Package ‘RiboProfiling’
March 14, 2024

Type Package
Title Ribosome Profiling Data Analysis: from BAM to Data Representation and Interpretation
Version 1.32.0
Date 2021-11-17
Author Alexandra Popa
Maintainer A. Popa <alexandra.mariela.popa@gmail.com>
Description Starting with a BAM file, this package provides the necessary functions for quality assessment, read start position recalibration, the counting of reads on CDS, 3'UTR, and 5'UTR, plotting of count data: pairs, log fold-change, codon frequency and coverage assessment, principal component analysis on codon coverage.

biocViews RiboSeq, Sequencing, Coverage, Alignment, QualityControl, Software, PrincipalComponent

Depends R (>= 3.2.2), Biostrings
Imports BiocGenerics, GenomeInfoDb, GenomicRanges, IRanges, reshape2, GenomicFeatures, grid, plyr, S4Vectors, GenomicAlignments, ggplot2, ggbio, Rsamtools, rtracklayer, data.table, sqldf
Suggests knitr, BiocStyle, TxDb.Hsapiens.UCSC.hg19.knownGene, BSgenome.Hsapiens.UCSC.hg19, testthat, SummarizedExperiment

LazyLoad yes
License GPL-3
VignetteBuilder knitr
NeedsCompilation no
RoxygenNote 5.0.1

git_url https://git.bioconductor.org/packages/RiboProfiling
git_branch RELEASE_3_18
git_last_commit 8cecb7bf
git_last_commit_date 2023-10-24
aroundPromoter

Returns the flank size around the TSS for the x % CDSs

Description

Returns the flank size around the TSS for the x % CDSs

Usage

aroundPromoter(txdb, alnGRanges, percBestExpressed, flankSize)

Arguments

txdb a TxDb object containing the annotations info to intersect with the alignment files.

alnGRanges A GRanges object containing the alignment information. In order to improve the performance of this function the GAlignments BAM object should be transformed into a GRanges object containing the cigar match size information as metadata.
The percentage of the best expressed CDSs on which to plot the coverage around the TSS. Necessary if the shiftValue parameter must be estimated. Default value 0.03 (3%).

A numeric positive integer. How many bp left and right of the TSS should the coverage be performed? Ex. flankSize <- 20

A GRanges object containing the 1 bp ranges for the selected CDSs in the TSS defined flanking region.

Examples

```r
#read the BAM into a GAlignments object using
#GenomicAlignments::readGAlignments
data(ctrlGAlignments)

aln <- ctrlGAlignments

#transform the GAlignments object into a GRanges object (faster processing)
alnGRanges <- readsToStartOrEnd(aln, what="start")

#make a txdb object containing the annotations for the specified species.
#In this case hg19.

#Please make sure that seqnames of txdb correspond to
#the seqnames of the alignment files ("chr" particle)
#if not rename the txdb seqlevels
#renameSeqlevels(txdb, sub("chr", "", seqlevels(txdb)))

#get the flanking region around the promoter of the best expressed CDSs
oneBinRanges <- aroundPromoter(txdb, alnGRanges)
```

**Description**

A list of start and end codons relative to transcript

**Usage**

```r
data(cdsPosTransc)
```

**Format**

A list

**Value**

A list
<table>
<thead>
<tr>
<th>codonDataCtrl</th>
<th>Codon frequency and coverage in ORFs on chromosome 1, for dataset ctrlGAlignments</th>
</tr>
</thead>
</table>

**Description**

A list of 2 data.frame objects: one with the number of times each codon type is found in each ORF and one with the number of reads for each codon type in each ORF.

**Usage**

```r
data(codonDataCtrl)
```

**Format**

A list of 2 lists.

**Value**

A list of 2 lists.

<table>
<thead>
<tr>
<th>codonIndexCovCtrl</th>
<th>The read coverage for each codon in ORFs on chromosome 1, for dataset ctrlGAlignments</th>
</tr>
</thead>
</table>

**Description**

A list containing the number of reads for each codon in each ORF. Codons are reported on their index in the ORF and no information is available about their type/sequence.

**Usage**

```r
data(codonIndexCovCtrl)
```

**Format**

A list of 2 columns dataframes.

**Value**

A list of 2 columns dataframes.
**codonInfo**

**Description**

Associates the read counts on codons with the codon type for each ORF.

