Package: TSSr (via r-universe)

November 25, 2026

Type Package

Title TSS sequencing data analysis

Version 1.0.0

Date 2026-08-21

Description TSSr package provides a comprehensive workflow on TSS data starts from identification of accurate TSS locations, clustering TSSs within small genomic regions corresponding to core promoters, and transcriptional activity quantifications, as well as specialized downstream analyses including core promoter shape, cluster annotation, gene differential expression, core promoter shift. TSSr can take multiple formats of files as input, such as Binary Sequence Alignment Map (BAM) files (single-ended or paired-ended), Browser Extension Data (bed) files, BigWig files, ctss files or tss tables. TSSr also generates various types of TSS or core promoter track files which can be visualized in the UCSC Genome Browser or Integrative Genomics Viewer (IGV). TSSr also exports downstream analyses result tables and plots. Multiple cores are supported on Linux or Mac platforms.

License MIT + file LICENSE

URL https://github.com/Linlab-slu/TSSr

BugReports https://github.com/Linlab-slu/TSSr/issues

Depends R (>= 4.5.0)

Imports BiocGenerics (>= 0.36.1), GenomeInfoDb (>= 1.26.7), GenomicFeatures (>= 1.42.3), GenomicRanges (>= 1.42.0), IRanges (>= 2.24.1), Rsamtools (>= 2.6.0), cigarillo (>= 0.99.2), data.table (>= 1.14.0), dplyr (>= 1.0.7), ggplot2 (>= 3.3.5), grDevices (>= 4.0.3), graphics (>= 4.0.3), methods (>= 4.0.3), parallel (>= 4.0.3), rtracklayer (>= 1.50.0), stats (>= 4.0.3), stringr (>= 1.4.0), txdbmaker (>= 1.0.0), utils (>= 4.0.3)

Suggests BSgenome.Scerevisiae.UCSC.sacCer3, DESeq2 (>= 1.30.1), Gviz (>= 1.34.1), calibrate (>= 1.7.7), ggfortify (>= 0.4.12), knitr, pkgdown, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData FALSE

NeedsCompilation no

RoxygenNote 7.3.3

biocViews Software, Transcription, Coverage, GeneExpression, GeneRegulation, PeakDetection, DataImport, DataRepresentation, Transcriptomics, Sequencing, Annotation, GenomeBrowsers, Normalization, Preprocessing, Visualization, DifferentialExpression, Alignment, Clustering

Config/pak/sysreqs
make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:58 UTC

RemoteUrl https://github.com/bioc/TSSr

RemoteRef RELEASE_3_24

RemoteSha 92839382338d0ceecc55b7a794ad46e2cb8ae734

Contents

annotateCluster . . . 3
callEnhancer . . . 4
clusterTSS . . . 5
consensusCluster . . . 6
deGene . . . 7
exampleTSSr . . . 9
exportClustersTable . . . 10
exportClustersToBed . . . 10
exportDETable . . . 11
exportEnhancerTable . . . 12
exportShapeTable . . . 13
exportShiftTable . . . 13
exportTSStable . . . 14
exportTSStoBedgraph . . . 15
filterTSS . . . 16
getTSS . . . 17
mergeSamples . . . 18
normalizeTSS . . . 19
plotCorrelation . . . 19
plotDE . . . 20
plotInterQuantile . . . 21
plotShape . . . 22
plotTSS . . . 22

plotTssPCA 24
shapeCluster 24
shiftPromoter 26
show,TSSr-method 27
TSSr-accessors 27
TSSr-class 30

Index **33**

`annotateCluster` *Annotate clusters with GFF annotation file.*

Description

Annotates clusters with gene or transcript names from GFF annotation file.

Usage

```
annotateCluster(object,clusters = "consensusClusters",filterCluster = TRUE
, filterClusterThreshold = 0.02, annotationType = "genes",upstream=1000
, upstreamOverlap = 500,downstream = 0)

## S4 method for signature 'TSSr'
annotateCluster(
  object,
  clusters = "consensusClusters",
  filterCluster = TRUE,
  filterClusterThreshold = 0.02,
  annotationType = "genes",
  upstream = 1000,
  upstreamOverlap = 500,
  downstream = 0
)
```

Arguments

`object`	A TSSr object containing the selected cluster tables and either a populated `refTable` slot or an existing GFF file in `refSource`.
`clusters`	Clusters to be annotated: "consensusClusters" or "tagClusters". Default is "consensusClusters".
`filterCluster`	Logical indicating whether clusters downstream of a highly expressed cluster are filtered. Setting filterCluster as "TRUE" would reduce weak clusters brought from recapping, transcriptional or sequencing noise. Default is TRUE.
`filterClusterThreshold`	Ignore downstream clusters if signal < filterClusterThreshold*the strongest clusters within the same gene promoter region. Default value = 0.02.

annotationType	Specify annotation feature to be associated with: "genes" or "transcripts". Default is "genes".
upstream	Upstream distance to the start position of annotation feature. Default value = 1000.
upstreamOverlap	Upstream distance to the start position of annotation feature if overlapped with the upstream neighboring feature. Default value = 500.
downstream	Downstream distance to the start position of annotation feature. Default value = 0. Note: if annotationType == "transctipt" or the gene annotations start from transcription start sites (TSSs), the recommended value = 500.

