Package 'RiboProR'

February 14, 2020

Version 1.0

Title R Tools for Ribosomal Footprints Profiling Analysis

Author Hongen Zhang [aut, cre],
Fujun Zhou [aut]

Maintainer Hongen Zhang <hzhang@mail.nih.gov>

Depends R(>= 3.4)

Imports Biostrings,corrplot,DESeq2,IRanges,GenomicAlignments,GenomicRanges, gplots,grDevices,graphics,Hmisc,parallel,Rsamtools,rtracklayer,S4Vectors, stats,utils

Description A light version of RiboProTools providing simple methods for Ribosomal footprints profiling analysis including of: fp framing, fp counting, DESeq2 analysis for translational efficiency change, and few plot methods for data visualization.

License GPL (>=2.0)

biocViews Visualization, Riboseq, RNASeq, DESeq2

NeedsCompilation no

R topics documented:

`RiboProTools-package` *R Tools for Ribosomal Footprints Profiling Analysis*

Description

A list of methods for Ribosomal footprints (fp) profiling analysis was provided including of: fp framing, fp counting, DESeq2 analysis, and data visualization.

Details

Package: RiboProTools
Type: Package
Version: 1.0.0
Date: 2018-8-08
License: GPL (>=2)

Author(s)

Hongen Zhang Fujun Zhou Maintainer: Hongen Zhang <hzhang@mail.nih.gov>

`absoluteOuter` *Convert A Relative Position to An Absolute Genomic Coordinate*

Description

Convert a transcript-relative position to an absolute genomic coordinate based on a transcript whose coordinates are given by a GRanges of exons (outer).

Usage

```
absoluteOuter(qpos, outer)
```

Arguments

`qpos`	positive integer, a position relative to the start of a transcript.
`outer`	GRanges object, genomic positions of a transctipt.

Value

Positive integer, the geneomic coordinate of the transcript-relative position.

References

Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379

alignASites *Align a-site for GAlignments Object*

Description

Covert GAligments objects for a transcript to a GAlignments object having the A site nucleotides only.

Usage

```
alignASites(asiteOffsets, alns)
```

Arguments

asiteOffsets	A data frame of one column, row names are read lengths and column is A sites for each read length.
alns	A GAlignments objects for a transript.

Value

A GAlignments object for A site nucleotides only.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

bedCdsIRangesList *Get IRanges List for All CDS*

Description

For each transcript in a BED file and the associated transcript GRanges as computed by bedGRangesList, find the transcript-local CDS as per transcriptCdsIRanges and collect these as an IRangesList.

Usage

```
bedCdsIRangesList(bed, bedgrl)
```

Arguments

bed	GRanges list imported from bed file.
bedgrl	GRanges list of exons for all transcripts.

Value

IRanges list for each exon (CDS).

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`bedGRangesList` *Get GRanges List for All Transcripts in A BED File*

Description

For each transcript in a BED file GRanges, get the GRanges of its exons as per transcriptGRanges and collec these into a GRangesList with names taken from the name metadata column of the BED file GRanges.

Usage

```
bedGRangesList(bed)
```

Arguments

`bed`	GRanges list with block information defined.

Value

A GRanges list object for exons of all transcripts.

References

Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379

`checkBlocksInBed` *Validate the Information of Blocks Defined in BED File*

Description

Check out if the block information defined in bed file are correct. Block count must be positvie integer. Block sizes and block starts must be comma seperated list. Number of block sized and starts must match to the block counts.

Usage

```
checkBlocksInBed(bed_info)
```

Arguments

`bed_info`	A data frame with 12 columns read from bigBed file.It should be checked before call this function and must have required standard column names.

Value

Logic, TRUE for everything is correct. Otherwise FALSE.

References

https://genome.ucsc.edu/FAQ/FAQformat#format1.7

`checkChromosomeInfo` *Check Chromosome Information Between Fasta and BED Files*

Description

Check out if the chromsome names and length are same in the given two files.

Usage

```
checkChromosomeInfo(DNA_seq, bed_info)
```

Arguments

`DNA_seq`	A data frame with rows for chromosome(s) and columns for chromosome name(s) and sequence(s).
`bed_info`	A data frame with contents same as bigBed file.

Details

The first file is a fasta file with chromosome(s) and sequences. The second file is an annotation file in BigBed format (12 columns). This function will check if the number of chromosomes and chromosome names in the two files are same, and sequence length in fastq file is same as the chromosome length defined in bed file.

Value

Logic, TRUE if the relevant information in two files are same. Otherwise return an error message.

`converToGenePredFormat`
Covert ORF Information to GenePred Format

Description

Covert ORF table in tab-delimited format to GenePrep format.

Usage

```
converToGenePredFormat(ORF_file, is.one.based = TRUE, min_codon = 0)
```

Arguments

`ORF_file`	Character vector, name (and path) of tab-delimited file for ORF information.
`is.one.based`	Logic, if position is 1-based in input file. Default TRUE.
`min_codon`	Integer. 0 or bigger. Number of codon(s) between start and stop codons.

Details

The output has following columns (one column more than the ucsc gebe prediction table formt: gene_name orf_ID chromosomes strand tx_start tx_end cds_start cds_end num_exon exon_starts exon_end

Value

Data frame with 11 columns in GenePred format.

References

https://genome.ucsc.edu/FAQ/FAQformat.html#format9

`countAligns` *Count Aligned A-site for All Transcripts in Bed File*

Description

Scan a bam file and count aligned a-site for each transcript defined in bed file.

Usage

```
countAligns(asiteOffsets, insets, bamfile, trxBed)
```

Arguments

`asiteOffsets`	A data frame of one column with read length as row names and a-sites for each read length in column.
`insets`	List of integer with length of 4, insets in number of nucleotides for avoiding start and stop position.
`bamfile`	Character vector, name of a bam file (and path).
`trxBed`	GRange object for all transcripts.

Value

List of list, number of aligned a-site in transcript, cds, 5'UTR, and 3'UTR region of each transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

countAtAsite *Count A-site Along One Chromosome*

Description

Count number of reads which overlap at A-site on transcripts of one chromosome.

Usage

```
countAtAsite(alignments, chrom_len, asite_table)
```

Arguments

alignments	GRange list for alignments on one chromosome.
chrom_len	Positive integer, length of the chromosome.
asite_table	A data frame with one column for A-sites and rownames for qualified read length.

Value

A list of 2 integer vectors for counts on forward and reverse strand at each base pair position.

References

https://github.com/ingolia-lab/RiboSeq

countingFrames *Count Total Number of Reading Frames for All Aligments*

Description

Find the reading frames based on the start position of alignments.

Usage

```
countingFrames(alignments, cds_ranges, inset_5=34, inset_3=31)
```

Arguments

alignments	GRanges list for all reads from a bam file.
cds_ranges	GRanges list for all cds annotations.
inset_5	positive integer, shift distance after start position of cds to avoid start codon.
inset_3	positive integer, shift distance before end position of cds to avoid stop codon.

Value

A data frame of two columns for reading frames and read length of each read.

References

(https://github.com/ingolia-lab/RiboSeq)

countingMetagenePosition *Count Metagene Positions for Aligments*

Description

Find the positions relative to cds start and stop positions for all alignments.

Usage

```
countingMetagenePosition(alignments, annotation, shift_start=0, shift_end=0)
```

Arguments

alignments	GRanges list for short reads in a bam file.
annotation	GRanges list of cds annotations.
shift_start	Positive integer, shift this distance to 5'end from start position of cds.
shift_end	Positive integer, shift this distance to 3'end from end position of cds.

Value

A data frame with 3 columns for 3 columns: distance to cds start, distance to cds end, and read length, for each aligments.

References

(https://github.com/ingolia-lab/RiboSeq)

countSizes *Calculate Region Sizes for All Transcripts*

Description

Count sizes for whole transcript,, 5'UTR, cds, 3'UTR regions of each transcript.

Usage

```
countSizes(insets, trxBed)
```

Arguments

insets	List of integer, insets in nucleotides for avoiding star tand stop codons.
trxBed	GRanges list for all transcripts.

Value

List of sizes of whole transcript, cds, 5-UTR, and 3-UTR for each transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`covertToRNASequence` *Convert DNA Sequence to RNA*

Description

Convert DNA sequence to RNA sequence and reverse it if it is for reverse strand.

Usage

```
covertToRNASequence(DNA_seq, strand)
```

Arguments

`DNA_seq`	Character vector, a fragment of DNA sequences from forward strand.
`strand`	Character, either "+" or "-".

Value

A character vector containing series of "A", "U", "G", "C".

`extractEfficiencyChange` *Extract Results from DESeqDataSet Object For Transclational Efficiency Change.*

Description

Extract translational efficiency (TE) and transclational efficiency change (TEC) from DESeqDataSet object after calling of DESeq(). The output will contain all columns from results() and mcols().

Usage

```
extractEfficiencyChange(dds_object, control_name, mutant_name,
efficiency_type="TE", meta_cols=c(1:3))
```

Arguments

`dds_object`	A DESeq object, on which DESeq() has already been called. The design model must be ~ genotype + condition + genotype:condition.
`control_name`	Character vectors, name of control group, used for generation of output file name.
`mutant_name`	Character vectors, name of mutant group, used for generation of output file name.
`efficiency_type`	Character vectors, name of efficiency. Use "TE" for translational efficiency analysis or "RRO" for relative ribosomal occupancy analysis.
`meta_cols`	Positive integers, columns of meta-data to be attached to output. Set to NULL if all columns are attached or 0 for no meta-data.

