

# Package ‘GenoGAM’

April 14, 2017

**Type** Package

**Title** A GAM based framework for analysis of ChIP-Seq data

**Version** 1.2.1

**Date** 2016-10-05

**Description** This package allows statistical analysis of genome-wide data with smooth functions using generalized additive models based on the implementation from the R-package 'mgcv'. It provides methods for the statistical analysis of ChIP-Seq data including inference of protein occupancy, and pointwise and region-wise differential analysis. Estimation of dispersion and smoothing parameters is performed by cross-validation. Scaling of generalized additive model fitting to whole chromosomes is achieved by parallelization over overlapping genomic intervals.

**License** GPL-2

**LazyData** true

**Depends** R (>= 3.3), Rsamtools (>= 1.18.2), SummarizedExperiment (>= 1.1.19), GenomicRanges (>= 1.23.16), methods

**Imports** BiocParallel (>= 1.5.17), data.table (>= 1.9.4), DESeq2 (>= 1.11.23), futile.logger (>= 1.4.1), GenomeInfoDb (>= 1.7.6), GenomicAlignments (>= 1.7.17), IRanges (>= 2.5.30), mgcv (>= 1.8), reshape2 (>= 1.4.1), S4Vectors (>= 0.9.34), Biostrings (>= 2.39.14)

**Suggests** BiocStyle, chipseq (>= 1.21.2), LSD (>= 3.0.0), genefilter (>= 1.54.2), ggplot2 (>= 2.1.0), testthat, knitr

**VignetteBuilder** knitr

**NeedsCompilation** no

**RoxygenNote** 5.0.1

**biocViews** Regression, DifferentialPeakCalling, ChIPSeq, DifferentialExpression, Genetics, Epigenetics

**Collate** 'GenomicTiles-class.R' 'GenoGAMSettings-class.R'  
'GenoGAM-class.R' 'GenoGAM-package.R' 'GenoGAMDataSet-class.R'  
'cv.R' 'devel.R' 'diffBinding.R' 'filter.R' 'genogam.R'  
'helper.R' 'peakCalling.R' 'plotting.R' 'qc.R' 'readData.R'  
'sf.R'

**URL** <https://github.com/gstricker/GenoGAM>

**BugReports** <https://github.com/gstricker/GenoGAM/issues>

**Author** Georg Stricker [aut, cre], Alexander Engelhardt [aut], Julien Gagneur [aut]

**Maintainer** Georg Stricker <georg.stricker@in.tum.de>

## R topics documented:

asDataFrame	3
callPeaks	3
changeSettings	4
checkSettings	5
computeRegionSignificance	6
computeSignificance	6
computeSizeFactors	7
dataRange	8
design,GenoGAMDataSet-method	8
filterData	9
GenoGAM	10
genogam	10
GenoGAM-class	11
GenoGAM-methods	12
GenoGAMDataSet	13
GenoGAMDataSet-class	14
GenoGAMDataSetToDataFrame	15
GenoGAMSettings	15
GenoGAMSettings-class	16
GenomicTiles	16
GenomicTiles-class	17
getChromosomes	18
getChunkIndex	19
getCoordinates	20
getIndex	20
getIndexCoordinates	21
getTile	22
makeTestGenoGAM	22
makeTestGenoGAMDataSet	23
makeTestGenomicTiles	23
plot.GenoGAM	24
qualityCheck	24
sizeFactors,GenoGAMDataSet-method	25
subset,GenoGAM-method	26
subset,GenoGAMDataSet-method	26
subset,GenomicTiles-method	27
subsetByOverlaps,GenoGAM,ANY-method	28
subsetByOverlaps,GenoGAMDataSet,GRanges-method	28
subsetByOverlaps,GenomicTiles,GRanges-method	29
Summary,GenomicTiles-method	30
tileSettings	32
until	32
view	33
view,GenoGAM-method	34

<code>asDataFrame</code>	3
<code>writeToBEDFile</code>	34
<code>[,GenoGAMDataSet,GRanges,ANY,ANY-method</code>	35
<code>[[,GenomicTiles,numeric,ANY-method</code>	35

**Index** **37**

`asDataFrame` *GenomicTiles to DataFrame*

**Description**

GenomicTiles to DataFrame

**See Also**

Other res: [GenoGAMDataSetToDataFrame](#)

`callPeaks` *Call peaks on a GenoGAM object*

**Description**

Call narrow or broad peaks on the GenoGAM fit and computing significance, respectively

**Usage**

```
callPeaks(fit, smooth = NULL, range = NULL, peakType = c("narrow",
  "broad"), threshold = NULL, thresholdType = c("fdr", "pvalue"),
  maxgap = 500, cutoff = 0.05, minregion = 1)
```

**Arguments**

- `fit` A GenoGAM object
- `smooth` The name of the smooth, i.e. the 'by' variables in the GenoGAMDataSet design. By default the last one will be taken.
- `range` A GRanges object specifying a range. By default the complete fit is taken.
- `peakType` The type of the peak (narrow or broad). Default is narrow, see details.
- `threshold` The significance threshold. Keep in mind that the threshold depends on the thresholdType. By default this is 0.05 for 'pvalue' and 0.1 for 'fdr'.
- `thresholdType` The threshold type. Either 'fdr'(default) or 'pvalue'. If the threshold is not provided it, will be set accordingly to the thresholdType.
- `maxgap` For broad peaks only. The maximum gap between two broad peaks, that can be tolerated in order to identify both as part of one broad peak. All broad peaks with distances smaller or equal to the maxgap will be merged.
- `cutoff` A separate threshold for broad peaks. Since pointwise pvalues are available, this threshold is used to identify all significantly high positions, which then make up a broad peak.
- `minregion` For broad peaks only. The minimum length of a broad peak. By default 1, thus catching also narrow peaks.

