Package 'InPAS'

November 12, 2026

Title Identify Novel Alternative PolyAdenylation Sites (PAS) from RNA-seq data

Version 2.20.0

Maintainer Jianhong Ou <jou@morgridge.org>

Description Alternative polyadenylation (APA) is one of the important post-transcriptional regulation mechanisms which occurs in most human genes. InPAS facilitates the discovery of novel APA sites and the differential usage of APA sites from RNA-Seq data. It leverages cleanUpdTSeq to fine tune identified APA sites by removing false sites.

biocViews Alternative Polyadenylation, Differential Polyadenylation Site Usage, RNA-seq, Gene Regulation, Transcription

License GPL (>= 2)

Imports AnnotationDbi,batchtools,Biobase,Biostrings,BSgenome,cleanUpdTSeq, depmixS4,dplyr,flock,future,future.apply,GenomeInfoDb,GenomicRanges, GenomicFeatures, ggplot2, IRanges, limma, magrittr,methods,parallelly, plyranges, preprocessCore, readr,reshape2, RSQLite, Seqinfo, stats, S4Vectors, utils

Depends R (>= 3.5)

Suggests BiocGenerics,BiocManager, BiocStyle, BSgenome.Mmusculus.UCSC.mm10, BSgenome.Hsapiens.UCSC.hg19, EnsDb.Hsapiens.v86, EnsDb.Mmusculus.v79, knitr, markdown, rmarkdown, rtracklayer, RUnit, grDevices, TxDb.Hsapiens.UCSC.hg19.knownGene,TxDb.Mmusculus.UCSC.mm10.knownGene

VignetteBuilder knitr

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

LazyData true

Encoding UTF-8

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

git_branch RELEASE_3_23

git_last_commit 596861b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Jianhong Ou [aut, cre],
Haibo Liu [aut],
Lihua Julie Zhu [aut],
Sungmi M. Park [aut],
Michael R. Green [aut]

Contents

.onAttach . . . 3
addChr2Exclude . . . 4
addInPASEnsDb . . . 4
addInPASGenome . . . 5
addInPASOutputDirectory . . . 5
addInPASTxDb . . . 6
addLockName . . . 6
adjust_distalCPs . . . 7
adjust_proximalCPs . . . 8
adjust_proximalCPsByNBC . . . 9
adjust_proximalCPsByPWM . . . 10
assemble_allCov . . . 11
assign_feature . . . 13
calculate_mse . . . 13
compensation . . . 14
extract_UTR3Anno . . . 15
fft.smooth . . . 17
filter_testOut . . . 17
find_minMSEDistr . . . 19
find_valleyBySpline . . . 20
gcComp . . . 21
gcContents . . . 22
getChr2Exclude . . . 22
getInPASEnsDb . . . 23
getInPASGenome . . . 23
getInPASOutputDirectory . . . 23
getInPASSQLiteDb . . . 24
getInPASTxDb . . . 24
getLockName . . . 25
get_chromosomes . . . 25
get_depthWeight . . . 26
get_lastCDSUTR3 . . . 27
get_PAscore . . . 28
get_PAscore2 . . . 29

get_regionCov . . . 30
get_seqLen . . . 31
get_ssRleCov . . . 32
get_totalCov . . . 34
get_usage4plot . . . 35
get_UTR3CDS . . . 37
get_UTR3eSet . . . 38
get_UTR3region . . . 40
get_UTR3TotalCov . . . 41
get_zScoreCutoff . . . 42
InPAS . . . 43
mapComp . . . 43
parse_TxDb . . . 44
plot_utr3Usage . . . 46
polish_CPs . . . 47
remove_convergentUTR3s . . . 47
run_coverageQC . . . 48
run_fisherExactTest . . . 51
run_limmaAnalysis . . . 52
run_singleGroupAnalysis . . . 53
run_singleSampleAnalysis . . . 53
search_CPs . . . 54
search_distalCPs . . . 59
search_proximalCPs . . . 60
setup_CPsSearch . . . 61
setup_GSEA . . . 64
setup_parCPsSearch . . . 65
setup_sqlitedb . . . 67
set_globals . . . 68
test_dPDUI . . . 69
trim_seqnames . . . 71
utr3.mm10 . . . 72
UTR3eSet-class . . . 72

Index **74**

`.onAttach` *A function called upon a package is attached to the search path*

Description

A function called upon a package is attached to the search path

Usage

```
.onAttach(libname, pkgname)
```

Arguments

libname	library name
pkgname	package name

---

addChr2Exclude *Add a globally-applied requirement for filtering out scaffolds from all analysis*

Description

This function will set the default requirement of filtering out scaffolds from all analysis.

Usage

```
addChr2Exclude(chr2exclude = c("chrM", "MT", "Pltd", "chrPltd"))
```

Arguments

chr2exclude	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. chrM and alternative scaffolds representing different haplotypes should be excluded.

---

addInPASEnsDb *Add a globally defined EnsDb to some InPAS functions.*

Description

Add a globally defined EnsDb to some InPAS functions.

Usage

```
addInPASEnsDb(EnsDb = NULL)
```

Arguments

EnsDb	An object of ensembldb::EnsDb

addInPASGenome *Add a globally defined genome to all InPAS functions.*

Description

This function will set the genome across all InPAS functions.

Usage

```
addInPASGenome(genome = NULL)
```

Arguments

genome	A BSgenome object indicating the default genome to be used for all InPAS functions. This value is stored as a global environment variable. This can be overwritten on a per-function basis using the given function's genome parameter.

addInPASOutputDirectory *Add a globally defined output directory to some InPAS functions.*

Description

Add a globally defined output directory to some InPAS functions.

Usage

```
addInPASOutputDirectory(outdir = NULL)
```

Arguments

outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.

addInPASTxDb *Add a globally defined TxDb for InPAS functions.*

Description

Add a globally defined TxDb for InPAS functions.

Usage

```
addInPASTxDb(TxDb = NULL)
```

Arguments

TxDb	An object of GenomicFeatures::TxDb

Examples

```
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
addInPASTxDb(TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene)
```

addLockName *Add a filename for locking a SQLite database*

Description

Add a filename for locking a SQLite database

Usage

```
addLockName(filename = NULL)
```

Arguments

filename	A character(1) vector, specifyong a path to a file for locking.

`adjust_distalCPs` *Adjust distal CP sites by the cleanUpdTSeq algorithm*

Description

Adjust distal CP sites by the cleanUpdTSeq algorithm

Usage

```
adjust_distalCPs(
  distalCPs,
  classifier,
  classifier_cutoff,
  shift_range,
  genome,
  seqname,
  step = 1
)
```

Arguments

`distalCPs`	the output of `search_distalCPs()`
`classifier`	An R object for Naive Bayes classifier model, like the one in the cleanUpdTSeq package.
`classifier_cutoff`	A numeric(1) vector. A cutoff of probability that a site is classified as true CP sites. The value should be between 0.5 and 1. Default, 0.8.
`shift_range`	An integer(1) vector, specifying a shift range for adjusting the proximal and distal CP sites. Default, 50. It determines the range flanking the candidate CP sites to search the most likely CP sites.
`genome`	a BSgenome::BSgenome object
`seqname`	A character(1) vector, specifying a chromososome/scaffold name
`step`	An integer (1) vector, specifying the step size used for adjusting the proximal or distal CP sites using the Naive Bayes classifier from the cleanUpdTSeq package. Default 1. It can be in the range of 1 to 5.

Author(s)

Jianhong Ou

See Also

`search_proximalCPs()`, `get_PAscore2()`

adjust_proximalCPs *Adjust the proximal CP sites*

Description

Adjust the proximal CP sites by PolyA PWM and cleanUpdTSeq. A few candidate sites, which are ranked by MSE from low to high, are used as input for adjusting. The final sites are the one with best score as PA sites, which are not necessary from the lowest MSE sites.

Usage

```
adjust_proximalCPs(
  CPs,
  PolyA_PWM,
  genome,
  classifier,
  classifier_cutoff,
  shift_range,
  search_point_START,
  step = 1,
  DIST2ANNOAPAP = 1000
)
```

Arguments

CPs	the outputs of search_proximalCPs()
PolyA_PWM	PolyA position weight matrix
genome	a BSgenome::BSgenome object
classifier	cleanUpdTSeq classifier
classifier_cutoff	cutoff value of the classifier
shift_range	the searching range for the better CP sites
search_point_START	just in case there is no better CP sites
step	An integer, specifying an adjusting step, default 1, means adjusting by each base by cleanUpdTSeq.
DIST2ANNOAPAP	An integer, specifying a cutoff for annotate MSE valleys with known proximal APAs in a given downstream distance. Default is 1500.

Value

keep same as search_proximalCPs(), which can be handled by polish_CPs().

Author(s)

Jianhong Ou

See Also

search_proximalCPs(), polish_CPs(), adjust_proximalCPsByPWM(), adjust_proximalCPsByNBC(), get_PAscore(),get_PAscore2()

---

adjust_proximalCPsByNBC *adjust the proximal CP sites by using Naive Bayes classifier from cleanUpdTSeq*

Description

adjust the proximal CP sites by using Naive Bayes classifier from cleanUpdTSeq

Usage

```
adjust_proximalCPsByNBC(
  idx.list,
  cov_diff.list,
  seqnames,
  starts,
  strands,
  genome,
  classifier,
  classifier_cutoff,
  shift_range,
  search_point_START,
  step = 1
)
```

Arguments

idx.list	the offset of positions of CP sites
cov_diff.list	the MSE values
seqnames	a character(n) vector, the chromosome/scaffolds' names
starts	starts
strands	strands
genome	a BSgenome::BSgenome object
classifier	cleanUpdTSeq classifier
classifier_cutoff	cutoff value of the classifier
shift_range	the searching range for the better CP sites
search_point_START	just in case there is no better CP sites
step	adjusting step, default 1, means adjust by each base by cleanUpdTSeq.

Details

the step for calculating is 10, can not do every base base it is really very slow.