Value

DESeqDataSet object, same for input. All results extracted are saved to files.

References

https://bioconductor.org/packages/release/bioc/manuals/DESeq2/man/DESeq2.pdf

---

`extractTranscriptionChange` *Extract Results from DESeqDataSet Object For Transcription Change*

Description

Extract transcription change between Riboseq samples and between RNASeq samples from DESeqDataSet object after running of DESeq(). The output will contain all columns from results() and mcols().

Usage

```
extractTranscriptionChange(dds_object, control_name, mutant_name,
change_one="mRNA", change_two="Ribo", meta_cols=c(1:3))
```

Arguments

`dds_object`	A DESeq object, on which DESeq() has already been called. The design model must use a grouping variable.
`control_name`	Character vectors, name of control group, used for generation of output file name.
`mutant_name`	Character vectors, name of mutant group, used for generation of output file name.
`change_one`	Character vector, name of changes between groups of RNASeq samples.
`change_two`	Character vector, name of changes between groups of RiboSeq samples.
`meta_cols`	Positive integers, columns of meta-data to be attached to output. Positive integers, columns of meta-data to be attached to output. Set to NULL if all columns are attached or 0 for no meta-data.

Value

A DESeq object, same as the input.

References

https://bioconductor.org/packages/release/bioc/manuals/DESeq2/man/DESeq2.pdf

filterCountMatrix *Filter Data Matrix Based on RiboSeq fp Counts and/or RNAseq Counts*

Description

Filter data matrix based on RiboSeq fp counts and RNAseq counts. Any rows with mean of RiboSeq fp counts or mean of RNASeq counts below the definded values will be removed.

Usage

```
filterCountMatrix(count_matrix=NULL, mRNA_col=NULL,
    ribo_col=NULL, mRNA_level=10, ribo_level=0)
```

Arguments

count_matrix	Data matrix with both RiboSeq fp counts and RNASeq reads counts.
mRNA_col	Positive integer, columns in the matrix for RNASeq reads counts.
ribo_col	Positive integer, columns in the matrix for RiboSeq reads counts.
mRNA_level	Numeric, threshold to filter rows by mean of RNASeq reads counts.
ribo_level	Numeric, threshold to filter rows by mean of RiboSeq fp counts.

Value

A data matrix same as input matrix with unqualified rows removed.

Examples

```
data(ribo_pro_data);

mRNA_col <- c(4:6, 10:12);
ribo_col <- c(1:3, 7:9);

mRNA_level <- 10;
Ribo_level <- 0;

count_matrix <- filterCountMatrix(ribo_pro_data, mRNA_col,
ribo_col, mRNA_level, Ribo_level);
```

filterOutAlignments *Filter Aligments/Reads*

Description

Filter out alignments/reads by read length and genomic regions.

Usage

```
filterOutAlignments(alignments, asite_table, annot_bed_file)
```

Arguments

alignments	GRange list for alignments/reads in GRange list.
asite_table	A data frame with one column for A-sites and rownames for qualified read length.
annot_bed_file	Character vector, name (and path) of annotation file in bed format.

Value

GRange list with qualified alignments/reads.

---

fpCountsBoxPlot *Counts Data Box Plot*

Description

Make box plot with counts data fron DESeqDataSet Object

Usage

```
fpCountsBoxPlot(dds, normalize=c("size", "fpm", "fpkm"),
label_area=2, main_text="Distribution of fp Counts")
```

Arguments

dds	A DESeqDataSet object
normalize	Character vector, method of normalization either "size", "fpm", "fpkm", or NULL.
label_area	Positive integer, height of area at botom of plot for sample labels.
main_text	Character vector, text for title of the plot.

Examples

```
## Not run:
data(dds_TE)
fpCountsBoxPlot(dds_TE)

## End(Not run)
```

fpFramePlot *Bar Plot of Reading Frames Data*

Description

Plot reading frames distributions of ribosomal foot prints (bar plot).

Usage

```
fpFramePlot(frame_file, by_column=TRUE, fp_column=c(2:4), read_len=c(25:30),
legend_pos="topleft", title_text="Ribosomal fp Frame Distribution")
```

Arguments

frame_file	Character vector, name of the file (and path) with reading frame counts for each read length.
by_column	Logic, if the frame data arraned by column (row is for each read length). If TRUE, fp_column must be defined.
fp_column	Positive integer, column number for reading frame counts.
read_len	Positive integer vector, length of reads (alignments).
legend_pos	Character vector, location of legend, one of "topleft", "center", "topright".
title_text	Character vector, text for title.

Examples

```
## Not run:
frame_file <- system.file("extdata", "RiboProTools_ribo.frame_len.txt",
package = "RiboProTools")
fpFramePlot(frame_file)

## End(Not run)
```

FpFraming *Get Framing Information for All Reads in A Bam File*

Description

Perform framing analysis for all reads in a bam file and generate profiles for reading frames, meta-genes, and a-sites.

Usage

```
FpFraming(bam_file, bed_file, parameters=NULL, validate.parameters=TRUE, save.file=TRUE)
```

Arguments

bam_file	Character vector, name of bam file (and path).
bed_file	Character vector, name of bed file (and path) for gene annotation.
parameters	List of integers for FpFraming parameters.
validate.parameters	Logic, if TRUE, non-default parameters will be validated.
save.file	Logic, if true, all tables (total of 6) will be saved to files.

Details

This is the main function for framing analysis. It scans a bam file and extract information for metagene, read frames, and find a-sites for reads of each qualified length. Outputs will be saved as tab-delimited text files.

Value

A data frame containing a-sites for fragments with selected length.

References

https://github.com/ingolia-lab/RiboSeq

---

getAlignASites *Get Aligned A sites for A Transcript*

Description

Scan bam file for all reads of a transcript then get aligned a-sites for the transcript.

Usage

```
getAlignASites(asiteOffsets, bamfile, trx)
```

Arguments

asiteOffsets	A data frame of one column with read length as row names and a-sites for each read length in column.
bamfile	Character vector, name of a bam file (and path).
trx	GRange object for a transcript.

Value

A GAlignments object for A site nucleotides only.

References

Oroginal code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

getAlignments *Get Alignments for A TRanscript from BAM File*

Description

Read in all reads compatible with a processed transcript on the correct strand, as per findSpliceOverlaps.

Usage

```
getAlignments(bamfile, trx)
```

Arguments

bamfile	Character vector, the name (and path) of the bam file to be read.
trx	GRanges object for a transcript.

Value

A GAlignments object with all reads overlapping with the transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

getAlignmentsFromBamFile *Read All Reads from BAM File*

Description

Scan a BAM file and hold all reads with GRanges list.

Usage

```
getAlignmentsFromBamFile(bam_file)
```

Arguments

bam_file	Character vector, a bam file name (and path).

Value

Granges list containing all reads in bam file. Keep the reads unfiltered but hold seqname, start and end, strand information only.

References

http://samtools.github.io/hts-specs/SAMv1.pdf

`getAllAlignments` *Get All Alignments for A Transcript from BAM File*

Description

Get all reads overlapping the genomic extent of a primary transcript on the correct strand, including unspliced and purely intronic reads.

Usage

```
getAllAlignments(bamfile, trx)
```

Arguments

`bamfile`	Character vector, the name (and path) of the bam file to be read.
`trx`	GRanges object for a transcript.

Value

A GAlignments object with all reads overlapping with the transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`getAnnotationFromBedFile`
Read Annotation Information from BED File

Description

Import annotation information from BED file and convert them to two GRanges lists.

Usage

```
getAnnotationFromBedFile(bed_file)
```

Arguments

`bed_file`	Character vector, file name (and path) for annotations in BED format. The file must be in Bed12 format (with thick columns defined).

Details

import() function from rtracklayer package is used to bring in annotation informaton to GRanges list.

Value

List of two GRanges objects. The first one contains all information in the bed file and the second one has cds information only.

References

https://genome.ucsc.edu/FAQ/FAQformat.html#format1.7

`getAnnotationFromBigBedFile`
Extract Annotation Information from BigBed File

Description

Read bigBed file (12 columns) into data frame and check the contents.

Usage

```
getAnnotationFromBigBedFile(bed_file, has.header=FALSE, sepcial_chrom=NULL)
```

Arguments

`bed_file`	Character vector, name (and path) of the bigbed file.
`has.header`	Logic, if the bigbed file has column headers. Default is FALSE.
`sepcial_chrom`	character vector, chromosome names other than digits, roman numbers, and "X", "Y", "M".

Details

Fields in bigBed file: 1) chrom: chromosome name of each feature 2) chromStart: start position of each feature 3) chromEnd: end position of each feature 4) name: gene name of each feature 5) score: used for graphic display only 6) strand: chromosome strand of each feature 7) thickStart: starting position drawn thickly 8) thickEnd: ending position drawn thickly 9) itemRgb: used for graphic display only 10) blockCount: number of blocks of each feature 11) blockSizes: comma-separated list of block sizes 12) blokStarts: comma-separated list of block starts, relative to chromStart

chromStart and chromEnd are 0-based and half-open.

Value

A data frame with 12 columns.

References

https://genome.ucsc.edu/FAQ/FAQformat#format1.7

getAnnotationInfo *Extract Annotation Information for a List of Genes*

Description

Extract gene ID, gene name, and description for a list of genes from annotation file.

Usage

```
getAnnotationInfo(annotation, gene_list, id_column=1, name_column=2, description=3)
```

Arguments

annotation	A data frame with at least 3 columns for gene ID, gene names, and descriptions.
gene_list	Character vector, list of gene names for which the annotation information is extracted.
id_column	Positive integer, column of gene ID in annotation file.
name_column	Positive integer, column of gene name in annotation file.
description	Positive integer, column of gene description in annotation file.