**Details**

Note, that broad peaks don't provide a specific highest location, but a region. Whereas narrow peaks provide both. However, the borders of narrow peaks are not necessarily informative. Additionally narrow peaks provide a 95% confidence interval for the position, namely 'start' and 'end', which gives a more informative uncertainty measure to the peak position. Also narrow peaks provide an occupancy estimate at the peak position, while broad peaks give the average occupancy across the region. The columns returned are:

**Value**

A data.table of identified peaks. The different columns loosely resemble the narrow and broad peak format (with different column order), such that it is easy to write them to a 'narrowPeak', 'broadPeak' file. See details for column description.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
load(system.file("extdata/Set1/fit.rda", package='GenoGAM'))
## calling narrow peaks
peaks <- callPeaks(fit, smooth = "genotype", threshold = 1)
peaks

## calling broad peaks
peaks <- callPeaks(fit, smooth = "genotype", threshold = 1,
                  peakType = "broad", cutoff = 0.75)
peaks
```

---

changeSettings

*Check data compliance with tile settings*

---

**Description**

Check if the indices were build correctly, according to the specified parameters. This is the recommended way of changing tile settings, as it triggers instant recomputation of the index.

**Usage**

```
changeSettings(object, param, value)

## S4 method for signature 'GenomicTiles,character'
changeSettings(object, param, value)
```

**Arguments**

object	A /codeGenomicTiles object.
param	The name of a tile settings parameter.
value	An appropriate value. In most cases integer.

**Value**

A /codeGenomicTiles object

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()  
gt2 <- changeSettings(gt, "chunkSize", 20)
```

---

checkSettings	<i>Check data compliance with tile settings</i>
---------------	---

---

**Description**

Check if the indices were build correctly, according to the specified parameters

**Usage**

```
checkSettings(object)  
  
## S4 method for signature 'GenomicTiles'  
checkSettings(object)
```

**Arguments**

object            A /codeGenomicTiles object.

**Value**

A logical value

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()  
checkSettings(gt)
```

computeRegionSignificance

*Compute significance for given regions*

---

### Description

For a given set of regions, region-wise pvalues and FDR is computed

### Usage

```
computeRegionSignificance(fit, regions, what = NULL)
```

### Arguments

fit	A GenoGAM object containing the fit
regions	A GRanges object of regions of interest
what	Which fit should be used. The names should be equivalent to the column names used in the config file. Lookup with <code>names(colData(my_GenoGAMDataSet_object))</code>

### Details

For a given set of regions, region-wise pvalues are computed by applying familywise hochberg correction and taking the minimal p-value. FDR is computed by further applying Benjamini-Hochberg correction.

### Value

The GRanges object from the 'region' parameter extended by two columns: pvalue and FDR

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

### Examples

```
gg <- makeTestGenoGAM()
gr <- GRanges("chrI", IRanges(1,100))
computeRegionSignificance(gg, gr)
```

---

computeSignificance *Compute significance.*

---

### Description

Based on the model fits this functions computes pointwise pvalues.

### Usage

```
computeSignificance(gg, log.p = FALSE)
```

**Arguments**

`gg` A fitted GenoGAM object.  
`log.p` Should pvalues be returned in log scale?

**Value**

A GenoGAM object which fits has been updated by the pvalue columns.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAM()
ggd <- computeSignificance(ggd)
head(getFits(ggd))
```

---

`computeSizeFactors`      *computeSizeFactors*

---

**Description**

The function computes the size factors for given factor groups based on the DESeq2 package.

**Usage**

```
computeSizeFactors(ggd, factorGroups = NULL)
```

**Arguments**

`ggd` A GenoGAMDataSet object.  
`factorGroups` A list of grouped IDs (same as the colnames of the GenoGAMDataSet object). Each element of the list represents a group of samples within which size factors are computed. If NULL all samples are regarded to belong to one group. Size factors are not computed between groups.

**Value**

An updated GenoGAMDataSet object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()
ggd <- computeSizeFactors(ggd)
```

---

dataRange	<i>The /codeGRanges of the underlying data</i>
-----------	--

---

**Description**

Just like the /codecoordinates slot but returns the genomic ranges of the underlying data.

**Usage**

```
dataRange(object)

## S4 method for signature 'GenomicTiles'
dataRange(object)

## S4 method for signature 'GPos'
dataRange(object)
```

**Arguments**

object            A /codeGenomicTiles object.

**Value**

A /codeGRanges object of genomic ranges of the underlying data

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
dataRange(gt)
```

---

design,GenoGAMDataSet-method	<i>Access the design slot</i>
------------------------------	-------------------------------

---

**Description**

The design slot contains the formula object which is used to fit the model

**Usage**

```
## S4 method for signature 'GenoGAMDataSet'
design(object)

## S4 replacement method for signature 'GenoGAMDataSet,ANY'
design(object) <- value
```



**Arguments**

object            A GenoGAMDataSet object.  
value            A formula object

**Value**

A formula object

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()  
design(ggd)  
design(ggd) <- ~1
```

---

filterData            *A filter function for lcodeGenoGAMDataSet*

---

**Description**

A function to filter the lcodeGenoGAMDataSet by the sum or mean of counts to significantly reduce the amount of models to compute

**Usage**

```
filterData(ggd, threshold = NULL, windowsize = 201, mode = c("sum",  
"mean"))
```

**Arguments**

ggd            A GenoGAMDataSet object  
threshold      A value for the mean or sum of counts, which will be used to filter on basepair level. By default it is taken as median + 3\*MAD  
windowsize    The sliding window size. Should be an odd value.  
mode           Should the sum or the mean of counts be used?

**Value**

A GenoGAMDataSet object containing the filtered regions

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()  
fdata <- filterData(ggd, windowsize = 10)  
fdata
```

---

GenoGAM	<i>GenoGAM: A package providing a framework to analyse ChIP-Seq data</i>
---------	--

---

### Description

GenoGAM: A package providing a framework to analyse ChIP-Seq data

---

genogam	<i>genogam</i>
---------	----------------

---

### Description

This is the fitting function for `GenoGAMDataSet`. It processes the data in `GenoGAMDataSet`, estimates the overdispersion and the penalization parameter, passes all information to `mgcv::gam` in parallel fashion and extracts the results. So far the model is restricted to Negative Binomial distribution (`mgcv::nb()`).

