Package 'CAGEr'

October 14, 2021

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

Version 1.34.0

Date 2021-15-19

Imports beanplot, BiocGenerics, BiocParallel, BSgenome, data.table, DelayedArray, formula.tools, GenomeInfoDb, GenomicAlignments, GenomicRanges (>= 1.37.16), ggplot2 (>= 2.2.0), gtools, IRanges (>= 2.18.0), KernSmooth, memoise, plyr, Rsamtools, reshape, rtracklayer, S4Vectors (>= 0.27.5), som, stringdist, stringi, SummarizedExperiment, utils, vegan, VGAM

Depends methods, MultiAssayExperiment, R (>= 3.5.0)

Suggests BSgenome.Drerio.UCSC.danRer7, DESeq2, FANTOM3and4CAGE, BiocStyle, knitr, rmarkdown

Description Preprocessing of CAGE sequencing data, identification and normalization of transcription start sites and downstream analysis of transcription start sites clusters (promoters).

License GPL-3

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

Collate 'CTSS.R' 'CAGEexp.R' 'AllClasses.R' 'CAGEr.R' 'AggregationFunctions.R' 'Multicore.R' 'ClusteringFunctions.R' 'ClusteringMethods.R' 'Annotations.R' 'AggregationMethods.R' 'CorrelationMethods.R' 'CumulativeDistributionFunctions.R' 'GetMethods.R' 'CumulativeDistributionMethods.R' 'ExportFunctions.R' 'ExportMethods.R' 'ExpressionProfilingMethods.R' 'ImportFunctions.R' 'SetMethods.R' 'ImportMethods.R' 'MergingMethods.R' 'NormalizationFunctions.R' 'NormalizationMethods.R' 'QCmethods.R' 'QuantileWidthFunctions.R' 'QuantileWidthMethods.R' 'Richness.R' 'ShiftingFunctions.R' 'ShiftingMethods.R' 'StrandInvaders.R'

LazyData true

VignetteBuilder knitr

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

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

git_branch RELEASE_3_13

git_last_commit 8e7ccda

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

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

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

R topics documented:

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

Description

The *CAGEr* package performs identification of transcription start sites and frequency of their usage from input CAGE sequencing data, normalization of raw CAGE tag count, clustering of TSSs into tag clusters (TC) and their aggregation across multiple CAGE experiments to construct the promoterome. It manipulates multiple CAGE experiments at once, performs expression profiling across experiments both at level of individual TSSs and clusters of TSSs, exports several different types of track files for visualization in the UCSC Genome Browser, performs analysis of promoter width and detects differential usage of TSSs (promoter shifting) between samples. Multicore option for parallel processing is supported on Unix-like platforms.

Author(s)

Vanja Haberle

.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 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)
```

.clusterAggregateAndSum *Aggregate identical clusters and sum their scores.*

Description

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

```
.clusterAggregateAndSum(clusters, key)

## S4 method for signature 'data.table'
.clusterAggregateAndSum(clusters, key)

## S4 method for signature 'data.frame'
.clusterAggregateAndSum(clusters, key)

## S4 method for signature 'GRanges'
.clusterAggregateAndSum(clusters, key)
```

Arguments

clusters	Clusters to be aggregated. data.frame, or GRanges, which will be coerced to data.frame.
key	Name of the column containing the factor used to aggregate the clusters.

.get.quant.pos *Get quantile positions*

Description

Function that calculates position of quantiles for CTSS clusters based on distribution of tpm within the cluster

Usage

```
.get.quant.pos(cumsums, clusters, q)
```

Arguments

cumsums	Named list of vectors containing cumulative sum for each cluster (returned by 'get.cumsum' function).
clusters	GRanges 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

```
## Not run:  #because it runs through quantilePositions() anyway.
cumsums  <- CTSScumulativesTagClusters(object, 1)
clusters <- tagClustersGR(object, 1)
.get.quant.pos(cumsums, clusters, c(.1, .9))

## End(Not run)
```

`.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.
`excludeSignalBelowThreshold`	When `TRUE` the tag clusters with normalized signal < `tpmThreshold` will not contribute to the total CAGE signal of a consensus cluster. When set to `FALSE` all TCs that overlap consensus clusters will contribute to the total signal, regardless whether they pass the threshold for constructing the clusters or not.
`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 `clusterCTSS` 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.

Value

For CAGEset objects, the consensusClusters slot will be populated with a data frame indicating the cluster name, chromosome, start and end coordinates, the strand, and the normalised expression score of the cluster. This table is returned by the consensusClusters function.

For CAGEexp objects, the experiment consensusClusters will be occupied by a RangedSummarizedExperiment containing the cluster coodinates as row ranges, and their expression levels in the counts and normalized assays. These genomic ranges are returned by the consensusClustersGR 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. This table is returned by the consensusClusters function.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr object modifiers: CTSStoGenes, CustomConsensusClusters, annotateCTSS, clusterCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, quantilePositions, summariseChrExpr

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribution, plotInterquantileWidth, quantilePositions, tagClusters

Examples

```
head(consensusClusters(exampleCAGEset))
aggregateTagClusters( exampleCAGEset, tpmThreshold = 50
                    , excludeSignalBelowThreshold = FALSE, maxDist = 100)
head(consensusClusters(exampleCAGEset))

aggregateTagClusters(object = exampleCAGEset, tpmThreshold = 50,
  excludeSignalBelowThreshold = FALSE, qLow = 0.1, qUp = 0.9, maxDist = 100)
head(consensusClusters(exampleCAGEset))

consensusClustersGR(exampleCAGEexp)
aggregateTagClusters( exampleCAGEexp, tpmThreshold = 50
                    , excludeSignalBelowThreshold = FALSE, maxDist = 100)
consensusClustersGR(exampleCAGEexp)

aggregateTagClusters( exampleCAGEexp, tpmThreshold = 50
                    , excludeSignalBelowThreshold = TRUE, maxDist = 100)
consensusClustersGR(exampleCAGEexp)
```

```
aggregateTagClusters( exampleCAGEexp, tpmThreshold = 50
                    , excludeSignalBelowThreshold = TRUE, maxDist = 100
                    , qLow = 0.1, qUp = 0.9)
consensusClustersGR(exampleCAGEexp)
```

---

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, ranges)

## S4 method for signature 'CAGEset,ANY'
annotateCTSS(object, ranges)

## S4 method for signature 'CAGEexp,GRanges'
annotateCTSS(object, ranges)

annotateConsensusClusters(object, ranges)

## S4 method for signature 'CAGEset,ANY'
annotateConsensusClusters(object, ranges)

## S4 method for signature 'CAGEexp,GRanges'
annotateConsensusClusters(object, ranges)
```

Arguments

object	CAGEexp object (CAGEsets are not supported).
ranges	A GRanges object, optionally containing gene_name, type and transcript_type metadata.

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.

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, clusterCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, quantilePositions, summariseChrExpr

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

Examples

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

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

---

bam2CTSS *bam2CTSS*

Description

Converts from BAM to CTSS

Usage

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

Arguments

gr	A 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.ctss, import.bam, import.bedCTSS, import.bedScore, import.bedmolecule, 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 is a replacement for the CAGEset class. 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

CAGEset, 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.

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$l1 <- colSums(CTSStagCountDf(exampleCAGEexp) > 0)
exampleCAGEexp$l1
```

`CAGEr-class` *CAGEr objects*

Description

The *CAGEr* package provides two classes of objects to load, contain and process CAGE data:

Details

- The `CAGEset` class is the original object format in *CAGEr*, as when published in Haberle *et al.*, 2015.

- The CAGEexp class is a new class format in 2017, which is based on the MultiAssayExperiment class. In comparison with CAGEset, objects, 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.

Most *CAGEr* functions support both classes interchangabely, and the CAGEr class was created for methods that support both classes identically.

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, 3)
CAGEr:::CAGEr_Multicore(FALSE, 3)
```

CAGEset-class *Class* "CAGEset"

Description

This class is used to store one or more CAGE (Cap Analysis of Gene Expression) datasets in the form of TSSs derived from CAGE tags and frequency of their usage, and to store and extract all information generated during the workflow.

Objects from the Class

Objects can be created by calls of the form `new("CAGEset",...)`.
Objects of the class contain information on the genomic coordinates of TSSs derived from sequenced CAGE tags from multiple experiments, number of tags supporting each TSS in each experiment, normalized CAGE signal at each TSS, all information on specified parameters and results of all downstream analyses. Object has to be created before reading in the data by specifying input files and their type, referent genome and labels for individual CAGE datasets, as described in the vignette. Data is read by applying a function to a created object and all further slots are filled during the workflow by applying specific functions.

Slots

`genomeName`**:** Object of class `"character"`: the name of the BSgenome package used as the referent genome

`inputFiles`**:** Object of class `"character"`: the paths to input files

`inputFilesType`**:** Object of class `"character"`: the type of input files (e.g. bam)

`sampleLabels`**:** Object of class `"character"`: the labels of individual CAGE experiments

`librarySizes`**:** Object of class `"integer"`: the total number of CAGE tags per experiment

`CTSScoordinates`**:** Object of class `"data.frame"`: the genomic coordinates of CAGE transcription start sites (CTSSs)

`tagCountMatrix`**:** Object of class `"data.frame"`: the number of CAGE tags supporting every CTSS in each experiment

`normalizedTpmMatrix`**:** Object of class `"data.frame"`: the normalized CAGE signal supporting every CTSS in each experiment

`CTSSexpressionClusteringMethod`**:** Object of class `"character"`: the method used for expression clustering of CTSSs

`CTSSexpressionClasses`**:** Object of class `"character"`: the labels of expression classes of CTSSs returned by expression clustering

`clusteringMethod`**:** Object of class `"character"`: the method used for clustering CTSSs into tag clusters (TC)

`filteredCTSSidx`**:** Object of class `"logical"`: the index of CTSSs included in tag clusters

`tagClusters`**:** Object of class `"list"`: the list of tag clusters per CAGE experiment

`CTSScumulativesTagClusters`: Object of class `"list"`: the cumulative distribution of CAGE signal along TCs

`tagClustersQuantileLow`: Object of class `"list"`: the positions of lower quantile(s) within TCs

`tagClustersQuantileUp`: Object of class `"list"`: the positions of upper quantile(s) within TCs

