s)

Ilari Scheinin

## See Also

`getBinAnnotations`().

## Examples

```
## Not run: # NOTE: These take a very long time to run.
library(BSgenome.Hsapiens.UCSC.hg19)
bins <- createBins(BSgenome.Hsapiens.UCSC.hg19, 15)
bins$mappability <- calculateMappability(bins,
    bigWigFile='/path/to/wgEncodeCrgMapabilityAlign50mer.bigWig',
    bigWigAverageOverBed='/path/to/bigWigAverageOverBed')
bins$blacklist <- calculateBlacklist(bins,
    bedFiles=c('/path/to/wgEncodeDacMapabilityConsensusExcludable.bed',
    '/path/to/wgEncodeDukeMapabilityRegionsExcludable.bed'))
bins$residual <- iterateResiduals(readCountsG1K)

## End(Not run)
```

---

`estimateCorrection` *Estimate correction to read counts for GC content and mappability*

---

## Description

Estimate correction to read counts for GC content and mappability.

## Usage

```
estimateCorrection(object, span=0.65, family="symmetric", adjustIncompletes=TRUE,
  maxIter=1, cutoff=4, variables=c("gc", "mappability"), ...)
```

## Arguments

| | |
|---|---|
| `object` | An `QDNAseqReadCounts` object with `counts` data. |
| `span` | For `loess`, the parameter alpha which controls the degree of smoothing. |
| `family` | For `loess`, if "gaussian" fitting is by least-squares, and if "symmetric" a re-descending M estimator is used with Tukey's biweight function. |
| `adjustIncompletes` | A boolean(1) specifying whether `counts` for bins with uncharacterized nucleotides (N's) in their reference genome sequence should be adjusted by dividing them with the percentage of characterized (A, C, G, T) nucleotides. Defaults to `TRUE`. |
| `maxIter` | An integer(1) specifying the maximum number of iterations to perform, default is 1. If larger, after the first loess fit, bins with median residuals larger than `cutoff` are removed, and the fitting repeated until the list of bins to use stabilizes or after `maxIter` iterations. |
| `cutoff` | A numeric(1) specifying the number of standard deviations (as estimated with `madDiff`) the cutoff for removal of bins with median residuals larger than the cutoff. Not used if `maxIter=1` (default). |
| `variables` | A character vector specifying which variables to include in the correction. Can be `c("gc", "mappability")` (the default), `"gc"`, or `"mappability"`. |
| `...` | Additional arguments passed to `loess`. |

## Value

Returns a `QDNAseqReadCounts` object with the assay data element `fit` added.

### Parallel processing

This function uses **future** to calculate the QDNAseq model across samples in parallel.

### Author(s)

Ilari Scheinin

### See Also

Internally, `loess` is used to fit the regression model.

### Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
```

---

## exportBins *Exports to a file*

---

### Description

Exports to a file.

### Usage

```
exportBins(object, file, format=c("tsv", "igv", "bed", "vcf", "seg"),
  type=c("copynumber", "segments", "calls"), filter=TRUE, logTransform=TRUE, digits=3,
   chromosomeReplacements=c(`23` = "X", `24` = "Y", `25` = "MT"), ...)
```

### Arguments

| | |
|---|---|
| `object` | A `QDNAseqReadCounts` or `QDNAseqCopyNumbers` object. |
| `file` | Filename. For formats that support only one sample per file, such as BED, '%s' can be used as a placeholder for sample name or '%d' for sample number. |
| `format` | Format to export in. Currently supported ones are "tsv" (tab separated values), "igv" (Integrative Genomics Viewer), and "bed" (BED file format). |
| `type` | Type of data to export, options are "copynumber" (corrected or uncorrected read counts), "segments", or "calls". |
| `filter` | If `TRUE`, bins are filtered, otherwise not. |
| `logTransform` | If `TRUE` (default), exported data will be log2 transformed for `format` in `"tsv"`, `"igv"`, and `"bed"`. This argument is ignored if `type = "calls"`. |
| `digits` | The number of digits to round to. If not `numeric`, no no rounding is performed. |
| `chromosomeReplacements` | A named character vector of chromosome name replacements to be done. Only used when `object` is of class `cghRaw`, `cghSeg`, `cghCall`, or `cghRegions`, since these classes store chromosome names as integers, whereas all QDNAseq object types use character vectors. Defaults to `c("23"="X", "24"="Y", "25"="MT")` for human. |
| `...` | Additional arguments passed to `write.table`. |

### Details

Exports `object` to a file.

### Value

Returns the pathnames of the files written.

### Author(s)

Ilari Scheinin

### Examples

```
## Not run:
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
exportBins(copyNumbersSmooth, file="LGG150.igv", format="igv")

