Package 'transcriptR'

November 25, 2026

Type Package

Title An Integrative Tool for ChIP- And RNA-Seq Based Primary Transcripts Detection and Quantification

Version 1.42.0

Date 2021-11-20

Author Armen R. Karapetyan <armen.karapetyan87@gmail.com>

Maintainer Armen R. Karapetyan <armen.karapetyan87@gmail.com>

Description The differences in the RNA types being sequenced have an impact on the resulting sequencing profiles. mRNA-seq data is enriched with reads derived from exons, while GRO-, nucRNA- and chrRNA-seq demonstrate a substantial broader coverage of both exonic and intronic regions. The presence of intronic reads in GRO-seq type of data makes it possible to use it to computationally identify and quantify all de novo continuous regions of transcription distributed across the genome. This type of data, however, is more challenging to interpret and less common practice compared to mRNA-seq. One of the challenges for primary transcript detection concerns the simultaneous transcription of closely spaced genes, which needs to be properly divided into individually transcribed units. The R package transcriptR combines RNA-seq data with ChIP-seq data of histone modifications that mark active Transcription Start Sites (TSSs), such as, H3K4me3 or H3K9/14Ac to overcome this challenge. The advantage of this approach over the use of, for example, gene annotations is that this approach is data driven and therefore able to deal also with novel and case specific events. Furthermore, the integration of ChIP- and RNA-seq data allows the identification all known and novel active transcription start sites within a given sample.

Imports BiocGenerics, caret, chipseq, GenomicAlignments, GenomicRanges, GenomicFeatures, GenomeInfoDb, ggplot2, graphics, grDevices, IRanges (>= 2.11.15), pROC, reshape2, Rsamtools, rtracklayer, S4Vectors, stats, utils

Depends methods, R (>= 3.3)

License GPL-3

Collate 'utils.R' 'TranscriptionDataSet-class.R' 'ChipDataSet-class.R' 'show.R' 'TranscriptionDataSet-generics.R'

'TranscriptionDataSet-methods.R' 'ChipDataSet-generics.R' 'ChipDataSet-methods.R'

Suggests BiocStyle, knitr, rmarkdown, TxDb.Hsapiens.UCSC.hg19.knownGene, testthat, e1071

VignetteBuilder knitr

biocViews ImmunoOncology, Transcription, Software, Sequencing, RNASeq, Coverage

RoxygenNote 7.3.3

NeedsCompilation no

LazyData true

Encoding UTF-8

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

git_branch RELEASE_3_24

git_last_commit 7f940ef

git_last_commit_date 2026-11-23

Repository Bioconductor 3.24

Date/Publication 2026-11-24

Contents

addFeature . . . 3
annot . . . 4
annotateTranscripts . . . 4
breakTranscriptsByPeaks . . . 5
cds . . . 7
ChipDataSet-class . . . 7
constructCDS . . . 8
constructTDS . . . 10
detectTranscripts . . . 11
estimateBackground . . . 13
estimateGapDistance . . . 14
exportCoverage . . . 16
getConfusionMatrix . . . 17
getGenomicAnnot . . . 18
getPeaks . . . 19
getPredictorSignificance . . . 20
getProbTreshold . . . 21
getQuadProb . . . 21
getTestedGapDistances . . . 23
getTranscripts . . . 23
peaksToBed . . . 24
plotErrorRate . . . 26
plotFeatures . . . 27
plotGenomicAnnot . . . 29
plotROC . . . 30
predictStrand . . . 31
predictTssOverlap . . . 33
show . . . 34

tds 35
TranscriptionDataSet-class 35
transcriptsToBed 36

Index **38**

---

addFeature *addFeature*

Description

Add feature(s) to the ChipDataSet object.

Usage

```
addFeature(object, feature)

## S4 method for signature 'ChipDataSet'
addFeature(object, feature)
```

Arguments

object	A ChipDataSet object.
feature	List. User specified characteristics of the peaks.

Value

The user specified feature(s) will be added to the slot features of the provided ChipDataSet object.

Author(s)

Armen R. Karapetyan

See Also

constructCDS predictTssOverlap

Examples

```
### Load ChipDataSet object
data(cds)

### As an example create some fake data
N <- length(getPeaks(cds))
fake.data <- rnorm(n = N)

addFeature(object = cds, feature = list(fake = fake.data))

### View newly added feature
getPeaks(cds)
```

annot *Reference annotation (knownGene from UCSC)*

Description

annot is an object of GRanges class, containing genomic coordinates of the genes located on human chromosome 15 (chr15:63261757-84081194).

Usage

```
annot
```

Format

GRanges object

Examples

```
data(annot)
annot
```

annotateTranscripts *annotateTranscripts*

Description

Annotate detected transcripts by the available reference annotations based on genomic overlap.

Usage

```
annotateTranscripts(object, annot, min.overlap = 0.3)

## S4 method for signature 'TranscriptionDataSet,GRanges'
annotateTranscripts(object, annot, min.overlap = 0.3)
```

Arguments

object	A TranscriptionDataSet object.
annot	GRanges. Reference annotations.
min.overlap	Numeric. A minimal proportion of the overlap between transcript and annotation. A value in the range (0, 1]. Default: 0.3.

Details

Genomic overlap between transcript and annotation is calculated as the mean of two proportions: 1) proportion of the transcript length overlapping annotation; 2) proportion of the annotation length overlapping transcript. This approach levels off differences in length between transcript and annotation and, thus better suitable for cases in which the length of either transcript or annotation is much longer than of compared element.

If there is an overlap between transcript and annotation, the ID of the associated annotation will be linked to the transcript.

Value

An extra column `annotation.overlap` will be added to the metadata portion of the `GRanges` object which is stored in the `transcripts` slot of the provided `TranscriptionDataSet` object.

