Package 'TSSi'

October 9, 2013

Type Package

Title Transcription Start Site Identification

Version 1.6.0

Date 2012-05-06

Author Clemens Kreutz, Julian Gehring

Maintainer Julian Gehring <julian.gehring@embl.de>

Depends R (>= 2.13.2), BiocGenerics (>= 0.3.2)

Imports BiocGenerics, methods, Hmisc, minqa, stats, Biobase (>= 0.3.2), plyr, IRanges

Suggests rtracklayer

Enhances multicore

Description Identify and normalize transcription start sites in high-throughput sequencing data.

License GPL-3

LazyLoad yes

URL http://julian-gehring.github.com/TSSi/

biocViews Sequencing, HighThroughputSequencing, RNAseq, Genetics,Preprocessing

R topics documented:

`TSSi-package` *Transcription Start Site Identification*

Description

The **TSSi** package normalizes and identifies transcription start sites in high-throughput sequencing data.

Details

High throughput sequencing has become an essential experimental approach for the investigation of transcriptional mechanisms. For some applications like ChIP-seq, there are several available approaches for the prediction of peak locations. However, these methods are not designed for the identification of transcription start sites (TSS) because such data sets have qualitatively different noise.

The **TSSi** provides a heuristic framework for the identification of TSS based on high-throughput sequencing data. Probabilistic assumptions for the count distribution as well as for systematic errors, i.e. for contaminating measurements close to a TSS, are made and can be adapted by the user. The framework also comprises a regularization procedure which can be applied as a preprocessing step to decrease the noise and thereby reduce the number of false predictions.

The package is published under the GPL-3 license.

Author(s)

Clemens Kreutz, Julian Gehring, Jens Timmer

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

References

C. Kreutz, J. Gehring, D. Lang, J. Timmer, and S. Rensing: TSSi - An R package for transcription start site identification from high throughput sequencing data.

in preparation

See Also

Package: `TSSi-package`

Methods: `segmentizeCounts`, `normalizeCounts`, `identifyStartSites`, `get-methods`, `plot-methods`, `asRangedData-methods`

Functions: `subtract-functions`

Classes: `TssData`, `TssNorm`, `TssResult`

Data set: `physcoCounts`

Examples

```
## load data set
data(physcoCounts)

## segmentize data
attach(physcoCounts)
x <- segmentizeCounts(counts=counts, start=start, chr=chromosome,
region=region, strand=strand)
detach(physcoCounts)

x
segments(x)

## normalize data, w/o and w/ fitting
yRatio <- normalizeCounts(x)
yFit <- normalizeCounts(x, fit=TRUE)
yFit

## identify TSS
z <- identifyStartSites(yFit)
z

## inspect results
head(tss(z, 1))
plot(z, 1)
```

---

asRangedData-methods *Convert to* RangedData

Description

Convert data and results to objects of class RangedData.

Usage

```
readsAsRangedData(x)

segmentsAsRangedData(x)

tssAsRangedData(x)
```

Arguments

x	Object of class TssData or inherited; for tssAsRangedData an object of class TssResult.

Details

The RangedData class provides a widely used framework for representing sequence information. Converting the data to an object of this class allows an easy interaction with other packages and export to common formats, using for example the **rtracklayer** package. For an example, please see the vignette of this package.

Value

An object of class RangedData

Methods

readsAsRangedData: `signature(x="TssData")`

Convert the reads, as obtained by the reads method, to an object of class RangedData.

segmentsAsRangedData: `signature(x="TssData")`

Convert the segments, as obtained by the segments method, to an object of class RangedData.

tssAsRangedData: `signature(x="TssResult")`

Convert the TSS predictions, as obtained by the tss method, to an object of class RangedData.

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

RangedData

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: **TSSi-package**

Examples

```
example(segmentizeCounts)

readsRd <- readsAsRangedData(x)
head(readsRd)

segmentsRd <- segmentsAsRangedData(x)
head(segmentsRd)
```

get-methods *Get methods*

Description

Get methods for objects of the classes TssData, TssNorm, and TssResult.

Value

A data frame or list

Methods

For class TssData, TssNorm (inherited), TssResult (inherited):

start: signature(x="TssData")
Get the read start sites. The second argument selects the individual segment; if missing returns a list containing the information of all segments.

end: signature(x="TssData")
Get the read end sites; see start.

counts: signature(x="TssData")
Get the raw read counts; see start.

reads: signature(x="TssData")
Get all the read data of the segments, including e.g. 'start', 'counts', 'replicate'; see start.

segments: signature(x="TssData")
Get the information associated with the segments, e.g. chromosome, strand, region. The optional second and third arguments select the segment and the variable of interest.

annotation: signature(x="TssData")
Get the annotation data, as passed through the annotation argument.