## End(Not run)
```

---

## `frequencyPlot` *Plot copy number aberration frequencies*

---

### Description

Plot copy number aberration frequencies.

### Usage

```
frequencyPlot(x, y, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `QDNAseqCopyNumbers` object with `calls` data. |
| `y` | missing |
| `...` | ... |

### Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
copyNumbersSegmented <- segmentBins(copyNumbersSmooth)
copyNumbersSegmented <- normalizeSegmentedBins(copyNumbersSegmented)
copyNumbersCalled <- callBins(copyNumbersSegmented)
frequencyPlot(copyNumbersCalled)
```

---

# getBinAnnotations *Gets bin annotation data for a particular bin size*

---

## Description

Gets bin annotation data for a particular bin size.

## Usage

```
getBinAnnotations(binSize, genome="hg19", type="SR50",
  path=getOption("QDNAseq::binAnnotationPath", NULL),
  verbose=getOption("QDNAseq::verbose", TRUE))
```

## Arguments

| | |
|---|---|
| binSize | A numeric scalar specifying the width of the bins in units of kbp (1000 base pairs), e.g. binSize=15 corresponds to 15 kbp bins. |
| genome | A character string specify the genome and genome version to be used. |
| type | A character string specify the experiment type, e.g. "SR50" or "PE100". |
| path | A character string specifying the path for the bin annotation file to be downloaded. The path can either be on the local file system or a URL online. If NULL (default), then data loaded from an R package named **QDNAseq.{{genome}}**. The default value can be controlled via R options QDNAseq::binAnnotationPath. |
| verbose | If TRUE, verbose messages are produced. |

## Details

Gets bin annotation data for a particular bin size.

## Value

Returns a AnnotatedDataFrame object.

## Author(s)

Ilari Scheinin

## See Also

createBins().

## Examples

```
## Not run:
bins <- getBinAnnotations(15)

## End(Not run)
```

---

highlightFilters *Highlights data points in a plotted profile to evaluate filtering*

---

## Description

Highlights data points in a plotted profile to evaluate filtering.

## Usage

```
highlightFilters(object, col="red", residual=NA, blacklist=NA, mappability=NA, bases=NA,
  type="union", ...)
```

## Arguments

| | |
|---|---|
| object | A QDNAseqCopyNumbers object. |
| col | The color used for highlighting. |
| residual | Either a logical specifying whether to filter based on loess residuals of the calibration set, or if a numeric, the cutoff as number of standard deviations estimated with madDiff to use for. Default is TRUE, which corresponds to 4.0 standard deviations. |
| blacklist | Either a logical specifying whether to filter based on overlap with blacklisted regions, or if numeric, the maximum percentage of overlap allowed. Default is TRUE, which corresponds to no overlap allowed (i.e. value of 0). |
| mappability | A numeric in $[0, 100]$ to specify filtering out bins with mappabilities lower than the number specified. NA (default) or FALSE will not filter based on mappability. |
| bases | A numeric specifying the minimum percentage of characterized bases (not Ns) in the reference genome sequence. NA (default) or FALSE will not filter based on uncharacterized bases. |
| type | When specifying multiple filters (residual, blacklist, mappability, bases), whether to highlight their union (default) or intersection. |
| ... | Further arguments to points. |

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
plot(readCounts)
highlightFilters(readCounts, residual=TRUE, blacklist=TRUE)
```

---

`isobarPlot` *Plot median read counts as a function of GC content and mappability*

---

## Description

Plot median read counts as a function of GC content and mappability.

## Usage

```
isobarPlot(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `QDNAseqReadCounts` object. |
| `y` | missing |
| `...` | ... |

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
isobarPlot(readCounts)
```

---

`LGG150` *LGG150 chromosomes 7-10*

---

## Description

An example data set of read counts from chromosomes 7-10 of sample LGG150, contained within a `QDNAseqReadCounts` object

## Author(s)

Ilari Scheinin

makeCgh *Constructs a 'cghRaw', 'cghSeg', or 'cghCall' object*

## Description

Constructs a 'cghRaw', 'cghSeg', or 'cghCall' object.

## Usage

```
makeCgh(object, filter=TRUE, chromosomeReplacements=c(X = 23, Y = 24, MT = 25), ...)
```

## Arguments

| | |
|---|---|
| object | A QDNAseqCopyNumbers object. |
| filter | If TRUE, bins are filtered, otherwise not. |
| chromosomeReplacements | A named integer vector of chromosome name replacements to be done. QDNAseq stores chromosome names as characters, but CGHcall expects them to be integers. Defaults to c(X=23, Y=24, MT=25) for human. Value of "auto" will use running numbers in order of appearance in the bin annotations. |
| ... | Not used. |

## Value

Returns a cghRaw if the object has not been segmented, a cghSeg if it has been segmented but not called, or cghCall if it has been called as well.