Author(s)

Armen R. Karapetyan

See Also

`detectTranscripts`

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Detect transcripts
detectTranscripts(object = tds, coverage.cutoff = 5, gap.dist = 4000,
estimate.params = TRUE, combine.by.annot = FALSE, annot = annot)

### Annotate detected transcripts
annotateTranscripts(object = tds, annot = annot)

### View detected transcripts and associated annotations
getTranscripts(tds)
```

---

breakTranscriptsByPeaks

breakTranscriptsByPeaks

Description

The function divides closely spaced transcripts into individually transcribed units using the detected active transcription start sites.

Usage

```
breakTranscriptsByPeaks(tdsObj, cdsObj, estimate.params = TRUE)

## S4 method for signature 'TranscriptionDataSet,ChipDataSet'
breakTranscriptsByPeaks(tdsObj, cdsObj, estimate.params = TRUE)
```

Arguments

`tdsObj`	A `TranscriptionDataSet` object.
`cdsObj`	A `ChipDataSet` object.
`estimate.params`	`Logical`. Whether to estimate expression level and coverage density of the newly detected transcripts. Default: TRUE.

Details

One of the challenges for primary transcript detection concerns the simultaneous transcription of closely spaced genes, which needs to be properly divided into individually transcribed units. `transcriptR` combines RNA-seq data with ChIP-seq data of histone modifications that mark active Transcription Start Sites (TSSs), such as, H3K4me3 or H3K9/14Ac to overcome this challenge. The advantage of this approach over the use of, for example, gene annotations is that this approach is data driven and therefore able to deal also with novel and case specific events. Furthermore, the integration of ChIP- and RNA-seq data allows the identification all known and novel active transcription start sites within a given sample. Transcription initiation within a peak region is investigated by comparing RNA-seq read densities upstream and downstream of empirically determined TSSs. Closely spaced transcripts are divided into individually transcribed units using the detected active TSSs.

Value

The slot `transcripts` of the provided `TranscriptionDataSet` object will be updated by the `GRanges` object, containing transcripts and, if estimated, corresponding expression levels.

Author(s)

Armen R. Karapetyan

See Also

`predictStrand`

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load ChipDataSet object
data(cds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Detect transcripts
detectTranscripts(object = tds, coverage.cutoff = 5, gap.dist = 4000,
estimate.params = TRUE, combine.by.annot = FALSE, annot = annot)

### Classify peaks on gene associated and background
predictTssOverlap(object = cds, feature = "pileup", p = 0.75)

### Predict peak 'strand'
predictStrand(cdsObj = cds, tdsObj = tds, coverage.cutoff = 5,
quant.cutoff = 0.1, win.size = 2500)

### If `estimate.params = TRUE`, FPKM and coverage density will be re-calculated
breakTranscriptsByPeaks(tdsObj = tds, cdsObj = cds, estimate.params = TRUE)

### View detected transcripts
getTranscripts(tds)
```

cds *Example of* ChipDataSet *object.*

Description

cds is an object of ChipDataSet class, containing H3K4me3 active histone mark ChIP-seq peaks from human chromosome 15 (chr15:63261757-84081194) profiled in prostate cancer LNCaP cells.

Usage

```
cds
```

Format

ChipDataSet object

Examples

```
data(cds)
cds
```

ChipDataSet-class *ChipDataSet*

Description

The ChipDataSet is a S4 class to store input values, intermediate calculations and results of ChIP-seq peaks analysis.

Slots

`peaks` GRanges. ChIP-seq peaks.

`region` GRanges. Genomic region(s) to extract peaks from.

`genomicAnnotation` Data.Frame. Genomic distribution of the peaks at distinct genomic features (TSSs, exons, introns, intergenic regions).

`features` Data.Frame. Estimated characteristics (features) of the peaks.

`tssOverlapPrediction` List. Prediction of the gene associated peaks. The following elements are reported:

- 'predicted.tssOverlap' - predicted class (yes - gene associated; no - background) and probability of a ChIP-seq peak being classified as gene associated.
- 'confusionMatrix' - cross-tabulation of observed and predicted classes with associated statistics.
- 'logFitSummary' - statistical significance of the predictors used in the logistic regression.
- 'roc' - results of the receiver operating characteristic analysis.

`strandPrediction` List. Prediction of the peak strandedness. The following elements are reported:

- 'predicted.strand' - predicted ChIP-seq peak strand.
- 'probability.cutoff' - probability cutoff for q2.
- 'results.plus' - intermediate calculations for the forward DNA strand.
- 'results.minus' - intermediate calculations for the reverse DNA strand.

Author(s)

Armen R. Karapetyan

See Also

constructCDS predictTssOverlap predictStrand

---

constructCDS *constructCDS*

Description

The function constructs an object of class ChipDataSet, which is a container for holding processed sequencing data and the results of all downstream analyses. All the slots of the created object are filled during the workflow by applying specific functions to the object directly.

Usage

```
constructCDS(
  peaks,
  reads,
  region,
  TxDb,
  tssOf = c("gene", "transcript"),
  tss.region = c(-2000, 2000),
  reduce.peaks = FALSE,
  gapwidth = 1000,
  fragment.size,
  unique = TRUE,
  swap.strand = FALSE,
  param = NULL
)
```

Arguments

peaks	A path to a file with peaks. The file needs to have at least 3 columns (tab-separated): chromosome, start (peak), end (peak). The 4th column - name (peak id) is optional.
reads	A path to a BAM file with sequencing reads.
region	GRanges. Genomic region(s) to extract reads from. If not supplied, all the reads from a BAM file are extracted.
TxDb	TxDb object.
tssOf	Character. Extract Transcription Start Site (TSS) regions from either "gene" or "transcript" annotations. Default: "gene".

tss.region	A numeric vector of length two, which specifies the size of TSS region. Default: -2kb to 2kb.
reduce.peaks	Logical. Whether to merge neighboring peaks. Default: FALSE.
gapwidth	Numeric. A minimum distance (in bp) between peaks to merge. Default: 1000.
fragment.size	Numeric. Extend read length to the fragment size.
unique	Logical. Whether to remove duplicated reads (based on the genomic coordinates). Default: FALSE.
swap.strand	Logical. Whether to reverse the strand of the read. Default: FALSE.
param	ScanBamParam object influencing what fields and which records (reads) are imported from the Bam file. Default: NULL.

Details

The function constructCDS initializes a ChipDataSet object, by providing the paths to the input files and information relevant to the ChIP-seq library preparation procedure. During the object construction the following steps are executed:

- The peak information is converted into the object of GRanges class.
- The genomic distribution of the peaks is evaluated (exonic, intronic, intergenic, TSSs).
- Each peak in the data set is functionally characterized:
  - length - the length of a peak (in base pairs).
  - fragments - total number of fragments overlapping a peak region.
  - density - number of fragments per base pair of the peak length.
  - pileup - highest fragment pileup in each peak region.
  - tssOverlap - overlap (binary, yes/no) of the peak with the annotated TSS region.

  The estimated features are used to predict which of the peaks are gene associated in the analysis downstream.

As many peak-calling algorithms tend to divide broader peaks into the several narrower closely spaced peaks, it is advised to merge these end-to-end peaks to decrease the number of false positives and prevent unnecessary truncation of transcripts in the downstream analysis.

Value

An object of class ChipDataSet.

