

Package ‘ribosomeProfilingQC’

December 25, 2024

Type Package

Title Ribosome Profiling Quality Control

Version 1.19.0

Description Ribo-Seq (also named ribosome profiling or footprinting) measures translato~~me~~me (unlike RNA-Seq, which sequences the transcriptome) by direct quantification of the ribosome-protected fragments (RPFs). This package provides the tools for quality assessment of ribosome profiling. In addition, it can preprocess Ribo-Seq data for subsequent differential analysis.

License GPL (>=3) + file LICENSE

Encoding UTF-8

LazyData true

biocViews RiboSeq, Sequencing, GeneRegulation, QualityControl, Visualization, Coverage

VignetteBuilder knitr

RoxygenNote 7.3.2

Depends R (>= 4.0), GenomicRanges

Imports AnnotationDbi, BiocGenerics, Biostrings, BSgenome, EDASeq, GenomicAlignments, GenomicFeatures, GenomeInfoDb, IRanges, methods, motifStack, rtracklayer, Rsamtools, RUVSeq, Rsubread, S4Vectors, XVector, ggplot2, ggfittext, scales, ggrepel, utils, cluster, stats, graphics, grid, txdbmaker, ggExtra

Suggests RUnit, BiocStyle, knitr, BSgenome.Drerio.UCSC.danRer10, edgeR, DESeq2, limma, ashR, testthat, rmarkdown, vsn, Biobase

git_url <https://git.bioconductor.org/packages/ribosomeProfilingQC>

git_branch devel

git_last_commit 153ace4

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-24

Author Jianhong Ou [aut, cre] (ORCID: <<https://orcid.org/0000-0002-8652-2488>>),
 Mariah Hoye [aut]

Maintainer Jianhong Ou <jianhong.ou@duke.edu>

Contents

| | |
|-----------------------------------|----|
| assignReadingFrame | 3 |
| codonBias | 4 |
| codonUsage | 5 |
| countReads | 6 |
| coverageDepth | 7 |
| coverageRates | 8 |
| cvgd-class | 9 |
| estimatePsite | 10 |
| filterCDS | 11 |
| FLOSS | 12 |
| frameCounts | 13 |
| getFPKM | 14 |
| getORFscore | 15 |
| getPsiteCoordinates | 16 |
| ggBar | 17 |
| metaPlot | 17 |
| normalizeTEbyLoess | 18 |
| normBy | 19 |
| normByRUVs | 20 |
| PAmotif | 21 |
| plotDistance2Codon | 21 |
| plotFrameDensity | 22 |
| plotSpliceEvent | 23 |
| plotTE | 24 |
| plotTranscript | 25 |
| prepareCDS | 25 |
| readsDistribution | 26 |
| readsEndPlot | 27 |
| readsLenToKeep | 29 |
| ribosomeReleaseScore | 29 |
| shiftReadsByFrame | 30 |
| simulateRPF | 31 |
| spliceEvent | 33 |
| strandPlot | 33 |
| summaryReadsLength | 35 |
| translationalEfficiency | 35 |

Index

37

assignReadingFrame *Assign reading frame*

Description

Set reading frame for each reads in CDS region to frame0, frame1 and frame2.

Usage

```
assignReadingFrame(reads, CDS, txdb, ignore.seqlevelsStyle = FALSE)
```

Arguments

| | |
|-----------------------|--|
| reads | Output of getPsiteCoordinates |
| CDS | Output of prepareCDS |
| txdb | A TxDb object. If it is set, assign reading frame for all reads. Default missing, only assign rading frame for reads in CDS. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

An GRanges object of reads with reading frame information.

Examples

```
library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
pc <- getPsiteCoordinates(bamfile, bestpsite=13)
pc.sub <- pc[pc$qwidth %in% c(29, 30)]
#library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
#txdb <- makeTxDbFromGFF(system.file("extdata",
#                                   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
#                                   package="ribosomeProfilingQC"),
#                       organism = "Danio rerio",
#                       chrominfo = seqinfo(Drerio)["chr1"],
#                       taxonomyId = 7955)
#CDS <- prepareCDS(txdb)
CDS <- readRDS(system.file("extdata", "CDS.rds",
                           package="ribosomeProfilingQC"))
pc.sub <- assignReadingFrame(pc.sub, CDS)
```

codonBias

Codon usage bias

Description

Calculate the codon usage for the reads in the identified CDSs. And then compared to the reference codon usage.

Usage

```
codonBias(
  RPFs,
  gtf,
  genome,
  bestpsite = 13,
  readsLen = c(28, 29),
  anchor = "5end",
  ignore.seqlevelsStyle = FALSE,
  summary = TRUE,
  removeDuplicates = TRUE,
  ...
)
```

Arguments

| | |
|-----------------------|--|
| RPFs | Bam file names of RPFs. |
| gtf | GTF file name for annotation or a TxDb object. |
| genome | A BSgenome object. |
| bestpsite | P site postion. |
| readsLen | Reads length to keep. |
| anchor | 5end or 3end. Default is 5end. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |
| summary | Return the summary of codon usage bias or full list. |
| removeDuplicates | Remove the PCR duplicates or not. Default TRUE. |
| ... | Parameters pass to makeTxDbFromGFF |

Value

A list of data frame of codon count table if summary is TRUE. list 'reads' means the counts by raw reads. list 'reference' means the counts by sequence extracted from reference by the coordinates of mapped reads. Otherwise, return the counts (reads/reference) table for each reads.