### Usage

```
genogam(ggd, lambda = NULL, family = mgcv::nb(), bpknots = 20,
        kfolds = 10, intervallSize = 20, m = 2)
```

### Arguments

<code>ggd</code>	A <code>GenoGAMDataSet</code> object to be fitted.
<code>lambda</code>	The penalization parameter. Will be estimated if missing.
<code>family</code>	A distribution family object. So far only <code>mgcv::nb()</code> is allowed.
<code>bpknots</code>	Number of basepairs per one knot, that is, how dense should the knots be placed. The denser the knots, the more sensitive the fit. Note however, that computation time increases approximately cubic with every additional knot.
<code>kfolds</code>	An integer number giving the number of k-folds to be used in cross validation, if parameters need to be estimated.
<code>intervallSize</code>	The size of the intervalls to be used in cross validation. Short intervalls are used instead of single points to be left out due to spatial correlation. If replicates are present it is advised to make them bigger, e.g. $2 \cdot \text{fragment size}$ . Otherwise, depending on the density of the data, they should not exceed the size of a short read.
<code>m</code>	The penalization order of the P-Splines.

### Value

A `GenoGAM` object containing the fits and parameters.

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
## Not run:
## simple example
config <- data.frame(ID = c("input", "IP"),
                    file = c("myInput.bam",
                            "myIP.bam"),,
                    paired = c(FALSE, FALSE),
                    type = c(0,1), stringsAsFactors = FALSE)
bpk <- 100 ## basepairs per one knot
chunkSize <- 5000
overhang <- round(7*chunkSize/bpk) ##overhang with 7 knots
knots <- chunkSize/bpk
## build the GenoGAMDataSet
gtiles <- GenoGAMDataSet(config = config, chunkSize = chunkSize, overhangSize = overhang,
                        design = ~ s(x) + s(x, by = type))
gtiles <- computeSizeFactors(gtiles)
fits <- genogam(gtiles, bpknots = bpk)

## End(Not run)
```

GenoGAM-class

*GenoGAM class***Description**

This class is designed to represent the model object containing the estimate parameters, arguments and final fits of the model on a basepair level.

**Slots**

**design** A mgcv-type formula object.

**fits** A data.frame of the fits, the standard error and the first and second derivative of the fits for each experiment.

**positions** A GPos object of the positions and seqnames corresponding to the rows in the 'fits' slot.

**smooths** A data.frame of knot positions and base function coefficients, in order to reproduce the splines and compute derivatives.

**vcov** A list of covariance matrices for each tile fit.

**experimentDesign** The design matrix according to which the fitting was performed.

**fitparams** Global parameters 'lambda', 'theta', 'Coefficient of Variation' and the 'penalty order' used to compute the model.

**family** The distribution family.

**cvparams** Parameters used for cross validation.

**settings** The global and local settings that were used to compute the model.

**tileSettings** A list of settings used to compute tiles.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Description**

The different accessor functions for the GenoGAM object

The 'positions' slot holds the positions of the fit in GPos format

The 'design' slot holds the formula of the fit

The 'fits' slot contains the fitted values of the model

The 'experimentDesign' slot contains the experimental design of the model as specified in the config file

**Usage**

```
## S4 method for signature 'GenoGAM'  
rowRanges(x)
```

```
## S4 method for signature 'GenoGAM'  
design(object)
```

```
getFits(x)
```

```
## S4 method for signature 'GenoGAM'  
getFits(x)
```

```
## S4 method for signature 'GenoGAM'  
colData(x)
```

**Arguments**

x, object      A GenoGAM object.

**Value**

The respective slot

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gg <- makeTestGenoGAM()  
ranges <- rowRanges(gg)  
gg <- makeTestGenoGAM()  
des <- design(gg)  
gg <- makeTestGenoGAM()  
fits <- getFits(gg)  
gg <- makeTestGenoGAM()  
exdesign <- colData(gg)
```

---

GenoGAMDataSet	<i>GenoGAMDataSet constructor.</i>
----------------	------------------------------------

---

## Description

This is the constructor function for GenoGAMDataSet. So far a GenoGAMDataSet can be constructed from either an experiment design file or data.frame or directly from a RangedSummarizedExperiment with a GPos object being the rowRanges.

## Usage

```
GenoGAMDataSet(experimentDesign, chunkSize, overhangSize, design,
  directory = ".", settings = NULL, ...)
```

## Arguments

experimentDesign	Either a character object specifying the path to a delimited text file (the delimiter will be determined automatically), or a data.frame specifying the experiment design. See details for the structure of the experimentDesign.
chunkSize	An integer specifying the size of one chunk in bp.
overhangSize	An integer specifying the size of the overhang in bp. As the overhang is taken to be symmetrical, only the overhang of one side should be provided.
design	A mgcv-like formula object. See details for its structure.
directory	The directory from which to read the data. By default the current working directory is taken.
settings	A GenoGAMSettings object. This class is already present but not yet fully tested and therefore not accessible to the user. This argument exists however in order to allow some workarounds if necessary. See the vignette for a possible use.
...	Further parameters, mostly for arguments of custom processing functions or to specify a different method for fragment size estimation. See details for further information.

## Details

The experimentDesign file/data.frame must contain at least three columns with fixed names: 'ID', 'file' and 'paired'. The field 'ID' stores a unique identifier for each alignment file. It is recommended to use short and easy to understand identifiers because they are subsequently used for labelling data and plots. The field 'file' stores the BAM file name. The field 'paired', values TRUE for paired-end sequencing data, and FALSE for single-end sequencing data. All other columns are stored in the colData slot of the GenoGAMDataSet object. Note that all columns which will be used for analysis must have at most two conditions, which are for now restricted to 0 and 1. For example, if the IP data should be corrected for input, then the input will be 0 and IP will be 1, since we are interested in the corrected IP. See examples.