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
cgh <- makeCgh(copyNumbersSmooth)
```

## noisePlot *Plot noise as a function of sequence depth*

### Description

Plot noise as a function of sequence depth.

### Usage

```
noisePlot(x, y, ...)
```

### Arguments

| | |
|---|---|
| x | A QDNAseqReadCounts object. |
| y | missing |
| ... | Further arguments to plot() and text. |

### Author(s)

Ilari Scheinin

### Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
noisePlot(readCountsFiltered)
```

## normalizeBins *Normalizes binned read counts*

### Description

Normalizes binned read counts.

### Usage

```
normalizeBins(object, method="median", force=FALSE, verbose=getOption("QDNAseq::verbose",
  TRUE))
```

### Arguments

| | |
|---|---|
| object | A QDNAseqCopyNumbers object with copynumber data. |
| method | A character string specifying the normalization method. Choices are "mean", "median" (default), or "mode". A partial string sufficient to uniquely identify the choice is permitted. |
| force | Running this function will remove possible segmentation and calling results. When they are present, running requires specifying force is TRUE. |
| verbose | If TRUE, verbose messages are produced. |

## Value

Returns a QDNAseqCopyNumbers object with the assay data element copynumber scaled with the chosen method.

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
```

---

# normalizeSegmentedBins

*Normalize segmented bins*

---

## Description

Normalize segmented bins.

## Usage

```
normalizeSegmentedBins(object, inter=c(-0.1, 0.1), force=FALSE)
```

## Arguments

| | |
|---|---|
| object | An object of class QDNAseqCopyNumbers. |
| inter | The interval in which the function should search for the normal level. |
| force | Whether to force execution when it causes removal of downstream calling results. |

## Details

This function recursively searches for the interval containing the most segmented data, decreasing the interval length in each recursion. The recursive search makes the post-segmentation normalization robust against local maxima. This function is particularly useful for profiles for which, after segmentation, the 0-level does not coincide with many segments. It is more or less harmless to other profiles. We advise to keep the search interval (inter) small, in particular at the positive (gain) side to avoid that the 0-level is set to a common gain level.

## Value

Returns an object of class QDNAseqCopyNumbers with re-normalized data.

## Author(s)

Ilari Scheinin

## See Also

Internally, postsegnormalize of the **CGHcall** package is used.

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
copyNumbersSegmented <- segmentBins(copyNumbersSmooth)
copyNumbersSegmented <- normalizeSegmentedBins(copyNumbersSegmented)
```

---

# plot *Plot copy number profile*

---

## Description

Plot copy number profile.

## Usage

```
plot(x, y, ...)
```

## Arguments

| | |
|---|---|
| x | A QDNAseqReadCounts or QDNAseqCopyNumbers object. |
| y | missing |
| ... | ... |

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
plot(copyNumbers)
```

poolRuns *Pools binned read counts across samples*

## Description

Pools binned read counts across samples.

## Usage

```
poolRuns(object, samples, force=FALSE)
```

## Arguments

| | |
|---|---|
| object | A QDNAseqReadCounts or QDNAseqCopyNumbers object. |
| samples | A character vector of new sample names. Samples with identical names will be pooled together. Must be the same length as there are samples in object. |
| force | Whether to force the operation even when downstream data will be lost. |

## Value

Returns a QDNAseqReadCounts or QDNAseqCopyNumbers object.

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
# Note: the following command will "pool" data from a single run, which
# does not really make sense:
pooledReadCounts <- poolRuns(readCounts, samples = "LGG150")
```

QDNAseq-defunct *Defunct functions in package 'QDNAseq'*

## Description

These functions are defunct and no longer available.

## Details

The following functions are defunct; use the replacement indicated below:

- downloadBinAnnotations: getBinAnnotations

QDNAseqCopyNumbers *Container for QDNAseq read count data*

## Description

Container for QDNAseq read count data

## Assay data elements

An object of this class contains the following elements:

`copynumber` (`numeric`) Corrected "count" signals in $[0, +\infty)$ An object with only this field is created by `correctBins`().

`segmented` (`numeric`; optional) Segmented data in $[0, +\infty)$, added by calling `segmentBins`().

`calls` (`integer`; optional) Calls as -2=deletion, -1=loss, 0=normal, 1=gain, 2=amplification, added by calling `callBins`().

`probdloss` (`numeric`; optional) Probabilities of deletions in $[0, 1]$, added by calling `callBins`().

`probloss` (`numeric`; optional) Probabilities of losses in $[0, 1]$, added by calling `callBins`().

`probnorm` (`numeric`; optional) Probabilities of normal copy number in $[0, 1]$, added by calling `callBins`().

`probgain` (`numeric`; optional) Probabilities of gains in $[0, 1]$, added by calling `callBins`().