Value

A modified TSSr object with updated `assignedClusters`, `unassignedClusters`, and optionally `filteredClusters` slots. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- annotateCluster(
    exampleTSSr, clusters = "consensusClusters", filterCluster = TRUE
)
head(assignedClusters(exampleTSSr, sample = "control"))
```

---

callEnhancer *Identification of enhancers*

Description

Calculates enhancer candidates, which are characterized by bidirectional clusters as described in Andersson et al. 2014.

Usage

```
callEnhancer(object, flanking = 400, dis2gene = 2000)

## S4 method for signature 'TSSr'
callEnhancer(object, flanking = 400, dis2gene = 2000)
```

Arguments

object	A TSSr object.
flanking	The flanking region range where bidirectional clusters composing a enhancer candidate. Default is 400.
dis2gene	The minimum distance to the main annotated core promoter of genes. Default is 2000.

Value

A modified TSSr object with updated `enhancers` slot. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- callEnhancer(exampleTSSr)
head(enhancers(exampleTSSr, sample = "control"))
```

---

clusterTSS *Cluster TSSs into tag clusters*

Description

Clusters TSSs within small genomic regions into tag clusters (TCs) using "peakclu" method. "peakclu" method is an implementation of peak-based clustering. The minimum distance of two neighboring peaks can be specified.

Usage

```
clusterTSS(object, method = "peakclu", peakDistance=100,extensionDistance=30
, localThreshold = 0.02,clusterThreshold = 1, useMultiCore=FALSE, numCores=NULL)

## S4 method for signature 'TSSr'
clusterTSS(
  object,
  method = "peakclu",
  peakDistance = 100,
  extensionDistance = 30,
  localThreshold = 0.02,
  clusterThreshold = 1,
  useMultiCore = FALSE,
  numCores = NULL
)
```

Arguments

`object`	A TSSr object
`method`	Clustering method to be used for clustering: "peakclu". Default is "peakclu".
`peakDistance`	Minimum distance of two neighboring peaks. Default value = 100.
`extensionDistance`	Maximal distance between peak and its neighboring TSS or two neighboring TSSs to be grouped in the same cluster. Default value = 30.
`localThreshold`	Ignore downstream TSSs with signal < localThreshold*peak within clusters, which is used to filter TSS signals brought from possible recapping events, or sequencing noise. Default value = 0.02.

clusterThreshold
: Ignore clusters if signal < clusterThreshold. Default value = 1.

useMultiCore
: Logical indicating whether multiple cores are used (TRUE) or not (FALSE). Default is FALSE.

numCores
: Number of cores are used in clustering step. Used only if useMultiCore = TRUE. Default is NULL.

Value

A modified TSSr object with updated `tagClusters` slot. The input object is not modified; assign the returned object to retain the changes.

Examples

```
example_input <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr",
    mustWork = TRUE
)
example_object <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = example_input,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2)
)
example_object <- getTSS(example_object)
example_object <- mergeSamples(example_object)
example_object <- normalizeTSS(example_object)
example_object <- clusterTSS(
    example_object, method = "peakclu", clusterThreshold = 1,
    useMultiCore = FALSE
)
head(tagClusters(example_object, sample = "control"))
```

---

consensusCluster *Make consensus clusters across multiple samples.*

Description

Makes consensus clusters from multiple samples in TSSr object and calculates inter-quantile positions within consensus clusters for each sample.

Usage

```
consensusCluster(object, dis = 50
, useMultiCore=FALSE, numCores = NULL)
```

```
## S4 method for signature 'TSSr'
consensusCluster(object, dis = 50, useMultiCore = FALSE, numCores = NULL)
```

Arguments

object	A TSSr object.
dis	Minimum distance between two peaks to be aggregated together into the same consensus cluster.
useMultiCore	Logical indicating whether multiple cores are used (TRUE) or not (FALSE). Default is FALSE.
numCores	Number of cores are used in clustering step. Used only if useMultiCore = TRUE. Default is NULL.

Value

A modified TSSr object with updated `consensusClusters` slot. The input object is not modified; assign the returned object to retain the changes.

Examples

```
example_input <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr",
    mustWork = TRUE
)
example_object <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = example_input,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2)
)
example_object <- getTSS(example_object)
example_object <- mergeSamples(example_object)
example_object <- normalizeTSS(example_object)
example_object <- clusterTSS(example_object, useMultiCore = FALSE)
example_object <- consensusCluster(
    example_object, useMultiCore = FALSE
)
head(consensusClusters(example_object, sample = "control"))
```

---

deGene *Analysis of gene differential expression.*

Description

Analyzes gene-level differential expression using DESeq2 method (Love et al., 2014).

Usage

```
deGene(object,comparePairs=list(c("control","treat")), pval = 0.01,
 useMultiCore=FALSE, numCores = NULL, fitType = "parametric")

## S4 method for signature 'TSSr'
deGene(
  object,
  comparePairs = list(c("control", "treat")),
  pval = 0.01,
  useMultiCore = FALSE,
  numCores = NULL,
  fitType = "parametric"
)
```

Arguments

`object`	A TSSr object.
`comparePairs`	Specified list of sample pairs for comparison with DESeq2 method.
`pval`	Genes with an adjusted p value below `pval` are stored in the significant-results table. Default is 0.01.
`useMultiCore`	Logical indicating whether multiple cores are used (TRUE) or not (FALSE). Default is FALSE.
`numCores`	Number of cores are used in clustering step. Used only if useMultiCore = TRUE. Default is NULL.
`fitType`	Dispersion trend fitting method passed to `DESeq2::DESeq()`. One of `"parametric"`, `"local"`, or `"mean"`. Default is `"parametric"`.