**Usage**

```r
codonInfo(listReadsCodon, genomeSeq, orfCoord, motifSize)
```

**Arguments**

- `listReadsCodon`: a list of data.frame objects. It contains the number of reads per codon in a CDS.
- `genomeSeq`: a BSgenome object. It contains the full genome sequences for the organism.
- `orfCoord`: a GRangesList. The coordinates of the ORFs on the genome.
- `motifSize`: an integer. The number of nucleotides in each motif on which to compute coverage and usage. Either 3, 6, or 9. Default 3 nucleotides (codon). Attention! For long motifs, the function can be quite slow!!

**Value**

A list of 2 data.frame objects: one with the number of times each codon type is found in each ORF and one with the number of reads for each codon type in each ORF.

**Examples**

```r
# for each codon in each ORF get the read coverage
# parameter listReadsCodon can be returned by the riboSeqFromBam function
# it corresponds to the 2nd element in the list returned by riboSeqFromBam
data(codonIndexCovCtrl)
listReadsCodon <- codonIndexCovCtrl
# get the names of the ORFs
# grouped by transcript
cds <- GenomicFeatures::cdsBy(txdb, use.names=TRUE)
orfCoord <- cds[names(cds) %in% names(listReadsCodon)]
# get the genome, please check that the genome has the same seqlevels
genomeSeq <- BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
# if not rename it
gSeq <- GenomeInfoDb::renameSeqlevels(genomeSeq,
  sub("chr", ",", GenomeInfoDb::seqlevels(genomeSeq)))
# codon frequency, coverage, and annotation
codonData <- codonInfo(listReadsCodon, genomeSeq, orfCoord)
```
PCA graphs on codon coverage

Description
PCA graphs on codon coverage

Usage
codonPCA(data, typeData)

Arguments
data a list of 2 data.frames: one with the number of times each codon type is found
in each ORF and one with the number of reads for each codon in each ORF.
typeData a character string. It is used as title for the PCA. Ex. typeData="codonCoverage"

Value
a list of length 2: PCA_scores - matrix of the scores on the first 4 principal components. PCA_plots
- a list of 5 PCA scatterplots.

Examples
#How to perform a PCA analysis based on codon coverage
#adapted from
#http://stackoverflow.com/questions/20260434/test-significance-of-clusters-on-a-pca-plot
#either get the codon frequency, coverage, and annotation using a function
#such as codonInfo in this package
#or create a list of matrices with the above information
data(codonDataCtrl)
codonData <- codonDataCtrl
codonUsage <- codonData[[1]]
codonCovMatrix <- codonData[[2]]

#keep only genes with a minimum number of reads
nbrReadsGene <- apply(codonCovMatrix, 1, sum)
ixExpGenes <- which(nbrReadsGene >= 50)
codonCovMatrix <- codonCovMatrix[ixExpGenes, ]

codonCovMatrixTransp <- t(codonCovMatrix)
rownames(codonCovMatrixTransp) <- colnames(codonCovMatrix)
colnames(codonCovMatrixTransp) <- rownames(codonCovMatrix)

listPCACodonCoverage <- codonPCA(codonCovMatrixTransp,"codonCoverage")
print(listPCACodonCoverage[[2]])
#See aditional examples in the pdf manual
countShiftReads

Apply an offset on the read start along the transcript and returns the coverage on the 5pUTR, CDS, 3pUTR, as well as a matrix of codon coverage per ORF.

Description

Apply an offset on the read start along the transcript and returns the coverage on the 5pUTR, CDS, 3pUTR, as well as a matrix of codon coverage per ORF.

Usage

countShiftReads(exonGRanges, cdsPosTransc, alnGRanges, shiftValue, motifSize)

Arguments

- **exonGRanges**: a GRangesList. It contains the exon coordinates grouped by transcript.
- **cdsPosTransc**: a list. It contains the relative positions of the start and end of the ORFs. The transcript names in exonGRanges and cdsPosTransc should be the same.
- **alnGRanges**: A GRanges object containing the alignment information. In order to improve the performance the GAlignments BAM object should be transformed into a GRanges object with cigar match size metadata.
- **shiftValue**: integer. The offset for recalibrating reads on transcripts when computing coverage. The default value for this parameter is 0, no offset should be performed.
- **motifSize**: an integer. The number of nucleotides in each motif on which to compute coverage and usage. Either 3, 6, or 9. Default 3 nucleotides (codon).

Value

A list with 2 objects. The first object in the list is a data.frame containing: information on ORFs (names, chromosomal position, length) as well as the counts on the 5pUTR, CDS and 3pUTR once the offset is applied. The second object in the list is a list in itself. It contains: for each ORF in the cdsPosTransc, for each codon the sum of read starts covering the 3 codon nucleotides. For motifs of size 6 nucleotides, the motif coverage is computed only for the first codon in the motif, considered as the codon in the P-site. For motifs of size 9 nucleotides, the motif coverage is computed only for the second codon in the motif, considered as the codon in the P-site. This per codon coverage does not contain information on the codon type, just its position in the ORF and its coverage.