Value

A data frame with 3 columns (gene ID, gene name, and description) for the list of genes.

getASiteProfile *Find All A-site for Alignments of A Transcript*

Description

Count number of aligned a-sites at each base position of a transcripts.

Usage

```
getASiteProfile(asiteOffsets, bamfile, trx)
```

Arguments

asiteOffsets	A data frame of one column with read length as row names and having a-sites for each read length in the column.
bamfile	Character vector, name of a bam file (and path).
trx	GRange object for a transcript.

Value

Integer vector, total aligned a-site at each base position of a transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

getASiteTable *Generate A-site Table*

Description

Generate a-site table from the three position profile tables.

Usage

```
getASiteTable(frame_table, at_start, at_end, parameters)
```

Arguments

frame_table	A data frame with 7 columns for read_len, fraction, counts of the three reading frames.
at_start	A numeric matrix with rows for metagene positions realtive to cds start and columns for read length.
at_end	A numeric matrix with rows for metagene positions realtive to cds end and columns for read length.
parameters	List of length 13, all parameters for a-site calculation.

Value

A data frame with rows for each read length and columns (total of 9) for read length, fraction of each length, peak relative to cds start, peak relative to cds end, other information, fraction of frame0, frame1, frame2, and best a-site.

References

(https://github.com/ingolia-lab/RiboSeq)

getBamFlagStat *Extract Flags from A BAM File*

Description

Scan bam flags to get statistics of the reads. This requires samtools available from system (either path to samtools is included in user's PATH variable or the module has been loaded in HPC system).

Usage

```
getBamFlagStat(bamFiles, outFile)
```

Arguments

bamFiles	Character vector, names of bam files.
outFile	Character vector, names of output file.

Value

None. Write the flags to output file.

References

http://www.htslib.org/doc/samtools.html

`getBestFrame` *Find the Best Reading Frame*

Description

Find the best frame from Frame 0~2 for each read length.

Usage

```
getBestFrame(frame_table)
```

Arguments

`frame_table`	An data frame with counts of 3 readind frames for each read length.

Value

Positive integer vector, frame number for each read length,

References

(https://github.com/ingolia-lab/RiboSeq)

`getBigWiggleFile` *Generate Coverage File in bigWig Format ()*

Description

Generate bigwig files from bam file. Only coverage of aligned reads are reported and no normalization performed. There will be two bigWig files saved to current directory, one for gorward strand and another for reverse strand.

Usage

```
getBigWiggleFile(bam_file, transcripts)
```

Arguments

`bam_file`	Character vector, bam file name (and path)
`transcripts`	GRange list for transcripts.

Author(s)

Henry Zhant

Examples

```
## Not run:
yeast_bed <- rtracklayer:import("yeast-all.bed");
getBigWiggleFile("riboseq_t1.bam", yeast_bed)

## End(Not run)
```

---

getCDSGRanges *Extract CDS Regions from Transcript Definition*

Description

Covert GRanges of transcripts to GRanges of CDS regions.

Usage

```
getCDSGRanges(bed_file)
```

Arguments

bed_file	character vector, name (and path) of a bed file defining transcripts and relevant CDS regions.

Value

GRange list for CDS regions.

---

getCDSInfoDefinedByBed *Extract CDS Information and Sequence*

Description

Extract genomice sequence and other annotation items for CDS defined in bed file.

Usage

```
getCDSInfoDefinedByBed(DNA_seq, bed_info)
```

Arguments

DNA_seq	A data frame with rows for chromosome(s) and columns for chromosome name(s) and their sequence(s).
bed_info	A data frame with contents same as bigBed file (12 columns)

Value

A data frame with 7 columns for: chromosome, start_pos, end_pos, strand, locus, sequence, type ("CDS")

References

https://genome.ucsc.edu/FAQ/FAQformat#format1.7

`getChromosomeSizesFromBam` *Get Chromosome Sizes from A BAM File*

Description

Extract length of each chromosome from bam file header.

Usage

```
getChromosomeSizesFromBam(bam_file)
```

Arguments

`bam_file`	Character vector, bam file name (and path).

Value

A data frame with 2 columns for chromosome names and lengths.

`getCodonFromSequence` *Split the Given mRNA Sequence to Codons*

Description

Covert the given mRNA sequence to codons starting the position of start_at.

Usage

```
getCodonFromSequence(mRNA_seq, start_at)
```

Arguments

`mRNA_seq`	Character vector, mRNA seqwence (base A, U, G, C only).
`start_at`	Positive integer, the start position to split the sequence. Must be 1, 2, or 3.

Value

Character vector, codons starting from the start_at position in the sequence.

Examples

```
getCodonFromSequence("GAUGAGCUAGGAC", start_at=2)
# "AUG" "AGC" "UAG" "GAC"
```

getCodonIndex *Get the Index of A Codon in Codon List*

Description

Find which codon(s) in the codon list matches to the target codon.

Usage

```
getCodonIndex(codon_list, target_codon)
```

Arguments

codon_list	Character vector, a serial codons from a RNA sequences.
target_codon	Character vector, one or more start or stop codon(s).

Value

Positive integer vector, index of the codon(s) which matched to target codon.

getComplementarySequence *Convert a fragment of sequence to its complementary contents*

Description

Get complementary sequence for a fragment of DNA sequence (no RNA)

Usage

```
getComplementarySequence(seq_fragment)
```

Arguments

seq_fragment	Character vector, a fragment of DNA sequence.

Value

Character vector, the complementary sequence of the input with A -> T, G -> C, T -> A, and C -> G.

getCountMatrixFromFiles *Read Raw Counts from Text Files and Generate Counts Matrix*

Description

Read multiple tab-delimited text files and extract raw counts column, then merge all raw counts as one matrix with rows for genes and columns for samples.

Usage

```
getCountMatrixFromFiles(directory_name, file_name_pattern, count_column,
rowname_column = 0, has.header = TRUE)
```

Arguments

directory_name	character vector, name of directory where raw counts files are stored.
file_name_pattern	character vector, common pattern in all raw counts files. This pattern should not be used by other files in same directory.
count_column	Positive integer, number of column for raw counts in the file.
rowname_column	Positive integer, the column in raw count file for row names. Set to 0 for raw count files saved by write.table() with row.names=TRUE.
has.header	Logical, if the raw count files have column headers.

Value

A numeric matrix with rows for genes and columns for samples. File names are used as column headers in this matrix.

Note

All raw counts files must be generated with same software or have same number of rows and columns as well as same row and column orders).

Examples

```
directory_name <- "raw_count_files"
file_name_pattern <- "raw_counts.txt"
count_column <- 2
rowname_column <- 0
has.header <- TRUE

## Not run: getCountMatrixFromFiles(directory_name, file_name_pattern,
count_column, rowname_column, has.header)
## End(Not run)
```

getCountMatrixFromTable *Extract Raw Counts from A Matrix or Data Frame*

Description

Generate a matrix from a matrix or data frame with defined columns.

Usage

```
getCountMatrixFromTable(count_table=NULL, ribo_control=NULL,
ribo_treatment=NULL, mRNA_control=NULL, mRNA_treatment=NULL)
```

Arguments

count_table	A data frame or matrix with raw counts from both Riboseq and RNAseq data for same samples. Row names must be gene names or gene IDs.
ribo_control	Positive integer vector, columns in input table for control samples of Riboseq data.
ribo_treatment	Positive integer vector, columns in input table for treatment or mutant samples of Riboseq data.
mRNA_control	Positive integer vector, columns in input table for control samples of RNASeq data.
mRNA_treatment	Positive integer vector, columns in input table for treatment or mutant samples of RNASeq data.

Value

A matrix with columns in the order of mRNA control samples, mRNA treatment samples, ribosomal control samples, and ribosomal treatment samples

Examples

```
## Not run:
data("ribo_pro_data.RData")
getCountMatrixFromTable(count_table=ribo_pro_data, ribo_control=7:9,
ribo_treatment=10:12, mRNA_control=1:3, mRNA_treatment=4:6)

## End(Not run)
```

getCountsData *Get Counts Data from DESeqDataSet Object*

Description

Extract raw counts data or normalized counts from DESeqDataSet object.

Usage

```
getCountsData(dds, normalize=c("size", "fpm", "fpkm"))
```

Arguments

dds	A DESeqDataSet object.
normalize	Character vector, method of normalization either "size", "fpm", "fpkm", or NULL.

Value

A matrix holds raw or normalize counts data.

Examples

```
## Not run:
data("dds_TE")
fpm <- getCountsData(dds_TE, normalize="fpm")

## End(Not run)
```

---

getDefaultParameters *Get Default Parameters*

Description

Methods used for get default parameters.

