Package 'GenoGAM'

November 11, 2019

Type Package

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

Version 2.4.0

Date 2019-06-08

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

Encoding UTF-8

Depends R (>= 3.5), SummarizedExperiment (>= 1.1.19), HDF5Array (>= 1.8.0), rhdf5 (>= 2.21.6), S4Vectors (>= 0.23.18), Matrix (>= 1.2-8), data.table (>= 1.9.4)

Imports Rcpp (>= 0.12.14), sparseinv (>= 0.1.1), Rsamtools (>= 1.18.2), GenomicRanges (>= 1.23.16), BiocParallel (>= 1.5.17), DESeq2 (>= 1.11.23), futile.logger (>= 1.4.1), GenomeInfoDb (>= 1.7.6), GenomicAlignments (>= 1.7.17), IRanges (>= 2.5.30), Biostrings (>= 2.39.14), DelayedArray (>= 0.3.19), methods, stats

LinkingTo Rcpp, RcppArmadillo

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

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 6.1.0

biocViews Regression, DifferentialPeakCalling, ChIPSeq, DifferentialExpression, Genetics, Epigenetics, WholeGenome, ChipOnChip, ImmunoOncology

Collate 'Coordinates-class.R' 'GenoGAMFamily-class.R' 'helper.R' 'GenoGAMSettings-class.R' 'GenoGAM-class.R' 'GenoGAM-package.R' 'GenoGAMDataSet-class.R' 'GenoGAMDataSetList-class.R' 'GenoGAMList-class.R' 'GenoGAMSetup-class.R' 'RcppExports.R' 'background.R' 'broadPeaks.R' 'cv.R' 'diffBinding.R' 'estimation.R' 'genogam.R' 'hdf5_handler.R' 'misc.R' 'narrowPeaks.R' 'peakCalling.R' 'plotting.R' 'pvals.R' 'readData.R' 'sf.R' 'zzz.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@protonmail.com>

git_url https://git.bioconductor.org/packages/GenoGAM

git_branch RELEASE_3_10

git_last_commit 6753a93

git_last_commit_date 2019-10-29

Date/Publication 2019-11-10

R topics documented:

asCoordinates *IRanges to Coordinates*

Description

IRanges to Coordinates

Coordinates to IRanges

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 colnames of the fitted object. By default all are 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 treshold 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 seperate 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 all found peaks are returned.

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 but taken as +- 100bp around the peak summit. A function for a more statistical estimation of the borders is being implemented. Also narrow peaks provide an occupancy estimate at the peak position, while broad peaks give the average occupancy accross the region.

Value

A list of data.tables 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 `writeToBEDFile`).

Author(s)

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

Examples

```
## creating test GenoGAM object
gg <- makeTestGenoGAM()
## call peaks
peaks <- callPeaks(gg)
peaks
```

`computeRegionSignificance`

Compute significance for given regions

Description

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

Usage

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

Arguments

`fit`	A GenoGAM object containing the fit
`regions`	A GRanges object of regions of interest
`smooth`	Which fit should be used. The names should be equivalent to the column names of the object. Lookup with `colnames(my_GenoGAM_object)`

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

```
## make test GenoGAM
gg <- makeTestGenoGAM()
## make region
region <- GRanges("chrXYZ", IRanges(c(2000, 4000, 6000), c(3000, 5000, 9000)))
res <- computeRegionSignificance(gg, region)
res
```

`computeSignificance` *computeSignificance*

Description

The function computes positionwise p-values

Usage

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

Arguments

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

Details

Note, that in case the data is stored in HDF5 format, the pvalue 'group' is added on hard drive. That is, unlike any other function in R, where the input object is not changed, it actually is in this case. If one wishes to have HDF5 data without the pvalue 'group', one has to backup the HDF5 files prior to computation or delete them after with `rhdf5::h5delete`

Value

An updated GenoGAM object, where the pvalue slot is added.

Author(s)

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

Examples

```
## make test GenoGAM
gg <- makeTestGenoGAM()
## compute pvalues
computeSignificance(gg)
pvalue(gg)
```

`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.

Value

A GenoGAMDataSet object, where the sizeFactors slot is updated.

Author(s)

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

Examples

```
ggd <- makeTestGenoGAMDataSet()
ggd <- computeSizeFactors(ggd)
sizeFactors(ggd)
groups <- list(c("wt_1", "wt_2"), c("mutant_1", "mutant_2"))
ggd <- computeSizeFactors(ggd, factorGroups = groups)
sizeFactors(ggd)
```

`Coordinates-class` *Coordinates class*

Description

This class mainly exists to overcome the lack of long integer use in R and the IRanges class in particular. The object is basically a restricted DataFrame object, with a few custom methods to mimic the behaviour of IRanges.

Coordinates is the constructor function for the Coordinates-class.

Usage

```
Coordinates(start = NULL, end = NULL)

## S4 method for signature 'Coordinates'
length(x)

## S4 method for signature 'Coordinates'
nrow(x)

## S4 method for signature 'Coordinates'
ncol(x)

## S4 method for signature 'Coordinates'
start(x)

## S4 method for signature 'Coordinates'
end(x)

## S4 method for signature 'Coordinates'
width(x)

## S4 replacement method for signature 'Coordinates'
end(x) <- value

## S4 replacement method for signature 'Coordinates'
start(x) <- value

## S4 method for signature 'Coordinates'
width(x)
```

Arguments

start	The start vector, must be of same length as the end vector
end	The end vector, must be of same length as the start vector
x	The Coordinates object
value	A numeric vector. For replace method in 'start' and 'end'.