`probamp` (`numeric`; optional) Probabilities of amplifications in $[0, 1]$, added by calling `callBins`().

## Missing values

The bin data (assay data) may contain missing values.

## Author(s)

Ilari Scheinin

QDNAseqReadCounts *Container for QDNAseq read count data*

## Description

Container for QDNAseq read count data

## Assay data elements

An object of this class contains (a subset) the following elements:

`counts` (`numeric`) Binned read counts as non-negative integers in $\{0, 1, 2, ...\}$. An object with only this field is created by `binReadCounts`().

`fit` (`numeric`; optional) Loess fit of "count" signals as doubles. Normally, these should all be positive values, but a small number of edge case bins might contain negatives, especially when fitting unfiltered data. This element is added after calling `estimateCorrection`().

### Missing values

The bin data (assay data) may contain missing values.

### Author(s)

Ilari Scheinin

---

## QDNAseqSignals *A parent class for containers of QDNAseq data*

---

### Description

A parent class for containers of QDNAseq data

### Author(s)

Ilari Scheinin

---

## segmentBins *Segments normalized copy number data*

---

### Description

Segments normalized copy number data.

### Usage

```
segmentBins(object, smoothBy=FALSE, alpha=1e-10, undo.splits="sdundo", undo.SD=1,
  force=FALSE, transformFun="log2", ...)
```

### Arguments

| | |
|---|---|
| object | An object of class QDNAseqCopyNumbers. |
| smoothBy | An optional integer value to perform smoothing before segmentation by taking the mean of every smoothBy bins, and then segment those means. Default (FALSE) is to perform no smoothing. smoothBy=1L is a special case that will not perform smoothing, but will split the segmentation process by chromosome instead of by sample. |
| alpha | Significance levels for the test to accept change-points. Default is 1e-10. |
| undo.splits | A character string specifying how change-points are to be undone, if at all. Default is "sdundo", which undoes splits that are not at least this many SDs apart. Other choices are "prune", which uses a sum of squares criterion, and "none". |
| undo.SD | The number of SDs between means to keep a split if undo.splits="sdundo". Default is 1.0. |
| force | Whether to force execution when it causes removal of downstream calling results. |

| | |
|---|---|
| `transformFun` | A function to transform the data with. This can be the default "log2" for log2(x + .Machine$double.xmin), "sqrt" for the Anscombe transform of sqrt(x * 3/8) which stabilizes the variance, "none" for no transformation, or any R function that performs the desired transformation and also its inverse when called with parameter `inv=TRUE`. |
| `...` | Additional arguments passed to `segment`. |

## Value

Returns an object of class QDNAseqCopyNumbers with segmentation results added.

## Numerical reproducibility

This method make use of random number generation (RNG) via the `segment` used internally. Because of this, calling the method with the same input data multiple times will each time give slightly different results. To get numerically reproducible results, the random seed must be fixed, e.g. by using 'set.seed()' at the top of the script.

## Parallel processing

This function uses **future** to segment samples in parallel.

## Author(s)

Ilari Scheinin

## References

[1] A.B. Olshen, E.S. Venkatraman (aka Venkatraman E. Seshan), R. Lucito and M. Wigler, *Circular binary segmentation for the analysis of array-based DNA copy number data*, Biostatistics, 2004
[2] E.S. Venkatraman and A.B. Olshen, *A faster circular binary segmentation algorithm for the analysis of array CGH data*, Bioinformatics, 2007

## See Also

Internally, `segment` of the **DNAcopy** package, which implements the CBS method [1,2], is used to segment the data.

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
copyNumbersSegmented <- segmentBins(copyNumbersSmooth)
```

smoothOutlierBins *Smooth outlier bins after normalization*

## Description

Smooth outlier bins after normalization.

## Usage

```
smoothOutlierBins(object, logTransform=TRUE, force=FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | A QDNAseqCopyNumbers object with copynumber data. |
| logTransform | If TRUE (default), data will be log2-transformed. |
| force | Running this function will remove possible segmentation and calling results. When they are present, running requires specifying force is TRUE. |
| ... | Additional arguments passed to smooth.CNA. |

## Value

Returns a QDNAseqCopyNumbers object with the values for outliers smoothed. See smooth.CNA for more details. If logTransform is TRUE, these signals are log2-transformed prior to smoothing, but afterwards back-transformed..

## Author(s)

Ilari Scheinin

## Examples

```
data(LGG150)
readCounts <- LGG150
readCountsFiltered <- applyFilters(readCounts)
readCountsFiltered <- estimateCorrection(readCountsFiltered)
copyNumbers <- correctBins(readCountsFiltered)
copyNumbersNormalized <- normalizeBins(copyNumbers)
copyNumbersSmooth <- smoothOutlierBins(copyNumbersNormalized)
```

# Index