Design must be a mgcv-like formula. At the moment only the following is possible: Either '~ 1' for a constant. ~ s(x) for a smooth fit over the entire data. s(x, by = "myColumn"), where 'myColumn' is a column name in the experimentDesign. This type of formula will then only fit the samples annotated with 1 in this column. Or ~ s(x) + s(x, by = "myColumn") + s(x, by = ...) + ... The last formula lets you combine any number of columns, given they are binary with 0 and 1. For example

the formula for correcting IP for input would look like this:  $\sim s(x) + s(x, \text{by} = \text{"experiment"})$ , where 'experiment' is a column with 0s and 1s, with the ip samples annotated with 1 and input samples with 0. ' In case of single-end data in might be useful to specify a different method for fragment size estimation. The argument 'shiftMethod' can be supplied with the values 'coverage' (default), 'correlation' or 'SISSR'. See `?chipseq::estimate.mean.fraglen` for explanation.

### Value

An object of class `GenoGAMDataSet`.

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

### Examples

```
## Not run:
myConfig <- data.frame(ID = c("input", "ip"),
  file = c("myInput.bam", "myIP.bam"),
  paired = c(FALSE, FALSE),
  experiment = factor(c(0,1)),
  stringsAsFactors = FALSE)
myConfig2 <- data.frame(ID = c("wildtype1", "wildtype2",
  "mutant1", "mutant2"),
  file = c("myWT1.bam", "myWT2.bam",
  "myMutant1.bam", "myMutant2.bam"),
  paired = c(FALSE, FALSE, FALSE, FALSE),
  experiment = factor(c(0, 0, 1, 1)),
  stringsAsFactors = FALSE)

gfiles <- GenoGAMDataSet(myConfig, chunkSize = 2000,
  overhang = 250, design = ~ s(x) + s(x, by = "experiment")
gfiles <- GenoGAMDataSet(myConfig2, chunkSize = 2000,
  overhang = 250, design = ~ s(x) + s(x, by = "experiment"))

## End(Not run)
## make a test dataset
ggd <- makeTestGenoGAMDataSet()
ggd
```

---

GenoGAMDataSet-class    *GenoGAMDataSet*

---

### Description

This class is designed to represent the input for the GenoGAM model. it extends the `GenomicTiles` class.

### Details

For all other slots see `SummarizedExperiment`.

**Slots**

`settings` The global and local settings that were used to compute the model.  
`design` The formula describing how to evaluate the data.  
`sizeFactors` The normalized values for each sample.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

GenoGAMDataSetToDataFrame

*GenoGAMDataSet to DataFrame*

---

**Description**

GenoGAMDataSet to DataFrame

**See Also**

Other res: [asDataFrame](#)

---

GenoGAMSettings

*The constructor function for GenoGAMSettings*

---

**Description**

The constructor function for GenoGAMSettings

**Usage**

```
GenoGAMSettings(...)
```

**Arguments**

... Any parameters corresponding to the slots and their possible values. See [GenoGAM-Settings](#)

**Value**

A GenoGAMSettings object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

GenoGAMSettings-class *GenoGAMSettings*

---

### Description

This class is designed to store settings for the computation of the GenoGAM package

### Details

Center can have three values: TRUE, FALSE, NULL. TRUE will trigger the center function, FALSE will trigger the use of the entire fragment. NULL should be used in case a custom process function is used.

### Slots

`center` A logical or NULL value to specify if the raw data should be centered, i.e. only the mid-point of the fragment will be used to represent its coverage. See details.

`chromosomeList` A character vector of chromosomes to be used. NULL for all chromosomes.

`bamParams` An object of class ScanBamParam. See `?Rsamtools::ScanBamParam`.

`parallel` A parallel backend of the respective class. See `BiocParalell` for the options

`processFunction` A custom function on how to process raw data. Not used if center is TRUE/FALSE.

`optimMethod` The optimisation method to be used in cross validation.

`optimControl` Settings for the `optim()` function.

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

---

GenomicTiles

*GenomicTiles constructor.*

---

### Description

This is the constructor function for GenomicTiles. The easiest construction is from Summarized-Experiment. However as the class operates on basepair level, the rowRanges are restricted to the GPos class.

### Usage

```
GenomicTiles(assays, chunkSize = 10000, overhangSize = 0, ...)
```



**Arguments**

assays	One of two things. Either directly an object of type 'RangedSummarizedExperiment'. Or in case the object is created from raw data, a 'list' or 'SimpleList' of matrix-like elements, or a matrix-like object. All elements of the list must have the same dimensions, and dimension names (if present) must be consistent across elements and with the row names of 'rowRanges' and 'colData'.
chunkSize	An integer specifying the size of one chunk in bp.
overhangSize	An integer specifying the size of the overhang in bp. The overhang is regarded to be symmetric, such that only the overhang of one side should be provided.
...	Further parameters passed to the <a href="#">SummarizedExperiment</a> constructor.

**Details**

Most, but not necessary all functionalities of SummarizedExperiment are yet provided.

**Value**

An object of class GenomicTiles.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
## from raw data
gp <- GPos(GRanges(c("chrI", "chrII"), IRanges(c(1,1), c(5,5))))
assay <- matrix(1:10, 10, 1)
gt <- GenomicTiles(assay, chunkSize = 3, rowRanges = gp)

## from SummarizedExperiment
se <- SummarizedExperiment(assay, rowRanges = gp)
gt <- GenomicTiles(se, chunkSize = 3)
```

---

GenomicTiles-class      *GenomicTiles class*

---

**Description**

This class is designed to represent the entire genome (or a subset of it) and any additional data associated with the samples or positions. It extends the RangedSummarizedExperiment class and adds two additional index slots to keep track of the data. The main change compared to RangedSummarizedExperiment is the use of a GPos (basepair level) instead of GRanges (ranges level) object as rowRanges and the use of two GRanges objects as indices. The GPos object allows to store raw instead of summarized data in the assays. Because of this the size of genomic data can increase tremendously. Thus the GenomicTiles class automatically divides the data in (overlapping) tiles, making any operation on this data easy executable in parallel.

**Details**

For all other slots see [SummarizedExperiment](#).

**Slots**

`index` A GRanges object that stores the tiles ranges and their index in the genome space. That is, ranges are the positions on the genome.

`coordinates` A GRanges object that stores the tiles ranges and their index in the DataFrame space. That is ranges are the row positions in the DataFrame.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

getChromosomes	<i>The single entries of the tile settings</i>
----------------	--

---

**Description**

Returns the single elements of the tile settings

**Usage**

```
getChromosomes(object)

## S4 method for signature 'GenomicTiles'
getChromosomes(object)

getTileSize(object)

## S4 method for signature 'GenomicTiles'
getTileSize(object)

getChunkSize(object)

## S4 method for signature 'GenomicTiles'
getChunkSize(object)

getOverhangSize(object)

## S4 method for signature 'GenomicTiles'
getOverhangSize(object)

getTileNumber(object, ...)