Examples

```r
#read the BAM file into a GAlignments object using #GenomicAlignments::readGAlignments
#the GAlignments object should be similar to ctrlGAlignments
data(ctrlGAlignments)
aln <- ctrlGAlignments

#transform the GAlignments object into a GRanges object (faster processing)
```
alnGRanges <- readsToStartOrEnd(aln, what="start")

# make a txdb object containing the annotations for the specified species.
# In this case hg19.
# Please make sure that seqnames of txdb correspond to
# the seqnames of the alignment files ("chr" particle)
# if not rename the txdb seqlevels
# renameSeqlevels(txdb, sub("chr", "", seqlevels(txdb)))

# get all CDSs by transcript
cds <- GenomicFeatures::cdsBy(txdb, by="tx", use.names=TRUE)

# get all exons by transcript
exonGRanges <- GenomicFeatures::exonsBy(txdb, by="tx", use.names=TRUE)
# get the per transcript relative position of start and end codons
# cdsPosTransc <- orfRelativePos(cds, exonGRanges)
data(cdsPosTransc)
# compute the counts on the different features after applying
# the specified shift value on the read start along the transcript
# countsData <- countShiftReads(exonGRanges[names(cdsPosTransc)], cdsPosTransc,
# alnGRanges, -14)

---

countsPlot

**Graphs of sample read counts (quality assessment)**

**Description**

Graphs of sample read counts (quality assessment)

**Usage**

```r
countsPlot(listCounts, ixCounts, log2Bool)
```

**Arguments**

- `listCounts`: a list of data.frame objects. It contains the counts on the genomic features. Each data.frame in the list should have the same number of columns.
- `ixCounts`: a numeric (a vector of integers). It contains the index of the columns containing counts in the dataFrame.
- `log2Bool`: a numeric, either 0 or 1. 0 (default) for no log2 transformation and 1 for log2 transformation.

**Value**

A list of pairs and boxplots between the counts data in each data.frame.
Examples

#read the BAM file into a GAlignments object using
#GenomicAlignments::readGAlignments
#the GAlignments object should be similar to ctrlGAlignments
data(ctrlGAlignments)
aln <- ctrlGAlignments

#transform the GAlignments object into a GRanges object (faster processing)
alnGRanges <- readsToStartOrEnd(aln, what="start")

#make a txdb object containing the annotations for the specified species.
#In this case hg19.
txdb <-.TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene
#Please make sure that seqnames of txdb correspond to
#the seqnames of the alignment files ("chr" particle)
#if not rename the txdb seqlevels
#renameSeqlevels(txdb, sub("chr", ",", seqlevels(txdb)))
#get the flanking region around the promoter of the best expressed CDSs

#get all CDSs by transcript
cds <- GenomicFeatures::cdsBy(txdb,by="tx",use.names=TRUE)

#get all exons by transcript
exonGRanges <- GenomicFeatures::exonsBy(txdb,by="tx",use.names=TRUE)

#get the per transcript relative position of start and end codons
cdsPosTransc <- orfRelativePos(cds, exonGRanges)

#compute the counts on the different features after applying
#the specified shift value on the read start along the transcript
countsData <-
countShiftReads(
  exonGRanges[names(cdsPosTransc)],
  cdsPosTransc,
  alnGRanges,
  -14
)

#now make the plots
listCountsPlots <- countsPlot(
  list(countsData[[1]]),
  grep("_counts$", colnames(countsData[[1]])),
  1
)
listCountsPlots

ctrlGAlignments

Ribosome profiling data on chr1 in human primary BJ fibroblasts control data: PMID: 23594524.
Description

A dataset containing the alignment information on chromosome 1 from the control BAM. The data object is a GAlignments object containing 3,504,859 hg19 mapped reads.

Usage

data(ctrlGAlignments)

Format

A GAlignments object with 3,504,859 reads.

Value

the GAlignments object of reads on chr 1

histMatchLength

Histogram of match length distribution of reads.

Description

Histogram of match length distribution of reads.

Usage

histMatchLength(aln, log10Transf = 0, titleHist)

Arguments

aln        A GAlignments object of the BAM mapping file.
log10Transf  A boolean. Either 0 (default) or 1 (log10).
titleHist  a character. The main title for the histogram. Default - none.