Usage

```
getAllDefaultParameters()
getDefaultInset()
getZeroInset()
getDefaultRiboSeqAsites()
getDefaultRNASeqAsites()
getDefaultFlank()
getDefaultCdsBody()
getDefaultLengths()
getDefaultMinLenFract()
getDefaultStartRange()
getDefaultEndRange()
getDefaultStartShift()
getDefaultEndShift()
getDefaultStartCodons()
getDefaultStopCodons()
getDefaultExtraBounds()
getDefaultMinNumberOfAminoAcid()
```

Value

getAllDefaultParameters() return a list of integers used for ribosomal footprints framing:

shift_start:100 shift_end:100 inset_5: 34 inset_3: 31 min_len: 25 max_len: 34 minLenFract: 0.05, minStartRange: -17, maxStartRange: -8, minEndRange: -22, maxEndRange: -13, startShift: 3, endShift: -2

getDefaultInset() return a list of integers in order to avoid start and stop codons:

utr5Inset3: 6, cdsInset5: 45, cdsInset3: 15, utr3Inset5: 6

getZeroInset() return a list of 4 zeros for utr5Inset3, cdsInset5, cdsInset3, and utr3Inset5.

getDefaultRiboSeqAsites() return a data frame of 1 colum for a-sites for Riboseq reads with length 26:31.

getDefaultRNASeqAsites() return a data frame of 1 colum for a sites for RANSeq reads with length 18:51.

getDefaultFlank() return an integer vector of length 2 for flank regions at both end of gene cds: -100 and 100.

getDefaultCdsBody() return an integer vector of length 2 to calculate CDS body range: 34 and 31.

getDefaultLengths() return an integer vector of length 2 for minimum and mazimum read length: 25 and 34.

getDefaultMinLenFract() return a numeric for minimum required fraction of read length: 0.05.

getDefaultStartRange() return an integer vector of length 2 for range before start codon.

getDefaultEndRange() return an integer vector of length 2 for range before stop codon.

getDefaultStartShift() return a integer (3) for start position when shift starts.

getDefaultEndShift() return a integer (-2) for stop position when shif stops.

getDefaultStartCodons() return a character vector of length 10 for start codons.

getDefaultStopCodons() return a character vector of length 3 for stop codons.

getDefaultExtraBounds() return an integer vector of length 2 for extra bound of a region (such as cds).

getDefaultMinNumberOfAminoAcid() return an integer (2) for minimum number of ammino acids required for a gene.

`getDefaultStartCodon` *Get the Default Start Codon List*

Description

The default start codon list will be "AUG", "AUC", "AUU", "AUA", "AGG", "ACG", "AAG", "CUG", "UUG", and "GUG".

Usage

```
getDefaultStartCodon()
```

Value

Character vector, the default set of start codons.

getDefaultStopCodon *Get the Default Stop Codon List*

Description

The default stop codon set is "UAG", "UGA", "UAA".

Usage

```
getDefaultStopCodon()
```

Value

Character vector, the default set of stop codons.

getDESeqDataSet *Initialize A DESeqDataSet object*

Description

Generate a DESeqDataSet object with a counts matrix that contains control and treatment/mutant samples from both RiboSeq and RNASeq data.

Usage

```
getDESeqDataSet(count_matrix, num_Ribo_wildtype, num_Ribo_mutant,
num_mRNA_wildtype, num_mRNA_mutant, annotation_info=NULL)
```

Arguments

count_matrix	A matrix with columns in the order of: mRNA control samples, and mRNA treatment samples, ribosomal control samples, and ribosomal treatment samples.
num_Ribo_wildtype	Positive integer, total number of wildtype/control samples with RiboSeq fp counts.
num_Ribo_mutant	Positive integer, total number of mutant/treament samples with RiboSeq fp counts.
num_mRNA_wildtype	Positive integer, total number of wildtype/control samples with RNASeq fp counts.
num_mRNA_mutant	Positive integer, total number of mutant/treament samples with RNASeq fp counts.
annotation_info	Data frame with columns for gene ID, gene name, and description.

Details

This function set up a DESeqDataSet using DESeqDataSetFromMatrix(). The countData will be the count_matrix, colData will have both condition ("mRNA and "Ribo") and genotype("wildtype" and "mutant"), and design will be ~ genotype + condition + genotype:condition. The annotation_info will be added into mcols, if provided.

Value

A DESeqDataSet object.

References

https://bioconductor.org/packages/release/bioc/manuals/DESeq2/man/DESeq2.pdf

---

getFilteredFPCenter *Calculate Center of Filtered Ribosomal Footprints for A Transcript*

Description

Calculate center of ribosomal footprints for a transcript by removing positions where have no a-site aligned.

Usage

```
getFilteredFPCenter(asiteProfile)
```

Arguments

asiteProfile	Vector of positive integer, counts of a-site on each nucleotide position of a transcript

Value

Positive float number, length of the 5' end of transcript that has half of total counts divided by transcript length

---

getFootprintDensityCenter *Main Function to Calculate Center of Ribosomal Footprint Densities*

Description

Calculate centers of ribosomal footprint density for each transcript in a bam file .

Usage

```
getFootprintDensityCenter(a_sites, transcripts, bam_file, weight = FALSE)
```

Arguments

a_sites	Data frame with 1 column for a-site. Row names are read length.
transcripts	GRanges list of all transcripts.
bam_file	Character vector, name (and path) of a bam file.
weight	Logic, if the position with zero count should be filtered out.

Value

A data frame with one column of positive integers for density center of all transcripts. Row names of the data frame are gene ID/names.

getFrameTable *Generate Reading Frame Table from Frame Profile Table*

Description

Convert readind frame profile to readind frame table for total number and fraction of each frame.

Usage

```
getFrameTable(frameProfile)
```

Arguments

frameProfile	A data frame with two columns for frames and length of each read

Value

A data frame with 7 columns for read_len, fraction, counts of the three reading frames.

References

(https://github.com/ingolia-lab/RiboSeq)

getMetageCountsMatrix *Generate Matrix of Reads Counts for Metagene Positions*

Description

By giving a list of metagene frame table files, generate a matrix for metagene counts plot. Rows of the matrix are reads counts on each defined metagene position and columns are samples. The matrix is sorted by maximum value of each column in decreasing order so that the y range of the plot will be automatically decided.

Usage

```
getMetageCountsMatrix(count_files, from_position, to_position)
```

Arguments

count_files	Character vector, names (and path) of metagene reads count files.
from_position	Integer, start position of metagene to plot.
to_position	Integer, stop position of metagene to plot

Value

Data matrix with read counts along metagene position from multiple samples.

Author(s)

Henry Zhang

getMetageneFrames *Write Reading Frames of Metagene to Files*

Description

Counting total number of reading frames for all metagene positions (positions of read start relative to cds start).

Usage

```
getMetageneFrames(posProfile, bam_file_name)
```

Arguments

posProfile	A data frame with three columns for to_start, to_end, and read_len of each reads in a Ba file.
bam_file_name	Character vector, name of bam file from which the reads are scaned for metagene positions. Used for output file generation.

Details

Two output files will be generated. One for metagene positions relative to cds start and one for metagene position relative to cds stop. Each file has three columns for total counts, frame, and metagene position.

getORFPositions *Calculate Relevant Genomic Positions for a ORF*

Description

Calculate genomic positions, for a ORF, includeing of start, stop, distance to cap (UTR start), and distance to main AUG codon.

Usage

```
getORFPositions(seq_info, start_index, stop_index, start_at, include_stop_codon)
```

Arguments

seq_info	Information of UTR (one row from data frame)
start_index	Integer, start codon index in an ORF.
stop_index	Integer, stop codon index in an ORF.
start_at	Integer, start point in sequence to read codons (frame)
include_stop_codon	Logic, if include stop codon in outputs.

Value

Positive integer vector for start and stop positions, distance from cap and to main AUG codon.

getOriginalFPCenter *Calculate Center of Ribosomal Footprint Density Without Filtering*

Description

Calculate center of ribosomal footprint density for a transcript without filtering.

Usage

```
getOriginalFPCenter(asiteProfile, trx_length)
```

Arguments

asiteProfile	Vector of positive integers, count of asite aligned to the transcript at each nt position.
trx_length	Positive integer, length of the transcript.

Value

Positive float number, ratio of length of the 5' end of transcript that has half of total counts divided by transcript length

getPlotColors *Get R Colors for Plot*

Description

Generate a default color list of length 24 or rainbow colors with length more than 24.

Usage

```
getPlotColors(num_colors)
getDefaultColors()
```

Arguments

num_colors	Positive integer, total number of colors.

Value

getPlotColors(num_colors) returns a vector of R colors with length of num_colors.

getDefaultColors() returns a vector of predifined R colors with length of 24.

getPositionProfileTable *Generate Metagene Table*

Description

Convert position profile table to metagene table.

Usage

```
getPositionProfileTable(positionProfile=NULL, at_which=1,
meta_start=-100, meta_end=100)
```

Arguments

positionProfile	A data frame of 3 columns for relative position to cds start, to cds end, and read length.
at_which	A positive integer, 1 for to start position or 2 for to end position.
meta_start	Negative integer, position before cds start in metagene table
meta_end	Positive integer, position after cds end inmetagene table

Value

A numeric matrix with rows for metagene positions and columns for read length.

References

(https://github.com/ingolia-lab/RiboSeq)

getPrettyLabels *Format Axis Labels*

Description

Modifiy the axis labels for metagene frame plot.

Usage

```
getPrettyLabels(min_val, max_val)
```

Arguments

min_val	Integer, minimum value of the axis label.
max_val	Integer, maximum value of the axis label.

Value

Integer vector.

References

http://bioconductor.org/packages/riboSeqR/

Examples

```
label_5p <- getPrettyLabels(-50, 200);
```

`getReadPeaks` *Find Read Peaks from Metagene Table*

Description

Find count peaks for each read length from metagene length table.

Usage

```
getReadPeaks(metagene, min_range, max_range)
```

Arguments

`metagene`	A data frame with columns for each read length and rows for counts in each position of metagene.
`min_range`	Positive integer, minimum index of the row to find peak.
`max_range`	Positive integer, maximum index of the row to find peak.