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
library(BSgenome.Drerio.UCSC.danRer10)
cb <- codonBias(RPFs[c(1,2)], gtf=gtf, genome=Drerio)
```

| | |
|------------|----------------------------------|
| codonUsage | <i>Start or Stop codon usage</i> |
|------------|----------------------------------|

Description

Calculate the start or stop codon usage for the identified CDSs.

Usage

```
codonUsage(reads, start = TRUE, genome)
```

Arguments

| | |
|--------|--|
| reads | Output of assignReadingFrame . |
| start | Calculate for start codon or stop codon. |
| genome | A BSgenome object. |

Value

Table of codon usage.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                          package="ribosomeProfilingQC"))
library(BSgenome.Drerio.UCSC.danRer10)
codonUsage(pcs, genome=Drerio)
codonUsage(pcs, start=FALSE, genome=Drerio)
```

countReads

*Extract counts for RPFs and RNAs***Description**

Calculate the reads counts for gene level or transcript level.

Usage

```
countReads(
  RPFs,
  RNAs,
  gtf,
  level = c("tx", "gene"),
  bestpsite = 13,
  readsLen = c(28, 29),
  anchor = "5end",
  ignore.seqlevelsStyle = FALSE,
  ...
)
```

Arguments

| | |
|-----------------------|---|
| RPFs | Bam file names of RPFs. |
| RNAs | Bam file names of RNAseq. |
| gtf | GTF file name for annotation. |
| level | Transcript or gene level. |
| bestpsite | numeric(1). P site postion. |
| readsLen | numeric(1). reads length to keep. |
| anchor | 5end or 3end. Default is 5end. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |
| ... | Parameters pass to featureCounts except isGTFAnnotationFile, GTF.attrType, and annot.ext. |

Value

A list with reads counts.

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?.[12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
RNAs <- dir(path, "mRNA.*?.[12].bam$", full.names = TRUE)
cnts <- countReads(RPFs[1], gtf=gtf, level="gene", readsLen=29)
#cnts <- countReads(RPFs[1], RNAs[1], gtf=gtf, level="gene", readsLen=29)
```

| | |
|---------------|--|
| coverageDepth | <i>Extract coverage depth for gene level or transcript level</i> |
|---------------|--|

Description

Calculate the coverage depth for gene level or transcript level. Coverage for RPFs will be the best P site coverage. Coverage for RNAs will be the coverage for 5' end of reads.

Usage

```
coverageDepth(
  RPFs,
  RNAs,
  gtf,
  level = c("tx", "gene"),
  bestpsite = 13,
  readsLen = c(28, 29),
  anchor = "5end",
  region = "cds",
  ext = 5000,
  ignore.seqlevelsStyle = FALSE,
  ...
)
```

Arguments

| | |
|-----------------------|--|
| RPFs | Bam file names of RPFs. |
| RNAs | Bam file names of RNAseq. |
| gtf | GTF file name for annotation or a TxDb object. |
| level | Transcript or gene level. |
| bestpsite | P site position. |
| readsLen | Reads length to keep. |
| anchor | 5end or 3end. Default is 5end. |
| region | Annotation region. It could be "cds", "utr5", "utr3", "exon", "transcripts", "feature with extension". |
| ext | Extension region for "feature with extension". |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |
| ... | Parameters pass to makeTxDbFromGFF |

Value

A cvgd object with coverage depth.

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
cvgs <- coverageDepth(RPFs[1], gtf=gtf, level="gene")
```

| | |
|---------------|--------------------------------|
| coverageRates | <i>Calculate coverage rate</i> |
|---------------|--------------------------------|

Description

Coverage is a measure as percentage of position with reads along the CDS. Coverage rate calculate coverage rate for RPFs and mRNAs in gene level. Coverage will be calculated based on best P sites for RPFs and 5' end for RNA-seq.

Usage

```
coverageRates(cvgs, RPFsampleOrder, mRNAsampleOrder)
```

Arguments

cvgs Output of [coverageDepth](#)

RPFsampleOrder, mRNAsampleOrder
Sample order of RPFs and mRNAs. The parameters are used to make sure that the order of RPFs and mRNAs in cvgs is corresponding samples.

Value

A list with coverage rate.

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
cvgs <- coverageDepth(RPFs[1], gtf=gtf, level="gene")
cr <- coverageRates(cvgs)
```

| | |
|------------|--------------|
| cvgd-class | Class "cvgd" |
|------------|--------------|

Description

An object of class "cvgd" represents output of coverageDepth.

Usage

```
cvgd(...)

## S4 method for signature 'cvgd'
x$name

## S4 replacement method for signature 'cvgd'
x$name <- value

## S4 method for signature 'cvgd,ANY,ANY'
x[[i, j, ..., exact = TRUE]]

## S4 replacement method for signature 'cvgd,ANY,ANY,ANY'
x[[i, j, ...]] <- value

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

Arguments

| | |
|--------|--|
| ... | Each argument in ... becomes an slot in the new "cvgd"-class. |
| x | cvgd object. |
| name | A literal character string or a name (possibly backtick quoted). |
| value | value to replace. |
| i, j | indexes specifying elements to extract or replace. |
| exact | see Extract |
| object | cvgd object. |

Value

A cvgd object.

Slots

coverage "list", list of [CompressedRleList](#), specify the coverage of features of each sample.
 granges [CompressedGRangesList](#), specify the features.

Examples

```
cvgd()
```

| | |
|---------------|---------------------------------|
| estimatePsite | <i>Estimate P site position</i> |
|---------------|---------------------------------|

Description

Estimate P site position from a subset reads.

Usage

```
estimatePsite(  
  bamfile,  
  CDS,  
  genome,  
  anchor = "5end",  
  readLen = c(25:30),  
  ignore.seqlevelsStyle = FALSE  
)
```

Arguments

| | |
|-----------------------|--|
| bamfile | A BamFile object. |
| CDS | Output of prepareCDS |
| genome | A BSgenome object. |
| anchor | 5end or 3end. Default is 5end. |
| readLen | The reads length used to estimate. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

A best P site position.