`consensusClusters`: Object of class `"data.frame"`: the genomic coordinates of consensus clusters created by aggregating TCs across experiments

`consensusClustersTpmMatrix`: Object of class `"matrix"` the normalized CAGE signal for every consensus cluster in each experiment

`consensusClustersExpressionClusteringMethod`: Object of class `"character"` the method used for expression clustering of consensus clusters

`consensusClustersExpressionClasses`: Object of class `"character"` the labels of expression classes of consensus clusters returned by expression clustering

`CTSScumulativesConsensusClusters`: Object of class `"list"` the cumulative distribution of CAGE signal along consensus clusters

`consensusClustersQuantileLow`: Object of class `"list"` the positions of lower quantile(s) within consensus clusters

`consensusClustersQuantileUp`: Object of class `"list"` the positions of upper quantile(s) within consensus clusters

`shiftingGroupX`: Object of class `"character"` the label(s) of experiment(s) in the first shifting group

`shiftingGroupY`: Object of class `"character"` the label(s) of experiment(s) in the second shifting group

`consensusClustersShiftingScores`: Object of class `"data.frame"` the shifting scores and P-values/FDR for comparison of consensus clusters between two (groups of) experiments

Methods

CTSSclusteringMethod `signature(object = "CAGEset")`: extracts the method used for clustering CTSSs into tag clusters (TC)

CTSScoordinates `signature(object = "CAGEset")`: extracts the genomic coordinates of all CTSSs

CTSSnormalizedTpm `signature(object = "CAGEset")`: extracts the normalized CAGE signal supporting every CTSS in each experiment

CTSStagCount `signature(object = "CAGEset")`: extracts the number of CAGE tags supporting every CTSS in each experiment

aggregateTagClusters `signature(object = "CAGEset")`: aggregates TCs across all experiments into consensus clusters

clusterCTSS `signature(object = "CAGEset")`: clusters CTSSs into TCs per experiment

consensusClusters `signature(object = "CAGEset")`: extracts the genomic coordinates and other information on consensus clusters

consensusClustersTpm `signature(object = "CAGEset")`: extracts the matrix with tpm values for consensus clusters across all samples

cumulativeCTSSdistribution `signature(object = "CAGEset")`: calculates the cumulative distribution of CAGE signal along TCs or consensus clusters

exportCTSStoBedGraph `signature(object = "CAGEset")`: creates bedGraph files of CTSSs for visualization in the UCSC Genome Browser

exportToBed `signature(object = "CAGEset")`: creates various types of BED files for visualization in the UCSC Genome Browser

expressionClasses `signature(object = "CAGEset")`: extracts the labels of the expression classes of CTSSs or consensus clusters returned from expression profiling

extractExpressionClass `signature(object = "CAGEset")`: extracts CTSSs or consensus clusters belonging to specified expression class

genomeName `signature(object = "CAGEset")`: extracts the name of the BSgenome package used as the referent genome

getCTSS `signature(object = "CAGEset")`: reads in specified input files and fills in information on detected CTSSs and their tag count

getExpressionProfiles `signature(object = "CAGEset")`: performs expression clustering of CTSSs or consensus clusters across experiments

getShiftingPromoters `signature(object = "CAGEset")`: extracts consensus clusters with shifting score and/or FDR above specified threshold

inputFiles `signature(object = "CAGEset")`: extracts the paths of input CAGE data files

inputFilesType `signature(object = "CAGEset")`: extracts the type of input CAGE data files

librarySizes `signature(object = "CAGEset")`: extracts the library sizes of individual CAGE experiments within CAGEset object

mergeSamples `signature(object = "CAGEset",mergeIndex = "numeric")`: merges specified experiments (samples) into one (e.g. replicas)

normalizeTagCount `signature(object = "CAGEset")`: normalizes raw CAGE tag count

plotCorrelation `signature(object = "CAGEset")`: plots pairwise scatter plots and calculates correlation between samples

plotExpressionProfiles `signature(object = "CAGEset")`: creates file with beanplots of expression across experiments for CTSSs or consensus clusters belonging to different expression classes

plotInterquantileWidth `signature(object = "CAGEset")`: creates file with histograms of interquantile width

plotReverseCumulatives `signature(object = "CAGEset")`: creates file with reverse cumulative plots of CAGE tag count per CTSS

quantilePositions `signature(object = "CAGEset")`: calculates the positions of specified quantiles within TCs or consensus clusters

sampleLabels `signature(object = "CAGEset")`: extracts the labels of individual CAGE experiments within CAGEset object

scoreShift `signature(object = "CAGEset",groupX = "character",groupY = "character")`: calculates the shifting score and tests the statistical significance of differential TSS usage for consensus clusters between two specified (groups of) samples

setColors `signature(object = "CAGEset")`: assigns color to each sample to be used in visualisation

show `signature(object = "CAGEset")`: displays CAGEset object in a user friendly way

tagClusters `signature(object = "CAGEset")`: extracts the tag clusters for specified CAGE experiment

Coercion

`as(from,"CAGEset")`: Creates a CAGEset object from a data.frame object.

Author(s)

Vanja Haberle

Examples

```
showClass("CAGEset")
```

---

clusterCTSS *Cluster CTSSs into tag clusters*

Description

Clusters individual CAGE transcription start sites (CTSSs) along the genome into tag clusters (TCs) using specified *ab initio* method, or assigns them to predefined genomic regions.

Usage

```
clusterCTSS(object, threshold = 1, nrPassThreshold = 1,
  thresholdIsTpm = TRUE, method = c("distclu", "paraclu", "custom"),
  maxDist = 20, removeSingletons = FALSE, keepSingletonsAbove = Inf,
  minStability = 1, maxLength = 500, reduceToNonoverlapping = TRUE,
  customClusters = NULL, useMulticore = FALSE, nrCores = NULL)

## S4 method for signature 'CAGEset'
clusterCTSS(object, threshold = 1,
  nrPassThreshold = 1, thresholdIsTpm = TRUE, method = c("distclu",
  "paraclu", "custom"), maxDist = 20, removeSingletons = FALSE,
  keepSingletonsAbove = Inf, minStability = 1, maxLength = 500,
  reduceToNonoverlapping = TRUE, customClusters = NULL,
  useMulticore = FALSE, nrCores = NULL)

## S4 method for signature 'CAGEexp'
clusterCTSS(object, threshold = 1,
  nrPassThreshold = 1, thresholdIsTpm = TRUE, method = c("distclu",
  "paraclu", "custom"), maxDist = 20, removeSingletons = FALSE,
  keepSingletonsAbove = Inf, minStability = 1, maxLength = 500,
  reduceToNonoverlapping = TRUE, customClusters = NULL,
  useMulticore = FALSE, nrCores = NULL)
```

Arguments

object	A CAGEr object.
threshold, nrPassThreshold	Ignore CTSSs with signal < threshold in < nrPassThreshold experiments.
thresholdIsTpm	Logical indicating if threshold is expressed in raw tag counts (FALSE) or normalized signal (TRUE).
method	Method to be used for clustering: "distclu", "paraclu" or "custom". See Details.
maxDist	Maximal distance between two neighbouring CTSSs for them to be part of the same cluster. Used only when method = "distclu", otherwise ignored.
removeSingletons	Logical indicating if tag clusters containing only one CTSS be removed. Ignored when method = "custom".
keepSingletonsAbove	Controls which singleton tag clusters will be removed. When removeSingletons = TRUE, only singletons with signal < keepSingletonsAbove will be removed. Useful to prevent removing highly supported singleton tag clusters. Default value Inf results in removing all singleton TCs when removeSingletons = TRUE. Ignored when method = "custom".
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. Used only when method = "paraclu".
maxLength	Maximal length of cluster in base-pairs. Clusters with length > maxLength will be discarded. Ignored when method = "custom".
reduceToNonoverlapping	Logical, should smaller clusters contained within bigger cluster be removed to make a final set of tag clusters non-overlapping. Used only method = "paraclu".
customClusters	Genomic coordinates of predefined regions to be used to segment the CTSSs. The format is either a GRanges object or a data.frame with the following columns: chr (chromosome name), start (0-based start coordinate), end (end coordinate), strand (either "+", or "-"). Used only when method = "custom".
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

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 distclu-functions for implementation details.

"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 (removeSingletons, keepSingletonsAbove, minStability,

maxLength and reduceToNonoverlapping) can be specified to simplify the output by discarding too small (singletons) or too big clusters, and to reduce the clusters to a final set of non-overlapping clusters.

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

The slots clusteringMethod, filteredCTSSidx and tagClusters of the provided CAGEset object will be occupied by the information on method used for clustering, CTSSs included in the clusters and list of tag clusters per CAGE experiment, respectively. To retrieve tag clusters for individual CAGE dataset use the tagClusters() function.

In CAGEexp objects, the results will be stored as a GRangesList of TagClusters objects in the metadata slot tagClusters. The TagClusters objects will contain a filteredCTSSidx column if appropriate. The clustering method name is saved in the metadata slot of the GRangesList.

Author(s)

Vanja Haberle

References

Frith *et al.* (2007) A code for transcription initiation in mammalian genomes, *Genome Research* **18**(1):1-12, (http://www.cbrc.jp/paraclu/).

See Also

tagClusters, aggregateTagClusters and CTSSclusteringMethod.

Other CAGEr object modifiers: CTSStoGenes, CustomConsensusClusters, aggregateTagClusters, annotateCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, quantilePositions, summariseChrExpr

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribution, plotInterquantileWidth, quantilePositions, tagClusters

Examples

```
head(tagClusters(exampleCAGEset, "sample1"))
clusterCTSS( object = exampleCAGEset, threshold = 50, thresholdIsTpm = TRUE
           , nrPassThreshold = 1, method = "distclu", maxDist = 20
           , removeSingletons = TRUE, keepSingletonsAbove = 100)
head(tagClusters(exampleCAGEset, "sample1"))

clusterCTSS( exampleCAGEexp, threshold = 50, thresholdIsTpm = TRUE
           , nrPassThreshold = 1, method = "distclu", maxDist = 20
           , removeSingletons = TRUE, keepSingletonsAbove = 100)
tagClustersGR(exampleCAGEexp, "Zf.30p.dome")
```

`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.

`consensusClusterConvertors`
Private functions to convert CC formats

Description

Interconvert consensus clusters (CC) formats used in classes CAGEset (`data.frame`) and CAGEexp (`GRanges`).

