

Package ‘TSRchitect’

November 23, 2017

Version 1.4.0

Date 2017-10-17

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). In addition, (new with version 1.2.0) TSRchitect provides the ability to import aligned EST and cDNA data. Along with the coordinates of identified TSRs, TSRchitect also calculates the width, abundance and 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 (>= 3.4)

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

License GPL-3

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

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

Suggests ENCODEExplorer, ggplot2, knitr, rmarkdown

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

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

addAnnotationToTSR	2
addTagCountsToTSR	4
bedToTSS	5
cDNAtoBED	5
createSummarizedExperiment	6
determineTSR	7
defTSR	8
getBamData	8
getFileNames	9
getTitle	10
getTSRdata	10
getTSScountData	11
getTSStagData	12
importAnnotationExternal	13
importAnnotationHub	14
inputToTSS	14
loadTSSobj	15
makeGRangesFromTSR	16
mergeSampleData	17
processTSS	18
TSRchitectUsersGuide	19
tssObject	19
writeTSR	20

Index **21**

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 <i>tssObject</i> with occupied data slots <i>@tsrData</i> (and/or <i>@tsrDataMerged</i>). The <i>tssObject</i> must already have an annotation attached to the slot <i>@annotation</i> , 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 <i>tsrSetType</i>) 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 TSSs required at a given position (numeric)

writeTable specifies whether the output should be written to a table. (logical)

Value

a matrix of tag counts (where the number of columns will equal the number of replicates in the sample) is appended to the data frame of the selected set of identified TSRs 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 <- 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*).

cDNAtoBED

cDNAtoBED**Description**

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

Usage

```
cDNAtoBED(gsqFile, fileName = "gsqOut.bed")

## S4 method for signature 'character,character'
cDNAtoBED(gsqFile, fileName = "gsqOut.bed")
```

Arguments

gsqFile a path to the gsq (GeneSeqer) output file (class character)
 fileName 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

```
tssObjectExample <- cDNAtoBED(gsqFile="gsq.At_GenomicSeq.fasta", fileName="testOut.bed")
```

```
createSummarizedExperiment
      createSummarizedExperiment
```

Description

createSummarizedExperiment creates and returns 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 <i>tssObject</i> containing information in slot <i>@tssTagData</i>
tsrSetType	specifies the TSR set to be converted into a <i>SummarizedExperiment</i> 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 <i>tsrData</i> 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/vi>

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, tsrSetType, tssSet, tagCountThreshold,
             clustDist, writeTable = FALSE)
```

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

Arguments

experimentName	an object of class <i>tssObject</i> containing information in slot <i>@tssTagData</i>
n.cores	the number of cores to be used for this job. ncores=1 means serial execution of function calls (numeric)
tsrSetType	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)

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,
                                tsrSetType="replicates", tssSet="1", tagCountThreshold=25, clustDist=20,
                                writeTable=FALSE)
```

detTSR	<i>detTSR</i>
--------	---------------

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, tsrSetType, tssSet = 1, tagCountThreshold, clustDist)
```

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

Arguments

`experimentName` - a S4 object of class `tssObject` containing information in slot `tssTagData`
`tsrSetType` - 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

getBamData	getBamData
------------	-------------------

Description

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

Usage

```
getBamData(experimentName, slot)
```

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

Arguments

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

Value

the contents of the specified slot "bamData" are returned

Examples

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

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` an 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
```

```
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 containing 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 (`/inst/doc/TSRchitect.Rmd`)

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](#)

```
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 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, isPairedBAM = FALSE,
  isPairedBED = FALSE, sampleNames, replicateIDs)
```

