Package 'CAGEr'

November 5, 2026

Title Analysis of CAGE (Cap Analysis of Gene Expression) sequencing data for precise mapping of transcription start sites and promoterome mining

Version 2.19.2

Date 2026-11-01

Imports BiocGenerics, BiocParallel, Biostrings, BSgenome, CAGEfightR, data.table, formula.tools, Seqinfo, GenomicAlignments (>= 1.45.1), GenomicFeatures (>= 1.61.4), GenomicRanges (>= 1.61.1), ggplot2 (>= 4.0.0), gtools, IRanges (>= 2.18.0), KernSmooth, Matrix, memoise, plyr, rlang, Rsamtools (>= 2.25.1), reshape2, rtracklayer (>= 1.69.1), S4Vectors (>= 0.27.5), scales, som, stringdist, stringi, SummarizedExperiment (>= 1.39.1), utils, vegan, VGAM

Depends methods, MultiAssayExperiment, R (>= 4.1.0)

Suggests BSgenome.Dmelanogaster.UCSC.dm3, BSgenome.Drerio.UCSC.danRer7, BSgenome.Hsapiens.UCSC.hg18, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Mmusculus.UCSC.mm9, DESeq2, FANTOM3and4CAGE, ggseqlogo, BiocStyle, knitr, rmarkdown, testthat (>= 3.1.5)

Description The _CAGEr_ package identifies transcription start sites (TSS) and their usage frequency from CAGE (Cap Analysis Gene Expression) sequencing data. It normalises raw CAGE tag count, clusters TSSs into tag clusters (TC) and aggregates them across multiple CAGE experiments to construct consensus clusters (CC) representing the promoterome. CAGEr provides functions to profile expression levels of these clusters by cumulative expression and rarefaction analysis, and outputs the plots in ggplot2 format for further facetting and customisation. After clustering, CAGEr performs analyses of promoter width and detects differential usage of TSSs (promoter shifting) between samples. CAGEr also exports its data as genome browser tracks, and as R objects for downsteam expression analysis by other Bioconductor packages such as DESeq2, CAGEfightR, or seqArchR.

License GPL-3

biocViews Preprocessing, Sequencing, Normalization, FunctionalGenomics, Transcription, GeneExpression, Clustering, Visualization

Collate 'Multicore.R' 'CTSS.R' 'CAGEexp.R' 'ClusteringFunctions.R' 'ClusteringMethods.R' 'CAGEr.R' 'Annotations.R'

'AggregationMethods.R' 'CAGEfightR.R' 'CAGEr-package.R'
'Paraclu.R' 'CorrelationMethods.R' 'GetMethods.R'
'CumulativeDistributionMethods.R' 'Distclu.R' 'ExportMethods.R'
'ExpressionProfilingMethods.R' 'ImportFunctions.R'
'SetMethods.R' 'ImportMethods.R' 'MergingMethods.R'
'NormalizationFunctions.R' 'NormalizationMethods.R'
'QCmethods.R' 'QuantileWidthMethods.R' 'ResetMethods.R'
'Richness.R' 'RleDataFrame.R' 'ShiftingFunctions.R'
'ShiftingMethods.R' 'StrandInvaders.R' 'TSSlogo.R'

LazyData true

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

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

git_branch devel

git_last_commit ddd5bf0

git_last_commit_date 2026-11-01

Repository Bioconductor 3.24

Date/Publication 2026-11-05

Author Vanja Haberle [aut],
Charles Plessy [cre],
Damir Baranasic [ctb],
Katalin Ferenc [ctb],
Sarvesh Nikumbh [ctb]

Maintainer Charles Plessy <charles.plessy@oist.jp>

Contents

CAGEr-package 4
.byCtss 5
.ctss_summary_for_clusters 5
.get.quant.pos 6
.powerLaw 7
aggregateTagClusters 8
annotateCTSS 10
bam2CTSS 11
CAGEexp-class 12
CAGEr-class 13
CAGEr_Multicore 14
coerceInBSgenome 15
ConsensusClusters-class 15
consensusClusters<- 16
consensusClustersDESeq2 16
consensusClustersGR 17
consensusClustersQuantile 18

consensusClustersTpm . . . 19
CTSS-class . . . 20
CTSScoordinatesGR . . . 21
CTSScumulativesTagClusters . . . 22
CTSSnormalizedTpmDF . . . 23
CTSStagCountDF . . . 24
CTSStoGenes . . . 25
cumulativeCTSSdistribution . . . 26
CustomConsensusClusters . . . 28
distclu . . . 29
exampleCAGEexp . . . 30
exampleZv9_annot . . . 31
exportToTrack . . . 33
expressionClasses . . . 36
FANTOM5humanSamples . . . 37
FANTOM5mouseSamples . . . 38
filteredCTSSidx . . . 38
flagByUpstreamSequences . . . 39
flagLowExpCTSS . . . 40
GeneExpDESeq2 . . . 42
GeneExpSE . . . 43
genomeName . . . 43
getCTSS . . . 45
getExpressionProfiles . . . 47
getShiftingPromoters . . . 49
hanabi . . . 51
hanabi-class . . . 53
hanabiPlot . . . 54
import.bam . . . 55
import.bam.ctss . . . 55
import.bedCTSS . . . 56
import.bedmolecule . . . 57
import.bedScore . . . 58
import.bigwig . . . 58
import.CAGEscanMolecule . . . 59
import.CTSS . . . 60
importPublicData . . . 60
inputFiles . . . 63
inputFilesType . . . 64
librarySizes . . . 65
loadFileIntoGPos . . . 66
mapStats . . . 67
mapStatsScopes . . . 68
mergeCAGEsets . . . 69
mergeSamples . . . 70
moleculesGR2CTSS . . . 71
normalizeTagCount . . . 72
paraclu . . . 74
parseCAGEscanBlocksToGrangeTSS . . . 77
plot.hanabi . . . 77
plotAnnot . . . 78
plotCorrelation . . . 81

plotExpressionProfiles . . . 85
plotInterquantileWidth . . . 86
plotReverseCumulatives . . . 88
quantilePositions . . . 90
quickEnhancers . . . 91
ranges2annot . . . 92
ranges2genes . . . 93
ranges2names . . . 94
resetCAGEexp . . . 95
rowsum.RleDataFrame . . . 96
rowSums.RleDataFrame . . . 97
sampleLabels . . . 97
scoreShift . . . 99
seqNameTotalsSE . . . 101
setColors . . . 102
Strand invaders . . . 103
summariseChrExpr . . . 104
TagClusters-class . . . 105
tagClustersGR . . . 105
TSSlogo . . . 106

Index **108**

`CAGEr-package` *CAGEr: Analysis of CAGE (Cap Analysis of Gene Expression) sequencing data for precise mapping of transcription start sites and promoterome mining*

Description

The _CAGEr_ package identifies transcription start sites (TSS) and their usage frequency from CAGE (Cap Analysis Gene Expression) sequencing data. It normalises raw CAGE tag count, clusters TSSs into tag clusters (TC) and aggregates them across multiple CAGE experiments to construct consensus clusters (CC) representing the promoterome. CAGEr provides functions to profile expression levels of these clusters by cumulative expression and rarefaction analysis, and outputs the plots in ggplot2 format for further facetting and customisation. After clustering, CAGEr performs analyses of promoter width and detects differential usage of TSSs (promoter shifting) between samples. CAGEr also exports its data as genome browser tracks, and as R objects for downsteam expression analysis by other Bioconductor packages such as DESeq2, CAGEfightR, or seqArchR.

Author(s)

Maintainer: Charles Plessy <charles.plessy@oist.jp>

Authors:

- Vanja Haberle <vanja.haberle@gmail.com>

Other contributors:

- Damir Baranasic <damir.baranasic@lms.mrc.ac.uk> [contributor]
- Katalin Ferenc <k.t.ferenc@ncmbm.uio.no> [contributor]
- Sarvesh Nikumbh <s.nikumbh@lms.mrc.ac.uk> [contributor]

.byCtss *Apply functions to identical CTSSes.*

Description

.byCTSS is a private function using data.table objects to preform grouping operations at a high performance. These functions use *non-standard evaluation* in a context that raises warnings in R CMD check. By separating these functions from the rest of the code, I hope to make the workarounds easier to manage.

Usage

```
.byCtss(ctssDT, colName, fun)

## S4 method for signature 'data.table'
.byCtss(ctssDT, colName, fun)
```

Arguments

ctssDT	A data.table::data.table representing CTSSes.
colName	The name of the column on which to apply the function.
fun	The function to apply.

Examples

```
ctssDT <- data.table::data.table(
  chr       = c("chr1", "chr1", "chr1", "chr2"),
  pos       = c(1     , 1     , 2     , 1     ),
  strand    = c("+"   , "+"   , "-"   , "-"   ),
  tag_count = c(1     , 1     , 1     , 1     ))
ctssDT
CAGEr:::.byCtss(ctssDT, "tag_count", sum)
```

.ctss_summary_for_clusters *Summarise CTSSs included in clusters*

Description

Summarise CTSSs included in clusters

Usage

```
.ctss_summary_for_clusters(ctss, clusters)
```

Arguments

ctss	A CTSS object.
clusters	A TagClusters, ConsensusClusters or any other object implementing the GRanges class.

Value

The clusters object with a new dominant_CTSS metadata in CTSS format reporting the genomic coordinate and expression score of most highly expressed position in each cluster, plus a nr_ctss metadata reporting the number of expressed CTSSs in each cluster.

Examples

```
# See also benchmarks/dominant_ctss.md
(ctss <- CTSS( 'chr1', IRanges(start = 1:10, end = 1:10)
             , '+', score = c(1, 0, 0, 1, 2, 0, 2, 1, 0, 1)))
(clusters <- GRanges( 'chr1', IRanges(start = c(1,9)
                    , end = c(8,10)), '+')) |> as("TagClusters")

# The function assumes that all CTSSes have a score above zero
.ctss_summary_for_clusters(ctss[score(ctss)>0], clusters)
# If not the case, it will give incorrect nr_ctss and  fail to remove singletons
.ctss_summary_for_clusters(ctss, clusters)

# The function needs its output to be sorted and is not going to check it.
.ctss_summary_for_clusters(rev(ctss), clusters)
.ctss_summary_for_clusters(ctss, rev(clusters))

# Ties are resolved with 5' preference for both plus and minus strands.
# This may create a small bias.
ctss_minus <- ctss
strand(ctss_minus) <- '-'
clusters_minus <- clusters
strand(clusters_minus) <- '-'
.ctss_summary_for_clusters(ctss_minus, clusters_minus)
```

.get.quant.pos *Get quantile positions*

Description

Private function that calculates position of quantiles for CTSS clusters based on distribution of tags within the clusters.

Usage

```
.get.quant.pos(cum.sums, clusters, q)
```

Arguments

cum.sums	Named list of vectors containing cumulative sum for each cluster (returned by the CTSScumulativesTagClusters or CTSScumulativesCC function).
clusters	TagClusters or ConsensusClusters object representing tag clusters or consensus clusters.
q	desired quantiles - single value or a vector of values.

Value

Returns the `clusters` object with one more metadata column per value in q, containing `Rle` integers giving the relative distance of the quantile boundaries to the start position.

Examples

```
cum.sums  <- RleList(`1` = Rle(1), `2` = cumsum(Rle(c(1, 1, 1, 2, 4, 0, 1, 1))))
clusters <- GRanges(c("chr1:100-101", "chr1:120-127"))
CAGEr:::.get.quant.pos(cum.sums, clusters, c(.2, .8))
```

`.powerLaw` *.powerLaw*

Description

Private funtion for normalizing CAGE tag count to a referent power-law distribution.

Usage

```
.powerLaw(tag.counts, fitInRange = c(10, 1000), alpha = 1.25, T = 10^6)
```

Arguments

`tag.counts`	Numerical values whose reverse cumulative distribution will be fitted to power-law (e.g. tag count or signal for regions, peaks, etc.)
`fitInRange`	Range in which the fitting is done (values outside of this range will not be considered for fitting)
`alpha`	Slope of the referent power-law distribution (the actual slope has negative sign and will be -1*alpha)
`T`	total number of tags (signal) in the referent power-law distribution.

Details

S4 Methods are provided for integer vectors, Rle objects, data.frame objects and DataFrame objects, so that the most complex objects can be deconstructed in simpler parts, normalized and reconstructed.

Value

Normalized values (vector of the same length as input values); i.e. what would be the value of input values in the referent distribution. Ouptut objects are numeric, possibly `Rle`-encoded or wrapped in `data.frames` or `DataFrames` according to the input.

References

Balwierz, P. J., Carninci, P., Daub, C. O., Kawai, J., Hayashizaki, Y., Van Belle, W., Beisel, C., et al. (2009). Methods for analyzing deep sequencing expression data: constructing the human and mouse promoterome with deepCAGE data. Genome Biology, 10(7), R79.

`aggregateTagClusters` *Aggregate TCs across all samples*

Description

Aggregates tag clusters (TCs) across all CAGE datasets within the CAGEr object to create a referent set of consensus clusters.

Usage

```
aggregateTagClusters(
  object,
  tpmThreshold = 5,
  excludeSignalBelowThreshold = TRUE,
  qLow = NULL,
  qUp = NULL,
  maxDist = 100,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CAGEr'
aggregateTagClusters(
  object,
  tpmThreshold = 5,
  excludeSignalBelowThreshold = TRUE,
  qLow = NULL,
  qUp = NULL,
  maxDist = 100,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

`object`	A `CAGEr` object
`tpmThreshold`	Ignore tag clusters with normalized signal < `tpmThreshold` when constructing the consensus clusters. This threshold is not applied to the CTSSs that contribute to the expression of the consensus clusters (see Details).
`excludeSignalBelowThreshold`	Deprecated and ignored; a warning is emitted when this argument is supplied. The expression of the consensus clusters is computed from all the CTSSs that pass the filter of `filterLowExpCTSS()` (see Details).
`qLow`, `qUp`	Set which "lower" (or "upper") quantile should be used as 5' (or 3') boundary of the tag cluster. If `NULL` the start (for `qLow`) or end (for `qUp`) position of the TC is used.
`maxDist`	Maximal length of the gap (in base-pairs) between two tag clusters for them to be part of the same consensus clusters.
`useMulticore`	Logical, should multicore be used (supported only on Unix-like platforms).
`nrCores`	Number of cores to use when `useMulticore = TRUE`. Default (`NULL`) uses all detected cores.

Details

Since the tag clusters (TCs) returned by the CTSS clustering functions function are constructed separately for every CAGE sample within the CAGEr object, they can differ between samples in both their number, genomic coordinates, position of dominant TSS and overall signal. To be able to compare all samples at the level of clusters of TSSs, TCs from all CAGE datasets are aggregated into a single set of consensus clusters. First, TCs with signal `>= tpmThreshold` from all CAGE datasets are selected, and their 5' and 3' boundaries are determined based on provided `qLow` and `qUp` parameter (or the start and end coordinates, if they are set to `NULL`). Finally, the defined set of TCs from all CAGE datasets is reduced to a non-overlapping set of consensus clusters by merging overlapping TCs and TCs `<= maxDist` base-pairs apart. Consensus clusters represent a referent set of promoters that can be further used for expression profiling or detecting "shifting" (differentially used) promoters between different CAGE samples.

The expression of a consensus cluster in a sample is the sum of the signal in that sample of the CTSSs that it contains, among the CTSSs flagged as passing in `filteredCTSSidx()` at the time `aggregateTagClusters()` is run (this flag is normally set by `filterLowExpCTSS()` before tag clustering). The raw and normalized signals are summed separately to give the `counts` and `normalized` assays. No other filter is applied to the CTSSs: in particular, `tpmThreshold` only selects the tag clusters that define the boundaries of the consensus clusters.

Value

Returns the object in which the *experiment* `consensusClusters` will be occupied by a `RangedSummarizedExperiment` containing the cluster coordinates as row ranges, and their expression levels in the `counts` and `normalized` assays. These genomic ranges are returned by the `consensusClustersGR` function and the whole object can be accessed with the `consensusClustersSE` function. The CTSS ranges of the `tagCountMatrix` *experiment* will gain a `cluster` column indicating which cluster they belong to. Lastly, the number of CTSS outside clusters will be documented in the `outOfClusters` column data.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr object modifiers: `CTSStoGenes()`, `CustomConsensusClusters()`, `annotateCTSS()`, `cumulativeCTSSdistribution()`, `distclu()`, `getCTSS()`, `normalizeTagCount()`, `paraclu()`, `quantilePositions()`, `quickEnhancers()`, `resetCAGEexp()`, `summariseChrExpr()`

Other CAGEr clusters functions: `CTSScumulativesTagClusters()`, `CustomConsensusClusters()`, `consensusClustersDESeq2()`, `consensusClustersGR()`, `cumulativeCTSSdistribution()`, `distclu()`, `paraclu()`, `plotInterquantileWidth()`, `quantilePositions()`, `tagClustersGR()`

Examples

```
consensusClustersGR(exampleCAGEexp)
ce <- aggregateTagClusters(exampleCAGEexp, tpmThreshold = 50, maxDist = 100)
consensusClustersGR(ce)

ce <- aggregateTagClusters( exampleCAGEexp, tpmThreshold = 50, maxDist = 100
                          , qLow = 0.1, qUp = 0.9)
consensusClustersGR(ce)
```

annotateCTSS *Annotate and compute summary statistics*

Description

annotateCTSS annotates the *CTSS* of a CAGEexp object and computes annotation statistics.

annotateConsensusClusters annotates the *consensus clusters* of a CAGEr object.

Usage

```
annotateCTSS(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,GRanges'
annotateCTSS(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,TxDb'
annotateCTSS(object, annot)

annotateTagClusters(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,GRanges'
annotateTagClusters(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,TxDb'
annotateTagClusters(object, annot)

annotateConsensusClusters(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,GRanges'
annotateConsensusClusters(object, annot, upstream = 500, downstream = 500)

## S4 method for signature 'CAGEexp,TxDb'
annotateConsensusClusters(object, annot)
```

Arguments

object	CAGEexp object.
annot	A GRanges or a TxDb object representing the genome annotation. See details for the GRanges object.
upstream	Number of bases *upstream* the start of the transcript models to be considered as part of the *promoter region*.
downstream	Number of bases *downstream* the start of the transcript models to be considered as part of the *promoter region*.

Details

If the annotation is a GRanges, gene names will be extracted from the gene_name metadata, the transcript_type metadata will be used to filter out entries that do not have promoters (such as immunogloblulin VDJ segments), and the type metadata is used to extract positions of introns and exons.

Value

annotateCTSS returns the input object with the following modifications:

- The Genomic Ranges of the tagCountMatrix experiment gains an annotation metadata column, with levels such as promoter, exon, intron and unknown. If the annotation has a gene_name metadata, then a genes column is also added, with gene symbols from the annotation.
- The sample metadata gets new columns, indicating total counts in each of the annotation levels. If the annotation has a gene_name metadata, then a genes column is added to indicate the number of different gene symbols detected.

annotateTagClusters returns the input object with the same modifications as above.

annotateConsensusClusters returns the input object with the same modifications as above.

Author(s)

Charles Plessy

See Also

CTSStoGenes, and the exampleZv9_annot example data.

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), cumulativeCTSSdistribution(), distclu(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Other CAGEr annotation functions: plotAnnot(), ranges2annot(), ranges2genes(), ranges2names()

Examples

```
annotateCTSS(exampleCAGEexp, exampleZv9_annot)
colData(exampleCAGEexp)

exampleCAGEexp <- annotateTagClusters(exampleCAGEexp, exampleZv9_annot)
tagClustersGR(exampleCAGEexp, 1)

annotateConsensusClusters(exampleCAGEexp, exampleZv9_annot)
consensusClustersGR(exampleCAGEexp)
```

---

bam2CTSS *bam2CTSS*

Description

Converts from BAM to CTSS

Usage

```
bam2CTSS(gr, removeFirstG, correctSystematicG, genome)
```

Arguments

gr	A GenomicRanges::GRanges object returned by import.bam().
removeFirstG	See getCTSS().
correctSystematicG	See getCTSS().
genome	See coerceInBSgenome().

Details

Converts genomic ranges representing SAM/BAM alignments into a CTSS object.

Value

Returns a CTSS object.

See Also

Other loadFileIntoGPos: import.CTSS(), import.bam(), import.bam.ctss(), import.bedCTSS(), import.bedScore(), import.bedmolecule(), import.bigwig(), loadFileIntoGPos(), moleculesGR2CTSS()

CAGEexp-class *CAGEr class to hold all data and metadata about one CAGE experiment.*

Description

The CAGEr class is a MultiAssayExperiment object containing all data and metadata about a set of CAGE libraries. It replaced the CAGEset class in 2017. The main difference is that the expression data is stored in DataFrame objects of Rle-encoded expression values, instead of plain data.frames. With large datasets, this saves considerable amounts of memory.

Details

If genomeName is NULL, checks of chromosome names will be disabled and G-correction will not be possible. See https://support.bioconductor.org/p/86437/ for an example on how to create a *BSgenome* package.

Sample labels must be *syntactically valid* in the sense of the make.names() function, because they will be used as column names in some tables.

Slots

metadata A list that must at least contain a genomeName member.

See Also

make.names

Examples

```
pathsToInputFiles <- list.files( system.file("extdata", package = "CAGEr")
                               , "ctss$"
                               , full.names = TRUE)
sampleLabels <- sub( ".chr17.ctss", "", basename(pathsToInputFiles))

# The CAGEexp object can be created using specific constructor commands

exampleCAGEexp <-
  CAGEexp( genomeName     = "BSgenome.Drerio.UCSC.danRer7"
         , inputFiles     = pathsToInputFiles
         , inputFilesType = "ctss"
         , sampleLabels   = sub( ".chr17.ctss", "", basename(pathsToInputFiles)))

# Alternatively, it can be created just like another MultiAssayExperiment.
# This is useful when providing pre-existing colData with many columns.

exampleCAGEexp <-
  CAGEexp( metadata = list(genomeName = "BSgenome.Drerio.UCSC.danRer7")
         , colData  = DataFrame( inputFiles     = pathsToInputFiles
                               , sampleLabels   = sampleLabels
                               , inputFilesType = "ctss"
                               , row.names      = sampleLabels))


# Expression data is loaded by the getCTSS() function, that also calculates
# library sizes and store them in the object's column data.

exampleCAGEexp <- getCTSS(exampleCAGEexp)
librarySizes(exampleCAGEexp)
colData(exampleCAGEexp)

# CTSS data is stored internally as a SummarizedExperiemnt that can be retreived
# as a whole, or as GRanges, or as an expression DataFrame.