References

1: Bazzini AA, Johnstone TG, Christiano R, Mackowiak SD, Obermayer B, Fleming ES, Vejnar CE, Lee MT, Rajewsky N, Walther TC, Giraldez AJ. Identification of small ORFs in vertebrates using ribosome footprinting and evolutionary conservation. *EMBO J.* 2014 May 2;33(9):981-93. doi: 10.1002/embj.201488411. Epub 2014 Apr 4. PubMed PMID: 24705786; PubMed Central PMCID: PMC4193932.

Examples

```
library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
#library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
#txdb <- makeTxDbFromGFF(system.file("extdata",
#   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
#   package="ribosomeProfilingQC"),
#   organism = "Danio rerio",
#   chrominfo = seqinfo(Drerio)["chr1"],
#   taxonomyId = 7955)
#CDS <- prepareCDS(txdb)
CDS <- readRDS(system.file("extdata", "CDS.rds",
                           package="ribosomeProfilingQC"))
estimatePsite(bamfile, CDS, Drerio)
```

filterCDS

Filter CDS by size

Description

Filter CDS by CDS size.

Usage

```
filterCDS(CDS, sizeCutoff = 100L)
```

Arguments

| | |
|------------|---|
| CDS | Output of preparedCDS |
| sizeCutoff | numeric(1). Cutoff size for CDS. If the size of CDS is less than the cutoff, it will be filtered out. |

Value

A GRanges object with filtered CDS.

Examples

```
#library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
#txdb <- makeTxDbFromGFF(system.file("extdata",
#   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
#   package="ribosomeProfilingQC"),
#   organism = "Danio rerio",
```

```

#       chrominfo = seqinfo(Drerio)["chr1"],
#       taxonomyId = 7955)
#CDS <- prepareCDS(txdb)
CDS <- readRDS(system.file("extdata", "CDS.rds",
                           package="ribosomeProfilingQC"))
filterCDS(CDS)

```

FLOSS

Fragment Length Organization Similarity Score (FLOSS)

Description

The FLOSS will be calculated from a histogram of read lengths for footprints on a transcript or reading frame.

Usage

```

FLOSS(
  reads,
  ref,
  CDS,
  readLengths = c(26:34),
  level = c("tx", "gene"),
  draw = FALSE,
  ignore.seqlevelsStyle = FALSE
)

```

Arguments

| | |
|-----------------------|---|
| reads | Output of getPsiteCoordinates |
| ref | Reference id list. If level is set to tx, the id should be transcript names. If level is set to gene, the id should be gene id. |
| CDS | Output of prepareCDS |
| readLengths | Read length used for calculation |
| level | Transcript or gene level |
| draw | Plot FLOSS vs total reads or not. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

A data frame with colnames as id, FLOSS, totalReads, wilcox.test.pval, cook's distance.

References

1: Ingolia NT, Brar GA, Stern-Ginossar N, Harris MS, Talhouarne GJ, Jackson SE, Wills MR, Weissman JS. Ribosome profiling reveals pervasive translation outside of annotated protein-coding genes. *Cell Rep.* 2014 Sep 11;8(5):1365-79. doi: 10.1016/j.celrep.2014.07.045. Epub 2014 Aug 21. PubMed PMID: 25159147; PubMed Central PMCID: PMC4216110.

Examples

```
library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
pc <- getPsiteCoordinates(bamfile, bestpsite=13)
#library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
#txdb <- makeTxDbFromGFF(system.file("extdata",
#   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
#   package="ribosomeProfilingQC"),
#   organism = "Danio rerio",
#   chrominfo = seqinfo(Drerio)["chr1"],
#   taxonomyId = 7955)
#CDS <- prepareCDS(txdb)
CDS <- readRDS(system.file("extdata", "CDS.rds",
                           package="ribosomeProfilingQC"))

set.seed(123)
ref <- sample(unique(CDS$gene_id), 100)
fl <- FLOSS(pc, ref, CDS, level="gene")
```

frameCounts

Extract counts for gene level or transcript level

Description

Calculate the reads counts or coverage rate for gene level or transcript level. Coverage is determined by measuring the proportion of in-frame CDS positions with ≥ 1 reads.

Usage

```
frameCounts(
  reads,
  level = c("tx", "gene"),
  frame0only = TRUE,
  coverageRate = FALSE
)
```

Arguments

| | |
|--------------|--|
| reads | Output of assignReadingFrame . |
| level | Transcript or gene level |
| frame0only | Only count for reading frame 0 or not |
| coverageRate | Calculate for coverage or not |

Value

A numeric vector with reads counts.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                          package="ribosomeProfilingQC"))
cnts <- frameCounts(pcs)
cnts.gene <- frameCounts(pcs, level="gene")
cvg <- frameCounts(pcs, coverageRate=TRUE)
```

getFPKM

Get FPKM values for counts

Description

Calculate Fragments Per Kilobase of transcript per Million mapped reads (FPKM) for counts.

Usage

```
getFPKM(counts, gtf, level = c("gene", "tx"))
```

Arguments

| | |
|--------|--|
| counts | Output of countReads or normByRUVs |
| gtf | GTF file name for annotation. |
| level | Transcript or gene level. |

Value

A list with FPKMs

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
#RPFs <- dir(path, "RPF.*?.[12].bam$", full.names=TRUE)
#RNAs <- dir(path, "mRNA.*?.[12].bam$", full.names=TRUE)
#gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
#cnts <- countReads(RPFs, RNAs, gtf, level="gene")
cnts <- readRDS(file.path(path, "cnts.rds"))
fpkm <- getFPKM(cnts)
```

| | |
|-------------|---------------------------|
| getORFscore | <i>Calculate ORFscore</i> |
|-------------|---------------------------|

Description

To calculate the ORFscore, reads were counted at each position within the ORF.

$$ORFscore = \log_2\left(\left(\sum_{n=1}^3 \frac{(F_n - \bar{F})^2}{\bar{F}}\right) + 1\right)$$

where F_n is the number of reads in reading frame n , \bar{F} is the total number of reads across all three frames divided by 3. If F_1 is smaller than F_2 or F_3 , $ORFscore = -1XORFscore$.