Details

Requires the suggested package **DESeq2**.

Value

A modified TSSr object with updated `DEtables` and `TAGtables` slots. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- deGene(
    exampleTSSr, comparePairs = list(c("control", "treat")),
    pval = 0.01, useMultiCore = FALSE, fitType = "mean"
)
head(DEtables(
    exampleTSSr,
    comparison = "control_VS_treat",
    result = "all"
))
```

exampleTSSr *TSSr example data*

Description

A pre-processed TSSr object containing a subset of the CAGE dataset from Lu and Lin, Genome Research 2019 Jul;29(7):1198-1210.

Usage

```
data(exampleTSSr)
```

Format

An S4 object of class TSSr containing:

genomeName BSgenome reference name

inputFiles An empty character vector because source BAM files are not bundled

inputFilesType The original input type, retained as provenance

sampleLabels Sample identifiers

TSSrawMatrix Raw TSS count matrix

TSSprocessedMatrix Processed TSS count matrix

refSource An empty character vector; no machine-specific GFF path is stored

refTable The bundled chromosome-I reference annotation

Details

This example object contains TSS data that has already been processed through the TSSr workflow. The original BAM files are not included, so its `inputFiles` and `refSource` slots are empty and `getTSS()` cannot be called on this object. The `inputFilesType = "bam"` value is retained only as provenance, and the bundled annotation is stored in `refTable`. However, all downstream analysis functions (such as `mergeSamples`, `normalizeTSS`, `filterTSS`, `clusterTSS`, etc.) can be used with this object.

Source

https://genome.cshlp.org/content/29/7/1198.short

Examples

```
data(exampleTSSr)
exampleTSSr
```

exportClustersTable *Export cluster tables*

Description

Export cluster tables to text files.

Usage

```
exportClustersTable(object, data = "assigned")

## S4 method for signature 'TSSr'
exportClustersTable(object, data = "assigned")
```

Arguments

object	A TSSr object.
data	Specify which cluster data will be exported: "tagClusters", "consensusClusters", "assigned", "unassigned". Default is "assigned".

Value

cluster tables

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportClustersTable(exampleTSSr, data = "tagClusters")
setwd(oldwd)
```

exportClustersToBed *Creating bed files of clusters*

Description

Creates bed files of clusters.

Usage

```
exportClustersToBed(object, data = "consensusClusters", assigned = TRUE)

## S4 method for signature 'TSSr'
exportClustersToBed(object, data = "consensusClusters", assigned = TRUE)
```

Arguments

object	A TSSr object.
data	Specify which data will be exported: "tagClusters" or "consensusClusters". Default is "consensusClusters".
assigned	Specify which consensus clusters will be exported. Used only if data = "consensusClusters. Default is TRUE.

Value

bed files of clusters

Examples

```
example_input <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr",
    mustWork = TRUE
)
example_object <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = example_input,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2)
)
example_object <- getTSS(example_object)
example_object <- mergeSamples(example_object)
example_object <- normalizeTSS(example_object)
example_object <- clusterTSS(example_object, useMultiCore = FALSE)
oldwd <- setwd(tempdir())
exportClustersToBed(example_object, data = "tagClusters")
setwd(oldwd)
```

---

exportDETable *Export gene differential expression results table*

Description

Exports gene differential expression results table to text files.

Usage

```
exportDETable(object, data = "sig")

## S4 method for signature 'TSSr'
exportDETable(object, data = "sig")
```

Arguments

object	A TSSr object.
data	Specify which data will be exported: "all" or "sig".

Value

gene differential expression tables

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportDETable(exampleTSSr, data = "sig")
setwd(oldwd)
```

---

exportEnhancerTable *Export enhancer tables*

Description

Exports enhancer tables to text files.

Usage

```
exportEnhancerTable(object)

## S4 method for signature 'TSSr'
exportEnhancerTable(object)
```

Arguments

object	A TSSr object.

Value

enhancer candidate tables

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportEnhancerTable(exampleTSSr)
setwd(oldwd)
```

exportShapeTable *Export core promoter shape score tables*

Description

Exports core promoter shape score tables to text files. Shape score is calculated with shapeCluster(object) method.

Usage

```
exportShapeTable(object)

## S4 method for signature 'TSSr'
exportShapeTable(object)
```

Arguments

object	A TSSr object.

Value

core promoter shape score tables

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportShapeTable(exampleTSSr)
setwd(oldwd)
```

exportShiftTable *Export core promoter shift table*

Description

Export core promoter shift tables to text files.

Usage

```
exportShiftTable(object)

## S4 method for signature 'TSSr'
exportShiftTable(object)
```

Arguments

object	A TSSr object.

Value

core promoter shift tables

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportShiftTable(exampleTSSr)
setwd(oldwd)
```

---

exportTSStable *Export TSS tables*

Description

Exports TSS tables to text file.