Value

An integet vector, distance from metagene start for each read length.

References

(https://github.com/ingolia-lab/RiboSeq)

`getRegionFrames` *Generate Metagene Frame Table for A Specific Region*

Description

Extract a subset from full metagene frame table and convert them to a three row data frame.

Usage

```
getRegionFrames(frame_file, from, to)
```

Arguments

`frame_file`	Character vecor, name (and path) of the file continning frames at each metagene position.
`from`	Integer, the start of a sub region of the metagene.
`to`	Integer, the end of a sub region of the metagene.

Value

A data frame with 3 rows for counts of frame0, frame1, and frame2 at each metagene position.

Examples

```
## Not run:
frame_file <- system.file("extdata", "frame_at_start.txt",
package="RiboProTools")
frame_table <- getRegionFrames(frame_file, -50, 200);

## End(Not run)
```

---

getReversedSequence *Reverse a Fragment of DNA/RNA Sequence*

Description

Simply reverse a fragment of sequence (character vector).

Usage

```
getReversedSequence(seq_fragment)
```

Arguments

seq_fragment	Character vector, a fragment of DNA/RNA sequence.

Value

Character vector, same bases as input sequences but in reversed order.

---

getSequence *Extract Sequence From Fasta File*

Description

Get sequence from fasta file for a transcript.

Usage

```
getSequence(fafile, trx)
```

Arguments

fafile	Character vector, the fasta file name (and path).
trx	GRanges object for a transcript.

Value

Character vector, DNA sequence of the transcript.

getSequenceFromMutipleFastaFiles

Extract Genomic Sequences from Multiple Fasta Files

Description

Read fasta files which are orgnaized by chromosome and extract relevant chromosome names and sequence.

Usage

```
getSequenceFromMutipleFastaFiles(file_path, file_type)
```

Arguments

file_path	character vector, path to the directory which holds the fasta files.
file_type	Character vector, file extention. Valid types are "fa", "FA", "fasta", and "FASTA".

Value

A data frame containing sequence data where rows are for each chromosome and columns are for chromosome names and sequences

getSequenceFromOneFastaFile

Extract Chromosome Names and Sequences from Fasta File

Description

Read fasta file which include multiple chromosomes and relevant sequences. Simply put them in data frame for easy use.

Usage

```
getSequenceFromOneFastaFile(file_name)
```

Arguments

file_name	Character vector, name (and path) of a fasta file.

Value

A data frame wiht row(s) for chromosome(s) and columns for chromosome names and sequences.

getStartCodonContext *Extract Contex for A Start Codon*

Description

Build the context (a short sequence fragment from the third nucleotide before start codon and the one nucleotide next to start codon). There will be total of 7 nucleotides, e.g., GACAUGG, AUCAUGC).

Usage

```
getStartCodonContext(codon_list, start_index, mRNA_seq, start_at)
```

Arguments

codon_list	Character vector, codons from the sequence.
start_index	Integer, index of start codon. Always greater than 0.
mRNA_seq	Character vector, mRNA sequence converted from UTR/CDS sequence
start_at	Integer, either 1, 2 or 3.

Value

Character vector, nucleotides around start codon.

getTranscriptLength *Calculate Length of Transcripts*

Description

Calculate transcript length for exon only (remove introns, if any).

Usage

```
getTranscriptLength(transcripts)
```

Arguments

transcripts	GRange list with transcripts

Value

A data frame of one column for transcript length and with gene names as row names.

getTranscriptReadsOnly

Filter Alignments/Reads in GRange List by Genomic Regions

Description

Filter out alignments/reads to keep that overlapped with genomic regions defined in annotation file.

Usage

```
getTranscriptReadsOnly(alignments, annot_bed_file)
```

Arguments

alignments	GRange object of all alignments to be filtered out.
annot_bed_file	Character vector, name (and path) of annotation file in bed format.

Value

GRange object of filtered alignments.

getUORCountMatrixFromOneFileSet

Generate uORF Counts Matrix from One Set of Files

Description

Generate uORF count matrix from files which contains both fp counts for uORF and cds regions.

Usage

```
getUORCountMatrixFromOneFileSet(directory_name,
uorf_file_pattern, mrna_file_pattern=NULL,
count_column=2, rowname_column=0, has.header=TRUE,
uorf_ID_patter1 = "^.{4}-", uorf_ID_patter2=NULL)
```

Arguments

directory_name	Character vector, name of the directory where raw counts files are stored.
uorf_file_pattern	Character vector, the common pattern of raw counts files.
mrna_file_pattern	Character vector, the common pattern of RNASeq raw count files if using RNASeq reads counts instead of cds fp counts.
count_column	Positive integer, the column of raw count in counts files.
rowname_column	Positive integer, the column in raw counts file for row names of new matrix. set to 0 if raw counts files were saved by write.table() with row.names=TRUE.
has.header	Logical, if the raw counts files have column headers.

uorf_ID_patter1
: Character vector, unique pattern used to extract gene ID from uorf ID.

uorf_ID_patter2
: Character vector, the second unique pattern, if exists, used to extract gene ID from uorf ID.

Value

An integer matrix with raw footprints (fp) counts of both uORF and fp counts (or reads counts, if using RNASeq) cds regions. Rows of the matrix are for uORF/genes and columns for samples. The left half of the matrix is fp counts from cds regions of each gene and the right half is fp counts from uORF region of same gene. The samples in uORF part and cds part have same orders.

---

getUORFCountMatrixFromTwoFileSets

Generate uORF Counts Matrix from Two Counts Files

Description

Generate raw uORF counts matrix from two sets of counts files: footprint counts for uORF regions and footprints counts for gene cds regions.

Usage

```
getUORFCountMatrixFromTwoFileSets(file_directory,
uorf_file_pattern="uorf_conserved_fpcounts.txt",
cds_file_pattern="cds_for_uorf_fpcounts.txt",
count_column=2, rowname_column=0, has.header=TRUE,
seperator = "\\.")
```

Arguments

file_directory
: Character vector, name of a directory where raw counts files are stored.

uorf_file_pattern
: Character vectors, common pattern in uORF counts files.

cds_file_pattern
: Character vectors, common pattern in cds counts files.

count_column
: Possitive interger, the column of raw counts in counts file.

rowname_column
: Possitive interger, which column will be used as row names. Set to 0 if count files are output from R write.table() with row.names=TRUE.

has.header
: Logic, if the input files have headers.

seperator
: Character vector, unique pattern to separate gene name and uorf ID in uorf names. It CANNOT be any character used in gene name or uorf id.

Value

An integer matrix with raw footprints (fp) counts of both uORF and cds regions. Rows of the matrix are for uORF/genes and columns for samples. The left half of the matrix is fp counts from cds regions of each gene and the right half is fp counts from uORF region of same gene. The samples in uORF part and cds part have same orders.

Note

The two file sets have different rows, one is for uORF fp counts only and another is for cds region fp counts only. The uORF ID/names in raw uORF counts file must contain gene name and uORF ID separated by the 'seperator', e.g., "YGL134W.255638" where YGL134W is gene name, "." is separator, and 255638 is uORF ID. The data matrix generated with this function is for DESeq2 analysis with normalization by subset, i.e., normalizing uORF and cds fp counts seperately. It MAY NOT be suitable for default normalization.

`getUORFCountMatrixInFujunFormat`
Generate uORF FP Counts Matrix in Fujun Format

Description

Generate a uORF footprints counts matrix in a special format. See Details.

Usage

```
getUORFCountMatrixInFujunFormat(file_directory,
uorf_file_pattern = "uorf_conserved_fpcounts.txt",
cds_file_pattern = "cds_for_uorf_fpcounts.txt",
count_column=2, rowname_column=0, has.header=TRUE)
```

Arguments

`file_directory`	Character vector, name of a directory where raw footprints counts files are stored.
`uorf_file_pattern`	Character vectors, common pattern in uORF footprints counts files.
`cds_file_pattern`	Character vectors, common pattern in cds footprints counts files.
`count_column`	Possitive interger, the column of raw counts in counts files.
`rowname_column`	Possitive interger, which column will be used as row names. Set to 0 if count files are output from R write.table() with row.names=TRUE.
`has.header`	Logic, if the input files have headers.

Details

The Fujun Format is specifically designed for relative risobomal occupancy (RRO, uORF fp counts divided by cds fo counts) analysis. Since in most cases the uORF fp counts is much lower than the counts of cds region in same gene, simply normalizing the uORF counts with relative cds fp counts together may extremely shift the values of log2 change ratio. With Fujun Format, the coumns for uORF counts in the matrix contains fp counts for all uORFs and all cds regions, the columns for cds region fp counts have cds region fp counts of genes matched to each uORF plus fp counts of all genes. A default DESeq2 analysis could be applied with data matrix in this format.

Value

An integer matrix with raw footprints (fp) counts of both uORF and cds regions. Rows of the matrix are for uORF/genes and columns for samples. The left half of the matrix is fp counts from uORF regions of each gene and the right half is fp counts from cds region of same gene. The samples in uORF part and cds part have same orders.

Note

The input files are outputs based on two different BED files but the rows in two BED files are matched each other. The uORF ID/names in raw uORF counts file must contain gene name and uORF ID separated by the 'seperator', e.g., "YGL134W.255638" where YGL134W is gene name, "." is separator, and 255638 is uORF ID.

---

getUTRInfoDefinedByBed

Extract UTR Information From Both of Fasta and BED Files

Description

Extract annotation information from bed file and sequence from fasta file.