Value

the offset of positions of CP sites after filter

Author(s)

Jianhong Ou

See Also

adjust_proximalCPsByPWM(), get_PAscore2()

`adjust_proximalCPsByPWM` *adjust the proximal CP sites by matching PWM*

Description

adjust the proximal CP sites by polyA Position Weight Matrix. It only need the PWM to get match in upstream or downstream shift_range nr.

Usage

```
adjust_proximalCPsByPWM(
  idx,
  PolyA_PWM,
  seqnames,
  starts,
  strands,
  genome,
  shift_range,
  search_point_START
)
```

Arguments

idx	the offset of positions of CP sites
PolyA_PWM	polyA PWM
seqnames	a character(n) vector, the chromosome/scaffolds' names
starts	start position in the genome
strands	strands
genome	an BSgenome::BSgenome object
shift_range	the shift range of PWM hits
search_point_START	Not use

Details

the hits is searched by `Biostrings::matchPWM()` and the cutoff is 70\

Value

the offset of positions of CP sites after filter

Author(s)

Jianhong Ou

See Also

`adjust_proximalCPsByNBC()`, `get_PAscore()`

---

assemble_allCov *Assemble coverage files for a given chromosome for all samples*

Description

Process individual sample-chromosome-specific coverage files in an experiment into a file containing a list of chromosome-specific Rle coverage of all samples

Usage

```
assemble_allCov(
  sqlite_db,
  seqname,
  outdir = getInPASOutputDirectory(),
  genome = getInPASGenome()
)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb()
`seqname`	A character(1) vector, the name of a chromosome/scaffold
`outdir`	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
`genome`	An object of BSgenome::BSgenome

Value

A list of paths to per-chromosome coverage files of all samples.

- seqname, chromosome/scaffold name
  - tag1, name tag for sample1
  - tag2, name tag for sample2
  - tagN, name tag for sampleN

Author(s)

Haibo Liu

Examples

```
if (interactive()) {
  library(BSgenome.Mmusculus.UCSC.mm10)
  genome <- BSgenome.Mmusculus.UCSC.mm10
  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )

  sqlite_db <- setup_sqlitedb(
    metadata = file.path(
      outdir,
      "metadata.txt"
    ),
    outdir
  )
  coverage <- list()
  addLockName(filename = tempfile())
  for (i in seq_along(bedgraphs)) {
    coverage[[tags[i]]] <- get_ssRleCov(
      bedgraph = bedgraphs[i],
      tag = tags[i],
      genome = genome,
      sqlite_db = sqlite_db,
      outdir = outdir,
      chr2exclude = "chrM"
    )
  }
  chr_coverage <- assemble_allCov(sqlite_db,
    seqname = "chr6",
    outdir = outdir,
    genome = genome
  )
}
```

assign_feature *Helper function to label the last component of a genomic feature for each transcript*

Description

Helper function to label the last component of a genomic feature for each transcript

Usage

```
assign_feature(gr, feature_alt = "utr3")
```

Arguments

gr	A tibble converted from an object of GenomicRanges::GRanges
feature_alt	A character(1) vector, specifying the type of genomic features, such as "CDS", "exon", "utr3", "utr5".

Value

An object of GenomicRanges::GRanges

Author(s)

Haibo Liu

calculate_mse *Calculate mean squared errors (MSE)*

Description

Calculate mean squared errors (MSE) for each searched site which is assumed bisection site (i.e. potential CP site).

Usage

```
calculate_mse(.ele, search_point_START, search_point_END)
```

Arguments

.ele	A numeric vector, storing 3' UTR coverage for a give sample or collapsed 3' UTR coverage for a given condition
search_point_START	An integer, specifying the start position to calculate MSE
search_point_END	An integer, specifying end position to calculate MSE

Value

a vector of numeric, containing mean squared errors for each searched site when which is assumed as a bisection site (i.e. potential CP site).

Author(s)

Jianhong Ou, Haibo Liu

`compensation` *Compensate the coverage with GC-content or mappability*

Description

Compensate the coverage with GC-content or mappability

Usage

```
compensation(view, comp, start, end)
```

Arguments

`view`	A list of view object
`comp`	A numeric vector of weight for GC composition or mappability
`start`	An integer vector, starting coordinates
`end`	An integer vector, end coordinates

Value

a list of GC composition or mappability corrected coverage

Author(s)

Jianhong Ou

`extract_UTR3Anno` *extract 3' UTR information from a GenomicFeatures::TxDb object*

Description

extract 3' UTR information from a GenomicFeatures::TxDb object. The 3'UTR is defined as the last 3'UTR fragment for each transcript and it will be cut if there is any overlaps with other exons.

Usage

```
extract_UTR3Anno(
  sqlite_db,
  TxDb = getInPASTxDb(),
  edb = getInPASEnsDb(),
  genome = getInPASGenome(),
  outdir = getInPASOutputDirectory(),
  chr2exclude = getChr2Exclude(),
  MAX_EXONS_GAP = 10000L
)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`TxDb`	An object of GenomicFeatures::TxDb
`edb`	An object of ensembldb::EnsDb
`genome`	An object of BSgenome::BSgenome
`outdir`	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
`chr2exclude`	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.
`MAX_EXONS_GAP`	An integer(1) vector, maximal gap sizes between the last known CP sites to a nearest downstream exon. Default is 10 kb for mammalian genomes. For other species, user need to adjust this parameter.

Details

A good practice is to perform read alignment using a reference genome from Ensembl/GenCode including only the primary assembly and build a TxDb and EnsDb using the GTF/GFF files downloaded from the same source as the reference genome, such as BioMart/Ensembl/GenCode. For instruction, see Vignette of the GenomicFeatures. The UCSC reference genomes and their annotation packages can be very cumbersome.

Value

An object of GenomicRanges::GRangesList, containing GRanges for extracted 3' UTRs, and the corresponding last CDSs and next.exon.gap for each chromosome/scaffold. Chromosome

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
library("EnsDb.Hsapiens.v86")
library("BSgenome.Hsapiens.UCSC.hg19")
library("GenomicFeatures")
## set a sqlite database
bedgraphs <- system.file("extdata", c(
  "Baf3.extract.bedgraph",
  "UM15.extract.bedgraph"
),
package = "InPAS"
)
tags <- c("Baf3", "UM15")
metadata <- data.frame(
  tag = tags,
  condition = c("Baf3", "UM15"),
  bedgraph_file = bedgraphs
)
outdir <- tempdir()

write.table(metadata,
  file = file.path(outdir, "metadata.txt"),
  sep = "\t", quote = FALSE, row.names = FALSE
)
sqlite_db <- setup_sqlitedb(
  metadata =
    file.path(outdir, "metadata.txt"),
  outdir
)

samplefile <- system.file("extdata",
  "hg19_knownGene_sample.sqlite",
  package = "GenomicFeatures"
)
TxDb <- loadDb(samplefile)
edb <- EnsDb.Hsapiens.v86
genome <- BSgenome.Hsapiens.UCSC.hg19
addInPASOutputDirectory(outdir)
seqnames <- seqnames(BSgenome.Hsapiens.UCSC.hg19)
chr2exclude <- c(
  "chrM", "chrMT",
  seqnames[grepl("_(hap\\d+|fix|alt)$",
    seqnames,
    perl = TRUE
  )]
)
utr3 <- extract_UTR3Anno(sqlite_db, TxDb, edb,
  genome = genome,
  chr2exclude = chr2exclude,
```

```
  outdir = tempdir(),
  MAX_EXONS_GAP = 10000L
)
```

`fft.smooth` *Smoothing using Fast Discrete Fourier Transform*

Description

Smoothing using Fast Discrete Fourier Transform

Usage

```
fft.smooth(sn, p)
```

Arguments

sn	a real or complex array containing the values to be transformed. see `stats::fft()`
p	An integer(1), fft smoothing power

Value

a numeric vector, the real part of inverse fft-transformed signal

Author(s)

Jianhong Ou

`filter_testOut` *filter 3' UTR usage test results*

Description

filter results of `test_dPDUI()`

Usage

```
filter_testOut(
  res,
  gp1,
  gp2,
  outdir = getInPASOutputDirectory(),
  background_coverage_threshold = 2,
  P.Value_cutoff = 0.05,
  adj.P.Val_cutoff = 0.05,
  dPDUI_cutoff = 0.2,
  PDUI_logFC_cutoff = log2(1.5)
)
```

Arguments

res	a UTR3eSet object, output of test_dPDUI()
gp1	tag names involved in group 1. gp1 and gp2 are used for filtering purpose if both are specified; otherwise only other specified thresholds are used for filtering.
gp2	tag names involved in group 2
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
background_coverage_threshold	background coverage cut off value. for each group, more than half of the long form should greater than background_coverage_threshold. for both group, at least in one group, more than half of the short form should greater than background_coverage_threshold.
P.Value_cutoff	cutoff of P value
adj.P.Val_cutoff	cutoff of adjust P value
dPDUI_cutoff	cutoff of dPDUI
PDUI_logFC_cutoff	cutoff of PDUI log2 transformed fold change

Value

A data frame converted from an object of GenomicRanges::GRanges.

Author(s)

Jianhong Ou, Haibo Liu

See Also

test_dPDUI()

Examples

```
library(limma)
path <- system.file("extdata", package = "InPAS")
load(file.path(path, "eset.MAQC.rda"))
tags <- colnames(eset@PDUI)
g <- factor(gsub("\\..*$", "", tags))
design <- model.matrix(~ -1 + g)
colnames(design) <- c("Brain", "UHR")
contrast.matrix <- makeContrasts(
  contrasts = "Brain-UHR",
  levels = design
)
res <- test_dPDUI(
  eset = eset,
  method = "limma",
  normalize = "none",
```