Usage

```
exportTSStable(object, data = "raw", merged = TRUE)

## S4 method for signature 'TSSr'
exportTSStable(object, data = "raw", merged = TRUE)
```

Arguments

object	A TSSr object.
data	Specify which data will be exported: "raw" or "processed". Default is "raw".
merged	Logical indicating whether raw sample columns should be merged according to mergeIndex. Used only if data = "raw".

Details

Raw exports always use TSSrawMatrix. With merged = FALSE, the file is ALL.samples.TSS.raw.txt;
with merged = TRUE, raw counts are combined according to mergeIndex and written to ALL.samples.TSS.raw.merged.txt.
Processed data are written to ALL.samples.TSS.processed.txt.

Value

Invisibly returns NULL after writing a TSS table.

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportTSStable(exampleTSSr, data = "raw", merged = TRUE)
setwd(oldwd)
```

exportTSStoBedgraph *Creating Bedgraph/BigWig tracks of TSSs*

Description

Creates bedGraph/BigWig files of TSSs that can be visualized in the UCSC Genome Browser and Integrative Genomics Viewer (IGV).

Usage

```
exportTSStoBedgraph(object, data = "processed", format = "bedGraph",oneFile = FALSE)

## S4 method for signature 'TSSr'
exportTSStoBedgraph(
  object,
  data = "processed",
  format = "bedGraph",
  oneFile = FALSE
)
```

Arguments

object	A TSSr object.
data	Specify which data will be exported: "raw" or "processed". Default is "processed".
format	The format of output files: "bedGraph" or "BigWig". Default is "bedGraph".
oneFile	Logical, specify whether to export individual TSS tracks into the one bedGraph file (TRUE) of in separate bedGraph files (FALSE).

Value

Bedgraph/BigWig tracks of TSSs

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
exportTSStoBedgraph(exampleTSSr, data = "processed", format = "bedGraph")
setwd(oldwd)
```

`filterTSS` *Filter raw TSS counts or normalized TSS*

Description

Filters transcriptional or sequencing noise.

Usage

```
filterTSS(object, method = "poisson", normalization = TRUE,
pVal =0.01, tpmLow = 0.1)

## S4 method for signature 'TSSr'
filterTSS(
  object,
  method = "poisson",
  normalization = TRUE,
  pVal = 0.01,
  tpmLow = 0.1
)
```

Arguments

`object`	A TSSr object.
`method`	Method to be used for TSS filtering: "poisson" or "TPM". "poisson" can be used only if the input TSS data in raw number of counts.
`normalization`	Define whether normalization data to TPM. Used only if method = "poisson". Default is TRUE.
`pVal`	Used only if method = "poisson". Default value is 0.01.
`tpmLow`	Used only if method = "TPM". Default value is 0.1. Filtering removes a nonzero count only when this threshold is greater than that sample's smallest possible nonzero TPM value, approximately `1e6 / libraySize`.

Value

A modified TSSr object with updated `TSSprocessedMatrix` slot after filtering. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- filterTSS(exampleTSSr, method = "TPM", tpmLow = 2)
```

getTSS *Precisely identify TSSs from bam files, paired end bam files, bed files, BigWig files, tss files, or tss tables.*

Description

getTSS function is used to precisely identify TSSs from multiple input file formats. The files include users' home-made alignment files (bam format) or downloaded files from public databases. See inputFilesType for details on the supported input file formats.

Usage

```
getTSS(object, sequencingQualityThreshold = 10,
mappingQualityThreshold = 20, softclippingAllowed = FALSE)

## S4 method for signature 'TSSr'
getTSS(
  object,
  sequencingQualityThreshold = 10,
  mappingQualityThreshold = 20,
  softclippingAllowed = FALSE
)
```

Arguments

`object`	A TSSr object.
`sequencingQualityThreshold`	Used only if inputFilesType == "bam" or "bamPairedEnd", otherwise ignored.
`mappingQualityThreshold`	Used only if inputFilesType == "bam" or "bamPairedEnd", otherwise ignored.
`softclippingAllowed`	Used only if inputFilesType == "bam" or "bamPairedEnd". Default is FALSE. When FALSE, TSSr applies G-only uncoded G correction to leading G bases on plus-strand reads and trailing C bases on minus-strand reads as transcript-sense G bases. When TRUE, TSSr uses the aligner's aligned 5' boundary directly and skips uncoded G correction.

Value

A modified TSSr object with updated `TSSrawMatrix`, `TSSprocessedMatrix`, and `librarySizes` slots. The input object is not modified; assign the returned object to retain the changes.

Examples

```
exampleInput <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr", mustWork = TRUE
)
```

```
importedTSSr <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = exampleInput,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2)
)
importedTSSr <- getTSS(importedTSSr)
head(TSSmatrix(importedTSSr, data = "raw"))
```

`mergeSamples` *Merge TSS samples*

Description

Merges individual samples within TSSr object into specified groups.

Usage

```
mergeSamples(object, mergeIndex)

## S4 method for signature 'TSSr'
mergeSamples(object, mergeIndex = NULL)
```

Arguments

`object`	A TSSr object
`mergeIndex`	Integer vector specifying which samples to be merged

Value

A modified TSSr object with updated `TSSprocessedMatrix` and `librarySizes` slots after merging samples. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- mergeSamples(exampleTSSr, mergeIndex = c(1, 1, 2, 2))
```

normalizeTSS *Normalize raw TSS counts*

Description

Normalizes raw TSS counts in all samples by tags per million (TPM)

Usage

```
normalizeTSS(object)

## S4 method for signature 'TSSr'
normalizeTSS(object)
```

Arguments

object	A TSSr object.