[: signature(x="TssData")
Subset the object, by name or index.

For class TssNorm, TssResult (inherited):

ratio: signature(x="TssNorm")
Get the normalized reads based on the Poisson ratios; see start.

fit: signature(x="TssNorm")
Get the normalized reads based on the fit; see start.

For class TssResult:

expect: signature(x="TssResult")
Get the expectation for non-specific reads; see start.

tss: signature(x="TssResult")
Get the identified transcription start sites; see start.

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
example(segmentizeCounts)

## some examples for get methods
start(x)
head(start(x, 1))

head(reads(x, 1))

segments(x)
names(x)
```

---

idenfify-methods *Identify methods*

Description

Identify transcription start sites in sequence read count data.

Usage

```
identifyStartSites(x, threshold=1, tau=c(20, 20), neighbor=TRUE,
fun=subtractExpectation, multicore=TRUE, ...)
```

Arguments

x	Object of class TssNorm with normalized data.
threshold	Numeric with the minimal number of reads to be treated as a potential TSS.
tau	Numeric vector of length two specifying the τ parameter of the exponential function for each side of the segment. For the forward strand ("+"), the first and second value refer to the side towards the 5' and 3' end, respectively. In the case that a single value is provided it is applied to both sides.

`neighbor`	Logical whether the background estimates should be iteratively assigned to the predicted TSS during the estimation (default: TRUE).
`fun`	Function to calculate the expectation for each TSS. For details, see the 'details' section.
`multicore`	Logical whether to use the **multicore** package to speed up the computation. Has only an effect if the package is available and loaded. For details, see the 'details' section.
`...`	Additional arguments passed for the **multicore** package if used. For details, see the 'details' section.

Details

After normalization of the count data, an iterative algorithm is applied for each segment to identify the TSS.

The expected number of false positive counts is initialized with a default value given by the read frequency in the whole data set. The position with the largest counts above is identified as a TSS, if the expected transcription level is at least one read above the expected number of false positive reads. The transcription levels for all TSS are calculated by adding all counts to their nearest neighbor TSS.

Then, the expected number of false positive reads is updated by convolution with exponential kernels. The decay rates τ in 3' direction and towards the 5'-end can be chosen differently to account for the fact that false positive counts are preferably found in 5' direction of a TSS. This procedure is iterated as long as the set of TSS increases.

In order to distribute the identification step over multiple processor cores, the `mclapply` function of the **multicore** package can be used. For this, the **multicore** package has to be loaded manually before starting the computation, additional parameters are passed via the `...` argument, e.g.as `normalizeCounts(x, mc.cores=2)`. The `multicore` argument can further be used to temporarily disable the parallel estimation by setting it to `FALSE`. Pleas note that the identification step is normally very fast and thus using parallel computation here may a minor impact as compared to the `normalizeCounts` method.

Value

An object of class `TssResult`.

Methods

Identify TSS:

identifyStartSites: `signature(x="TssData")`

```
identifyStartSites(x, ...)
```

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
## preceding steps
example(normalizeCounts)

## identify TSS
z <- identifyStartSites(yFit)

z
```

---

normalizeCounts-methods

Normalize methods

Description

Normalize sequence read count data.

Usage

```
normalizeCounts(x, fun=mean, offset=10L, basal=1e-4, lambda=c(0.1, 0.1),
fit=FALSE, multicore=TRUE, optimizer="all", ...)
```

Arguments

x	Object of class TssData with raw data to normalize.
fun	Function used to average over replicates (default: mean).
offset	Integer defining the number of bases add to the ends of each segment with basal rate.
basal	Numeric specifying the basal rate.
lambda	Numeric vector of length two specifying the regulation parameter for each side of the segment.
fit	Logical whether the fitting should be performed in addition to the estimation based on the Poisson ratios obtained from all reads.
multicore	Logical whether to use the **multicore** package to speed up the fitting. Has only an effect if the package is available and loaded. For details, see the 'details' section.

optimizer	Character string choosing the optimizer for the fit (default: "all"). Possible choices are "optim" for the optim function from the **stats** package, "bobyqa" for the bobyqa function from the **minqu** package,or "all" for taking the best fit out of both.
...	Additional arguments passed for the **multicore** package if used. For details, see the 'details' section.