Value

An object of class Coordinates

Methods (by generic)

- length: The length method
- nrow: The nrow method
- ncol: The ncol method
- start: The start accessor
- end: The end accessor
- width: The width accessor
- end<-: Replacement method for end

- `start<-`: Replacement method for end
- `width`: The width accessor

Author(s)

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

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

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.

Author(s)

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

Authors:

- Alexander Engelhardt <alexander.engelhardt@ibe.med.uni-muenchen.de>
- Julien Gagneur <gagneur@in.tum.de>

See Also

Useful links:

- https://github.com/gstricker/GenoGAM
- Report bugs at https://github.com/gstricker/GenoGAM/issues

`genogam` *genogam*

Description

The main modelling function.

Usage

```
genogam(ggd, lambda = NULL, theta = NULL, family = "nb", eps = 0,
  kfolds = 10, intervalSize = 20, regions = 20, order = 2, m = 2)
```

Arguments

ggd	The GenoGAMDataSet object to be fitted
lambda	The penalization parameter. If NULL (default) estimated by cross validation. So far only one parameter for all splines is supported.
theta	The global overdispersion parameter. If NULL (default) estimated by cross validation.
family	The distribution to be used. So far only Negative-Binomial (nb) is supported.
eps	The factor for additional first-order regularization. This should be zero (default) in most cases. It can be useful for sparse data with many zero-counts regions or very low coverage. In this cases it is advised to use a small factor like 0.01, which would penalize those regions but not the ones with higher coverage. See Wood S., Generalized Additive Models (2006) for more.
kfolds	The number of folds for cross validation
intervalSize	The size of the hold-out intervals in cross validation. If replicates are present, this can easily be increased to twice the fragment size to capture more of the local correlation. If no replicates are present, keep the number low to avoid heavy interpolation (default).
regions	How many regions should be used in cross validation? The number is an upper limit. It is usually corrected down, such that the total number of models computed during cross validation does not exceed the total number of models to compute for the entire genome. This is usually the case for small organisms such as yeast.
order	The order of the B-spline basis, which is order + 2, where 0 is the lowest order. Thus order = 2 is equivalent to cubic order (= 3).
m	The order of penalization. Thus m = 2 penalizes the second differences.

Value

The fit as a GenoGAM object

Author(s)

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

Examples

```
ggd <- makeTestGenoGAMDataSet(sim = TRUE)
res <- genogam(ggd, lambda = 266.8368, theta = 2.415738)
```

GenoGAM-class *GenoGAM class*

Description

This is the class the holds the complete model as well all hyperparameters and settings that were used to fit it. It extends the RangedSummarizedExperiment class by adding a couple of more slots to hold hyperparameters and settings. The 'assays' slot holds the basepair fit and standard deviation. Additionally all knot positions and beta coefficients will be stored in the 'smooths' slot in order to be able to make use of the piecewise function that produces the fit. For information on the slots inherited from SummarizedExperiment check the respective class.

The GenoGAM constructor, not designed to be actually used, by the user. Rather to be a point of reference and documentation for slots and how to access them.

Usage

```
GenoGAM(..., ggd = NULL, fromHDF5 = FALSE, split = FALSE)

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

## S4 method for signature 'GenoGAM'
sizeFactors(object)

## S4 method for signature 'GenoGAM'
getSettings(object)

## S4 method for signature 'GenoGAM'
getFamily(object)

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

## S4 method for signature 'GenoGAM'
getParams(object)

## S4 method for signature 'GenoGAM'
getCoefs(object)

## S4 method for signature 'GenoGAM'
getKnots(object)

## S4 method for signature 'GenoGAM,missing'
assay(x, i)

## S4 method for signature 'GenoGAM'
fits(object)

## S4 method for signature 'GenoGAM'
se(object)

## S4 method for signature 'GenoGAM'
pvalue(object)

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

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

## S4 method for signature 'GenoGAM'
is.HDF5(object)

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

Arguments

`...`	Slots of the GenoGAM class. See the slot description.
`ggd`	The initial GenoGAMDataSet object. Only needed if fromHDF5 is TRUE.
`fromHDF5`	A convenience argument to create a GenoGAM object from the already computed fits that are stored as HDF5 files
`split`	A logical argument indicating if the model was fitted in a per-chromosome fashion or not. Only needed if fromHDF5 is TRUE.
`object, x`	For use of S4 methods. The GenoGAM object.
`i`	A GRanges object (only for subsetting)

Value

An object of the type GenoGAM.

Methods (by generic)

- `design`: An accessor to the design slot
- `sizeFactors`: An accessor to the sizeFactors slot
- `getSettings`: An accessor to the settings slot
- `getFamily`: An accessor to the family slot
- `colData`: An accessor to the factorialDesign slot.
- `getParams`: An accessor to the params slot
- `getCoefs`: An accessor to the coefs slot
- `getKnots`: An accessor to the knots slot
- `assay`: The accessor to the fits and standard errors
- `fits`: An accessor to the fits
- `se`: An accessor to the standard errors
- `pvalue`: An accessor to the pvalues
- `colnames`: column names of GenoGAM
- `dimnames`: The names of the dimensions of GenoGAM
- `is.HDF5`: A boolean function that is true if object uses HDF5 backend
- `[`: Additional subsetting by single brackets

Slots

`family` A GenoGAMFamily object

`design` The formula of the model

`sizeFactors` The offset used in the model.

`factorialDesign` The factorial design used. The same as colData in the GenoGAMDataSet

`params` All hyperparameters used to fit the data. The parameters estimated by cross validation can also be found here. But the parameters used in cross validation are in the settings slot.

`settings` A GenoGAMSettings object representing the global settings that were used to compute the model.

`coefs` The coefficients of the knots

`knots` The relative knot positions

`hdf5` A logical slot indicating if the data is stored in a HDF5 format on hard drive

Author(s)

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

Examples