Usage

```
CCgranges2dataframe(gr)

CCdataframe2granges(df)
```

Arguments

`gr`	Consensus clusters in `GRanges` format.
`df`	Consensus clusters in `data.frame` format.

See Also

Other df2granges converters: `tagClusterConvertors`

Examples

```
df <- consensusClusters(exampleCAGEset)
head(df)
gr <- CCdataframe2granges(df)
gr
# No round-trip because start and end were not integer in df.
# identical(df, CCgranges2dataframe(gr))
```

`consensusClusters<-` *Set consensus clusters from CAGEr objects*

Description

Set the information on consensus clusters in a `CAGEr` object.

Usage

```
consensusClusters(object) <- value

## S4 replacement method for signature 'CAGEset'
consensusClusters(object) <- value

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

consensusClustersSE(object) <- value

## S4 replacement method for signature 'CAGEset,ANY'
consensusClustersSE(object) <- value


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

consensusClustersGR(object) <- value

## S4 replacement method for signature 'CAGEset'
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 'CAGEset'
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: `CTSSclusteringMethod`, `CTSScumulativesTagClusters`, `CustomConsensusClusters`, `aggregateTagClusters`, `clusterCTSS`, `consensusClustersGR`, `cumulativeCTSSdistribution`, `plotInterquantileWidth`, `quantilePositions`, `tagClusters`

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,
  returnInterquantileWidth = FALSE, qLow = NULL, qUp = NULL)

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

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

consensusClustersSE(object)

## S4 method for signature 'CAGEset'
consensusClustersSE(object)

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

consensusClusters(object, sample = NULL,
  returnInterquantileWidth = FALSE, qLow = NULL, qUp = NULL)

## S4 method for signature 'CAGEr'
consensusClusters(object, sample = NULL,
  returnInterquantileWidth = FALSE, qLow = NULL, qUp = NULL)
```

Arguments

object	A CAGEr object.
sample	Optional. Label of the CAGE dataset (experiment, sample) for which to extract sample-specific information on consensus clusters.

returnInterquantileWidth
: Should the interquantile width of consensus clusters in specified sample be returned. Used only when sample argument is specified, otherwise ignored.

qLow, qUp
: Position of which quantile should be used as a left (lower) or right (upper) boundary when calculating interquantile width. Used only when sample argument is specified and returnInterquantileWidth = TRUE, otherwise ignored.

Value

consensusClustersGR returns a ConsensusClusters object, which is a GRanges wrapper containing similar information as the data frame returned by consensusClustersGR. The score columns indicates the normalised expression value of each cluster, either across all samples (sample = NULL), or for the selected sample. The tpm column provides the same information for compatibility with CAGEset objects but may be removed in the future.

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

consensusClusters returns a data.frame with information on consensus clusters, including genomic coordinates. 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. When returnInterquantileWidth = TRUE, additional sample-specific information is returned, including position of the dominant TSS, and interquantile width of the consensus clusters in the specified sample.

Author(s)

Vanja Haberle

See Also

consensusClusters<-()

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, cumulativeCTSSdistribution, plotInterquantileWidth, quantilePositions, tagClusters

Examples

```
consensusClustersGR( exampleCAGEexp, sample = 2
                   , returnInterquantileWidth = TRUE
                   , qLow = 0.1, qUp = 0.9)
head(consensusClusters(exampleCAGEset))
head(consensusClusters(exampleCAGEset, sample = "sample2"))

cumulativeCTSSdistribution(exampleCAGEset, "consensusClusters") # Defaults in object do not fit
quantilePositions(exampleCAGEset, "consensusClusters")
head(consensusClusters(exampleCAGEset, sample = "sample2"
                     , returnInterquantileWidth = TRUE
```

```
                        , qLow = 0.1, qUp = 0.9))

head(consensusClusters(exampleCAGEexp))
consensusClustersGR(exampleCAGEexp, 2)
```

---

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 'CAGEset'
consensusClustersQuantileLow(object, samples = NULL)

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

consensusClustersQuantileUp(object, samples = NULL)

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

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

consensusClustersQuantile(object, sample = NULL, q)

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

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

consensusClustersQuantileLow(object, samples = NULL) <- value

## S4 replacement method for signature 'CAGEset'
consensusClustersQuantileLow(object,
  samples = NULL) <- value

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

```
## S4 replacement method for signature 'CAGEset'
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 CAGEset 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 'CAGEset'
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

consensusClusters

Examples

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

---

coverage-functions *Private functions behind* cumulativeCTSSdistribution

---

Description

.getCumsum calculates cumulative sums of tpm along the clusters.

Usage

```
.getCAGEsignalCoverage(ctss.chr, clusters)

.getCumsumChr2(clusters, ctss, chrom, str)

.getCumsum(ctss, clusters, useMulticore = FALSE, nrCores = NULL)
```

Arguments

ctss.chr	A 'CTSS.chr" object (guaranteed to have only one chromosome).
clusters	GRanges as per tagClustersGR().
ctss	GRanges as per CTSScoordinatesGR, with the score of one sample.
chrom	a chromosome name
str	a strand name
useMulticore, nrCores	See clusterCTSS.

Details

'.getCAGEsignalCoverage" does... Note that strand is not taken into account.

.getCumsumChr2

Value

.getCumsum returns a list of Rle vectors (IRanges package) containing cumulative sum for each cluster (length of list is equal to number of clusters and names of the list components corespond to the name of the corresponding cluster) v.

Examples

```
library(GenomicRanges)
library(IRanges)
ctss <- CTSS( seqnames = "chr1"
            , IRanges(c(1,3,4,12,14,25,28,31,35), w=1)
            , strand = "+")
score(ctss) <- 1
ctss.chr <- as(ctss, "CTSS.chr")
clusters <- GRanges( seqnames = Rle("chr1")
                   , ranges = IRanges(c(1,12,25,31,32), c(4,14,28,31,33))
                   , strand = "+")
chrom <- "chr1"
str <- "+"
CAGEr:::.getCAGEsignalCoverage(ctss.chr, clusters)
CAGEr:::.getCumsumChr2(clusters, ctss, chrom, str)
ctss      <- CTSSnormalizedTpmGR(exampleCAGEset, "sample1")
ctss      <- ctss[ctss$filteredCTSSidx]
clusters  <- tagClustersGR(exampleCAGEset, "sample1")
clusters.cumsum <- RleList(CAGEr:::.getCumsum(ctss, clusters))
identical( lapply(exampleCAGEset@CTSScumulativesTagClusters[[1]],decode)
         , lapply(clusters.cumsum, decode))
# Not identical if not decoded because Rle method is attached to S4Vectors in one case
# and to IRanges in the other case.
decode(clusters.cumsum[[1]])
ctss[queryHits(findOverlaps(ctss, clusters[1]))]
clusters[1]

ctss      <- CTSSnormalizedTpmGR(exampleCAGEexp, "Zf.30p.dome")
ctss      <- ctss[ctss$filteredCTSSidx]
clusters  <- tagClustersGR(exampleCAGEexp, "Zf.30p.dome")
clusters.cumsum <- CAGEr:::.getCumsum(ctss, head(clusters))
decode(clusters.cumsum[[3]])
ctss[queryHits(findOverlaps(ctss, clusters[3]))]
clusters[3]
```

`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 minuse `stitch`, which is hardcoded to `FALSE`.

The `CTSS.chr` class represents a `CTSS` object that is guaranteed to be only on a single chromosome. It is used internally by *CAGEr* for type-safe polymorphic dispatch.

Usage

```
## 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)
```

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

# (internal use) Transform CTSS to CTSS.chr object
ctss.chr <- as(CTSScoordinatesGR(exampleCAGEexp), "CTSS.chr")
```

`CTSSclusteringMethod` *Get /set CTSS clustering method*

Description

Returns or sets the name of the method that was used make tag clusters from the CTSSs of a `CAGEr` object.

Usage

```
CTSSclusteringMethod(object)

## S4 method for signature 'CAGEset'
CTSSclusteringMethod(object)

## S4 method for signature 'GRangesList'
CTSSclusteringMethod(object)

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

CTSSclusteringMethod(object) <- value

## S4 replacement method for signature 'CAGEset'
CTSSclusteringMethod(object) <- value

## S4 replacement method for signature 'GRangesList'
CTSSclusteringMethod(object) <- value

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

Arguments

`object`	A CAGEr object.
`value`	character

Author(s)

Vanja Haberle

Charles Plessy

See Also

clusterCTSS

Other CAGEr accessor methods: CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Other CAGEr clusters functions: CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribut plotInterquantileWidth, quantilePositions, tagClusters

Examples

```
CTSSclusteringMethod(exampleCAGEset)
CTSSclusteringMethod(exampleCAGEexp)
```

CTSScoordinates *Genomic coordinates of TSSs from a* CAGEr *object*

Description

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

Usage

```
CTSScoordinates(object)

## S4 method for signature 'CAGEset'
CTSScoordinates(object)

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

CTSScoordinatesGR(object)

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

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

CTSScoordinatesGR(object) <- value

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

## S4 replacement method for signature 'CAGEexp'
```

```
CTSScoordinatesGR(object) <- value

CTSStagCountSE(object) <- value

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

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

Arguments

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

Value

CTSScoordinates() returns a data.frame with genomic coordinates of all TSSs. The pos column contains 1-based coordinate of the TSS.

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

Author(s)

Vanja Haberle

Charles Plessy

See Also

getCTSS

clusterCTSS

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Examples

```
CTSS <- CTSScoordinates(exampleCAGEset)
head(CTSS)

CTSScoordinatesGR(exampleCAGEset)
```

CTSScumulativesTagClusters *Get/set CTSS cumulative TC or CC data*

Description

Accessor function.

Usage

```
CTSScumulativesTagClusters(object, samples = NULL)

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

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

CTSScumulativesCC(object, samples = NULL)

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

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

CTSScumulativesTagClusters(object) <- value

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

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

Arguments

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

Value

List of numeric Rle.

See Also

Other CAGEr clusters functions: CTSSclusteringMethod, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribution, plotInterquantileWidth, quantilePositions, tagClusters

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

---

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

Description

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

Usage

```
CTSSnormalizedTpm(object)

## S4 method for signature 'CAGEset'
CTSSnormalizedTpm(object)

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

CTSSnormalizedTpmDf(object)

## S4 method for signature 'CAGEset'
CTSSnormalizedTpmDf(object)

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

CTSSnormalizedTpmDF(object)

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

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

CTSSnormalizedTpmGR(object, samples)

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

Arguments

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

Value

CTSSnormalizedTpm returns a data.frame containing coordinates and normalized CAGE signal supporting each TSS (rows) in every CAGE dataset (columns).