Details

The normalization reduces the noise by shrinking the counts towards zero. This step is intended to eliminate false positive counts as well as making further analyzes more robust by reducing the impact of large counts. Such a shrinkage or regularization procedure constitutes a well-established strategy in statistics to make predictions conservative, i.e. to reduce the number of false positive predictions.

An objective function is minimized to estimate the transcription level in a regularized manner. The log-likelihood is given by the product of the probabilities of the counts which is assumed as a Poisson distribution by default.

For $\lambda_1 > 0$, counts unequal to zero are penalized to obtain conservative estimates of the transcription levels with a preferably small number components, i.e. genomic positions, unequal to zero. The larger λ_1, the more conservative is the identification procedure.

To enhance the shrinkage of isolated counts in comparison to counts in regions of strong transcriptional activity, the information of consecutive genomic positions in the measurements is regarded by evaluating differences between adjacent count estimates.

In order to distribute the identification step over multiple processor cores, the `mclapply` function of the **multicore** package can be used. For this, the **multicore** package has to be loaded manually before starting the computation, additional parameters are passed via the `...` argument, e.g.as `normalizeCounts(x, mc.cores=2)`. The `multicore` argument can further be used to temporarily disable the parallel estimation by setting it to `FALSE`.

Value

An object of class `TssNorm`.

Methods

Normalize read data:

normalizeCounts: `signature(x="TssData")`

```
normalizeCounts(x, ...)
```

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
## preceding steps
example(segmentizeCounts)

## normalize data, w/o and w/ fitting
yRatio <- normalizeCounts(x)
yFit <- normalizeCounts(x, fit=TRUE)

yFit

## Not run:
## multicore computation
library(multicore)
yFit <- normalizeCounts(x, fit=TRUE, mc.cores=2)

## End(Not run)
```

---

physcoCounts-data *CAP capture data*

Description

Data of a 5'-CAP capture experiment, after mapping the reads to the genome.

Usage

```
data(physcoCounts)
```

Format

physcoCounts Data frame with results from a 5'-CAP capture experiment, with the columns:

- **chromosome** Chromosome the read is mapped to.
- **region** Predefined region based on annotations, which can be treated independently in the analysis.
- **start** Start position, given as bp, of the 5' end of the read.
- **strand** Forward and reverse strand, given as '+' and '-', respectively.
- **counts** Number of reads at the respective position.

Details

The transcription data from Physcomitrella patens was mapped using the bowtie software. Then, the positions of 5' ends of each read were extracted and the number of reads at each position counted.

For further details, please see the publication.

Source

Rensing et al., 2011.

in preparation

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
## load data set
data(physcoCounts)
```

---

plot-methods *Plot methods*

Description

Plot the data and results of the **TSSi** package.

Usage

```
plot(x, y, ...)
```

Arguments

x	An object of class TssData, TssNorm, or TssResult.
y	A single integer or character string specifying which segment to plot. An integer is interpreted as the index of the segment while a character string is matched against the segment names.
...	Optional arguments used in order to customize the figure. See the 'details' section.

Details

With the `plot` method, the raw, normalized, or final data can easily be visualized.

The `plot` method uses a special system in order to customize the graphical elements of the figure. It allows to refer to the different components with the name of the additional input argument; its value is a list containing named graphical parameters for the underlying plot function. The following list describes the available names and their contribution.

`plot` Graphical parameters for the axes and the labeling, passed to the `plot` function.

`counts` Logical indicating whether the raw counts should be plotted.

`countsArgs` Graphical parameters for the 'counts' variable, passed to the `points` function.

`ratio` Logical indicating whether the estimates based on the Poisson ratios should be plotted.

`ratioArgs` Graphical parameters for the 'ratio' variable, passed to the `points` function.

`fit` Logical indicating whether the estimates based on the fitting should be plotted.

`fitArgs` Graphical parameters for the 'fit' variable, passed to the `points` function.

`expect` Logical indicating whether the background estimates should be plotted.

`expectArgs` Graphical parameters for the 'expect' variable, passed to the `points` function.

`expect` Logical indicating whether the background estimates should be computed for all positions, rather than only for those with reads.

`tss` Logical indicating whether the identified TSS should be plotted.

`tssArgs` Graphical parameters for the 'tss' variable, passed to the `points` function.