Value

A modified TSSr object with updated `TSSprocessedMatrix` slot containing normalized TPM values. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- mergeSamples(exampleTSSr)
exampleTSSr <- normalizeTSS(exampleTSSr)
```

plotCorrelation *Pairwise scatter plots and correlations of TSS signal*

Description

Calculates the pairwise correlation coefficients between samples and creates a matix showing pairwise scatter plots and correlation coefficients. Scatterplot panels use logarithmic axes, so observations with a non-positive value in either sample are omitted from that panel with a warning. Correlation coefficients are calculated from the original, untransformed values, including zeros.

Usage

```
plotCorrelation(object, samples = "all")

## S4 method for signature 'TSSr'
plotCorrelation(object, samples = "all")
```

Arguments

object	A TSSr object.
samples	Specify samples to be plotted. Can be either "all" to plot all samples in the object or a subset of samples in the object. Default is "all".

Value

pairwise correlations visualized in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotCorrelation(exampleTSSr, samples = "all")
setwd(oldwd)
```

---

plotDE *Plot gene differential expressions*

Description

Vocano plots of gene differential expression (with DESeq2 method) results.

Usage

```
plotDE(object, withGeneName = TRUE, xlim, ylim)

## S4 method for signature 'TSSr'
plotDE(object, withGeneName = TRUE, xlim = c(-2.5, 2.5), ylim = c(0, 10))
```

Arguments

object	A TSSr object.
withGeneName	Logical indicating whether to display names for genes which are differentially expressed. Default is TRUE.
xlim	Only enes of which log2FoldChange value within the xlim range are plotted. Default xlim = c(-2.5, 2.5).
ylim	Only genes of which -log10(pvalue) within the ylim range are plotted. Default ylim = c(0, 10).

Details

Displaying gene names requires the suggested package **calibrate**.

Value

gene differential expression visualized in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotDE(exampleTSSr, withGeneName = TRUE)
setwd(oldwd)
```

`plotInterQuantile` *Plot core promoter interquantile width*

Description

Plots histograms of the interquantile width of processed clusters.

Usage

```
plotInterQuantile(object, samples ="all", tagsThreshold = 1)

## S4 method for signature 'TSSr'
plotInterQuantile(object, samples = "all", tagsThreshold = 1)
```

Arguments

`object`	A TSSr object.
`samples`	Specify samples to be plotted. Default is "all".
`tagsThreshold`	Excludes clusters with tags < tagsThreshold.

Value

core promoter interquantile width visualized in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotInterQuantile(exampleTSSr, samples = "all", tagsThreshold = 1)
setwd(oldwd)
```

plotShape *Plot core promoter shape*

Description

Plots histograms of core promoter shape scores.

Usage

```
plotShape(object, samples = "all")

## S4 method for signature 'TSSr'
plotShape(object, samples = "all")
```

Arguments

object	A TSSr object.
samples	Specify samples to be plotted. Default is "all".

Value

core promoter shape score visualized in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotShape(exampleTSSr, samples = "all")
setwd(oldwd)
```

plotTSS *Plot TSSs and clusters*

Description

Plots Gviz-track of TSSs, clusters, and genes.

Usage

```
plotTSS(object,samples,tssData = "processed",clusters = "assigned",
clusterThreshold = 0.02,genelist,Bidirection = TRUE,
up.dis =500,down.dis = 500,yFixed = TRUE)

## S4 method for signature 'TSSr'
plotTSS(
  object,
```

```
  samples,
  tssData = "processed",
  clusters = "assigned",
  clusterThreshold = 0.02,
  genelist,
  Bidirection = TRUE,
  up.dis = 500,
  down.dis = 500,
  yFixed = TRUE
)
```

Arguments

object	A TSSr object.
samples	Specify samples to be included for plotting.
tssData	Specify which TSS data to be included for plotting: "raw" or "processed".
clusters	Specify which cluster data to be included for plotting: "all" or "assigned".
clusterThreshold	Ignore downstream clusters if signal < filterClusterThreshold*the strongest clusters within the same gene promoter region. Default value = 0.02.
genelist	List of gene names used for plotting.
Bidirection	Specify whether to display bidirectional TSS signals within defined region. Default is TRUE.
up.dis	Distance upstream of genes to specify plotting range. Default value = 500.
down.dis	Distance downstream of genes to specify plotting range. Default value = 500.
yFixed	Logical, specify whether to fix y axis limits. Default is TRUE.

Details

Requires the suggested package **Gviz**.

Value

TSS and cluster examples visualized in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotTSS(exampleTSSr, samples = c("control", "treat"),
    genelist = c("YBL017C", "YBL067C"), up.dis = 500, down.dis = 500)
setwd(oldwd)
```

plotTssPCA *Plotting principle component analysis (PCA)*

Description

Calculates principle component analysis (PCA) of all samples and creates a biplot which includes the position of each sample in terms of PC1 and PC2.