Author(s)

Armen R. Karapetyan

See Also

ChipDataSet predictTssOverlap

Examples

```
### Load ChipDataSet object
data(cds)

### View a short summary of the object
cds
```

constructTDS *constructTDS*

Description

The function constructs an object of class TranscriptionDataSet, which is a container for holding processed sequencing data and the results of all downstream analyses. All the slots of the created object are filled during the workflow by applying specific functions to the object directly.

Usage

```
constructTDS(
  file,
  region,
  fragment.size = 250,
  unique = FALSE,
  paired.end = FALSE,
  swap.strand = FALSE,
  param = NULL
)
```

Arguments

file	A path to a BAM file with sequencing reads.
region	GRanges. Genomic region(s) to extract reads from. If not supplied, all the reads from a BAM file are extracted.
fragment.size	Numeric. Extend read length to the fragment size. Default: 250.
unique	Logical. Whether to remove duplicated reads (based on the genomic coordinates). Default: FALSE.
paired.end	Logical. Whether to treat a BAM file as paired-end. Default: FALSE.
swap.strand	Logical. Whether to reverse the strand of the read. Default: FALSE.
param	ScanBamParam object influencing what fields and which records (reads) are imported from a BAM file. Default: NULL.

Details

The slots fragments, fragmentSize, region, coveragePlus, coverageMinus are filled during the object construction. The fragments holds information about genomic coordinates of the sequenced fragments (reads extended to the fragmento size). coveragePlus and coverageMinus for each position in the genome counts the number of fragments that cover it (for the details, see coverage). region holds information about the region used for fragments extraction.

Value

An object of class TranscriptionDataSet.

Author(s)

Armen R. Karapetyan

Examples

```
### Load TranscriptionDataSet object
data(tds)

### View a short summary of the object
tds
```

---

detectTranscripts *detectTranscripts*

Description

The function dissects transcribed regions (transcripts) genome-wide and performs expression level quantification.

Usage

```
detectTranscripts(
  object,
  coverage.cutoff,
  gap.dist,
  estimate.params = TRUE,
  total.reads,
  combine.by.annot = FALSE,
  annot
)

## S4 method for signature 'TranscriptionDataSet'
detectTranscripts(
  object,
  coverage.cutoff,
  gap.dist,
  estimate.params = TRUE,
  total.reads,
  combine.by.annot = FALSE,
  annot
)
```

Arguments

object	A TranscriptionDataSet object.
coverage.cutoff	Numeric. A cutoff value to discard regions with the low fragments coverage, representing expression noise. By default, the value stored in the coverageCutoff slot of the supplied TranscriptionDataSet object is used. The optimal cutoff value can be calculated by estimateBackground function call.
gap.dist	Numeric. Maximum allowed distance between transcribed regions to be merged into the one transcript. By default, the value stored in the gapDistanceTest slot of the supplied TranscriptionDataSet object is used. The optimal gap distance can be calculated by estimateGapDistance function call.

`estimate.params`
: `Logical`. Whether to estimate expression level and coverage density of the detected transcripts. Default: TRUE.

`total.reads`
: `Numeric`. Total number of reads used for the normalization, when calculating FPKM. By default, the total number of reads stored in the provided `TranscriptionDataSet` object is used.

`combine.by.annot`
: `Logical`. Whether to combine transcripts overlapping the same reference annotation. Default: FALSE.

`annot`
: `GRanges`. Reference annotations.

Details

The function uses two parameters to identify transcribed regions: `coverage.cutoff` and `gap.dist` as calculated by the `estimateBackground` and `estimateGapDistance`, respectively and stored in the `TranscriptionDataSet` object. Alternatively, the user may specify his/her own values to be passed to the function. By increasing the `gap.dist`, fewer transcripts of longer size will be identified, and an increase in the `coverage.cutoff` will result in fewer transcripts of shorter size (a typical transcript tends to have a lower fragments coverage at the 3' end, and thus, the `coverage.cutoff` value will have an impact on the resulting length of the detected transcript).

If `estimate.params` is set TRUE, the following metrics are estimated for each transcript:

- `length` - transcript length (in base pairs).
- `bases.covered` - the number of bases covered by the sequencing fragments.
- `coverage` - the proportion of transcript length covered by fragments. Value in the range (0, 1].
- `fragments` - total number of fragments per transcript.
- `fpkm` - Fragments Per Kilobase of transcript per Million mapped reads.

The `coverage` is a measure of how densely the transcript is covered by the sequencing fragments. Modestly/highly expressed transcripts will have a value close to 1, whereas lowly expressed transcripts will have a value close to 0, indicating the sparse distribution of sequencing fragments along the transcript body.

Value

The slot `transcripts` of the provided `TranscriptionDataSet` object will be updated by the `GRanges` object, containing detected transcripts and, if estimated, corresponding expression levels.

Author(s)

Armen R. Karapetyan

See Also

`constructTDS`

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Detect transcripts
detectTranscripts(object = tds, coverage.cutoff = 5, gap.dist = 4000,
estimate.params = TRUE, combine.by.annot = FALSE, annot = annot)

### View detected transcripts
getTranscripts(tds)
```

`estimateBackground` *estimateBackground*

Description

Gene expression is a stochastic process, which often results in substantial expression noise. To obtain a putative set of transcribed regions, it is necessary to identify those regions that are expressed significantly above the background level. Using a Poisson-based approach for estimating the noise distribution from the frequency of the transcribed regions with the low fragments coverage, `estimateBackground` function returns a coverage cutoff value for a specific False Discovery Rate (FDR).

Usage

```
estimateBackground(object, fdr.cutoff = 0.05)

## S4 method for signature 'TranscriptionDataSet'
estimateBackground(object, fdr.cutoff = 0.05)
```

Arguments

`object`	A `TranscriptionDataSet` object.
`fdr.cutoff`	`Numeric`. False Discovery Rate cutoff value. Default: 0.05.

Value

The slots `coverageCutoffFdr` and `coverageCutoff` of the provided `TranscriptionDataSet` object will be updated by the FDR cutoff value used in the calculations and by the corresponding estimated coverage cutoff value, respectively.

Author(s)

Armen R. Karapetyan

See Also

`constructTDS` `peakCutoff`

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Estimate coverage cutoff at different FDR levels
estimateBackground(object = tds, fdr = 0.01)
```

`estimateGapDistance` *estimateGapDistance*

Description

The ultimate goal of `transcriptR` is to identify continuous regions of transcription. However, in some areas of the genome it is not possible to detect transcription, because of the presence of the low mappability regions and (high copy number) repeats. Sequencing reads can not be uniquely mapped to these positions, leading to the formation of gaps in otherwise continuous coverage profiles and segmentation of transcribed regions into multiple smaller fragments. The gap distance describes the maximum allowed distance between adjacent fragments to be merged into one transcript. To choose the optimal value for the gap distance, the detected transcripts should largely be in agreement with available reference annotations. To accomplish this, the function is build on the methodology proposed by Hah et al. (Cell, 2011). In brief, the two types of erros are defined:

- `dissected` error - the ratio of annotations that is segmented into two or more fragments.
- `merged` error - the ratio of non-overlapping annotations that merged by mistake in the experimental data.