Usage

```
getUTRInfoDefinedByBed(DNA_seq, bed_info)
```

Arguments

DNA_seq	A data frame with rows for chromosome(s) and columns for chromosome name(s) and sequence.
bed_info	A data frame with contents same as bigBed file.

Value

A data frame with 7 columns for:

chromosome: chromosome name of each UTR start_pos: start position of each UTR end_pos: end position of each UTR strand: strand of each UTR locus: gene name of each UTR sequence: DNA sequence of each UTR type: type of each fragment (5UTR or 3UTR)

All positions are forward strand based.

References

https://genome.ucsc.edu/FAQ/FAQformat#format1.7

---

getUTRInfoFromFastaFile

Extract UTR Information from Fasta File

Description

Process fasta file for UTR to get annotation items and sequence for each UTR, e.g., 5UTR.fa downloaded from https://www.pombase.org/downloads/utr

Usage

```
getUTRInfoFromFastaFile(fasta_file, header_name)
```

Arguments

fasta_file	Character vector, name (and path) of fasta file.
header_name	Character vector, definition of each header field.

Details

A typical record in fast format is much like below:

>SPAC212.06c|SPAC212.06c.1|18558|18974|1||I|protein_coding| DNA helicase in rearranged telomeric region, truncated| CTACACATTACGCTGAGAGGTAAAATACTCTGACAACATTCGTTCGATTGTATAAAACAA AATCCAGCCGAAACGATTGTTGTCAGTAATCAAGATTACGATCTAAATTGAGTACCAAGA CAAAACGAAATGGTTAAAAAGTTAAAGTCGTTTTTGTATGGACACAATTTCTATAAAATA GACATGAGTAAAATCTCGCTATTTGTTTGTTATTGTGGAATAATGAAGAGTCATGGGAGA TGAATGTTGTAAACGATGGCATAGAATTGGTAACGAAAAGTGAAATCGTTGGGATCAACT ATTTCAGTATTTTGTTTAAAGAAAATGTTGAACTCGACAAGTAATGAGAGGTGGTGCTTT CGTTAAATAATGAGTGGTGGTTACGGTTATACAGGATATGATATGTGTATGGTGAGA

The header line includes items for Systematic_ID UTR ID Gene Start Gene_End Strand Chromosome Feature Type Description

Value

A data frame with rows for UTR and columns for annotation items and sequence.

References

https://www.pombase.org/downloads/utr

---

getWiggleCounts *Get A-site Counts for Wiggle File Generation*

Description

Count total number of aligned a-sites for full genome or transcripts only.

Usage

```
getWiggleCounts(bam_file, asite_table, annot_bed_file)
```

Arguments

bam_file	Chraracter vector, name of bam file (and path)
asite_table	A data frame with 1 column for a-site and rownames for read length. A file name is also accepted.
annot_bed_file	Chraracter vector, name of a bed file for gene/transcript annotation.

Value

List of list with each sub-list is wiggle counts for forward and reverse strand of one chromosome.

Note

This function counts coverage for forward and reverse strand separately and the coverage is on the A-site only.

References

https://github.com/ingolia-lab/RiboSeq

`initializeMetageneFrameTable`
Initialize A Metagene Frame Table

Description

Initialized a matirx with 3 columns for counts, frames, and unique metagene positions.

Usage

```
initializeMetageneFrameTable(position_set)
```

Arguments

`position_set`	Integer vector, unique position set of all metagene positions (position relative to cds start or cds stop positions)

Value

A data frame of 3 columns for number of frames, frame names, and metagene positions.

References

https://github.com/ingolia-lab/RiboSeq

Examples

```
## Not run:
positions <- -100:100;
frames_table <- initializeMetageneFrameTable(positions);
head(frames_table);

## End(Not run)
```

initializeMetageneTable *Initialize A New Metagene Table*

Description

Initialize an empty matrix with rows for base positions of metagene and columns for read length.

Usage

```
initializeMetageneTable(meta_start=-100, meta_end=100, min_len=25, max_len=34)
```

Arguments

meta_start	Negative integer, distance before cds start for metagene start.
meta_end	Positive integer, distance after cds end for metagene end.
min_len	Positive integer, minimum length of a read.
max_len	Positive integer, maximum length of a read.

Value

A matrix of 0s(zero) with rows for metagene positions and columns for read length.

References

(https://github.com/ingolia-lab/RiboSeq)

irangeOuter *Get Genomic Ranges for Transcript-Relative IRanges*

Description

Convert transcript-relative query IRanges to an absolute genomic GRanges based on a transcript coordinates in a GRanges of exons.

Usage

```
irangeOuter(qranges, outer)
```

Arguments

qranges	IRanges object with relative positions to a start of transcript.
outer	GRanges object, genomic positions of a transctipt.

Value

A GRanges object for the genomic positions.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`is.DNA.sequence` *Check Out If A Sequence Is from DNA*

Description

Check out the given sequence is DNA by matching its based to A, T, G, and C.

Usage

```
is.DNA.sequence(DNA_seq)
```

Arguments

`DNA_seq`	Character vector, s fragment of DNA sequence. No 'N' or white space allowed.

Value

Logic, TRUE if the given sequence is from DNA Otherwise FALSE.

Examples

```
is.DNA.sequence("AGCTTAGGCCAAT") # TRUE
is.DNA.sequence("AGCUUAGGCCAAU") # FALSE
```

`is.mRNA.sequence` *Check Out If A Sequence Is from mRNA*

Description

Check out if the given sequence is from mRNA, i.e., its bases must be A, U, G, or C.

Usage

```
is.mRNA.sequence(mRNA_seq)
```

Arguments

`mRNA_seq`	Character vector, s fragment of mRNA sequence. No 'N' or white space allowed.

Value

Logic, TRUE if the given sequence is from mRNA Otherwise FALSE.

Examples

```
is.mRNA.sequence("AUGCGAAUGGCC") # TRUE
is.mRNA.sequence("ATGCGAATGGCC") # FALSE
```

mapASite *Mapping A-site for Each Alignment/Read*

Description

Convert the GRange object to one base width GRanges based on relevant a-site.

Usage

```
mapASite(alignments, asite_table)
```

Arguments

alignments	GRange object of alignments on one strand to be filtered out.
asite_table	A data frame of 1 columns to hold a-site for each read length which are represented by row names.

Value

GRange object with one base width.

normalizeWiggleCounts *Normalize Wiggle Track Counts*

Description

Normalize wiggle track counts to a defined total counts.

Usage

```
normalizeWiggleCounts(all_counts, normalize_factor = NULL)
```

Arguments

all_counts	List of list, each list element has two numeric vectors for read counts forward and reverse strand at base pair level.
normalize_factor	positive numeric, scaling factor. If not provided, total wiggle counts will be scaled to 1000000000.

Value

List of list, normalized counts for each strand.

References

https://github.com/ingolia-lab/RiboSeq

plotCorrelationHeatmap *Plot Correlation Heastmap for samples*

Description

Make a correlation image for samples in DESeqDataSet with blue and red colors.

Usage

```
plotCorrelationHeatmap(dds, normalize=c("size", "fpm", "fpkm"),
image_name, image_type="pdf", image_size=12)
```

Arguments

dds	A DESeqDataSet object.
normalize	Character vector, method of normalization either "size", "fpm", "fpkm", or NULL.
image_name	Character vector, output image file name.
image_type	Character vector, output image format, either "pdf", "tiff", or "png".
image_size	Positive integer, image height.

Examples

```
## Not run:
data("dds_TE")
plotCorrelationHeatmap(dds_TE, image_name="corr.image.pdf")

## End(Not run)
```

plotCorrelationMatrix *Plot Correlation Matrix Data*

Description

Plot correlation matrix with corrplot() provided by corrplot package.

Usage

```
plotCorrelationMatrix(plot_data, cor_method="rcorr",
cor_type="spearman", shape_type="ellipse", p_threshold=0.01)
```

Arguments

plot_data	A numeric matrix, the data with which correlation coefficients will be calculated.
cor_method	Character vector, function name for correlation calculation, either "rcorr" (default) or "cor".
cor_type	Character vector, correlation type, either "spearman" or "pearson".
shape_type	Character vector, shape to represent the correlation coefficients on the plot image, one of "circle", "square", "ellipse", "number", "shade", "color", or "pie".
p_threshold	Numeric, the threshold for significant level to filter the correlation coefficients.

plotHeatmap *Heatmap Plot with Data Matrix*

Description

Make heatmap plot with data matrix and heatmap.2() in gplots package.

Usage

```
plotHeatmap(plot_value, sample_name, gene_name,
image_type="pdf", image_width = 8,
is.log2=FALSE, scale_by="row")
```

Arguments

plot_value	Numeric matrix, values for heatmap plot.
sample_name	Character vector, column (sample) labels.
gene_name	Character vector, row (gene) labels.
image_type	Character vector, output image format, one of "pdf", "tiff", and "png".
image_width	Positive integer, width of output image in inches.
is.log2	Logic, is the data log2 transformed.
scale_by	Character vector, how the data is scaled, either "row" or "column"

plotMetageneCounts *Plot Metagene Counts Distribution*

Description

Make line plot showing metage counts distribution around positions relative to CDS start and stop positions.

Usage

```
plotMetageneCounts(count_file, from_position=-50,
to_position=50, codon="Start", x_interval=10)
```

Arguments

count_file	Character vector, name of the file which contains frame name and frame counts for each metagene position relative to cds start position.
from_position	Integer, leftmost position relative to cds start or stop position.
to_position	Integer, rightmost position relative to cds start or stop position.
codon	Character vector, either 'Start' or 'Stop'.
x_interval	Integer, length between tick-marks for x-axis.