CTSSnormalizedTpmDf returns a data.frame of normalised expression values.

CTSSnormalizedTpmDF returns a DataFrame of normalised expression values.

Author(s)

Vanja Haberle

See Also

normalizeTagCount

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Examples

```
head(CTSSnormalizedTpm(exampleCAGEset))

normalizeTagCount(exampleCAGEexp)
head(CTSSnormalizedTpm(exampleCAGEexp))

head(CTSSnormalizedTpmDf(exampleCAGEset))

normalizeTagCount(exampleCAGEexp)
head(CTSSnormalizedTpmDf(exampleCAGEexp))
```

CTSStagCount *Extract raw CAGE TSSs expression tables from CAGEr objects*

Description

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

Usage

```
CTSStagCount(object)

## S4 method for signature 'CAGEset'
CTSStagCount(object)

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

CTSStagCountDf(object)

## S4 method for signature 'CAGEset'
CTSStagCountDf(object)

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

CTSStagCountDF(object)

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

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

CTSStagCountDA(object)

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

CTSStagCountGR(object, samples)

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

CTSStagCountSE(object)
```

Arguments

object	A CAGEr object.
samples	(For CTSStagCountGR.) Name(s) or number(s) identifying sample(s).

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:

- CTSStagCount: A data.frame, containing coordinates and expression values.

- CTSStagCountDf: A ‘data.frame“, containing only expression values.
- CTSStagCountDF: A DataFrame of Rle integers.
- CTSStagCountDA: A DelayedArray wrapping 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.

Author(s)

Vanja Haberle

Charles Plessy

See Also

getCTSS()

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Examples

```
head(CTSStagCount(exampleCAGEset))
head(CTSStagCountDf(exampleCAGEset))
CTSStagCountDF(exampleCAGEset)
CTSStagCountDA(exampleCAGEset)

head(CTSStagCount(exampleCAGEexp))
head(CTSStagCountDf(exampleCAGEexp))
CTSStagCountDF(exampleCAGEset)
CTSStagCountDA(exampleCAGEset)
```

CTSStagCountTable *Extracting CAGE tag count for TSSs from CAGEr objects*

Description

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

Usage

```
CTSStagCountTable(object)

## S4 method for signature 'CAGEset'
CTSStagCountTable(object)

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

## S4 method for signature 'CAGEset'
CTSStagCountSE(object)

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

Arguments

object	A CAGEset or CAGEexp object.

Value

Returns an expression table with the number of CAGE tags supporting each TSS (rows) in every CAGE dataset (columns). The table is in `data.frame` format for `CAGEset` objects and in `DataFrame` format for `CAGEexp` objects. Use this function when the next consumer can handle both formats.

Author(s)

Vanja Haberle

See Also

`getCTSS`

Other CAGEr accessor methods: `CTSSclusteringMethod`, `CTSScoordinates`, `CTSScumulativesTagClusters`, `CTSSnormalizedTpm`, `CTSStagCount`, `GeneExpDESeq2`, `GeneExpSE`, `consensusClustersGR`, `genomeName`, `inputFilesType`, `inputFiles`, `librarySizes`, `sampleLabels`, `seqNameTotalsSE`, `tagClusters`

Examples

```
tagCount <- CTSStagCount(exampleCAGEset)
head(tagCount)
```

CTSStoGenes *Make a gene expression table.*

Description

Add a gene expression table in the GeneExpSE experiment slot of an annotated CAGEexp object. CAGEset objects are not supported.

Usage

```
CTSStoGenes(object)

## S4 method for signature 'CAGEset'
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, clusterCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, quantilePositions, 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 raw 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 'CAGEr'
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

For CAGEset objects, the slot CTSScumulativesTagClusters (when clusters = "tagClusters") or CTSScumulativesConsensusClusters (when clusters = "consensusClusters") of the will be occupied by the list containing cumulative sum of the CAGE signal along genomic regions per CAGE experiment. In CAGEexp objects, cumulative sums are stored in the metadata slot using the RleList class.

Author(s)

Vanja Haberle

See Also

Other CAGEr object modifiers: CTSStoGenes, CustomConsensusClusters, aggregateTagClusters, annotateCTSS, clusterCTSS, getCTSS, normalizeTagCount, quantilePositions, summariseChrExpr

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, plotInterquantileWidth, quantilePositions, tagClusters

Examples

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

clusterCTSS( exampleCAGEexp, threshold = 50, thresholdIsTpm = TRUE
           , nrPassThreshold = 1, method = "distclu", maxDist = 20
           , removeSingletons = TRUE, keepSingletonsAbove = 100)
cumulativeCTSSdistribution(exampleCAGEexp, clusters = "tagClusters")
aggregateTagClusters( exampleCAGEexp, tpmThreshold = 50
                    , excludeSignalBelowThreshold = FALSE, maxDist = 100)
cumulativeCTSSdistribution(exampleCAGEexp, clusters = "consensusClusters")
```

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`, `clusterCTSS`, `cumulativeCTSSdistribution`, `getCTSS`, `normalizeTagCount`, `quantilePositions`, `summariseChrExpr`

Other CAGEr clusters functions: `CTSSclusteringMethod`, `CTSScumulativesTagClusters`, `aggregateTagClusters`, `clusterCTSS`, `consensusClustersDESeq2`, `consensusClustersGR`, `cumulativeCTSSdistribution`, `plotInterquantileWidth`, `quantilePositions`, `tagClusters`

Examples

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

---

distclu-functions *Private functions for distance clustering.*

Description

The flow of data is that a CTSS object of CTSSes is progressively deconstructed, and data to form the clusters is progressively integrated in a data.table object, which is finally converted to GRanges at the end. Doing the whole clustering with `GRanges` is more elegant, but looping on a `GRangesList` was just too slow. Maybe the operation on the `data.table` is more efficient because it is vectorised.

`.cluster.ctss.strand` does the strandless distance clustering of strandless CTSS positions from a single chromosome. Input does not need to be sorted, but *pay attention that the output is sorted*.

`.cluster.ctss.chr` does the stranded distance clustering of CTSS on a single chromosome, by dispatching both strands to `.cluster.ctss.strand` and merging the results, taking care keep the cluster IDs unique. Be careful that this function does not look at the score.

`.ctss2clusters` does the stranded distance clustering of CTSS.

`.summarize.clusters` calculates the number of CTSS, and the position and score of a main peak, for each cluster.

.distclu receives the data from the main clusterCTSS and dispatches each for (possibly parallel) processing.

Usage

```
.cluster.ctss.strand(ctss.ipos.chr, max.dist)

.cluster.ctss.chr(ctss.chr, max.dist)

.ctss2clusters(ctss, max.dist = 20, useMulticore = FALSE, nrCores = NULL)

.summarize.clusters(
  ctss.clustered,
  removeSingletons = FALSE,
  keepSingletonsAbove = Inf
)

.distclu(
  se,
  max.dist = 20,
  removeSingletons = FALSE,
  keepSingletonsAbove = Inf,
  useMulticore = FALSE,
  nrCores = NULL
)
```

Arguments

ctss.ipos.chr	A IPos object.
max.dist	See clusterCTSS().
ctss.chr	A CTSS.chr object.
ctss	A CTSS object with a score column.
useMulticore, nrCores	See clusterCTSS.
ctss.clustered	A data.table object representing the cluster ID (id), chromosome coordinates (chr, pos, strand) and the score of each CTSS.
removeSingletons	Remove "singleton" clusters that span only a single nucleotide ? (default = FALSE).
keepSingletonsAbove	Even if removeSingletons = TRUE, keep singletons when their score is aboove threshold (default = Inf).
se	A SummarizedExperiment object representing the CTSSes and their expression in each sample.

Value

`.cluster.ctss.strand` returns an data.table object containing arbitrary cluster IDs (as integers) for each CTSS.

`.cluster.ctss.chr` returns a `data.table` object representing the chromosome coordinates (`chr`, `pos`, `strand`) of each CTSS, with their cluster ID (`id`).

`.ctss2clusters` returns a `data.table` object representing the cluster ID (`id`), chromosome coordinates (`chr`, `pos`, `strand`) and the `score` of each CTSS.

`.summarize.clusters` returns GRanges describing the clusters.

`.distclu` returns GRanges describing the clusters.

Examples

```
# Get example data
library(IRanges)
library(GenomicRanges)

#.cluster.ctss.strand
ctss.ipos.chr <- IPos(c(1,3,4,12,14,25,28))
CAGEr:::.cluster.ctss.strand(ctss.ipos.chr, 5)

ctss.chr <- CTSScoordinatesGR(exampleCAGEexp)
ctss.chr <- ctss.chr[strand(ctss.chr) == "+"]
ctss.ipos.chr <- ranges(ctss.chr)
# Same result if not sorted
identical(
  CAGEr:::.cluster.ctss.strand(ctss.ipos.chr, 20),
  CAGEr:::.cluster.ctss.strand(ctss.ipos.chr[sample(seq_along(ctss.ipos.chr))], 20)
)
# Returns an emtpy data.table object if given an empty IRanges object.
identical(CAGEr:::.cluster.ctss.strand(IPos(), 20), data.table::data.table())

#.cluster.ctss.chr
ctss.chr <- as(CTSScoordinatesGR(exampleCAGEexp), "CTSS.chr")
CAGEr:::.cluster.ctss.chr(ctss.chr, 20)

# .ctss2clusters
ctss <- CTSScoordinatesGR(exampleCAGEexp)
score(ctss) <- CTSSnormalizedTpmDF(exampleCAGEexp)[[1]]
seqnames(ctss)[rep(c(TRUE,FALSE), length(ctss) / 2)] <- "chr16"
ctss
clusters <- CAGEr:::.ctss2clusters(ctss, 20)
clusters