There is an interdependence between two types of errors. Increasing the gap distance decreases the `dissected` error, by detecting fewer, but longer transcripts, while the `merged` error will increase as more detected transcripts will span multiple annotations. The gap distance with the lowest sum of two error types is chosen as the optimal value.

Usage

```
estimateGapDistance(
  object,
  annot,
  coverage.cutoff,
  filter.annot = TRUE,
  fpkm.quantile = 0.25,
  gap.dist.range = seq(from = 0, to = 10000, by = 100)
)

## S4 method for signature 'TranscriptionDataSet,GRanges'
estimateGapDistance(
  object,
  annot,
  coverage.cutoff,
  filter.annot = TRUE,
  fpkm.quantile = 0.25,
  gap.dist.range = seq(from = 0, to = 10000, by = 100)
)
```

Arguments

object	A TranscriptionDataSet object.
annot	GRanges. Reference annotations.
coverage.cutoff	Numeric. A cutoff value to discard regions with the low fragments coverage, representing expression noise. By default, the value stored in the coverageCutoff slot of the supplied TranscriptionDataSet object is used. The optimal cutoff value can be calculated by estimateBackground function call.
filter.annot	Logical. Whether to filter out lowly expressed annotations, before estimating error rates. Default: TRUE.
fpkm.quantile	Numeric. A number in a range (0, 1). A cutoff value used for filtering lowly expressed annotations. The value corresponds to the FPKM quantile estimated for the supplied annotations. Default: 0.25.
gap.dist.range	A numeric vector specifying a range of gap distances to test. By default, the range is from 0 to 10000 with a step of 100.

Value

The slot gapDistanceTest of the provided TranscriptionDataSet object will be updated by the data.frame, containing estimated error rates for each tested gap distance (see getTestedGapDistances, for the details).

Author(s)

Armen R. Karapetyan

References

Hah N, Danko CG, Core L, Waterfall JJ, Siepel A, Lis JT, Kraus WL. A rapid, extensive, and transient transcriptional response to estrogen signaling in breast cancer cells. Cell. 2011.

See Also

constructTDS plotErrorRate getTestedGapDistances

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Estimate gap distance minimazing error rate
### Define the range of gap distances to test
gdr <- seq(from = 0, to = 10000, by = 1000)

estimateGapDistance(object = tds, annot = annot, coverage.cutoff = 5,
filter.annot = FALSE, gap.dist.range = gdr)

### View estimated gap distance
tds
```

exportCoverage *exportCoverage*

Description

RNA-seq coverage profiles for both forward and reverse DNA strand can be visualized separately in the UCSC genome browser using `exportCoverage`. This function can generate tracks in BigWig and bedGraph formats, which can be uploaded to the genome browser as custom tracks.

Usage

```
exportCoverage(
  object,
  file,
  type,
  strand,
  color,
  filter.by.coverage.cutoff = FALSE,
  coverage.cutoff = NULL,
  rpm = FALSE,
  total.reads
)

## S4 method for signature 'TranscriptionDataSet'
exportCoverage(
  object,
  file,
  type,
  strand,
  color,
  filter.by.coverage.cutoff = FALSE,
  coverage.cutoff = NULL,
  rpm = FALSE,
  total.reads
)
```

Arguments

`object`	A `TranscriptionDataSet` object.
`file`	`Character`. A file name.
`type`	`Character`. Track type, either "bigWig" or "bedGraph". Default: "bedGraph".
`strand`	`Character`. The strand to create a track for. One of ["+", "-"].
`color`	Object of class "integer" representing the track color (as from col2rgb). Only works with tracks of type "bedGraph". Default: c(0L, 0L, 255L).
`filter.by.coverage.cutoff`	`Logical`. Whether to discard regions with low fragment coverage, representing expression noise from the resulting track. Default: FALSE.

`coverage.cutoff`
: `Numeric`. A cutoff value to discard regions with the low fragment coverage, representing expression noise. By default, the value stored in the `coverageCutoff` slot of the supplied object is used. The optimal cutoff value can be calculated by `estimateBackground` function call. Default: NULL.

`rpm`
: `Logical`. Whether to perform normalization ('Reads Per Million'). Default: FALSE.

`total.reads`
: `Numeric`. Total number of reads used for normalization. By default, the total number of reads stored in the provided `TranscriptionDataSet` object is used.

Details

There is an option to filter coverage profiles by the coverage cutoff value, either estimated for a specific FDR via `estimateBackground` or a user specified value. By default, the coverage cutoff value stored in the `TranscriptionDataSet` object is used. In order to make an informed decision about a proper FDR level, it is useful to explore the output at different FDR levels and determine the optimal cutoff value.

Value

A file in either BigWig or bedGraph format.

Author(s)

Armen R. Karapetyan

See Also

`estimateBackground` UCSC genome browser BigWig

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Look at the coverage profile of the regions expressed above the background level
# exportCoverage(object = tds, file = "plus.bg", type = "bedGraph", strand = "+",
# filter.by.coverage.cutoff = TRUE, coverage.cutoff = 3, rpm = FALSE)

### Or check the raw coverage (all expressed regions)
# exportCoverage(object = tds, file = "plus_raw.bg", type = "bedGraph",
# strand = "+", filter.by.coverage.cutoff = FALSE, rpm = FALSE)
```

`getConfusionMatrix` *getConfusionMatrix*

Description

Retrieve a cross-tabulation of observed and predicted classes (prediction of gene associated peaks) with associated statistics.

Usage

```
getConfusionMatrix(object)

## S4 method for signature 'ChipDataSet'
getConfusionMatrix(object)
```

Arguments

object	A ChipDataSet object.

Value

An object of Confusion Matrix class. For the details see caret package.

Author(s)

Armen R. Karapetyan

See Also

predictTssOverlap confusionMatrix

Examples

```
### Load ChipDataSet object
data(cds)

getConfusionMatrix(cds)
```

---

getGenomicAnnot *getGenomicAnnot*

Description

Retrieve genomic distribution of ChIP-seq peaks at distinct genomic features (exons, introns, TSSs, intergenic regions)

Usage

```
getGenomicAnnot(object)

## S4 method for signature 'ChipDataSet'
getGenomicAnnot(object)
```

Arguments

object	A ChipDataSet object.

