Package 'TSRchitect'

April 12, 2022

Version 1.20.0

Date 2020-3-27

Title Promoter identification from large-scale TSS profiling data

Description In recent years, large-scale transcriptional sequence data has yielded considerable insights into the nature of gene expression and regulation in eukaryotes. Techniques that identify the 5' end of mRNAs, most notably CAGE, have mapped the promoter landscape across a number of model organisms. Due to the variability of TSS distributions and the transcriptional noise present in datasets, precisely identifying the active promoter(s) for genes from these datasets is not straightforward. TSRchitect allows the user to efficiently identify the putative promoter (the transcription start region, or TSR) from a variety of TSS profiling data types, including both single-end (e.g. CAGE) as well as paired-end (RAMPAGE, PEAT, STRIPE-seq). In addition, (new with version 1.3.0) TSRchitect provides the ability to import aligned EST and cDNA data. Along with the coordiantes of identified TSRs, TSRchitect also calculates the width, abundance and two forms of the Shape Index, and handles biological replicates for expression profiling. Finally, TSRchitect imports annotation files, allowing the user to associate identified promoters with genes and other genomic features. Three detailed examples of TSRchitect's utility are provided in the User's Guide, included with this package.

Author R. Taylor Raborn [aut, cre, cph]
Volker P. Brendel [aut, cph]
Krishnakumar Sridharan [ctb]

Maintainer R. Taylor Raborn <rtraborn@indiana.edu>

Depends R (>= 3.5)

Imports AnnotationHub, BiocGenerics, BiocParallel, dplyr, GenomicAlignments, GenomeInfoDb, GenomicRanges, gtools, IRanges, methods, readxl, Rsamtools (>= 1.14.3), rtracklayer, S4Vectors, SummarizedExperiment, tools, utils

License GPL-3

URL https://github.com/brendelgroup/tsrchitect

BugReports https://github.com/brendelgroup/tsrchitect/issues

Suggests ENCODExplorer, ggplot2, knitr, rmarkdown

biocViews Clustering, FunctionalGenomics, GeneExpression, GeneRegulation, GenomeAnnotation, Sequencing, Transcription

VignetteBuilder knitr

RoxygenNote 7.0.2

NeedsCompilation no

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

git_branch RELEASE_3_14

git_last_commit 3f5e213

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

addAnnotationToTSR **addAnnotationToTSR**

Description

addAnnotationToTSR associates an identified promoter with a given gene, if found upstream and on the same strand within a specified range.

Usage

```
addAnnotationToTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  upstreamDist,
  downstreamDist,
  feature,
  featureColumnID,
  writeTable = TRUE
)

## S4 method for signature
## 'tssObject,
##   character,
##   numeric,
##   numeric,
##   numeric,
##   character,
##   character,
##   logical'
addAnnotationToTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  upstreamDist = 1000,
  downstreamDist = 200,
  feature = "gene",
  featureColumnID = "ID",
  writeTable = TRUE
)
```

Arguments

experimentName	an object of class *tssObject* with occupied data slots *@tsrData* (and/or *@tsrDataMerged*). The *tssObject* must alrady have an annotation attached to the slot *@annotation*, which is provided by either importAnnotationExternal or importAnnotationHub.

tsrSetType	Specifies the type of TSR set to be processed. Options are "replicates" or "merged".
tsrSet	Number of the data set (of type *tsrSetType*) to be processed. (numeric)
upstreamDist	the maximum distance (in bp) upstream of the selected interval necessary to associate a TSR with a given annotation. (numeric)
downstreamDist	the maximum distance (in bp) downstream of the start of the selected interval to associate a TSR with a given annotation. (numeric)
feature	Specifies the feature to be used for annotation (typically "gene" [default] or "mRNA" for GFF3 input); set to "all" if all annotations from the input are to be used. (character)
featureColumnID	Name of the column identifier in the GRanges annotation object. This should be "ID" (default) for GFF3 input or "name" for bed input. (character)
writeTable	logical, specifying whether the output should be written to a tab-delimited file. Defaults to TRUE.

Value

addAnnotationToTSR adds feature annotation to the (merged) @*tsrData* data frame and returns the updated *tssObject*.