## S4 method for signature 'GenomicTiles'
getTileNumber(object)
```

**Arguments**

<code>object</code>	A /codeGenomicTiles object.
<code>...</code>	Additional arguments

**Value**

An integer value, or in case of `getChromosomes` a `GRanges` object

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
getChromosomes(gt)
getTileSize(gt)
getChunkSize(gt)
getOverhangSize(gt)
getTileNumber(gt)
```

---

getChunkIndex	<i>Compute the index for chunks instead tiles</i>
---------------	---

---

**Description**

The chunk index holds the `GRanges` object that splits the entire dataset in chunk, that is non-overlapping intervals.

**Usage**

```
getChunkIndex(object, ...)
```

## S4 method for signature 'GenomicTiles'

```
getChunkIndex(object, id = NULL)
```

**Arguments**

object	A <code>GenomicTiles</code> object.
...	Additional arguments
id	A vector if tile ids. By default the complete index is returned.

**Value**

A `GRanges` object representing the index

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
getChunkIndex(gt)
```

---

getCoordinates	<i>Accessor to the /codecoordinates slot</i>
----------------	--

---

### Description

The /codecoordinates slot contains the row coordinates of each chromosome in the data. Such that taken a genomic position from a chromosome it's easy to detect the correct row in the assay

### Usage

```
getCoordinates(object)

## S4 method for signature 'GenomicTiles'
getCoordinates(object)
```

### Arguments

object            A /codeGenomicTiles object.

### Value

A /codeGRanges object of row coordinates

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

### Examples

```
gt <- makeTestGenomicTiles()
getCoordinates(gt)
```

---

getIndex	<i>Accessor to the 'index' slot</i>
----------	-------------------------------------

---

### Description

The index holds the Granges object that splits the entire dataset in tiles.

### Usage

```
getIndex(object, ...)
```

```
## S4 method for signature 'GenomicTiles'
getIndex(object, id = NULL)
```

### Arguments

object            A /codeGenomicTiles object.  
 ...                Additional arguments  
 id                 A vector if tile ids. By default the complete index is returned.

**Value**

A GRanges object representing the index

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
getIndex(gt)
getIndex(gt, 1:3)
```

---

getIndexCoordinates    *Compute the row coordinates for a given index*

---

**Description**

Given an index of genomic positions, this method computes the corresponding row positions in the assay

**Usage**

```
getIndexCoordinates(object, ...)
```

## S4 method for signature 'GenomicTiles'

```
getIndexCoordinates(object, id = NULL,
  index = NULL)
```

**Arguments**

object	A <code>/codeGenomicTiles</code> object.
...	Additional arguments Usually the original index or the chunk index.
id	A vector if tile ids. By default the complete index is returned.
index	A <code>/codeGranges</code> object representing an index of genomic positions.

**Value**

A `/codeGRanges` object of row coordinates

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
getIndexCoordinates(gt)
```

---

getTile	<i>Tile extraction as a DataFrame</i>
---------	---------------------------------------

---

**Description**

Extracting one or multiple tiles from a GenomicTiles object and coercing them to a DataFrameList.

**Usage**

```
getTile(object, id, ...)

## S4 method for signature 'GenomicTiles'
getTile(object, id, size = 3e+09)
```

**Arguments**

object	A GenomicTiles object
id	A vector of tile ids
...	Additional arguments
size	The maximal number of rows that should be handled at once. If the dataset is bigger it will be processed in chunks. This is to lower memory consumption on big datasets, which in turn is slower.

**Value**

A SimpleDataFrameList

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
getTile(gt, 1:3)
```

---

makeTestGenoGAM	<i>Make an example /codeGenoGAM</i>
-----------------	-------------------------------------

---

**Description**

Make an example /codeGenoGAM

**Usage**

```
makeTestGenoGAM()
```

**Value**

A /codeGenoGAM object

### Examples

```
test <- makeTestGenoGAM()
```

---

makeTestGenoGAMDataSet

*Make an example /codeGenoGAMDataSet*

---

### Description

Make an example /codeGenoGAMDataSet

### Usage

```
makeTestGenoGAMDataSet()
```

### Value

A /codeGenoGAMDataSet object

### Examples

```
test <- makeTestGenoGAMDataSet()
```

---

makeTestGenomicTiles

*Make an example /codeGenomicTile*

---

### Description

Make an example /codeGenomicTile

### Usage

```
makeTestGenomicTiles()
```

### Value

A /codeGenomicTiles object

### Examples

```
test <- makeTestGenomicTiles()
```

---

plot.GenoGAM	<i>The pot function for a GenoGAM object</i>
--------------	--

---

**Description**

This functions plots the fit of a given region and optionally the read counts from the GenoGAM-DataSet object

**Usage**

```
plot.GenoGAM(x, ggd = NULL, ranges = NULL, seqnames = NULL,
             start = NULL, end = NULL, scale = TRUE, ...)
```

**Arguments**

x	A GenoGAM object
ggd	A GenoGAMDataSet object to plot raw counts
ranges	A GRanges object specifying a particular region
seqnames	A chromosome name. Together with start and end it is an alternative way of selecting a region
start	The start of a region
end	The end of a region
scale	Logical, should all tracks be scaled to the same y-axis?
...	Additional parameters that will be passed to the basic plot routine

**Value**

A plot of all tracks either using the ggplot2 or the base R framework

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

qualityCheck	<i>A function to quality check the data</i>
--------------	---

---

**Description**

This function checks some data attributes in the given class. Check details for more information.