```
  design = design,
  contrast.matrix = contrast.matrix
)
filter_testOut(res,
  gp1 = c("Brain.auto", "Brain.phiX"),
  gp2 = c("UHR.auto", "UHR.phiX"),
  background_coverage_threshold = 2,
  P.Value_cutoff = 0.05,
  adj.P.Val_cutoff = 0.05,
  dPDUI_cutoff = 0.3,
  PDUI_logFC_cutoff = .59
)
```

`find_minMSEDistr` *Visualization of MSE profiles, 3' UTR coverage and minimal MSE distribution*

Description

Visualization of MSE profiles, 3' UTR coverage and minimal MSE distribution

Usage

```
find_minMSEDistr(
  CPs,
  outdir = NULL,
  MSE.plot = "MSE.pdf",
  coverage.plot = "coverage.pdf",
  min.MSE.to.end.distr.plot = "min.MSE.to.end.distr.pdf"
)
```

Arguments

`CPs`	A list, output from `search_proximalCPs()` or `adjust_distalCPs()` or `adjust_proximalCPs()`
`outdir`	A character(1) vector, specifying the output directory
`MSE.plot`	A character(1) vector, specifying a PDF file name for outputting plots of MSE profiles. No directory path is allowed.
`coverage.plot`	A character(1) vector, specifying a PDF file name for outputting per-sample coverage profiles. No directory path is allowed.
`min.MSE.to.end.distr.plot`	A character(1) vector, specifying a PDF file name for outputting histograms showing minimal MSE distribution relative to longer 3' UTR end. No directory path is allowed.

`find_valleyBySpline` *Find major valleys after spline smoothing*

Description

Find major valleys after spline smoothing

Usage

```
find_valleyBySpline(
  x,
  ss,
  se = length(x),
  nknots = ceiling((se - ss + 1)/1000 * 10),
  n = -1,
  min.dist = 200,
  filter.last = TRUE,
  DIST2END = 1200,
  plot = FALSE
)
```

Arguments

`x`	A vector of numeric(n), containing MSEs for a given range
`ss`	An positive integer, search start site relative to the leftmost base
`se`	An positive integer, search end site relative to the leftmost base
`nknots`	An positive integer, the number of knots for smoothing using spline`stats::smooth.spline()`. By default, set to 10 knots per kb.
`n`	An integer, specifying the number of location where MSE are local minima (candidate CP sites). If set to -1, return all candidate CP sites.
`min.dist`	An integer, minimal distance allowed between two adjacent candidate CP sites otherwise collapsed by selecting the one with lower MSE.
`filter.last`	A logical(1), whether to filter out the last valley, which is likely the 3' end of the longer 3' UTR if no novel distal CP site is detected and the 3' end excluded by setting cutEnd/search_point_END is small.
`DIST2END`	An integer, specifying a cutoff of the distance between last valley and the end of the 3' UTR (where MSE of the last base is calculated). If the last valley is closer to the end than the specified distance, it will be not be considered because it is very likely due to RNA coverage decay at the end of mRNA. Default is 1200. User can consider a value between 1000 and 1500, depending on the library preparation procedures: RNA fragmentation and size selection.
`plot`	A logical(1), whether to plot the MSE profile and the candidate valleys.

Value

A vector of integer.

gcComp *Calculate weights for GC composition*

Description

Calculate read weights for GC composition-based coverage correction

Usage

```
gcComp(genome, seqnames, window = 50, future.chunk.size = NULL)
```

Arguments

genome	An object of BSgenome::BSgenome
seqnames	a character(n) vector, the chromosome/scaffolds' names in the same forms of seqnames in the BSgenome
window	size of a sliding window, which optimally is set to the read length
future.chunk.size	The average number of elements per future ("chunk"). If Inf, then all elements are processed in a single future. If NULL, then argument future.scheduling = 1 is used by default. Users can set future.chunk.size = total number of elements/number of cores set for the backend. See the future.apply package for details.

Value

A list of numeric vectors containing the weight (scaffold-level GC\ / GC\ chromosome/scaffold.

Author(s)

Jianhong Ou, Haibo Liu

References

Cheung et al. Systematic bias in high-throughput sequencing data and its correction by BEADS. Nucleic Acids Res. 2011 Aug;39(15):e103.

Examples

```
## Not run:
library(BSgenome.Mmusculus.UCSC.mm10)
genome <- BSgenome.Mmusculus.UCSC.mm10
InPAS:::gcComp(genome, "chr1")

## End(Not run)
```

gcContents *helper function to calculate chromosome/scaffold level GC content*

Description

helper function to calculate chromosome/scaffold level GC content

Usage

```
gcContents(genome, seqname, nonATCGExclude = TRUE)
```

Arguments

genome	an object of BSgenome::BSgenome
seqname	a character(1) vector, the chromosome/scaffold's name
nonATCGExclude	a logical(1) vector, whether nucleotides other than A, T, C, and G should be excluded when GC content is calculated

Value

a numeric(1) vector, containing the chromosome/scaffold -specific GC content in the range of 0 to 1

Author(s)

Haibo Liu

Examples

```
## Not run:
library(BSgenome.Mmusculus.UCSC.mm10)
genome <- BSgenome.Mmusculus.UCSC.mm10
InPAS:::gcContents(genome, "chr1")

## End(Not run)
```

getChr2Exclude *Get a globally-applied requirement for filtering scaffolds.*

Description

This function will get the default requirement of filtering scaffolds.

Usage

```
getChr2Exclude()
```

getInPASEnsDb *Get the globally defined EnsDb.*

Description

Get the globally defined EnsDb.

Usage

```
getInPASEnsDb()
```

Value

An object of ensembldb::EnsDb

getInPASGenome *Get the globally defined genome*

Description

This function will retrieve the genome that is currently in use by InPAS.

Usage

```
getInPASGenome()
```

getInPASOutputDirectory *Get the path to a output directory for InPAS analysis*

Description

Get the path to a output directory for InPAS analysis

Usage

```
getInPASOutputDirectory()
```

Value

a normalized path to a output directory for InPAS analysis

getInPASSQLiteDb *Get the path to an SQLite database*

Description

Get the path to an SQLite database

Usage

```
getInPASSQLiteDb()
```

Value

A path to an SQLite database

getInPASTxDb *Get the globally defined TxDb.*

Description

Get the globally defined TxDb.

Usage

```
getInPASTxDb()
```

Value

An object of GenomicFeatures::TxDb

Examples

```
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
addInPASTxDb(TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene)
getInPASTxDb()
```

getLockName *Get the path to a file for locking the SQLite database*

Description

Get the path to a file for locking the SQLite database

Usage

```
getLockName()
```

Value

A path to a file for locking

get_chromosomes *Identify chromosomes/scaffolds for CP site discovery*

Description

Identify chromosomes/scaffolds which have both coverage and annotated 3' utr3 for CP site discovery

Usage

```
get_chromosomes(utr3, sqlite_db)
```

Arguments

utr3	An object of GenomicRanges::GRangesList. An output of `extract_UTR3Anno()`.
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.

Value

A vector of characters, containing names of chromosomes/scaffolds for CP site discovery

Examples

```
library(BSgenome.Mmusculus.UCSC.mm10)
genome <- BSgenome.Mmusculus.UCSC.mm10
data(utr3.mm10)
utr3 <- split(utr3.mm10, seqnames(utr3.mm10), drop = TRUE)
bedgraphs <- system.file("extdata", c(
  "Baf3.extract.bedgraph",
  "UM15.extract.bedgraph"
),
```

```
package = "InPAS"
)
tags <- c("Baf3", "UM15")
metadata <- data.frame(
  tag = tags,
  condition = c("Baf3", "UM15"),
  bedgraph_file = bedgraphs
)
outdir <- tempdir()
write.table(metadata,
  file = file.path(outdir, "metadata.txt"),
  sep = "\t", quote = FALSE, row.names = FALSE
)

sqlite_db <- setup_sqlitedb(
  metadata = file.path(
    outdir,
    "metadata.txt"
  ),
  outdir
)
addLockName(filename = tempfile())
coverage <- list()
for (i in seq_along(bedgraphs)) {
      coverage[[tags[i]]] <- get_ssRleCov(
      bedgraph = bedgraphs[i],
      tag = tags[i],
      genome = genome,
      sqlite_db = sqlite_db,
      outdir = outdir,
      chr2exclude = "chrM"
    )
}
get_chromosomes(utr3, sqlite_db)
```

`get_depthWeight` *Calculate the depth weight for each sample or each experimental condition*

Description

Calculate the depth weight for each sample of non-hugeData or each experimental condition for hugeData: depth/mean(depth)

Usage

```
get_depthWeight(metadata, hugeData)
```

Arguments

metadata	A data frame containing the metadata for a RNA-seq experiment, which can be extract from the SQLite database set up by setup_sqlitedb()
hugeData	A logical(1), indicating whether it is huge data

Value

A named numeric vector containing depth weight for each sample for non-hugeData, or depth weight for each condition if hugeData.

Author(s)

Jianhong Ou, Haibo Liu

---

get_lastCDSUTR3 *Extract the last unspliced region of each transcript*

Description

Extract the last unspliced region of each transcript from a TxDb. These regions could be the last 3'UTR exon for transcripts whose 3' UTRs are composed of multiple exons or last CDS regions and 3'UTRs for transcripts whose 3'UTRs and last CDS regions are on the same single exon.

Usage

```
get_lastCDSUTR3(
  TxDb = getInPASTxDb(),
  genome = getInPASGenome(),
  chr2exclude = getChr2Exclude(),
  outdir = getInPASOutputDirectory(),
  MAX_EXONS_GAP = 10000
)
```

Arguments

TxDb	An object of GenomicFeatures::TxDb
genome	An object of BSgenome::BSgenome
chr2exclude	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. chrM and alternative scaffolds representing different haplotypes should be excluded.
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
MAX_EXONS_GAP	An integer(1) vector, maximal gap sizes between the last known CP sites to a nearest downstream exon. Default is 10 kb for mammalian genomes. For other species, user need to adjust this parameter.

Value

A BED file with 6 columns: chr, chrStart, chrEnd, name, score, and strand.

`get_PAscore` *Calculate the CP score*

Description

Calculate the CP score by using PWM of polyadenylation signal with sequence around given position

Usage