Details

A simple quality check of the supplied ChIP-seq peaks can be performed by investigating their genomic distribution. Ideally, these peaks should demonstrate substantial enrichment at TSS regions. Enrichment of the peaks at a given genomic feature (e.g. TSS) is defined as the ratio between the observed and expected number of peaks. The expected number of peaks is calculated from the proportion of the genome covered by the given genomic feature.

Value

A four column `Data.Frame`, storing information about observed and expected number of peaks at distinct genomic features.

Author(s)

Armen R. Karapetyan

See Also

`constructCDS`

Examples

```
### Load ChipDataSet object
data(cds)

getGenomicAnnot(cds)
```

`getPeaks` *getPeaks*

Description

Retrieve ChIP-seq peak information from the `ChipDataSet` object.

Usage

```
getPeaks(object)

## S4 method for signature 'ChipDataSet'
getPeaks(object)
```

Arguments

`object`	A `ChipDataSet` object.

Value

A `GRanges` object.

Author(s)

Armen R. Karapetyan

See Also

constructCDS predictTssOverlap predictStrand

Examples

```
### Load ChipDataSet object
data(cds)

getPeaks(cds)
```

---

getPredictorSignificance
getPredictorSignificance

Description

Retrieve significance of each predictor used in the classification model fit (prediction of gene associated peaks).

Usage

```
getPredictorSignificance(object)

## S4 method for signature 'ChipDataSet'
getPredictorSignificance(object)
```

Arguments

object	A ChipDataSet object.

Value

A vector of p-values.

Author(s)

Armen R. Karapetyan

See Also

predictTssOverlap

Examples

```
### Load ChipDataSet object
data(cds)

getPredictorSignificance(cds)
```

getProbTreshold *getProbTreshold*

Description

Retrieve estimated P(q2) threshold, used to select peaks with a putative transcription initiation event.

Usage

```
getProbTreshold(object)

## S4 method for signature 'ChipDataSet'
getProbTreshold(object)
```

Arguments

object	A ChipDataSet object.

Value

Estimated P(q2) threshold.

Author(s)

Armen R. Karapetyan

See Also

predictStrand

Examples

```
### Load ChipDataSet object
data(cds)

getProbTreshold(cds)
```

getQuadProb *getQuadProb*

Description

Retrieve all internal calculations performed by predictStrand function.

Usage

```
getQuadProb(object, strand)

## S4 method for signature 'ChipDataSet'
getQuadProb(object, strand)
```

Arguments

object	A ChipDataSet object.
strand	Character. The strand to extract calculations for. One of ["+", "-"].

Value

A nine column Data.Frame, where each row corresponds to a ChIP-seq peak and each column keeps one of the intermediate calculations:

- max.cov - maximum coverage of the RNA-seq fragments inside the peak region.
- pass.cov.treshold - whether the max.cov exceeds the coverage.cutoff, either user defined or estimated from RNA-seq data by estimateBackground function call and stored in TranscriptionDataSet object.
- q1q2.sepline.coord - genomic coordinate corresponding to the transcription start position inside the peak region.
- q1.coord - genomic coordinates of q1.
- q2.coord - genomic coordinates of q2.
- q1.count - total number of fragments in q1.
- q2.count - total number of fragments in q2.
- q1.prob - probability of a fragment being sampled from the q1.
- q2.prob - probability of a fragment being sampled from the q2.

Author(s)

Armen R. Karapetyan

See Also

predictStrand

Examples

```
### Load ChipDataSet object
data(cds)

### Load TranscriptionDataSet object
data(tds)

head(getQuadProb(cds, strand = "+"))
head(getQuadProb(cds, strand = "-"))
```

getTestedGapDistances *getTestedGapDistances*

Description

Retrieve a data.frame, containing a range of tested gap distances and estimated error rates.

Usage

```
getTestedGapDistances(object)

## S4 method for signature 'TranscriptionDataSet'
getTestedGapDistances(object)
```

Arguments

object	A TranscriptionDataSet object.

Value

A data.frame containing estimated error rates (dissected, merged and sum of two errors) and corresponding gap distances.

Author(s)

Armen R. Karapetyan

See Also

estimateGapDistance

Examples

```
### Load TranscriptionDataSet object
data(tds)

head(getTestedGapDistances(tds))
```

getTranscripts *getTranscripts*

Description

Retrieve transcripts information from the TranscriptionDataSet object.

Usage

```
getTranscripts(object, min.length, min.fpkm, min.coverage)

## S4 method for signature 'TranscriptionDataSet'
getTranscripts(object, min.length, min.fpkm, min.coverage)
```

Arguments

object	A TranscriptionDataSet object.
min.length	Numeric. A minimum length (in bp) of the reported transcripts.
min.fpkm	Numeric. A minimum FPKM of the reported transcripts.
min.coverage	Numeric. A minimum coverage of the reported transcripts.

Details

The coverage is a measure of how densely the transcript is covered by the sequencing fragments. Modestly/highly expressed transcripts will have a value close to 1, whereas lowly expressed transcripts will have a value close to 0, indicating the sparse distribution of sequencing fragments along the transcript body.

Value

A GRanges object.

Author(s)

Armen R. Karapetyan

See Also

detectTranscripts

Examples

```
### Load TranscriptionDataSet object
data(tds)

### View detected transcripts
getTranscripts(tds)
```

---

peaksToBed *peaksToBed*

Description

A convenient way to explore output of the predictions made on the ChIP peaks is to visualize them in the UCSC genome browser. The peaksToBed function returns a file in BED format, which can be uploaded directly to the genome browser. To improve the visual perception, peaks are color-coded by the predicted strand.

Usage

```
peaksToBed(
  object,
  file,
  strand.pred.color = c("blue", "red", "green4", "black"),
  gene.associated.peaks = TRUE
)

## S4 method for signature 'ChipDataSet'
peaksToBed(
  object,
  file,
  strand.pred.color = c("blue", "red", "green4", "black"),
  gene.associated.peaks = TRUE
)
```

Arguments

object	A ChipDataSet object.
file	Character. A file name.
strand.pred.color	Character. A vector of length four, specifying colors of the predicted peak strand. The order of colors corresponds to 1) plus strand, 2) minus strand, 3) bideractional, 4) strand not determined. Default: c("blue", "red", "green4", "black").
gene.associated.peaks	Logical. Whether to return gene associated peaks (based on the prediction) only. Default: TRUE

Value

A file in the BED format.