CTSStagCountSE(exampleCAGEexp)
CTSScoordinatesGR(exampleCAGEexp)
CTSStagCountDF(exampleCAGEexp)

# Columns of the "colData" table are accessible directly via the "$" operator.

exampleCAGEexp$l1 <- CTSStagCountDF(exampleCAGEexp) |> sapply ( \(col) sum(col > 0) )
exampleCAGEexp$l1
```

`CAGEr-class` *CAGEr objects*

Description

The *CAGEr* package provides one class of objects to load, contain and process CAGE data: the `CAGEexp` class, introduced 2017, which is based on the `MultiAssayExperiment` class. In comparison with the original `CAGEset` class (removed in 2021) `CAGEexp` objects benefit from a a more efficient data storage, using `DataFrames` of run-length-encoded (`Rle`) integers, allowing for the loading and use of much larger transcriptome datasets.

References

Haberle V, Forrest ARR, Hayashizaki Y, Carninci P and Lenhard B (**2015**). "CAGEr: precise TSS data retrieval and high-resolution promoterome mining for integrative analyses." *Nucleic Acids Research*, 43, pp. e51., http://nar.oxfordjournals.org/content/43/8/e51

`CAGEr_Multicore` *Multicore support in CAGEr*

Description

CAGEr is in the transition towards using the BiocParallel for multicore parallelisation. On Windows platforms, the multicore support is disabled transparently, that is, attempts to use multiple cores are silently ignored.

Usage

```
CAGEr_Multicore(useMulticore = FALSE, nrCores = NULL)
```

Arguments

`useMulticore`	TRUE or FALSE
`nrCores`	number of cores to use (leave `NULL` to let BiocParallel choose).

Value

Returns either a `MulticoreParam` object or a `SerialParam` object.

Author(s)

Charles Plessy

Examples

```
CAGEr:::CAGEr_Multicore()
CAGEr:::CAGEr_Multicore(TRUE,)
CAGEr:::CAGEr_Multicore(TRUE,  2)
CAGEr:::CAGEr_Multicore(FALSE, 2)
```

`coerceInBSgenome` *coerceInBSgenome*

Description

A private (non-exported) function to discard any range that is not compatible with the CAGEr object's BSgenome.

Usage

```
coerceInBSgenome(gr, genome)
```

Arguments

`gr`	The genomic ranges to coerce.
`genome`	The name of a BSgenome package, which must me installed, or `NULL` to skip coercion.

Value

A GRanges object in which every range is guaranteed to be compatible with the given BSgenome object. The sequnames of the GRanges are also set accordingly to the BSgenome.

`ConsensusClusters-class` *ConsensusClusters*

Description

The `ConsensusClusters` class represents consensus clusters. It is used internally by CAGEr for type safety.

Details

Consensus clusters must not overlap, so that a single TSS in the genome can only be attributed to a single cluster.

consensusClusters<- *Set consensus clusters from CAGEr objects*

Description

Set the information on consensus clusters in a CAGEr object.

Usage

```
consensusClustersSE(object) <- value

## S4 replacement method for signature 'CAGEexp,RangedSummarizedExperiment'
consensusClustersSE(object) <- value

consensusClustersGR(object) <- value

## S4 replacement method for signature 'CAGEexp'
consensusClustersGR(object) <- value
```

Arguments

object	A CAGEr object.
value	A data.frame of consensus clusters

Details

These setter methods are mostly for internal use, but are exported in case they may be useful to advanced users.

Author(s)

Vanja Haberle

Charles Plessy

consensusClustersDESeq2 *Export* consensus cluster *expression data for DESeq2 analysis*

Description

Creates a DESeqDataSet using the consensus cluster expression data in the experiment slot consensusClusters and the sample metadata of the CAGEexp object. The formula must be built using factors already present in the sample metadata.

Usage

```
consensusClustersDESeq2(object, design)

## S4 method for signature 'CAGEexp'
consensusClustersDESeq2(object, design)
```

Arguments

object	A CAGEexp object.
design	A formula for the DESeq2 analysis.

Author(s)

Charles Plessy

See Also

DESeqDataSet in the DESeq2 package.

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersGR(), cumulativeCTSSdistribution(), distclu(), paraclu(), plotInterquantileWidth(), quantilePositions(), tagClustersGR()

Examples

```
exampleCAGEexp$group <- c("a", "a", "b", "b", "a")
consensusClustersDESeq2(exampleCAGEexp, ~group)
```

---

consensusClustersGR *Get consensus clusters from CAGEr objects*

Description

Extracts the information on consensus clusters from a CAGEr object.

Usage

```
consensusClustersGR(object, sample = NULL, qLow = NULL, qUp = NULL)

## S4 method for signature 'CAGEexp'
consensusClustersGR(object, sample = NULL, qLow = NULL, qUp = NULL)

consensusClustersSE(object)

## S4 method for signature 'CAGEexp'
consensusClustersSE(object)
```

Arguments

object	A CAGEr object.
sample	Optional. Label of the CAGE dataset (experiment, sample) for which to extract sample-specific information on consensus clusters.
qLow, qUp	Lower and upper quantiles to compute interquantile width.

Value

consensusClustersGR returns a ConsensusClusters object, which wraps the GRanges class. The score columns indicates the normalised expression value of each cluster, either across all samples (sample = NULL), or for the selected sample. The legacy tpm column may be removed in the future. When sample argument is NOT specified, total CAGE signal across all CAGE datasets (samples) is returned in the tpm column. When sample argument is specified, the tpm column contains CAGE signal of consensus clusters in that specific sample. In addition, sample-specific information is returned, including position of the dominant TSS, and (if applicable) interquantile width of the consensus clusters in the specified sample or otherwise, sample-agnostic information is returned.

consensusClustersSE returns the SummarizedExperiment stored in the consensusClusters experiment slot of the CAGEexp object.

Author(s)

Vanja Haberle

Charles Plessy

See Also

consensusClusters<-()

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), cumulativeCTSSdistribution(), distclu(), paraclu(), plotInterquantileWidth(), quantilePositions(), tagClustersGR()

Examples

```
consensusClustersGR( exampleCAGEexp, sample = 2
                   , qLow = 0.1, qUp = 0.9)
```

---

consensusClustersQuantile

Quantile metadata stored in CAGEr objects.

Description

Accessors for consensus cluster quantile data in CAGEr objects.

Usage

```
consensusClustersQuantileLow(object, samples = NULL)

## S4 method for signature 'CAGEexp'
consensusClustersQuantileLow(object, samples = NULL)

consensusClustersQuantileUp(object, samples = NULL)
```

```
## S4 method for signature 'CAGEexp'
consensusClustersQuantileUp(object, samples = NULL)

consensusClustersQuantile(object, sample = NULL, q)

## S4 method for signature 'CAGEexp'
consensusClustersQuantile(object, sample = NULL, q)

consensusClustersQuantileLow(object, samples = NULL) <- value

consensusClustersQuantileUp(object, samples = NULL) <- value
```

Arguments

object	A CAGEr object.
samples	Sample name(s), number(s) or NULL (default) for all samples.
sample	A single sample name or number, or NULL (default) for all samples.
q	A quantile.
value	A list (one entry per sample) of data frames with multiple columns: cluster for the cluster ID, and then q_0.n where 0.n indicates a quantile.

---

consensusClustersTpm *Extracting consensus clusters tpm matrix from CAGEr object*

Description

Extracts a table with normalized CAGE tag values for consensus clusters across all samples from a CAGEr object.

Usage

```
consensusClustersTpm(object)

## S4 method for signature 'CAGEexp'
consensusClustersTpm(object)
```

Arguments

object	A CAGEr object.

Value

Returns the matrix of normalized expression values of CAGE clusters across all samples.

Author(s)

Vanja Haberle

See Also

consensusClustersSE

Other CAGEr clustering methods: distclu(), paraclu()

Examples

```
head(consensusClustersTpm(exampleCAGEexp))
```

CTSS-class *CAGE Transcription Start Sites*

Description

The CTSS class represents CAGE transcription start sites (CTSS) at single-nucleotide resolution, using GenomicRanges::UnstitchedGPos as base class. It is used by *CAGEr* for type safety.

The CTSS constructor takes the same arguments as GenomicRanges::GPos, plus bsgenomeName, and minus stitch, which is hardcoded to FALSE.

Usage

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

## S4 method for signature 'CTSS'
initialize(.Object, ..., bsgenomeName = NULL)

CTSS(
  seqnames = NULL,
  pos = NULL,
  strand = NULL,
  ...,
  seqinfo = NULL,
  seqlengths = NULL,
  bsgenomeName = NULL
)

## S4 method for signature 'CTSS,GRanges'
coerce(from, to = "GRanges", strict = TRUE)

## S4 method for signature 'GRanges,CTSS'
coerce(from, to = "CTSS", strict = TRUE)
```

Arguments

object	See methods::show
.Object	See methods::new
bsgenomeName	String containing the name of a *BSgenome* package.
seqnames, pos, strand, seqinfo, seqlengths, ...	See the documentation of GenomicRanges::GPos for further details.
from, to, strict	See methods::coerce.

Details

The genomeName element of the metadata slot is used to store the name of the *BSgenome* package used when constructing the CAGEr object.

Coercion from GRanges to CTSS loses information, but it seems to be fine, since other coercions like as(1.2, "integer") do the same.

Author(s)

Charles Plessy

Examples

```
# Convert an UnstitchedGPos object using the new() constructor.
gp <- GPos("chr1:2:-", stitch = FALSE)
ctss <- new("CTSS", gp, bsgenomeName = "BSgenome.Drerio.UCSC.danRer7")
genomeName(ctss)

# Create a new object using the CTSS() constructor.
CTSS("chr1", 2, "-", bsgenomeName = "BSgenome.Drerio.UCSC.danRer7")

# Coerce CTSS to GRanges
as(ctss, "GRanges")

# Coerce a GRanges object to CTSS using the as() method.
gr <- GRanges("chr1:1-10:-")
gr$seq <- "AAAAAAAAAA"
seqlengths(gr) <- 100
genome(gr) <- "foo"
as(gr, "CTSS")
identical(seqinfo(gr), seqinfo(as(gr, "CTSS")))
as(as(gr, "CTSS"), "CTSS") # Make sure it works twice in a row
```

CTSScoordinatesGR *Genomic coordinates of TSSs from a CAGEr object*

Description

Extracts the genomic coordinates of all detected TSSs from CAGEexp objects.

Usage

```
CTSScoordinatesGR(object)

## S4 method for signature 'CAGEexp'
CTSScoordinatesGR(object)

CTSScoordinatesGR(object) <- value

## S4 replacement method for signature 'CAGEexp'
CTSScoordinatesGR(object) <- value

CTSStagCountSE(object) <- value
```

```
## S4 replacement method for signature 'CAGEexp'
CTSStagCountSE(object) <- value
```

Arguments

object	A CAGEexp object.
value	Coordinates to update, in a format according to the function name.

Value

CTSScoordinatesGR returns the coordinates as a CTSS() object wrapping genomic ranges. A filteredCTSSidx column metadata will be present if filterLowExpCTSS was ran earlier.

Author(s)

Vanja Haberle

Charles Plessy

See Also

getCTSS

Other CAGEr accessor methods: CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
CTSScoordinatesGR(exampleCAGEexp)

CTSScoordinatesGR(exampleCAGEexp)
```

CTSScumulativesTagClusters

Get/set CTSS cumulative TC or CC data

Description

Accessor function.

Usage

```
CTSScumulativesTagClusters(object, samples = NULL)

## S4 method for signature 'CAGEexp'
CTSScumulativesTagClusters(object, samples = NULL)

CTSScumulativesCC(object, samples = NULL)
```

```
## S4 method for signature 'CAGEexp'
CTSScumulativesCC(object, samples = NULL)

CTSScumulativesTagClusters(object) <- value

## S4 replacement method for signature 'CAGEexp'
CTSScumulativesTagClusters(object) <- value
```

Arguments

object	A CAGEexp object.
samples	One or more valid sample names.
value	CTSScumulativesTagClusters data

Value

List of numeric Rle.

See Also

Other CAGEr clusters functions: CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESe
consensusClustersGR(), cumulativeCTSSdistribution(), distclu(), paraclu(), plotInterquantileWidth(),
quantilePositions(), tagClustersGR()

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSSnormalizedTpmDF(), CTSStagCountDF(),
GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(),
genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(),
tagClustersGR()

CTSSnormalizedTpmDF *Extracting normalized CAGE signal for TSSs from CAGEr objects*

Description

Extracts the normalized CAGE signal for all detected TSSs in all CAGE datasets from CAGEexp objects.

Usage

```
CTSSnormalizedTpmDF(object)

## S4 method for signature 'CAGEexp'
CTSSnormalizedTpmDF(object)

CTSSnormalizedTpmGR(object, samples)

## S4 method for signature 'CAGEexp'
CTSSnormalizedTpmGR(object, samples)
```

Arguments

object	A CAGEexp object.
samples	The name of sample(s) as reported by sampleLabels(object), or the number identifying the sample(s).

Value

CTSSnormalizedTpmDF returns a DataFrame of normalised expression values.

Author(s)

Vanja Haberle

Charles Plessy

See Also

normalizeTagCount

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
CTSSnormalizedTpmDF(exampleCAGEexp)

CTSSnormalizedTpmGR(exampleCAGEexp, 1)
exampleCAGEexp |> CTSSnormalizedTpmGR("all")
```

CTSStagCountDF *Raw CAGE TSSs expression counts*

Description

Extracts the tag count for all detected TSSs in all CAGE datasets from CAGEexp objects.

Usage

```
CTSStagCountDF(object)

## S4 method for signature 'CAGEexp'
CTSStagCountDF(object)

CTSStagCountGR(object, samples)

## S4 method for signature 'CAGEexp'
CTSStagCountGR(object, samples)

CTSStagCountSE(object)

## S4 method for signature 'CAGEexp'
CTSStagCountSE(object)
```

Arguments

object	A CAGEexp object.
samples	For CTSStagCountGR only: name(s) or number(s) identifying sample(s) or "all" to return a GRangesList of all the samples.

Value

Returns an object with number of CAGE tags supporting each TSS (rows) in every CAGE dataset (columns). The class of the object depends on the function being called:

- CTSStagCountDF: A DataFrame of Rle integers.
- CTSStagCountSE: A RangedSummarizedExperimentcontaining aDataFrameofRle‘ integers.
- CTSStagCountGR: A CTSS object (wrapping GRanges) containing a score column indicating expression values for a given sample, or a GRangesList of CTSS objects.

Author(s)

Vanja Haberle

Charles Plessy

See Also

getCTSS()

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
CTSStagCountDF(exampleCAGEexp)

CTSStagCountGR(exampleCAGEexp, 1)
CTSStagCountGR(exampleCAGEexp, "all")

CTSStagCountSE(exampleCAGEexp)
```

CTSStoGenes *Make a gene expression table.*

Description

Add a gene expression table in the GeneExpSE experiment slot of an annotated CAGEexp object.

Usage

```
CTSStoGenes(object)

## S4 method for signature 'CAGEexp'
CTSStoGenes(object)
```

Arguments

object	A CAGEexp object that was annotated with the annotateCTSS() function.

Value

The input object with the following modifications:

- A new geneExpMatrix experiment containing gene expression levels as a SummarizedExperiment object with one assay called counts, which is plain matrix of integers. (This plays better than Rle DataFrames when interfacing with downstream packages like DESeq2, and since the number of genes is limited, a matrix will not cause problems of performance.)
- New genes column data added, indicating total number of gene symbols detected per library.
- New unannotated column data added, indicating for each sample the number of counts that did not overlap with a known gene.

Author(s)

Charles Plessy

See Also

annotateCTSS().

Other CAGEr object modifiers: CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), distclu(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Other CAGEr gene expression analysis functions: GeneExpDESeq2(), ranges2genes()

Examples

```
CTSStoGenes(exampleCAGEexp)
all( librarySizes(exampleCAGEexp) -
     colSums(SummarizedExperiment::assay(GeneExpSE(exampleCAGEexp))) ==
     exampleCAGEexp$unannotated)
```

---

cumulativeCTSSdistribution

Cumulative sums of CAGE counts along genomic regions

Description

Calculates the cumulative sum of normalised CAGE counts along each tag cluster or consensus cluster in every sample within a CAGEr object.

Usage

```
cumulativeCTSSdistribution(
  object,
  clusters = c("tagClusters", "consensusClusters"),
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CAGEexp'
cumulativeCTSSdistribution(
  object,
  clusters = c("tagClusters", "consensusClusters"),
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

object	A CAGEr object
clusters	tagClusters or consensusClusters.
useMulticore	Logical, should multicore be used. useMulticore = TRUE has no effect on non-Unix-like platforms.
nrCores	Number of cores to use when useMulticore = TRUE (set to NULL to use all detected cores).

Value

In CAGEexp objects, cumulative sums for the *tag clusters* are stored in the metadata slot using the RleList class. For *consensus clusters*, they are stored in *assays* of the consensusClusters experiment slot of the CAGEexp object.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), distclu(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), consensusClustersGR(), distclu(), paraclu(), plotInterquantileWidth(), quantilePositions(), tagClustersGR()

Examples

```
cumulativeCTSSdistribution(exampleCAGEexp, clusters = "tagClusters")
CTSScumulativesTagClusters(exampleCAGEexp)[[1]][1:6]
cumulativeCTSSdistribution(exampleCAGEexp, clusters = "consensusClusters")
CTSScumulativesCC(exampleCAGEexp)[[1]][1:6]
```

CustomConsensusClusters *Expression levels of consensus cluster*

Description

Intersects custom consensus clusters with the CTSS data in a CAGEexp object, and stores the result as a expression matrices (raw and normalised tag counts).

Usage

```
CustomConsensusClusters(
  object,
  clusters,
  threshold = 0,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'CAGEexp,GRanges'
CustomConsensusClusters(
  object,
  clusters,
  threshold = 0,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)
```

Arguments

object	A CAGEexp object
clusters	Consensus clusters in GRanges format.
threshold, nrPassThreshold	Only CTSSs with signal >= threshold in >= nrPassThreshold experiments will be used for clustering and will contribute towards total signal of the cluster.
thresholdIsTpm	Logical, is threshold raw tag count value (FALSE) or normalized signal (TRUE).

Details

Consensus clusters must not overlap, so that a single base of the genome can only be attributed to a single cluster. This is enforced by the .ConsensusClusters constructor.

Value

stores the result as a new RangedSummarizedExperiment in the experiment slot of the object. The assays of the new experiment are called counts and normalized. An outOfClusters column is added to the sample metadata to reflect the number of molecules that do not have their TSS in a consensus cluster.

Author(s)

Charles Plessy

See Also

Other CAGEr object modifiers: `CTSStoGenes()`, `aggregateTagClusters()`, `annotateCTSS()`, `cumulativeCTSSdistribution()`, `distclu()`, `getCTSS()`, `normalizeTagCount()`, `paraclu()`, `quantilePositions()`, `quickEnhancers()`, `resetCAGEexp()`, `summariseChrExpr()`

Other CAGEr clusters functions: `CTSScumulativesTagClusters()`, `aggregateTagClusters()`, `consensusClustersDESeq2()`, `consensusClustersGR()`, `cumulativeCTSSdistribution()`, `distclu()`, `paraclu()`, `plotInterquantileWidth()`, `quantilePositions()`, `tagClustersGR()`

Examples

```
cc <- consensusClustersGR(exampleCAGEexp)
CustomConsensusClusters(exampleCAGEexp, cc)
```

---

distclu *Distance clustering*

Description

The "distclu" method is an implementation of simple distance-based clustering of data attached to sequences, where two neighbouring TSSs are joined together if they are closer than some specified distance (see `GenomicRanges::reduce` for implementation details.

Usage

```
distclu(object, maxDist = 20, keepSingletonsAbove = 0)

## S4 method for signature 'SummarizedExperiment'
distclu(object, maxDist = 20, keepSingletonsAbove = 0)

## S4 method for signature 'CTSS'
distclu(object, maxDist = 20, keepSingletonsAbove = 0)

## S4 method for signature 'CAGEexp'
distclu(object, maxDist = 20, keepSingletonsAbove = 0)
```

Arguments

`object`	The `SummarizedExperiment::RangedSummarizedExperiment` object containing CTSS information, or just a `CTSS` object.
`maxDist`	Maximal distance between two neighbouring CTSSs for them to be part of the same cluster.
`keepSingletonsAbove`	Remove "singleton" tag clusters of width 1 with signal < `keepSingletonsAbove`. Default value `0` results in keeping all TCs by default. Setting it to `Inf` removes all singletons.

Details

Clustering is done for every CAGE dataset within the CAGEr object separately, resulting in a different set of tag clusters for every CAGE dataset. TCs from different datasets can further be aggregated into a single referent set of consensus clusters by calling the aggregateTagClusters function.

Value

For CTSS input, a TagClusters object, for SummarizedExperiment input, a GRangesList of TagClusters objects, and for CAGEexp input, a modified object containing the tag clusters stored as a GRangesList of TagClusters objects in its metadata slot tagClusters.

Author(s)

Vanja Haberle

Charles Plessy

See Also

aggregateTagClusters

Other CAGEr clustering methods: consensusClustersTpm(), paraclu()

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), consensusClustersGR(), cumulativeCTSSdistributio paraclu(), plotInterquantileWidth(), quantilePositions(), tagClustersGR()

Examples

```
distclu(CTSSnormalizedTpmGR(exampleCAGEexp, 1)[1:10])
distclu(CTSStagCountSE(exampleCAGEexp)[1:25,])
ce <- distclu(exampleCAGEexp, maxDist = 20, keepSingletonsAbove = 100)
tagClustersGR(ce, "Zf.30p.dome")
```

---

exampleCAGEexp *Example CAGEexp object.*

Description

Lazy-loaded example CAGEexp object, containing most of the CAGEr data structures created with the CAGEr modifier functions.

Usage

```
exampleCAGEexp
```

Format

A CAGEexp object.