Note

An example similar to the this one can be found in the vignette (/inst/doc/TSRchitect.Rmd)

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
tssObjectExample <- addAnnotationToTSR(experimentName=tssObjectExample,
tsrSetType="merged", tsrSet=1, upstreamDist=1000, downstreamDist=200,
feature="transcript", featureColumnID="ID", writeTable=FALSE)
#if the object attached to @annotation is a gff/gff3 file
```

---

addTagCountsToTSR **addTagCountsToTSR**

Description

addTagCountsToTSR adds a matrix of tag counts to a set of identified TSRs

Usage

```
addTagCountsToTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  tagCountThreshold = 1,
  writeTable = TRUE
)

## S4 method for signature 'tssObject,character,numeric,numeric,logical'
addTagCountsToTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  tagCountThreshold = 1,
  writeTable = TRUE
)
```

Arguments

`experimentName`	a S4 object of class *tssObject* containing information in slot *@tssTagData*
`tsrSetType`	specifies the set to be written to file. Options are "replicates" or "merged". (character)
`tsrSet`	number of the dataset to be processed, where 1 corresponds to the first slot, and so on. (numeric)
`tagCountThreshold`	number of tags required at a given TSS position in order to be included in the overall count for a TSR (numeric)
`writeTable`	specifies whether the output should be written to a table. (logical)

Value

a matrix of tag counts and median-normalized tag counts is appended to the data frame of the selected set of identified TSRs in the returned *tssObject*; note that the number of new columns is twice the number of replicates in the sample

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd)

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
tssObjectExample <- addTagCountsToTSR(experimentName=tssObjectExample,
tsrSetType="merged", tsrSet=1, tagCountThreshold=25, writeTable=FALSE)
```

---

bedToTSS **bedToTSS**

Description

bedToTSS extracts TSS information from each attached .bed file in a tssObject object

Usage

```
bedToTSS(experimentName)

## S4 method for signature 'tssObject'
bedToTSS(experimentName)
```

Arguments

experimentName	an S4 object of class tssObject with bed files loaded

Value

produces a GRangesList containing separate GRanges objects for each .bed file contained within *experimentName*, placing them them in the returned *tssObject*.

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd).

---

createSummarizedExperiment
createSummarizedExperiment

Description

createSummarizedExperiment creates and reutrns a *SummarizedExperiment* object from the tag counts on a selected TSR dataset.

Usage

```
createSummarizedExperiment(
  experimentName,
  tsrSetType = "merged",
  tsrSet = 1,
  samplePrefix
)

## S4 method for signature 'tssObject,character,numeric,character'
```

```
createSummarizedExperiment(
  experimentName,
  tsrSetType = "merged",
  tsrSet = 1,
  samplePrefix
)
```

Arguments

experimentName	an S4 object of class *tssObject* containing information in slot *@tssTagData*
tsrSetType	specifies the TSR set to be converted into a *SummarizedExperiment* object. Options are "replicates" or "merged". (character)
tsrSet	number of the dataset to be processed (numeric).
samplePrefix	the prefix (or prefixes) that match the sample identifiers in the tsrData column. (character)

Value

a summarizedExperiment object from the specified TSR data set that is to be written to your working directory.

Note

For more information on the SummarizedExperiment class, please visit https://bioconductor.org/packages/release/bioc/vignet

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
createSummarizedExperiment(tssObjectExample, tsrSetType="merged", tsrSet=1,
samplePrefix=c("sample1","sample2"))
```

---

determineTSR `determineTSR`

Description

`determineTSR` Identifies TSRs from entire TSS datasets as specified.

Usage

```
determineTSR(
  experimentName,
  n.cores,
  tssSetType,
  tssSet,
  tagCountThreshold,
```

```
  clustDist,
  ...
)