```
get_PAscore(seqname, pos, str, idx, PWM, genome, ups = 50, dws = 50)
```

Arguments

`seqname`	a character(n) vector, the chromosome/scaffold' name
`pos`	genomic positions
`str`	DNA strand
`idx`	offset position
`PWM`	An R object for a position weight matrix (PWM) for a hexamer polyadenylation signal (PAS), such as AAUAAA.
`genome`	an object of BSgenome::BSgenome
`ups`	the number of upstream bases for PAS search.
`dws`	the number of downstream bases for PAS search.

Value

A list containing offset positions after PA score-based filtering

Author(s)

Jianhong Ou

See Also

`get_PAscore2()`

get_PAscore2 *calculate the CP score*

Description

calculate CP score by cleanUpdTSeq

Usage

```
get_PAscore2(
  seqname,
  pos,
  str,
  idx,
  idx.gp,
  genome,
  classifier,
  classifier_cutoff
)
```

Arguments

seqname	a character(1) vector, the chromosome/scaffold's name
pos	genomic positions
str	DNA strand
idx	offset position
idx.gp	group number of the offset position
genome	an object of BSgenome::BSgenome
classifier	An R object for Naive Bayes classifier model, like the one in the cleanUpdTSeq package.
classifier_cutoff	A numeric(1) vector. A cutoff of probability that a site is classified as true CP sites. The value should be between 0.5 and 1. Default, 0.8.

Value

a data frame or NULL

Author(s)

Jianhong Ou, Haibo Liu

See Also

get_PAscore()

get_regionCov *Get coverage for 3' UTR and last CDS regions on a single chromosome*

Description

Get coverage for 3' UTR and last CDS regions on a single chromosome

Usage

```
get_regionCov(
  chr.utr3,
  sqlite_db,
  outdir = getInPASOutputDirectory(),
  phmm = FALSE,
  min.length.diff = 200
)
```

Arguments

chr.utr3	An object of GenomicRanges::GRanges, one element of an output of `extract_UTR3Anno()`
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
phmm	A logical(1) vector, indicating whether data should be prepared for singleSample analysis? By default, FALSE
min.length.diff	An integer(1) vector, specifying minimal length difference between proximal and distal APA sites which should be met to be considered for differential APA analysis. Default is 200 bp.

Value

coverage view in GRanges

Author(s)

Jianhong Ou, Haibo Liu

`get_seqLen` *Get sequence lengths for chromosomes/scaffolds*

Description

Get sequence lengths for chromosomes/scaffolds from a BSgenome::BSgenome object

Usage

```
get_seqLen(genome = getInPASGenome(), chr2exclude = getChr2Exclude())
```

Arguments

`genome`	An object of BSgenome::BSgenome
`chr2exclude`	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.

Value

A named numeric vector containing lengths per seqname, with the seqnames as the names

Author(s)

Jianhong Ou, Haibo Liu

See Also

Seqinfo::Seqinfo

Examples

```
library(BSgenome.Mmusculus.UCSC.mm10)
genome <- BSgenome.Mmusculus.UCSC.mm10
InPAS:::get_seqLen(
  genome = genome,
  chr2exclude = "chrM"
)
```

get_ssRleCov *Get Rle coverage from a bedgraph file for a sample*

Description

Get RLe coverage from a bedgraph file for a sample

Usage

```
get_ssRleCov(
  bedgraph,
  tag,
  genome = getInPASGenome(),
  sqlite_db,
  future.chunk.size = NULL,
  outdir = getInPASOutputDirectory(),
  chr2exclude = getChr2Exclude()
)
```

Arguments

bedgraph	A path to a bedGraph file
tag	A character(1) vector, a name tag used to label the bedgraph file. It must match the tag specified in the metadata file used to setup the SQLite database
genome	an object BSgenome::BSgenome. To make things easy, we suggest users creating a BSgenome::BSgenome instance from the reference genome used for read alignment. For details, see the documentation of `BSgenome::forgeBSgenomeDataPkg()`.
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
future.chunk.size	The average number of elements per future ("chunk"). If Inf, then all elements are processed in a single future. If NULL, then argument future.scheduling = 1 is used by default. Users can set future.chunk.size = total number of elements/number of cores set for the backend. See the future.apply package for details. You may adjust this number based based on the available computing resource: CPUs and RAM. This parameter affects the time for converting coverage from bedgraph to Rle.
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
chr2exclude	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.

Value

A data frame, as described below.

tag the sample tag

chr chromosome name

coverage_file path to Rle coverage files for each chromosome per sample tag

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
if (interactive()) {
  library(BSgenome.Mmusculus.UCSC.mm10)
  genome <- BSgenome.Mmusculus.UCSC.mm10
  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )

  sqlite_db <- setup_sqlitedb(
    metadata = file.path(
      outdir,
      "metadata.txt"
    ),
    outdir
  )
  addLockName()
  coverage_info <- get_ssRleCov(
    bedgraph = bedgraphs[1],
    tag = tags[1],
    genome = genome,
    sqlite_db = sqlite_db,
    outdir = outdir,
    chr2exclude = "chrM"
  )
  # check read coverage depth
```

```
  db_connect <- dbConnect(drv = RSQLite::SQLite(), dbname = sqlite_db)
  dbReadTable(db_connect, "metadata")
  dbDisconnect(db_connect)
}
```

`get_totalCov` *Calculate the total coverage*

Description

For hugeData, coverage of samples in each condition is merged chromosome by chromosome. For non-hugeData, per-chromosome coverage of all samples

Usage

```
get_totalCov(sqlite_db, chr.cov, seqname, metadata, outdir, hugeData)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`chr.cov`	A list of Rle objects storing coverage per sample for a given chromosome/scaffold
`seqname`	A character(1), the chromosome/scaffold name
`metadata`	A data frame containing the metadata for a RNA-seq experiment, which can be extract from the SQLite database set up by `setup_sqlitedb()`
`outdir`	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
`hugeData`	A logical(1), indicating whether it is huge data

Value

A list containing pooled coverage data. For hugeData, coverage of samples under each condition is merged chromosome by chromosome. For non-hugeData, per-chromosome coverage of all samples are returned.

seqname chromosome/scaffold name

condition1 condition name 1

condition1 condition name 2

Author(s)

Haibo Liu, Jianhong Ou

get_usage4plot *prepare coverage data and fitting data for plot*

Description

prepare coverage data and fitting data for plot

Usage

```
get_usage4plot(gr, proximalSites, sqlite_db, hugeData)
```

Arguments

gr	An object of GenomicRanges::GRanges
proximalSites	An integer(n) vector, specifying the coordinates of proximal CP sites. Each of the proximal sites must match one entry in the GRanges object, gr.
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
hugeData	A logical(1), indicating whether it is huge data

Value

An object of GenomicRanges::GRanges with metadata:

dat	A data.frame, first column is the position, the other columns are Coverage and value
offset	offset from the start of 3' UTR

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
library(BSgenome.Mmusculus.UCSC.mm10)
library(TxDb.Mmusculus.UCSC.mm10.knownGene)
genome <- BSgenome.Mmusculus.UCSC.mm10
TxDb <- TxDb.Mmusculus.UCSC.mm10.knownGene

## load UTR3 annotation and convert it into a GRangesList
data(utr3.mm10)
utr3 <- split(utr3.mm10, seqnames(utr3.mm10), drop = TRUE)

bedgraphs <- system.file("extdata", c(
  "Baf3.extract.bedgraph",
  "UM15.extract.bedgraph"
),
package = "InPAS"
)
```

```
tags <- c("Baf3", "UM15")
metadata <- data.frame(
  tag = tags,
  condition = c("baf", "UM15"),
  bedgraph_file = bedgraphs
)
outdir <- tempdir()
write.table(metadata,
  file = file.path(outdir, "metadata.txt"),
  sep = "\t", quote = FALSE, row.names = FALSE
)

sqlite_db <- setup_sqlitedb(
  metadata = file.path(
    outdir,
    "metadata.txt"
  ),
  outdir
)
addLockName(filename = tempfile())
coverage <- list()
for (i in seq_along(bedgraphs)) {
  coverage[[tags[i]]] <- get_ssRleCov(
    bedgraph = bedgraphs[i],
    tag = tags[i],
    genome = genome,
    sqlite_db = sqlite_db,
    outdir = outdir,
    chr2exclude = "chrM"
  )
}
data4CPsSearch <- setup_CPsSearch(sqlite_db,
  genome,
  chr.utr3 = utr3[["chr6"]],
  seqname = "chr6",
  background = "10K",
  TxDb = TxDb,
  hugeData = TRUE,
  outdir = outdir
)

gr <- GRanges("chr6", IRanges(128846245, 128850081), strand = "-")
names(gr) <- "chr6:128846245-128850081"
data4plot <- get_usage4plot(gr,
  proximalSites = 128849148,
  sqlite_db,
  hugeData = TRUE
)
plot_utr3Usage(
  usage_data = data4plot,
  vline_color = "purple",
  vline_type = "dashed"
)
```

get_UTR3CDS *Get 3' UTRs and their last CDS regions based on CP sites*

Description

Get 3' UTRs and their last CDS regions based on CP sites

Usage

```
get_UTR3CDS(
  sqlite_db,
  chr.utr3,
  outdir = getInPASOutputDirectory(),
  min.length.diff = 200
)
```

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
chr.utr3	An object of GenomicRanges::GRanges, specifying UTR3 GRanges for a chromosome. It must be one element of an output of extract_UTR3Anno().
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
min.length.diff	An integer(1) vector, specifying minimal length difference between proximal and distal APA sites which should be met to be considered for differential APA analysis. Default is 200 bp.

Value

An object of GenomicRanges::GRanges containing GRanges for UTRs with alternative CP sites and the corresponding last CDSs.