Usage

```
plotTssPCA(object, TSS.threshold =10)

## S4 method for signature 'TSSr'
plotTssPCA(object, TSS.threshold = 10)
```

Arguments

object	A TSSr object.
TSS.threshold	Only TSSs with raw signal >= TSS.threshold will be included in PCA analysis

Details

Requires the suggested package **ggfortify**.

Value

PCA plotted in a graph

Examples

```
data(exampleTSSr)
oldwd <- setwd(tempdir())
plotTssPCA(exampleTSSr, TSS.threshold = 10)
setwd(oldwd)
```

shapeCluster *Analysis of core promoter shape*

Description

Calculates core promoter shape based on the distributions of TSSs within core promoters using Shape Index (SI) algorithm (Hoskins et al. 2011) or Promoter Shape Score (PSS) algorithm (Lu et al. 2019).

Usage

```
shapeCluster(object, clusters = "consensusClusters", method = "PSS",
 useMultiCore=FALSE, numCores = NULL)

## S4 method for signature 'TSSr'
shapeCluster(
  object,
  clusters = "consensusClusters",
  method = "PSS",
  useMultiCore = FALSE,
  numCores = NULL
)
```

Arguments

object	A TSSr object.
clusters	Clusters to be used for calculating shape score: "tagClusters" or "consensusClusters". Default is "consensusClusters".
method	Method to be used for calculating core promoter shape score: "SI" or "PSS". Default is "PSS".
useMultiCore	Logical indicating whether multiple cores are used (TRUE) or not (FALSE). Default is FALSE.
numCores	Number of cores are used in clustering step. Used only if useMultiCore = TRUE. Default is NULL.

Value

A modified TSSr object with updated `clusterShape` slot. The input object is not modified; assign the returned object to retain the changes.

Examples

```
example_input <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr",
    mustWork = TRUE
)
example_object <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = example_input,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2)
)
example_object <- getTSS(example_object)
example_object <- mergeSamples(example_object)
example_object <- normalizeTSS(example_object)
example_object <- clusterTSS(example_object, useMultiCore = FALSE)
example_object <- consensusCluster(
```

```
    example_object, useMultiCore = FALSE
)
example_object <- shapeCluster(
    example_object, clusters = "consensusClusters", method = "PSS",
    useMultiCore = FALSE
)
head(clusterShape(example_object, sample = "control"))
```

`shiftPromoter` *Select genes which have core promoter shift across different experiments.*

Description

Selects genes which have multiple core promoters and undergo core promoter shifting across different experiments. Generates gene list with Ds (degree of shift) value (Lu et al., 2019), p value and adjusted p value. Chi-squared tests use raw counts reconstructed from TPM values and the corresponding library sizes. When sparse 2-by-2 tables have small expected counts, the function emits one summary warning for all affected gene-level tests. Their approximate p values may be unreliable; the warning should not be treated as safe to ignore for inferential use.

Usage

```
shiftPromoter(object, comparePairs, pval=0.01)

## S4 method for signature 'TSSr'
shiftPromoter(object, comparePairs, pval = 0.01)
```

Arguments

`object`	A TSSr object.
`comparePairs`	Specified list of sample pairs for comparison.
`pval`	Genes with adjusted p value less than or equal to `pval` will be returned. Default value = 0.01.

Value

A modified TSSr object with updated `PromoterShift` slot. The input object is not modified; assign the returned object to retain the changes.

Examples

```
data(exampleTSSr)
exampleTSSr <- shiftPromoter(
    exampleTSSr, comparePairs = list(c("control", "treat")), pval = 0.01
)
head(PromoterShift(exampleTSSr, comparison = "control_VS_treat"))
```

show,TSSr-method *Display a compact summary of a TSSr object*

Description

Shows the genome, sample and TSS counts, and concise row counts for completed analyses without expanding the full contents of the object's slots.

Usage

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

Arguments

object	A TSSr object.

Value

The input TSSr object, invisibly.

Examples

```
data(exampleTSSr)
show(exampleTSSr)
```

TSSr-accessors *Access data stored in a TSSr object*

Description

These read-only accessors provide stable alternatives to direct slot access. Tables are returned as independent base `data.frame` copies, so modifying a returned value does not modify `object` or expose its internal `data.table` representation.

Usage

```
TSSmatrix(object, data = c("raw", "processed"))

## S4 method for signature 'TSSr'
TSSmatrix(object, data = c("raw", "processed"))

librarySizes(object)

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

```
refTable(object)

## S4 method for signature 'TSSr'
refTable(object)

tagClusters(object, sample = NULL)

## S4 method for signature 'TSSr'
tagClusters(object, sample = NULL)

consensusClusters(object, sample = NULL)

## S4 method for signature 'TSSr'
consensusClusters(object, sample = NULL)

clusterShape(object, sample = NULL)

## S4 method for signature 'TSSr'
clusterShape(object, sample = NULL)

assignedClusters(object, sample = NULL)

## S4 method for signature 'TSSr'
assignedClusters(object, sample = NULL)

unassignedClusters(object, sample = NULL)

## S4 method for signature 'TSSr'
unassignedClusters(object, sample = NULL)

filteredClusters(object, sample = NULL)

## S4 method for signature 'TSSr'
filteredClusters(object, sample = NULL)

enhancers(object, sample = NULL)