## S4 method for signature
## 'tssObject,numeric,character,character,numeric,numeric'
determineTSR(
  experimentName,
  n.cores = 1,
  tssSetType = c("replicates", "merged"),
  tssSet = "all",
  tagCountThreshold = 1,
  clustDist = 20,
  writeTable = FALSE,
  mixedorder = FALSE
)
```

Arguments

experimentName	an object of class *tssObject* containing information in slot *@tssTagData*
n.cores	the number of cores to be used for this job. ncores=1 means serial execution of function calls (numeric)
tssSetType	specifies the set to be clustered. Options are "replicates" or "merged". (character)
tssSet	default is "all"; if a single TSS dataset is desired, specify tssSet number (character)
tagCountThreshold	the number of TSSs required at a given position for it to be considered in TSR identification. (numeric)
clustDist	the maximum distance of TSSs between two TSRs in base pairs. (numeric)
writeTable	specifies whether the output should be written to a table. (logical)
mixedorder	a logical specifying whether the sequence names should be ordered alphanumerically in the output table ("10" following "9" rather than "1"). (logical)

Value

creates a list of GenomicRanges-containing TSR positions in slot *@tsrData* of the returned *tssObject* object

Note

An example similar to this one can be found in the vignette (/inst/doc/TSRchitect.Rmd)

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
tssObjectExample <- determineTSR(experimentName=tssObjectExample, n.cores=1,
tssSetType="replicates", tssSet="1", tagCountThreshold=25, clustDist=20,
```

```
writeTable=FALSE)
```

---

detTSR *detTSR*

Description

An internal function, which is invoked using the user-level function determineTSR that identifies TSRs from the selected tssSet (Internal function)

Usage

```
detTSR(experimentName, tssSetType, tssSet = 1, tagCountThreshold, clustDist)

## S4 method for signature 'tssObject,character,numeric,numeric,numeric'
detTSR(
  experimentName,
  tssSetType,
  tssSet = 1,
  tagCountThreshold = 1,
  clustDist
)
```

Arguments

`experimentName`	- a S4 object of class tssObject containing information in slot tssTagData
`tssSetType`	- specifies the set to be clustered. Options are "replicates" or "merged"
`tssSet`	- number of the dataset to be analyzed
`tagCountThreshold`	- number of TSSs required at a given position
`clustDist`	- maximum distance of TSSs between two TSRs (in base pairs)

Value

via the user-level function determineTSR, creates a list of GenomicRanges objects containing TSR positions in slot 'tsrData' on the tssObject object

getBamDataFirstRead **getBamDataFirstRead**

Description

an accessor function that retrieves the contents of a specified slot "bamDataFirstRead" from a given *tssObject*

Usage

```
getBamDataFirstRead(experimentName, slot)

## S4 method for signature 'tssObject,numeric'
getBamDataFirstRead(experimentName, slot)
```

Arguments

experimentName	an S4 object of class *tssObject*
slot	'numeric' a number corresponding to the slot in "bamDataFirstRead" to be retrieved.

Value

the contents of the specified slot "bamDataFirstRead" are returned

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
example.bamDataFirstRead <-
getBamDataFirstRead(experimentName=tssObjectExample, slot = 1)
example.bamDataFirstRead
```

getBamDataLastRead **getBamDataLastRead**

Description

an accessor function that retrieves the contents of a specified slot "bamDataLastRead" from a given *tssObject*

Usage

```
getBamDataLastRead(experimentName, slot)

## S4 method for signature 'tssObject,numeric'
getBamDataLastRead(experimentName, slot)
```

Arguments

experimentName	an S4 object of class *tssObject*
slot	'numeric' a number corresponding to the slot in "bamDataLastRead" to be retrieved.

Value

the contents of the specified slot "bamDataLastRead" are returned

---

getFileNames **getFileNames**

Description

an accessor function that retrieves the contents of slot "fileNames" from a given *tssObject*

Usage

```
getFileNames(experimentName)

## S4 method for signature 'tssObject'
getFileNames(experimentName)
```

Arguments

experimentName	an S4 object of class *tssObject*

Value

the contents of slot "fileNames" are returned, which are a vector of class "character"

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
example.file.names <- getFileNames(experimentName=tssObjectExample)
example.file.names
```

getTitle **getTitle**

Description

an accessor function that retrieves the contents of slot "title" from a given *tssObject*

Usage

```
getTitle(experimentName)

## S4 method for signature 'tssObject'
getTitle(experimentName)
```

Arguments

experimentName	n S4 object of class *tssObject*

Value

the contents of slot "title" are returned, which are of class "character"

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
example.title <- getTitle(experimentName=tssObjectExample)
example.title
```

getTSRdata **getTSRdata**

Description

an accessor function that retrieves the contents of a specified slot "tsrData"/"tsrDataMerged" from a given *tssObject*

Usage

```
getTSRdata(experimentName, slotType, slot)