**Usage**

```
qualityCheck(object, ...)
```

**Arguments**

object	Any object for which this methods is implemented
...	further parameters. See details.



**Details**

So far this method is only implemented for the class `GenoGAMDataSet`. In this case some general metrics are printed and some plots are stored in the folder "qc", which will be created in the working directory.

Additional parameters: `factorGroups` (for `GenoGAMDataSet`), which is used to specify factor groups for normalization plots. By default the groups will be identified automatically. See `?computeSizeFactors` for parameter description.

**Value**

Based on the object provided, see details.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

sizeFactors, GenoGAMDataSet-method

*Access the sizeFactor slot*

---

**Description**

The `sizeFactor` slot contains the vector of normalization values for each sample

**Usage**

```
## S4 method for signature 'GenoGAMDataSet'  
sizeFactors(object)
```

```
## S4 replacement method for signature 'GenoGAMDataSet,ANY'  
sizeFactors(object) <- value
```

**Arguments**

<code>object</code>	A <code>GenoGAMDataSet</code> object.
<code>value</code>	A named numeric vector

**Value**

A named numeric vector

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()  
sizeFactors(ggd)  
sizeFactors(ggd) <- c(a = 5, b = 1/5)
```

---

subset,GenoGAM-method *Subset method for* GenoGAM

---

**Description**

Subsetting the GenoGAM by a logical statement

**Usage**

```
## S4 method for signature 'GenoGAM'  
subset(x, ...)
```

**Arguments**

x	A GenoGAM object.
...	Further arguments. Mostly a logical statement. Note that the columnnames for chromosomes and positions are: seqnames and pos.

**Value**

A subsetting GenoGAM object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gg <- makeTestGenoGAM()  
subset(gg, pos <= 40)
```

---

subset,GenoGAMDataSet-method  
*Subset method for* GenoGAMDataSet

---

**Description**

Subsetting the GenoGAMDataSet by a logical statement

**Usage**

```
## S4 method for signature 'GenoGAMDataSet'  
subset(x, ...)
```

**Arguments**

x	A GenoGAMDataSet object.
...	Further arguments. Mostly a logical statement. Note that the columnnames for chromosomes and positions are: seqnames and pos.

**Value**

A subsetted GenomicTiles object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()
res <- subset(ggd, seqnames == "chrI" & pos <= 50)
```

---

subset,GenomicTiles-method

*Subset method for /codeGenomicTiles*

---

**Description**

Subsetting the /codeGenomicTiles by a logical statement

**Usage**

```
## S4 method for signature 'GenomicTiles'
subset(x, ...)
```

**Arguments**

x	A /codeGenomicTiles object.
...	Further arguments. Mostly a logical statement. Note that the columnnames for chromosomes and positions are: seqnames and pos.

**Value**

A subsetted /codeGenomicTiles object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
res <- subset(gt, seqnames == "chrI" & pos <= 50)
```

---

subsetByOverlaps,GenoGAM,ANY-method  
*Subset by overlaps method for GenoGAM*

---

### Description

Subsetting the GenoGAM by a GRanges object

### Usage

```
## S4 method for signature 'GenoGAM,ANY'
subsetByOverlaps(query, subject)
```

### Arguments

query	A GenoGAM object.
subject	A GRanges object
...	Additional parameters

### Value

A subsetting GenoGAM object.

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

### Examples

```
gg <- makeTestGenoGAM()
gr <- GRanges("chrI", IRanges(1,40))
subsetByOverlaps(gg, gr)
```

---

subsetByOverlaps,GenoGAMDataSet,GRanges-method  
*Subset by overlaps method for GenoGAMDataSet*

---

### Description

Subsetting the GenoGAMDataSet by a GRanges object

### Usage

```
## S4 method for signature 'GenoGAMDataSet,GRanges'
subsetByOverlaps(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), ...)
```

**Arguments**

query	A GenoGAMDataSet object.
subject	A GRanges object
maxgap, minoverlap	Intervals with a separation of maxgap or less and a minimum of minoverlap overlapping positions, allowing for maxgap, are considered to be overlapping. maxgap should be a scalar, non-negative, integer. minoverlap should be a scalar, positive integer.
type	By default, any overlap is accepted. By specifying the type parameter, one can select for specific types of overlap. The types correspond to operations in Allen's Interval Algebra (see references). If type is start or end, the intervals are required to have matching starts or ends, respectively. While this operation seems trivial, the naive implementation using outer would be much less efficient. Specifying equal as the type returns the intersection of the start and end matches. If type is within, the query interval must be wholly contained within the subject interval. Note that all matches must additionally satisfy the minoverlap constraint described above.  The maxgap parameter has special meaning with the special overlap types. For start, end, and equal, it specifies the maximum difference in the starts, ends or both, respectively. For within, it is the maximum amount by which the query may be wider than the subject.
...	Additional parameters

**Value**

A subsetted GenoGAMDataSet object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
ggd <- makeTestGenoGAMDataSet()
gr <- GRanges("chrI", IRanges(1,50))
res <- subsetByOverlaps(ggd, gr)
```

---

subsetByOverlaps, GenomicTiles, GRanges-method  
*Subset by overlaps method for GenomicTiles*

---

**Description**

Subsetting the GenomicTiles by a GRanges object

**Usage**