Examples

```
## Not run:
pathsToInputFiles <- list.files( system.file("extdata", package = "CAGEr")
                                 , "ctss$"
                                 , full.names = TRUE)
sampleLabels <- sub( ".chr17.ctss", "", basename(pathsToInputFiles))
exampleCAGEexp <-
  CAGEexp( genomeName     = "BSgenome.Drerio.UCSC.danRer7"
         , inputFiles     = pathsToInputFiles
         , inputFilesType = "ctss"
         , sampleLabels   = sub( ".chr17.ctss", "", basename(pathsToInputFiles)))
exampleCAGEexp <- getCTSS(exampleCAGEexp)
librarySizes(exampleCAGEexp)
colData(exampleCAGEexp)
exampleCAGEexp$l1 <- NULL
exampleCAGEexp <- exampleCAGEexp[,c(5, 2, 1, 3, 4)] # Non-aplhabetic order may help catch bugs
CTSStagCountSE(exampleCAGEexp) <- CTSStagCountSE(exampleCAGEexp)[1:5000,] # Slim the object
exampleCAGEexp$librarySizes <- sapply(CTSStagCountDF(exampleCAGEexp), sum) # Repair metadata
exampleCAGEexp <-
  summariseChrExpr(exampleCAGEexp)               |>
  annotateCTSS(exampleZv9_annot)                 |>
  CTSStoGenes()                                  |>
  normalizeTagCount()                            |>
  getExpressionProfiles("CTSS")                  |>
  filterLowExpCTSS()                             |>
  distclu()                                      |>
  annotateTagClusters(exampleZv9_annot)          |>
  cumulativeCTSSdistribution("tagClusters")      |>
  quantilePositions("tagClusters")               |>
  aggregateTagClusters()                         |>
  annotateConsensusClusters(exampleZv9_annot)    |>
  cumulativeCTSSdistribution("consensusClusters") |>
  quantilePositions("consensusClusters")         |>
  getExpressionProfiles("consensusClusters")     |>
  scoreShift( groupX = c("Zf.unfertilized.egg")
            , groupY = "Zf.30p.dome"
            , testKS = TRUE, useTpmKS = FALSE)
save(exampleCAGEexp, file = "data/exampleCAGEexp.RData", compress = "xz")

## End(Not run)
```

`exampleZv9_annot` *Example zebrafish annotation data*

Description

Annotation data for zebrafish's chromosome 17's interval 26000000-54000000 (Zv9/danRer7 genome), to be used in documentation examples.

Usage

```
exampleZv9_annot
```

Format

An object of class GRanges of length 7467.

Details

Data was retreived from ENSEMBL's Biomart server using a query to extract gene, transcripts and exon coordinates. For the record, here it is as URL (long, possibly overflowing).

http://mar2015.archive.ensembl.org/biomart/martview/78d86c1d6b4ef51568ba6d46f7d8b254?VIRTUALSCHEMANA

And here it is as XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Query>
<Query  virtualSchemaName = "default" formatter = "TSV" header = "0" uniqueRows = "0" count = "" datas
  <Dataset name = "drerio_gene_ensembl" interface = "default" >
    <Attribute name = "ensembl_gene_id" />
    <Attribute name = "ensembl_transcript_id" />
    <Attribute name = "start_position" />
    <Attribute name = "end_position" />
    <Attribute name = "transcript_start" />
    <Attribute name = "transcript_end" />
    <Attribute name = "strand" />
    <Attribute name = "chromosome_name" />
    <Attribute name = "external_gene_name" />
    <Attribute name = "gene_biotype" />
    <Attribute name = "exon_chrom_start" />
    <Attribute name = "exon_chrom_end" />
    <Attribute name = "is_constitutive" />
    <Attribute name = "rank" />
  </Dataset>
</Query>
```

The downloaded file was then transformed as follows.

```
x <- read.delim("~/Downloads/mart_export.txt", stringsAsFactors = FALSE)
e <- GRanges(paste0("chr", x$Chromosome.Name), IRanges(x$Exon.Chr.Start..bp., x$Exon.Chr.End..bp.)
e$gene_name <- Rle(x$Associated.Gene.Name)
e$transcript_type <- Rle(x$Gene.type)
e$type <- "exon"
e$type <- Rle(e$type)

e <- GRanges(paste0("chr", x$Chromosome.Name), IRanges(x$Exon.Chr.Start..bp., x$Exon.Chr.End..bp.)
e$gene_name <- Rle(x$Associated.Gene.Name)
e$transcript_type <- Rle(x$Gene.type)
e$type <- "exon"
e$type <- Rle(e$type)
e <- sort(unique(e))

g <- GRanges( paste0("chr", x$Chromosome.Name)
            , IRanges(x$Gene.Start..bp., x$Gene.End..bp.)
            , ifelse( x$Strand + 1, "+", "-"))
```

```
g$gene_name <- Rle(x$Associated.Gene.Name)
g$transcript_type <- Rle(x$Gene.type)
g$type <- "gene"
g$type <- Rle(g$type)
g <- sort(unique(g))

t <- GRanges( paste0("chr", x$Chromosome.Name)
            , IRanges(x$Transcript.Start..bp., x$Transcript.End..bp.)
            , ifelse( x$Strand + 1, "+", "-"))

t$gene_name <- Rle(x$Associated.Gene.Name)
t$transcript_type <- Rle(x$Gene.type)
t$type <- "transcript"
t$type <- Rle(t$type)
t <- sort(unique(t))

gff <- sort(c(g, t, e))
gff <- gff[seqnames(gff) == "chr17"]
gff <- gff[start(gff) > 26000000 & end(gff) < 54000000]
seqlevels(gff) <- seqlevelsInUse(gff)

save(gff, "data/exampleZv9_annot.RData", compress = "xz")
```

Author(s)

Prepared by Charles Plessy <plessy@riken.jp> using archive ENSEMBL data.

References

http://mar2015.archive.ensembl.org/biomart/

`exportToTrack` *Converts TSSs and clusters of TSSs to a genome browser track format*

Description

Converts *CTSS*, *tag clusters* or *consensus clusters* to the `UCSCData` format of the `rtracklayer` package, that can be exported to BED file(s) with track information for genome browsers. *CTSSes* and *consensus clusters* are optionally colored by their expression class. *Tag clusters* and *consensus clusters* can be displayed in a whiskerplot-like representation with a line showing full span on the cluster, filled block showing interquantile range and a thick box denoting position of the dominant (most frequently) used *TSS*.

Usage

```
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
```

```
  oneTrack = TRUE
)

## S4 method for signature 'CAGEexp'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)

## S4 method for signature 'GRangesList'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)

## S4 method for signature 'GRanges'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)

## S4 method for signature 'CTSS'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)

## S4 method for signature 'TagClusters'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)
```

```
## S4 method for signature 'ConsensusClusters'
exportToTrack(
  object,
  what = c("CTSS", "tagClusters", "consensusClusters"),
  qLow = NULL,
  qUp = NULL,
  colorByExpressionProfile = FALSE,
  oneTrack = TRUE
)
```

Arguments

object	A CAGEexp object.
what	Which elements should be exported: CTSS for individual *CTSSs*, tagClusters for *tag clusters* or consensusClusters for *consensus clusters*.
qLow, qUp	Position of which "lower" (resp. "upper") quantile should be used as 5' (resp. 3') boundary of the filled block in whiskerplot-like representation of the cluster. Default: NULL (plain line representation). Ignored when what = "CTSS".
colorByExpressionProfile	Logical, should blocks be colored in the color of their corresponding expression class. Ignored when what equals"tagClusters".
oneTrack	Logical, should the data be converted in an individual object or a list of objects?

Details

The BED representations of *CTSSs*, *tag cluster* and *consensus clusters* can be directly visualised in the ZENBU or UCSC Genome Browsers.

When what = "CTSS", one UCSCData object with single track of 1 bp blocks representing all detected CTSSs (in all CAGE samples) is created. CTSSs can be colored according to their expression class (see getExpressionProfiles and plotExpressionProfiles). For colorByExpressionProfile = FALSE, CTSSs included in the clusters are shown in black and CTSSs that were filtered out in gray.

When what = "tagClusters", one track per CAGE dataset is created, which can be exported to a single UCSCData object (by setting oneFile = TRUE) or separate ones (FALSE). If no quantile boundaries were provided (qLow and qUp are NULL, TCs are represented as simple blocks showing the full span of TC fromthe start to the end. Setting qLow and/or qUp parameters to a value of the desired quantile creates a gene-like representation with a line showing full span of the TC, filled block showing specified interquantile range and a thick 1 bp block denoting position of the dominant (most frequently used) TSS. All TCs in one track (one CAGE dataset) are shown in the same color.

When what = "consensusClusters" *consensus clusters* are exported. Since there is only one set of consensus clusters common to all CAGE datasets, only one track is created in case of a simple representation. This means that when qLow = NULL and qUp = NULL one track with blocks showing the full span of consensus cluster from the start to the end is created. However, the distribution of the CAGE signal within consensus cluster can be different in different CAGE samples, resulting in different positions of quantiles and dominant TSS. Thus, when qLow and/or qUp parameters are set to a value of the desired quantile, a separate track with a gene-like representation is created for every CAGE dataset. These tracks can be exported to a single UCSCData object (by setting oneFile = TRUE) or separate ones (by setting oneFile = FALSE). The gene-like representation is analogous to the one described above for the TCs. In all cases consensus clusters can be colored according to their expression class (provided the expression profiling of consensus clusters was

done by calling `getExpressionProfiles` function). Colors of expression classes match the colors in which they are shown in the plot returned by the `plotExpressionProfiles` function. For `colorByExpressionProfile = FALSE` all consensus clusters are shown in black.

Value

Returns either a `rtracklayer` `UCSCData` object, or a `GRangesList` of them.

Author(s)

Vanja Haberle

Charles Plessy

Examples

```
# You can export from a CAGEexp object or from a cluster object directly:
exportToTrack(exampleCAGEexp, what = "CTSS")  # Is same as:
exportToTrack(CTSScoordinatesGR(exampleCAGEexp))  # Or:
exampleCAGEexp |> CTSScoordinatesGR() |> exportToTrack()

# Export a single sample,
exampleCAGEexp |> CTSStagCountGR(2)      |> exportToTrack()
exampleCAGEexp |> CTSSnormalizedTpmGR(2) |> exportToTrack()

# Exporting multiple samples results in a GRangesList of UCSCData objects.
exportToTrack(exampleCAGEexp, what = "CTSS", oneTrack = FALSE)
exampleCAGEexp |> CTSStagCountGR("all")  |> exportToTrack()
exampleCAGEexp |> CTSSnormalizedTpmGR("all")  |> exportToTrack()

### exporting CTSSs colored by expression class
# Temporarly disabled
# exportToTrack(exampleCAGEexp, what = "CTSS", colorByExpressionProfile = TRUE)

### exporting tag clusters in gene-like representation
exportToTrack(exampleCAGEexp, what = "tagClusters", qLow = 0.1, qUp = 0.9)
tagClustersGR(exampleCAGEexp, 1) |> exportToTrack(qLow = 0.1, qUp = 0.9)

### exporting consensus clusters
exportToTrack( exampleCAGEexp, what = "consensusClusters")
exampleCAGEexp |>
  consensusClustersGR("Zf.high", qLow = .1, qUp = .9) |>
  exportToTrack(qLow = .1, qUp = .9)
exportToTrack( exampleCAGEexp, what = "consensusClusters"
             , qLow = 0.1, qUp = 0.9, oneTrack = FALSE)
```

`expressionClasses` *Extract labels of* expression classes

Description

Retrieves labels of *expression classes* of individual CTSSs or consensus clusters from a `CAGEr` object.

Usage

```
expressionClasses(object)

## S4 method for signature 'CTSS'
expressionClasses(object)

## S4 method for signature 'ConsensusClusters'
expressionClasses(object)
```

Arguments

object	A CAGEr object.

Value

Returns a Rle-encoded vector of labels of *expression classes*. The number of labels matches the number of expression clusters returned by getExpressionProfiles function.

See Also

Other CAGEr expression clustering functions: getExpressionProfiles(), plotExpressionProfiles()

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
expressionClasses(CTSScoordinatesGR(exampleCAGEexp))
exampleCAGEexp |> consensusClustersGR() |> expressionClasses()
```

FANTOM5humanSamples *FANTOM5 human samples*

Description

Lazy-loaded data.frame object, containing information about FANTOM5 libraries. Its use is described in more details in the vignette "Use of CAGE resources with CAGEr".

Usage

```
FANTOM5humanSamples
```

Format

A data.frame with sample, type, description, library_id and data_url columns.

See Also

Other FANTOM data: FANTOM5mouseSamples, importPublicData()

FANTOM5mouseSamples *FANTOM5 mouse samples*

Description

Lazy-loaded `data.frame` object, containing information about FANTOM5 libraries. Its use is described in more details in the vignette "Use of CAGE resources with CAGEr".

Usage

```
FANTOM5mouseSamples
```

Format

A `data.frame` with `sample`, `type`, `description`, `library_id` and `data_url` columns.

See Also

Other FANTOM data: `FANTOM5humanSamples`, `importPublicData()`

filteredCTSSidx *The* `filteredCTSSidx()` *function is in* CAGEr *functions to retrieve the result of the* `flagLowExpCTSS()` *function in a safe way.*

Description

The `filteredCTSSidx()` function is in *CAGEr* functions to retrieve the result of the `flagLowExpCTSS()` function in a safe way.

Usage

```
filteredCTSSidx(object)

## S4 method for signature 'CAGEexp'
filteredCTSSidx(object)
```

Arguments

`object`	A `CAGEexp` object

Value

Returns the value of `filteredCTSSidx` in the row ranges of the tag count matrix experiment of the `CAGEexp` object, or `Rle(TRUE)` if it was `NULL`

See Also

Other CAGEr filter functions: `flagByUpstreamSequences()`, `flagLowExpCTSS()`

Other CAGEr accessor methods: `CTSScoordinatesGR()`, `CTSScumulativesTagClusters()`, `CTSSnormalizedTpmDF()`, `CTSStagCountDF()`, `GeneExpDESeq2()`, `GeneExpSE()`, `consensusClustersGR()`, `expressionClasses()`, `genomeName()`, `inputFiles()`, `inputFilesType()`, `librarySizes()`, `sampleLabels()`, `seqNameTotalsSE()`, `tagClustersGR()`

Examples

```
filteredCTSSidx(exampleCAGEexp)
```

`flagByUpstreamSequences`
Filter by upstream sequences

Description

Looks up the bases directly upstream provided *genomic ranges* and searches for a gapless match with a *target* seqence within a given edit *distance*.

Usage

```
flagByUpstreamSequences(object, target, distance = 0)

## S4 method for signature 'CTSS'
flagByUpstreamSequences(object, target, distance = 0)

## S4 method for signature 'TagClusters'
flagByUpstreamSequences(object, target, distance = 0)

## S4 method for signature 'ConsensusClusters'
flagByUpstreamSequences(object, target, distance = 0)

## S4 method for signature 'GRanges'
flagByUpstreamSequences(object, target, distance = 0)
```

Arguments

`object`	A `CTSS`, a `TagClusters`, `ConsensusClusters` or a `GenomicRanges::GRanges` object from which a *BSgenome* object can be reached.
`target`	A target sequence.
`distance`	The maximal edit distance between the genome and the target sequence (default: 0).

Details

If the provided `object` represents *tag clusters* or *consensus clusters*, the search will be done upstream its *dominant peak*. Convert the object to the `GRanges` class if this is not the behaviour you want.

Value

A `logical-RLe` vector indicating if ranges matched the target.

Author(s)

Charles Plessy

See Also

Other CAGEr filter functions: `filteredCTSSidx()`, `flagLowExpCTSS()`

`flagLowExpCTSS` *Flag CTSSes based on sample expression*

Description

Flag CTSSes for that do not pass an expression threshold in at least a given number of samples. This is typically used to ignore CTSSes that have been seen only once in a single sample, as they can be considered to not be reproduced.

Usage

```
flagLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'CAGEr'
flagLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'RangedSummarizedExperiment'
flagLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'DataFrame'
flagLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'matrix'
flagLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
```

```
  thresholdIsTpm = TRUE
)

filterLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)

## S4 method for signature 'CAGEr'
filterLowExpCTSS(
  object,
  threshold = 1,
  nrPassThreshold = 1,
  thresholdIsTpm = TRUE
)
```

Arguments

`object`	An object from the *CAGEr* package that contains expression values for multiple samples.
`threshold`	A CTSS passes the threshold in a sample when its signal in that sample is `> threshold`. A signal equal to the threshold does not pass it.
`nrPassThreshold`	Minimal number of samples in which a CTSS must pass the threshold to not be flagged. If it is larger than the number of samples, the number of samples is used instead. With the default value of 1, only the CTSSs whose signal is `<= threshold` in all the samples are flagged.
`thresholdIsTpm`	Logical, is threshold raw tag count value (`FALSE`) or normalized signal (`TRUE`).

Value

`flagLowExpCTSS` returns a `Rle` vector where `TRUE` indicates the index of a CTSS that passes the filter (signal `> threshold` in at least `nrPassThreshold` samples), and `FALSE` a flagged CTSS.

`filterLowExpCTSS` returns the `CAGEr` object where the output of `flagLowExpCTSS` was stored internally.

See Also

Other CAGEr filter functions: `filteredCTSSidx()`, `flagByUpstreamSequences()`

Examples

```
flagLowExpCTSS(exampleCAGEexp, threshold = 100, nrPassThreshold = 2)
```

GeneExpDESeq2 *Export gene expression data for DESeq2 analysis*

Description

Creates a DESeqDataSet using the gene expression data in the experiment slot geneExpMatrix and the sample metadata of the CAGEexp object. The formula must be built using factors already present in the sample metadata.

Usage

```
GeneExpDESeq2(object, design)

## S4 method for signature 'CAGEexp'
GeneExpDESeq2(object, design)
```

Arguments

object	A CAGEexp object.
design	A formula for the DESeq2 analysis.

Author(s)

Charles Plessy

See Also

DESeqDataSet in the DESeq2 package.

Other CAGEr gene expression analysis functions: CTSStoGenes(), ranges2genes()

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
exampleCAGEexp$group <- factor(c("a", "a", "b", "b", "a"))
GeneExpDESeq2(exampleCAGEexp, ~group)
```

GeneExpSE *Retreives the SummarizedExperiment containing gene expression levels.*

Description

Get or set a SummarizedExperiment using the gene expression data in the experiment slot geneExpMatrix and the sample metadata of the CAGEexp object.

Usage

```
GeneExpSE(object)

## S4 method for signature 'CAGEexp'
GeneExpSE(object)
```

Arguments

object	A CAGEexp object.

Author(s)

Charles Plessy

See Also

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
GeneExpSE(exampleCAGEexp)
```

genomeName *Extracting genome name from CAGEr objects*

Description

Extracts the name of a referent genome from a CAGEexp or a CTSS object.

Usage

```
genomeName(object)

## S4 method for signature 'CAGEexp'
genomeName(object)

## S4 method for signature 'CTSS'
genomeName(object)

genomeName(object) <- value

## S4 replacement method for signature 'CAGEexp'
genomeName(object) <- value

## S4 replacement method for signature 'CTSS'
genomeName(object) <- value
```

Arguments

object	A CAGEexp or a CTSS object.
value	The name of a BSgenome package.

Details

CAGEexp objects constructed with NULL in place of the genome name can not run some commands that need access to genomic data, such as BigWig export or G-correction.

Value

Returns a name of a BSgenome package used as a referent genome.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Other CAGEr setter methods: inputFiles(), inputFilesType(), sampleLabels(), setColors()

Examples

```
genomeName(exampleCAGEexp)
```

getCTSS *Reading CAGE data from input file(s) and detecting TSSs*

Description

Reads input CAGE datasets into CAGEr object, constructs CAGE transcriptions start sites (CTSSs) and counts number of CAGE tags supporting every CTSS in each input experiment. See `inputFilesType` for details on the supported input formats. Preprocessing and quality filtering of input CAGE tags, as well as correction of CAGE-specific 'G' nucleotide addition bias can be also performed before constructing TSSs.

Usage

```
getCTSS(
  object,
  sequencingQualityThreshold = 10,
  mappingQualityThreshold = 20,
  removeFirstG = TRUE,
  correctSystematicG = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CAGEexp'
getCTSS(
  object,
  sequencingQualityThreshold = 10,
  mappingQualityThreshold = 20,
  removeFirstG = TRUE,
  correctSystematicG = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

`object` A `CAGEexp` object.

`sequencingQualityThreshold`
Only CAGE tags with average sequencing quality >= `sequencingQualityThreshold` and mapping quality >= `mappingQualityThreshold` are kept. Used only if `inputFileType(object) == "bam"` or `inputFileType(object) == "bamPairedEnd"`, *i.e* when input files are BAM files of aligned sequenced CAGE tags, otherwise ignored. If there are no sequencing quality values in the BAM file (*e.g.* HeliScope single molecule sequencer does not return sequencing qualities) all reads will by default have this value set to -1. Since the default value of `sequencingQualityThreshold` is 10, all the reads will consequently be discarded. To avoid this behaviour and keep all sequenced reads set `sequencingQualityThreshold` to -1 when processing data without sequencing qualities. If there is no information on mapping quality in the BAM file (*e.g.* software used to align CAGE tags to the referent genome does not provide mapping quality) the `mappingQualityThreshold` parameter is ignored. In case of paired-end sequencing BAM file (i.e. `inputFileType(object)`

== "bamPairedEnd") only the first mate of the properly paired reads (i.e. the five prime end read) will be read and subject to specified thresholds.

mappingQualityThreshold
: See sequencingQualityThreshold.

removeFirstG
: Logical, should the first nucleotide of the CAGE tag be removed in case it is a G and it does not map to the referent genome (*i.e.* it is a mismatch). Used only if `inputFileType(object) == "bam"` or `inputFileType(object) == "bamPairedEnd"`, *i.e* when input files are BAM files of aligned sequenced CAGE tags, otherwise ignored. See Details.

correctSystematicG
: Logical, should the systematic correction of the first G nucleotide be performed for the positions where there is a G in the CAGE tag and G in the genome. This step is performed in addition to removing the first G of the CAGE tags when it is a mismatch, *i.e.* this option can only be used when `removeFirstG = TRUE`, otherwise it is ignored. The frequency of adding a G to CAGE tags is estimated from mismatch cases and used to systematically correct the G addition for positions with G in the genome. Used only if `inputFileType(object) == "bam"` or `inputFileType(object) == "bamPairedEnd"`, *i.e* when input files are BAM files of aligned sequenced CAGE tags, otherwise ignored. See Details.

useMulticore
: Logical, should multicore be used. `useMulticore = TRUE` has no effect on non-Unix-like platforms.

nrCores
: Number of cores to use when `useMulticore = TRUE` (set to `NULL` to use all detected cores).

Details

In the CAGE experimental protocol an additional G nucleotide is often attached to the 5' end of the tag by the template-free activity of the reverse transcriptase used to prepare cDNA (Harbers and Carninci, Nature Methods 2005). In cases where there is a G at the 5' end of the CAGE tag that does not map to the corresponding genome sequence, it can confidently be considered spurious and should be removed from the tag to avoid misannotating actual TSS. Thus, setting `removeFirstG = TRUE` is highly recommended.

However, when there is a G both at the beginning of the CAGE tag and in the genome, it is not clear whether the original CAGE tag really starts at this position or the G nucleotide was added later in the experimental protocol. To systematically correct CAGE tags mapping at such positions, a general frequency of adding a G to CAGE tags can be calculated from mismatch cases and applied to estimate the number of CAGE tags that have G added and should actually start at the next nucleotide/position. The option `correctSystematicG` is an implementation of the correction algorithm described in Carninci *et al.*, Nature Genetics 2006, Supplementary Information section 3-e.

Value

Returns the object, in which the `tagCountMatrix` experiment will be occupied by a `RangedSummarizedExperiment` containing the expression data as a `DataFrame` of `Rle` integers, and the CTSS coordinates as genomic ranges in a `CTSS` object. The expression data can be retrieved with the `CTSStagCountDF` function. In addition, the library sizes are calculated and stored in the object's sample data (see `librarySizes`).

Author(s)

Vanja Haberle

References

Harbers and Carninci (2005) Tag-based approaches for transcriptome research and genome annotation, *Nature Methods* **2**(7):495-502.

Carninci *et al.* (2006) Genome-wide analysis of mammalian promoter architecture and evolution, *Nature Genetics* **38**(7):626-635.

See Also

inputFilesType, librarySizes.

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), distclu(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)

pathsToInputFiles <- system.file("extdata", c("Zf.unfertilized.egg.chr17.ctss",
  "Zf.30p.dome.chr17.ctss", "Zf.prim6.rep1.chr17.ctss"), package="CAGEr")

labels <- paste("sample", seq(1,3,1), sep = "")

myCAGEexp <- new("CAGEexp", genomeName = "BSgenome.Drerio.UCSC.danRer7",
 inputFiles = pathsToInputFiles, inputFilesType = "ctss", sampleLabels = labels)

myCAGEexp <- getCTSS(myCAGEexp)
```

---

getExpressionProfiles *CAGE data based expression clustering*

Description

Clusters CAGE expression across multiple experiments, both at level of individual TSSs or entire clusters of TSSs.