## S4 method for signature 'tssObject,character,numeric'
getTSRdata(experimentName, slotType = c("replicates", "merged"), slot)
```

Arguments

experimentName	an S4 object of class *tssObject*
slotType	'character' which data type is to be selected. Either "replicates" (tsrCountData) or "merged" (tsrCountDataMerged)
slot	'numeric' a number corresponding to the slot in "tsrData"/"tsrDataMerged" to be retrieved.

Value

the contents of the selected slot (either "tsrData" or "tsrDataMerged" are returned)

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
ex.tsrData <- getTSRdata(experimentName=tssObjectExample,
slotType="replicates", slot = 1)
ex.tsrData
```

---

getTSScountData **getTSScountData**

Description

an accessor function that retrieves the contents of a specified slot "tssCountData"/"tssCountDataMerged" from a given *tssObject*

Usage

```
getTSScountData(experimentName, slotType, slot)

## S4 method for signature 'tssObject,character,numeric'
getTSScountData(experimentName, slotType = c("replicates", "merged"), slot)
```

Arguments

experimentName	an S4 object of class *tssObject*
slotType	'character' which data type is to be selected. Either "replicates" (tssCountData) or "merged" (tssCountDataMerged)
slot	'numeric' a number corresponding to the slot in "tssCountData"/"tssCountDataMerged" to be retrieved.

Value

the contents of the selected slot (either "tssCountData" or "tssCountDataMerged" are returned)

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
ex.tssCountData <- getTSScountData(experimentName=tssObjectExample,
slotType="replicates", slot = 1)
ex.tssCountData
```

---

getTSStagData **getTSStagData**

Description

an accessor function that retrieves the contents of a specified slot "tssTagData" from a given *tssObject*

Usage

```
getTSStagData(experimentName, slot)

## S4 method for signature 'tssObject,numeric'
getTSStagData(experimentName, slot)
```

Arguments

experimentName	an S4 object of class *tssObject*
slot	'numeric' a number corresponding to the slot in "tssTagData" to be retrieved.

Value

the contents of the specified slot "tssTagData" are returned

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
example.tssTagData <- getTSStagData(experimentName=tssObjectExample, slot=1)
example.tssTagData
```

gsq2bed **gsq2bed**

Description

gsq2bed converts aligned cDNA data (in .gsq format) to BED format, extracting the 5'-most base.

Usage

```
gsq2bed(gsqFile, outfile)

## S4 method for signature 'character,character'
gsq2bed(gsqFile, outfile)
```

Arguments

gsqFile	a path to the gsq (GeneSeqer) output file (class character)
outfile	the name (class character) of the BED file to be written (default is "gsqOut.bed")

Value

a BED file containing a list of the 5'-most base from each of the alignments contained in the GeneSeqer (.gsq) output file is written to the user's working directory.

Examples

```
extdata.dir <- system.file("extdata", package="TSRchitect")
tssObjectExample <- gsq2bed(gsqFile=paste(extdata.dir,"AtEST.gsq",sep="/"),
                            outfile="")
```

importAnnotationExternal **importAnnotationExternal**

Description

importAnnotationExternal imports an annotation from an external file and attaches it to the *tssObject*

Usage

```
importAnnotationExternal(experimentName, fileType, annotFile)

## S4 method for signature 'tssObject,character,character'
importAnnotationExternal(
  experimentName,
  fileType = c("bed", "gff", "gff3"),
  annotFile
)
```

Arguments

`experimentName`	- an S4 object of class *tssObject* that contains information about the experiment
`fileType`	- the format of the annotation file to be imported. Must be one of: "bed", "gff" or "gff3".
`annotFile`	- a path (full or relative) to the annotation file to be imported.

Value

fills the slot *@annotation* in the returned *tssObject* with a GRanges object contining a parsed annotation file of the selected type.

Note

`importAnnotationExternal` makes use of three functions from the *rtracklayer* package: `import.bed`, `import.gff`, and `import.gff3`.

An example similar to the one provided can be found in *Example 2* from the vignette (run `TSRchitectUsersGuide()` to view the documentation).

To import directly from the AnnotationHub client, please refer to `importAnnotationHub`.