Author(s)

Armen R. Karapetyan

See Also

constructCDS predictTssOverlap predictStrand UCSC genome browser BED

Examples

```
### Load ChipDataSet object
data(cds)

### Load TranscriptionDataSet object
data(tds)

### Classify peaks on gene associated and background
predictTssOverlap(object = cds, feature = "pileup", p = 0.75)

### Predict peak 'strandedness'
```

```
predictStrand(cdsObj = cds, tdsObj = tds, coverage.cutoff = 5,
quant.cutoff = 0.1, win.size = 2500)

# peaksToBed(object = cds, file = "peaks.bed", gene.associated.peaks = TRUE)
```

---

plotErrorRate *plotErrorRate*

Description

A simple helper function that plot results of `estimateGapDistance` function call.

Usage

```
plotErrorRate(
  object,
  color = c("#1B9E77", "#D95F02", "#7570B3"),
  xlab = "Gap distance (kb)",
  ylab = "Error rate",
  ...
)

## S4 method for signature 'TranscriptionDataSet'
plotErrorRate(
  object,
  color = c("#1B9E77", "#D95F02", "#7570B3"),
  xlab = "Gap distance (kb)",
  ylab = "Error rate",
  ...
)
```

Arguments

`object`	A `TranscriptionDataSet` object.
`color`	`Character`. Colors to be used to plot estimated errors.
`xlab`	`Character`. Label of the x-axis.
`ylab`	`Character`. Lable of the y-axis.
`...`	Further arguments passed to plot.

Details

The tested gap distances are plotted on the x-axis and corresponding error rates on the y-axis. Three curved lines depict the two error types calculated by `estimateGapDistance` and the sum of both errors. The vertical dashed line depicts the gap distance with the smallest sum of two errors.

Value

plot

Author(s)

Armen R. Karapetyan

See Also

estimateGapDistance

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Estimate gap distance minimazing error rate
### Define the range of gap distances to test
gdr <- seq(from = 0, to = 10000, by = 1000)

estimateGapDistance(object = tds, annot = annot, coverage.cutoff = 5,
filter.annot = FALSE, gap.dist.range = gdr)

plotErrorRate(object = tds, lwd = 2)
```

---

plotFeatures *plotFeatures*

Description

Visualize the relations between predictors and response variable ('tssOverlap').

Usage

```
plotFeatures(
  object,
  plot.type = c("box", "density"),
  feature,
  ncol,
  xlab,
  ylab,
  color = c("#E41A1C", "#377EB8"),
  alpha = 1
)

## S4 method for signature 'ChipDataSet'
plotFeatures(
  object,
  plot.type = c("box", "density"),
  feature,
  ncol,
  xlab,
```

```
  ylab,
  color = c("#E41A1C", "#377EB8"),
  alpha = 1
)
```

Arguments

object	A ChipDataSet object.
plot.type	One of ["box", "density"]. Default: "box"
feature	Feature to plot. By default, all the features are plotted.
ncol	Numeric. Arrange individual plots in columns. By default, the number of columns correspond to the number of features used for plotting.
xlab	Character. Title of the x-axis
ylab	Character. Title of the y-axis.
color	A character vector of length two. Default: ["#E41A1C","#377EB8"].
alpha	Color transparency. In a range [0, 1]. Default: 1.

Details

In order to discriminate between functional or gene associated peaks and non-functional or background peaks, each peak in the data set is characterized by several features. Moreover, the user might supply her/his own list of features with the addFeature. Prior to fitting the logistic model, the relations between predictors and response variable (tssOverlap) can be explored with plotFeatures. Based on the plots, poor predictors can be excluded from the analysis to improve the model fit.

Value

ggplot2 object.

Author(s)

Armen R. Karapetyan

See Also

constructCDS

Examples

```
### Load ChipDataSet object
data(cds)

### The data can be plotted in two ways
### As a boxplot
plotFeatures(object = cds, plot.type = "box")

### Or as a density plot
plotFeatures(object = cds, plot.type = "density")

### Additionally, only the subset of features can be shown
plotFeatures(object = cds, plot.type = "box", feature = c("pileup", "length"))

### The position of the graphs on the plot, can be adjusted by 'ncol' argument
```

```
plotFeatures(object = cds, plot.type = "box", ncol = 2)
```

---

plotGenomicAnnot *plotGenomicAnnot*

Description

Visualize genomic distribution of ChIP-seq peaks.

Usage

```
plotGenomicAnnot(object, plot.type = c("distrib", "enrich"), xlab, ylab, color)

## S4 method for signature 'ChipDataSet'
plotGenomicAnnot(object, plot.type = c("distrib", "enrich"), xlab, ylab, color)
```

Arguments

object	A ChipDataSet object.
plot.type	Character. One of ("distrib", "enrich"). Default: "enrich".
xlab	Character. Title of the x-axis.
ylab	Character. Title of the y-axis.
color	A character vector of length four, specifying colors for distinct genomic features (TSSs, exons, introns, intergenic regions).

Details

Genomic distribution of the peaks can be visualized in two ways, either by observing the total number of peaks overlapping given genomic feature or by looking at the enrichment levels.

Value

ggplot2 object.

Author(s)

Armen R. Karapetyan

See Also

constructCDS

Examples

```
### Load ChipDataSet object
data(cds)

### Plot the total number of peaks overlapping distinct genomic features
plotGenomicAnnot(object = cds, plot.type = "distrib")

### Plot enrichment of the peaks at a given genomic feature (e.g. TSS)
plotGenomicAnnot(object = cds, plot.type = "enrich")
```

plotROC *plotROC*

Description

Visualize the performance of the classification model fit (prediction of the gene associated peaks).

Usage

```
plotROC(object, ...)