## S4 method for signature 'TSSr'
enhancers(object, sample = NULL)

DEtables(object, comparison = NULL, result = c("all", "significant"))

## S4 method for signature 'TSSr'
DEtables(object, comparison = NULL, result = c("all", "significant"))

TAGtables(object, sample = NULL)
```

```
## S4 method for signature 'TSSr'
TAGtables(object, sample = NULL)

PromoterShift(object, comparison = NULL)

## S4 method for signature 'TSSr'
PromoterShift(object, comparison = NULL)
```

Arguments

object	A TSSr object.
data	Which TSS matrix to return: "raw" or "processed".
sample	A sample name, or NULL to return every sample.
comparison	A comparison name, or NULL to return every comparison.
result	For DEtables(), return either "all" tested genes or only "significant" genes.

Value

TSSmatrix() and refTable() return a base data.frame. librarySizes() returns a named numeric vector. The analysis-result accessors return a base data.frame when a sample or comparison is selected, and otherwise return a named list of base data.frame objects.

Examples

```
data(exampleTSSr)

head(TSSmatrix(exampleTSSr, data = "raw"))
librarySizes(exampleTSSr)
head(refTable(exampleTSSr))

head(tagClusters(exampleTSSr, sample = "control"))
head(consensusClusters(exampleTSSr, sample = "control"))
head(clusterShape(exampleTSSr, sample = "control"))
head(assignedClusters(exampleTSSr, sample = "control"))
head(unassignedClusters(exampleTSSr, sample = "control"))
head(filteredClusters(exampleTSSr, sample = "control"))
head(enhancers(exampleTSSr, sample = "control"))
head(DEtables(
    exampleTSSr,
    comparison = "control_VS_treat",
    result = "significant"
))
head(TAGtables(exampleTSSr, sample = "control"))
head(PromoterShift(exampleTSSr, comparison = "control_VS_treat"))
```

TSSr-class *TSSr: A package for transcription start site sequencing data analyses.*

Description

TSSr is designed to analyze transcription start sites (TSSs) and core promoters with most types of 5'end sequencing data, such as cap analysis of gene expression (CAGE) (Takahashi, Lassmann et al. 2012), no-amplification non-tagging CAGE libraries for Illumina next-generation sequencers (nAnT-iCAGE) (Murata, Nishiyori-Sueki et al. 2014), a Super-Low Input Carrier-CAGE (SLIC-CAGE) (Cvetesic, Leitch et al. 2018), NanoCAGE (Cumbie, Ivanchenko et al. 2015), TSS-seq (Malabat, Feuerbach et al. 2015), transcript isoform sequencing (TIF-seq) (Pelechano, Wei et al. 2013), transcript-leaders sequencing (TL-seq) (Arribere and Gilbert 2013), precision nuclear run-on sequencing (PRO-Cap) (Mahat, Kwak et al. 2016), and GRO-Cap/5'GRO-seq (Kruesi, Core et al. 2013).

Creates a new TSSr object for TSS sequencing data analysis.

Usage

```
TSSr(
  genomeName,
  inputFiles,
  inputFilesType,
  sampleLabels,
  sampleLabelsMerged = character(),
  mergeIndex = numeric(),
  refSource = character(),
  organismName = character()
)
```

Arguments

genomeName	A character string specifying the BSgenome object name (e.g., "BSgenome.Scerevisiae.UCSC.sacCer3").
inputFiles	A character vector of input file paths. For inputFilesType = "TSStable", provide one table containing all sample columns; other input types require one file per sample. Every path must identify an existing regular file when the object is created.
inputFilesType	A character string specifying the type of input files. Must be one of: "bam", "bamPairedEnd", "bed", "tss", "TSStable", "BigWig".
sampleLabels	A character vector of sample labels corresponding to input files, or to sample columns in a TSStable table.
sampleLabelsMerged	An optional character vector of merged sample labels for biological replicates. Supply this together with mergeIndex; when both are omitted, every sample forms its own group.
mergeIndex	An optional numeric vector indicating which samples to merge together. Supply this together with sampleLabelsMerged.

`refSource` An optional path to a GFF annotation file. If supplied, it must identify an existing regular file. `annotateCluster()` requires either this path or a populated `refTable` slot.

`organismName` An optional character string describing the organism. This metadata is retained for backward compatibility and is not required by `annotateCluster()`.

Details

TSSr package provides a comprehensive workflow on TSS data starts from identification of accurate TSS locations, clustering TSSs within small genomic regions corresponding to core promoters, and transcriptional activity quantifications, as well as specialized downstream analyses including core promoter shape, cluster annotation, gene differential expression, core promoter shift. TSSr can take multiple formats of files as input, such as Binary Sequence Alignment Map (BAM) files (single-ended or paired-ended), Browser Extension Data (bed) files, BigWig files, ctss files or tss tables. TSSr also generates various types of TSS or core promoter track files which can be visualized in the UCSC Genome Browser or Integrative Genomics Viewer (IGV). TSSr also exports downstream analyses result tables and plots. Multiple cores are supported on Linux or Mac platforms.

Value

A TSSr object.

Slots

`genomeName` character.

`inputFiles` character.

`inputFilesType` character.

`sampleLabels` character.

`sampleLabelsMerged` character.

`librarySizes` numeric.

`normalizationStatus` A scalar character value describing whether `TSSprocessedMatrix` contains `"raw"` counts or `"normalized"` values. `NA` indicates that no state has been established yet.

`TSSrawMatrix` data.frame.

`mergeIndex` numeric.

`TSSprocessedMatrix` data.frame.

`tagClusters` list.

`consensusClusters` list.

`clusterShape` list.

`refSource` character.

`refTable` data.frame.

`organismName` Optional character metadata describing the organism. It is retained for backward compatibility and is not used for cluster-to-gene assignment.

`assignedClusters` list.

unassignedClusters list.

filteredClusters list.

enhancers list.

DEtables list.

TAGtables list

PromoterShift list.

Examples