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
extdata.dir <- system.file("extdata", package="TSRchitect")
annotation <- dir(extdata.dir, pattern="\\.gff3$", full.names=TRUE)
tssObjectExample <- importAnnotationExternal(experimentName=tssObjectExample,
fileType="gff3", annotFile=annotation)
```

`importAnnotationHub` **importAnnotationHub**

Description

imports an annotation directly from AnnotationHub and attaches it to the *tssObject*

Usage

```
importAnnotationHub(experimentName, provider, annotType, species, annotID)

## S4 method for signature 'tssObject,character,character,character,character'
importAnnotationHub(experimentName, provider, annotType, species, annotID)
```

Arguments

experimentName	- an S4 object of class *tssObject* that contains information about the experiment
provider	- 'character' the source of the annotation in AnnotationHub (e.g. 'gencode')
annotType	- 'character' the format of the annotationHub annotation to be imported. Using 'gff' will find annotations in both gff or gff3 formats
species	- 'character' the species identifier in AnnotationHub (e.g. 'human')
annotID	- 'character' the AnnotationHub identifier to be retrieved

Value

fills the slot @*annotation* in the returned *tssObject* with an AnnotationHub record. The record retrieved must be an object of class GRanges.

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd)

Please consult the available records in AnnotationHub beforehand using `AnnotationHub-class`

---

inputToTSS **inputToTSS**

Description

`inputToTSS` extracts TSS information from each attached .bam or .bed file in a tssObject object

Usage

```
inputToTSS(experimentName)

## S4 method for signature 'tssObject'
inputToTSS(experimentName)
```

Arguments

experimentName	an S4 object of class tssObject with bam files loaded

Value

produces a GRangesList containing separate GRanges objects for each .bam file contained within *experimentName*, placing them them in the returned *tssObject*.

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd).

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
tssObjectExample <- inputToTSS(experimentName=tssObjectExample)
```

`loadTSSobj` **loadTSSobj**

Description

`loadTSSobj` processes alignment files in .bam or .bed formats from the local directory supplied.

Usage

```
loadTSSobj(experimentTitle, inputDir, ...)

## S4 method for signature 'character,character'
loadTSSobj(
  experimentTitle,
  inputDir,
  n.cores = 1,
  isPairedBAM = TRUE,
  isPairedBED = TRUE,
  sampleSheet = NA,
  sampleNames = NA,
  replicateIDs = NA
)
```

Arguments

`experimentTitle`	a descriptive title for the experiment (character).
`inputDir`	path to the directory containing the alignment files (in either .bam or .bed formats) (character). Note that all the paths to all files in *inputDir* with the extension .bam or .bed will be imported with this function.
`n.cores`	the number of cores to be used for this job. (numeric)
`isPairedBAM`	if the input is in BAM format, specifies whether the TSS profiling experiment is paired-end (if TRUE) or single-end (if FALSE). Set to TRUE by default. (logical)

isPairedBED	if the input is in BED format, specifies whether the TSS profiling experiment is paired-end (if TRUE) or single-end (if FALSE). Set to TRUE by default. (logical) Note: if TRUE, the input data must be in bedpe format, as described here: http://bedtools.readthedocs.io/en/latest/content/general-usage.html
sampleSheet	file providing TSS sample information; if provided, sampleNames and replicateIDs are ignored; input format is tab-delimited 3-column rows with sample name, replicate ID, and sample data file name; the file has to include column headers "SAMPLE ReplicateID FILE" and can be either tab-delimited or an Excel spreadsheet (file extension ".xls" or "xlsx" (character)
sampleNames	unique labels of class character for each TSS sample within the experiment (character).
replicateIDs	identifiers indicating which samples are biological replicates. Note that loadTSSobj imports alignment data in ascending alphanumeric order, so the arguments to replicateIDs must be arranged in this order also so that they directly correspond to the intended file. Thus replicateIDs must be ordinal values in consecutive order without gaps (e.g. 1, 2, 3, ...) (numeric).

Value

loadTSSobj fills the slot *bamDataFirstRead* and/or *bedData* on the returned *tssObject* with GAlignments objects (for .bam files), or GRanges objects (for .bed files).

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd).

All files found in *inputDir* will be retrieved and written in ascending alphanumeric order to the *@fileNamesBAM* and/or *@fileNamesBED* slot(s) on the *tssObject* that is created.