```
## creating test GenoGAM object
gg <- makeTestGenoGAM()
gg

## using accessors
design(gg)
sizeFactors(gg)
getSettings(gg)
getFamily(gg)
colData(gg)
getParams(gg)
getCoefs(gg)
getKnots(gg)
rowRanges(gg)
assay(gg)
assays(gg)
fits(gg)
se(gg)
```

GenoGAMDataSet-class *GenoGAMDataSet*

Description

The GenoGAMDataSet class contains the pre-processed raw data and additional slots that define the input and framework for the model. It extends the RangedSummarizedExperiment class by adding an index that defines ranges on the entire genome, mostly for purposes of parallel evaluation. Furthermore adding a couple more slots to hold information such as experiment design. It also contains the GenoGAMSettings class that defines global settings for the session. For information on the slots inherited from SummarizedExperiment check the respective class.

GenoGAMDataSet is the constructor function for the GenoGAMDataSet-class.

Usage

```
GenoGAMDataSet(experimentDesign, design, chunkSize = NULL,
  overhangSize = NULL, directory = ".", settings = NULL,
  hdf5 = FALSE, split = hdf5, fromHDF5 = FALSE, ignoreM = FALSE,
  ...)

## S4 method for signature 'GenoGAMDataSet'
getIndex(object)

## S4 method for signature 'GenoGAMDataSet'
getCountMatrix(object)

## S4 method for signature 'GenoGAMDataSet'
tileSettings(object)

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

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

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

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

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

## S4 method for signature 'GenoGAMDataSet'
getTileNumber(object)

## S4 method for signature 'GenoGAMDataSet'
is.HDF5(object)

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

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

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

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

## S4 replacement method for signature 'GenoGAMDataSet,numeric'
getChunkSize(object) <- value

## S4 replacement method for signature 'GenoGAMDataSet,numeric'
```

```
getTileSize(object) <- value

## S4 replacement method for signature 'GenoGAMDataSet,numeric'
getOverhangSize(object) <- value

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

Arguments

experimentDesign	Either a character object specifying the path to a delimited text file (the delimiter will be determined automatically), a data.frame specifying the experiment design or a RangedSummarizedExperiment object with the GPos class being the rowRanges. See details for the structure of the experimentDesign.
design	A formula object. See details for its structure.
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.
directory	The directory from which to read the data. By default the current working directory is taken.
settings	A GenoGAMSettings object. Not needed by default, but might be of use if only specific regions should be read in. See GenoGAMSettings.
hdf5	Should the data be stored on HDD in HDF5 format? By default this is disabled, as the Rle representation of count data already provides a decent compression of the data. However in case of large organisms, a complex experiment design or just limited memory, this might further decrease the memory footprint. Note this only applies to the input count data, results are usually stored in HDF5 format due to their space requirements for type double. Exceptions are small organisms like yeast.
split	A logical argument specifying if the data should be stored as a list split by chromosome. This is useful and necessary for huge organisms like human, as R does not support long integers.
fromHDF5	A logical argument specifying if the data is already present in form of HDF5 files and should be rather read in from there.
ignoreM	A logical argument to ignore the Mitochondria DNA on data read in. This is useful, if one is not interested in chrM, but it's size prevents the tiles to be larger, as all tiles has to be of some size.
...	Further parameters, mostly for arguments of custom processing functions or to specify a different method for fragment size estimation. See details for further information.
object	For use of S4 methods. The GenoGAMDataSet object.
value	For use of S4 methods. The value to be assigned to the slot.

Value

An object of class GenoGAMDataSet or the respective slot.