Usage

```
getORFscore(reads)
```

Arguments

reads Output of [getPsiteCoordinates](#)

Value

A numeric vector with ORFscore.

References

1: Bazzini AA, Johnstone TG, Christiano R, Mackowiak SD, Obermayer B, Fleming ES, Vejnar CE, Lee MT, Rajewsky N, Walther TC, Giraldez AJ. Identification of small ORFs in vertebrates using ribosome footprinting and evolutionary conservation. EMBO J. 2014 May 2;33(9):981-93. doi: 10.1002/emj.201488411. Epub 2014 Apr 4. PubMed PMID: 24705786; PubMed Central PMCID: PMC4193932.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                           package="ribosomeProfilingQC"))
ORFscore <- getORFscore(pcs)
```

getPsiteCoordinates *Get P site coordinates*

Description

Extract P site coordinates from a bam file to a GRanges object.

Usage

```
getPsiteCoordinates(  
  bamfile,  
  bestpsite,  
  anchor = "5end",  
  param = ScanBamParam(what = c("qwidth"), tag = character(0), flag =  
    scanBamFlag(isSecondaryAlignment = FALSE, isUnmappedQuery = FALSE,  
    isNotPassingQualityControls = FALSE, isSupplementaryAlignment = FALSE))  
)
```

Arguments

| | |
|-----------|--|
| bamfile | A BamFile object. |
| bestpsite | P site position. See estimatePsite |
| anchor | 5end or 3end. Default is 5end. |
| param | A ScanBamParam object. Please note the 'qwidth' is required. |

Value

A GRanges object with qwidth metadata which indicates the width of reads.

Examples

```
library(Rsamtools)  
bamfilename <- system.file("extdata", "RPF.WT.1.bam",  
                           package="ribosomeProfilingQC")  
yieldSize <- 10000000  
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)  
pc <- getPsiteCoordinates(bamfile, bestpsite=13)
```

| | |
|-------|---------------------------|
| ggBar | <i>barplot by ggplot2</i> |
|-------|---------------------------|

Description

barplot with number in top.

Usage

```
ggBar(height, fill = "gray80", draw = TRUE, xlab, ylab, postfix)
```

Arguments

| | |
|------------------|---------------------------------------|
| height | data for plot |
| fill, xlab, ylab | parameters pass to ggplot. |
| draw | plot or not |
| postfix | Postfix of text labled in top of bar. |

Value

ggplot object.

Examples

```
ribosomeProfilingQC:::ggBar(sample.int(100, 3))
```

| | |
|----------|-------------------------------|
| metaPlot | <i>Metagene analysis plot</i> |
|----------|-------------------------------|

Description

Plot the average coverage of UTR5, CDS and UTR3.

Usage

```
metaPlot(  
  UTR5coverage,  
  CDSCoverage,  
  UTR3coverage,  
  sample,  
  xaxis = c("RPFs", "mRNA"),  
  bins = c(UTR5 = 100, CDS = 500, UTR3 = 100),  
  ...  
)
```

Arguments

UTR5coverage, CDScoverage, UTR3coverage
 Coverages of UTR5, CDS, and UTR3 region. Output of [coverageDepth](#)

sample character(1). Sample name to plot.

xaxis What to plot for x-axis.

bins Bins for UTR5, CDS and UTR3.

... Parameter pass to plot.

Value

A list contain the data for plot.

Examples

```
## Not run:
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)
RNAs <- dir(path, "mRNA.*?\\. [12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
cvgs <- coverageDepth(RPFs[1], RNAs[1], gtf)
cvgs.utr3 <- coverageDepth(RPFs[1], RNAs[1], gtf, region="utr3")
cvgs.utr5 <- coverageDepth(RPFs[1], RNAs[1], gtf, region="utr5")
metaPlot(cvgs.utr5, cvgs, cvgs.utr3, sample=1)

## End(Not run)
```

normalizeTEbyLoess *Normalize the TE by Loess*

Description

Fitting the translational efficiency values with the mRNA value by [loess](#).

Usage

```
normalizeTEbyLoess(
  TE,
  log2 = TRUE,
  pseudocount = 0.001,
  span = 2/3,
  family.loess = "symmetric"
)
```


Examples

```

path <- system.file("extdata", package="ribosomeProfilingQC")
cnts <- readRDS(file.path(path, "cnts.rds"))
norm <- normBy(cnts, method = 'edgeR')
norm2 <- normBy(cnts, method = 'DESeq2')
norm3 <- normBy(cnts, 'vsn')

```

normByRUVs

*Normalization by RUVSeq***Description**

Normalization by RUVSeq:RUVs methods

Usage

```
normByRUVs(counts, RPFgroup, mRNAgroup = RPFgroup, k = 1)
```

Arguments

counts Output of [countReads](#)

RPFgroup, mRNAgroup
 Groups for RPF and mRNA files

k The number of factor of unwanted variation to be estimated from the data. See [RUVs](#)

Value

Normalized counts list

Examples

```

## Not run: ##waiting for EDASeq fix the issue.
path <- system.file("extdata", package="ribosomeProfilingQC")
#RPFs <- dir(path, "RPF.*?.[12].bam$", full.names=TRUE)
#RNAs <- dir(path, "mRNA.*?.[12].bam$", full.names=TRUE)
#gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
#cnts <- countReads(RPFs, RNAs, gtf, level="gene")
cnts <- readRDS(file.path(path, "cnts.rds"))
gp <- c("KD1", "KD1", "WT", "WT")
norm <- normByRUVs(cnts, gp, gp)