## S4 method for signature 'ChipDataSet'
plotROC(object, ...)
```

Arguments

object	A ChipDataSet object.
...	Further arguments passed to plot.

Details

The plotROC is a simple wrapper for the plot function implemented in pROC package.

The ROC curve is created by plotting the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various threshold settings. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test. The area under the curve (AUC) is a measure of accuracy.

Value

ROC plot.

Author(s)

Armen R. Karapetyan

See Also

predictTssOverlap

Examples

```
### Load ChipDataSet object
data(cds)

### Classify peaks on gene associated and background
predictTssOverlap(object = cds, feature = "pileup", p = 0.75)

### Visualize the performance of the model fit
plotROC(object = cds, col = "red3", grid = TRUE, auc.polygon = TRUE)
```

predictStrand *predictStrand*

Description

The function evaluates transcription initiation within a peak region by comparing RNA-seq read densities upstream and downstream of an empirically determined transcription start sites. Putative transcription of both forward and reverse genomic strands is tested and the results are stored with each ChIP-seq peak.

Usage

```
predictStrand(
  cdsObj,
  tdsObj,
  coverage.cutoff,
  quant.cutoff = 0.1,
  win.size = 2500,
  prob.cutoff
)

## S4 method for signature 'ChipDataSet,TranscriptionDataSet'
predictStrand(
  cdsObj,
  tdsObj,
  coverage.cutoff,
  quant.cutoff = 0.1,
  win.size = 2500,
  prob.cutoff
)
```

Arguments

cdsObj	A ChipDataSet object.
tdsObj	A TranscriptionDataSet object.
coverage.cutoff	Numeric. A cutoff value to discard regions with the low fragments coverage, representing expression noise. By default, the value stored in the coverageCutoff slot of the supplied TranscriptionDataSet object is used. The optimal cutoff value can be calculated by estimateBackground function call.
quant.cutoff	Numeric. A cutoff value for the cumulative distribution of the RNA-seq signal along the ChIP-seq peak region. Must be in a range (0, 1). For the details, see step 1 in the "Details" section below. Default: 0.1.
win.size	Numeric. The size of the q1 and q2 regions flanking transcription start position at the 5' and 3', respectively. For the details, see step 2 in the "Details" section below. Default: 2500.
prob.cutoff	Numeric. A cutoff value for the probability of reads to be sampled from the q2 flanking region. If not supplied, the value estimated from the data will be used. Must be in a range (0, 1). For the details, see step 6 in the "Details" section below.

Details

RNA-seq data is incorporated to find direct evidence of active transcription from every putatively gene associated peak. In order to do this, we determine the 'strandedness' of the ChIP-seq peaks, using strand specific RNA-seq data. The following assumptions are made in order to retrieve the peak 'strandedness':

- The putatively gene associated ChIP-seq peaks are commonly associated with transcription initiation.
- This transcription initiation occurs within the ChIP peak region.
- When a ChIP peak is associated with a transcription initiation event, we expect to see a strand-specific increase in RNA-seq fragment count downstream the transcription initiation site.

Each peak in the data set is tested for association with transcription initiation on both strands of DNA. Steps 1-5 are performed for both forward and reverse DNA strand separately and step 6 combines the data from both strands. If the peak is identified as associated with the transcription on both strands, than it is considered to be a bidirectional.

ChIP peak 'strandedness' prediction steps:

1. Identify a location within the ChIP-seq peak near the transcription start site. This is accomplished by calculating the cumulative distribution of RNA-seq fragments within a peak region. The position is determined where 100% - 'quant.cutoff' * 100% of RNA-seq fragments are located downstream. This approach performs well on both gene-poor and gene-dense regions where transcripts may overlap.
2. Two equally sized regions are defined (q1 and q2), flanking the position identified in (1) on both sides. RNA-seq fragments are counted in each region.
3. ChIP peaks with an RNA-seq fragment coverage below an estimated threshold are discarded from the analysis.
4. The probability is calculated for RNA-seq fragments to be sampled from either q1 or q2. Based on the assumptions we stated above, a ChIP peak that is associated with transcription initiation should have more reads in q2 (downstream of the transcription start position) compared to q1, and subsequently, the probability of a fragment being sampled from q2 would be higher.
5. ChIP-seq peaks are divided into gene associated and background based on the prediction.
6. Iteratively, the optimal P(q2) threshold is identified, which balances out the False Discovery Rate (FDR) and False Negative Rate (FNR). Peaks with the P(q2) exceeding the estimated threshold are considered to be associated with the transcription initiation event.

Value

The slot `strandPrediction` of the provided `ChipDataSet` object will be updated by the the following elements: 'predicted.strand', 'probability.cutoff', 'results.plus' and 'results.minus'.

Author(s)

Armen R. Karapetyan

See Also

`ChipDataSet` `constructCDS`

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load ChipDataSet object
data(cds)

### Classify peaks on gene associated and background
predictTssOverlap(object = cds, feature = "pileup", p = 0.75)

### Predict peak 'strand'
predictStrand(cdsObj = cds, tdsObj = tds, coverage.cutoff = 5,
quant.cutoff = 0.1, win.size = 2500)

### View a short summary of the 'strand' prediction
cds

### View 'strand' prediction
getPeaks(cds)
```

---

predictTssOverlap *predictTssOverlap*

Description

The function classifies ChIP-seq peaks on gene associated and background using classification model based on a logistic regression.

Usage

```
predictTssOverlap(object, feature, p = 0.8)

## S4 method for signature 'ChipDataSet'
predictTssOverlap(object, feature, p = 0.8)
```

Arguments

object	A ChipDataSet object.
feature	Character. Specify feature(s) that are be used as predictors in the classification model. By default, all the features stored in the ChipDataSet object will be used.
p	Numeric. The proportion of data that is used to train the model. Default: 0.8.

Details

In order to improve the accuracy of the model the data is internally partitioned into a training and testing data sets. A repeated 10-Fold Cross-Validation is used to calculate performance measures on the training data set and to prevent over-fitting.

The model fit and validation is internally acomplised by the functions implemented in the caret package.

Value

The slot tssOverlapPrediction of the provided TranscriptionDataSet object will be updated by the the following elements: 'predicted.tssOverlap', 'confusionMatrix', 'logFitSummary' and 'roc'.

Author(s)

Armen R. Karapetyan

See Also

ChipDataSet constructCDS caret

Examples

```
### Load ChipDataSet object
data(cds)

predictTssOverlap(object = cds, feature = "pileup", p = 0.75)

### View a short summary of the gene associated peaks prediction
cds

### View peaks and associated prediction
getPeaks(cds)
```

---

show *Show method for ChipDataSet and TranscriptionDataSet objects*

Description

Show method for objects of class ChipDataSet and TranscriptionDataSet

Usage

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

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

Arguments

object	A TranscriptionDataSet or ChipDataSet object.

Value

Displays an overview of the TranscriptionDataSet or ChipDataSet object.

Examples

```
### Load TranscriptionDataSet object
data(tds)

### View a short summary of the object
tds
```

`tds` *Example of* `TranscriptionDataSet` *object.*

Description

`tds` is an object of `TranscriptionDataSet` class, containing nuclear RNA-seq data for human chromosome 15 (chr15:63261757-84081194), profiled in prostate cancer LNCaP cells.

Usage

```
tds
```

Format

`TranscriptionDataSet` object

Examples

```
data(tds)
tds
```

`TranscriptionDataSet-class` *TranscriptionDataSet*

Description

The `TranscriptionDataSet` is a S4 class to store input values, intermediate calculations and results of the transcripts detection and quantification analysis.