`threshold` Logical indicating whether the threshold parameter used in the identification step should be indicated.

`thresholdArgs` Graphical parameters for the 'threshold' variable, passed to the `abline` function.

`rug` Logical indicating whether the location of the identified TSS should be indicated.

`rugArgs` Graphical parameters for the 'rug' variable, passed to the `rug` function.

`baseline` Logical indicating whether a baseline indicating zero reads should be drawn.

`baselineArgs` Graphical parameters for the 'baseline' variable, passed to the `abline` function.

`legend` Logical indicating whether a legend should be plotted.

`legendArgs` Graphical parameters for the 'legend' variable, passed to the `legend` function.

Thus, for (a) omitting the ratio estimates, the threshold, and the legend, (b) customizing the graphical parameters of the raw read counts, (c) customizing the axis labels and the title, the following code can be used:

```
plot(x, 1, ratio=FALSE, threshold=FALSE, legend=FALSE,  countsArgs=list(type="h", col="darkgray", pch
's1_-_155'")
```

Methods

Visualize the raw data:

plot: `signature(x="TssData")`

```
plot(x, y, counts=TRUE, legend=TRUE, ...)
```

Visualize the normalized data:

plot: `signature(x="TssNorm")`
`plot(x, y, counts=TRUE, ratio=TRUE, fit=TRUE, legend=TRUE,...)`

Visualize the normalized data along with the identified TSS:

plot: `signature(x="TssResult")`
`plot(x, y, counts=TRUE, ratio=TRUE, fit=TRUE, expect=FALSE,tss=TRUE, threshold=TRUE, rug=TRUE, leg`

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
## preceding steps
example(identifyStartSites)

## plot
plot(yFit, 1)

## plot w/ some custom settings
plot(z, 1, ratio=FALSE, threshold=FALSE, countsArgs=list(type="h",
col="darkgray", pch=NA), plotArgs=list(xlab="Genomic position",
main="TSS for segment 's1_-_155'"))
```

---

segmentizeCounts-methods

Segmentize methods

Description

Import sequence read count data and transform it into segments.

Usage

```
segmentizeCounts(counts, start, end=start, chr=rep(1L, length(start)),
region=rep(1L, length(start)), strand=rep("*", length(start)),
replicate=rep(1L, length(start)), annotation=NULL, ...)
```

Arguments

counts	Integer vector with the number of reads for each position in start.
start	Integer vector with the start positions of the reads.
end	Optional integer vector with the end positions of the reads. If not supplied the values of start will be used.
chr	Optional vector with the chromosomal locations of the reads. If not supplied all reads will are assumed to be located on one chromosome.
region	Optional vector with an assignment of each read to a separate region, for example based on additional annotation. If not supplied all reads will be part of one region.
strand	Optional vector with the strand location of each read.
replicate	Optional integer vector identifying the replicate each read was obtained from, in the case of data from multiple measurements.
annotation	Optional object containing meta data passed along it the analysis. This argument does not influence the analysis.
...	Optional arguments.

pattern Regular expression specifying the naming of the segments. The terms %1$s, %2$s, and %3$s refer to the chromosome, the strand, and the region, respectively. If not supplied the standard naming pattern %1$s_%2$s_%3$s will be used.

Details

The segmentizeCounts method takes the raw data and breaks it into segments which will be analyzed separately in the subsequent steps. Segments are defined in a way such that any has a unique combination of the input arguments chr, region, and strand. In case any of these is not supplied it is assumed that all reads belong to one chromosome, region, or strand, respectively. Usage of the region argument is beneficial if the location of potential TSS can be constrained below the level of chromosomes and strands.

Value

An object of class TssData.

Methods

Import read data and transform it into segments:

segmentizeCounts: `signature(nReads="integer", start="integer")`

```
segmentizeCounts(counts, start, ...)
```

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
## load data set
example(physcoCounts)

## import and segmentize data
attach(physcoCounts)
x <- segmentizeCounts(counts=counts, start=start, chr=chromosome,
region=region, strand=strand)
detach(physcoCounts)

x
```

---

subtract-functions *Subtract functions*

Description

Functions subtracting the expectation value used in the identification of TSS.

Usage

```
subtractExpectation(fg, bg, indTss, pos, basal, tau, extend=FALSE)
```

Arguments

fg	Numeric vector with foreground.
bg	Numeric vector with background of the same length as fg.
indTss	Index vector indicating the expected TSS sites in fg.
pos	Positions of the reads in the segment, as given in the segmentizeCounts method.
basal	See the normalizeCounts method.
tau	See the identifyStartSites method.
extend	Logical indicating whether the background estimates should be computed for all positions, rather than only for those with reads.