Usage

```
getExpressionProfiles(
  object,
  what = c("CTSS", "consensusClusters"),
  tpmThreshold = 5,
  nrPassThreshold = 1,
  method = c("som", "kmeans"),
  xDim = 5,
  yDim = 5
)

## S4 method for signature 'CAGEexp'
getExpressionProfiles(
  object,
```

```
  what = c("CTSS", "consensusClusters"),
  tpmThreshold = 5,
  nrPassThreshold = 1,
  method = c("som", "kmeans"),
  xDim = 5,
  yDim = 5
)

## S4 method for signature 'matrix'
getExpressionProfiles(
  object,
  what = c("CTSS", "consensusClusters"),
  tpmThreshold = 5,
  nrPassThreshold = 1,
  method = c("som", "kmeans"),
  xDim = 5,
  yDim = 5
)
```

Arguments

object	A CAGEexp object
what	At which level the expression clustering is done (CTSS or consensusClusters)
tpmThreshold, nrPassThreshold	Ignore clusters when their normalized CAGE signal is lower than tpmThreshold in at least nrPassThreshold experiments.
method	Method to be used for expression clustering. som uses the self-organizing map (SOM) algorithm of Toronen and coll., FEBS Letters (1999) som::som] function from *som* package. kmeans uses the K-means algorithm implemented in the stats::kmeans] function.
xDim, yDim	With method = "kmeans", xDim specifies number of clusters that will be returned by K-means algorithm and yDim is ignored. With method = "som", xDim specifies the the first and yDim the second dimension of the self-organizing map, which results in total $xDim x yDim$ clusters returned by SOM.

Details

Expression clustering can be done at level of individual CTSSs, in which case the feature vector used as input for clustering algorithm contains log-transformed and scaled (divided by standard deviation) normalized CAGE signal at individual TSS across multiple experiments. Only TSSs with normalized CAGE signal >= tpmThreshold in at least nrPassThreshold CAGE experiments are used for expression clustering. However, CTSSs along the genome can be spatially clustered into tag clusters for each experiment separately using a CTSS clustering function, and then aggregated across experiments into consensus clusters using aggregateTagClusters function. Once the consensus clusters have been created, expression clustering at the level of these wider genomic regions (representing entire promoters rather than individual TSSs) can be performed. In that case the feature vector used as input for clustering algorithm contains normalized CAGE signal within entire consensus cluster across multiple experiments, and threshold values in tpmThreshold and nrPassThreshold are applied to entire consensus clusters.

Value

Returns a modified CAGEexp object. If what = "CTSS" the objects's metadata elements CTSSexpressionClusteringMeth and CTSSexpressionClasses will be set accordingly, and if what = "consensusClusters" the elements consensusClustersExpressionClusteringMethod and consensusClustersExpressionClasses will be set. Labels of expression classes (clusters) can be retrieved using expressionClasses function.

Author(s)

Vanja Haberle

Charles Plessy

References

Toronen *et al.* (1999) Analysis of gene expression data using self-organizing maps, *FEBS Letters 451*:142-146.

See Also

Other CAGEr expression clustering functions: expressionClasses(), plotExpressionProfiles()

Examples

```
getExpressionProfiles( exampleCAGEexp, "CTSS"
                     , tpmThreshold = 50, nrPassThreshold = 1
                     , method = "som", xDim = 3, yDim = 3)

getExpressionProfiles( exampleCAGEexp, "CTSS"
                     , tpmThreshold = 50, nrPassThreshold = 1
                     , method = "kmeans", xDim = 3)

getExpressionProfiles(exampleCAGEexp, "consensusClusters")
```

---

getShiftingPromoters *Select consensus clusters with shifting score above threshold*

Description

Extracts consensus clusters with shifting score and/or FDR (adjusted P-value from Kolmogorov-Smirnov test) above specified threshold. Returns their genomic coordinates, total CAGE signal and the position of dominant TSS in the two compared groups of CAGE samples, along with the value of the shifting score, P-value and FDR. Scores and P-values/FDR have to be calculated beforehand by calling scoreShift function.

Usage

```
getShiftingPromoters(
  object,
  groupX,
  groupY,
  tpmThreshold = 0,
```

```
  scoreThreshold = -Inf,
  fdrThreshold = 1
)

## S4 method for signature 'CAGEexp'
getShiftingPromoters(
  object,
  groupX,
  groupY,
  tpmThreshold = 0,
  scoreThreshold = -Inf,
  fdrThreshold = 1
)
```

Arguments

`object`	A `CAGEexp` object.
`groupX`, `groupY`	Character vector of the one or more CAGE dataset labels in the first (`groupX`) and in the second group (`groupY`). Shifting promoters for the specified group pair are returned.
`tpmThreshold`	Consensus clusters with total CAGE signal >= `tpmThreshold` in each of the compared groups will be returned.
`scoreThreshold`	Consensus clusters with shifting score >= `scoreThreshold` will be returned. The default value `-Inf` returns all consensus clusters (for which score could be calculated, *i.e.* the ones that have at least one tag in each of the compared samples).
`fdrThreshold`	Consensus clusters with adjusted P-value (FDR) from Kolmogorov-Smirnov test >= `fdrThreshold` will be returned. The default value `1` returns all consensus clusters (for which K-S test could be performed, *i.e.* the ones that have at least one tag in each of the compared samples).

Value

Returns a `data.frame` of shifting promoters with genomic coordinates and positions of dominant TSS and CAGE signal in the two compared (groups of) samples, along with shifting score and adjusted P-value (FDR).

Author(s)

Vanja Haberle

Sarvesh Nikumbh

See Also

Other CAGEr promoter shift functions: `scoreShift()`

Examples

```
getShiftingPromoters( exampleCAGEexp
                    , groupX = "Zf.unfertilized.egg"
                    , groupY = "Zf.30p.dome") |> head()
```

hanabi *Calcultate richness in preparation for plotting*

Description

Rarefy data at multiple sample sizes using the vegan package and return a 'hanabi' object that can be passed to plot functions.

The computation can be long, so the steps of rarefaction and plotting are kept separate.

Usage

```
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'Rle'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'numeric'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'integer'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)
```

```
## S4 method for signature 'GRanges'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'List'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'list'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'matrix'
hanabi(
  x,
  n = 20,
  step = 0.75,
  from = NULL,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

x	An object contained expression counts on which richness scores can be calculated. For example an expression table in DataFrame or data.frame format where columns are samples and rows are featuressuch as genes, TSS, etc, or a vector of counts (tag counts, molecule counts, ...), or GRanges or GRangesList objects, etc.
n	The maximum number of rarefactions per sample.
step	Subsample sizes are calculated by taking the largest sample and multiplying it by the step "n" times.
from	Add one sample size (typically "0") in order to extend the plot on the left-hand side.

useMulticore	Logical, should multicore be used. useMulticore = TRUE has no effect on non-Unix-like platforms. At the moment, it also has only effects on lists and list-derived classes (data frames but not matrices).
nrCores	Number of cores to use when useMulticore = TRUE (set to NULL to use all detected cores).

Details

This function does not take directly CAGEr objects as input, because hanabi plots can be made from CTSS, clustered or gene-level data, therefore it is not possible to guess which one to use.

Value

A list-based object of class "hanabi".

Author(s)

Charles Plessy

See Also

vegan::rarecurve.

Other CAGEr richness functions: hanabiPlot(), plot.hanabi()

Examples

```
h <- hanabi(CTSStagCountDF(exampleCAGEexp))
h
plot(h)
hanabi(CTSStagCountGR(exampleCAGEexp, 2))
```

---

hanabi-class *Hanabi class*

Description

TBD

Details

TBD

hanabiPlot *hanabiPlot*

Description

Plot feature discovery curves

Usage

```
hanabiPlot(x, group, col = NULL, legend.pos = "topleft", pch = 1, ...)
```

Arguments

x	A hanabi object.
group	A character vector or a factor grouping the samples.
col	A character vector colors (at most one per group).
legend.pos	Position of the legend, passed to the legend function.
pch	Plot character at the tip of the lines and in the legend.
...	Further arguments to be passed to the plot.hanabi function.

Details

Plots the number of features (genes, transcripts, ...) detected for a given number of counts (reads, unique molecules, ...). Each library is sub-sampled by rarefaction at various sample sizes, picked to provide enough points so that the curves look smooth. The final point is plotted as an open circle, hence the name "hanabi", which means fireworks in Japanese.

The rarefactions take time to do, so this step is done by a separate function, so that the result is easily cached.

Author(s)

Charles Plessy

See Also

Other CAGEr richness functions: hanabi(), plot.hanabi()

Other CAGEr plot functions: TSSlogo(), plotAnnot(), plotCorrelation(), plotExpressionProfiles(), plotInterquantileWidth(), plotReverseCumulatives()

Examples

```
h <- hanabi(CTSStagCountDF(exampleCAGEexp))
hanabiPlot(h, group = 1:5)
hanabiPlot(hanabi(CTSStagCountDF(exampleCAGEexp), n = 20, step = 0.8, from = 25000), group = 1:5)
hanabiPlot(hanabi(CTSStagCountDF(exampleCAGEexp), n = 10, step = 0.98), group = 1:5)
hanabiPlot(h, group=c("A", "A", "B", "C", "B"), col=c("red", "green", "blue"))
hanabiPlot(h, group = 1:5, pch=1:5, col="purple")
```

`import.bam` *import.bam*

Description

Imports CTSS data from a BAM file.

Usage

```
import.bam(
  filepath,
  filetype,
  sequencingQualityThreshold = 10,
  mappingQualityThreshold = 20
)
```

Arguments

`filepath`	The path to the BAM file.
`filetype`	bam or bamPairedEnd.
`sequencingQualityThreshold`	See getCTSS().
`mappingQualityThreshold`	See getCTSS().

See Also

Other loadFileIntoGPos: `bam2CTSS()`, `import.CTSS()`, `import.bam.ctss()`, `import.bedCTSS()`, `import.bedScore()`, `import.bedmolecule()`, `import.bigwig()`, `loadFileIntoGPos()`, `moleculesGR2CTSS()`

Examples

```
# TODO: add exmaple file
# import.bam(system.file("extdata", "example.bam", package = "CAGEr"))
```

`import.bam.ctss` *import.bam.ctss*

Description

Imports CTSS data from a BAM file.

Usage

```
import.bam.ctss(
  filepath,
  filetype,
  sequencingQualityThreshold,
  mappingQualityThreshold,
  removeFirstG,
  correctSystematicG,
  genome
)
```

Arguments

filepath	The path to the BAM file.
filetype	bam or bamPairedEnd.
sequencingQualityThreshold	See getCTSS().
mappingQualityThreshold	See getCTSS().
removeFirstG	See getCTSS().
correctSystematicG	See getCTSS().
genome	See coerceInBSgenome().

Value

Returns a CTSS object.

See Also

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bedCTSS(), import.bedScore(), import.bedmolecule(), import.bigwig(), loadFileIntoGPos(), moleculesGR2CTSS()

---

import.bedCTSS *import.bedCTSS*

Description

Imports a BED file where each line represents a single base, with a score counting the number of CAGE transcription start sites (CTSS).

Usage

```
import.bedCTSS(filepath)
```

Arguments

filepath	The path to the BED file.

Value

A GRanges object where each line represents one nucleotide.

See Also

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bam.ctss(), import.bedScore(), import.bedmolecule(), import.bigwig(), loadFileIntoGPos(), moleculesGR2CTSS()

Examples

```
# TODO: add exmaple file
# import.BED(system.file("extdata", "example.bed", package = "CAGEr"))
```

---

import.bedmolecule *import.bedmolecule*

Description

Imports a BED file where each line counts for one molecule in a GRanges object where each line represents one nucleotide.

Usage

```
import.bedmolecule(filepath)
```

Arguments

filepath	The path to the BED file.

Value

Returns a CTSS object.

See Also

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bam.ctss(), import.bedCTSS(), import.bedScore(), import.bigwig(), loadFileIntoGPos(), moleculesGR2CTSS()

Examples

```
# TODO: add exmaple file
# import.BED(system.file("extdata", "example.bed", package = "CAGEr"))
```

import.bedScore *import.bedScore*

Description

Imports a BED file where the score indicates a number of counts for a given alignment.

Usage

```
import.bedScore(filepath)
```

Arguments

filepath	The path to the BED file.

Value

A GRanges object where each line represents one nucleotide.

See Also

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bam.ctss(), import.bedCTSS(), import.bedmolecule(), import.bigwig(), loadFileIntoGPos(), moleculesGR2CTSS()

Examples

```
# TODO: add exmaple file
# import.bedScore(system.file("extdata", "example.bed", package = "CAGEr"))
```

import.bigwig *Read in BigWig files to CAGEexp object*

Description

Read in BigWig files to CAGEexp object

Usage

```
import.bigwig(filepath)
```

Arguments

filepath	Path to an input bigwig file on the plus strand.

Value

a CAGEexp object

Note

The filename must contain the string `str1` and have a pair for the minus strand that has identical path except with the `str2` substring, so that `inputFiles` CAGEexp object will only point to plus-strand BigWig files.

Author(s)

Katalin Ferenc

Damir Baranasic

See Also

Other loadFileIntoGPos: `bam2CTSS()`, `import.CTSS()`, `import.bam()`, `import.bam.ctss()`, `import.bedCTSS()`, `import.bedScore()`, `import.bedmolecule()`, `loadFileIntoGPos()`, `moleculesGR2CTSS()`

Examples

```
pathsToInputFiles <- list.files( system.file("extdata", package = "CAGEr")
                                 , "str1.out.wig.bw$" , full.names = TRUE)

CAGEr:::import.bigwig(pathsToInputFiles[1])

## Not run:
sampleLabels <- sub( "_subsampled_str1.Signal.Unique.str1.out.wig.bw"
                   , "", basename(pathsToInputFiles))
CAGEexp( genomeName     = "BSgenome.Drerio.UCSC.danRer7"
       , inputFiles     = pathsToInputFiles
       , inputFilesType = "bigwig"
       , sampleLabels   = sampleLabels) |> getCTSS()

## End(Not run)
```

`import.CAGEscanMolecule` *import.CAGEscanMolecule*

Description

Imports a CAGEscan "molecule" file in a `GRanges` object

Usage

```
import.CAGEscanMolecule(filepath)
```

Arguments

`filepath`	The path to the "molecule" file.

See Also

parseCAGEscanBlocksToGrangeTSS

Examples

```
# TODO import.CAGEscanMolecule(system.file("extdata", "example.molecule.txt", package = "CAGEr"))
```

`import.CTSS` *import.CTSS*

Description

Imports a "CTSS" file in a GPos object

Usage

```
import.CTSS(filepath)
```

Arguments

`filepath`	The path to the "CTSS" file. Note that the format of the "CTSS" files handled in this function is not the same as the FANTOM5 "CTSS" files (which are plain BED).

See Also

Other loadFileIntoGPos: `bam2CTSS()`, `import.bam()`, `import.bam.ctss()`, `import.bedCTSS()`, `import.bedScore()`, `import.bedmolecule()`, `import.bigwig()`, `loadFileIntoGPos()`, `moleculesGR2CTSS()`

Examples

```
CAGEr:::import.CTSS(system.file("extdata", "Zf.high.chr17.ctss", package = "CAGEr"))
```

`importPublicData` *importPublicData*

Description

Imports CAGE data from different sources into a `CAGEexp` object. After the object has been created the data can be further manipulated and visualized using other functions available in the *CAGEr* package and integrated with other analyses in R. Available resources include:

Usage

```
importPublicData(
  origin = c("FANTOM5", "FANTOM3and4", "ENCODE", "ZebrafishDevelopment"),
  dataset,
  group,
  sample
)

## S4 method for signature 'character,character,ANY,character'
importPublicData(
```

```
  origin = c("FANTOM5", "FANTOM3and4", "ENCODE", "ZebrafishDevelopment"),
  dataset,
  group,
  sample
)
```

Arguments

origin	Character vector specifying one of the available resources for CAGE data ("FANTOM5", "FANTOM3and4", "ENCODE" or "ZebrafishDevelopment").
dataset	Character vector specifying one or more of the datasets available in the selected resource. For FANTOM5 it can be either "human" or "mouse", and only one of them can be specified at a time. For other resources please refer to the vignette of the corresponding data package for the list of available datasets. Multiple datasets mapped to the same genome can be specified to combine selected samples from each.
group	Character string specifying one or more groups within specified dataset(s), from which the samples should be selected. The group argument is used only when importing TSSs from data packages and ignored for "FANTOM5". For available groups in each dataset please refer to the vignette of the corresponding data package. Either only one group has to be specified (if all selected samples belong to the same group) or one group per sample (if samples belong to different groups). In the latter case, the number of elements in group must match the number of elements in sample.
sample	Character string specifying one or more CAGE samples. Check the corresponding data package for available samples within each group and their labels. For FANTOM5 resource, list of all human (~1000) and mouse (~) samples can be obtained in *CAGEr* by loading data(FANTOM5humanSamples) and data(FANTOM5mouseSamples), respectively. Use the names from the sample column to specify which samples should be imported.

Details

- FANTOM5 datasets (Forrest *et al.,* Nature 2014) for numerous human and mouse samples (primary cells, cell lines and tissues), which are fetched directly from FANTOM5 online resource at https://fantom.gsc.riken.jp/5/data.
- FANTOM3 and 4 datasets (Carninci _et al., _ Science 2005, Faulkner *et al.,* Nature Genetics 2009, Suzuki *et al.* Nature Genetics 2009) from *FANTOM3and4CAGE* data package available from Bioconductor.
- ENCODE datasets (Djebali *et al.* Nature 2012) for numerous human cell lines from *ENCODEprojectCAGE* data package, which is available for download from http://promshift.genereg.net/CAGEr/.
- Zebrafish (*Danio rerio*) developmental timecourse datasets (Nepal *et al.* Genome Research 2013) from *ZebrafishDevelopmentalCAGE* data package, which is available for download from http://promshift.genereg.net/CAGEr/.

Value

A CAGEexp object is returned, containing information on library size, CTSS coordinates and tag count matrix. The object is ready for *CAGEr* analysis (normalisation, tag clustering, . . .).

Author(s)

Vanja Haberle

Charles Plessy

References

- Carninci *et al.,* (2005). *The Transcriptional Landscape of the Mammalian Genome*. Science **309**(5740):1559-1563.
- Djebali *et al.,* (2012). *Landscape of transcription in human cells.* Nature **488**(7414):101-108.
- Faulkner *et al.,* (2009). *The regulated retrotransposon transcriptome of mammalian cells.*, Nature Genetics **41**:563-571.
- Forrest *et al.,* (2014). *A promoter-level mammalian expression atlas.* Nature **507**(7493):462-470.
- Nepal *et al.,* (2013). *Dynamic regulation of the transcription initiation landscape at single nucleotide resolution during vertebrate embryogenesis*. Genome Research **23**(11):1938-1950.
- Suzuki_et al.,_ (2009). The transcriptional network that controls growth arrest and differentiation in a human myeloid leukemia cell line_. Nature Genetics **41**:553-562.

See Also

Other FANTOM data: FANTOM5humanSamples, FANTOM5mouseSamples

Examples

```
## Not run:
### importing FANTOM5 data

# list of FANTOM5 human tissue samples

data(FANTOM5humanSamples)
head(subset(FANTOM5humanSamples, type == "tissue"))

# import selected samples
f5 <- importPublicData(
  origin="FANTOM5", dataset = "human",
  sample = c("adipose_tissue__adult__pool1", "adrenal_gland__adult__pool1",
             "aorta__adult__pool1"))

CTSScoordinatesGR(f5)

### importing FANTOM3/4 data from a data package

library(FANTOM3and4CAGE)

# list of mouse datasets available in this package

data(FANTOMmouseSamples)
unique(FANTOMmouseSamples$dataset)
head(subset(FANTOMmouseSamples, dataset == "FANTOMtissueCAGEmouse"))
head(subset(FANTOMmouseSamples, dataset == "FANTOMtimecourseCAGEmouse"))

# import selected samples from two different mouse datasets
```

```
f34 <- importPublicData(
 origin="FANTOM3and4", dataset = c("FANTOMtissueCAGEmouse", "FANTOMtimecourseCAGEmouse"),
  group = c("brain", "adipogenic_induction"),
 sample = c("CCL-131_Neuro-2a_treatment_for_6hr_with_MPP+", "DFAT-D1_preadipocytes_2days"))

f34 <- importPublicData(
  origin="FANTOM3and4", dataset = c("FANTOMtissueCAGEmouse"),
  group = c("brain"),
  sample = c("CCL-131_Neuro-2a_treatment_for_6hr_with_MPP+"))

CTSScoordinatesGR(f34)


## End(Not run)
```

`inputFiles` *Extracting paths to input files from CAGEr objects*

Description

Extracts the paths to CAGE data input files from `CAGEexp` objects.

Usage

```
inputFiles(object)