# .summarize.clusters
CAGEr:::.summarize.clusters(clusters)
CAGEr:::.summarize.clusters(clusters, removeSingletons = TRUE)
CAGEr:::.summarize.clusters(clusters, removeSingletons = TRUE, keepSingletonsAbove = 5)

# .distclu
CAGEr:::.distclu(CTSStagCountSE(exampleCAGEexp))
```

```
## Not run:
CAGEr:::.distclu(CTSStagCountSE(exampleCAGEexp), useMulticore = TRUE)

## End(Not run)

CAGEr:::.distclu(CTSStagCountSE(exampleCAGEset))
```

`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$"
                                 , full.names = TRUE)
sampleLabels <- sub( ".chr17.ctss", "", basename(pathsToInputFiles))
exampleCAGEexp <-
  CAGEexp( genomeName     = "BSgenome.Drerio.UCSC.danRer7"
         , inputFiles     = pathsToInputFiles
         , inputFilesType = "ctss"
         , sampleLabels   = sub( ".chr17.ctss", "", basename(pathsToInputFiles)))
getCTSS(exampleCAGEexp)
librarySizes(exampleCAGEexp)
colData(exampleCAGEexp)
exampleCAGEexp$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
summariseChrExpr(exampleCAGEexp)
annotateCTSS(exampleCAGEexp, exampleZv9_annot)
CTSStoGenes(exampleCAGEexp)
normalizeTagCount(exampleCAGEexp)
clusterCTSS(exampleCAGEexp)
cumulativeCTSSdistribution(exampleCAGEexp, "tagClusters")
quantilePositions(exampleCAGEexp, "tagClusters")
aggregateTagClusters(exampleCAGEexp)
```

```
annotateConsensusClusters(exampleCAGEexp, exampleZv9_annot)
cumulativeCTSSdistribution(exampleCAGEexp, "consensusClusters")
quantilePositions(exampleCAGEexp, "consensusClusters")
save(exampleCAGEexp, file = "data/exampleCAGEexp.RData", compress = "xz")

## End(Not run)
```

`exampleCAGEset` *Example CAGEset object with zebrafish CAGE data*

Description

This is a `CAGEset` object that contains CAGE data for part of chromosome 13 from five different zebrafish (*Danio rerio*) samples. It is intended to be used as an input data in running examples from `CAGEr` package help pages.

Usage

```
data(exampleCAGEset)
```

Format

A `CAGEset` object

Examples

```
exampleCAGEset
```

`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?VIRTUALSCHEMANAME=

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 = "" datasetC
  <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$Exon.Chr.End..bp.), i
e$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$Exon.Chr.End..bp.), i
e$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/

---

exportCTSStoBedGraph *Creating bedGraph/bigWig tracks of CAGE transcription starts sites*

Description

Creates bedGraph or BigWig file(s) with track(s) of CAGE signal supporting each TSS that can be visualised in the UCSC Genome Browser.

Usage

```
exportCTSStoBedGraph(object, values = "normalized", format = "BigWig",
  oneFile = TRUE)

## S4 method for signature 'CAGEr'
exportCTSStoBedGraph(object, values = "normalized",
  format = "BigWig", oneFile = TRUE)
```

Arguments

object	A CAGEr object.
values	Specifies which values will be exported to the bedGraph file. Can be either "raw" to export raw tag count values or "normalized" to export normalized values.
format	The format of the output.
oneFile	Logical, should all CAGE datasets be exported as individual tracks into the same bedGraph file (TRUE) or into separate bedGraph files (FALSE). Used only when format="bedGraph", otherwise ignored.

Value

Creates bedGraph or BigWig file(s) in the working directory that can be directly visualised as custom tracks in the UCSC Genome Browser. If format="bedGraph" and oneFile = TRUE one bedGraph file containing multiple annotated tracks will be created, otherwise two files per CAGE dataset will be created, one for plus strand and one for minus strand CTSSs, and they will be named according to the labels of individual datasets. All bedGraph files contain headers with track description and can be directly uploaded as custom tracks to the UCSC Genome Browser.

When format="bigWig", two binary BigWig files per CAGE dataset are created, one for plus strand and one for minus strand CTSSs. Since BigWig files cannot contain headers with track description, a separate file named "CTSS.normalized.all.samples.track.description.txt" is created, which contains track headers for all BigWig files. To use these headers for adding custom tracks to the UCSC Genome Browser, move the BigWig files to a web location and edit the bigDataUrl sections in the headers file to point to corresponding BigWig files.

Author(s)

Vanja Haberle

See Also

normalizeTagCount

Other CAGEr export functions: exportToBed

Examples

```
exportCTSStoBedGraph(exampleCAGEset, values = "normalized", format = "bedGraph", oneFile = TRUE)
exportCTSStoBedGraph(exampleCAGEset, values = "normalized", format = "BigWig", oneFile = TRUE)
```

exportToBed *Create BED tracks of TSSs and clusters of TSSs*

Description

Creates BED file(s) with track(s) of individual CTSSs, tag clusters or consensus clusters. CTSSs and consensus clusters can be optionally colored in the color of their expression class. *Tag clusters* and *consensus clusters* can be displayed in a gene-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

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

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

Arguments

object	A CAGEr object.
what	Which elements should be exported to BED track. CTSS to export individual CTSSs, tagClusters to export tag clusters or consensusClusters to export consensus clusters.
qLow, qUp	Position of which "lower" (resp. "upper") quantile should be used as 5' (resp. 3') boundary of the filled block in gene-like representation of the cluster. Default value NULL uses start (resp. end) position of the cluster. Ignored when what = "CTSS".
colorByExpressionProfile	Logical, should blocks be colored in the color of their corresponding expression class. Ignored when what = "tagClusters".
oneFile	Logical, should all CAGE datasets be exported as individual tracks into the same BED file (TRUE) or into separate BED files (FALSE). Ignored when what = "CTSS", which by default produces only one track.

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 BED file 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 (provided the expression profiling of CTSSs 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, 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 BED file (by setting oneFile = TRUE) or separate BED files (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 to BED file. 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 BED file (by setting oneFile = TRUE) or separate BED files (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

Creates BED file(s) in the working directory.

Author(s)

Vanja Haberle

See Also

Other CAGEr export functions: exportCTSStoBedGraph

Examples

```
### exporting CTSSs colored by expression class
exportToBed(object = exampleCAGEset, what = "CTSS", colorByExpressionProfile = TRUE)

### exporting tag clusters in gene-like representation
exportToBed( object = exampleCAGEset, what = "tagClusters"
           , qLow = 0.1, qUp = 0.9, oneFile = TRUE)

exportToBed( object = exampleCAGEexp, what = "tagClusters"
           , qLow = 0.1, qUp = 0.9, oneFile = TRUE)
```

expressionClasses *Extract labels of expression classes*

Description

Retrieves labels of expression classes of either individual CTSSs or consensus clusters from a CAGEset object.

Usage

```
expressionClasses(object, what = c("CTSS", "consensusClusters"))

## S4 method for signature 'CAGEset'
expressionClasses(object, what = c("CTSS",
  "consensusClusters"))

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

Arguments

object	A CAGEset object.
what	Which level of expression clustering should be used. Can be either "CTSS" to extract labels of expression classes of individual CTSSs or "consensusClusters" to extract labels of expression classes of consensus clusters.

Value

Returns character vector of labels of expression classes. The number of labels matches the number of expression clusters returned by getExpressionProfiles function.

See Also

getExpressionProfiles plotExpressionProfiles extractExpressionClass

Examples

```
exprClasses <- expressionClasses(exampleCAGEset, what = "CTSS")
exprClasses
```

extractExpressionClass *Extract elements of the specified expression class*

Description

Extracts CTSSs or consensus clusters belonging to a specified expression class.

Usage

```
extractExpressionClass(object, what, which = "all")

## S4 method for signature 'CAGEexp'
extractExpressionClass(object, what, which = "all")

## S4 method for signature 'CAGEset'
extractExpressionClass(object, what, which = "all")
```

Arguments

object	A CAGEset object.
what	Which level of expression clustering should be used. Can be either "CTSS" to extract expression class of individual CTSSs or "consensusClusters" to extract expression class of consensus clusters.
which	Which expression class should be extracted. It has to be one of the valid expression class labels (as returned by expressionClasses function), or "all" to extract members of all expression classes.

Value

Returns a data.frame of CTSSs (when what = "CTSS") or consensus clusters (when what = "consensusClusters") belonging to a specified expression class, with genomic coordinates and additional information. Last column contains the label of the corresponding expression class.

Author(s)

Vanja Haberle

See Also

getExpressionProfiles, plotExpressionProfiles, expressionClasses.

Examples

```
CTSSexprClasses <- extractExpressionClass(exampleCAGEset, what = "CTSS", which = "all")
head(CTSSexprClasses)
```

FANTOM5humanSamples *List of FANTOM5 human CAGE samples*

Description

This is a `data.frame` object that contains information about all human primary cell, cell line and tissue CAGE datasets (988 datasets) produced by FANTOM5 constortium, which are available for import with `CAGEr`. The list containes all samples published in the main FANTOM5 publication (Forrest *et al.* Nature 2014) as presented in the Supplementary Table 1. The columns provide the following information:
`sample`: name/label of individual sample, which can be used as `sample` argument in the `importPublicData` function to retrieve specified sample.
`type`: type of the sample, either `"cell line"`, `"primary cell"` or `"tissue"`.
`description`: description of individual sample as provided by the FANTOM5 consortium.
`library_id`: unique ID of the CAGE library as provided by the FANTOM5 consortium.
`data_url`: URL to gzip-ed TSS file at online FANTOM5 data resource, which is used by `importPublicData` to fetch given sample.

Usage

```
data(FANTOM5humanSamples)
```

Format

A `data.frame` object

FANTOM5mouseSamples *List of FANTOM5 mouse CAGE samples*

Description

This is a `data.frame` object that contains information about all mouse primary cell, cell line and tissue CAGE datasets (395) produced by FANTOM5 constortium, which are available for import with `CAGEr`. The list containes all samples published in the main FANTOM5 publication (Forrest *et al.* Nature 2014) as presented in the Supplementary Table 1. The columns provide the following information:
`sample`: name/label of individual sample, which can be used as `sample` argument in the `importPublicData` function to retrieve specified sample.
`type`: type of the sample, either `"cell line"`, `"primary cell"` or `"tissue"`.
`description`: description of individual sample as provided by the FANTOM5 consortium.
`library_id`: unique ID of the CAGE library as provided by the FANTOM5 consortium.
`data_url`: URL to gzip-ed TSS file at online FANTOM5 data resource, which is used by `importPublicData` to fetch given sample.

Usage

```
data(FANTOM5humanSamples)
```

Format

A data.frame object

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 'CAGEset'
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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

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 'CAGEset'
GeneExpSE(object)

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

Arguments

object	A CAGEexp object.

Author(s)

Charles Plessy

See Also

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

Examples

```
GeneExpSE(exampleCAGEexp)
```

genomeName *Extracting genome name from CAGEr objects*

Description

Extracts the name of a referent genome from a CAGEset and CAGEexp objects.

Usage

```
genomeName(object)

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

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

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

genomeName(object) <- value

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

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

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

Arguments

object	A CAGEset or CAGEexp 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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

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

Examples

```
genomeName(exampleCAGEset)
```

---

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 'CAGEset'
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 CAGEset or a 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

For `CAGEset` objects, the slots `librarySizes`, `CTSScoordinates` and `tagCountMatrix` will be occupied by the information on CTSSs created from input CAGE files. For `CAGEexp` objects, the `tagCountMatrix` experiment will be occupied by a `SummarizedExperiment` containing the expression data as a `DataFrame` of `Rle` integers, and the CTSS coordinates as a `GRanges` object. In both cases the expression data can be retreived with `CTSStagCount` functions. In addition, the library sizes are calculated and stored in the object.