Methods (by generic)

- `getIndex`: An accessor to the index slot
- `getCountMatrix`: An accessor to the countMatrix slot
- `tileSettings`: The accessor to the list of settings, that were used to generate the tiles.
- `dataRange`: The actual underlying GRanges showing the range of the data.
- `getChromosomes`: A GRanges object representing the chromosomes or chromosome regions on which the model will be computed
- `getTileSize`: The size of the tiles
- `getChunkSize`: The size of the chunks
- `getOverhangSize`: The size of the overhang (on one side)
- `getTileNumber`: The total number of tiles
- `is.HDF5`: A boolean function that is true if object uses HDF5 backend
- `design`: Access to the design slot.
- `design<-`: Replace method of the design slot.
- `sizeFactors`: Access to the sizeFactors slot
- `sizeFactors<-`: Replace method of the sizeFactors slot
- `getChunkSize<-`: Replace method of the chunkSize parameter, that triggers a new computation of the tiles based on the new chunk size.
- `getTileSize<-`: Replace method of the tileSize parameter, that triggers a new computation of the tiles based on the new tile size.
- `getOverhangSize<-`: Replace method of the overhangSize parameter, that triggers a new computation of the tiles based on the new overhang size.
- `getTileNumber<-`: Replace method of the tileNumber parameter, that triggers a new computation of the tiles based on the new number of tiles.

Slots

`settings` The global and local settings that will be used to compute the model.

`design` The formula describing how to evaluate the data. See details.

`sizeFactors` The normalized values for each sample. A named numeric vector.

`index` A GRanges object representing an index of the ranges defined on the genome. Mostly used to store tiles.

`hdf5` A logical slot indicating if the object should be stored as HDF5

`countMatrix` Either a matrix or HDF5Matrix to store the sums of counts of the regions (could also be seen as bins) for later use especially by DESeq2

Config

The config file/data.frame contains the actual experiment design. It 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/Formula

Design must be a formula. At the moment only the following is possible: Either ~ s(x) for a smooth fit over the entire data or s(x, by = myColumn), where 'myColumn' is a column name in the experimentDesign. Any combination of this is possible:

~ s(x) + s(x, by = myColumn) + s(x, by = ...) + ...

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. '

Further parameters

In case of single-end data it 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.

Author(s)

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

Examples

```
# Build from config file

config <- system.file("extdata/Set1", "experimentDesign.txt", package = "GenoGAM")
dir <- system.file("extdata/Set1/bam", package = "GenoGAM")

## For all data
ggd <- GenoGAMDataSet(config, chunkSize = 1000, overhangSize = 200,
    design = ~ s(x) + s(x, by = genotype), directory = dir)
ggd

## Read data of a particular chromosome
settings <- GenoGAMSettings(chromosomeList = "chrXIV")
ggd <- GenoGAMDataSet(config, chunkSize = 1000, overhangSize = 200,
    design = ~ s(x) + s(x, by = genotype), directory = dir,
    settings = settings)
ggd

## Read data of particular range
region <- GenomicRanges::GRanges("chrI", IRanges(10000, 15000))
params <- Rsamtools::ScanBamParam(which = region)
settings <- GenoGAMSettings(bamParams = params)
ggd <- GenoGAMDataSet(config, chunkSize = 1000, overhangSize = 200,
    design = ~ s(x) + s(x, by = genotype), directory = dir,
    settings = settings)
ggd
```

```
# Build from data.frame config

df <- read.table(config, header = TRUE, sep = '\t')
ggd <- GenoGAMDataSet(df, chunkSize = 1000, overhangSize = 200,
    design = ~ s(x) + s(x, by = genotype), directory = dir,
    settings = settings)
ggd

# Build from SummarizedExperiment

gr <- GenomicRanges::GPos(GRanges("chr1", IRanges(1, 10000)))
seqlengths(gr) <- 1e6
df <- S4Vectors::DataFrame(colA = 1:10000, colB = round(runif(10000)))
se <- SummarizedExperiment::SummarizedExperiment(rowRanges = gr, assays = list(df))
ggd <- GenoGAMDataSet(se, chunkSize = 2000, overhangSize = 250,
                      design = ~ s(x) + s(x, by = experiment))
ggd
```

GenoGAMDataSetList-class
GenoGAMDataSetList

Description

The GenoGAMDataSetList class contains the pre-processed raw data and additional slots that define the input and framework for the model. It extends upon the idea of the GenoGAMDataSet class to make it possible to store genomes and data of size > 2^31 (maximum size of integers in R). Thus the only difference to a GenoGAMDataSet is the arrangement as a list of RangedSummarizedExperiments under the hood. On the surface is does still behave like a GenoGAMDataSet. It is not intended to be used by the user. For more information check the GenoGAMDataSet class documentation.

GenoGAMDataSetList is the constructor function for the GenoGAMDataSetList-class.

Usage

```
GenoGAMDataSetList(...)

## S4 method for signature 'GenoGAMDataSetList'
dim(x)

## S4 method for signature 'GenoGAMDataSetList'
length(x)

## S4 method for signature 'GenoGAMDataSetList'
seqlengths(x)

## S4 method for signature 'GenoGAMDataSetList'
seqlevels(x)

## S4 method for signature 'GenoGAMDataSetList'
seqlevelsInUse(x)
```

```
## S4 method for signature 'GenoGAMDataSetList'
colData(x, ...)

## S4 method for signature 'GenoGAMDataSetList'
rowRanges(x, ...)

## S4 method for signature 'GenoGAMDataSetList,ANY'
assay(x, i, ...)

## S4 method for signature 'GenoGAMDataSetList'
assays(x, ..., withDimnames = TRUE)

## S4 method for signature 'GenoGAMDataSetList'
colnames(x)

## S4 method for signature 'GenoGAMDataSetList'
getIndex(object)

## S4 method for signature 'GenoGAMDataSetList'
getCountMatrix(object)

## S4 method for signature 'GenoGAMDataSetList'
tileSettings(object)

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

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

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

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

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

## S4 method for signature 'GenoGAMDataSetList'
getTileNumber(object)

## S4 method for signature 'GenoGAMDataSetList'
is.HDF5(object)

## S4 method for signature 'GenoGAMDataSetList'
design(object)

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

## S4 method for signature 'GenoGAMDataSetList'
```

```
sizeFactors(object)

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

## S4 replacement method for signature 'GenoGAMDataSetList,numeric'
getChunkSize(object) <- value

## S4 replacement method for signature 'GenoGAMDataSetList,numeric'
getTileSize(object) <- value

## S4 replacement method for signature 'GenoGAMDataSetList,numeric'
getOverhangSize(object) <- value

## S4 replacement method for signature 'GenoGAMDataSetList,numeric'
getTileNumber(object) <- value
```