Author(s)

Jianhong Ou, Haibo Liu

get_UTR3eSet *prepare 3' UTR coverage data for usage test*

Description

generate a UTR3eSet object with PDUI information for statistic tests

Usage

```
get_UTR3eSet(
  sqlite_db,
  normalize = c("none", "quantiles", "quantiles.robust", "mean", "median"),
  ...,
  singleSample = FALSE
)
```

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
normalize	A character(1) vector, spcifying the normalization method. It can be "none", "quantiles", "quantiles.robust", "mean", or "median"
...	parameter can be passed into preprocessCore::normalize.quantiles.robust()
singleSample	A logical(1) vector, indicating whether data is prepared for analysis in a single-Sample mode? Default, FALSE

Value

An object of UTR3eSet which contains following elements: usage: an GenomicRanges::GRanges object with CP sites info. PDUI: a matrix of PDUI PDUI.log2: log2 transformed PDUI matrix short: a matrix of usage of short form long: a matrix of usage of long form if singleSample is TRUE, one more element, signals, will be included.

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
if (interactive()) {
  library(BSgenome.Mmusculus.UCSC.mm10)
  library(TxDb.Mmusculus.UCSC.mm10.knownGene)
  genome <- BSgenome.Mmusculus.UCSC.mm10
  TxDb <- TxDb.Mmusculus.UCSC.mm10.knownGene

  ## load UTR3 annotation and convert it into a GRangesList
  data(utr3.mm10)
  utr3 <- split(utr3.mm10, seqnames(utr3.mm10), drop = TRUE)
```

```
bedgraphs <- system.file("extdata", c(
  "Baf3.extract.bedgraph",
  "UM15.extract.bedgraph"
),
package = "InPAS"
)
tags <- c("Baf3", "UM15")
metadata <- data.frame(
  tag = tags,
  condition = c("Baf3", "UM15"),
  bedgraph_file = bedgraphs
)
outdir <- tempdir()
write.table(metadata,
  file = file.path(outdir, "metadata.txt"),
  sep = "\t", quote = FALSE, row.names = FALSE
)

sqlite_db <- setup_sqlitedb(metadata = file.path(
  outdir,
  "metadata.txt"
), outdir)
addLockName(filename = tempfile())
coverage <- list()
for (i in seq_along(bedgraphs)) {
  coverage[[tags[i]]] <- get_ssRleCov(
    bedgraph = bedgraphs[i],
    tag = tags[i],
    genome = genome,
    sqlite_db = sqlite_db,
    outdir = outdir,
    chr2exclude = "chrM"
  )
}

data4CPsSearch <- setup_CPsSearch(sqlite_db,
  genome,
  chr.utr3 = utr3[["chr6"]],
  seqname = "chr6",
  background = "10K",
  TxDb = TxDb,
  hugeData = TRUE,
  outdir = outdir,
  minZ = 2,
  cutStart = 10,
  MINSIZE = 10,
  coverage_threshold = 5
)
## polyA_PWM
load(system.file("extdata", "polyA.rda", package = "InPAS"))

## load the Naive Bayes classifier model from the cleanUpdTSeq package
```

```
  library(cleanUpdTSeq)
  data(classifier)

  CPs <- search_CPs(
    seqname = "chr6",
    sqlite_db = sqlite_db,
    genome = genome,
    MINSIZE = 10,
    window_size = 100,
    search_point_START = 50,
    search_point_END = NA,
    cutEnd = 0,
    adjust_distal_polyA_end = TRUE,
    long_coverage_threshold = 2,
    PolyA_PWM = pwm,
    classifier = classifier,
    classifier_cutoff = 0.8,
    shift_range = 100,
    step = 5,
    outdir = outdir
  )
  utr3_cds_cov <- get_regionCov(
    chr.utr3 = utr3[["chr6"]],
    sqlite_db,
    outdir,
    phmm = FALSE
  )
  eSet <- get_UTR3eSet(sqlite_db,
    normalize = "none",
    singleSample = FALSE
  )
  test_out <- test_dPDUI(
    eset = eSet,
    method = "fisher.exact",
    normalize = "none",
    sqlite_db = sqlite_db
  )
}
```

`get_UTR3region` *extract long and short 3UTR region*

Description

extract long and short 3UTR region

Usage

```
get_UTR3region(.grs)
```

Arguments

.grs	output of search_CPs()

Value

A GenomicRanges::GRanges object with short form and long 3' UTR forms

Author(s)

Jianhong Ou

---

get_UTR3TotalCov *extract coverage of 3' UTR for CP sites prediction*

Description

extract 3' UTR coverage from totalCov according to the GenomicRanges::GRanges object utr3.

Usage

```
get_UTR3TotalCov(
  chr.utr3,
  chr.totalCov,
  gcCompensation = NA,
  mappabilityCompensation = NA,
  FFT = FALSE,
  fft.sm.power = 20
)
```

Arguments

chr.utr3	An object of GenomicRanges::GRanges. It must be an element of the output of extract_UTR3Anno() for a given chromosome.
chr.totalCov	total coverage for each condition of a given chromosome. It must be an output of get_totalCov()
mappabilityCompensation	mappability compensation vector. Not support yet.
FFT	Use FFT smooth or not.
fft.sm.power	the cut-off frequency of FFT smooth.
gcCompensationensation	GC compensation vector. Not support yet.

Value

path to a file storing the UTR3 total coverage for a given chromosome/scaffold

Author(s)

Jianhong Ou

`get_zScoreCutoff` *Calculate local background cutoff value*

Description

calculate local background z-score cutoff

Usage

```
get_zScoreCutoff(
  background,
  chr.introns,
  chr.totalCov,
  chr.utr3,
  seqname,
  z = 2
)
```

Arguments

`background`	A character(1) vector, indicating how background coverage is defined.
`chr.introns`	An object of GenomicRanges::GRanges for introns of a give chromosome/scaffold
`chr.totalCov`	total coverage for a given chromosome/scaffold, an output from `get_totalCov()` for a given chromosome/scaffold
`chr.utr3`	An object of GenomicRanges::GRanges, an element of the output of `extract_UTR3Anno()` for a given chromosome/scaffold
`seqname`	A character(1), the name of a chromosome/scaffold
`z`	Z score cutoff value

Value

A named numeric vector containing local background Z-score cutoff values. The names are GRanges's name for 3' UTRs.

Author(s)

Jianhong Ou, Haibo Liu

InPAS — *A package for identifying novel Alternative PolyAdenylation Sites (PAS) based on RNA-seq data*

Description

The InPAS package provides three categories of important functions: parse_TxDb, extract_UTR3Anno, get_ssRleCov, assemble_allCov, get_UTR3eSet, test_dPDUI, run_singleSampleAnalysis, run_singleGroupAnalysis, run_limmaAnalysis, filter_testOut, get_usage4plot, setup_GSEA, run_coverageQC

functions for retrieving 3' UTR annotation

parse_TxDb, extract_UTR3Anno, get_lastCDSUTR3

functions for processing read coverage data

assemble_allCov, get_ssRleCov, run_coverageQC, setup_parCPsSearch

functions for alternative polyadenylation site analysis

test_dPDUI, run_singleSampleAnalysis, run_singleGroupAnalysis, run_limmaAnalysis, filter_testOut, get_usage4plot

mapComp — *Calculate weights for mappability-base coverage correction*

Description

mappability is calculated by using GEM with the following command lines: PATH=$PATH:~/bin/GEM-binaries-Linux-x86_64-core_i3-20130406-045632/bin ./gem-indexer -i genome.fa -o mm10.index.gem ./gem-mappability -I mm10.index.gem.gem -l 100 -o mm10.mappability ./gem-2-wig -I mm10.index.gem.gem -i mm10.mappability -o mm10.mappability.wig

Usage

```
mapComp(mi)
```

Arguments

mi	A numeric vector of mappability along per chromosome/scaffold

Details

Calculate weights for mappability-base coverage correction

Value

A numeric vector of weights for mappability-based coverage correction

Author(s)

Jianhong Ou

References

Derrien et al. Fast computation and applications of genome mappability. PLoS One. 2012;7(1):e30377. doi: 10.1371/journal.pone.0030377.

`parse_TxDb` *Extract gene models from a TxDb object*

Description

Extract gene models from a TxDb object and annotate last 3' UTR exons and the last CDSs

Usage

```
parse_TxDb(
  sqlite_db = NULL,
  TxDb = getInPASTxDb(),
  edb = getInPASEnsDb(),
  genome = getInPASGenome(),
  chr2exclude = getChr2Exclude(),
  outdir = getInPASOutputDirectory()
)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`. It can be `NULL`.
`TxDb`	An object of GenomicFeatures::TxDb
`edb`	An object of ensembldb::EnsDb
`genome`	An object of BSgenome::BSgenome
`chr2exclude`	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.
`outdir`	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.

Details

A good practice is to perform read alignment using a reference genome from Ensembl/GenCode including only the primary assembly and build a TxDb using the GTF/GFF files downloaded from the same source as the reference genome, such as BioMart/Ensembl/GenCode. For instruction, see Vignette of the GenomicFeatures. The UCSC reference genomes and their annotation can be very cumbersome.

Value

A GenomicRanges::GRanges object for gene models

Author(s)

Haibo Liu

Examples

```
library("EnsDb.Hsapiens.v86")
library("BSgenome.Hsapiens.UCSC.hg19")
library("GenomicFeatures")

## set a sqlite database
bedgraphs <- system.file("extdata", c(
  "Baf3.extract.bedgraph",
  "UM15.extract.bedgraph"
),
package = "InPAS"
)
tags <- c("Baf3", "UM15")
metadata <- data.frame(
  tag = tags,
  condition = c("Baf3", "UM15"),
  bedgraph_file = bedgraphs
)
outdir <- tempdir()
write.table(metadata,
  file = file.path(outdir, "metadata.txt"),
  sep = "\t", quote = FALSE, row.names = FALSE
)
sqlite_db <- setup_sqlitedb(
  metadata =
    file.path(outdir, "metadata.txt"),
  outdir
)

samplefile <- system.file("extdata",
  "hg19_knownGene_sample.sqlite",
  package = "GenomicFeatures"
)
TxDb <- loadDb(samplefile)
edb <- EnsDb.Hsapiens.v86
```

```
genome <- BSgenome.Hsapiens.UCSC.hg19
seqnames <- seqnames(BSgenome.Hsapiens.UCSC.hg19)
chr2exclude <- c(
  "chrM", "chrMT",
  seqnames[grepl("_(hap\\d+|fix|alt)$",
    seqnames,
    perl = TRUE
  )]
)
parsed_Txdb <- parse_TxDb(sqlite_db, TxDb, edb, genome,
  chr2exclude = chr2exclude
)
```

`plot_utr3Usage` *Visualize the dPDUI events using ggplot2*

Description

Visualize the dPDUI events by plotting the MSE, and total coverage per group along 3' UTR regions with dPDUI using `ggplot2::geom_line()`.