Details

The subtractExpectation function is one approach on how to subtract the expectation value.

Other functions with the same call structure can be used in the detection of the TSS by passing it as fun argument in the identifyStartSites method.

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
args(subtractExpectation)
```

TssData-class *Class* TssData

Description

Class TssData storing raw sequence read data.

Slots

reads: List with the read data for each segment.

segments: Data frame with the information of the individual segments.

parameters: List with parameters used for the analysis.

annotation: Annotation data object, as passed through the annotation argument.

Methods

segmentizeCounts methods:

segmentizeCounts: `signature(nReads="integer", start="integer")`
Import raw data and divide into segments.

plot method:

plot: `signature(x="TssData", y="ANY")`
Visualize the data.

get methods:

start: `signature(x="TssData")`
Get the read start sites. The second argument selects the individual segment; if missing returns a list containing the information of all segments.

end: `signature(x="TssData")`
Get the read end sites, see `start`.

counts: `signature(x="TssData")`
Get the raw read counts, see `start`.

reads: `signature(x="TssData")`
Get all the read data of the segments, including e.g. 'start', 'counts', 'replicate', , see `start`.

segments: `signature(x="TssData")`
Get the information associated with the segments, e.g. chromosome, strand, region. The second and third argument select the segment and the variable of interest.

parameters: `signature(x="TssData")`
Get the parameters used in the analysis.

annotation: `signature(x="TssData")`
Get the annotation data, as passed through the `annotation` argument.

[: `signature(x="TssData")`
Subset the object, by name or index.

show methods:

show: `signature(object="TssData")`

asRangedData methods:

readsAsRangedData: `signature(x="TssData")`
Convert the reads, as obtained by the `reads` method, to an object of class `RangedData`.

segmentsAsRangedData: `signature(x="TssData")`
Convert the segments, as obtained by the `segments` method, to an object of class `RangedData`.

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: `TssData`, `TssNorm`, `TssResult`

Methods: `segmentizeCounts`, `normalizeCounts`, `identifyStartSites`, `get-methods`, `plot-methods`, `asRangedData-methods`

Functions: `subtract-functions`

Data set: `physcoCounts`

Package: `TSSi-package`

Examples

```
showClass("TssData")
```

TssNorm-class *Class* TssNorm

Description

Class TssNorm storing normalized sequence read data.

Slots

reads: List with the read data for each segment.

segments: Data frame with the information of the individual segments.

annotation: Annotation data object, as passed through the annotation argument.

Methods

All methods for class TssData, as well as:

identifyStartSites methods:

identifyStartSites: signature(x="TssNorm")
Identify TSS in the normalized data.

get methods:

ratio: signature(x="TssNorm")
Get the normalized reads based on the Poisson ratios, seestart.

fit: signature(x="TssNorm")
Get the normalized reads based on the fit, seestart.

[: signature(x="TssNorm")
Subset the object, by name or index.

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
showClass("TssNorm")
```

TssResult-class *Class* TssResult

Description

Class TssResult storing final sequence read data with identified TSS.

Slots

reads: List with the read data for each segment.

segments: Data frame with the information of the individual segments.

tss: List of the identified TSS locations.

annotation: Annotation data object, as passed through the annotation argument.

parameters: List with the parameters.

Methods

All methods for class TssNorm, as well as:

identify methods:

identifyStartSites: signature(x="TssNorm")
Identify TSS in the normalized data.

get methods:

expect: signature(x="TssResult")
Get the expectation for non-specific reads, seestart.

tss: signature(x="TssResult")
Get the identified transcription start sites, seestart.

[: signature(x="TssResult")
Subset the object, by name or index.

asRangedData methods:

tssAsRangedData: signature(x="TssResult")
Convert the tss predictions, as obtained by the tss method, to an object of class RangedData.

Author(s)

Maintainer: Julian Gehring <julian.gehring@fdm.uni-freiburg.de>

See Also

Classes: TssData, TssNorm, TssResult

Methods: segmentizeCounts, normalizeCounts, identifyStartSites, get-methods, plot-methods, asRangedData-methods

Functions: subtract-functions

Data set: physcoCounts

Package: TSSi-package

Examples

```
showClass("TssResult")
```

Index