Package ‘ribosomeProfilingQC’

March 7, 2024

**Type** Package

**Title** Ribosome Profiling Quality Control

**Version** 1.14.0

**Description** Ribo-Seq (also named ribosome profiling or footprinting) measures translome (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** 7.2.3

**Depends** R (>= 4.0), GenomicRanges

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

**Suggests** RUnit, BiocStyle, knitr, BSgenome.Drerio.UCSC.danRer10, edgeR, limma, testthat, rmarkdown

**git_url** https://git.bioconductor.org/packages/ribosomeProfilingQC

**git_branch** RELEASE_3_18

**git_last_commit** ae73f4a

**git_last_commit_date** 2023-10-24

**Repository** Bioconductor 3.18

**Date/Publication** 2024-03-06
R topics documented:

assignReadingFrame ........................................ 3
codonUsage ...................................................... 4
countReads ...................................................... 4
coverageDepth ................................................... 5
coverageRates ................................................... 7
cvgd-class ...................................................... 7
estimatePsite .................................................... 9
filterCDS ......................................................... 10
FLOSS ............................................................ 11
frameCounts ..................................................... 12
getFPKM ......................................................... 13
getORFscore ..................................................... 14
ggBar ............................................................ 15
metaPlot .......................................................... 16
normByRUVs ..................................................... 17
PAmotif ........................................................... 18
plotDistance2Codon ............................................ 18
plotFrameDensity .............................................. 19
plotSpliceEvent ............................................... 20
plotTE ........................................................... 21
plotTranscript .................................................. 22
prepareCDS ..................................................... 22
readsDistribution ............................................. 23
readsEndPlot ................................................... 24
readsLenToKeep ................................................ 26
ribosomeReleaseScore ....................................... 26
shiftReadsByFrame ........................................... 27
simulateRPF ..................................................... 28
spliceEvent ..................................................... 30
strandPlot ....................................................... 30
summaryReadsLength ......................................... 32
translationalEfficiency ...................................... 32

Index 34
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)
codonUsage

**Start or Stop codon usage**

**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**
```r
codons <- readRDS(system.file("extdata", "samplePc.rds", package="ribosomeProfilingQC"))
library(BSgenome.Drerio.UCSC.danRer10)
codonUsage(codons, genome=Drerio)
codonUsage(codons, 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",
...
coverageDepth

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 position.
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)

countReads

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

---

**Calculate coverage rate**

**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**

```r
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**

```r
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`. 

cvgd-class

Usage

```r
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

```r
cvgd()
```
**estimatePsite**

**Estimate P site position**

**Description**

Estimate P site position from a subset reads.

**Usage**

```r
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**


**Examples**

```r
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", 
  taxonomyId = 7955)
#txdb <- makeTxDbFromGFF(system.file("extdata", 
#  "Danio_rerio.GRCz10.91.chr1.gtf.gz", 
#  package="ribosomeProfilingQC"), 
#  organism = "Danio rerio", 
#  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", 
  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

Examples

```r
library(Rsamtools)
bamfilename <- system.file("extdata", "RPF.WT.1.bam",
    package="ribosomeProfilingQC")
yieldSize <- 1000000
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 \( \geq 1 \) reads.

Usage

```r
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
getFPKM

Value

A numeric vector with reads counts.

Examples

cnts <- getFPKM(counts, gtf, level = c("gene", "tx"))

tags <- readRDS(file.path(path, "cnts.rds"))
fkm <- getFPKM(tags)

description

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

Usage

generateFPKM(counts, gtf, level = c("gene", "tx"))

Arguments

Counts Output of countReads or normByRUVs

GTF file name for annotation.

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

Calculate ORFscore

Description

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

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

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 \( F_3 \), \( ORFscore = -1 \times ORFscore \).

Usage

getORFscore(reads)

Arguments

reads

Output of getPsitCoordinates

Value

A numeric vector with ORFscore.

References


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

```r
getPsiteCoordinates(bamfile, bestpsite, anchor = "5end")
```

Arguments

- `bamfile`: A BamFile object.
- `bestpsite`: P site position. See `estimatePsite`.
- `anchor`: 5end or 3end. Default is 5end.

Value

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

Examples

```r
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  barplot by ggplot2

Description

barplot with number in top.

Usage

```r
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 labeled in top of bar.

Value

ggplot object.

Examples

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

---

**Description**

Plot the average coverage of UTR5, CDS and UTR3.

**Usage**

```r
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

```r
## 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")
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)
```

**Description**

Normalization by RUVSeq: RUVs methods

**Usage**

```r
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**

```r
## 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**

```r
cpcs <- 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")
)
```
plotFrameDensity

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(pcs)
plotSpliceEvent

---

**plotSpliceEvent**  
*Plot splice event*

**Description**

Plot the splice event

**Usage**

```r
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**

```r
## 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,
  breaks.length = 50,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TE</td>
<td>Output of <code>translationalEfficiency</code></td>
</tr>
<tr>
<td>sample</td>
<td>character(1). Sample name to plot.</td>
</tr>
<tr>
<td>xaxis</td>
<td>What to plot for x-axis.</td>
</tr>
<tr>
<td>removeZero</td>
<td>Remove the 0 values from plots.</td>
</tr>
<tr>
<td>log2</td>
<td>Do log2 transform or not.</td>
</tr>
<tr>
<td>breaks.length</td>
<td>Length of breaks for histogram.</td>
</tr>
<tr>
<td>...</td>
<td>Parameters pass to plot.</td>
</tr>
</tbody>
</table>

Value

A invisible data.frame with x, y of points.

Examples