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

`CTSScoordinates`, `CTSStagCount`, `CTSStagCountTable`, `inputFilesType`, `librarySizes`.

Other CAGEr object modifiers: `CTSStoGenes`, `CustomConsensusClusters`, `aggregateTagClusters`, `annotateCTSS`, `clusterCTSS`, `cumulativeCTSSdistribution`, `normalizeTagCount`, `quantilePositions`, `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 = "")

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

```
getCTSS(myCAGEset)
```

`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, tpmThreshold = 5,
  nrPassThreshold = 1, method = "som", xDim = 5, yDim = 5)

## S4 method for signature 'CAGEset'
getExpressionProfiles(object, what, tpmThreshold = 5,
  nrPassThreshold = 1, method = "som", xDim = 5, yDim = 5)
```

Arguments

`object`	A `CAGEset` object
`what`	At which level should the expression clustering be done. Can be either `"CTSS"` to perform clustering of individual CTSSs or `"consensusClusters"` to perform clustering of consensus clusters. See Details.
`tpmThreshold, nrPassThreshold`	Only CTSSs or consensus clusters (depending on `what` parameter) with normalized CAGE signal >= `tpmThreshold` in >= `nrPassThreshold` experiments will be included in expression clustering.
`method`	Method to be used for expression clustering. Can be either `"som"` to use the self-organizing map (SOM) algorithm (Toronen *et al*., FEBS Letters 1999) implemented in the the `som` function from `som` package, or `"kmeans"` to use the K-means algorithm implemented in the `kmeans` function from `stats` package.
`xDim, yDim`	When `method = "kmeans"`, `xDim` specifies number of clusters that will be returned by K-means algorithm and `yDim` is ignored. When `method = "som"`, `xDim` specifies the the first and `yDim` the second dimension of the self -organizing map, which results in total `xDim * 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 the clusterCTSS 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

If what = "CTSS" the slots CTSSexpressionClusteringMethod and CTSSexpressionClasses will be occupied, and if what = "consensusClusters" the slots consensusClustersExpressionClusteringMethod and consensusClustersExpressionClasses of the provided CAGEset object will be occupied with the results of expression clustering. Labels of expression classes (clusters) can be retrieved using expressionClasses function, and elements belonging to a specific expression class can be selected using extractExpressionClass function.

Author(s)

Vanja Haberle

References

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

See Also

plotExpressionProfiles, expressionClasses, extractExpressionClass.

Examples

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

---

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, tpmThreshold = 0, scoreThreshold = -Inf,
  fdrThreshold = 1)

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

Arguments

`object`	A `CAGEset` object.
`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 comapred 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

See Also

Other CAGEr promoter shift functions: `scoreShift`

Examples

```
head(getShiftingPromoters( exampleCAGEset, tpmThreshold = 100
                         , scoreThreshold = 0.4, fdrThreshold = 0.01))
```

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 richness functions: hanabi, plot.hanabi

Other CAGEr plot functions: 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`, `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, 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.ctss, import.bam, import.bedScore, import.bedmolecule, 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.ctss, import.bam, import.bedCTSS, import.bedScore, 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.ctss`, `import.bam`, `import.bedCTSS`, `import.bedmolecule`, `loadFileIntoGPos`, `moleculesGR2CTSS`

Examples

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

`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.ctss`, `import.bam`, `import.bedCTSS`, `import.bedScore`, `import.bedmolecule`, `loadFileIntoGPos`, `moleculesGR2CTSS`

Examples

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

importPublicData *Importing publicly available CAGE data from various resources*

Description

Imports CAGE data from different sources into a CAGEset object. After the CAGEset 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:
- 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.
- 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 developmental timecourse datasets (Nepal *et al.* Genome Research 2013) from *ZebrafishDevelopmentalCAGE* data package, which is available for download from http://promshift.genereg.net/CAGEr/.

Usage

```
importPublicData(source, dataset, group, sample)
```

Arguments

source	Character vector specifying one of the available resources for CAGE data. Can be one of the following: "FANTOM5": for fetching and importing CAGE data for various human or mouse primary cells, cell lines and tissues from the online FANTOM5 resource (http://fantom.gsc.riken.jp/5/data/ All data published in main FANTOM5 publication by Forrest *et al.* is available. "FANTOM3and4": for importing CAGE data for various human or mouse tissues produced within FANTOM3 and FANTOM4 projects. Requires data package *FANTOM3and4CAGE* to be installed. This data package is available from Bioconductor. "ENCODE": for importing CAGE data for human cell lines from ENCODE project published by Djebali *et al.*. Requires data package *ENCODEprojectCAGE* to be installed. This data package is available for download from http://promshift.genereg.net/CAGEr/. "ZebrafishDevelopment": for importing CAGE data from developmental timecourse of zebrafish (*Danio rerio*) published by Nepal *et al.*. Requires data package *ZebrafishDevelopmentalCAGE* to be installed. This data package is available for download from http://promshift.genereg.net/CAGEr/. See Details for further explanation of individual resources.
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. `group` argument is used only when importing TSSs from data packages and ignored when `source="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

CAGE data from different sources is available for importing directly into `CAGEset` object for further manipulation with *CAGEr*.

FANTOM consortium provides single base-pair resolution TSS data for numerous human and mouse primary cells, cell lines and tissues produced within FANTOM5 project (Forrest *et al.* Nature 2014). These are directly fetched from their online resource at http://fantom.gsc.riken.jp/5/data and imported into a `CAGEset` object. To use this resource specify `source="FANTOM5"`. The `dataset` argument can be either `"human"` or `"mouse"`, but not both at the same time. The list of all human and mouse samples can be obtained by loading `data(FANTOM5humanSamples)` and `data(FANTOM5mouseSamples)`. The `sample` column gives the names of individual samples that should be provided as `sample` argument. See example below.

TSS data from previous FANTOM3 and FANTOM4 projects (Carninci *et al.*, Faulkner *et al.*, Suzuki *et al.*) are also available through *FANTOM3and4CAGE* data package. This data package can be installed directly from Bioconductor. To use this resource install and load *FANTOM3and4CAGE* package and specify `source="FANTOM3and4"`. The `dataset` argument can be a name of any of the datasets available in this package. Load `data(FANTOMhumanSamples)` or `data(FANTOMmouseSamples)` for the list of available datasets with group and sample labels for specific human or mouse samples. These have to be provided as `dataset`, `group` and `sample` arguments to import selected samples. If all samples belong to the same group, only this one group can be provided, otherwise, for each sample a corresponding group has to be specified, *i.e.* the number of elements in `group` must match the numer of elements in `sample`.

ENCODE consortium produced CAGE data for numerous human cell lines (Djebali *et al.* Nature 2012). We have used these data to derive single base-pair resolution TSSs and collected them into an R data package *ENCODEprojectCAGE*. This data package is available for download from http://promshift.genereg.net/CAGEr/. To use this resource install and load *ENCODEprojectCAGE* data package and specify `source="ENCODE"`. The `dataset` argument can be a name of any of the datasets available in this package. Load `data(ENCODEhumanCellLinesSamples)` for the list of available datasets with group and sample labels for specific samples. These have to be provided as `dataset`, `group` and `sample` arguments to import selected samples. Multiple datasets can be combined together, by specifying them as `dataset` argument. If all samples belong to the same dataset and the same group, these dataset and group can be specified only once, otherwise, for each sample a corresponding dataset and group have to be specified, *i.e.* the number of elements in `dataset` and

group must match the numer of elements in sample.
Precise TSSs are also available for zebrafish (*Danio Rerio*) from CAGE data published by Nepal *et al.* for 12 developmental stages. These have been collected into a data package *ZebrafishDevelopmentalCAGE*, which is available for download from http://promshift.genereg.net/CAGEr/. To use this resource install and load *ZebrafishDevelopmentalCAGE* data package and specify source="ZebrafishDevelopment". Load data(ZebrafishSamples) for the list of available datasets and group and sample labels, which have to be specified to import these data.

Value

A CAGEset object is returned. Slots librarySizes, CTSScoordinates and tagCountMatrix are occupied by the single base-pair resolution TSS data imported from the selected resource.

Author(s)

Vanja Haberle

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

getCTSS

Examples

```
### importing FANTOM5 data

# list of FANTOM5 human tissue samples
data(FANTOM5humanSamples)
head(subset(FANTOM5humanSamples, type == "tissue"))

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

exampleCAGEset


### 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
exampleCAGEset <- importPublicData(source="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"))

exampleCAGEset
```

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

Description

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

Usage

```
inputFiles(object)

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

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

inputFiles(object) <- value

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

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

Arguments

`object`	A CAGEset or 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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

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

Examples

```
inputFiles(exampleCAGEset)
```

---

inputFilesType *Input file formats for CAGEr objects*

Description

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

Usage

```
inputFilesType(object)

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

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

inputFilesType(object) <- value

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

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

Arguments

object	A CAGEset or 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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE, tagClusters

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

Examples

```
inputFilesType(exampleCAGEset)
```

librarySizes *Extracting library sizes from CAGEr objects*

Description

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

Usage

```
librarySizes(object)

## S4 method for signature 'CAGEset'
librarySizes(object)

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

Arguments

object	A CAGEset or CAGEexp object.