Arguments

`...`	The slots and their respective values
`withDimnames`	For use of S4 methods.
`object, x`	For use of S4 methods. The GenoGAMDataSetList object.
`value, i`	For use of S4 methods. The value to be assigned to the slot.

Value

An object of class GenoGAMDataSetList

Methods (by generic)

- `dim`: Get the dimension of the object
- `length`: The length of the object
- `seqlengths`: The seqlengths of the object
- `seqlevels`: The seqlevels of the object
- `seqlevelsInUse`: The seqlevelsInUse of the object
- `colData`: get colData from the first element of the SummarizedExperiment list
- `rowRanges`: get a list of rowRanges from the GenoGAMDataSetList object
- `assay`: get a list of assays from the GenoGAMDataSetList object
- `assays`: get a list of list of assays from the GenoGAMDataSetList object. Just for completeness, shouldn't be needed.
- `colnames`: get colnames from the first element of the SummarizedExperiment list
- `getIndex`: accessor to the index slot
- `getCountMatrix`: An accessor to the countMatrix slot
- `tileSettings`: The accessor to the list of settings, that were used to generate the tiles.
- `dataRange`: The actual underlying GRanges showing the range of the data.
- `getChromosomes`: A GRanges object representing the chromosomes or chromosome regions on which the model will be computed
- `getTileSize`: The size of the tiles

- `getChunkSize`: The size of the chunks
- `getOverhangSize`: The size of the overhang (on one side)
- `getTileNumber`: The total number of tiles
- `is.HDF5`: A boolean function that is true if object uses HDF5 backend
- `design`: Access to the design slot.
- `design<-`: Replace method of the design slot.
- `sizeFactors`: Access to the sizeFactors slot
- `sizeFactors<-`: Replace method of the sizeFactors slot
- `getChunkSize<-`: Replace method of the chunkSize parameter, that triggers a new computation of the tiles based on the new chunk size.
- `getTileSize<-`: Replace method of the tileSize parameter, that triggers a new computation of the tiles based on the new tile size.
- `getOverhangSize<-`: Replace method of the overhangSize parameter, that triggers a new computation of the tiles based on the new overhang size.
- `getTileNumber<-`: Replace method of the tileNumber parameter, that triggers a new computation of the tiles based on the new number of tiles.

Slots

`settings` The global and local settings that will be used to compute the model.

`design` The formula describing how to evaluate the data. See details.

`sizeFactors` The normalized values for each sample. A named numeric vector.

`index` A GRanges object representing an index of the ranges defined on the genome. Mostly used to store tiles.

`data` A list of RangedSummarizedExperiment objects

`id` A GRanges object keeping the identifiers assigning the regions to the respective list elements

`hdf5` A logical slot indicating if the object should be stored as HDF5

`countMatrix` Either a matrix or HDF5Matrix to store the sums of counts of the regions (could also be seen as bins) for later use especially by DESeq2

Author(s)

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

GenoGAMFamily-class *GenoGAMFamily class*

Description

This class holds the distribution family with the log-likelihood functions and the first two derivatives, the gradient vector and the Hessian matrix. It is not meant to be used directly, but rather for development convenience.

Slots

`ll` The log-likelihood function. Gives a scalar

`gradient` The first derivative of the log-likelihood functions. Gives a vector.

`hessian` An integer indicating the family. Negative Binomial = 1, Quasi-Binomial = 2.

Author(s)

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

GenoGAMList-class *GenoGAMList class*

Description

This is the class the holds the complete model as well all hyperparameters and settings that were used to fit it. It is basically identical to the GenoGAM class, except for the data being inside a list of RangedSummarizedExperiment objects.

The GenoGAMList constructor, not designed to be actually used, by the user. Rather to be a point of reference and documentation for slots and how to access them.