```
## S4 method for signature 'GenomicTiles,GRanges'
subsetByOverlaps(query, subject, maxgap = 0L,
  minoverlap = 1L, type = c("any", "start", "end", "within", "equal"), ...)
```

**Arguments**

query	A GenomicTiles object.
subject	A GRanges object
maxgap, minoverlap	Intervals with a separation of maxgap or less and a minimum of minoverlap overlapping positions, allowing for maxgap, are considered to be overlapping. maxgap should be a scalar, non-negative, integer. minoverlap should be a scalar, positive integer.
type	By default, any overlap is accepted. By specifying the type parameter, one can select for specific types of overlap. The types correspond to operations in Allen's Interval Algebra (see references). If type is start or end, the intervals are required to have matching starts or ends, respectively. While this operation seems trivial, the naive implementation using outer would be much less efficient. Specifying equal as the type returns the intersection of the start and end matches. If type is within, the query interval must be wholly contained within the subject interval. Note that all matches must additionally satisfy the minoverlap constraint described above.  The maxgap parameter has special meaning with the special overlap types. For start, end, and equal, it specifies the maximum difference in the starts, ends or both, respectively. For within, it is the maximum amount by which the query may be wider than the subject.
...	Additional parameters

**Value**

A subsetted GenomicTiles object.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
gr <- GRanges(c("chrI", "chrII"), IRanges(c(1, 120), c(40, 150)))
res <- subsetByOverlaps(gt, gr)
```

---

Summary, GenomicTiles-method

*Computing metrics*

---

**Description**

Computing metrics on each tile of the GenomicTiles object. So far all metrics from the Summary generics group, as well as mean, var, sd, median, mad and IQR are supported.

**Usage**

```
## S4 method for signature 'GenomicTiles'
Summary(x, ..., na.rm = FALSE)

## S4 method for signature 'GenomicTiles'
mean(x)

## S4 method for signature 'GenomicTiles,ANY'
var(x)

## S4 method for signature 'GenomicTiles'
sd(x)

## S4 method for signature 'GenomicTiles'
median(x)

## S4 method for signature 'GenomicTiles'
mad(x)

## S4 method for signature 'GenomicTiles'
IQR(x)
```

**Arguments**

x	A GenomicTiles object
...	Additional arguments
na.rm	Should NAs be dropped. Otherwise the result is NA

**Value**

A list of as many elements as there are assays. Each element contains of a matrix with the specified metric computed per tile per column of the assay data.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
sum(gt)
min(gt)
max(gt)
mean(gt)
var(gt)
sd(gt)
median(gt)
mad(gt)
IQR(gt)
```

---

tileSettings	<i>Return tile settings</i>
--------------	-----------------------------

---

**Description**

Returns a list settings used to generate the tile index

**Usage**

```
tileSettings(object)

## S4 method for signature 'GenomicTiles'
tileSettings(object)
```

**Arguments**

object            A /codeGenomicTiles object.

**Value**

A list of tile settings

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
tileSettings(gt)
```

---

until	<i>Set index to chunkIndex</i>
-------	--------------------------------

---

**Description**

Replace the tile index with the chunk index in /codeGenomicTiles object

**Usage**

```
until(object, ...)
```

```
## S4 method for signature 'GenomicTiles'
until(object, id = NULL)
```

**Arguments**

object            A /codeGenomicTiles object.  
 ...                Additional arguments  
 id                 A vector if tile ids. By default the complete index is taken.



**Value**

A modified `/codeGenomicTiles` object

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
newGT <- until(gt)
```

---

view

*View the dataset*

---

**Description**

Cbinding the columns all together and coercing to `data.frame`

**Usage**

```
view(object, ...)
```

```
## S4 method for signature 'GenomicTiles'
view(object, ranges = NULL, seqnames = NULL,
      start = NULL, end = NULL)
```

**Arguments**

<code>object</code>	A <code>GenomicTiles</code> object
<code>...</code>	Additional arguments
<code>ranges</code>	A <code>GRanges</code> object. Makes it possible to select regions by <code>GRanges</code> . Either <code>ranges</code> or <code>seqnames</code> , <code>start</code> and <code>end</code> must be supplied
<code>seqnames</code>	A chromosomes name. Either <code>ranges</code> or <code>seqnames</code> , <code>start</code> and <code>end</code> must be supplied
<code>start</code>	A start site. Either <code>ranges</code> or <code>seqnames</code> , <code>start</code> and <code>end</code> must be supplied
<code>end</code>	An end site. Either <code>ranges</code> or <code>seqnames</code> , <code>start</code> and <code>end</code> must be supplied

**Value**

A `data.frame` of the selected data.

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

**Examples**

```
gt <- makeTestGenomicTiles()
gr <- GRanges(c("chrI", "chrII"), IRanges(c(1, 10), c(40, 30)))
head(view(gt, ranges = gr))
head(view(gt, seqnames = "chrI", start = 1, end = 20))
```

---

view,GenoGAM-method    *View the dataset*

---

### Description

Combining the columns all together and coercing to data.frame

### Usage

```
## S4 method for signature 'GenoGAM'
view(object, ranges = NULL, seqnames = NULL,
      start = NULL, end = NULL)
```

### Arguments

object	A GenoGAM object
ranges	A GRanges object. Makes it possible to select regions by GRanges. Either ranges or seqnames, start and end must be supplied
seqnames	A chromosomes name. Either ranges or seqnames, start and end must be supplied
start	A start site. Either ranges or seqnames, start and end must be supplied
end	An end site. Either ranges or seqnames, start and end must be supplied

### Value

A data.frame of the selected data.

### Author(s)

Georg Stricker <georg.stricker@in.tum.de>

### Examples

```
gg <- makeTestGenoGAM()
gr <- GRanges("chrI", IRanges(1,40))
head(view(gg, gr))
```

---

writeToBEDFile    *Write peaks to BED6+3/4 format*

---

### Description

A function to write the data.table of peaks into a narrowPeaks or broadPeaks file

### Usage

```
writeToBEDFile(peaks, file = NULL)
```

**Arguments**

peaks	A data.table or data.frame of peaks as produced by callPeaks()
file	A file name without suffix. It will be determined automatically. If no file is given, it will be written to a generic 'peaks_[timestamp]' file in the current working directory

**Value**

Nothing. A narrowPeaks or broadPeaks file written to 'file'

**Author(s)**

Georg Stricker <georg.stricker@in.tum.de>

---

[,GenoGAMDataSet,GRanges,ANY,ANY-method  
*Subsetting by GRanges*

---

**Description**

Providing subsetting by GRanges through the single-bracket operator

**Usage**