Usage

```
plot_utr3Usage(usage_data, vline_color = "purple", vline_type = "dashed")
```

Arguments

`usage_data`	An object of GenomicRanges::GRanges, an output from `get_usage4plot()`.
`vline_color`	color for vertical line showing position of predicated proximal CP site. Default, purple.
`vline_type`	line type for vertical line showing position of predicated proximal CP site. Default, dashed. See ggplot2 linetype.

Value

A ggplot object for refined plotting

Author(s)

Haibo Liu

See Also

For example, see `get_usage4plot()`.

polish_CPs *polish the searching results of CP sites*

Description

remove the multiple positions of CP sites for the same 3' UTRs and only keep the best CP sites for proximal and distal.

Usage

```
polish_CPs(CPs, output.all, DIST2END = 200)
```

Arguments

CPs	output of search_proximalCPs() or adjust_proximalCPs()
output.all	A logical(1), indicating whether to output entries with only single CP site for a 3' UTR.
DIST2END	An integer(1) vector, specifying minimal length difference between proximal and distal APA sites which should be met to be considered for outputted if *output.all* is set to TRUE. Default is 200 bp.

Value

a data.frame with columns: "fit_value", "Predicted_Proximal_APA", "Predicted_Distal_APA", "utr3start", "utr3end", "Predicted_Distal_APA_type"

Author(s)

Jianhong Ou

See Also

adjust_proximalCPs(), adjust_proximalCPsByPWM(), adjust_proximalCPsByNBC(), get_PAscore2()

remove_convergentUTR3s *remove the converging candidates 3' UTRs LIKE UTR3___UTR3*

Description

some of the results is from connected two 3' UTRs. We want to remove them.

Usage

```
remove_convergentUTR3s(x)
```

Arguments

x	the collapsed next.exon.gap coverage

Details

The algorithm need to be improved.

Value

the collapsed next.exon.gap after removing the next 3UTR

Author(s)

Jianhong Ou, Haibo Liu

---

run_coverageQC *Quality control on read coverage over gene bodies and 3UTRs*

Description

Calculate coverage over gene bodies and 3UTRs. This function is used for quality control of the coverage.The coverage rate can show the complexity of RNA-seq library.

Usage

```
run_coverageQC(
  sqlite_db,
  TxDb = getInPASTxDb(),
  edb = getInPASEnsDb(),
  genome = getInPASGenome(),
  cutoff_readsNum = 1,
  cutoff_expdGene_cvgRate = 0.1,
  cutoff_expdGene_sampleRate = 0.5,
  chr2exclude = getChr2Exclude(),
  which = NULL,
  future.chunk.size = 1,
  ...
)
```

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
TxDb	An object of GenomicFeatures::TxDb
edb	An object of ensembldb::EnsDb
genome	An object of BSgenome::BSgenome

`cutoff_readsNum`
: cutoff reads number. If the coverage in the location is greater than cutoff_readsNum,the location will be treated as covered by signal

`cutoff_expdGene_cvgRate`
: cutoff_expdGene_cvgRate and cutoff_expdGene_sampleRate are the parameters used to calculate which gene is expressed in all input dataset. cutoff_expdGene_cvgRateset the cutoff value for the coverage rate of each gene; cutoff_expdGene_sampleRateset the cutoff value for ratio of numbers of expressed and all samples for each gene. for example, by default, cutoff_expdGene_cvgRate=0.1 and cutoff_expdGene_sampleRate=0.5,suppose there are 4 samples, for one gene, if the coverage rates by base are:0.05, 0.12, 0.2, 0.17, this gene will be count as expressed gene because mean(c(0.05,0.12, 0.2, 0.17) > cutoff_expdGene_cvgRate) > cutoff_expdGene_sampleRate if the coverage rates by base are: 0.05, 0.12, 0.07, 0.17, this gene will be count as unexpressed gene because mean(c(0.05, 0.12, 0.07, 0.17) > cutoff_expdGene_cvgRate)<= cutoff_expdGene_sampleRate

`cutoff_expdGene_sampleRate`
: See cutoff_expdGene_cvgRate

`chr2exclude`
: A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.

`which`
: an object of GenomicRanges::GRanges or NULL. If it is not NULL, only the exons overlapping the given ranges are used. For fast data quality control, set which to Granges for one or a few large chromosomes.

`future.chunk.size`
: The average number of elements per future ("chunk"). If Inf, then all elements are processed in a single future. If NULL, then argument future.scheduling = 1 is used by default. Users can set future.chunk.size = total number of elements/number of cores set for the backend. See the future.apply package for details.

`...`
: Not used yet

Value

A data frame as described below.

gene.coverage.rate overage per base for all genes

expressed.gene.coverage.rate coverage per base for expressed genes

UTR3.coverage.rate coverage per base for all 3' UTRs

UTR3.expressed.gene.subset.coverage.rate coverage per base for 3' UTRs of expressed genes

rownames the names of coverage

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
if (interactive()) {
  library("BSgenome.Mmusculus.UCSC.mm10")
  library("TxDb.Mmusculus.UCSC.mm10.knownGene")
  library("EnsDb.Mmusculus.v79")

  genome <- BSgenome.Mmusculus.UCSC.mm10
  TxDb <- TxDb.Mmusculus.UCSC.mm10.knownGene
  edb <- EnsDb.Mmusculus.v79

  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )

  sqlite_db <- setup_sqlitedb(
    metadata = file.path(
      outdir,
      "metadata.txt"
    ),
    outdir
  )
  tx <- parse_TxDb(
    sqlite_db = sqlite_db,
    TxDb = TxDb,
    edb = edb,
    genome = genome,
    outdir = outdir,
    chr2exclude = "chrM"
  )
  addLockName(filename = tempfile())
  coverage <- list()
  for (i in seq_along(bedgraphs)) {
    coverage[[tags[i]]] <- get_ssRleCov(
      bedgraph = bedgraphs[i],
      tag = tags[i],
      genome = genome,
      sqlite_db = sqlite_db,
      outdir = outdir,
```

```
    chr2exclude = "chrM"
  )
}
chr_coverage <- assemble_allCov(sqlite_db,
  seqname = "chr6",
  outdir,
  genome
)
run_coverageQC(sqlite_db, TxDb, edb, genome,
  chr2exclude = "chrM",
  which = GRanges("chr6",
    ranges = IRanges(98013000, 140678000)
  )
)
}
```

`run_fisherExactTest` *Run Fisher Exact Test for differential usage of 3' UTRs for a two-group experimental design*

Description

Run Fisher Exact Test for differential usage of 3' UTRs for a two-group experimental design

Usage

```
run_fisherExactTest(UTR3eset, gp1, gp2)
```

Arguments

`UTR3eset`	An object of UTR3eSet, output of `get_UTR3eSet()`
`gp1`	tag names of group 1
`gp2`	tag names of group 2

Value

a matrix of test results

Author(s)

Jianhong Ou

See Also

`run_singleSampleAnalysis()` for a single-sample APA analysis,`run_singleGroupAnalysis()` for a single-group sample APA analysis, `run_limmaAnalysis()` for limma-based APA analysis of complex experimental design

`run_limmaAnalysis` *use limma to analyze the PDUI*

Description

use limma to analyze the PDUI

Usage

```
run_limmaAnalysis(
  UTR3eset,
  design,
  contrast.matrix,
  coef = 1,
  robust = FALSE,
  ...
)
```

Arguments

`UTR3eset`	An object of UTR3eSet, output of `get_UTR3eSet()`
`design`	A design matrix of the experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates. see `stats::model.matrix()`
`contrast.matrix`	A numeric matrix with rows corresponding to coefficients in fit and columns containing contrasts. May be a vector if there is only one contrast. see `limma::makeContrasts()`
`coef`	An integer(1) vector specifying which coefficient or a character(1) vector specifying which contrast of the linear model is to test. see more `limma::topTable()`. Default, 1.
`robust`	A logical(1) vector,indicating whether the estimation of the empirical Bayes prior parameters be robustified against outlier sample variances?
`...`	other arguments which are passed to `limma::lmFit()`

Value

fit results of eBayes by limma. It is an object of class limma::MArrayLM containing everything found by fit. see `limma::eBayes()`

Author(s)

Jianhong Ou

See Also

`run_singleSampleAnalysis()`, `run_singleGroupAnalysis()`, `run_fisherExactTest()`

run_singleGroupAnalysis

do analysis for single group samples

Description

do analysis for single group samples by ANOVA test

Usage

```
run_singleGroupAnalysis(UTR3eset)
```

Arguments

UTR3eset	An object of UTR3eSet, output of get_UTR3eSet()

Value

a matrix of test results

Author(s)

Jianhong Ou

Examples

```
path <- system.file("extdata", package = "InPAS")
load(file.path(path, "eset.MAQC.rda"))
res <- InPAS:::run_singleGroupAnalysis(eset)
```

run_singleSampleAnalysis

do APA analysis for a single sample

Description

do APA event analysis for a single sample Using Poisson Hidden Markov models

Usage