## S4 method for signature 'CAGEexp'
inputFiles(object)

inputFiles(object) <- value

## S4 replacement method for signature 'CAGEexp'
inputFiles(object) <- value
```

Arguments

`object`	A CAGEexp object.
`value`	A character vector with one file path per sample.

Value

Returns a character vector of paths to CAGE data input files.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Other CAGEr setter methods: genomeName(), inputFilesType(), sampleLabels(), setColors()

Examples

```
inputFiles(exampleCAGEexp)
```

---

inputFilesType *Input file formats for CAGEr objects*

Description

Get or set the information on the type of CAGE data input files from CAGEexp objects.

Usage

```
inputFilesType(object)

## S4 method for signature 'CAGEexp'
inputFilesType(object)

inputFilesType(object) <- value

## S4 replacement method for signature 'CAGEexp'
inputFilesType(object) <- value
```

Arguments

object	A CAGEexp object.
value	A character vector with one file type per sample.

Details

The following input file types are supported:

- bam: A single-ended BAM file.
- bamPairedEnd: A paired-ended BAM file.
- bed: A BED file where each line counts for one molecule.
- bedScore: A BED file where the score indicates a number of counts for a given alignment.
- CAGEscanMolecule: Experimental. For the CAGEscan 3.0 pipeline.
- ctss: A tabulation-delimited file describing CAGE Transcription Start Sites (CTSS) with four columns indicating *chromosome*, *1-based coordinate*, *strand* and *score* respectively.
- CTSStable
- FANTOM5

- ENCODE
- FANTOM3and4
- ZebrafishDevelopment

Value

Returns the type of the file format of CAGE data input files, *e.g.* "bam" or "ctss". In the case of CAGEexp objects, the return value is character vector with one member per sample.

Author(s)

Vanja Haberle

Charles Plessy

See Also

getCTSS

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), librarySizes(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Other CAGEr setter methods: genomeName(), inputFiles(), sampleLabels(), setColors()

Examples

```
inputFilesType(exampleCAGEexp)
```

---

librarySizes *Extracting library sizes from CAGEr objects*

Description

Extracts the library sizes (total number of CAGE tags) for all CAGE datasets from CAGEexp objects.

Usage

```
librarySizes(object)

## S4 method for signature 'CAGEexp'
librarySizes(object)
```

Arguments

object	A CAGEexp object.

Details

Library sizes are calculated when loading data with the getCTSS function and stored in the librarySizes column of the colData of CAGEexp objects.

Value

Returns an integer vector of total number of CAGE tags (library size) for all CAGE datasets in the CAGEr object.

Author(s)

Vanja Haberle

See Also

getCTSS

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), sampleLabels(), seqNameTotalsSE(), tagClustersGR()

Examples

```
librarySizes(exampleCAGEexp)
```

---

loadFileIntoGPos *loadFileIntoGPos*

Description

A private (non-exported) function to load from each file format supported by CAGEr

Usage

```
loadFileIntoGPos(
  filepath,
  filetype = c("bam", "bamPairedEnd", "bigwig", "bed", "bedScore", "bedctss",
    "CAGEscanMolecule", "ctss"),
  sequencingQualityThreshold,
  mappingQualityThreshold,
  removeFirstG,
  correctSystematicG,
  genome
)
```

Arguments

filepath	The path to the file to load.
filetype	The type of the file
sequencingQualityThreshold	See getCTSS().
mappingQualityThreshold	See getCTSS().
removeFirstG	See getCTSS().

correctSystematicG
: See getCTSS().

genome
: See coerceInBSgenome().

Value

A GPos() object where the score represents the number of CAGE tags starting on that nucleotide.

See Also

import.CTSS

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bam.ctss(), import.bedCTSS(), import.bedScore(), import.bedmolecule(), import.bigwig(), moleculesGR2CTSS()

---

mapStats *Process mapping statistics*

Description

Using a data frame containing mapping statistics in counts, transform the data in percentages that can be used for stacked barplots.

Usage

```
mapStats(libs, scope, group = "sampleLabels", facet = NULL, normalise = TRUE)
```

Arguments

libs
: A data frame with containing columns required by the scope chosen.

scope
: The name of a "scope", that defines which data is plotted and how it is normalised, or a function that implements a custom scope. See mapStatsScopes() for details on each scope.

group
: A vector of factors defining groups in the data. By default, the sample labels (which means no grouping).

facet
: A vector of factors defining facets in the data (in the sense of ggplot2's facet_wrap function).

normalise
: Whether to normalise or not. Default: TRUE.

Details

See the plotAnnot vignette and the mapStatsScopes() help page for details on what the scopes are.

See http://stackoverflow.com/questions/10417003/stacked-barplot-with-errorbars-using-ggplot2 about stacked barplot.

Value

Returns a data frame with mean and standard deviation of normalised mapping statistics, plus absolute positions for the error bars. The first column, group, is a vector of factors sorted with the gtools::mixedorder() function. The facet column, if any, is always called facet.

Author(s)

Charles Plessy

See Also

plotAnnot, mapStatsScopes

Examples

```
CAGEr:::mapStats(as.data.frame(colData(exampleCAGEexp)), "counts", sampleLabels(exampleCAGEexp))
CAGEr:::mapStats(as.data.frame(colData(exampleCAGEexp)), "counts", c("A", "A", "B", "B", "C"))
```

---

mapStatsScopes *mapStats scopes*

Description

Functions implementing the scope parameter of the \link{mapStats} function.

Usage

```
msScope_counts(libs)

msScope_mapped(libs)

msScope_qc(libs)

msScope_steps(libs)

msScope_all(libs)

msScope_annotation(libs)
```

Arguments

libs	A data frame containing metadata describing samples in sequence libraries.

Details

The counts scope reports the number of molecules aligning in *promoter*, *exon*, *intron* and otherwise *intergenic*. regions.

The mapped scope reports the number of molecules aligning in *promoter*, *exon*, *intron* and otherwise *intergenic*, plus the number of PCR duplicates (mapped tags minus molecule counts), plus the number of non-properly paired mapped tags.

The qc scope reports the number of tags removed as *tag dust*, *rRNA*, *spikes*, plus the *unmapped* tags, plus the number of non-properly paired mapped tags, plus the number of PCR duplicates (mapped tags minus molecule counts), plus the number of unique molecule counts.

The steps scope reports the number of tags removed by *cleaning*, *mapping*, and *deduplication*, plus the number of *unique molecule counts*.

The legacy `all` scope reports the number of tags in *promoters*, *exons*, *introns*, or *mapped* elswhere, or removed because they match rRNA or are likely primer artefacts, normalised by the total nubmer of extracted tags.

The legacy `annotation` scope reports the number of tags in *promoters*, *exons*, *introns*, or *mapped* elswhere, or removed because they match rRNA or are likely primer artefacts, normalised by the total nubmer of mapped tags.

Value

Returns a list with three elements: `libs` contains a modified version of the input data frame where columns have been reorganised as needed, `colums` contains the names of the columns to use for plotting and provides the order of the stacked bars of the `plotAnnot` function, `total` indicates the total counts used for normalising the data.

`mergeCAGEsets` *Merge two CAGEr objects into one*

Description

Merges two `CAGEr` objects into one by combining the CTSS genomic coordinates and raw tag counts. The resulting object will contain a union of TSS positions present in the two input objects and raw tag counts for those TSSs in all samples from both input objects.

Usage

```
mergeCAGEsets(cs1, cs2)

## S4 method for signature 'CAGEexp,CAGEexp'
mergeCAGEsets(cs1, cs2)
```

Arguments

`cs1`	A `CAGEr` object
`cs2`	A `CAGEr` object

Value

Note that merging discards all other information present in the two `CAGEr` objects, that is, the merged object will not contain any normalised tag counts, CTSS clusters, quantile positions, etc., so these have to be calculated again by calling the appropriate functions on the merged object. Also, it is only possible to merge two objects that contain TSS information for the same reference genome and do not share any sample names.

Returns a `CAGEexp` object, which contains a union of TSS positions present in the two input objects and raw tag counts for those TSSs in all samples from both input objects.

Author(s)

Vanja Haberle

Charles Plessy

See Also

CAGEexp

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)

pathsToInputFiles <- system.file("extdata", c("Zf.unfertilized.egg.chr17.ctss",
  "Zf.30p.dome.chr17.ctss", "Zf.prim6.rep1.chr17.ctss"), package="CAGEr")

ce1 <- CAGEexp(genomeName = "BSgenome.Drerio.UCSC.danRer7",
inputFiles = pathsToInputFiles[1:2], inputFilesType = "ctss", sampleLabels =
c("sample1", "sample2"))
ce1 <- getCTSS(ce1)

ce2 <- CAGEexp(genomeName = "BSgenome.Drerio.UCSC.danRer7",
inputFiles = pathsToInputFiles[3], inputFilesType = "ctss", sampleLabels =
"sample3")

ce2 <- getCTSS(ce2)

ce <- mergeCAGEsets(ce1, ce2)
```

---

mergeSamples *Merge CAGE samples*

Description

Merges individual CAGE samples (datasets, experiments) within the CAGEr object into specified groups.

Usage

```
mergeSamples(object, mergeIndex, mergedSampleLabels)

## S4 method for signature 'CAGEexp'
mergeSamples(object, mergeIndex, mergedSampleLabels)
```

Arguments

object	A CAGEr object.
mergeIndex	Integer vector specifying which experiments should be merged. (one value per sample, see Details).
mergedSampleLabels	Labels for the merged datasets (same length as the number of unique values in mergeIndex)

Details

The samples within the CAGEr object are merged by adding the raw tag counts of individual CTSS that belong tho the same group. After merging, all other slots in the CAGEr object will be reset and any previous data for individual experiments will be removed.

`mergeIndex` controls which samples will be merged. It is an integer vector that assigns a group identifier to each sample, in the same order as they are returned by `sampleLabels(object)`. For example, if there are 8 CAGE samples in the CAGEr object and `mergeIndex = c(1,1,2,2,3,2,4,4)`, this will merge a) samples 1 and 2, b) samples 3, 4 and 6, c) samples 7 and 8, and d) it will leave sample 5 as it is, resulting in 4 final merged datasets.

Labels provided in `mergedSampleLabels` will be assigned to merged datasets in the ascending order of `mergeIndex` values, *i.e.* first label will be assigned to a dataset created by merging datasets labeled with lowest `mergeIndex` value (in this case 1), *etc*.

Value

The slots `sampleLabels`, `librarySizes` and `tagCountMatrix` of the provided `CAGEr` object will be updated with the information on merged CAGE datasets and will replace the previous information on individual CAGE datasets. All further slots with downstream information will be reset.

Author(s)

Vanja Haberle

Charles Plessy

Examples

```
mergeSamples( exampleCAGEexp
            , mergeIndex = c(3,2,4,4,1)
        , mergedSampleLabels = c("zf_unfertilized", "zf_high", "zf_30p_dome", "zf_prim6"))
exampleCAGEexp
```

---

moleculesGR2CTSS *moleculesGR2CTSS*

Description

Calculates CTSS positions from a GenomicRanges object where each element represents a single molecule.

Usage

```
moleculesGR2CTSS(gr)
```

Arguments

`gr`	A GRanges object.

Value

Returns a GRanges object.

See Also

Other loadFileIntoGPos: bam2CTSS(), import.CTSS(), import.bam(), import.bam.ctss(), import.bedCTSS(), import.bedScore(), import.bedmolecule(), import.bigwig(), loadFileIntoGPos()

Examples

```
gr <- GenomicRanges::GRanges("chr1", IRanges::IRanges(1, 10), c("+", "-", "+"))
CAGEr:::moleculesGR2CTSS(gr)
```

---

normalizeTagCount *Normalizing raw CAGE tag count*

Description

Normalizes raw CAGE tag count per CTSS in all experiments to a same referent distribution. A simple tag per million normalization or normalization to a referent power-law distribution (Balwierz *et al*., Genome Biology 2009) can be specified.

Usage

```
normalizeTagCount(
  object,
  method = c("powerLaw", "simpleTpm", "none"),
  fitInRange = c(10, 1000),
  alpha = 1.25,
  T = 10^6
)

## S4 method for signature 'CAGEexp'
normalizeTagCount(
  object,
  method = c("powerLaw", "simpleTpm", "none"),
  fitInRange = c(10, 1000),
  alpha = 1.25,
  T = 10^6
)
```

Arguments

object	A CAGEexp object
method	Method to be used for normalization. Can be either "simpleTpm" to convert tag counts to tags per million or "powerLaw" to normalize to a referent power-law distribution, or "none" to keep using the raw tag counts in downstream analyses.
fitInRange	An integer vector with two values specifying a range of tag count values to be used for fitting a power-law distribution to reverse cumulatives. Used only when method = "powerLaw", otherwise ignored. See Details.
alpha	-1 * alpha will be the slope of the referent power-law distribution in the log-log representation. Used only when method = "powerLaw", otherwise ignored. See Details.

T — Total number of CAGE tags in the referent power-law distribution. Setting `T = 10^6` results in normalized values that correspond to tags per million in the referent distribution. Used only when `method = "powerLaw"`, otherwise ignored. See Details.

Details

It has been shown that many CAGE datasets follow a power-law distribution (Balwierz *et al.*, Genome Biology 2009). Plotting the number of CAGE tags (X-axis) against the number of TSSs that are supported by >= of that number of tags (Y-axis) results in a distribution that can be approximated by a power-law. On a log-log scale this theoretical referent distribution can be described by a monotonically decreasing linear function `y = -1 * alpha * x + beta`, which is fully determined by the slope `alpha` and total number of tags `T` (which together with `alpha` determines the value of `beta`). Thus, by specifying parameters `alpha` and `T` a desired referent power-law distribution can be selected. However, real CAGE datasets deviate from the power-law in the areas of very low and very high number of tags, so it is advisable to discard these areas before fitting a power-law distribution. `fitInRange` parameter allows to specify a range of values (lower and upper limit of the number of CAGE tags) that will be used to fit a power-law. Plotting reverse cumulatives using `plotReverseCumulatives` function can help in choosing the best range of values. After fitting a power-law distribution to each CAGE dataset individually, all datasets are normalized to a referent distribution specified by `alpha` and `T`. When `T = 10^6`, normalized values are expressed as tags per million (tpm).

Value

The slot `normalizedTpmMatrix` of the provided `CAGEexp` object will be occupied by normalized CAGE signal values per CTSS across all experiments, or with the raw tag counts (in case `method = "none"`).

Author(s)

Vanja Haberle

References

Balwierz *et al.* (2009) Methods for analyzing deep sequencing expression data: constructing the human and mouse promoterome with deepCAGE data, *Genome Biology* **10**(7):R79.

See Also

`plotReverseCumulatives`, `CTSSnormalizedTpmDF`

Other CAGEr object modifiers: `CTSStoGenes()`, `CustomConsensusClusters()`, `aggregateTagClusters()`, `annotateCTSS()`, `cumulativeCTSSdistribution()`, `distclu()`, `getCTSS()`, `paraclu()`, `quantilePositions()`, `quickEnhancers()`, `resetCAGEexp()`, `summariseChrExpr()`

Other CAGEr normalised data functions: `plotReverseCumulatives()`

Examples

```
ce1 <- normalizeTagCount(exampleCAGEexp, method = "simpleTpm")
ce2 <- normalizeTagCount(exampleCAGEexp, method = "powerLaw")
```

paraclu *Parametric clustering*

Description

"paraclu" is an implementation of Paraclu algorithm for parametric clustering of data attached to sequences (Frith *et al.*, Genome Research, 2007). Since Paraclu finds clusters within clusters (unlike distclu), additional parameters (minStability, maxLength and reduceToNonoverlapping) can be specified to simplify the output by discarding too big clusters, and to reduce the clusters to a final set of non-overlapping clusters.

Usage

```
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'Pairs'
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CTSS'
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'GRanges'
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
```

```
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'SummarizedExperiment'
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CAGEexp'
paraclu(
  object,
  minStability = 1,
  maxLength = 500,
  keepSingletonsAbove = 0,
  reduceToNonoverlapping = TRUE,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

`object`	A `CTSS`, or a `S4Vectors::Pairs` object with positions *first* and scores *second*.
`minStability`	Minimal stability of the cluster, where stability is defined as ratio between maximal and minimal density value for which this cluster is maximal scoring. For definition of stability refer to Frith *et al.*, Genome Research, 2007. Clusters with stability < `minStability` will be discarded.
`maxLength`	Maximal length of cluster in base-pairs. Clusters with length > `maxLength` will be discarded.
`keepSingletonsAbove`	Remove "singleton" tag clusters of width 1 with signal < `keepSingletonsAbove`. Default value `0` results in keeping all TCs by default. Setting it to `Inf` removes all singletons.
`reduceToNonoverlapping`	Logical, should smaller clusters contained within bigger cluster be removed to make a final set of tag clusters non-overlapping.
`useMulticore`	Logical, should multicore be used. `useMulticore = TRUE` has no effect on non-Unix-like platforms.
`nrCores`	Number of cores to use when `useMulticore = TRUE`. Default value `NULL` uses all detected cores.

Details

Clustering is done for every CAGE dataset within the CAGEr object separately, resulting in a different set of tag clusters for every CAGE dataset. TCs from different datasets can further be aggregated

into a single referent set of consensus clusters by calling the aggregateTagClusters function.

Value

Running Paraclu on a Pairs object containing positions and scores returns an IRanges object containing the start and end positions of the clusters, as well as the minimum and maximum density in min_d and max_d metadata columns.

Running Paraclu on a CTSS object dispatches the computation on each strand of each sequence level of the object, collects the IRanges and assemble them back in a TagClusters object after filtering them by size and by expression following the minStability, maxLength, keepSingletonsAbove and reduceToNonoverlapping parameters.

Running Paraclu on a RangedSummarizedExperiment object will loop on each sample, and return the results as a GRangesList of TagClusters.

Running Paraclu on a CAGEexp returnts is with the clusters stored as a GRangesList of TagClusters objects in its metadata slot tagClusters.

Author(s)

Vanja Haberle

Charles Plessy

References

MC Frith, E Valen, A Krogh, Y Hayashizaki, P Carninci, A Sandelin. *A code for transcription initiation in mammalian genomes.* Genome Research 2008 18(1):1-12)

See Also

aggregateTagClusters

Other CAGEr clustering methods: consensusClustersTpm(), distclu()

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), distclu(), getCTSS(), normalizeTagCount(), quantilePositions(), quickEnhancers(), resetCAGEexp(), summariseChrExpr()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), consensusClustersGR(), cumulativeCTSSdistributio distclu(), plotInterquantileWidth(), quantilePositions(), tagClustersGR()

Examples

```
(ctss <- CTSSnormalizedTpmGR(exampleCAGEexp,1))
(pair <- Pairs(pos(ctss), score(ctss)))
CAGEr:::.paraclu_params(first(pair), second(pair))
CAGEr:::.paraclu(first(pair)[1:10], second(pair)[1:10])
paraclu(pair[1:10])
paraclu(ctss[1:10])
paraclu(CTSStagCountSE(exampleCAGEexp)[1:25,])
ce <- paraclu( exampleCAGEexp,
             , keepSingletonsAbove = 100
             , maxLength = 500, minStability = 1
             , reduceToNonoverlapping = TRUE)
tagClustersGR(ce, "Zf.30p.dome")
```

parseCAGEscanBlocksToGrangeTSS *parseCAGEscanBlocksToGrangeTSS*

Description

Parse a string describing a block in a CAGEscan molecule, as output by the "CAGEscan 3.0" pipeline.

Usage

```
parseCAGEscanBlocksToGrangeTSS(blocks)
```

Arguments

blocks	A character string representing a block in a CAGEscan molecule.

Value

A GRanges object representing a TSS.

In CAGEscan molecules, blocks are separated by 'l', ',' or ';' for gap of coverage, splice junction (confident) and splice junction (maybe) respectively. Strand is "+" if first coordinate is lower than the second one, and "-" otherwise.

See Also

import.CAGEscanMolecule

Examples

```
myMolecule <- paste0( "chr11:66268633-66268693,"
                    , "chr11:66271796-66271869;"
                    , "chr11:66272156-66272252|"
                    , "chr11:66272364-66272460")
myFirstBlock <- sub("[,;|].*", "", myMolecule)

CAGEr:::parseCAGEscanBlocksToGrangeTSS(myFirstBlock)
```

plot.hanabi *Plotting Hanabi objects*

Description

S3 method to plot hanabi objects. Used by the hanabiPlot function.

Usage

```
## S3 method for class 'hanabi'
plot(
  x,
  alpha = 0.5,
  col = "black",
  xlab = "Total counts",
  ylab = "Unique features",
  main = "Hanabi plot",
  pch = 1,
  ...
)

## S3 method for class 'hanabi'
points(x, ...)

## S3 method for class 'hanabi'
lines(x, ...)
```

Arguments

x	The hanabi object to plot.
alpha	The alpha transparency of the plot lines.
col	A vector indicating a color per sample (or a vector that can be recycled that way).
xlab	Horizontal axis label.
ylab	Vertical axis label.
main	Plot title.
pch	Plot character at the tip of the lines.
...	Other parameters passed to the generic plot, points or lines functions.

Author(s)

Charles Plessy

See Also

Other CAGEr richness functions: hanabi(), hanabiPlot()

---

plotAnnot *Plot annotation statistics*

Description

Extracts processing and alignment statistics from a *CAGEr* object and plots them as counts or percentages in stacked barplots.

Usage

```
plotAnnot(
  x,
  scope,
  title,
  group = "sampleLabels",
  facet = NULL,
  normalise = TRUE
)

## S4 method for signature 'data.frame'
plotAnnot(
  x,
  scope,
  title,
  group = "sampleLabels",
  facet = NULL,
  normalise = TRUE
)

## S4 method for signature 'DataFrame'
plotAnnot(
  x,
  scope,
  title,
  group = "sampleLabels",
  facet = NULL,
  normalise = TRUE
)

## S4 method for signature 'CAGEexp'
plotAnnot(
  x,
  scope,
  title,
  group = "sampleLabels",
  facet = NULL,
  normalise = TRUE
)