Examples

```
extdata.dir <- system.file("extdata/bamFiles", package="TSRchitect")
test.Obj <- loadTSSobj(experimentTitle="Code example", inputDir=extdata.dir,
n.cores=2, isPairedBAM=TRUE, sampleNames=c("sample1-rep1", "sample1-rep2",
"sample2-rep1","sample2-rep2"), replicateIDs=c(1,1,2,2))
```

---

makeGRangesFromTSR **makeGRangesFromTSR**

Description

makeGRangesFromTSR creates a GRanges object from a specified TSR data set

Usage

```
makeGRangesFromTSR(experimentName, tsrSetType, tsrSet = 1)

## S4 method for signature 'tssObject,character,numeric'
makeGRangesFromTSR(experimentName, tsrSetType, tsrSet = 1)
```

Arguments

experimentName	an S4 object of class *tssObject* containing information in slot *@tssTagData*
tsrSetType	specifies the set to be written to file. Options are "replicates" or "merged". (character)
tsrSet	number of the dataset to be processed (numeric).

Value

An object of class *GRanges* containing the specified TSR data set. Headers include 'seqnames', 'ranges' (including start and end), 'strand', 'name' (TSR ID) and 'score' (Shape Index/SI) value

Note

For more information on the GRanges class, please visit: http://web.mit.edu/~r/current/arch/i386_linux26/lib/R/library/Genor
class.html

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
makeGRangesFromTSR(experimentName=tssObjectExample, tsrSetType="replicates",
tsrSet=1)
```

---

mergeSampleData **mergeSampleData**

Description

mergeSampleData combines samples from multiple TSS experiments into a single GRanges object

Usage

```
mergeSampleData(experimentName, n.cores, tagCountThreshold)

## S4 method for signature 'tssObject,numeric,numeric'
mergeSampleData(experimentName, n.cores = 1, tagCountThreshold = 1)
```

Arguments

experimentName	an S4 object of class *tssObject* that contains information about the experiment.
n.cores	the number of cores to be used for this job. ncores=1 means serial execution of function calls (numeric)
tagCountThreshold	the number of TSSs required at a given position for it to be considered in sample data merging. (numeric) Note: Merged data sets can become very large when the tagCountThreshold is set low (leading to inclusion of a lot of "noise" and resulting in long execution times in the necessary TSS position ordering step).

Value

tssCountData datasets are merged (according to the *sampleIDs*) and put in the tssCountDataMerged slot in the returned *tssObject*.

Note

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd).

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
tssObjectExample <- mergeSampleData(experimentName=tssObjectExample,
                                    n.cores=1, tagCountThreshold=1)
```

`processTSS` **processTSS**

Description

`processTSS` calulates the number of observed reads at a given TSS coordinate across an entire dataset.

Usage

```
processTSS(experimentName, n.cores, tssSet, writeTable)

## S4 method for signature 'tssObject,numeric,character,logical'
processTSS(experimentName, n.cores = 1, tssSet = "all", writeTable = FALSE)
```

Arguments

`experimentName`	an S4 object of class *tssObject* containing information in slot *@tssTagData*
`n.cores`	the number of cores to be used for this job. n.cores=1 means serial execution of function calls (numeric)
`tssSet`	default is "all"; to select a single *tssSet*, specify it (as character)
`writeTable`	specifies whether the output should be written to a file. (logical)

Value

Creates a list of GenomicRanges containing TSS positions in slot *tssTagData* of the returned *tssObject*.

Note

Note that the *tssSet* parameter must be of class *character*, even when selecting an individual dataset.

An example similar to the one provided can be found in the vignette (/inst/doc/TSRchitect.Rmd).

Examples

```
load(system.file("extdata", "tssObjectExample.RData",
package="TSRchitect"))
tssObjectExample <- processTSS(experimentName=tssObjectExample, n.cores=1,
tssSet="all", writeTable=FALSE)
```

`TSRchitectUsersGuide` **TSRchitectUsersGuide**

Description

Opens the TSRchitect User's Guide in a pdf viewer on the user's system.

Usage

```
TSRchitectUsersGuide(view = TRUE)
```

Arguments

`view`	(logical) if TRUE (default) the User's Guide is opened in the local pdf viewer. If FALSE then the full path to the User's Guide is returned.

Value

if view=FALSE, then the full path to the User's Guide is returned.