Value

A list with 2 elements. The first element: a data.frame of the number of counts per match length distribution. The second element in the list: a ggplot2 histogram of the match length distribution.

Examples

#starting from a GAlignment object
data(ctrlGAlignments)
aln <- ctrlGAlignments
#no log10 scaling
matchLenDistr <- histMatchLength(aln, 0)
#to plot the histogram
matchLenDistr[[2]]
orfRelativePos  

**Description**

Relative position of the start and stop codon along the transcript

**Usage**

orfRelativePos(cdsTransc, exonGRanges)

**Arguments**

cdsTransc a GRangesList. It contains the CDS coordinates grouped by transcript.

exonGRanges a GRangesList. It contains the exon coordinates grouped by transcript.

**Value**

a list. A list of relative positions of the start and end of ORFs.

**Examples**

# make a txdb object containing the annotations for the specified species.
# In this case hg19.

cds <- GenomicFeatures::cdsBy(txdb, by="tx", use.names=TRUE)

exonGRanges <- GenomicFeatures::exonsBy(txdb, by="tx", use.names=TRUE)

cdsPosTransc <- orfRelativePos(cds, exonGRanges)

plotSummarizedCov  

**Description**

Plots the summarized coverage in a specified range (e.g. around TSS) for the specified match sizes

**Usage**

plotSummarizedCov(covSummarized)
Arguments

covSummarized a list of GRanges objects. For each matchSize a GRanges object of the summarized coverage.

Value

a ggplot2 plot of read coverage in interval

Examples

```
#read the BAM file into a GAlignments object using
#GenomicAlignments::readGAlignments
#the GAlignments object should be similar to ctrlGAlignments
data(ctrlGAlignments)
aln <- ctrlGAlignments
#transform the GAlignments object into a GRanges object (faster processing)
alnGRanges <- readsToStartOrEnd(aln, what="start")
#make a txdb object containing the annotations for the specified species.
#In this case hg19.
#Please make sure that seqnames of txdb correspond to
#the seqnames of the alignment files ("chr" particle)
#if not rename the txdb seqlevels
#renameSeqlevels(txdb, sub("chr", "", seqlevels(txdb)))
#get the flanking region around the promoter of the best expressed CDSs
oneBinRanges <- aroundPromoter(txdb, alnGRanges)
#the read start coverage around the TSS as a percentage for all match sizes.
covSummarized <- readStartCov(alnGRanges, oneBinRanges, matchSize="all",
c(-20,20), "aroundTSS", charPerc="perc")
trackPlotTSS <- plotSummarizedCov(covSummarized)
print(trackPlotTSS)
```

printPCA

Plots the PCA scatterplots produced by codonPCA function.

Description

Plots the PCA scatterplots produced by codonPCA function.

Usage

```
printPCA(listPCAGraphs)
```

Arguments

listPCAGraphs a list of 5 PCA ggplot scatterplots.

Value

a unique plot with the 5 PCA scatterplots.
Examples

# How to perform a PCA analysis based on codon coverage
data(codonDataCtrl)
codonData <- codonDataCtrl
codonUsage <- codonData[[1]]
codonCovMatrix <- codonData[[2]]

# Keep only genes with a minimum number of reads
nbrReadsGene <- apply(codonCovMatrix, 1, sum)
ixExpGenes <- which(nbrReadsGene >= 50)
codonCovMatrix <- codonCovMatrix[ixExpGenes, ]

# Get the PCA on the codon coverage
codonCovMatrixTransp <- t(codonCovMatrix)
rownames(codonCovMatrixTransp) <- colnames(codonCovMatrix)
colnames(codonCovMatrixTransp) <- rownames(codonCovMatrix)

listPCACodonCoverage <- codonPCA(codonCovMatrixTransp,"codonCoverage")
printPCA(listPCACodonCoverage[[2]])

---

**readStartCov**  
*Read start coverage around the TSS on the predefined CDSs*

**Description**

Read start coverage around the TSS on the predefined CDSs

**Usage**

```
readStartCov(alnGRanges, oneBinRanges, matchSize = "all", fixedInterval, renameChr, charPerc = "perc")
```

**Arguments**

- `alnGRanges`: A GRanges object containing the alignment information. In order to improve the performance transform the GAlignments BAM object into a GRanges object containing cigar match size as metadata.

- `oneBinRanges`: A GRanges object. Transform the gene GRangesList into one big GRanges object. Add the info on the cds_id.