## S4 method for signature 'GRangesList'
plotAnnot(
  x,
  scope,
  title,
  group = "sampleLabels",
  facet = NULL,
  normalise = TRUE
)
```

Arguments

x	An object from which can be extracted a table with columns named promoter, exon, intron, mapped, extracted, rdna, and tagdust, that will be passed to the mapStats function.
scope	The name of a *scope*, that defines which data is plotted and how it is normalised, or a function implementing that scope. See mapStatsScopes for details on each scope.
title	The title of the plot.
group	A factor to group the samples, or the name of a colData column of a CAGEexp object, or a formula giving the names of columns to be pasted together. If no group is provided the sample labels will be used.
facet	A factor or the name of a colData column of a CAGEexp object, to facet the samples in the sense of ggplot2's ggplot2::facet_wrap() function.
normalise	Whether to normalise or not. Default: TRUE.

Details

When given a CAGEexp object or its *column data*, what will be counted is the number of *CAGE tags*. When given cluster objects (CTSS, TagClusters or ConsensusClusters) wrapped as a GenomicRanges::GRangesList, what will be counted is the number of *clusters*.

Stacked barplots with error bars inspired from http://stackoverflow.com/questions/10417003/stacked-barplot-with-errorbars-using-ggplot2. See http://www.biomedcentral.com/1471-2164/14/665/figure/F1 for example.

Value

Returns a ggplot2::ggplot object.

Author(s)

Charles Plessy

See Also

mapStats for a list of *scopes*.

Other CAGEr annotation functions: annotateCTSS(), ranges2annot(), ranges2genes(), ranges2names()

Other CAGEr plot functions: TSSlogo(), hanabiPlot(), plotCorrelation(), plotExpressionProfiles(), plotInterquantileWidth(), plotReverseCumulatives()

Examples

```
p <- plotAnnot(exampleCAGEexp, 'counts', 'Here is the title')
print(p)
p + ggplot2::theme_bw()
ggplot2::theme_set(ggplot2::theme_bw()) ; p
plotAnnot(exampleCAGEexp, 'counts', 'Same, non-normalised', normalise = FALSE)
exampleCAGEexp$myGroups <- factor(c("A", "A", "B", "B", "C"))
plotAnnot(exampleCAGEexp, 'counts', group = "myGroups")
plotAnnot(exampleCAGEexp, 'counts', group = ~myGroups)
plotAnnot(exampleCAGEexp, 'counts', group = ~sampleLabels + myGroups)
plotAnnot(exampleCAGEexp, CAGEr:::msScope_counts , group = "myGroups")
```

plotCorrelation *Pairwise scatter plots and correlations of CAGE signal*

Description

Functions to compute pairwise sample correlations and display them in a matrix with scatter plots below the diagonal, sample names on the diagonal, and correlation coefficients above.

Usage

```
plotCorrelation(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  plotSize = 800
)

## S4 method for signature 'CAGEr'
plotCorrelation(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  plotSize = 800
)

plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)

## S4 method for signature 'CAGEexp'
plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
```

```
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)

## S4 method for signature 'SummarizedExperiment'
plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)

## S4 method for signature 'DataFrame'
plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)

## S4 method for signature 'data.frame'
plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)

## S4 method for signature 'matrix'
plotCorrelation2(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE,
  digits = 3
)
```

```
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE
)

## S4 method for signature 'CAGEexp'
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE
)

## S4 method for signature 'SummarizedExperiment'
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE
)

## S4 method for signature 'DataFrame'
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE
)

## S4 method for signature 'data.frame'
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
```

```
  applyThresholdBoth = FALSE
)

## S4 method for signature 'matrix'
correlationMatrix(
  object,
  what = c("CTSS", "consensusClusters"),
  values = c("raw", "normalized"),
  samples = "all",
  method = "pearson",
  tagCountThreshold = 1,
  applyThresholdBoth = FALSE
)
```

Arguments

object	A CAGEexp object or a SummarizedExperiment::SummarizedExperiment or an expression table as a S4Vectors::DataFrame, data.frame or matrix object. plotCorrelation() only supports CAGEexp input.
what	The clustering level to be used for plotting and calculating correlations: CTSS (individual TSSs) or consensusClusters. Ignored for anything else than CAGEexp and SummarizedExperiment objects.
values	Use either raw (default) or normalized CAGE signal. Ignored for plain expression tables.
samples	Character vector indicating which samples to use. Can be either all to select all samples in a CAGEexp object, or a subset of valid sample labels as returned by sampleLabels().
method	A character string indicating which correlation coefficient should be computed. Passed to cor(). Can be one of pearson, spearman, or kendall.
tagCountThreshold, applyThresholdBoth	Only TSSs with tag count >= tagCountThreshold in either one (applyThresholdBoth = FALSE) or both samples (applyThresholdBoth = TRUE) are plotted and used to calculate correlation.
plotSize	Size of the individual comparison plot in pixels - the total size of the resulting png will be length(samples) * plotSize in both dimensions. Ignored in plotCorrelation2.
digits	The number of significant digits for the data to be kept in log scale. Ignored in plotCorrelation. In plotCorrelation2, the number of points plotted is considerably reduced by rounding the point coordinates to a small number of significant digits before removing duplicates. Chose a value that makes the plot visually indistinguishable with non-deduplicated data, by making tests on a subset of the data.

Details

In the scatter plots, a pseudo-count equal to half the lowest score is added to the null values so that they can appear despite logarithmic scale.

SummarizedExperiment objects are expected to contain raw tag counts in a counts assay and the normalized expression scores in a normalized assay.

Avoid using large matrix objects as they are coerced to DataFrame class without special care for efficiency.

plotCorrelation2 speeds up the plotting by a) deduplicating that data: no point is plot twice at the same coordinates, b) rounding the data so that indistinguishable positions are plotted only once, c) using a black square glyph for the points, d) caching some calculations that are made repeatedly (to determine where to plot the correlation coefficients), and e) preventing coercion of DataFrames to data.frames.

Value

Displays the plot and returns a matrix of pairwise correlations between selected samples. The scatterplots of plotCorrelation are colored according to the density of points, and in plotCorrelation2 they are just black and white, which is much faster to plot. Note that while the scatterplots are on a logarithmic scale with pseudocount added to the zero values, the correlation coefficients are calculated on untransformed (but thresholded) data.

correlationMatrix() returns a matrix of pairwise correlations between selected samples.

Author(s)

Vanja Haberle

Charles Plessy

Katalin Ferenc

See Also

Other CAGEr plot functions: TSSlogo(), hanabiPlot(), plotAnnot(), plotExpressionProfiles(), plotInterquantileWidth(), plotReverseCumulatives()

Examples

```
plotCorrelation2(exampleCAGEexp, what = "consensusClusters", value = "normalized")

correlationMatrix(exampleCAGEexp, what = "consensusClusters", value = "normalized")
```

---

plotExpressionProfiles

Plot CAGE expression profiles

Description

Beanplot of distribution of normalized expression across CAGE experiments for individual *expression classes*, colored and labeled according to the information set when expression clustering was performed.

Usage

```
plotExpressionProfiles(object, what)

## S4 method for signature 'CAGEexp'
plotExpressionProfiles(object, what = c("CTSS", "consensusClusters"))
```

Arguments

object	A CAGEr object.
what	CTSS or consensusClusters.

Details

The beanplots are shown in one labeled box per *expression class*. Each beanplot represents one CAGE experiment. The vertical axis represents scaled normalized expression. The color of each class is determined by the labels returned by expression clustering.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr plot functions: TSSlogo(), hanabiPlot(), plotAnnot(), plotCorrelation(), plotInterquantileWidth(), plotReverseCumulatives()

Other CAGEr expression clustering functions: expressionClasses(), getExpressionProfiles()

Examples

```
plotExpressionProfiles(exampleCAGEexp, what = "CTSS")
exampleCAGEexp |> plotExpressionProfiles("consensusClusters")
```

---

plotInterquantileWidth

Plot cluster widths

Description

Histograms of the interquantile width of tag clusters or consensus clusters in each CAGE experiment.

Usage

```
plotInterquantileWidth(
  object,
  clusters = c("tagClusters", "consensusClusters"),
  tpmThreshold = 5,
  qLow = 0.1,
  qUp = 0.9,
  xlim = c(0, 150)
)

## S4 method for signature 'CAGEexp'
plotInterquantileWidth(
  object,
  clusters = c("tagClusters", "consensusClusters"),
```

```
  tpmThreshold = 5,
  qLow = 0.1,
  qUp = 0.9,
  xlim = c(0, 150)
)
```

Arguments

object	A CAGEexp object
clusters	tagClusters or consensusClusters.
tpmThreshold	Exclude clusters with normalized signal lower than tpmThreshold.
qLow, qUp	Quantile defining the 5' ("lower") and 3' ("upper") boundaries of the clusters.
xlim	Range of width to be plotted.

Details

Interquantile width is a more robust measure of the promoter width than the total span of the region, because it takes into account the magnitude of the expression in the region. Positions of specified quantiles within each cluster have to be calculated beforehand by calling quantilePositions.

Value

Plots the histograms with the ggplot2 engine and returns the plot object invisibly.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr plot functions: TSSlogo(), hanabiPlot(), plotAnnot(), plotCorrelation(), plotExpressionProfiles(), plotReverseCumulatives()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), consensusClustersGR(), cumulativeCTSSdistributio distclu(), paraclu(), quantilePositions(), tagClustersGR()

Examples

```
plotInterquantileWidth( exampleCAGEexp, clusters = "tagClusters"
                      , tpmThreshold = 50, qLow = 0.1, qUp = 0.9
                      , xlim = c(2,200))

plotInterquantileWidth( exampleCAGEexp, clusters = "consensusClusters"
                      , tpmThreshold = 50, qLow = 0.1, qUp = 0.9
                      , xlim = c(2,200))
```

plotReverseCumulatives *Plot reverse cumulative number of CAGE tags per CTSS*

Description

Plots the reverse cumulative distribution of the expression values of the CTSS for all CAGE datasets present in the CAGEexp object. The horizontal axis represents an expression value and the vertical axis represents the number of CTSS positions supported by >= of that value. The plot uses a log-log scale. Use these plots as help in choosing the parameters range of values and the referent slope for power-law normalization (Balwierz *et al*., 2009).

Usage

```
plotReverseCumulatives(
  object,
  values = c("raw", "normalized"),
  fitInRange = c(10, 1000),
  group = NULL
)

## S4 method for signature 'CAGEexp'
plotReverseCumulatives(
  object,
  values = c("raw", "normalized"),
  fitInRange = c(10, 1000),
  group = NULL
)

## S4 method for signature 'GRangesList'
plotReverseCumulatives(
  object,
  values = c("raw", "normalized"),
  fitInRange = c(10, 1000),
  group = NULL
)

## S4 method for signature 'GRanges'
plotReverseCumulatives(
  object,
  values = c("raw", "normalized"),
  fitInRange = c(10, 1000),
  group = NULL
)
```

Arguments

object	A CAGEexp object
values	Plot raw CAGE tag counts (default) or normalized values.

`fitInRange` An integer vector with two values specifying a range of tag count values to be used for fitting a power-law distribution to reverse cumulatives. Ignored is set to `NULL`. See Details.

`group` The name of a column data of the `CAGEexp` object, to be used to facet the plot. If `NULL` (default), all the distributions will be plotted together. Set to `sampleLabels` to plot each sample separately.

Details

A power law distribution is fitted to each reverse cumulative using the values in the range specified `fitInRange`. The fitted distribution is defined by

$$y = -1 * alpha * x + beta$$

on the log-log scale, and the value of *alpha* for each sample is shown on the plot's legend. In addition, a suggested referent power law distribution to which all samples could be normalized is drawn on the plot and corresponding parameters (slope *alpha* and total number of tags T) are denoted on the plot. This referent distribution is chosen so that its slope (*alpha*) is the median of slopes fitted to individual samples and its total number of tags (T) is the power of 10 nearest to the median number of tags of individual samples. Resulting plots are helpful in deciding whether power-law normalization is appropriate for given samples and reported `alpha` values aid in choosing optimal *alpha* value power law normalization (see `normalizeTagCount` for details).

Value

A `ggplot2::ggplot` object containing the plots. The plot can be further modified to change its title or axis labels (see `ggplot2::labs`). The legend can be removed with `ggplot2::guides(col=FALSE)`. The alpha parameter of the power law is saved in the `reference.slope` element of the `meta` slot of the object.

Author(s)

Vanja Haberle (original work)

Charles Plessy (port to ggplot2)

References

Balwierz *et al*. (2009) Methods for analyzing deep sequencing expression data: constructing the human and mouse promoterome with deepCAGE data, *Genome Biology* **10**(7):R79. https://doi.org/10.1186/gb-2009-10-7-r79

See Also

`normalizeTagCount`

Other CAGEr plot functions: `TSSlogo()`, `hanabiPlot()`, `plotAnnot()`, `plotCorrelation()`, `plotExpressionProfiles()`, `plotInterquantileWidth()`

Other CAGEr normalised data functions: `normalizeTagCount()`

Examples

```
exampleCAGEexp <- setColors(exampleCAGEexp,
  c("salmon", "darkkhaki", "darkturquoise", "blueviolet", "blueviolet"))
exampleCAGEexp$grp <- c("a", "b", "b", "c", "c")
```

```
plotReverseCumulatives( exampleCAGEexp, fitInRange = c(5,100))
plotReverseCumulatives( exampleCAGEexp, values = "normalized"
                      , fitInRange = c(200, 2000), group = "sampleLabels")
plotReverseCumulatives( exampleCAGEexp[,4:5], fitInRange = c(5,100)) +
  ggplot2::ggtitle("prim6 replicates")
tagClustersGR(exampleCAGEexp) |> plotReverseCumulatives()
```

`quantilePositions` *Determine CTSS quantile positions within clusters*

Description

Calculates the positions of "upper" and "lower" quantiles of CAGE signal along *tag clusters* or *consensus clusters* in each sample of a `CAGEexp` object.

Usage

```
quantilePositions(
  object,
  clusters = c("tagClusters", "consensusClusters"),
  qLow = 0.1,
  qUp = 0.9,
  useMulticore = FALSE,
  nrCores = NULL
)

## S4 method for signature 'CAGEexp'
quantilePositions(
  object,
  clusters = c("tagClusters", "consensusClusters"),
  qLow = 0.1,
  qUp = 0.9,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

`object`	A CAGEexp object.
`clusters`	Either `tagClusters` or `consensusClusters`.
`qLow`, `qUp`	Which "lower" or "upper" quantiles should be calculated. Numeric vector of values in range `[0,1]`.
`useMulticore`	Logical, should multicore be used. `useMulticore = TRUE` has only effect on Unix-like platforms.
`nrCores`	Number of cores to use when `useMulticore = TRUE`. Default value `NULL` uses all detected cores.

Details

From the 5' end the position, the position of a quantile q is determined as the first base in which of the cumulative expression is higher or equal to $q\%$ of the total CAGE signal of that cluster. Promoter *interquantile width* is defined as the distance (in base pairs) between a "lower" and an "upper" quantile position.

Value

Returns the objects, in which the positions of the quantiles are defined relatively to the start point of their cluster, for more efficient `Rle` compression. The quantile data for *tag clusters* are stored in the `TagClusters` objects directly. The quantile data for `consensus clusters` are stored in `integer` matrices named "q_*x*", where *x* represents the quantile (for instance, `q_0.1`), and these matrices are *assays* of the `consensusClusters` `RangedSummarizedExperiment`.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr object modifiers: `CTSStoGenes()`, `CustomConsensusClusters()`, `aggregateTagClusters()`, `annotateCTSS()`, `cumulativeCTSSdistribution()`, `distclu()`, `getCTSS()`, `normalizeTagCount()`, `paraclu()`, `quickEnhancers()`, `resetCAGEexp()`, `summariseChrExpr()`

Other CAGEr clusters functions: `CTSScumulativesTagClusters()`, `CustomConsensusClusters()`, `aggregateTagClusters()`, `consensusClustersDESeq2()`, `consensusClustersGR()`, `cumulativeCTSSdistributio`
`distclu()`, `paraclu()`, `plotInterquantileWidth()`, `tagClustersGR()`

Examples

```
quantilePositions(exampleCAGEexp, "tagClusters",     qLow = c(0.1, 0.2), qUp = c(0.8, 0.9))
tagClustersGR(exampleCAGEexp)
quantilePositions(exampleCAGEexp, "consensusClusters", qLow = c(0.1, 0.2), qUp = c(0.8, 0.9))
```

`quickEnhancers` *Identify and quantify enhancers.*

Description

A convenient wrapper to the function `CAGEfightR::quickEnhancers()`.

Usage

```
quickEnhancers(object)

## S4 method for signature 'CAGEexp'
quickEnhancers(object)
```

Arguments

`object`	A `CAGEexp` object

Details

The CAGEr object will be converted to a format similar to the output of CAGEfightR::quantifyCTSSs(), and then passed to the quickEnhancers function.

Value

A RangedSummarizedExperiment object. See the example below on how to attach it to the experiment list of a CAGEexp object.

Author(s)

Charles Plessy

Katalin Ferenc

See Also

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), distclu(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), resetCAGEexp(), summariseChrExpr()

Examples

```
# Can not run as long as the test data has nothing on the minus strand!
## Not run:
quickEnhancers(exampleCAGEexp)

## End(Not run)
```

---

ranges2annot *Hierarchical annotation of genomic regions.*

Description

Assigns region types such as promoter, exon or unknown to genomic regions such as *CTSS*, *tag clusters*, or *consensus clusters*.

Usage

```
ranges2annot(ranges, annot, upstream = 500, downstream = 500)
```

Arguments

ranges	A GenomicRanges::GRanges object, for example extracted from a RangedSummarizedExperiment object with the rowRanges command.
annot	A GRanges from which promoter positions will be inferred. Typically GENCODE. If the type metadata is present, it should contain gene, exon and transcript among its values. Otherwise, all entries are considered transcripts. If the transcript_type metadata is available, the entries that may not be primary products (for instance 'snoRNA') are discarded.

upstream — Number of bases *upstream* the start of the transcript models to be considered as part of the *promoter region*.

downstream — Number of bases *downstream* the start of the transcript models to be considered as part of the *promoter region*.

Details

Only the biotypes that are likely to have a pol II promoter will be filtered in. This is currently hardcoded in the function; see its source code. Example of biotypes without a pol II promoter: VDJ segments, miRNA, but also snoRNA, etc. Thus, the *Intergenic* category displayed in output of the plotAnnot may include counts overlaping with real exons of discarded transcribed regions: be careful that large percentages do not necessarly suggest abundance of novel promoters.

Value

A Run-length-encoded (Rle) factor of same length as the CTSS object, indicating if the interval is promoter, exon, intron or unknown, or just promoter, gene, unknown if the type metadata is absent.

Author(s)

Charles Plessy

See Also

CTSScoordinatesGR, exampleZv9_annot

Other CAGEr annotation functions: annotateCTSS(), plotAnnot(), ranges2genes(), ranges2names()

Examples

```
CAGEr:::ranges2annot(CTSScoordinatesGR(exampleCAGEexp), exampleZv9_annot)

ctss <- GenomicRanges::GRanges("chr1", IRanges::IPos(c(1,100,200,1500)), "+")
ctss <- GenomicRanges::GPos(ctss, stitch = FALSE)
ctss <- as(ctss, "CTSS")
gr1   <- GenomicRanges::GRanges( "chr1"
                                , IRanges::IRanges(c(650, 650, 1400), 2000), "+")
CAGEr:::ranges2annot(ctss, gr1)
gr2 <- gr1
gr2$type            <- c("transcript",     "exon",           "transcript")
gr2$transcript_type <- c("protein_coding", "protein_coding", "miRNA")
CAGEr:::ranges2annot(ctss, gr2, up=500, down=20)
```

---

ranges2genes *ranges2genes*

Description

Assign gene symbol(s) to Genomic Ranges.

Usage

```
ranges2genes(ranges, genes)
```

Arguments

ranges	GenomicRanges::GRanges object, for example extracted from a SummarizedExperiment::RangedS object with the SummarizedExperiment::rowRanges command.
genes	A *GRanges* object containing gene_name metadata.

Details

This private (non-exported) function is used to assign gene symbols to genomic ranges. It is run by annotateCTSS, which has to be run before CTSStoGenes.

Value

A S4Vectors::Rle factor of same length as the *GRanges* object, indicating one gene symbol or a semicolon-separated list of gene symbols for each range. The levels are alphabetically sorted.

Author(s)

Charles Plessy

See Also

CTSScoordinatesGR, exampleZv9_annot

Other CAGEr annotation functions: annotateCTSS(), plotAnnot(), ranges2annot(), ranges2names()

Other CAGEr gene expression analysis functions: CTSStoGenes(), GeneExpDESeq2()

Examples

```
CAGEr:::ranges2genes(CTSScoordinatesGR(exampleCAGEexp), exampleZv9_annot)
```

---

ranges2names *ranges2names*

Description

Intersection of genomic ranges

Usage

```
ranges2names(rangesA, rangesB)
```

Arguments

rangesA	A GenomicRanges::GRanges object.
rangesB	A second GRanges object.

Details

This private (non-exported) function intersects two genomic ranges and for each element of the first object returns the name of the elements of the second object that it intersects with.

Value

A S4Vectors::Rle factor of same length as the rangesA *GRanges* object, indicating one name or a semicolon-separated list of names from the each rangesB object. The levels are in order of appearance to to maintain genomic coordinate sort order when the names are cluster names.

Author(s)

Charles Plessy

See Also

Other CAGEr annotation functions: annotateCTSS(), plotAnnot(), ranges2annot(), ranges2genes()

Examples

```
names(exampleZv9_annot) <- exampleZv9_annot$gene_name
CAGEr:::ranges2names(CTSScoordinatesGR(exampleCAGEexp), exampleZv9_annot)
```

---

resetCAGEexp *Reset a CAGEexp object*

Description

Removes all data but the raw CTSS counts and coordinates from a CAGEexp object. Useful after removing samples.

Usage

```
resetCAGEexp(object)

## S4 method for signature 'CAGEexp'
resetCAGEexp(object)
```

Arguments

object	A CAGEexp object

Value

Returns a CAGEexp object, which contains a non-normalised tagCountMatrix experiment.

Author(s)

Charles Plessy

See Also

Other CAGEr object modifiers: `CTSStoGenes()`, `CustomConsensusClusters()`, `aggregateTagClusters()`, `annotateCTSS()`, `cumulativeCTSSdistribution()`, `distclu()`, `getCTSS()`, `normalizeTagCount()`, `paraclu()`, `quantilePositions()`, `quickEnhancers()`, `summariseChrExpr()`

Examples

```
resetCAGEexp(exampleCAGEexp)
```

---

rowsum.RleDataFrame *rowsum function for Rle DataFrames*

Description

Drop-in replacement for the `rowsum` function, which does not work natively on `S4Vectors::DataFrame` objects containing `S4Vectors::Rle`-encoded numerical values.

Usage

```
## S3 method for class 'RleDataFrame'
rowsum(x, group, reorder = TRUE, na.rm = FALSE, ...)
```

Arguments

`x`	A `DataFrame` containing only numerical `Rle` columns.
`group`	a vector or factor giving the grouping, with one element per row of `x`. Missing values will be treated as another group and a warning will be given.
`reorder`	If `TRUE`, then the result will be in order of `sort(unique(group))`, if `FALSE`, it will be in the order that groups were encountered.
`na.rm`	Logical (`TRUE` or `FALSE`). Should `NA` (including `NaN`) values be discarded?
`...`	Other arguments to be passed to or from methods.