```
## S4 method for signature 'character,character,ANY,ANY,character,numeric'
loadTSSObj(experimentTitle,
  inputDir, isPairedBAM = FALSE, isPairedBED = FALSE, sampleNames,
  replicateIDs)
```

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.

isPairedBAM

if the input is in BAM format, specifies whether the TSS profiling experiment is paired-end (if TRUE) or single-end (if FALSE) (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 FALSE 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>

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 (numeric).

Value

`loadTSSobj` fills the slot `bamData` 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", package="TSRchitect")
test.Obj <- loadTSSobj(experimentTitle="Code example", inputDir=extdata.dir,
  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/Gclass.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)

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

Arguments

experimentName an S4 object of class *tssObject* that contains information about the experiment.

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)
```

 processTSS

processTSS

Description

processTSS calculates 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 <i>tssObject</i> containing information in slot <i>@tssTagData</i>
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 <i>tssSet</i> , 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, bamData = NA, bedData = NA)
```

Arguments

`title` 'character' A short descriptive title for the experiment. Is set to NA by default.
`bamData` 'list' the name of a list of [GAlignments](#) objects (originating from .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 either tab or BED formats

Usage

```
writeTSR(experimentName, tsrSetType, tsrSet = 1, fileType = "tab")
```

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

Arguments

experimentName	an S4 object of class <i>tssObject</i> containing information in slot <i>@tssTagData</i>
tsrSetType	specifies the set to be written to file. Options are "replicates" or "merged". (character)
tsrSet	number of the dataset to be processed (numeric).
fileType	the format of the file to be written. Possible choices are "tab" for tab-delimited output and "bed" for BED 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, fileType="tab")
```

Index

*Topic **methods**

- getBamData, 8
 - getFileNames, 9
 - getTitle, 10
 - getTSRdata, 10
 - getTSScountData, 11
 - getTSStagData, 12
- addAnnotationToTSR, 2
- addAnnotationToTSR, tssObject, character, numeric, numeric-method (addAnnotationToTSR), 2
- addTagCountsToTSR, 4
- addTagCountsToTSR, tssObject, character, numeric, numeric-method (addTagCountsToTSR), 4
- AnnotationHub, 14
- bedToTSS, 5
- bedToTSS, tssObject-method (bedToTSS), 5
- cDNAtoBED, 5
- cDNAtoBED, character, character-method (cDNAtoBED), 5
- createSummarizedExperiment, 6
- createSummarizedExperiment, tssObject, character, numeric, character-method (createSummarizedExperiment), 6
- determineTSR, 7
- determineTSR, tssObject, numeric, character, character-method (determineTSR), 7
- detTSR, 8
- detTSR, tssObject, character, numeric, numeric, numeric-method (detTSR), 8
- GAlignments, 16, 19
- GenomicRanges, 7, 18
- getBamData, 8
- getBamData, tssObject, numeric-method (getBamData), 8
- getFileNames, 9
- getFileNames, tssObject-method (getFileNames), 9
- getTitle, 10
- getTitle, tssObject-method (getTitle), 10
- getTSRdata, 10
- getTSRdata, tssObject, character, numeric-method (getTSRdata), 10
- getTSScountData, 11
- getTSScountData, tssObject, character, numeric-method (getTSScountData), 11
- getTSStagData, 12
- getTSStagData, tssObject, numeric-method (getTSStagData), 12
- GRanges, 3, 5, 13–17, 19
- GRangesList, numeric, character, character, logical-method
- import.bed, 13
- import.gff, 13
- import.gff3, 13
- importAnnotationExternal, 3, 13
- importAnnotationExternal, tssObject, character, character-method (importAnnotationExternal), 13
- importAnnotationHub, 3, 14
- importAnnotationHub, tssObject, character, character, character-method (importAnnotationHub), 14
- inputToTSS, 14
- inputToTSS, tssObject-method (inputToTSS), 14
- loadTSSobj, 15
- loadTSSobj, character, character, ANY, ANY, character, numeric-method (loadTSSobj), 15
- makeGRangesFromTSR, 16
- makeGRangesFromTSR, tssObject, character, numeric-method (makeGRangesFromTSR), 16
- mergeSampleData, 17
- mergeSampleData, tssObject-method (mergeSampleData), 17
- Pairs, 19
- processTSS, 18
- processTSS, tssObject, numeric, character, logical-method (processTSS), 18
- TSRchitectUsersGuide, 19
- tssObject, 19
- writeTSR, 20
- writeTSR, tssObject, character, numeric, character-method (writeTSR), 20