## End(Not run)

```

PAmotif *Metaplot of P site distribution*

Description

Metaplot of P site distribution in all the CDS aligned by the start codon or stop codon.

Usage

```
PAmotif(reads, genome, plot = TRUE, ignore.seqlevelsStyle = FALSE)
```

Arguments

| | |
|-----------------------|---|
| reads | Output of assignReadingFrame or shiftReadsByFrame . |
| genome | A BSgenome object. |
| plot | Plot the motif or not. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

A [pcm](#) object

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                          package="ribosomeProfilingQC"))
library(BSgenome.Drerio.UCSC.danRer10)
#PAmotif(pcs, Drerio)
```

plotDistance2Codon *Metaplot of P site distribution*

Description

Metaplot of P site distribution in all the CDS aligned by the start codon or stop codon.

Usage

```
plotDistance2Codon(
  reads,
  start = TRUE,
  anchor = 50,
  col = c(Frame_0 = "#009E73", Frame_1 = "#D55E00", Frame_2 = "#0072B2")
)
```

Arguments

| | |
|--------|---|
| reads | Output of assignReadingFrame . |
| start | Plot for start codon or stop codon. |
| anchor | The maximal xlim or (min, max) position for plot. |
| col | Colors for different reading frame. |

Value

Invisible height of the barplot.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                          package="ribosomeProfilingQC"))
plotDistance2Codon(pcs)
#plotDistance2Codon(pcs, start=FALSE)
#plotDistance2Codon(pcs, anchor=c(-10, 20))
```

plotFrameDensity *Plot density for each reading frame*

Description

Plot density for each reading frame.

Usage

```
plotFrameDensity(
  reads,
  density = TRUE,
  col = c(Frame_0 = "#009E73", Frame_1 = "#D55E00", Frame_2 = "#0072B2")
)
```

Arguments

| | |
|---------|--|
| reads | Output of assignReadingFrame |
| density | Plot density or counts |
| col | Colors for reading frames |

Value

A ggplot object.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",
                          package="ribosomeProfilingQC"))
plotFrameDensity(pcs)
```

| | |
|-----------------|--------------------------|
| plotSpliceEvent | <i>Plot splice event</i> |
|-----------------|--------------------------|

Description

Plot the splice event

Usage

```
plotSpliceEvent(  
  se,  
  tx_name,  
  coverage,  
  group1,  
  group2,  
  cutoffFDR = 0.05,  
  resetIntronWidth = TRUE  
)
```

Arguments

| | |
|------------------|--|
| se | Output of spliceEvent |
| tx_name | Transcript name. |
| coverage | Coverages of feature region with extensions. Output of coverageDepth |
| group1, group2 | The sample names of group 1 and group 2 |
| cutoffFDR | Cutoff of FDR |
| resetIntronWidth | logical(1). If set to true, reset the region with no read to minimal width. |

Value

A ggplot object.

Examples

```
## Not run:  
path <- system.file("extdata", package="ribosomeProfilingQC")  
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)  
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")  
coverage <- coverageDepth(RPFs, gtf=gtf, level="gene",  
  region="feature with extension")  
group1 <- c("RPF.KD1.1", "RPF.KD1.2")  
group2 <- c("RPF.WT.1", "RPF.WT.2")  
se <- spliceEvent(coverage, group1, group2)  
plotSpliceEvent(se, se$feature[1], coverage, group1, group2)  
  
## End(Not run)
```

plotTE *Plot translational efficiency*

Description

Scatterplot of RNA/RPFs level compared to the translational efficiency.

Usage

```
plotTE(
  TE,
  sample,
  xaxis = c("mRNA", "RPFs"),
  removeZero = TRUE,
  log2 = TRUE,
  theme = theme_classic(),
  type = "histogram",
  margins = "y",
  ...
)
```

Arguments

| | |
|--------------------|---|
| TE | Output of translationalEfficiency |
| sample | Sample names to plot. |
| xaxis | What to plot for x-axis. |
| removeZero | Remove the 0 values from plots. |
| log2 | Do log2 transform for TE or not. |
| theme | Theme for ggplot2. |
| type, margins, ... | Parameters pass to ggMarginal |

Value

A ggExtraPlot object.

Examples

```
path <- system.file("extdata", package="ribosomeProfilingQC")
#RPFs <- dir(path, "RPF.*?\.[12].bam$", full.names=TRUE)
#RNAs <- dir(path, "mRNA.*?\.[12].bam$", full.names=TRUE)
#gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
#cnts <- countReads(RPFs, RNAs, gtf, level="gene")
cnts <- readRDS(file.path(path, "cnts.rds"))
fpkm <- getFPKM(cnts)
te <- translationalEfficiency(fpkm)
plotTE(te, 1)
```

| | |
|----------------|--|
| plotTranscript | <i>Plot reads P site abundance for a specific transcript</i> |
|----------------|--|

Description

Plot the bundances of P site on a transcript.

Usage

```
plotTranscript(  
  reads,  
  tx_name,  
  col = c(Frame_0 = "#009E73", Frame_1 = "#D55E00", Frame_2 = "#0072B2")  
)
```

Arguments

| | |
|---------|--|
| reads | Output of assignReadingFrame |
| tx_name | Transcript names. |
| col | Colors for reading frames |

Value

Invisible heights of the barplot.

Examples

```
pcs <- readRDS(system.file("extdata", "samplePc.rds",  
  package="ribosomeProfilingQC"))  
  
plotTranscript(pcs, c("ENSDART00000152562", "ENSDART00000054987"))
```

| | |
|------------|--------------------|
| prepareCDS | <i>Prepare CDS</i> |
|------------|--------------------|

Description

Prepare CDS library from a TxDb object.