Usage

```
GenoGAMList(..., ggd = NULL, fromHDF5 = FALSE)

## S4 method for signature 'GenoGAMList'
dim(x)

## S4 method for signature 'GenoGAMList'
length(x)

## S4 method for signature 'GenoGAMList'
seqlengths(x)

## S4 method for signature 'GenoGAMList'
seqlevels(x)

## S4 method for signature 'GenoGAMList'
seqlevelsInUse(x)

## S4 method for signature 'GenoGAMList'
design(object)

## S4 method for signature 'GenoGAMList'
sizeFactors(object)

## S4 method for signature 'GenoGAMList'
getSettings(object)

## S4 method for signature 'GenoGAMList'
getFamily(object)
```

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

## S4 method for signature 'GenoGAMList'
getParams(object)

## S4 method for signature 'GenoGAMList'
getCoefs(object)

## S4 method for signature 'GenoGAMList'
getKnots(object)

## S4 method for signature 'GenoGAMList,missing'
assay(x, i)

## S4 method for signature 'GenoGAMList'
assays(x, ..., withDimnames = TRUE)

## S4 method for signature 'GenoGAMList'
rowRanges(x, ...)

## S4 method for signature 'GenoGAMList'
fits(object)

## S4 method for signature 'GenoGAMList'
se(object)

## S4 method for signature 'GenoGAMList'
pvalue(object)

## S4 method for signature 'GenoGAMList'
colnames(x)

## S4 method for signature 'GenoGAMList'
dimnames(x)

## S4 method for signature 'GenoGAMList'
is.HDF5(object)
```

Arguments

`...`	Slots of the GenoGAM class. See the slot description.
`ggd`	The initial GenoGAMDataSet object. Only needed if fromHDF5 is TRUE.
`fromHDF5`	A convenience argument to create a GenoGAM object from the already computed fits that are stored as HDF5 files
`object, x`	For use of S4 methods. The GenoGAMList object.
`i`	A GRanges object (only for subsetting)
`withDimnames`	For use of S4 methods. The GenoGAMList object.

Value

An object of the type GenoGAM.

Methods (by generic)

- `dim`: Get the dimension of the object
- `length`: The length of the object
- `seqlengths`: The seqlengths of the object
- `seqlevels`: The seqlevels of the object
- `seqlevelsInUse`: The seqlevelsInUse of the object
- `design`: An accessor to the design slot
- `sizeFactors`: An accessor to the sizeFactors slot
- `getSettings`: An accessor to the settings slot
- `getFamily`: An accessor to the family slot
- `colData`: An accessor to the factorialDesign slot.
- `getParams`: An accessor to the params slot
- `getCoefs`: An accessor to the coefs slot
- `getKnots`: An accessor to the knots slot
- `assay`: The accessor to the fits and standard errors
- `assays`: get a list of list of assays from the GenoGAMList object. Just for completeness, shouldn't be needed.
- `rowRanges`: get a list of rowRanges from the GenoGAMList object
- `fits`: An accessor to the fits
- `se`: An accessor to the standard errors
- `pvalue`: An accessor to the pvalues
- `colnames`: column names of GenoGAMList
- `dimnames`: The names of the dimensions of GenoGAMList
- `is.HDF5`: A boolean function that is true if object uses HDF5 backend

Slots

`family` The name of the distribution family used

`design` The formula of the model

`sizeFactors` The offset used in the model.

`factorialDesign` The factorial design used. The same as colData in the GenoGAMDataSet

`params` All hyperparameters used to fit the data. The parameters estimated by cross validation can also be found here. But the parameters used in cross validation are in the settings slot.

`settings` A GenoGAMSettings object representing the global settings that were used to compute the model.

`data` A list of RangedSummarizedExperiment that holds the actual data

`id` A GRanges object keeping the identifiers assigning the regions to the respective list elements

`coefs` The coefficients of the knots

`knots` The relative knot positions

Author(s)

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

Examples

```
## creating test GenoGAM object
gg <- makeTestGenoGAM()
gg

## using accessors
design(gg)
sizeFactors(gg)
getSettings(gg)
getFamily(gg)
colData(gg)
getParams(gg)
getCoefs(gg)
getKnots(gg)
rowRanges(gg)
assay(gg)
assays(gg)
fits(gg)
se(gg)
```

GenoGAMSettings-class *GenoGAMSettings*

Description

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

Usage

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

Arguments

...	Any parameters corresponding to the slots and their possible values.

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. In case a custom function is used, it has to satisfy the following: It has to handle a GAlignments object as input and output a GRanges object of regions, e.g. fragments. This regions are in turn used to compute the coverage via the IRanges::coverage function. Note, that there is a difference between the GAlignments object in the single and paired end case.

Value

An object of class GenoGAMSettings

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 for possible settings. Usually used to set specific ranges, to read in.

`processFunction` A custom function on how to process raw data. Not used if center is TRUE/FALSE. This is not intended for the user, but if needed anyway, see details.

`optimMethod` The optiomisation method to be used in cross validation. See ?optim for a complete list.

`optimControl` List of control settings for the optim function. Almost all parameters are supported, with a couple of exceptions. See details. For a complete list of parameters see ?optim.

`estimControl` List of control settings for the parameter estimation algorithm.

`hdf5Control` List of control settings for the HDF5 backend

`dataControl` List of control settings for the processed data. The size of the region to use for the computation of the count matrix, that is later used by DESeq2. Also the size of the regions that will be used for Cross Validation. And the spacing between knots.

Author(s)

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

Examples

```
# Construct the class
GenoGAMSettings()

# Construct the class with custom parameters
## specify chromosomes
center <- FALSE
chromosomeList <- c('chr1', 'chr2')
GenoGAMSettings(center = center, chromosomeList = chromosomeList)