Details

See the file `benchmarks/rowsum_on_Rle_DF.md` in the source Git repository of *CAGEr* for the alternatives that were considered.

Author(s)

Charles Plessy

See Also

Other Rle DataFrames: `rowSums.RleDataFrame()`

Examples

```
exampleCAGEexp |> CTSStagCountDF() |>
 CAGEr:::rowsum.RleDataFrame(decode(CTSScoordinatesGR(exampleCAGEexp)$cluster), reorder = FALSE)
```

rowSums.RleDataFrame *rowSums function for Rle DataFrames*

Description

Drop-in replacement for the rowSums function, which does not work natively on S4Vectors::DataFrame objects containing S4Vectors::Rle-encoded numerical values.

Usage

```
rowSums.RleDataFrame(x, na.rm = FALSE)
```

Arguments

x	A DataFrame containing only numerical Rle columns.
na.rm	logical. Should missing values (including NaN) be omitted from the calculations?

Details

See the file benchmarks/rowSums_on_Rle_DF.md in the source Git repository of *CAGEr* for the alternatives that were considered.

Value

A Rle-encoded numerical vector of the same class as in the DataFrame.

Author(s)

Charles Plessy

See Also

Other Rle DataFrames: rowsum.RleDataFrame()

Examples

```
exampleCAGEexp |> CTSStagCountDF() |> CAGEr:::rowSums.RleDataFrame(na.rm = TRUE)
```

sampleLabels *Get and set sample labels*

Description

sampleLabels gets or sets the labels and colors of CAGE datasets (samples) from CAGEr objects.

sampleList is an accessory function for convenience iteration in functions such as lapply or mapply. There is no set method for sampleList.

Usage

```
sampleLabels(object)

## S4 method for signature 'CAGEexp'
sampleLabels(object)

## S4 method for signature 'CTSS'
sampleLabels(object)

sampleList(object)

## S4 method for signature 'CAGEr'
sampleList(object)

sampleLabels(object) <- value

## S4 replacement method for signature 'CAGEexp'
sampleLabels(object) <- value

## S4 replacement method for signature 'CTSS'
sampleLabels(object) <- value
```

Arguments

object	A CAGEr object.
value	A character vector with a unique and valid name for each sample. The names attributes indicate the colors.

Details

In CAGEexp objects, renaming samples is possible only before data is loaded.

Value

sampleLabels returns a named character vector representing labels of all CAGE datasets present in the CAGEr object. The vector values are the labels and the vector names are the colors.

sampleList returns a named list where elements and their names are the sample names, for instance: list(sampleA = "sampleA", sampleB = "sampleB"). Thus, after iterating on it with lapply, the element names will be sample names.

Note

If no colors are supplied, then default colors will be assigned usign the rainbow function. Assigned colors are not guaranteed to be stable.

Author(s)

Vanja Haberle

Charles Plessy

See Also

setColors

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), seqNameTotalsSE(), tagClustersGR()

Other CAGEr setter methods: genomeName(), inputFiles(), inputFilesType(), setColors()

Examples

```
sampleLabels(exampleCAGEexp)

sampleList(exampleCAGEexp)
```

---

scoreShift *Calculate promoter shifting score*

Description

Calculates the shifting score for all consensus clusters (promoters) between two specified (groups of) CAGE datasets. Shifting score is a measure of differential usage of TSSs within consensus cluster between two samples, which indicates the degree of physical separation of TSSs used in these samples within given consensus cluster. In addition to shifting score, a statistical significance (P-value and FDR) of differential TSS usage is calculated for each consensus cluster using Kolmogorov-Smirnov test.

Usage

```
scoreShift(
  object,
  groupX,
  groupY,
  testKS = TRUE,
  useTpmKS = TRUE,
  useMulticore = F,
  nrCores = NULL
)

## S4 method for signature 'CAGEexp'
scoreShift(
  object,
  groupX,
  groupY,
  testKS = TRUE,
  useTpmKS = TRUE,
  useMulticore = F,
  nrCores = NULL
)
```

Arguments

object	A CAGEr object.
groupX, groupY	Character vector of the one or more CAGE dataset labels in the first (groupX) and in the second group (groupY). Shifting score for each consensus cluster will be calculated by comparing CAGE signal in the samples from groupX against the signal in the samples from groupY. If there is more than one CAGE dataset in the group, the datasets within that group will be merged together before comparison with the other group. See Details.
testKS	Logical, should Kolomogorov-Smirnov test for statistical significance of differential TSS usage be performed, and P-values and FDR returned. See Details.
useTpmKS	Logical, should normalized (tpm) values (TRUE) or raw tag counts (FALSE) be used to derive sample sizes for Kolomogorov-Smirnov test. Used only when testKS = TRUE, otherwise ignored. See Details.
useMulticore	Logical, should multicore be used. useMulticore = TRUE is supported only on Unix-like platforms.
nrCores	Number of cores to use when useMulticore = TRUE. Default value NULL uses all detected cores.

Details

TSSs within one consensus cluster (promoter) can be used differently in different samples (cell types, tissues, developmental stages), with respect to their position and frequency of usage detected by CAGE. This function calculates shifting scores of all consensus clusters between two specified (groups of) CAGE samples to detect promoters that are used differently in these two samples. Shifting score is a measure of differential TSS usage defined as:

`score = max(F1 - F2) / max(F1)`

where F1 is a cumulative sum of CAGE signal along consensus cluster in the group of samples with lower total signal in that consensus cluster, and F2 in the opposite group. Since cumulative sum can be calculated in both forward (5' -> 3') and reverse (3' -> 5') direction, shifting score is calculated for both cases and the bigger value is selected as final shifting score. Value of the shifting score is in the range `[-Inf, 1]`, where value of 1 means complete physical separation of TSSs used in the two samples for given consensus cluster. In general, any non-negative value of the shifting score can be interpreted as the proportion of transcription initiation in the sample with lower expression that is happening "outside" (either upstream or downstream) of the region used for transcription initiation in the other sample. Negative values indicate no physical separation, *i.e.* the region used for transcription initiation in the sample with lower expression is completely contained within the region used for transcription initiation in the other sample.

In addition to shifting score which indicates only physical separation (upstream or downstream shift of TSSs), a more general assessment of differential TSS usage can be obtained by performing a two-sample Kolmogorov-Smirnov test on cumulative sums of CAGE signal along the consensus cluster. In that case, cumulative sums in both samples are scaled to range `[0,1]` and are considered to be empirical cumulative distribution functions (ECDF) reflecting sampling of TSS positions during transcription initiation. Kolmogorov-Smirnov test is performed to assess whether the two underlying probability distributions differ. To obtain P-value (*i.e.* the level at which the null-hypothesis can be rejected), sample sizes that generated the ECDFs are required, in addition to actual K-S statistics calculated from ECDFs. These are derived either from raw tag counts, *i.e.* exact number of times each TSS in the cluster was sampled during sequencing (when `useTpmKS = FALSE`), or from normalized tpm values (when `useTpmKS = TRUE`). P-values obtained from K-S tests are further adjusted for multiple testing using Benjamini & Hochberg (BH) method and for each P-value a corresponding false-discovery rate (FDR) is also reported.

Since calculation of shifting scores and Kolmogorov-Smirnov test require cumulative sums along consensus clusters, they have to be calculated beforehand by calling cumulativeCTSSdistribution function.

The slots shiftingGroupX, shiftingGroupY and consensusClustersShiftingScores of the provided CAGEexp object will be occupied by the information on the groups of CAGE datasets that have been compared and shifting scores of all consensus clusters. Consensus clusters (promoters) with shifting score and/or FDR above specified threshold can be extracted by calling getShiftingPromoters function.

Author(s)

Vanja Haberle

Sarvesh Nikumbh

See Also

cumulativeCTSSdistribution

Other CAGEr promoter shift functions: getShiftingPromoters()

Examples

```
scoreShift( exampleCAGEexp
          , groupX = c("Zf.unfertilized.egg")
          , groupY = "Zf.30p.dome"
          , testKS = TRUE, useTpmKS = FALSE)
```

---

seqNameTotalsSE *Retreives the SummarizedExperiment containing chromosome expression totals.*

Description

Get or set a SummarizedExperiment summarising whole-chromosome expression levels in the experiment slot seqNameTotals and the sample metadata of the CAGEexp object.

Usage

```
seqNameTotalsSE(object)

## S4 method for signature 'CAGEexp'
seqNameTotalsSE(object)

seqNameTotalsSE(object) <- value
```

Arguments

object	A CAGEexp object.
value	A SummarizedExperiment object where rows represent reference sequences such as chromosomes.

Author(s)

Charles Plessy

See Also

summariseChrExpr

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), tagClustersGR()

Examples

```
seqNameTotalsSE(exampleCAGEexp)
```

---

setColors *Set colors for samples*

Description

Assigns one color to each sample in the CAGEr object. These colors are used in various plots and exported tracks to consistently represent corresponding samples.

Usage

```
setColors(object, colors = NULL)

## S4 method for signature 'CAGEr'
setColors(object, colors = NULL)
```

Arguments

object	A CAGEr object.
colors	A character vector of one valid R color specification per sample (see col2rgb for details). Provided colors are assigned to samples in the order they are returned by the sampleLabels function.

Value

Assigns one color to each sample in the CAGEr object and modifies it in place.

Author(s)

Vanja Haberle

See Also

Other CAGEr setter methods: genomeName(), inputFiles(), inputFilesType(), sampleLabels()

Examples

```
sampleLabels(exampleCAGEexp)
setColors(exampleCAGEexp, 5)
sampleLabels(exampleCAGEexp)
setColors(exampleCAGEexp, c("#ff0000ff", "#CCFF00", "blue", "grey", 1))
sampleLabels(exampleCAGEexp)
setColors(exampleCAGEexp, c("red", "darkgreen", "blue", "grey", "black"))
sampleLabels(exampleCAGEexp)
```

Strand invaders *Detect and remove strand invasion artefacts*

Description

`findStrandInvaders` detects strand invasion artefacts in the CTSS data. `removeStrandInvaders` removes them.

Strand invaders are artefacts produced by *template switching* reactions used in methods such as *nanoCAGE* and its derivatives (*C1 CAGE*, ...). They are described in details in Tang *et al.*, 2013. Briefly, these artefacts create CAGE-like signal downstream of genome sequences highly similar to the tail of template-switching oligonucleotides, which is `TATAGGG` in recent (2017) nanoCAGE protocols. Since these artefacts represent truncated cDNAs, they do not indicate promoter regions. It is therefore advisable to remove these artefacts. Moreover, when a sample barcode is near the linker sequence (which is not the case in recent nanoCAGE protocols), the strand-invasion artefacts can produce *sample-specific biases*, which can be confounded with biological effects depending on how the barcode sequences were chosen. A `barcode` parameter is provided to incorporate this information.

Usage

```
findStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")

removeStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")

## S4 method for signature 'CAGEexp'
findStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")

## S4 method for signature 'CAGEexp'
removeStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")

## S4 method for signature 'CTSS'
findStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")

## S4 method for signature 'CTSS'
removeStrandInvaders(object, distance = 1, barcode = NULL, linker = "TATAGGG")
```

Arguments

`object`	A `CAGEexp` object object containing CTSS data and the name of a reference genome.

distance	The maximal edit distance between the genome and linker sequences. Regardless this parameter, only a single mismatch is allowed in the last three bases of the linker.
barcode	A vector of sample barcode sequences, or the name of a column metadata of the CAGEexp object containing this information. (*Not implemented yet*)
linker	The sequence of the tail of the template-switching oligonucleotide, that will be matched with the genome sequence (defaults to TATAGGG).

Value

findStrandInvaders returns a logical-Rle vector indicating the position of the strand invaders in the input ranges.

With CTSS objects as input removeStrandInvaders returns the object after removing the CTSS positions identified as strand invaders. In the case of CAGEexp objects, a modified object is returned. Its sample metadata is also updated by creating a new strandInvaders column that indicates the number of molecule counts removed. This value is subtracted from the counts colum so that the total number of tags is still equal to librarySizes.

References

Tang *et al.*, "Suppression of artifacts and barcode bias in high-throughput transcriptome analyses utilizing template switching." *Nucleic Acids Res.* **2013** Feb 1;41(3):e44. PubMed ID: 23180801, DOI: 10.1093/nar/gks112

Examples

```
# Note that these examples do not do much on the example data since it was
# not constructed using a protocol based using the template-switching method.

findStrandInvaders(exampleCAGEexp)
removeStrandInvaders(exampleCAGEexp)
```

---

summariseChrExpr *Expression levels by chromosomes*

Description

Counts the number of molecules detected per chromosome, normalises by library size and stores the raw and normalised results in the CAGEr object.

Usage

```
summariseChrExpr(object)

## S4 method for signature 'CAGEexp'
summariseChrExpr(object)
```

Arguments

object	A CAGEexp object objects are not supported).

Value

Modifies the CAGEexp by adding a "seqNameTotals" experiment containing matrices where rows represent chromosomes and columns represent samples.

Author(s)

Charles Plessy

See Also

seqNameTotals

Other CAGEr object modifiers: CTSStoGenes(), CustomConsensusClusters(), aggregateTagClusters(), annotateCTSS(), cumulativeCTSSdistribution(), distclu(), getCTSS(), normalizeTagCount(), paraclu(), quantilePositions(), quickEnhancers(), resetCAGEexp()

Examples

```
summariseChrExpr(exampleCAGEexp)
```

TagClusters-class *TagClusters*

Description

TagClusters

Details

The TagClusters class represents tag clusters. It is used internally by CAGEr for type safety.

---

tagClustersGR *Extract tag clusters (TCs) for individual CAGE experiments*

Description

Extracts tag clusters (TCs) for a specified CAGE experiment from a CAGEexp object.

Usage

```
tagClustersGR(object, sample = NULL, qLow = NULL, qUp = NULL)

## S4 method for signature 'CAGEexp'
tagClustersGR(object, sample = NULL, qLow = NULL, qUp = NULL)

tagClustersGR(object, sample = NULL) <- value

## S4 replacement method for signature 'CAGEexp,ANY,TagClusters'
tagClustersGR(object, sample = NULL) <- value

## S4 replacement method for signature 'CAGEexp,missing,GRangesList'
tagClustersGR(object, sample = NULL) <- value
```

Arguments

object	A CAGEexp object.
sample	Label of the CAGE dataset (experiment, sample) for which to extract tag clusters. If samples = NULL, a list of all the clusters for each sample is returned.
qLow, qUp	Position of which quantile should be used as a left (lower) or right (upper) boundary (for qLow and qUp respectively) when calculating interquantile width. Default value NULL results in using the start coordinate of the cluster.
value	A TagClusters object.

Value

Returns a GRangesList or a TagClusters object with genomic coordinates, position of dominant TSS, total CAGE signal and additional information for all TCs from specified CAGE dataset (sample). If quantile information is provided, interquantile width for each TC is also calculated. The S4Vectors::metadata slot of the object contains a copy of the CAGEexp object's *column data*.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr accessor methods: CTSScoordinatesGR(), CTSScumulativesTagClusters(), CTSSnormalizedTpmDF(), CTSStagCountDF(), GeneExpDESeq2(), GeneExpSE(), consensusClustersGR(), expressionClasses(), filteredCTSSidx(), genomeName(), inputFiles(), inputFilesType(), librarySizes(), sampleLabels(), seqNameTotalsSE()

Other CAGEr clusters functions: CTSScumulativesTagClusters(), CustomConsensusClusters(), aggregateTagClusters(), consensusClustersDESeq2(), consensusClustersGR(), cumulativeCTSSdistributio, distclu(), paraclu(), plotInterquantileWidth(), quantilePositions()

Examples

```
tagClustersGR( exampleCAGEexp, "Zf.high", 0.1, 0.9 )
tagClustersGR( exampleCAGEexp, 1, qLow = 0.1, qUp = 0.9 )
tagClustersGR( exampleCAGEexp )@metadata$colData
```

TSSlogo *TSS logo*

Description

Plot the sequence logo of the region flanking the TSS. When this function is given *tag clusters* or *consensus clusters*, it uses the *dominant peak* as the transcription start site.

Usage

```
TSSlogo(x, upstream = 10, downstream = 10)

## S4 method for signature 'CAGEexp'
TSSlogo(x, upstream = 10, downstream = 10)

## S4 method for signature 'TagClusters'
TSSlogo(x, upstream = 10, downstream = 10)

## S4 method for signature 'ConsensusClusters'
TSSlogo(x, upstream = 10, downstream = 10)

## S4 method for signature 'CTSS'
TSSlogo(x, upstream = 10, downstream = 10)
```

Arguments

x	A CTSS, a TagClusters or a ConsensusClusters object.
upstream	Number of bases to plot upstream the TSS.
downstream	Number of bases to plot downstream the TSS, including the TSS itself.

Details

This function will only work if the CAGEexp object was built with a BSgenome package, as it needs to extract genomic sequences.

Value

A ggplot2::ggplot object showing the sequence logo. The coordinates displayed are negative for upstream sequences and positive downstream. The position of the TSS is set to 1.

Author(s)

Charles Plessy

See Also

Other CAGEr plot functions: hanabiPlot(), plotAnnot(), plotCorrelation(), plotExpressionProfiles(), plotInterquantileWidth(), plotReverseCumulatives()

Examples

```
TSSlogo(exampleCAGEexp|>consensusClustersGR(), 20, 10)
```

Index

∗ **CAGEfightR**
quickEnhancers, 91
∗ **CAGEr CTSS methods**
CTSStagCountDF, 24
∗ **CAGEr TSS functions**
TSSlogo, 106
∗ **CAGEr accessor methods**
consensusClustersGR, 17
CTSScoordinatesGR, 21
CTSScumulativesTagClusters, 22
CTSSnormalizedTpmDF, 23
CTSStagCountDF, 24
expressionClasses, 36
filteredCTSSidx, 38
GeneExpDESeq2, 42
GeneExpSE, 43
genomeName, 43
inputFiles, 63
inputFilesType, 64
librarySizes, 65
sampleLabels, 97
seqNameTotalsSE, 101
tagClustersGR, 105
∗ **CAGEr annotation functions**
annotateCTSS, 10
plotAnnot, 78
ranges2annot, 92
ranges2genes, 93
ranges2names, 94
∗ **CAGEr clustering methods**
consensusClustersTpm, 19
distclu, 29
paraclu, 74
∗ **CAGEr clusters functions**
aggregateTagClusters, 8
consensusClustersDESeq2, 16
consensusClustersGR, 17
CTSScumulativesTagClusters, 22
cumulativeCTSSdistribution, 26
CustomConsensusClusters, 28
distclu, 29
paraclu, 74
plotInterquantileWidth, 86
quantilePositions, 90
tagClustersGR, 105
∗ **CAGEr export functions**
exportToTrack, 33
∗ **CAGEr expression analysis functions**
consensusClustersDESeq2, 16
∗ **CAGEr expression clustering functions**
expressionClasses, 36
getExpressionProfiles, 47
plotExpressionProfiles, 85
∗ **CAGEr filter functions**
filteredCTSSidx, 38
flagByUpstreamSequences, 39
flagLowExpCTSS, 40
∗ **CAGEr gene expression analysis functions**
CTSStoGenes, 25
GeneExpDESeq2, 42
ranges2genes, 93
∗ **CAGEr normalised data functions**
normalizeTagCount, 72
plotReverseCumulatives, 88
∗ **CAGEr object modifiers**
aggregateTagClusters, 8
annotateCTSS, 10
CTSStoGenes, 25
cumulativeCTSSdistribution, 26
CustomConsensusClusters, 28
distclu, 29
getCTSS, 45
normalizeTagCount, 72
paraclu, 74
quantilePositions, 90
quickEnhancers, 91
resetCAGEexp, 95
summariseChrExpr, 104
∗ **CAGEr plot functions**
hanabiPlot, 54
plotAnnot, 78
plotCorrelation, 81
plotExpressionProfiles, 85
plotInterquantileWidth, 86
plotReverseCumulatives, 88
TSSlogo, 106