Usage

```
prepareCDS(txdb, withUTR = FALSE)
```

Arguments

| | |
|---------|-----------------------------------|
| txdb | A TxDb object. |
| withUTR | Including UTR information or not. |

Value

A GRanges object with metadata which include: tx_id: transcript id; tx_name: transcript name; gene_id: gene id; isFirstExonInCDS: is first exon in CDS or not; idFirstExonInCDS: the id for the first exon; isLastExonInCDS: is last exon in CDS or not; wid.cumsu: cumulative sums of number of bases in CDS; internalPos: offset position from 1 base;

Examples

```
library(GenomicFeatures)
txdb_file <- system.file("extdata", "Biomart_Ensembl_sample.sqlite",
                        package="GenomicFeatures")
txdb <- loadDb(txdb_file)
CDS <- prepareCDS(txdb)
```

readsDistribution *Plot reads distribution in genomic elements*

Description

Plot the percentage of reads in CDS, 5'UTR, 3'UTR, introns, and other elements.

Usage

```
readsDistribution(
  reads,
  txdb,
  upstreamRegion = 3000,
  downstreamRegion = 3000,
  plot = TRUE,
  precedence = NULL,
  ignore.seqlevelsStyle = FALSE,
  ...
)
```

Arguments

| | |
|----------------------------------|---|
| reads | Output of getPsiteCoordinates |
| txdb | A TxDb object |
| upstreamRegion, downstreamRegion | The range for promoter region and downstream region. |
| plot | Plot the distribution or not |
| precedence | If no precedence specified, double count will be enabled, which means that if the reads overlap with both CDS and 5'UTR, both CDS and 5'UTR will be incremented. If a precedence order is specified, for example, if promoter is specified before 5'UTR, then only promoter will be incremented for the same example. The values could be any combinations of "CDS", "UTR5", "UTR3", "OtherExon", "Intron", "upstream", "downstream" and "InterGenic", Default=NULL |

```

ignore.seqlevelsStyle
                        Ignore the sequence name style detection or not.
...
                        Not use.

```

Value

The reads with distribution assignment

Examples

```

library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
pc <- getPsiteCoordinates(bamfile, bestpsite=11)
pc.sub <- pc[pc$qwidth %in% c(29, 30)]
library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
txdb <- makeTxDbFromGFF(system.file("extdata",
                                   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
                                   package="ribosomeProfilingQC"),
                       organism = "Danio rerio",
                       chrominfo = seqinfo(Drerio)["chr1"],
                       taxonomyId = 7955)
pc.sub <- readsDistribution(pc.sub, txdb, las=2)
pc.sub <- readsDistribution(pc.sub, txdb, las=2,
                          precedence=c(
                            "CDS", "UTR5", "UTR3", "OtherExon",
                            "Intron", "upstream", "downstream",
                            "InterGenic"
                          ))

```

readsEndPlot

Plot start/stop windows

Description

Plot the reads shifted from start/stop position of CDS.

Usage

```

readsEndPlot(
  bamfile,
  CDS,
  toStartCodon = TRUE,
  fiveEnd = TRUE,
  shift = 0,
  window = c(-29, 30),

```

```

    readLen = 25:30,
    ignore.seqlevelsStyle = FALSE
  )

```

Arguments

| | |
|-----------------------|--|
| bamfile | A BamFile object. |
| CDS | Output of prepareCDS |
| toStartCodon | What to search: start or end codon |
| fiveEnd | Search from five or three ends of the reads. |
| shift | number(1). Search from 5' end or 3' end of given number. if fiveEnd set to false, please set the shift as a negative number. |
| window | The window of CDS region to plot |
| readLen | The reads length used to plot |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

The invisible list with counts numbers and reads in GRanges.

Examples

```

library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
#library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
#txdb <- makeTxDbFromGFF(system.file("extdata",
#   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
#   package="ribosomeProfilingQC"),
#   organism = "Danio rerio",
#   chrominfo = seqinfo(Drerio)["chr1"],
#   taxonomyId = 7955)
#CDS <- prepareCDS(txdb)
CDS <- readRDS(system.file("extdata", "CDS.rds",
                           package="ribosomeProfilingQC"))

re <- readsEndPlot(bamfile, CDS, toStartCodon=TRUE)
readsEndPlot(re$reads, CDS, toStartCodon=TRUE, fiveEnd=FALSE)
#re <- readsEndPlot(bamfile, CDS, toStartCodon=FALSE)
#readsEndPlot(re$reads, CDS, toStartCodon=FALSE, fiveEnd=FALSE)
readsEndPlot(bamfile, CDS, shift=13)
#readsEndPlot(bamfile, CDS, fiveEnd=FALSE, shift=-16)

```

readsLenToKeep *Get reads length to keep by cutoff percentage*

Description

Set the percentage to filter the reads.

Usage

```
readsLenToKeep(readsLengthDensity, cutoff = 0.8)
```

Arguments

readsLengthDensity Output of [summaryReadsLength](#)
 cutoff Cutoff value.

Value

Reads length to be kept.

Examples

```
reads <- GRanges("chr1", ranges=IRanges(seq.int(100), width=1),
                 qwidth=sample(25:31, size = 100, replace = TRUE,
                               prob = c(.01, .01, .05, .1, .77, .05, .01)))
readsLenToKeep(summaryReadsLength(reads, plot=FALSE))
```

ribosomeReleaseScore *Ribosome Release Score (RRS)*

Description

RRS is calculated as the ratio of translational efficiency in the CDS with RPFs in the 3'UTR.