## Specify ranges
gr <- GenomicRanges::GRanges("chr1", IRanges(1, 10000))
bamParams <- Rsamtools::ScanBamParam(which = gr)
GenoGAMSettings(bamParams = bamParams, center = TRUE)
```

---

makeTestGenoGAM *Make an example /codeGenoGAM*

Description

Make an example /codeGenoGAM

Usage

```
makeTestGenoGAM()
```

Value

A /codeGenoGAM object

Examples

```
gg <- makeTestGenoGAM()
```

---

makeTestGenoGAMDataSet

Make an example /codeGenoGAMDataSet

Description

Make an example /codeGenoGAMDataSet

Usage

```
makeTestGenoGAMDataSet(sim = FALSE)
```

Arguments

sim	Use simulated data (TRUE) or test data from a real experiment

Value

A /codeGenoGAMDataSet object

Examples

```
realdt <- makeTestGenoGAMDataSet()
simdt <- makeTestGenoGAMDataSet(sim = TRUE)
```

---

makeTestGenoGAMDataSetList

Make an example /codeGenoGAMDataSet

Description

Make an example /codeGenoGAMDataSet

Usage

```
makeTestGenoGAMDataSetList()
```

Value

A /codeGenoGAMDataSet object

Examples

```
ggdl <- makeTestGenoGAMDataSetList()
```

makeTestGenoGAMList *Make an example /codeGenoGAMList*

Description

Make an example /codeGenoGAMList

Usage

```
makeTestGenoGAMList()
```

Value

A /codeGenoGAMList object

Examples

```
ggl <- makeTestGenoGAMList()
```

plotGenoGAM *The pot function for a GenoGAM object*

Description

This functions plots the fit of a given region and optionally the read counts from the GenoGAMDataSet object

Usage

```
plotGenoGAM(x, ggd = NULL, ranges = NULL, seqnames = NULL,
  start = NULL, end = NULL, scale = TRUE, cap = TRUE,
  log = FALSE, ...)
```

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 ob 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?
cap	If FALSE deactivates the cap that prevents to accidentaly plot a too larger area. The default cap is 1Mbp.
log	Should log values be plotted on the 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>

`readData` *Read Data function*

Description

This is the core function to read and parse raw data from a config file. At the moment only the BAM format is supported. It is not intended to be used by the user directly, as it is called internally by the GenoGAMDataSet constructor. However it is exported if people wish to separately assemble their data and construct the GenoGAMDataSet from SummarizedExperiment afterwards. It also offers the possibility to use the HDF5 backend.

Usage

```
readData(config, hdf5 = FALSE, split = FALSE,
  settings = GenoGAMSettings(), ...)
```

Arguments

`config`	A data.frame containing the experiment design of the model to be computed with the first three columns fixed. See the 'experimentDesign' parameter in `GenoGAMDataSet` or details here.
`hdf5`	Should the data be stored on HDD in HDF5 format? By default this is disabled, as the Rle representation of count data already provides a decent compression of the data. However in case of large organisms, a complex experiment design or just limited memory, this might further decrease the memory footprint.
`split`	If TRUE the data will be stored as a list of DataFrames by chromosome instead of one big DataFrame. This is only necessary if organisms with a genome size bigger than 2^31 (approx. 2.14Gbp) are analyzed, in which case Rs lack of long integers prevents having a well compressed Rle of sufficient size.
`settings`	A GenoGAMSettings object. Not needed by default, but might be of use if only specific regions should be read in. See `GenoGAMSettings`.
`...`	Further parameters that can be passed to low-level functions. Mostly to pass arguments to custom process functions. In case the default process functions are used, i.e. the default settings paramenter, the most interesting parameters might be fragment length estimator method from ?chipseq::estimate.mean.fraglen for single-end data.

Details

The config data.frame contains the actual experiment design. It 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 complete path to the BAM file.

The field 'paired', values TRUE for paired-end sequencing data, and FALSE for single-end sequencing data.

Other columns will be ignored by this function.

Value

A DataFrame of counts for each sample and position. Or if split = TRUE, a list of DataFrames by chromosomes

Author(s)

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

Examples

```
# Read data

## Set config file
config <- system.file("extdata/Set1", "experimentDesign.txt", package = "GenoGAM")
config <- read.table(config, header = TRUE, sep = '\t', stringsAsFactors = FALSE)
for(ii in 1:nrow(config)) {
    absPath <- system.file("extdata/Set1/bam", config$file[ii], package = "GenoGAM")
    config$file[ii] <- absPath
}

## Read all data
df <- readData(config)
df

## Read data of a particular chromosome
settings <- GenoGAMSettings(chromosomeList = "chrI")
df <- readData(config, settings = settings)
df