Examples

```
## Not run:
at_start <- system.file("data", "ribo_meta_start_frames.txt",
plotMetageneCounts(at_start);
title("Metagene Counts Distribution");
## End(Not run)
```

---

plotMetageneFrames *Plot Frames of Metagene Regions*

Description

Make bar plot showing frame distribution at positions relative to CDS start and stop positions.

Usage

```
plotMetageneFrames(metagene_atStart=NULL, metagene_atStop=NULL,
    min_5p=-20, max_5p=200, min_3p=-200, max_3p=20,
    frame_colors=c("red", "green", "blue"), beside=TRUE)
```

Arguments

metagene_atStart	Character vector, name of the file which contains frame name and frame counts for each metagene position relative to cds start position.
metagene_atStop	Character vector, name of the file which contains frame name and frame counts for each metagene position relative to cds stop position.
min_5p	Integer, minimum distance to cds start positon.
max_5p	Integer, maximum distance to cds start positon.
min_3p	Integer, minimum distance to cds stop positon.
max_3p	Integer, maximum distance to cds stop positon.
frame_colors	Character vector or R colors vector of length 3, plot colors for reading frames.
beside	Logic, if the columns for same group alongside each other.

Examples

```
## Not run:
at_start <- system.file("data", "ribo_meta_start_frames.txt",
package="RiboProTools");
at_stopt <- system.file("data", "ribo_meta_stop_frames.txt",
package="RiboProTools");
plotMetageneFrames(at_start, at_stopt);
title("Metagene Frame Distribution");
## End(Not run)
```

plotMultiMetageneCounts *Plot Reads Counts on Metagene Positions*

Description

Generate a line plot to show reads counts on each defined metagene position for multiple samples.

Usage

```
plotMultiMetageneCounts(count_files, from_position=-50, to_position=50,
codon="Start", x_interval=10, line_colors = c("red", "blue", "green"))
```

Arguments

count_files	Character vector, names (and path) of metagene reads count files generated by FpFraming().
from_position	Integer, start position of metagene to plot.
to_position	Integer, stop position of metagene to plot.
codon	Character vector, codon name, either "Start" or "Stop" for x-axis labeling.
x_interval	Positive integer, interval for metagene position lables on x-axis.
line_colors	Character vector of R color names for lines.

Author(s)

Henry Zhang

plotRedBlueCorrelationImage *Plot Correlation Image with Red-Blue Colors*

Description

Plot pairwise correlation matrix with blue and red colors.

Usage

```
plotRedBlueCorrelationImage(corr_data, image_name,
image_type="pdf", image_width=12)
```

Arguments

corr_data	Numeric matrix, pairwise correlation coefficients of samples.
image_name	Character vector, output image file name.
image_type	Character vector, output image format, either "pdf", "tiff", or "png".
image_width	Positive integer, size of squared image.

Note

There is no scree output and image will be saved to file.

plotReplicates *Plot Raw or Normalized Counts of Sample Replicates*

Description

Make scatter plot with count data of two replicate samples in DESeqDataSet.

Usage

```
plotReplicates(dds, normalize=c("size", "fpm", "fpkm"),
replicates=c(1, 2), show.cor=TRUE, is.log2=FALSE,
point_color="grey", line_color="red",
x_label="replicate 1", y_label="replicate 2",
main_text="Distribution of fp Counts")
```

Arguments

dds	A DESeqDataSet object.
normalize	Character vector, method of normalization either "size", "fpm", "fpkm", or NULL.
replicates	Positive integer vector of length 2. The columns in counts data to be plotted.
show.cor	Logic, if plot corelation coefficiency and the regression line.
is.log2	Logic, is the data log2 transformed.
point_color	Character vector of R colors names, color for the point background.
line_color	Character vector, colors for regression line.
x_label	Character vector, text for x axis label
y_label	Character vector, text for y axis label
main_text	Character vector, text for title of the plot

Examples

```
## Not run:
data("dds_TE")
plotReplicates(dds_TE, replicates=1:2)

## End(Not run)
```

plotTranslationEfficiency *Scatter Plot with Translational Efficiency Data*

Description

Scatterplot with too translational efficiency (TE) values.

Usage

```
plotTranslationEfficiency(dds, ratio_level=1,
x_label="TE of Wild Type", y_label="TE of Mutant",
title_text="Translation Efficiency Of Mutant",
x_pos=0, y_pos=12)
```

Arguments

dds	A DESeqDataSet object.
ratio_level	Positive numeric, threshold to change point colors.
x_label	Character vector, text for x axis label.
y_label	Character vector, text for y axis label.
title_text	Character vector, text for title of the plot.
x_pos	Integer, x coordinate for text showing pearson's r
y_pos	Integer, y coordinate for text showing pearson's r

Examples

```
## Not run:
data("dds_TE")
plotTranslationEfficiency(dds_TE)

## End(Not run)
```

---

quickDESeq2Test *Perform A Quick DESeq Analysis for Ribosomal Footprints Profiling*

Description

A simple way to perform DESeq2 analysis with ribosomal footprints profiling data and gene annotation information.

Usage

```
quickDESeq2Test(count_table, Ribo_wildtype, Ribo_mutants,
mRNA_wildtype, mRNA_mutants, control_name, mutant_name,
mRNA_level, Ribo_level, annotation_file)
```

Arguments

count_table	Numeric matrix with RNASeq reads counts and Riboseq footprints counts for both control and treamtment/mutant samples.
Ribo_wildtype	Positive integer vectors, column numbers in count matrix for raw counts of RiboSeq wildtype samples.
Ribo_mutants	Positive integer vectors, column numbers in count matrix for raw counts of RiboSeq treatment/mutant samples.
mRNA_wildtype	Positive integer vectors, column numbers in count matrix for raw counts of RNASeq wildtype samples.

mRNA_mutants	Positive integer vectors, column numbers in count matrix for raw counts of RNASeq treatment/mutant samples.
control_name	Character vector, name of control group.
mutant_name	Character vector, names of treatment/mutant group.
mRNA_level	Numeric, threshold to filter out the matrix based on row means of RNASeq samples.
Ribo_level	Numeric, threshold to filter out the matrix based on row means of RiboSeq samples.
annotation_file	Character vector, name of gene annotation file

Value

None. All results will be saved to file.

Examples

```
## Not run:
data("ribo_pro_data.RData")
annotation_file <- system.file("data", "yeast_gene_descriptions.txt",
package="RiboProTools");
quickDESeq2Test(ribo_pro_data, Ribo_wildtype=c(1:3), Ribo_mutants=c(7:9),
mRNA_wildtype=c(4:6), mRNA_mutants=c(10:12),
control_name="ribo_WT", mutant_name="ribo_MT",
mRNA_level=10, Ribo_level=1, annotation_file)

## End(Not run)
```

---

readAsiteFromFile *Read a_site from A-site file*

Description

Read a-site table generated by FpFraming(), covert it to a one column data frame with required column name.

Usage

```
readAsiteFromFile(a_site_file)
```

Arguments

a_site_file	Characte vector, name (and path) of a-site file.

Value

A data frame with one column for a-site of each read length. Read length are used for row names.

References

Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

ReadBamToTable *Scan A BAM File and Convert Results to A Data Frame*

Description

Read a bam file and convert the object (list of list) to a data frame

Usage

```
ReadBamToTable(bam.file)
```

Arguments

bam.file	character vector, name of the bam file (and path).

Value

A data frame with all conternts read from BAM file.

References

Morgan M, Pages H, Obenchain V and Hayden N (2016). Rsamtools: Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import. R package version 1.26.1,

http://bioconductor.org/packages/release/bioc/html/Rsamtools.html.

regionCountFrame *Convert Aligned A-site of All Transcripts to A Data Frame*

Description

Convert the aligned a-site of all transcripts from a list of list to a data frame.

Usage

```
regionCountFrame(counts)
```

Arguments

counts	List of list, number of aligned a-site in transcript, cds, 5'UTR, and 3'UTR region of each transcript.

Value

Data frame with 4 columns, total number of aligned a-site in transcript, cds, 5'UTR, and 3'UTR region of each transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`relativeWithin` *Convert a Genomic Position to Position Relative to CDS Start*

Description

Convert an absolute genomic query position (qpos) to a transcript-relative coordinate position within a transcript.

Usage

```
relativeWithin(qpos, outer)
```

Arguments

`qpos`	A positive integer, a genomic coordinate inside of a transcript.
`outer`	GRanges object, genomic positions of a transctipt.

Value

Positive integer, a poistion relative to the start of transctipt or NA if the absolute position is not in any exon (no hit).

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`ribo_pro_data` *Sample Data for DESeq2 Analysis*

Description

A numeric matrix of raw counts from RiboSeq and RNASeq Data including 5440 genes and 6 samples for both RNASeq and RiboSeq. Row names are gene names.

Usage

```
data("ribo_pro_data")
```

Format

The format is: numeric matrix.

Source

Unpublished data.

Examples

```
## Not run:
data(ribo_pro_data)
dim(ribo_pro_data)
head(ribo_pro_data)
## End(Not run)
```

`runDESeq` *Run DESeq on A DESeqDataSet*

Description

Run DESeq() or other related functions based on the contents of the DESeqDataSet.

Usage

```
runDESeq(deseq_dataset, has.SizeFactors=FALSE,
reset.design=FALSE, fit_type="parametric")
```

Arguments

`deseq_dataset`	A DESeqDataSet with design model of ~ genotype + condition + genotype:condition, on which DESeq() or other related functions will be called.
`has.SizeFactors`	Logic, if the deseq_dataset has size factor already or not.
`reset.design`	Logic, if need reset design model.
`fit_type`	Character vector, fit type used by DESeq() or estimateDispersions().