Slots

`bamFile Character`. Path to a BAM file.

`fragments GRanges`. Sequencing reads extended to the fragment size.

`fragmentSize Numeric`. Fragment size in base pairs (bp).

`region GRanges`. Genomic region(s) to extract reads from.

`coveragePlus RleList`. Fragment coverage profile for the forward DNA strand.

`coverageMinus RleList`. Fragment coverage profile for the reverse DNA strand.

`coverageCutoff Numeric`. Background coverage cutoff value.

`coverageCutoffFdr` `Numeric`. False Discovery Rate (FDR) used to estimate background coverage cutoff.

`gapDistanceTest` `Data.Frame`. Tested gap distances and corresponding error rates.

`gapDistanceTestCovCutoff` `Numeric`. Coverage cutoff value used for the gap distance estimation.

`transcripts` `GRanges`. Identified transcripts.

`transcriptsCovCutoff` `Numeric`. Coverage cutoff value used for the transcripts detection.

`transcriptsGapDist` `Numeric`. Gap distance value used for the transcripts detection.

`transcriptsNormalization` `Numeric`. Total number of reads used for normalization when calculating FPKM.

Author(s)

Armen R. Karapetyan

See Also

`constructTDS`

`transcriptsToBed` *transcriptsToBed*

Description

A convenient graphical way to explore the identified transcripts is to visualize them in the UCSC genome browser. The `transcriptsToBed` function returns a file in BED format, which can be directly uploaded to the genome browser. To improve the visual perception, transcripts are color-coded by DNA strand orientation.

Usage

```
transcriptsToBed(object, file, strand.color = c("blue", "red"))

## S4 method for signature 'GRanges'
transcriptsToBed(object, file, strand.color = c("blue", "red"))
```

Arguments

`object`	A `GRanges` object.
`file`	`Character`. A file name.
`strand.color`	A character vector of length two, specifying color for each DNA strand. Default: c("blue", "red").

Value

A file in the BED format.

Author(s)

Armen R. Karapetyan

See Also

estimateBackground UCSC genome browser BED

Examples

```
### Load TranscriptionDataSet object
data(tds)

### Load reference annotations (knownGene from UCSC)
data(annot)

### Detect transcripts
detectTranscripts(object = tds, coverage.cutoff = 5, gap.dist = 4000,
estimate.params = TRUE, combine.by.annot = FALSE, annot = annot)

### View detected transcripts
trx <- getTranscripts(tds)

### Export to BED
# transcriptsToBed(object = trx, file = "transcripts.bed",
# strand.color = c("blue", "red"))
```

Index

∗ **datasets**
annot, 4
cds, 7
tds, 35

addFeature, 3, *28*
addFeature,ChipDataSet-method *(addFeature)*, 3
annot, 4
annotateTranscripts, 4
annotateTranscripts,TranscriptionDataSet,GRanges-method *(annotateTranscripts)*, 4

breakTranscriptsByPeaks, 5
breakTranscriptsByPeaks,TranscriptionDataSet,ChipDataSet-method *(breakTranscriptsByPeaks)*, 5

cds, 7
ChipDataSet, *3*, *5*, *7–9*, *18–22*, *25*, *28–34*
ChipDataSet *(ChipDataSet-class)*, 7
ChipDataSet-class, 7
confusionMatrix, *18*
constructCDS, *3*, *8*, 8, *19, 20*, *25*, *28, 29*, *32*, *34*
constructTDS, 10, *12, 13*, *15*, *36*
coverage, *10*

detectTranscripts, *5*, 11, *24*
detectTranscripts,TranscriptionDataSet-method *(detectTranscripts)*, 11

estimateBackground, *11–13*, 13, *15*, *17*, *22*, *31*, *37*
estimateBackground,TranscriptionDataSet-method *(estimateBackground)*, 13
estimateGapDistance, *11, 12*, 14, *23*, *26, 27*
estimateGapDistance,TranscriptionDataSet,GRanges-method *(estimateGapDistance)*, 14
exportCoverage, 16
exportCoverage,TranscriptionDataSet-method *(exportCoverage)*, 16

getConfusionMatrix, 17
getConfusionMatrix,ChipDataSet-method *(getConfusionMatrix)*, 17
getGenomicAnnot, 18
getGenomicAnnot,ChipDataSet-method *(getGenomicAnnot)*, 18
getPeaks, 19
getPeaks,ChipDataSet-method *(getPeaks)*, 19
getPredictorSignificance, 20
getPredictorSignificance,ChipDataSet-method *(getPredictorSignificance)*, 20
getProbTreshold, 21
getProbTreshold,ChipDataSet-method *(getProbTreshold)*, 21
getQuadProb, 21
getQuadProb,ChipDataSet-method *(getQuadProb)*, 21
getTestedGapDistances, *15*, 23
getTestedGapDistances,TranscriptionDataSet-method *(getTestedGapDistances)*, 23
getTranscripts, 23
getTranscripts,TranscriptionDataSet-method *(getTranscripts)*, 23
GRanges, *4–10*, *12*, *15*, *19*, *24*, *35, 36*

peakCutoff, *13*
peaksToBed, 24
peaksToBed,ChipDataSet-method *(peaksToBed)*, 24
plotErrorRate, *15*, 26
plotErrorRate,TranscriptionDataSet-method *(plotErrorRate)*, 26
plotFeatures, 27
plotFeatures,ChipDataSet-method *(plotFeatures)*, 27
plotGenomicAnnot, 29
plotGenomicAnnot,ChipDataSet-method *(plotGenomicAnnot)*, 29
plotROC, 30
plotROC,ChipDataSet-method *(plotROC)*, 30
predictStrand, *6*, *8*, *20–22*, *25*, 31
predictStrand,ChipDataSet,TranscriptionDataSet-method *(predictStrand)*, 31
predictTssOverlap, *3*, *8, 9*, *18*, *20*, *25*, *30*, 33
predictTssOverlap,ChipDataSet-method *(predictTssOverlap)*, 33

pROC, *30*

RleList, *35*

ScanBamParam, *9, 10*
show, 34
show,ChipDataSet-method *(*show*)*, 34
show,TranscriptionDataSet-method *(*show*)*, 34

tds, 35
TranscriptionDataSet, *4, 5*, *10–13*, *15–17*, *22–24*, *26*, *31*, *34, 35*
TranscriptionDataSet *(*TranscriptionDataSet-class*)*, 35
TranscriptionDataSet-class, 35
transcriptsToBed, 36
transcriptsToBed,GRanges-method *(*transcriptsToBed*)*, 36
TxDb, *8*