Usage

```
ribosomeReleaseScore(
  cdsTE,
  utr3TE,
  CDSsampleOrder,
  UTR3sampleOrder,
  pseudocount = 0,
  log2 = FALSE
)
```

Arguments

| | |
|---------------------------------|---|
| cdsTE, utr3TE | Translational efficiency of CDS and UTR3 region. Output of translationalEfficiency |
| CDSsampleOrder, UTR3sampleOrder | Sample order of cdsTE and utr3TE. The parameters are used to make sure that the order of CDS and UTR3 in TE is corresponding samples. |
| pseudocount | The number will be add to sum of reads count to avoid X/0. |
| log2 | Do log2 transform or not. |

Value

A vector of RRS.

Examples

```
## Not run:
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\.[12].bam$", full.names=TRUE)
RNAs <- dir(path, "mRNA.*?\\.[12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
cvgs <- coverageDepth(RPFs, RNAs, gtf)
cvgs.utr3 <- coverageDepth(RPFs, RNAs, gtf, region="utr3")
TE90 <- translationalEfficiency(cvgs, window = 90)
TE90.utr3 <- translationalEfficiency(cvgs.utr3, window = 90)
rrs <- ribosomeReleaseScore(TE90, TE90.utr3)

## End(Not run)
```

shiftReadsByFrame *Shift reads by reading frame*

Description

Shift reads P site position by reading frame. After shifting, all reading frame will be set as 0

Usage

```
shiftReadsByFrame(reads, txdb, ignore.seqlevelsStyle = FALSE)
```

Arguments

| | |
|-----------------------|--|
| reads | Output of getPsiteCoordinates |
| txdb | A TxDb object. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |

Value

Reads with reading frame information

Examples

```
library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
                           package="ribosomeProfilingQC")

yieldSize <- 10000000
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)
pc <- getPsiteCoordinates(bamfile, bestpsite=11)
pc.sub <- pc[pc$qwidth %in% c(29, 30)]
library(GenomicFeatures)
library(BSgenome.Drerio.UCSC.danRer10)
txdb <- makeTxDbFromGFF(system.file("extdata",
                                   "Danio_rerio.GRCz10.91.chr1.gtf.gz",
                                   package="ribosomeProfilingQC"),
                       organism = "Danio rerio",
                       chrominfo = seqinfo(Drerio)["chr1"],
                       taxonomyId = 7955)
pc.sub <- shiftReadsByFrame(pc.sub, txdb)
```

simulateRPF

Simulation function

Description

Simulate the RPFs reads in CDS, 5'UTR and 3'UTR

Usage

```
simulateRPF(
  txdb,
  outPath,
  genome,
  samples = 6,
  group1 = c(1, 2, 3),
  group2 = c(4, 5, 6),
  readsPerSample = 1e+06,
  readsLen = 28,
  psite = 13,
  frame0 = 0.9,
  frame1 = 0.05,
  frame2 = 0.05,
  DRegions = GRanges(),
  size = 1,
  sd = 0.02,
```

```

    minDElevel = log2(2),
    includeReadsSeq = FALSE
  )

```

Arguments

| | |
|------------------------|---|
| txdb | A TxDb object |
| outPath | Output folder for the bam files |
| genome | A BSgenome object |
| samples | Total samples to simulate. |
| group1, group2 | Numeric to index the sample groups. |
| readsPerSample | Total reads number per sample. |
| readsLen | Reads length, default 100bp. |
| psite | P-site position. default 13. |
| frame0, frame1, frame2 | Percentage of reads distribution in frame0, frame1 and frame2 |
| DEregions | The regions with differential reads in exon, utr5 and utr3. |
| size | Dispersion parameter. Must be strictly positive. |
| sd | Standard deviations. |
| minDElevel | Minimal differential level. default: log2(2). |
| includeReadsSeq | logical(1). Include reads sequence or not. |

Value

An invisible list of GAlignments.

Examples

```

library(GenomicFeatures)
txdb_file <- system.file("extdata", "Biomart_Ensembl_sample.sqlite",
                        package="GenomicFeatures")
txdb <- loadDb(txdb_file)
simulateRPF(txdb, samples=1, readsPerSample = 1e3)
## Not run:
cds <- prepareCDS(txdb, withUTR = TRUE)
cds <- cds[width(cds)>200]
DEregions <- cds[sample(seq_along(cds), 10)]
simulateRPF(txdb, samples=6, readsPerSample = 1e5, DEregions=DEregions)

## End(Not run)

```

| | |
|-------------|----------------------------|
| spliceEvent | <i>Get splicing events</i> |
|-------------|----------------------------|

Description

Get differential usage of alternative Translation Initiation Sites, alternative Polyadenylation Sites or alternative splicing sites

Usage

```
spliceEvent(coverage, group1, group2)
```

Arguments

coverage Coverages of feature region with extensions. Output of [coverageDepth](#)
group1, group2 The sample names of group 1 and group 2

Value

A GRanges object of splice events.

Examples

```
## Not run:  
path <- system.file("extdata", package="ribosomeProfilingQC")  
RPFs <- dir(path, "RPF.*?\\. [12].bam$", full.names=TRUE)  
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")  
coverage <- coverageDepth(RPFs, gtf=gtf,  
                          level="gene", region="feature with extension")  
group1 <- c("RPF.KD1.1", "RPF.KD1.2")  
group2 <- c("RPF.WT.1", "RPF.WT.2")  
se <- spliceEvent(coverage, group1, group2)  
  
## End(Not run)
```

| | |
|------------|---|
| strandPlot | <i>Plot the distribution of reads in sense and antisense strand</i> |
|------------|---|

Description

Plot the distribution of reads in sense and antisense strand to check the mapping is correct.

Usage

```
strandPlot(  
  reads,  
  CDS,  
  col = c("#009E73", "#D55E00"),  
  ignore.seqlevelsStyle = FALSE,  
  ...  
)
```

Arguments

| | |
|-----------------------|--|
| reads | Output of getPsiteCoordinates |
| CDS | Output of prepareCDS |
| col | Color for sense and antisense strand. |
| ignore.seqlevelsStyle | Ignore the sequence name style detection or not. |
| ... | Parameter passed to barplot |

Value

A ggplot object.