Details

This function will apply DESeq() or other related functions based on the contents of the DESeq-DataSet object. If the DESeqDataSet has no sizeFactors calculated, it will call DESeq() otherwise it will call estimateDispersions() and nbinomWaldTest(). By default, the design model is ~ genotype + condition + genotype:condition, which is for translational efficiency test. For tranascription test, it must be changed to ~group. This will be done if set reset.design to TRUE.

Value

A DESeqDataSet, on which DESeq() or relative functions has been called.

References

https://bioconductor.org/packages/release/bioc/manuals/DESeq2/man/DESeq2.pdf

screenORF *Extract ORF Information from A Sequence Fragment*

Description

Screen a sequence fragment to get start codon, context from -3 to +4 base positions of start codon, length of ORF, distance from the cap, and distance to the uAUG in a list.

Usage

```
screenORF(seq_info, codon_list, codon_pair, start_at, mRNA_seq)
```

Arguments

seq_info	One row from a data frame, sequence info including of chromosome, start_pos, end_pos, strand, locus, sequence, and type.
codon_list	Character vector, codons converted from UTR sequence.
codon_pair	Matrix of column 2 for paired start and stop index.
start_at	Integer in 1, 2, 3, base where start to read (frame).
mRNA_seq	Character vector, a fragment of mRNA sequence.

Details

Screen a mRNA sequence to get information including of: gene_name chromosome seq_start seq_stop strand orf_id orf_start orf_stop start_codon context orf_length dis_from_cap dis_to_MUG

Value

A data frame with columns for information above.

setCodonIndexPair *Pairing Start and Stop Codons in An ORF Range*

Description

Pair start and stop codons in an ORF ranges. For each start codon, only the first stop codon in its downstream can be used. If there is no stop codon found, the end of sequence is used (index 0).

Usage

```
setCodonIndexPair(start_index, stop_index)
```

Arguments

start_index	Integer vector, start codon index, always greater than 0.
stop_index	Integer vector, stop codon index, could be 0 or integer(s) greater than 0.

`sortTableByChromosomeNames` *Sort A Table by Chromosome Names*

Description

Sort a table by chromosome names (either Arabic or Roman numbers)

Usage

```
sortTableByChromosomeNames(chrom_info, name_col=1, type="digit")
```

Arguments

`chrom_info`	A data frame or matrix with one column for chromosome names.
`name_col`	Positive integer, number of the column for chromosome names.
`type`	Character vector, type of chromosome numbers, either "digit" or "roman".

Value

Data frame or matrix same as the input but sorted by chromosome names.

`summerizeMetageneFrames` *Summerize Metagene Framse from Metagene Position Profile*

Description

Summrize reading frames for each metagene position (position of read start relative to cds start or to cds stop position)

Usage

```
summerizeMetageneFrames(posProfile, pos_col=1)
```

Arguments

`posProfile`	A data frame with three columns for to_start, to_end, and read_len for each reads.
`pos_col`	Positive integer, column number in the data frame above.

Value

A data frame of 3 columns for total counts, frame, metagene position, and rows for each metagene position

transcriptCdsIRanges *Convert Exons in GRanges Object to IRanges Object*

Description

Convert the thick block information in a GRanges object to transcript-relative IRanges object.

Usage

```
transcriptCdsIRanges(trx, thickStart, thickEnd)
```

Arguments

trx	GRanges object for a transcript with block information.
thickStart	Positive integer, start position of CDS in the transcript.
thickEnd	Positive integer, end position of CDS in the transcript.

Value

An IRanges object for relative start and end position of CDS.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

transcriptGRanges *Get GRanges from BED Content for Exons of A Transcript*

Description

Extract exon information from a transcript GRanges object and hold with GRanges list.

Usage

```
transcriptGRanges(bedGRange)
```

Arguments

bedGRange	A GRange object for a transcript with blocks in metadata columns.

Value

GRanges list with one GRange per exon.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

trxRegionCountAligns *Count Aligned A-site in A Transcripts by Regions*

Description

Check out if reads of a transcript has A site. If yes, get the total counts for transcript, cds, 5-UTR, and 3-UTR. Otherwise, set counts of the transcript to 0, and 0 or NA(if no cds defined) for cds, 5-UTR, and 3-UTR.

Usage

```
trxRegionCountAligns(asiteOffsets, insets, bamfile, trx, cds)
```

Arguments

asiteOffsets	A data frame of one column with read length as row names and a-sites for each read length in column.
insets	List of integer of length 4, insets in number of nucleotides to avoid start and stop cpdon.
bamfile	Character vector, name of a bam file (and path).
trx	GRange object for a transcript.
cds	IRange object for cds in the transcript.

Value

List of integer with length 4, number of aligned a-site in transcript, cds, 5-UTR, and 3-UTR regions.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

trxRegionCountSizes *Get Region Sized of A Transcript*

Description

Calculate sizes of whole transcript, cds, 5'UTR, and 3'UTR of a transcript defined in bed file.

Usage

```
trxRegionCountSizes(insets, trx, cds)
```

Arguments

insets	List of integers of length 2, insets in nucleotides to avoid start and stop positions.
trx	GRange object for transcripts.
cds	IRanges object for cds of the transcripts.

Value

List of integers pf length 4 for sizes of transcript, cds, 5-UTR, and 3-UTR..

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

`trxRegions` *Calculate Transcript Regions for UTR and CDS*

Description

Calculate length of whole transcript, 5'UTR, cds, and 3'UTR regions for a transcript.

Usage

```
trxRegions(trx, cds, insetUtr5Start=0, insetUtr5End=0, insetCdsStart=0,
insetCdsEnd=0, insetUtr3Start=0, insetUtr3End=0)
```

Arguments

`trx`	GRanges object for a transcript.
`cds`	GRanges object for cds in a transcript.
`insetUtr5Start`	Positive integer, adjustment after start position of 5'UTR.
`insetUtr5End`	Positive integer, adjustment before end position of 5'UTR .
`insetCdsStart`	Positive integer, adjustment after start position of cds.
`insetCdsEnd`	Positive integer, adjustment before end position of cds.
`insetUtr3Start`	Positive integer, adjustment after start position of 3'UTR.
`insetUtr3End`	Positive integer, adjustment before end position of 3'UTR .

Value

GRanges list representing cds, 5-UTR, and 3-UTR for a transcript.

References

Original code from Nicholas T. Ingolia, et al. (2014). Ribosome Profiling Reveals Pervasive Translation Outside of Annotated Protein-Coding Genes. Cell Reports 8, 1365-1379.

validateCodons *Check Out If the Given Codons Are Qualified Ones*

Description

Check out if each codon has correct length and correct bases.

Usage

```
validateCodons(codons)
```

Arguments

codons	character vector which holds one or more codons.

Details

Each codon must have exactly 3 bases from A, U, G, and C.

Value

None. Error message will be generated if any codon is a invalid one.

validateParameters *Validate All Parameters of FpFraming*

Description

Validate all user defined parameters for FpFraming.

Usage

```
validateParameters(parameters)
```

Arguments

parameters	parameters, list of numeric variables.

Value

None.

writeChromosomeSizesToFile *Write Chromosome Sizes to File*

Description

Read in chromosome names and sizes from or Write chromosome names and sizes to a tab-delimited text file.

Usage

```
writeChromosomeSizesToFile(chromSizes, file_name)
readChromosomeSizesFromFile(file_name)
```

Arguments

chromSizes	A data frame with 2 columns for chromosome names and lengths.
file_name	Character vector, name of file (and path) for read from or write to.

Value

readChromosomeSizesFromFile() returns a data frame with 2 columns for chromosome names and lengths.

writeORFInfoToFile *Save ORF Information to File*

Description

Write ORF information to tab-delimited text file including column headers.

Usage

```
writeORFInfoToFile(seq_info, start_codons, stop_codons, out_file)
```

Arguments

seq_info	Data frame, UTR/CDS info including chromosome, start and stop position, strand, sequence, and type (CDS, 5_UTR or 3_UTR).
start_codons	Character vector, start codons.
stop_codons	Character vector, stop codons.
out_file	Character vector, name (and path) of the file to write.

Details

The output file will include, for each ORF, the gene_name, chromosome, seq_start, seq_stop, strand, orf_id, orf_start, orf_stop, start_codon, context, orf_length, dis_from_cap, dis_to_MUG.

Value

None

writeWiggleFiles *Write Aligned A-site Counts to Wiggle Files*

Description

Write aligned a-site Counts on each chromosome to wiggle track files by strand.

Usage

```
writeWiggleFiles(all_counts, bam_file, is.normalized=FALSE)
```

Arguments

all_counts	List of list, each list element has two numeric vectors for read counts on forward and reverse strand at base pair level.
bam_file	character vector, name of bam file from which the wiggle track data is generated.
is.normalized	Logic, if the all_counts are normalized.

Value

None. Write files only

References

https://github.com/ingolia-lab/RiboSeq https://genome.ucsc.edu/goldenpath/help/wiggle.html

yeast_gene_description *Yeast Gene Description*

Description

A data Frame with three columns for gene ID, gene name, and gene description.

Usage

```
data("yeast_gene_description")
```

Format

A data frame with 7133 observations on the following 3 variables.

V1 a character vector

V2 a character vector

V3 a character vector

Examples

```
data(yeast_gene_description)
```

Index