```
run_singleSampleAnalysis(UTR3eset)
```

Arguments

UTR3eset	the output of get_UTR3eSet()

Details

the test will be performed by comparing a two-state versus an one-state Poisson Hidden Markov models.

Value

a matrix containing test results

Author(s)

Jianhong Ou

See Also

UTR3eSet, get_UTR3eSet(), depmixS4::depmix()

Examples

```
path <- system.file("extdata", package = "InPAS")
load(file.path(path, "eset.MAQC.rda"))
res <- InPAS:::run_singleSampleAnalysis(eset)
```

---

search_CPs *Estimate the CP sites for UTRs on a given chromosome*

Description

Estimate the CP sites for UTRs on a given chromosome

Usage

```
search_CPs(
  seqname,
  sqlite_db,
  genome = getInPASGenome(),
  MINSIZE = 10,
  window_size = 200,
  search_point_START = 100,
  search_point_END = NA,
  cutEnd = NA,
  filter.last = TRUE,
  adjust_distal_polyA_end = FALSE,
  long_coverage_threshold = 2,
  PolyA_PWM = NA,
  classifier = NA,
  classifier_cutoff = 0.8,
  shift_range = 100,
```

```
  step = 2,
  outdir = getInPASOutputDirectory(),
  silence = FALSE,
  cluster_type = c("interactive", "multicore", "torque", "slurm", "sge", "lsf",
    "openlava", "socket"),
  template_file = NULL,
  mc.cores = 1,
  future.chunk.size = 50,
 resources = list(walltime = 3600 * 8, ncpus = 4, mpp = 1024 * 4, queue = "long", memory
    = 4 * 4 * 1024),
  DIST2ANNOAPAP = 500,
  DIST2END = 1000,
  output.all = FALSE
)
```

Arguments

seqname	A character(1) vector, specifying a chromososome/scaffold name
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
genome	A BSgenome::BSgenome object
MINSIZE	A integer(1) vector, specifying the minimal length in bp of a short/proximal 3' UTR. Default, 10
window_size	An integer(1) vector, the window size for novel distal or proximal CP site searching. default: 200.
search_point_START	A integer(1) vector, starting point relative to the 5' extremity of 3' UTRs for searching for proximal CP sites
search_point_END	A integer(1) vector, ending point relative to the 3' extremity of 3' UTRs for searching for proximal CP sites
cutEnd	An integer(1) vector a numeric(1) vector. What percentage or how many nucleotides should be removed from 5' extremities before searching for proximal CP sites? It can be a decimal between 0, and 1, or an integer greater than 1. 0.1 means 10 percent, 25 means cut first 25 bases
filter.last	A logical(1), whether to filter out the last valley, which is likely the 3' end of the longer 3' UTR if no novel distal CP site is detected and the 3' end excluded by setting cutEnd/search_point_END is small.
adjust_distal_polyA_end	A logical(1) vector. If true, distal CP sites are subject to adjustment by the Naive Bayes classifier from the cleanUpdTSeq::cleanUpdTSeq-package
long_coverage_threshold	An integer(1) vector, specifying the cutoff threshold of coverage for the terminal of long form 3' UTRs. If the coverage of first 100 nucleotides is lower than coverage_threshold, that transcript will be not considered for further analysis. Default, 2.

PolyA_PWM
: An R object for a position weight matrix (PWM) for a hexamer polyadenylation signal (PAS), such as AAUAAA.

classifier
: An R object for Naive Bayes classifier model, like the one in the cleanUpdTSeq package.

classifier_cutoff
: A numeric(1) vector. A cutoff of probability that a site is classified as true CP sites. The value should be between 0.5 and 1. Default, 0.8.

shift_range
: An integer(1) vector, specifying a shift range for adjusting the proximal and distal CP sites. Default, 50. It determines the range flanking the candidate CP sites to search the most likely real CP sites.

step
: An integer (1) vector, specifying the step size used for adjusting the proximal or distal CP sites using the Naive Bayes classifier from the cleanUpdTSeq package. Default 1. It can be in the range of 1 to 10.

outdir
: A character(1) vector, a path with write permission for storing the CP sites. If it doesn't exist, it will be created.

silence
: A logical(1), indicating whether progress is reported or not. By default, FALSE

cluster_type
: A character (1) vector, indicating the type of cluster job management systems. Options are "interactive","multicore", "torque", "slurm", "sge", "lsf", "openlava", and "socket". see batchtools vignette

template_file
: A charcter(1) vector, indicating the template file for job submitting scripts when cluster_type is set to "torque", "slurm", "sge", "lsf", or "openlava".

mc.cores
: An integer(1), number of cores for making multicore clusters or socket clusters using **batchtools**, and for `parallel::mclapply()`

future.chunk.size
: The average number of elements per future ("chunk"). If Inf, then all elements are processed in a single future. If NULL, then argument future.scheduling = 1 is used by default. Users can set future.chunk.size = total number of elements/number of cores set for the backend. See the future.apply package for details. Default, 50. This parameter is used to split the candidate 3' UTRs for alternative SP sites search.

resources
: A named list specifying the computing resources when cluster_type is set to "torque", "slurm", "sge", "lsf", or "openlava". See batchtools vignette

DIST2ANNOAPAP
: An integer, specifying a cutoff for annotate MSE valleys with known proximal APAs in a given downstream distance. Default is 500.

DIST2END
: An integer, specifying a cutoff of the distance between last valley and the end of the 3' UTR (where MSE of the last base is calculated). If the last valley is closer to the end than the specified distance, it will not be considered because it is very likely due to RNA coverage decay at the end of mRNA. Default is 1200. User can consider a value between 1000 and 1500, depending on the library preparation procedures: RNA fragmentation and size selection.

output.all
: A logical(1), indicating whether to output entries with only single CP site for a 3' UTR. Default, FALSE.

Value

An object of GenomicRanges::GRanges containing distal and proximal CP site information for each 3' UTR if detected.

Author(s)

Jianhong Ou, Haibo Liu

See Also

search_proximalCPs(), adjust_proximalCPs(), adjust_proximalCPsByPWM(), adjust_proximalCPsByNBC(), get_PAscore(), get_PAscore2()

Examples

```
if (interactive()) {
  library(BSgenome.Mmusculus.UCSC.mm10)
  library(TxDb.Mmusculus.UCSC.mm10.knownGene)
  genome <- BSgenome.Mmusculus.UCSC.mm10
  TxDb <- TxDb.Mmusculus.UCSC.mm10.knownGene

  ## load UTR3 annotation and convert it into a GRangesList
  data(utr3.mm10)
  utr3 <- split(utr3.mm10, seqnames(utr3.mm10), drop = TRUE)

  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )

  sqlite_db <- setup_sqlitedb(metadata = file.path(
    outdir,
    "metadata.txt"
  ), outdir)
  addLockName(filename = tempfile())
  coverage <- list()
  for (i in seq_along(bedgraphs)) {
    coverage[[tags[i]]] <- get_ssRleCov(
      bedgraph = bedgraphs[i],
```

```
    tag = tags[i],
    genome = genome,
    sqlite_db = sqlite_db,
    outdir = outdir,
    chr2exclude = "chrM"
  )
}
data4CPsSearch <- setup_CPsSearch(sqlite_db,
  genome,
  chr.utr3 = utr3[["chr6"]],
  seqname = "chr6",
  background = "10K",
  TxDb = TxDb,
  hugeData = TRUE,
  outdir = outdir,
  minZ = 2,
  cutStart = 10,
  MINSIZE = 10,
  coverage_threshold = 5
)
## polyA_PWM
load(system.file("extdata", "polyA.rda", package = "InPAS"))

## load the Naive Bayes classifier model from the cleanUpdTSeq package
library(cleanUpdTSeq)
data(classifier)
## the following setting just for demo.
if (.Platform$OS.type == "window") {
  plan(multisession)
} else {
  plan(multicore)
}
CPs <- search_CPs(
  seqname = "chr6",
  sqlite_db = sqlite_db,
  genome = genome,
  MINSIZE = 10,
  window_size = 100,
  search_point_START = 50,
  search_point_END = NA,
  cutEnd = 0,
  filter.last = TRUE,
  adjust_distal_polyA_end = TRUE,
  long_coverage_threshold = 2,
  PolyA_PWM = pwm,
  classifier = classifier,
  classifier_cutoff = 0.8,
  shift_range = 100,
  step = 5,
  outdir = outdir
)
}
```

search_distalCPs *search distal CP sites*

Description

search distal CP sites

Usage

```
search_distalCPs(
  chr.cov.merge,
  conn_next_utr3,
  curr_UTR,
  window_size,
  depth.weight,
  long_coverage_threshold,
  background,
  z2s
)
```

Arguments

chr.cov.merge	merged coverage data for a given chromosome
conn_next_utr3	A logical(1) vector, indicating whether joint to next 3UTR or not (used by remove_convergentUTR3s())
curr_UTR	GRanges of 3' UTR for a given chromosome
window_size	An integer(1) vector, the window size for novel distal or proximal CP site searching. default: 100.
depth.weight	A named vector. One element of an output of setup_CPsSearch() for coverage depth weight, which is the output of get_depthWeight()
long_coverage_threshold	An integer(1) vector, specifying the cutoff threshold of coverage for the terminal of long form 3' UTRs. If the coverage of first 100 nucleotides is lower than coverage_threshold, that transcript will be not considered for further analysis. Default, 2.
background	A character(1) vector, the range for calculating cutoff threshold of local background. It can be "same_as_long_coverage_threshold", "1K", "5K","10K", or "50K".
z2s	one element of an output of setup_CPsSearch() for Z-score cutoff values, which is the output of get_zScoreCutoff()

Value

a list #'

- dCPs, a data frame converted from GRanges

- chr.cov.merge, depth-normalized sample/condition specific coverage
- next.exon.gap, all-in-one collapsed, refined next.exon.gap coverage
- annotated.utr3,all-in-one collapsed coverage for annotated proximal UTRs

Author(s)

Jianhong Ou

See Also

`get_PAscore2()`

`search_proximalCPs` *search proximal CPsites*

Description

search proximal CPsites

Usage