```r
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  
*Plot reads P site abundance for a specific transcript*

**Description**
Plot the abundances of P site on a transcript.

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

prepareCDS  
*Prepare CDS*

**Description**
Prepare CDS library from a TxDb object.

**Usage**
```r
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

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

```r
readsDistribution
```

Plot reads distribution in genomic elements

Description

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

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reads</td>
<td>Output of getPsitesCoordinates</td>
</tr>
<tr>
<td>txdb</td>
<td>A TxDb object</td>
</tr>
<tr>
<td>upstreamRegion, downstreamRegion</td>
<td>The range for promoter region and downstream region.</td>
</tr>
<tr>
<td>plot</td>
<td>Plot the distribution or not</td>
</tr>
<tr>
<td>precedence</td>
<td>If no precedence specified, double count will be enabled, which means that if</td>
</tr>
<tr>
<td></td>
<td>the reads overlap with both CDS and 5'UTR, both CDS and 5'UTR will be in-</td>
</tr>
<tr>
<td></td>
<td>cremented. If a precedence order is specified, for example, if promoter is</td>
</tr>
<tr>
<td></td>
<td>specified before 5'UTR, then only promoter will be incremented for the same</td>
</tr>
<tr>
<td></td>
<td>example. The values could be any combinations of &quot;CDS&quot;, &quot;UTR5&quot;, &quot;UTR3&quot;, &quot;Ot-</td>
</tr>
<tr>
<td></td>
<td>herExon&quot;, &quot;Intron&quot;, &quot;upstream&quot;, &quot;downstreama&quot; and &quot;InterGenic&quot;, Default=NULL</td>
</tr>
</tbody>
</table>
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"  
  ))

readEndPlot

Plot start/stop windows

Description

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

Usage

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

   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 counts numbers.

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"))
readsEndPlot(bamfile, CDS, toStartCodon=TRUE)
#readsEndPlot(bamfile, CDS, toStartCodon=TRUE, fiveEnd=FALSE)
#readsEndPlot(bamfile, CDS, toStartCodon=FALSE)
#readsEndPlot(bamfile, CDS, toStartCodon=FALSE, fiveEnd=FALSE)
readsEndPlot(bamfile, CDS, shift=13)
#readsEndPlot(bamfile, CDS, fiveEnd=FALSE, shift=-16)
**ribosomeReleaseScore**

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

### Usage

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

---

**readsLenToKeep**  
*Get reads length to keep by cutoff percentage*

### Description
Set the percentage to filter the reads.

### Usage

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

### Arguments
- **readsLengthDensity**  
  Output of `summaryReadsLength`
- **cutoff**  
  Cutoff value.

### Value
Reads length to be kept.

### Examples

```r
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))
```
shiftReadsByFrame

**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 `getPsitCoordinates`
- **txdb**: A TxDb object.
- **ignore.seqlevelsStyle**: Ignore the sequence name style detection or not.
**Value**

Reads with reading frame information

**Examples**

```r
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**

```r
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,
  DEregions = 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[seq_along(cds), 10]
simulateRPF(txdb, samples=6, readsPerSample = 1e5, DEregions=DEregions)
## End(Not run)
spliceEvent  

Get splicing events

**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**

```r
## 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  

Plot the distribution of reads in sense and antisense strand

**Description**

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

Usage

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

Arguments

reads Output of `getPsiteCoordinates`

CDS Output of `prepareCDS`

col Colour for sense and antisense strand.

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

... Parameter passed to barplot

Value

A `ggplot` object.

Examples

```r
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**

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

**Arguments**

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

**Value**

The reads length distribution

**Examples**

```r
reads <- GRanges("chr1", ranges=IRanges(seq.int(100), width=1),
width=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.
translationalEfficiency

Usage

translationalEfficiency(
  x,
  window,
  RPFsampleOrder,
  mRNAsampleOrder,
  pseudocount = 1,
  log2 = FALSE,
  normByLibSize = 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 cvgs 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 provied and normByLibSize is set to TRUE, the coverage will be normalized by library size.

Value

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

Examples

```r
## 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), 7
[[<-, cvgd, ANY, ANY, ANY-method
 (cvgd-class), 7
$, cvgd-method (cvgd-class), 7
$<-, cvgd-method (cvgd-class), 7
assignReadingFrame, 3, 4, 12, 18, 19, 22

codonUsage, 4
CompressedGRangesList, 8
CompressedRleList, 8
countReads, 4, 13, 17
coverageDepth, 5, 7, 16, 20, 30, 33
coverageRates, 7
cvgd (cvgd-class), 7
cvgd-class, 7
estimatePsite, 9, 15
Extract, 8

featureCounts, 5
filterCDS, 10
FLOSS, 11
frameCounts, 12

getFPKM, 13, 33
getORFscore, 14
getPsiteCoordinates, 3, 11, 14, 15, 23, 27, 31

ggBar, 15

makeTxDbFromGFF, 6
metaPlot, 16

normByRUVs, 13, 17, 33

PAmotif, 18
pcm, 18
plotDistance2Codon, 18
plotFrameDensity, 19
plotSpliceEvent, 20

plotTE, 21
plotTranscript, 22
prepareCDS, 3, 9, 11, 22, 25, 31

readsDistribution, 23
readsEndPlot, 24
readsLenToKeep, 26
ribosomeReleaseScore, 26
RUVs, 17

shiftReadsByFrame, 18, 27
show, cvgd-method (cvgd-class), 7
simulateRPF, 28
spliceEvent, 20, 30
strandPlot, 30
summaryReadsLength, 26, 32

translationalEfficiency, 21, 27, 32