Examples

```
myPath <- TSRchitectUsersGuide(view=FALSE)
```

`tssObject` **tssObject**

Description

S4 constructor function for *tssObject*

Usage

```
tssObject(
  title = NA,
  bamDataFirstRead = NA,
  bamDataLastRead = NA,
  bedData = NA
)
```

Arguments

title	'character' A short descriptive title for the experiment. Is set to NA by default.
bamDataFirstRead	'list' the name of a list of GAlignments objects (originating from .bam files) in the workspace. Set to NA by default.
bamDataLastRead	'list' the name of a list of GAlignments objects (originating from paired-read .bam files) in the workspace. Set to NA by default.
bedData	'list' the name of a list of GRanges or Pairs objects (originating from .bed files) in the workspace. Set to NA by default.

Value

a new *tssObject* is returned to the user's workspace.

Examples

```
new.tssObj <- tssObject(title="Example")
```

---

writeTSR **writeTSR**

Description

writeTSR writes identified TSRs from a specified data set to a file in selected format

Usage

```
writeTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  tsrLabel = "TSR_",
  mixedorder = FALSE,
  fileType = "tab"
)

## S4 method for signature
## 'tssObject,character,numeric,character,logical,character'
writeTSR(
  experimentName,
  tsrSetType,
  tsrSet = 1,
  tsrLabel = "TSR_",
  mixedorder = FALSE,
  fileType = "tab"
)
```

Arguments

experimentName	an S4 object of class *tssObject* containing information in slot *@tssTagData*
tsrSetType	specifies the set to be written to file. Options are "replicates" or "merged". (character)
tsrSet	number of the dataset to be processed (numeric).
tsrLabel	specifies the label to be used in the name column of BED format output
mixedorder	a logical specifying whether the sequence names should be ordered alphanumerically ("10" following "9" rather than "1"). (logical)
fileType	the format of the file to be written. Possible choices are "tab" for tab-delimited output, "bed" for BED format, "bedGraph" for BedGraph format, and "gff" for for GFF3 format (character).

Value

A table containing the specified TSR data set that is to be written to your working directory.

Note

The .bed file written adheres to the standard six-column BED format, while "tab" format is identical to that of the data.frames containing TSR data.

For more information on the BED format, please visit https://genome.ucsc.edu/FAQ/FAQformat#format1

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
writeTSR(experimentName=tssObjectExample, tsrSetType="replicates",
         tsrSet=1, tsrLabel="TSRsample1_", mixedorder=FALSE, fileType="tab")
```

---

writeTSS — writeTSS

Description

writeTSS writes identified TSSs from a specified data set to a file in either tab or BED formats

Usage

```
writeTSS(
  experimentName,
  tssSetType,
  tssSet = 1,
  tssLabel = "TSS_",
  mixedorder = FALSE,
  fileType = "tab"
)
```

```
## S4 method for signature
## 'tssObject,character,numeric,character,logical,character'
writeTSS(
  experimentName,
  tssSetType,
  tssSet = 1,
  tssLabel = "TSS_",
  mixedorder = FALSE,
  fileType = "tab"
)
```

Arguments

experimentName	an S4 object of class *tssObject* containing information in slot *@tssCountData*
tssSetType	specifies the set to be written to file. Options are "replicates" or "merged". (character)
tssSet	number of the dataset to be processed (numeric).
tssLabel	specifies the label to be used in the name column of BED format output
mixedorder	a logical specifying whether the sequence names should be ordered alphanumerically ("10" following "9" rather than "1"). (logical)
fileType	the format of the file to be written. Possible choices are "tab" for tab-delimited output, "bed" for BED format, and "bedGraph" for BedGraph format (character). In case "bedGraph," three output files are produced: *_PLUS.bedGraph and *_MINUS.bedGraph, which record the number of TSS tags per position on plus and minus strand separately; and a combined *.bedGraph file which lists plus strand tag counts as positive numbers, minus strand tag counts as negative numbers, and positions with both plus and minus strand tag counts display the tag count corresponding to the highest absolute value.

Value

A table containing the specified TSS data set that is to be written to your working directory.

Note

The .bed file written adheres to the standard six-column BED format, while "tab" format is identical to that of the data.frames containing TSS data.

For more information on the BED format, please visit https://genome.ucsc.edu/FAQ/FAQformat#format1

Examples

```
load(system.file("extdata", "tssObjectExample.RData", package="TSRchitect"))
writeTSS(experimentName=tssObjectExample, tssSetType="replicates",
         tssSet=1, tssLabel="TSSsample1_", mixedorder=FALSE, fileType="tab")
```

Index