- `matchSize`: either "all" or a vector of read match sizes. If matchSize <- "all", then all the reads are used to compute the coverage. If the matchSize is a vector of read match sizes, the summarized coverage is reported per match size and for the sum up.

- `fixedInterval`: a numeric vector with the extremities of the interval. Ex. fixedInterval <- c(-20,20) or fixedInterval <- c(0,40) ...

- `renameChr`: a character object. It contains the name to be given to the new summarized coverage interval. Ex. renameChr <- "aroundTSS" the summarized region around the TSS.
readsToStartOrEnd

Description
Reads in GAlignments converted to either Read Start (5') or End (3') Positions

Usage
readsToStartOrEnd(aln, what)
**Arguments**

- **aln**: A GAlignments object of the BAM mapping file.
- **what**: A character object. Either "start" (the default) or "end" for read start or read end.

**Value**

A GRanges object containing either the read start or end genomic positions.

**Examples**

```r
# read the BAM file into a GAlignments object using
# GenomicAlignments::readGAlignments
# the GAlignments object should be similar to ctrlGAlignments object
data(ctrlGAlignments)
aln <- ctrlGAlignments
# transform the GAlignments object into a GRanges object (faster processing)
alnGRanges <- readsToStartOrEnd(aln, what = "end")
```

---

**RiboProfiling**

*RiboProfiling.*

**Description**

RiboProfiling.

---

**riboSeqFromBAM**

*Starting from a BAM file path: quality plots, shift ribosome position, coverage on multiple transcript features and on codons.*

**Description**

Starting from a BAM file path: quality plots, shift ribosome position, coverage on multiple transcript features and on codons.

**Usage**

```r
riboSeqFromBAM(listeInputBamFile, paramScanBAM, genomeName, txdb,
               percBestExpressed, flankSize, offsetStartEnd, listShiftValue)
```
**Arguments**

listInputBamFile
A character path or a vector of paths to the ribo-seq BAM file(s). If multiple BAM files are provided, they should come from the same genome alignment.

paramScanBAM
NULL or ScanBamParam object specifying what fields and which records are imported. Default value is NULL.

geneName
a character object containing the name of the genome used for the alignment BAM file. The name should be one of the UCSC ensGene list: ucscGenomes()[, "db"]. Ex. "hg19" or "mm10". This parameter is used to build the TxDb object.

txdb
a TxDb object containing the annotations info to confront with the alignment files. Either genomeName or txdb parameters should be provided.

percBestExpressed
numeric [between 0 and 1]. The percentage of best expressed CDSs on which to plot the coverage around the TSS. Necessary if the shiftValue parameter must be estimated. Default value 0.03 (3%).

flankSize
an integer. How many bp left and right of the TSS should the coverage be performed?

offsetStartEnd
a character object. Either "start" (the default) or "end" for read start or read end to define the offset.

listShiftValue
a vector of integer. It should have the same length as the inputBamFile vector. The numeric value for shifting ranges of reads on genomic features when computing coverage. Set this parameter to 0 if no shift should be performed. If this parameter is missing, the shiftValue is computed based on the maximum peak of read start coverage around the TSS. A plot is produced to illustrate this estimation.

**Value**

A list of list for each BAM file in the inputBamFile list. For each BAM file 2 objects are returned: one data.frame with info on the genomic features and the corresponding coverage column, and one list of per ORF codon coverage.

**Examples**

#the txdb object can be given as parameter or not.
#If it is not specified, a txdb object is build from UCSC.

#in this example only one BAM file is treated.
#However, multiple BAM files can be analyzed together.
myFile <- system.file("extdata", "ctrl_sample.bam", package="RiboProfiling")
listeInputBam <- c(myFile)

covData <- riboSeqFromBAM(listeInputBam, txdb=txdb, listShiftValue=c(-14))

#when running this function it is important that chromosome names
#in UCSC and your BAM correspond: the "chr" particle

Index

* datasets
  cdsPosTransc, 3
  codonDataCtrl, 4
  codonIndexCovCtrl, 4
  ctrlGAlignments, 9
  aroundPromoter, 2
  cdsPosTransc, 3
  codonDataCtrl, 4
  codonIndexCovCtrl, 4
  codonInfo, 5
  codonPCA, 6
  countShiftReads, 7
  countsPlot, 8
  ctrlGAlignments, 9
  histMatchLength, 10
  orfRelativePos, 11
  plotSummarizedCov, 11
  printPCA, 12
  readStartCov, 13
  readsToStartOrEnd, 14
  RiboProfiling, 15
  RiboProfiling-package (RiboProfiling), 15
  riboSeqFromBAM, 15