## Read data of particular range
region <- GenomicRanges::GRanges("chrI", IRanges(10000, 20000))
params <- Rsamtools::ScanBamParam(which = region)
settings <- GenoGAMSettings(bamParams = params)
df <- readData(config, settings = settings)
df
```

subset,GenoGAMDataSet-method
Subset methods for GenoGAMDataSet

Description

Subset methods for GenoGAMDataSet

Usage

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

## S4 method for signature 'GenoGAMDataSet,GRanges'
subsetByOverlaps(x, ranges,
  maxgap = -1L, minoverlap = 0L, type = c("any", "start", "end",
  "within", "equal"), invert = FALSE, ...)

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

Arguments

x	A GenoGAMDataSet object.
...	Further arguments. Mostly a logical statement in case of the 'subset' function. Note that the columnnames for chromosomes and positions are: 'seqnames' and 'pos'.
ranges, i	A GRanges object. In case of subsetting by double brackets 'i' is the index of the tile.
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 in). 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.
invert	If TRUE, keep only the query ranges that do _not_ overlap the subject.

Details

Those are various methods to subset the GenoGAMDataSet object. By logical statement or GRanges overlap. The '[' subsetter is just a short version of 'subsetByOverlaps'.

Value

A subsetted GenoGAMDataSet object.

Author(s)

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

References

Allen's Interval Algebra: James F. Allen: Maintaining knowledge about temporal intervals. In: Communications of the ACM. 26/11/1983. ACM Press. S. 832-843, ISSN 0001-0782

Examples

```
# subset by overlaps
ggd <- makeTestGenoGAMDataSet()
SummarizedExperiment::rowRanges(ggd)
gr <- GenomicRanges::GRanges("chrI", IRanges(10000,19000))
res <- IRanges::subsetByOverlaps(ggd, gr)
SummarizedExperiment::rowRanges(res)

# Subset by logical statement
ggd <- makeTestGenoGAMDataSet()
SummarizedExperiment::rowRanges(ggd)
res <- subset(ggd, seqnames == "chrI" & pos <= 17000)
SummarizedExperiment::rowRanges(res)
```

---

subset,GenoGAMDataSetList-method

Subset method for GenoGAMDataSetList

Description

Subset method for GenoGAMDataSetList

Usage

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

## S4 method for signature 'GenoGAMDataSetList,GRanges'
subsetByOverlaps(x, ranges,
  maxgap = -1L, minoverlap = 0L, type = c("any", "start", "end",
  "within", "equal"), invert = FALSE, ...)

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

Arguments

x	A GenoGAMDataSetList object.
...	Further arguments. Mostly a logical statement in case of the 'subset' function. Note that the columnnames for chromosomes and positions are: 'seqnames' and 'pos'.
ranges, i	A GRanges object. In case of subsetting by double brackets 'i' is the index of the tile.
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 in). 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.
invert	If TRUE, keep only the query ranges that do _not_ overlap the subject.

Details

Those are various methods to subset the GenoGAMDataSetList object. By logical statement or GRanges overlap. The '[' subsetter is just a short version of 'subsetByOverlaps'.

Value

A subsetted GenoGAMDataSetList object.

Author(s)

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

References

Allen's Interval Algebra: James F. Allen: Maintaining knowledge about temporal intervals. In: Communications of the ACM. 26/11/1983. ACM Press. S. 832-843, ISSN 0001-0782

subset,GenoGAMList-method
Subset method for GenoGAMList

Description

Subset method for GenoGAMList

Usage

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

## S4 method for signature 'GenoGAMList,GRanges'
subsetByOverlaps(x, ranges, maxgap = -1L,
  minoverlap = 0L, type = c("any", "start", "end", "within", "equal"),
  invert = FALSE, ...)

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

Arguments

x	A GenoGAMList object.
...	Further arguments. Mostly a logical statement in case of the 'subset' function. Note that the columnnames for chromosomes and positions are: 'seqnames' and 'pos'.
ranges, i	A GRanges object. In case of subsetting by double brackets 'i' is the index of the tile.
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 in). 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.
invert	If TRUE, keep only the query ranges that do _not_ overlap the subject.

Details

Those are various methods to subset the GenoGAMList object. By logical statement or GRanges overlap. The '[' subsetter is just a short version of 'subsetByOverlaps'.

Value

A subsetted GenoGAMList object.

Author(s)

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

References

Allen's Interval Algebra: James F. Allen: Maintaining knowledge about temporal intervals. In: Communications of the ACM. 26/11/1983. ACM Press. S. 832-843, ISSN 0001-0782

Summary,GenoGAMDataSet-method

Computing metrics

Description

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

Usage

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

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

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

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

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

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

## S4 method for signature 'GenoGAMDataSet'
IQR(x)

## S4 method for signature 'GenoGAMDataSetList'
mean(x)
```

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

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

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

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

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

Arguments

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

Value

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

```
ggd <- makeTestGenoGAMDataSet()
sum(ggd)
min(ggd)
max(ggd)
mean(ggd)
var(ggd)
sd(ggd)
median(ggd)
mad(ggd)
IQR(ggd)
```

Summary,GenoGAMDataSetList-method *Computing metrics*

Description

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

Usage

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

Arguments

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

Value

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

```
ggd <- makeTestGenoGAMDataSetList()
sum(ggd)
min(ggd)
max(ggd)
mean(ggd)
var(ggd)
sd(ggd)
median(ggd)
mad(ggd)
IQR(ggd)
```

---

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

Details

Note, the narrow peak calling process does not yet implement any functionality for estimating the start and end of a peak region. Thus the start and end is taken as -100 and +100 around the peak summit. This is mostly an arbitrary choice. A more statistical approach is in development.

Value

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

Author(s)

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

Index