Details

Library sizes are calculated when loading data with the getCTSS function and stored in the librarySizes slot of CAGEset objects, or in the librarySizes column of the colData ov 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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, sampleLabels, seqNameTotalsSE, tagClusters

Examples

```
librarySizes(exampleCAGEset)
```

loadFileIntoGPos *loadFileIntoGPos*

Description

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

Usage

```
loadFileIntoGPos(filepath, filetype = c("bam", "bamPairedEnd", "bed",
  "bedctss", "bedScore", "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.ctss, import.bam, import.bedCTSS, import.bedScore, import.bedmolecule, 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 = "default", 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 "group" column of the "libs" table.
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

```
library(SummarizedExperiment)
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 `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 'CAGEset,CAGEset'
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 CAGEset or 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

See Also

CAGEset, 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")

myCAGEset1 <- new("CAGEset", genomeName = "BSgenome.Drerio.UCSC.danRer7",
inputFiles = pathsToInputFiles[1:2], inputFilesType = "ctss", sampleLabels =
c("sample1", "sample2"))
getCTSS(myCAGEset1)

myCAGEset2 <- new("CAGEset", genomeName = "BSgenome.Drerio.UCSC.danRer7",
inputFiles = pathsToInputFiles[3], inputFilesType = "ctss", sampleLabels =
"sample3")

getCTSS(myCAGEset2)

myCAGEset <- mergeCAGEsets(myCAGEset1, myCAGEset2)


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

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

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 'CAGEset,numeric'
mergeSamples(object, mergeIndex,
  mergedSampleLabels)

## S4 method for signature 'CAGEexp,ANY'
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

Examples

```
mergeSamples( exampleCAGEset
            , mergeIndex = c(1,1,2)
```

```
            , mergedSampleLabels = c("mergedSample1", "mergedSample2"))
exampleCAGEset

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.ctss`, `import.bam`, `import.bedCTSS`, `import.bedScore`, `import.bedmolecule`, `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 'CAGEset'
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 CAGEset 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 `CAGEset` 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`, `CTSSnormalizedTpm`

Other CAGEr object modifiers: `CTSStoGenes`, `CustomConsensusClusters`, `aggregateTagClusters`, `annotateCTSS`, `clusterCTSS`, `cumulativeCTSSdistribution`, `getCTSS`, `quantilePositions`, `summariseChrExpr`

Examples

```
normalizeTagCount(exampleCAGEset, method = "simpleTpm")
normalizeTagCount(exampleCAGEset, method = "powerLaw")
normalizeTagCount(exampleCAGEexp, method = "simpleTpm")
normalizeTagCount(exampleCAGEexp, method = "powerLaw")
```

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 '|', ',' 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: `hanabiPlot`, `hanabi`

plotAnnot *Plot annotation statistics*

Description

Plot maping statistics of an object containing mapping statistics in counts as percentages in stacked barplots.

Usage

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

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

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

## S4 method for signature 'CAGEexp'
plotAnnot(x, scope, title, group = "default",
  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.
facet	A factor or the name of a colData column of a CAGEexp object, to facet the samples in the sense of ggplot2's facet_wrap function.
normalise	Whether to normalise or not. Default: TRUE.

Details

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 invisibly a ggplot2 object of class c("gg","ggplot").

Author(s)

Charles Plessy

See Also

mapStats() for a list of *scopes*.

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

Other CAGEr plot functions: 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

Calculates the pairwise correlation between samples and creates a plot matrix showing the correlation coeficients in the upper triangle, the sample names in the diagonal, and the catter plots in the lower triangle.

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 'CAGEset'
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)
```

Arguments

object	A CAGEr object or (only for plotCorrelation2) a SummarizedExperiment or an expression table as a DataFrame, data.frame or matrix object.

what The clustering level to be used for plotting and calculating correlations. Can be either "CTSS" to use individual TSSs or "consensusClusters" to use consensus clusters, *i.e.* entire promoters. Ignored for anything else than CAGEr 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 CAGEr object, or a subset of valid sample labels as returned by the sampleLabels function.

method A character string indicating which correlation coefficient should be computed. Passed to cor function. Can be one of "pearson", "spearman", or "kendall".

tagCountThreshold
Only TSSs with tag count >= tagCountThreshold in either one (applyThresholdBoth = FALSE) or both samples (applyThresholdBoth = TRUE) are plotted and used to calculate correlation.

applyThresholdBoth
See tagCountThreshold above.

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.

Author(s)

Vanja Haberle

Charles Plessy

See Also

Other CAGEr plot functions: hanabiPlot, plotAnnot, plotExpressionProfiles, plotInterquantileWidth, plotReverseCumulatives

Examples

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

---

plotExpressionProfiles

Plotting expression profiles derived from CAGE data

Description

Creates a chart with beanplots representing distribution of normalized expression across CAGE experiments for individual expression classes. Different expression classes are shown in different colors and are labeled according to the labels returned by expression clustering.

Usage

```
plotExpressionProfiles(object, what)

## S4 method for signature 'CAGEset'
plotExpressionProfiles(object, what)
```

Arguments

object	A CAGEset object.
what	Which level of expression clustering should be used. Can be either "CTSS" to plot expression profiles of individual CTSSs or "consensusClusters" to plot expression profiles of consensus clusters.

Details

The created file contains beanplots representing distribution of normalized expression across CAGE experiments for individual expression classes shown in separate boxes. Each labeled box represents one expression class and contains a set of beanplots - one per CAGE experiment. Individual CAGE experiments are shown on X-axis and scaled normalized expression on Y-axis. Individual beanplots show distribution of normalized expression values of elements belonging to specific expression class in particular CAGE experiment, and the entire box represents single expression profile. Different expression classes (boxes) are plotted in different colors and are labeled with labels returned by expression clustering.

Author(s)

Vanja Haberle

See Also

getExpressionProfiles, expressionClasses, extractExpressionClass.

Other CAGEr plot functions: hanabiPlot, plotAnnot, plotCorrelation, plotInterquantileWidth, plotReverseCumulatives

Examples

```
plotExpressionProfiles(object = exampleCAGEset, what = "CTSS")
```

---

plotInterquantileWidth

Plot cluster widths

Description

Plots histograms of the interquantile width of tag clusters or consensus clusters in each CAGE dataset.

Usage

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

## S4 method for signature 'CAGEr'
plotInterquantileWidth(object,
  clusters = c("tagClusters", "consensusClusters"), tpmThreshold = 5,
  qLow = 0.1, qUp = 0.9, xlim = c(0, 150), ...)
```

Arguments

object	A CAGEr object
clusters	Which clusters to be used. ("tagClusters" for *tag clusters* or "consensusClusters" for *consensus clusters*.
tpmThreshold	Exclude clusters with normalized signal < tpmThreshold.
qLow, qUp	Position of which "lower" (resp. "upper") quantile should be used as 5' (resp. 3') boundary. See Details.
xlim	The x axis limits of the plot.
...	Additional arguments passed to plot() function, such as ylim, *etc.*.

Details

Interquantile width is a width (in base-pairs) of the central part of the genomic region (bounded by the positions of specified qLow and qUp quantiles) that contains >= (qUp -qLow) * 100% of the CAGE signal. Positions of specified quantiles within each cluster have to be calculated beforehand by calling quantilePositions function. 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.

Author(s)

Vanja Haberle

See Also

Other CAGEr plot functions: hanabiPlot, plotAnnot, plotCorrelation, plotExpressionProfiles, plotReverseCumulatives

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribut quantilePositions, tagClusters

Examples

```
plotInterquantileWidth( object = exampleCAGEset, clusters = "tagClusters"
                      , tpmThreshold = 50, qLow = 0.1, qUp = 0.9)

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

---

plotReverseCumulatives

Plot reverse cumulative number of CAGE tags per CTSS

Description

Plots the reverse cumulative distribution of the number of CAGE tags per CTSS for all CAGE datasets present in the CAGEr object. The plots should be used as help in choosing the parameters for power-law normalization: the range of values to fit the power-law and the slope of the referent power-law distribution (Balwierz *et al.*, Genome Biology 2009).

Usage

```
plotReverseCumulatives(object, values = c("raw", "normalized"),
  fitInRange = c(10, 1000), onePlot = FALSE, main = NULL,
  legend = TRUE, xlab = "number of CAGE tags",
  ylab = "number of CTSSs (>= nr tags)", xlim = c(1, 1e+05),
  ylim = c(1, 1e+06))
```

```
## S4 method for signature 'CAGEr'
plotReverseCumulatives(object, values = c("raw",
  "normalized"), fitInRange = c(10, 1000), onePlot = FALSE,
  main = NULL, legend = TRUE, xlab = "number of CAGE tags",
  ylab = "number of CTSSs (>= nr tags)", xlim = c(1, 1e+05),
  ylim = c(1, 1e+06))
```

Arguments

object	A CAGEr object
values	Which values should be plotted: raw (default) for raw CAGE tag counts or normalized for normalized tag count 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.
onePlot	Logical, should all CAGE datasets be plotted in the same plot (TRUE) or in separate plots (FALSE).
main	Main title for the plot.
legend	Set to NULL to prevent the display of the sample legend.
xlab, ylab	Axis labels passed to plot.
xlim, ylim	Axis range parameters passed to plot.

Details

Number of CAGE tags (X-axis) is plotted against the number of TSSs that are supported by >= of that number of tags (Y-axis) on a log-log scale for each sample. In addition, a power-law distribution is fitted to each reverse cumulative using the values in the range specified by fitInRange parameter. 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. In addition, a suggested referent power-law distribution to which all samples should be normalized is drawn on the plot and corresponding parameters (slope alpha and total number of tags T) are denoted on the plot. 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 for referent power-law distribution to which all samples will be normalized. For details about normalization see normalizeTagCount function.

Value

Plots of reverse cumulative number of CAGE tags per CTSS for each CAGE dataset within CAGEr object. Alpha values of fitted power-laws and suggested referent power-law distribution are reported on the plot in case values = "raw".