```
## S4 method for signature 'GenoGAMDataSet,GRanges,ANY,ANY'
x[i]
```

**Arguments**

x	A GenoGAMDataSet object
i	A GRanges object

**Value**

A subsetting GenoGAMDataSet object

---

[[,GenomicTiles,numeric,ANY-method  
*Providing pseudo-list functionality*

---

**Description**

Getting a specific tile

**Usage**

```
## S4 method for signature 'GenomicTiles,numeric,ANY'
x[[i]]

## S4 method for signature 'GenomicTiles,GRanges,ANY,ANY'
x[i]
```

**Arguments**

- x            A GenomicTiles object
- i            An integer (for '[') or a GRanges object (for '[')

**Value**

A DataFrame (for '[') or a subsetted GenomicTiles object (for '[')

# Index

- [, GenoGAMDataSet, GRanges, ANY, ANY-method, [35](#)
- [, GenomicTiles, GRanges, ANY, ANY-method ([\[\[, GenomicTiles, numeric, ANY-method](#)]), [35](#)
- [\[\[, GenomicTiles, numeric, ANY-method](#), [35](#)
- asDataFrame, [3](#), [15](#)
- callPeaks, [3](#)
- changeSettings, [4](#)
- changeSettings, GenomicTiles, character-method ([changeSettings](#)), [4](#)
- checkSettings, [5](#)
- checkSettings, GenomicTiles-method ([checkSettings](#)), [5](#)
- colData, GenoGAM-method ([GenoGAM-methods](#)), [12](#)
- computeRegionSignificance, [6](#)
- computeSignificance, [6](#)
- computeSizeFactors, [7](#)
- dataRange, [8](#)
- dataRange, GenomicTiles-method ([dataRange](#)), [8](#)
- dataRange, GPos-method ([dataRange](#)), [8](#)
- design, GenoGAM-method ([GenoGAM-methods](#)), [12](#)
- design, GenoGAMDataSet-method, [8](#)
- design<- , GenoGAMDataSet, ANY-method ([design, GenoGAMDataSet-method](#)), [8](#)
- filterData, [9](#)
- GenoGAM, [10](#)
- genogam, [10](#)
- GenoGAM-class, [11](#)
- GenoGAM-methods, [12](#)
- GenoGAMDataSet, [13](#)
- GenoGAMDataSet-class, [14](#)
- GenoGAMDataSetToDataFrame, [3](#), [15](#)
- GenoGAMSettings, [15](#), [15](#)
- GenoGAMSettings-class, [16](#)
- GenomicTiles, [16](#)
- GenomicTiles-class, [17](#)
- getChromosomes, [18](#)
- getChromosomes, GenomicTiles-method ([getChromosomes](#)), [18](#)
- getChunkIndex, [19](#)
- getChunkIndex, GenomicTiles-method ([getChunkIndex](#)), [19](#)
- getChunkSize ([getChromosomes](#)), [18](#)
- getChunkSize, GenomicTiles-method ([getChromosomes](#)), [18](#)
- getCoordinates, [20](#)
- getCoordinates, GenomicTiles-method ([getCoordinates](#)), [20](#)
- getFits ([GenoGAM-methods](#)), [12](#)
- getFits, GenoGAM-method ([GenoGAM-methods](#)), [12](#)
- getIndex, [20](#)
- getIndex, GenomicTiles-method ([getIndex](#)), [20](#)
- getIndexCoordinates, [21](#)
- getIndexCoordinates, GenomicTiles-method ([getIndexCoordinates](#)), [21](#)
- getOverhangSize ([getChromosomes](#)), [18](#)
- getOverhangSize, GenomicTiles-method ([getChromosomes](#)), [18](#)
- getTile, [22](#)
- getTile, GenomicTiles-method ([getTile](#)), [22](#)
- getTileNumber ([getChromosomes](#)), [18](#)
- getTileNumber, GenomicTiles-method ([getChromosomes](#)), [18](#)
- getTileSize ([getChromosomes](#)), [18](#)
- getTileSize, GenomicTiles-method ([getChromosomes](#)), [18](#)
- IQR, GenomicTiles-method ([Summary, GenomicTiles-method](#)), [30](#)
- mad, GenomicTiles-method ([Summary, GenomicTiles-method](#)), [30](#)
- makeTestGenoGAM, [22](#)
- makeTestGenoGAMDataSet, [23](#)

makeTestGenomicTiles, [23](#)  
mean, GenomicTiles-method  
    (Summary, GenomicTiles-method),  
    [30](#)  
median, GenomicTiles-method  
    (Summary, GenomicTiles-method),  
    [30](#)  
  
plot.GenoGAM, [24](#)  
  
qualityCheck, [24](#)  
  
rowRanges, GenoGAM-method  
    (GenoGAM-methods), [12](#)  
  
sd, GenomicTiles-method  
    (Summary, GenomicTiles-method),  
    [30](#)  
sizeFactors, GenoGAMDataSet-method, [25](#)  
sizeFactors<-, GenoGAMDataSet, ANY-method  
    (sizeFactors, GenoGAMDataSet-method),  
    [25](#)  
subset, GenoGAM-method, [26](#)  
subset, GenoGAMDataSet-method, [26](#)  
subset, GenomicTiles-method, [27](#)  
subsetByOverlaps, GenoGAM, ANY-method,  
    [28](#)  
subsetByOverlaps, GenoGAMDataSet, GRanges-method,  
    [28](#)  
subsetByOverlaps, GenomicTiles, GRanges-method,  
    [29](#)  
SummarizedExperiment, [17](#)  
Summary, GenomicTiles-method, [30](#)  
  
tileSettings, [32](#)  
tileSettings, GenomicTiles-method  
    (tileSettings), [32](#)  
  
until, [32](#)  
until, GenomicTiles-method (until), [32](#)  
  
var, GenomicTiles, ANY-method  
    (Summary, GenomicTiles-method),  
    [30](#)  
view, [33](#)  
view, GenoGAM-method, [34](#)  
view, GenomicTiles-method (view), [33](#)  
  
writeToBEDFile, [34](#)