Examples

```
library(Rsamtools)  
bamfilename <- system.file("extdata", "RPF.WT.1.bam",  
                           package="ribosomeProfilingQC")  
  
yieldSize <- 10000000  
bamfile <- BamFile(bamfilename, yieldSize = yieldSize)  
pc <- getPsiteCoordinates(bamfile, bestpsite=11)  
pc.sub <- pc[pc$qwidth %in% c(29, 30)]  
library(GenomicFeatures)  
library(BSgenome.Drerio.UCSC.danRer10)  
txdb <- makeTxDbFromGFF(system.file("extdata",  
                                   "Danio_rerio.GRCz10.91.chr1.gtf.gz",  
                                   package="ribosomeProfilingQC"),  
                       organism = "Danio rerio",  
                       chrominfo = seqinfo(Drerio)["chr1"],  
                       taxonomyId = 7955)  
CDS <- prepareCDS(txdb)  
strandPlot(pc.sub, CDS)
```

summaryReadsLength *Summary the reads lengths*

Description

Plot the reads length distribution

Usage

```
summaryReadsLength(reads, widthRange = c(20:35), plot = TRUE, ...)
```

Arguments

| | |
|------------|-------------------------------|
| reads | Output of getPsiteCoordinates |
| widthRange | The reads range to be plot |
| plot | Do plot or not |
| ... | Not use. |

Value

The reads length distribution

Examples

```
reads <- GRanges("chr1", ranges=IRanges(seq.int(100), width=1),
                qwidth=sample(25:31, size = 100, replace = TRUE,
                              prob = c(.01, .01, .05, .1, .77, .05, .01)))
summaryReadsLength(reads)
```

translationalEfficiency
Translational Efficiency

Description

Calculate Translational Efficiency (TE). TE is defined as the ratios of the absolute level of ribosome occupancy divided by RNA levels for transcripts.

Usage

```
translationalEfficiency(
  x,
  window,
  RPFsampleOrder,
  mRNAsampleOrder,
  pseudocount = 1,
  log2 = FALSE,
  normByLibSize = FALSE,
  shrink = FALSE,
  ...
)
```

Arguments

| | |
|---------------------------------|--|
| x | Output of getFPKM or normByRUVs . if window is set, it must be output of coverageDepth . |
| window | numeric(1). window size for maximal counts. |
| RPFsampleOrder, mRNAsampleOrder | Sample order of RPFs and mRNAs. The parameters are used to make sure that the order of RPFs and mRNAs in <code>cvgs</code> is corresponding samples. |
| pseudocount | The number will be add to sum of reads count to avoid X/0. |
| log2 | Do log2 transform or not. |
| normByLibSize | Normalization by library size or not. If window size is provided and <code>normByLibSize</code> is set to TRUE, the coverage will be normalized by library size. |
| shrink | Shrink the TE or not. |
| ... | Parameters will be passed to <code>ash</code> function from <code>ashr</code> . |

Value

A list with RPFs, mRNA levels and TE as a matrix with translational efficiency

Examples

```
## Not run:
path <- system.file("extdata", package="ribosomeProfilingQC")
RPFs <- dir(path, "RPF.*?\\.[12].bam$", full.names=TRUE)
RNAs <- dir(path, "mRNA.*?\\.[12].bam$", full.names=TRUE)
gtf <- file.path(path, "Danio_rerio.GRCz10.91.chr1.gtf.gz")
cnts <- countReads(RPFs, RNAs, gtf, level="gene")
fpkm <- getFPKM(cnts)
te <- translationalEfficiency(fpkm)

## End(Not run)
```

Index

`[[`, cvgd, ANY, ANY-method (cvgd-class), 9
`[[<-`, cvgd, ANY, ANY, ANY-method (cvgd-class), 9
`$`, cvgd-method (cvgd-class), 9
`$<-`, cvgd-method (cvgd-class), 9
`assignReadingFrame`, 3, 5, 14, 21, 22, 25
`codonBias`, 4
`codonUsage`, 5
`CompressedGRangesList`, 9
`CompressedRleList`, 9
`countReads`, 6, 14, 19, 20
`coverageDepth`, 7, 8, 18, 23, 33, 36
`coverageRates`, 8
`cvgd` (cvgd-class), 9
cvgd-class, 9
`estimatePsite`, 10, 16
`Extract`, 9
`featureCounts`, 6
`filterCDS`, 11
`FLOSS`, 12
`frameCounts`, 13
`getFPKM`, 14, 19, 36
`getORFscore`, 15
`getPsiteCoordinates`, 3, 12, 15, 16, 26, 30, 34
`ggBar`, 17
`loess`, 18, 19
`makeTxDbFromGFF`, 4, 7
`metaPlot`, 17
`normalizeTEbyLoess`, 18
`normBy`, 19
`normByRUVs`, 14, 19, 20, 36
`PAmotif`, 21
`pcm`, 21
`plotDistance2Codon`, 21
`plotFrameDensity`, 22
`plotSpliceEvent`, 23
`plotTE`, 24
`plotTranscript`, 25
`prepareCDS`, 3, 10, 12, 25, 28, 34
`readsDistribution`, 26
`readsEndPlot`, 27
`readsLenToKeep`, 29
`ribosomeReleaseScore`, 29
RUVs, 20
`shiftReadsByFrame`, 21, 30
`show`, cvgd-method (cvgd-class), 9
`simulateRPF`, 31
`spliceEvent`, 23, 33
`strandPlot`, 33
`summaryReadsLength`, 29, 35
`translationalEfficiency`, 19, 24, 30, 35