Package 'GenoGAM'

October 12, 2016

Type Package

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

Version 1.0.3

Date 2016-03-13

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)

Suggests BiocStyle, chipseq (>= 1.21.2), testthat, knitr

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 5.0.1.9000

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' 'genogam.R' 'helper.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` *GenomicTiles to DataFrame*

Description

GenomicTiles to DataFrame

See Also

Other res: `GenoGAMDataSetToDataFrame`

`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 *Check data compliance with tile settings*

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)
```

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>

`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 *The /codeGRanges of the underlying data*

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 *Access the* design *slot*

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
```

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

Description

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

---

genogam *genogam*

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 Negativ Binomial distribution (mgcv::nb()).

Usage

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

Arguments

ggd	A GenoGAMDataSet object to be fitted.
lambda	The penalization parameter. Will be estimated if missing.
family	A distribution family object. So far only mgcv::nb() is allowed.
bpknots	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.
kfolds	An integer number giving the number of k-folds to be used in cross validation, if parameters need to be estimated.
intervallSize	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*fragment size. Otherwise, depending on the density of the data, they should not exceed the size of a short read.
m	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 finals 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.

`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>

GenoGAMDataSet *GenoGAMDataSet constructor.*

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 = ".", ...)
```

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.
`...`	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 schould 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: ~ s(x) + s(x, by = "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 usefull 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)

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

## End(Not run)
```

`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 midpoint 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 optiomisation 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 fropm SummarizedExperiment. 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 SummarizedExperiment 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` *The single entries of the tile settings*

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

object	A /codeGenomicTiles object.
...	Additional arguments

Value

An integer value, or in case of /codegetChromosomes a /codeGRanges object

Author(s)

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

Examples

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

---

getChunkIndex *Compute the index for chunks instead tiles*

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

Value

A /codeGRanges object representing the index

Author(s)

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

Examples

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

---

getCoordinates *Accessor to the /codecoordinates slot*

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)
```

getFits *Accessor to* fits *slot*

Description

The fits slot contains the fitted values of the model

Usage

```
getFits(object)

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

Arguments

object	A GenomicTiles object

Value

A data.frame of the fits

Examples

```
gg <- makeTestGenoGAM()
fits <- getFits(gg)
```

getIndex *Accessor to the 'index' slot*

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 /codeGRanges 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 /codeGenomicTiles 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 /codeGranges 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 *Tile extraction as a DataFrame*

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+07)
```

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 *Make an example /codeGenoGAM*

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()
```

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

object	A GenoGAMDataSet object.
value	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 subsetted 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 /codeGenomciTiles

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 subsetted 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 GenomciTiles

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 *Return tile settings*

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)
```

---

untile *Set index to chunkIndex*

Description

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

Usage

```
untile(object, ...)

## S4 method for signature 'GenomicTiles'
untile(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 <- untile(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

object	A GenomicTiles object
...	Additional arguments
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

```
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

Cbinding 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))
```

`[[,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