∗ **CAGEr promoter shift functions**
getShiftingPromoters, 49
scoreShift, 99
∗ **CAGEr richness functions**
hanabi, 51
hanabiPlot, 54
plot.hanabi, 77
∗ **CAGEr setter methods**
genomeName, 43
inputFiles, 63
inputFilesType, 64
sampleLabels, 97
setColors, 102
∗ **FANTOM data**
FANTOM5humanSamples, 37
FANTOM5mouseSamples, 38
importPublicData, 60
∗ **Rle DataFrames**
rowsum.RleDataFrame, 96
rowSums.RleDataFrame, 97
∗ **datasets**
exampleCAGEexp, 30
exampleZv9_annot, 31
FANTOM5humanSamples, 37
FANTOM5mouseSamples, 38
∗ **internal**
CAGEr-package, 4
∗ **loadFileIntoGPos**
bam2CTSS, 11
import.bam, 55
import.bam.ctss, 55
import.bedCTSS, 56
import.bedmolecule, 57
import.bedScore, 58
import.bigwig, 58
import.CTSS, 60
loadFileIntoGPos, 66
moleculesGR2CTSS, 71
.ConsensusClusters, *28*
.ConsensusClusters *(*ConsensusClusters-class*)*, 15
.TagClusters *(*TagClusters-class*)*, 105
.byCtss, 5
.byCtss,data.table-method *(*.byCtss*)*, 5
.ctss_summary_for_clusters, 5
.get.quant.pos, 6
.hanabi *(*hanabi-class*)*, 53
.powerLaw, 7

aggregateTagClusters, 8, *30*, *48*, *76*
aggregateTagClusters(), *11*, *17*, *18*, *23*, *26*, *27*, *29*, *30*, *47*, *73*, *76*, *87*, *91*, *92*, *96*, *105*, *106*
aggregateTagClusters,CAGEr-method *(*aggregateTagClusters*)*, 8
annotateConsensusClusters *(*annotateCTSS*)*, 10
annotateConsensusClusters,CAGEexp,GRanges-method *(*annotateCTSS*)*, 10
annotateConsensusClusters,CAGEexp,TxDb-method *(*annotateCTSS*)*, 10
annotateCTSS, 10, *94*
annotateCTSS(), *9*, *26*, *27*, *29*, *30*, *47*, *73*, *76*, *80*, *91–96*, *105*
annotateCTSS,CAGEexp,GRanges-method *(*annotateCTSS*)*, 10
annotateCTSS,CAGEexp,TxDb-method *(*annotateCTSS*)*, 10
annotateTagClusters *(*annotateCTSS*)*, 10
annotateTagClusters,CAGEexp,GRanges-method *(*annotateCTSS*)*, 10
annotateTagClusters,CAGEexp,TxDb-method *(*annotateCTSS*)*, 10

bam2CTSS, 11
bam2CTSS(), *55–60*, *67*, *72*
BSgenome, *107*

CAGEexp, *10*, *13*, *16*, *21*, *23–25*, *28*, *30*, *35*, *38*, *42–45*, *48*, *50*, *60*, *61*, *63–65*, *70*, *72*, *73*, *76*, *80*, *84*, *87*, *88*, *90*, *95*, *101*, *103*, *105*, *107*
CAGEexp *(*CAGEexp-class*)*, 12
CAGEexp-class, 12
CAGEfightR::quantifyCTSSs(), *92*
CAGEfightR::quickEnhancers(), *91*
CAGEr, *8*, *12*, *16*, *17*, *19*, *27*, *37*, *69–71*, *86*, *97*, *100*, *102*, *104*
CAGEr *(*CAGEr-package*)*, 4
CAGEr-class, 13
CAGEr-package, 4
CAGEr_Multicore, 14
coerce,CTSS,GRanges-method *(*CTSS-class*)*, 20
coerce,data.frame,CAGEexp-method *(*CAGEexp-class*)*, 12
coerce,GRanges,CTSS-method *(*CTSS-class*)*, 20
coerceInBSgenome, 15
col2rgb, *102*
ConsensusClusters, *5*, *6*, *18*, *39*, *80*, *107*
ConsensusClusters *(*ConsensusClusters-class*)*, 15
ConsensusClusters-class, 15
consensusClusters<-, 16
consensusClustersDESeq2, 16

consensusClustersDESeq2(), 9, 18, 23, 27, 29, 30, 76, 87, 91, 106
consensusClustersDESeq2,CAGEexp-method (consensusClustersDESeq2), 16
consensusClustersGR, 9, 17
consensusClustersGR(), 9, 17, 22–25, 27, 29, 30, 37, 38, 42–44, 64–66, 76, 87, 91, 99, 102, 106
consensusClustersGR,CAGEexp-method (consensusClustersGR), 17
consensusClustersGR<- (consensusClusters<-), 16
consensusClustersGR<-,CAGEexp-method (consensusClusters<-), 16
consensusClustersQuantile, 18
consensusClustersQuantile,CAGEexp-method (consensusClustersQuantile), 18
consensusClustersQuantileLow (consensusClustersQuantile), 18
consensusClustersQuantileLow,CAGEexp-method (consensusClustersQuantile), 18
consensusClustersQuantileLow<- (consensusClustersQuantile), 18
consensusClustersQuantileUp (consensusClustersQuantile), 18
consensusClustersQuantileUp,CAGEexp-method (consensusClustersQuantile), 18
consensusClustersQuantileUp<- (consensusClustersQuantile), 18
consensusClustersSE, 9, 20
consensusClustersSE (consensusClustersGR), 17
consensusClustersSE,CAGEexp-method (consensusClustersGR), 17
consensusClustersSE<- (consensusClusters<-), 16
consensusClustersSE<-,CAGEexp,RangedSummarizedExperiment-method (consensusClusters<-), 16
consensusClustersTpm, 19
consensusClustersTpm(), 30, 76
consensusClustersTpm,CAGEexp-method (consensusClustersTpm), 19
cor(), 84
correlationMatrix (plotCorrelation), 81
correlationMatrix,CAGEexp-method (plotCorrelation), 81
correlationMatrix,data.frame-method (plotCorrelation), 81
correlationMatrix,DataFrame-method (plotCorrelation), 81
correlationMatrix,matrix-method (plotCorrelation), 81
correlationMatrix,SummarizedExperiment-method (plotCorrelation), 81
CTSS, 5, 12, 29, 39, 46, 56, 57, 75, 80, 104, 107
CTSS (CTSS-class), 20
CTSS(), 22
CTSS-class, 20
CTSScoordinatesGR, 21, 93, 94
CTSScoordinatesGR(), 18, 23–25, 37, 38, 42–44, 64–66, 99, 102, 106
CTSScoordinatesGR,CAGEexp-method (CTSScoordinatesGR), 21
CTSScoordinatesGR<- (CTSScoordinatesGR), 21
CTSScoordinatesGR<-,CAGEexp-method (CTSScoordinatesGR), 21
CTSScumulativesCC (CTSScumulativesTagClusters), 22
CTSScumulativesCC,CAGEexp-method (CTSScumulativesTagClusters), 22
CTSScumulativesTagClusters, 22
CTSScumulativesTagClusters(), 9, 17, 18, 22, 24, 25, 27, 29, 30, 37, 38, 42–44, 64–66, 76, 87, 91, 99, 102, 106
CTSScumulativesTagClusters,CAGEexp-method (CTSScumulativesTagClusters), 22
CTSScumulativesTagClusters<- (CTSScumulativesTagClusters), 22
CTSScumulativesTagClusters<-,CAGEexp-method (CTSScumulativesTagClusters), 22
CTSSnormalizedTpmDF, 23, 73
CTSSnormalizedTpmDF(), 18, 22, 23, 25, 37, 38, 42–44, 64–66, 99, 102, 106
CTSSnormalizedTpmDF,CAGEexp-method (CTSSnormalizedTpmDF), 23
CTSSnormalizedTpmGR (CTSSnormalizedTpmDF), 23
CTSSnormalizedTpmGR,CAGEexp-method (CTSSnormalizedTpmDF), 23
CTSStagCountDF, 24, 46
CTSStagCountDF(), 18, 22–24, 37, 38, 42–44, 64–66, 99, 102, 106
CTSStagCountDF,CAGEexp-method (CTSStagCountDF), 24
CTSStagCountGR (CTSStagCountDF), 24
CTSStagCountGR,CAGEexp-method (CTSStagCountDF), 24
CTSStagCountSE (CTSStagCountDF), 24

CTSStagCountSE,CAGEexp-method (CTSStagCountDF), 24
CTSStagCountSE<- (CTSScoordinatesGR), 21
CTSStagCountSE<-,CAGEexp-method (CTSScoordinatesGR), 21
CTSStoGenes, *11*, 25, *94*
CTSStoGenes(), *9*, *11*, *27*, *29, 30*, *42*, *47*, *73*, *76*, *91, 92*, *94*, *96*, *105*
CTSStoGenes,CAGEexp-method (CTSStoGenes), 25
cumulativeCTSSdistribution, 26, *101*
cumulativeCTSSdistribution(), *9*, *11*, *17, 18*, *23*, *26*, *29, 30*, *47*, *73*, *76*, *87*, *91, 92*, *96*, *105, 106*
cumulativeCTSSdistribution,CAGEexp-method (cumulativeCTSSdistribution), 26
CustomConsensusClusters, 28
CustomConsensusClusters(), *9*, *11*, *17, 18*, *23*, *26, 27*, *30*, *47*, *73*, *76*, *87*, *91, 92*, *96*, *105, 106*
CustomConsensusClusters,CAGEexp,GRanges-method (CustomConsensusClusters), 28

data.frame, *84*
data.table::data.table, *5*
DataFrame, *12*, *25*
distclu, 29, *74*
distclu(), *9*, *11*, *17, 18*, *20*, *23*, *26, 27*, *29*, *47*, *73*, *76*, *87*, *91, 92*, *96*, *105, 106*
distclu,CAGEexp-method (distclu), 29
distclu,CTSS-method (distclu), 29
distclu,SummarizedExperiment-method (distclu), 29

exampleCAGEexp, 30
exampleZv9_annot, *11*, 31, *93, 94*
exportToTrack, 33
exportToTrack,CAGEexp-method (exportToTrack), 33
exportToTrack,ConsensusClusters-method (exportToTrack), 33
exportToTrack,CTSS-method (exportToTrack), 33
exportToTrack,GRanges-method (exportToTrack), 33
exportToTrack,GRangesList-method (exportToTrack), 33
exportToTrack,TagClusters-method (exportToTrack), 33
expressionClasses, 36, *49*
expressionClasses(), *18*, *22–25*, *38*, *42–44*, *49*, *64–66*, *86*, *99*, *102*, *106*
expressionClasses,ConsensusClusters-method (expressionClasses), 36
expressionClasses,CTSS-method (expressionClasses), 36

facet_wrap, *67*
FANTOM5humanSamples, 37, *38*, *62*
FANTOM5mouseSamples, *37*, 38, *62*
filteredCTSSidx, 38
filteredCTSSidx(), *9*, *18*, *22–25*, *37*, *40–44*, *64–66*, *99*, *102*, *106*
filteredCTSSidx,CAGEexp-method (filteredCTSSidx), 38
filterLowExpCTSS, *22*
filterLowExpCTSS (flagLowExpCTSS), 40
filterLowExpCTSS(), *8, 9*
filterLowExpCTSS,CAGEr-method (flagLowExpCTSS), 40
findStrandInvaders (Strand invaders), 103
findStrandInvaders,CAGEexp-method (Strand invaders), 103
findStrandInvaders,CTSS-method (Strand invaders), 103
flagByUpstreamSequences, 39
flagByUpstreamSequences(), *38*, *41*
flagByUpstreamSequences,ConsensusClusters-method (flagByUpstreamSequences), 39
flagByUpstreamSequences,CTSS-method (flagByUpstreamSequences), 39
flagByUpstreamSequences,GRanges-method (flagByUpstreamSequences), 39
flagByUpstreamSequences,TagClusters-method (flagByUpstreamSequences), 39
flagLowExpCTSS, 40
flagLowExpCTSS(), *38*, *40*
flagLowExpCTSS,CAGEr-method (flagLowExpCTSS), 40
flagLowExpCTSS,DataFrame-method (flagLowExpCTSS), 40
flagLowExpCTSS,matrix-method (flagLowExpCTSS), 40
flagLowExpCTSS,RangedSummarizedExperiment-method (flagLowExpCTSS), 40

GeneExpDESeq2, 42
GeneExpDESeq2(), *18*, *22–26*, *37, 38*, *43, 44*, *64–66*, *94*, *99*, *102*, *106*
GeneExpDESeq2,CAGEexp-method (GeneExpDESeq2), 42
GeneExpSE, 43
GeneExpSE(), *18*, *22–25*, *37, 38*, *42*, *44*, *64–66*, *99*, *102*, *106*

GeneExpSE,CAGEexp-method *(GeneExpSE)*, 43
genomeName, 43
genomeName(), *18*, *22–25*, *37, 38*, *42, 43*, *64–66*, *99*, *102*, *106*
genomeName,CAGEexp-method *(genomeName)*, 43
genomeName,CTSS-method *(genomeName)*, 43
genomeName<- *(genomeName)*, 43
genomeName<-,CAGEexp-method *(genomeName)*, 43
genomeName<-,CTSS-method *(genomeName)*, 43
GenomicRanges::GPos, *20*
GenomicRanges::GRanges, *12*, *39*, *92*, *94*
GenomicRanges::GRangesList, *80*
GenomicRanges::reduce, *29*
GenomicRanges::UnstitchedGPos, *20*
getCTSS, *22*, 45, *65, 66*
getCTSS(), *9*, *11*, *25–27*, *29, 30*, *73*, *76*, *91, 92*, *96*, *105*
getCTSS,CAGEexp-method *(getCTSS)*, 45
getExpressionProfiles, *35*, *37*, 47
getExpressionProfiles(), *37*, *86*
getExpressionProfiles,CAGEexp-method *(getExpressionProfiles)*, 47
getExpressionProfiles,matrix-method *(getExpressionProfiles)*, 47
getShiftingPromoters, 49, *101*
getShiftingPromoters(), *101*
getShiftingPromoters,CAGEexp-method *(getShiftingPromoters)*, 49
ggplot2::facet_wrap(), *80*
ggplot2::ggplot, *80*, *89*, *107*
ggplot2::guides, *89*
ggplot2::labs, *89*
GPos, *60*
GPos(), *67*
GRanges, *5*, *10*, *18*, *28*, *59*, *71*
GRangesList, *30*, *76*
gtools::mixedorder(), *67*

hanabi, 51
hanabi(), *54*, *78*
hanabi,GRanges-method *(hanabi)*, 51
hanabi,integer-method *(hanabi)*, 51
hanabi,List-method *(hanabi)*, 51
hanabi,list-method *(hanabi)*, 51
hanabi,matrix-method *(hanabi)*, 51
hanabi,numeric-method *(hanabi)*, 51
hanabi,Rle-method *(hanabi)*, 51
hanabi-class, 53
hanabiPlot, 54, *77*
hanabiPlot(), *53*, *78*, *80*, *85–87*, *89*, *107*

import.bam, 55
import.bam(), *12*, *56–60*, *67*, *72*
import.bam.ctss, 55
import.bam.ctss(), *12*, *55*, *57–60*, *67*, *72*
import.bedCTSS, 56
import.bedCTSS(), *12*, *55–60*, *67*, *72*
import.bedmolecule, 57
import.bedmolecule(), *12*, *55–60*, *67*, *72*
import.bedScore, 58
import.bedScore(), *12*, *55–57*, *59, 60*, *67*, *72*
import.bigwig, 58
import.bigwig(), *12*, *55–58*, *60*, *67*, *72*
import.CAGEscanMolecule, 59
import.CTSS, 60
import.CTSS(), *12*, *55–59*, *67*, *72*
importPublicData, 60
importPublicData(), *37, 38*
importPublicData,character,character,ANY,character-meth
(importPublicData), 60
initialize,CTSS-method *(CTSS-class)*, 20
inputFiles, 63
inputFiles(), *18*, *22–25*, *37, 38*, *42–44*, *65, 66*, *99*, *102*, *106*
inputFiles,CAGEexp-method *(inputFiles)*, 63
inputFiles<- *(inputFiles)*, 63
inputFiles<-,CAGEexp-method *(inputFiles)*, 63
inputFilesType, *45*, *47*, 64
inputFilesType(), *18*, *22–25*, *37, 38*, *42–44*, *64*, *66*, *99*, *102*, *106*
inputFilesType,CAGEexp-method *(inputFilesType)*, 64
inputFilesType<- *(inputFilesType)*, 64
inputFilesType<-,CAGEexp-method *(inputFilesType)*, 64
integer, *91*

lapply, *97*
librarySizes, *46, 47*, 65
librarySizes(), *18*, *22–25*, *37, 38*, *42–44*, *64, 65*, *99*, *102*, *106*
librarySizes,CAGEexp-method *(librarySizes)*, 65
lines.hanabi *(plot.hanabi)*, 77
loadFileIntoGPos, 66
loadFileIntoGPos(), *12*, *55–60*, *72*

make.names, *12*
make.names(), *12*
mapply, *97*
mapStats, 67, *80*
mapStatsScopes, *68*, 68, *80*

mapStatsScopes(), *67*
matrix, *84*
mergeCAGEsets, 69
mergeCAGEsets,CAGEexp,CAGEexp-method *(mergeCAGEsets)*, 69
mergeSamples, 70
mergeSamples,CAGEexp-method *(mergeSamples)*, 70
methods::coerce, *20*
methods::new, *20*
methods::show, *20*
moleculesGR2CTSS, 71
moleculesGR2CTSS(), *12*, *55–60*, *67*
msScope_all *(mapStatsScopes)*, 68
msScope_annotation *(mapStatsScopes)*, 68
msScope_counts *(mapStatsScopes)*, 68
msScope_mapped *(mapStatsScopes)*, 68
msScope_qc *(mapStatsScopes)*, 68
msScope_steps *(mapStatsScopes)*, 68
MultiAssayExperiment, *12*, *13*

normalizeTagCount, *24*, 72, *89*
normalizeTagCount(), *9*, *11*, *26*, *27*, *29*, *30*, *47*, *76*, *89*, *91*, *92*, *96*, *105*
normalizeTagCount,CAGEexp-method *(normalizeTagCount)*, 72

paraclu, 74
paraclu(), *9*, *11*, *17*, *18*, *20*, *23*, *26*, *27*, *29*, *30*, *47*, *73*, *87*, *91*, *92*, *96*, *105*, *106*
paraclu,CAGEexp-method *(paraclu)*, 74
paraclu,CTSS-method *(paraclu)*, 74
paraclu,GRanges-method *(paraclu)*, 74
paraclu,Pairs-method *(paraclu)*, 74
paraclu,SummarizedExperiment-method *(paraclu)*, 74
parseCAGEscanBlocksToGrangeTSS, 77
plot.hanabi, 77
plot.hanabi(), *53*, *54*
plotAnnot, *68*, 78, *93*
plotAnnot(), *11*, *54*, *85–87*, *89*, *93–95*, *107*
plotAnnot,CAGEexp-method *(plotAnnot)*, 78
plotAnnot,data.frame-method *(plotAnnot)*, 78
plotAnnot,DataFrame-method *(plotAnnot)*, 78
plotAnnot,GRangesList-method *(plotAnnot)*, 78
plotCorrelation, 81
plotCorrelation(), *54*, *80*, *84*, *86*, *87*, *89*, *107*
plotCorrelation,CAGEr-method *(plotCorrelation)*, 81
plotCorrelation2 *(plotCorrelation)*, 81
plotCorrelation2,CAGEexp-method *(plotCorrelation)*, 81
plotCorrelation2,data.frame-method *(plotCorrelation)*, 81
plotCorrelation2,DataFrame-method *(plotCorrelation)*, 81
plotCorrelation2,matrix-method *(plotCorrelation)*, 81
plotCorrelation2,SummarizedExperiment-method *(plotCorrelation)*, 81
plotExpressionProfiles, *35*, 85
plotExpressionProfiles(), *37*, *49*, *54*, *80*, *85*, *87*, *89*, *107*
plotExpressionProfiles,CAGEexp-method *(plotExpressionProfiles)*, 85
plotInterquantileWidth, 86
plotInterquantileWidth(), *9*, *17*, *18*, *23*, *27*, *29*, *30*, *54*, *76*, *80*, *85*, *86*, *89*, *91*, *106*, *107*
plotInterquantileWidth,CAGEexp-method *(plotInterquantileWidth)*, 86
plotReverseCumulatives, *73*, 88
plotReverseCumulatives(), *54*, *73*, *80*, *85–87*, *107*
plotReverseCumulatives,CAGEexp-method *(plotReverseCumulatives)*, 88
plotReverseCumulatives,GRanges-method *(plotReverseCumulatives)*, 88
plotReverseCumulatives,GRangesList-method *(plotReverseCumulatives)*, 88
points.hanabi *(plot.hanabi)*, 77

quantilePositions, *87*, 90
quantilePositions(), *9*, *11*, *17*, *18*, *23*, *26*, *27*, *29*, *30*, *47*, *73*, *76*, *87*, *92*, *96*, *105*, *106*
quantilePositions,CAGEexp-method *(quantilePositions)*, 90
quickEnhancers, 91
quickEnhancers(), *9*, *11*, *26*, *27*, *29*, *30*, *47*, *73*, *76*, *91*, *96*, *105*
quickEnhancers,CAGEexp-method *(quickEnhancers)*, 91

RangedSummarizedExperiment, *9*, *25*, *28*, *46*, *76*, *91*
ranges2annot, 92
ranges2annot(), *11*, *80*, *94*, *95*
ranges2genes, 93
ranges2genes(), *11*, *26*, *42*, *80*, *93*, *95*
ranges2names, 94
ranges2names(), *11*, *80*, *93*, *94*

removeStrandInvaders (Strand invaders), 103
removeStrandInvaders,CAGEexp-method (Strand invaders), 103
removeStrandInvaders,CTSS-method (Strand invaders), 103
resetCAGEexp, 95
resetCAGEexp(), *9*, *11*, *26, 27*, *29, 30*, *47*, *73*, *76*, *91, 92*, *105*
resetCAGEexp,CAGEexp-method (resetCAGEexp), 95
Rle, *12*, *25*, *37*, *41*, *93*, *104*
rowRanges, *92*
rowsum.RleDataFrame, 96
rowsum.RleDataFrame(), *97*
rowSums.RleDataFrame, 97
rowSums.RleDataFrame(), *96*

S4Vectors::DataFrame, *84*, *96, 97*
S4Vectors::metadata, *106*
S4Vectors::Pairs, *75*
S4Vectors::Rle, *94–97*
sampleLabels, 97, *102*
sampleLabels(), *18*, *22–25*, *37, 38*, *42–44*, *64–66*, *84*, *102*, *106*
sampleLabels,CAGEexp-method (sampleLabels), 97
sampleLabels,CTSS-method (sampleLabels), 97
sampleLabels<- (sampleLabels), 97
sampleLabels<-,CAGEexp-method (sampleLabels), 97
sampleLabels<-,CTSS-method (sampleLabels), 97
sampleList (sampleLabels), 97
sampleList,CAGEr-method (sampleLabels), 97
scoreShift, *49*, 99
scoreShift(), *50*
scoreShift,CAGEexp-method (scoreShift), 99
seqNameTotalsSE, 101
seqNameTotalsSE(), *18*, *22–25*, *37, 38*, *42–44*, *64–66*, *99*, *106*
seqNameTotalsSE,CAGEexp-method (seqNameTotalsSE), 101
seqNameTotalsSE<- (seqNameTotalsSE), 101
setColors, *99*, 102
setColors(), *44*, *64, 65*, *99*
setColors,CAGEr-method (setColors), 102
show,CTSS-method (CTSS-class), 20
som::som, *48*
stats::kmeans, *48*
Strand invaders, 103
summariseChrExpr, 104
summariseChrExpr(), *9*, *11*, *26, 27*, *29, 30*, *47*, *73*, *76*, *91, 92*, *96*
summariseChrExpr,CAGEexp-method (summariseChrExpr), 104
SummarizedExperiment, *18*, *26*
SummarizedExperiment::RangedSummarizedExperiment, *29*, *94*
SummarizedExperiment::rowRanges, *94*
SummarizedExperiment::SummarizedExperiment, *84*

TagClusters, *5, 6*, *30*, *39*, *76*, *80*, *106, 107*
TagClusters (TagClusters-class), 105
TagClusters-class, 105
tagClustersGR, 105
tagClustersGR(), *9*, *17, 18*, *22–25*, *27*, *29, 30*, *37, 38*, *42–44*, *64–66*, *76*, *87*, *91*, *99*, *102*
tagClustersGR,CAGEexp-method (tagClustersGR), 105
tagClustersGR<- (tagClustersGR), 105
tagClustersGR<-,CAGEexp,ANY,TagClusters-method (tagClustersGR), 105
tagClustersGR<-,CAGEexp,missing,GRangesList-method (tagClustersGR), 105
TSSlogo, 106
TSSlogo(), *54*, *80*, *85–87*, *89*
TSSlogo,CAGEexp-method (TSSlogo), 106
TSSlogo,ConsensusClusters-method (TSSlogo), 106
TSSlogo,CTSS-method (TSSlogo), 106
TSSlogo,TagClusters-method (TSSlogo), 106
TxDb, *10*