```
exampleInput <- system.file(
    "extdata", "example-tss-table.tsv", package = "TSSr", mustWork = TRUE
)
exampleAnnotation <- system.file(
    "extdata", "example-annotation.gff3", package = "TSSr", mustWork = TRUE
)
myTSSr <- TSSr(
    genomeName = "BSgenome.Scerevisiae.UCSC.sacCer3",
    inputFiles = exampleInput,
    inputFilesType = "TSStable",
    sampleLabels = c("SL01", "SL02", "SL03", "SL04"),
    sampleLabelsMerged = c("control", "treat"),
    mergeIndex = c(1, 1, 2, 2),
    refSource = exampleAnnotation
)
```

Index

∗ **datasets**
exampleTSSr, 9

annotateCluster, 3
annotateCluster,TSSr-method
(annotateCluster), 3
assignedClusters (TSSr-accessors), 27
assignedClusters,TSSr-method
(TSSr-accessors), 27

callEnhancer, 4
callEnhancer,TSSr-method
(callEnhancer), 4
clusterShape (TSSr-accessors), 27
clusterShape,TSSr-method
(TSSr-accessors), 27
clusterTSS, 5
clusterTSS,TSSr-method (clusterTSS), 5
consensusCluster, 6
consensusCluster,TSSr-method
(consensusCluster), 6
consensusClusters (TSSr-accessors), 27
consensusClusters,TSSr-method
(TSSr-accessors), 27

deGene, 7
deGene,TSSr-method (deGene), 7
DEtables (TSSr-accessors), 27
DEtables,TSSr-method (TSSr-accessors),
27

enhancers (TSSr-accessors), 27
enhancers,TSSr-method (TSSr-accessors),
27
exampleTSSr, 9
exportClustersTable, 10
exportClustersTable,TSSr-method
(exportClustersTable), 10
exportClustersToBed, 10
exportClustersToBed,TSSr-method
(exportClustersToBed), 10
exportDETable, 11
exportDETable,TSSr-method
(exportDETable), 11
exportEnhancerTable, 12
exportEnhancerTable,TSSr-method
(exportEnhancerTable), 12
exportShapeTable, 13
exportShapeTable,TSSr-method
(exportShapeTable), 13
exportShiftTable, 13
exportShiftTable,TSSr-method
(exportShiftTable), 13
exportTSStable, 14
exportTSStable,TSSr-method
(exportTSStable), 14
exportTSStoBedgraph, 15
exportTSStoBedgraph,TSSr-method
(exportTSStoBedgraph), 15

filteredClusters (TSSr-accessors), 27
filteredClusters,TSSr-method
(TSSr-accessors), 27
filterTSS, 16
filterTSS,TSSr-method (filterTSS), 16

getTSS, 17
getTSS,TSSr-method (getTSS), 17

librarySizes (TSSr-accessors), 27
librarySizes,TSSr-method
(TSSr-accessors), 27

mergeSamples, 18
mergeSamples,TSSr-method
(mergeSamples), 18

normalizeTSS, 19
normalizeTSS,TSSr-method
(normalizeTSS), 19

plotCorrelation, 19

plotCorrelation,TSSr-method
(plotCorrelation), 19
plotDE, 20
plotDE,TSSr-method (plotDE), 20
plotInterQuantile, 21
plotInterQuantile,TSSr-method
(plotInterQuantile), 21
plotShape, 22
plotShape,TSSr-method (plotShape), 22
plotTSS, 22
plotTSS,TSSr-method (plotTSS), 22
plotTssPCA, 24
plotTssPCA,TSSr-method (plotTssPCA), 24
PromoterShift (TSSr-accessors), 27
PromoterShift,TSSr-method
(TSSr-accessors), 27

refTable (TSSr-accessors), 27
refTable,TSSr-method (TSSr-accessors),
27

shapeCluster, 24
shapeCluster,TSSr-method
(shapeCluster), 24
shiftPromoter, 26
shiftPromoter,TSSr-method
(shiftPromoter), 26
show,TSSr-method, 27

tagClusters (TSSr-accessors), 27
tagClusters,TSSr-method
(TSSr-accessors), 27
TAGtables (TSSr-accessors), 27
TAGtables,TSSr-method (TSSr-accessors),
27
TSSmatrix (TSSr-accessors), 27
TSSmatrix,TSSr-method (TSSr-accessors),
27
TSSr, 9
TSSr (TSSr-class), 30
TSSr-accessors, 27
TSSr-class, 30

unassignedClusters (TSSr-accessors), 27
unassignedClusters,TSSr-method
(TSSr-accessors), 27