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

normalizeTagCount

Other CAGEr plot functions: hanabiPlot, plotAnnot, plotCorrelation, plotExpressionProfiles, plotInterquantileWidth

Examples

```
plotReverseCumulatives(exampleCAGEset, fitInRange = c(10,500), onePlot = TRUE)
plotReverseCumulatives(exampleCAGEset, values = "normalized", onePlot = TRUE)

plotReverseCumulatives( exampleCAGEexp, xlim = c(1, 1e4), ylim = c(1, 1e5)
                      , fitInRange = c(5,100), onePlot = TRUE)
plotReverseCumulatives( exampleCAGEexp, values = "normalized"
                      , fitInRange = c(200, 2000), onePlot = TRUE)
plotReverseCumulatives( exampleCAGEexp[,4:5], fitInRange = c(5,100)
                      , onePlot = TRUE, main = "prim6 replicates")
```

---

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 CAGEr object.

Usage

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

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

Arguments

object	A CAGEr 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 detemined 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

When clusters = "tagClusters", the slots tagClustersQuantileLow and tagClustersQuantileUp of a provided CAGEset object will be occupied with the positions of specified quantiles in all tag clusters for all CAGE datasets. When clusters = "consensusClusters" the slots consensusClustersQuantileLow and consensusClustersQuantileUp will be occupied by the corresponding information for consensus clusters.

In CAGEexp objects, the positions of the quantiles are defined reliatively 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

See Also

Other CAGEr object modifiers: CTSStoGenes, CustomConsensusClusters, aggregateTagClusters, annotateCTSS, clusterCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, summariseChrExpr

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribut plotInterquantileWidth, tagClusters

Examples

```
head(cbind(
  CAGEr:::tagClustersQuantileLow(exampleCAGEset, 1),
  CAGEr:::tagClustersQuantileUp (exampleCAGEset, 1)
))
quantilePositions( object = exampleCAGEset, clusters = "tagClusters"
                 , qLow = c(0.1, 0.2), qUp = c(0.8, 0.9))
head(cbind(
```

```
  CAGEr:::tagClustersQuantileLow(exampleCAGEset, 1),
  CAGEr:::tagClustersQuantileUp (exampleCAGEset,1 )
))

cumulativeCTSSdistribution(exampleCAGEset, "consensusClusters") # Defaults in object do not fit
quantilePositions( object = exampleCAGEset, clusters = "consensusClusters"
                 , qLow = c(0.1, 0.2), qUp = c(0.8, 0.9))

head(cbind(
  CAGEr:::consensusClustersQuantileLow(exampleCAGEset, 1),
  CAGEr:::consensusClustersQuantileUp (exampleCAGEset , 1)
))

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))
```

`ranges2annot` *Hierarchical annotation of CTSSes*

Description

Assigns region types such as `promoter`, `exon` or `unknown` to CTSSes.

Usage

```
ranges2annot(ranges, annot)
```

Arguments

`ranges`	A `CTSS` 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.

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)
```

---

ranges2genes *ranges2genes*

Description

Assign gene symbol(s) to Genomic Ranges.

Usage

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

Arguments

ranges	Genomics Ranges object, for example extracted from a RangedSummarizedExperiment object with the 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 Rle character vector of same length as the GRanges object, indicating one gene symbol or a semicolon-separated list of gene symbols for each range.

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 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 Rle character vector of same length as the rangesA GRanges object, indicating one name or a semicolon-separated list of names from the each rangesB object.

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)
```

---

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 'CAGEset'
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 'CAGEset'
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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, seqNameTotalsSE, tagClusters

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

Examples

```
sampleLabels(exampleCAGEset)

sampleList(exampleCAGEset)
```

---

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 'CAGEset'
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 `CAGEset` 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

See Also

`cumulativeCTSSdistribution`

Other CAGEr promoter shift functions: getShiftingPromoters

Examples

```
scoreShift( exampleCAGEset
          , groupX = c("sample1", "sample2")
          , groupY = "sample3"
          , testKS = TRUE, useTpmKS = FALSE)
head(getShiftingPromoters(exampleCAGEset))
```

---

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 'CAGEset'
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: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, tagClusters

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`, `inputFilesType`, `inputFiles`, `sampleLabels`

Examples

```
sampleLabels(exampleCAGEset)
setColors(exampleCAGEset, colors = c("darkred", "navy", "forestgreen"))
sampleLabels(exampleCAGEset)

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)
```

`show-methods` *Methods for function* `show`

Description

Methods for function `show`.

Methods

`signature(object = "CAGEset")` Displays a CAGEset object in a user-friendly way, giving an overview of its content.

`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, the input object is modified in place. 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.
"BSgenome.Drerio.UCSC.danRer7"

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 (CAGEset 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, clusterCTSS, cumulativeCTSSdistribution, getCTSS, normalizeTagCount, quantilePositions

Examples

```
summariseChrExpr(exampleCAGEexp)
```

`tagClusterConvertors` *Private functions to convert TC formats*

Description

Interconvert tag clusters (TC) formats used in classes CAGEset (`data.frame`) and CAGEexp (`GRanges`).

Usage

```
TCgranges2dataframe(gr)

TCdataframe2granges(df)
```

Arguments

`gr`	Consensus clusters in `GRanges` format.
`df`	Consensus clusters in `data.frame` format.

Details

The original format used in `CAGEset` objects follows BED ("0-based") conventsion for the start and end coordinates. On the other hand, the GRanges objects used in `CAGEexp` objects follow the "1-based" convention. Therefore a value of 1 has to be added or subtracted to the start positions when converting between both formats.

See Also

Other df2granges converters: `consensusClusterConvertors`

Examples

```
df <- tagClusters(exampleCAGEset, 1)
head(df)
gr <- CAGEr:::TCdataframe2granges(df)
gr
head(CAGEr:::TCgranges2dataframe(gr))
# No exact round-trip because start and end were not integer in df.
identical(df, CAGEr:::TCgranges2dataframe(gr))
if (! all(df == CAGEr:::TCgranges2dataframe(gr)))
  stop("No round-trip between TCdataframe2granges and TCgranges2dataframe")

tagClustersGR(exampleCAGEexp)
head(CAGEr:::TCgranges2dataframe(CAGEr:::tagClustersGR(exampleCAGEexp, 1)))
```

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

Description

Extracts tag clusters (TCs) produced by `clusterCTSS` function for a specified CAGE experiment from a CAGEr object.

Usage

```
tagClusters(object, samples = NULL, returnInterquantileWidth = FALSE,
  qLow = NULL, qUp = NULL)

## S4 method for signature 'CAGEr'
tagClusters(object, samples = NULL,
  returnInterquantileWidth = FALSE, qLow = NULL, qUp = NULL)

tagClustersGR(object, sample = NULL, returnInterquantileWidth = FALSE,
  qLow = NULL, qUp = NULL)

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

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

tagClustersGR(object, samples = NULL) <- value

## S4 replacement method for signature 'CAGEset,ANY,ANY'
tagClustersGR(object,
  samples = NULL) <- value

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

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

Arguments

`object`	A `CAGEr` object.
`samples`	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.

returnInterquantileWidth
: Should the interquantile width for each tag cluster be 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. Used only when returnInterquantileWidth = TRUE, otherwise ignored.

sample
: Label of one CAGE dataset (experiment, sample) for which to extract tag clusters. (For tagClustersGR, only one sample can be extracted.)

value
: A TagClusters object.

Value

Returns a data.frame with genomic coordinates, position of dominant TSS, total CAGE signal and additional information for all TCs from specified CAGE dataset (sample). If returnInterquantileWidth = TRUE, interquantile width for each TC is also calculated using specified quantile positions and returned in the data frame.

Author(s)

Vanja Haberle

See Also

Other CAGEr accessor methods: CTSSclusteringMethod, CTSScoordinates, CTSScumulativesTagClusters, CTSSnormalizedTpm, CTSStagCountTable, CTSStagCount, GeneExpDESeq2, GeneExpSE, consensusClustersGR, genomeName, inputFilesType, inputFiles, librarySizes, sampleLabels, seqNameTotalsSE

Other CAGEr clusters functions: CTSSclusteringMethod, CTSScumulativesTagClusters, CustomConsensusClusters, aggregateTagClusters, clusterCTSS, consensusClustersDESeq2, consensusClustersGR, cumulativeCTSSdistribut plotInterquantileWidth, quantilePositions

Examples

```
head(tagClusters( exampleCAGEset, "sample2"
                , returnInterquantileWidth = TRUE, qLow = 0.1, qUp = 0.9))
tagClustersGR(exampleCAGEexp, "Zf.high", TRUE, 0.1, 0.9)
```

---

tagClustersQuantile *Quantile metadata stored in CAGEr objects.*

Description

Accessor functions to quantile metadata.

Usage

```
tagClustersQuantile(object, samples = NULL, q = NULL)

## S4 method for signature 'TagClusters'
tagClustersQuantile(object, samples = NULL,
  q = NULL)

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

tagClustersQuantileLow(object, samples = NULL, q = NULL)

## S4 method for signature 'CAGEset'
tagClustersQuantileLow(object, samples = NULL,
  q = NULL)

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

tagClustersQuantileUp(object, samples = NULL, q = NULL)

## S4 method for signature 'CAGEset'
tagClustersQuantileUp(object, samples = NULL,
  q = NULL)

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

tagClustersQuantileLow(object, samples = NULL) <- value

## S4 replacement method for signature 'CAGEset'
tagClustersQuantileLow(object,
  samples = NULL) <- value

## S4 replacement method for signature 'CAGEexp'
tagClustersQuantileLow(object,
  samples = NULL) <- value

tagClustersQuantileUp(object, samples = NULL) <- value

## S4 replacement method for signature 'CAGEset'
tagClustersQuantileUp(object,
  samples = NULL) <- value

## S4 replacement method for signature 'CAGEexp'
```

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

Arguments

object	A CAGEr object.
samples	Sample name(s), number(s) or NULL (default) for all samples.
q	A single quantile (not a list)
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.

Value

Returns a data.frame where the first column gives cluster names and the next columns give quantile positions, in *zero-based* chromosome coordinates (because the tag clusters in CAGEset objects are represented in zero-based coordinates as well)).

Index