```
search_proximalCPs(
  CPs,
  curr_UTR,
  window_size,
  MINSIZE,
  cutEnd = NA,
  search_point_START,
  search_point_END = NA,
  filter.last = TRUE,
  DIST2END = 1000
)
```

Arguments

`CPs`	output from `search_distalCPs()`
`curr_UTR`	GRanges for current 3' UTR
`window_size`	window size
`MINSIZE`	MINSIZE for short form
`cutEnd`	A numeric(1) between 0 and 1 or an integer(1) greater than 1, specifying the percentage of or the number of nucleotides should be removed from the end before search for proximal CP sites, 0.1 means 10 percent. It is recommended to use an integer great than 1, such as 200, 400 or 600, because read coverage at 3' extremities is determined by fragment size due to RNA fragmentation and size selection during library construction.

search_point_START
: An integer, specifying the start position to calculate MSE

search_point_END
: A numeric(1) between 0 and 1 or an integer(1) greater than 1, specifying the percentage of or the number of nucleotides should not be excluded from the end to calculate MSE.

filter.last
: A logical(1), whether to filter out the last valley, which is likely the 3' end of the longer 3' UTR if no novel distal CP site is detected and the 3' end excluded by setting cutEnd/search_point_END is small.

DIST2END
: An integer, specifying a cutoff of the distance between last valley and the end of the 3' UTR (where MSE of the last base is calculated). If the last valley is closer to the end than the specified distance, it will be not be considered because it is very likely due to RNA coverage decay at the end of mRNA. Default is 1200. User can consider a value between 1000 and 1500, depending on the library preparation procedures: RNA fragmentation and size selection.

Value

a list

Author(s)

Jianhong Ou

See Also

adjust_proximalCPs(), polish_CPs(), adjust_proximalCPsByPWM(), adjust_proximalCPsByNBC(), get_PAscore(), get_PAscore2()

---

setup_CPsSearch *prepare data for predicting cleavage and polyadenylation (CP) sites*

Description

prepare data for predicting cleavage and polyadenylation (CP) sites

Usage

```
setup_CPsSearch(
  sqlite_db,
  genome = getInPASGenome(),
  chr.utr3,
  seqname,
  background = c("same_as_long_coverage_threshold", "1K", "5K", "10K", "50K"),
  TxDb = getInPASTxDb(),
  hugeData = TRUE,
  outdir = getInPASOutputDirectory(),
```

```
  silence = FALSE,
  minZ = 2,
  cutStart = 10,
  MINSIZE = 10,
  coverage_threshold = 5
)
```

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
genome	An object of BSgenome::BSgenome
chr.utr3	An object of GenomicRanges::GRanges, an element of the output of extract_UTR3Anno()
seqname	A character(1), the name of a chromosome/scaffold
background	A character(1) vector, the range for calculating cutoff threshold of local background. It can be "same_as_long_coverage_threshold", "1K", "5K","10K", or "50K".
TxDb	an object of GenomicFeatures::TxDb
hugeData	A logical(1) vector, indicating whether it is huge data
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
silence	report progress or not. By default it doesn't report progress.
minZ	A numeric(1), a Z score cutoff value
cutStart	An integer(1) vector a numeric(1) vector. What percentage or how many nucleotides should be removed from 5' extremities before searching for CP sites? It can be a decimal between 0, and 1, or an integer greater than 1. 0.1 means 10 percent, 25 means cut first 25 bases
MINSIZE	A integer(1) vector, specifying the minimal length in bp of a short/proximal 3' UTR. Default, 10
coverage_threshold	An integer(1) vector, specifying the cutoff threshold of coverage for first 100 nucleotides. If the coverage of first 100 nucleotides is lower than coverage_threshold, that transcript will be not considered for further analysis. Default, 5.

Value

A file storing a list as described below:

background The type of methods for background coverage calculation

z2s Z-score cutoff thresholds for each 3' UTRs

depth.weight A named vector containing depth weight

chr.cov.merge A matrix storing condition/sample-specific coverage for 3' UTR and next.exon.gap (if exist)

conn_next_utr3 A logical vector, indicating whether a 3'UTR has a convergent 3' UTR of its downstream transcript

chr.utr3 A GRangesList, storing extracted 3' UTR annotation of transcript on a given chr

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
if (interactive()) {
  library(BSgenome.Mmusculus.UCSC.mm10)
  library("TxDb.Mmusculus.UCSC.mm10.knownGene")
  genome <- BSgenome.Mmusculus.UCSC.mm10
  TxDb <- TxDb.Mmusculus.UCSC.mm10.knownGene

  ## load UTR3 annotation and convert it into a GRangesList
  data(utr3.mm10)
  utr3 <- split(utr3.mm10, seqnames(utr3.mm10), drop = TRUE)

  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )

  sqlite_db <- setup_sqlitedb(
    metadata = file.path(
      outdir,
      "metadata.txt"
    ),
    outdir
  )
  addLockName(filename = tempfile())
  coverage <- list()
  for (i in seq_along(bedgraphs)) {
    coverage[[tags[i]]] <- get_ssRleCov(
      bedgraph = bedgraphs[i],
      tag = tags[i],
      genome = genome,
      sqlite_db = sqlite_db,
      outdir = outdir,
      chr2exclude = "chrM"
    )
  }
```

```
  data4CPsitesSearch <- setup_CPsSearch(sqlite_db,
    genome,
    chr.utr3 = utr3[["chr6"]],
    seqname = "chr6",
    background = "10K",
    TxDb = TxDb,
    hugeData = TRUE,
    outdir = outdir
  )
}
```

---

setup_GSEA *prepare files for GSEA analysis*

Description

output the log2 transformed delta PDUI txt file, chip file, rank file and phynotype label file for GSEA analysis

Usage

```
setup_GSEA(
  eset,
  groupList,
  outdir = getInPASOutputDirectory(),
  preranked = TRUE,
  rankBy = c("logFC", "P.value"),
  rnkFilename = "InPAS.rnk",
  chipFilename = "InPAS.chip",
  dataFilename = "dPDUI.txt",
  PhenFilename = "group.cls"
)
```

Arguments

eset	A UTR3eSet object, output of test_dPDUI()
groupList	A list of grouped sample tag names, with the group names as the list's name, such as list(groupA = c("sample_1", "sample_2", "sample_3"), groupB = c("sample_4", "sample_5", "sample_6"))
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
preranked	A logical(1) vector, out preranked or not
rankBy	A character(1) vector, indicating how the gene list is ranked. It can be "logFC" or "P.value".
rnkFilename	A character(1) vector, specifying a filename for the preranked file
chipFilename	A character(1) vector, specifying a filename for the chip file

dataFilename	A character(1) vector, specifying a filename for the dataset file
PhenFilename	A character(1) vector, specifying a filename for the file containing samples' phenotype labels

Author(s)

Jianhong Ou, Haibo Liu

See Also

data formats for GSEA. https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats

Examples

```
library(limma)
path <- system.file("extdata", package = "InPAS")
load(file.path(path, "eset.MAQC.rda"))
tags <- colnames(eset@PDUI)
g <- factor(gsub("\\..*$", "", tags))
design <- model.matrix(~ -1 + g)
colnames(design) <- c("Brain", "UHR")
contrast.matrix <- makeContrasts(
  contrasts = "Brain-UHR",
  levels = design
)
res <- test_dPDUI(
  eset = eset,
  method = "limma",
  normalize = "none",
  design = design,
  contrast.matrix = contrast.matrix
)
gp1 <- c("Brain.auto", "Brain.phiX")
gp2 <- c("UHR.auto", "UHR.phiX")
groupList <- list(Brain = gp1, UHR = gp2)
setup_GSEA(res,
  groupList = groupList,
  outdir = tempdir(),
  preranked = TRUE,
  rankBy = "P.value"
)
```

---

setup_parCPsSearch *Prepare data for predicting cleavage and polyadenylation (CP) sites using parallel computing*

Description

Prepare data for predicting cleavage and polyadenylation (CP) sites using parallel computing

Usage

```
setup_parCPsSearch(
  sqlite_db,
  genome = getInPASGenome(),
  utr3,
  seqnames,
  background = c("same_as_long_coverage_threshold", "1K", "5K", "10K", "50K"),
  TxDb = getInPASTxDb(),
  future.chunk.size = 1,
  chr2exclude = getChr2Exclude(),
  hugeData = TRUE,
  outdir = getInPASOutputDirectory(),
  silence = FALSE,
  minZ = 2,
  cutStart = 10,
  MINSIZE = 10,
  coverage_threshold = 5
)
```

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
genome	An object of BSgenome::BSgenome
utr3	An object of GenomicRanges::GRangesList, the output of extract_UTR3Anno()
seqnames	A character(1), the names of all chromosomes/scaffolds with both coverage and 3' UTR annotation. Users can get this by calling the get_chromosomes().
background	A character(1) vector, the range for calculating cutoff threshold of local background. It can be "same_as_long_coverage_threshold", "1K", "5K","10K", or "50K".
TxDb	an object of GenomicFeatures::TxDb
future.chunk.size	The average number of elements per future ("chunk"). If Inf, then all elements are processed in a single future. If NULL, then argument future.scheduling = 1 is used by default. Users can set future.chunk.size = total number of elements/number of cores set for the backend. See the future.apply package for details.
chr2exclude	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. chrM and alternative scaffolds representing different haplotypes should be excluded.
hugeData	A logical(1) vector, indicating whether it is huge data
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
silence	report progress or not. By default it doesn't report progress.
minZ	A numeric(1), a Z score cutoff value

`cutStart` An integer(1) vector a numeric(1) vector. What percentage or how many nucleotides should be removed from 5' extremities before searching for CP sites? It can be a decimal between 0, and 1, or an integer greater than 1. 0.1 means 10 percent, 25 means cut first 25 bases

`MINSIZE` A integer(1) vector, specifying the minimal length in bp of a short/proximal 3' UTR. Default, 10

`coverage_threshold` An integer(1) vector, specifying the cutoff threshold of coverage for first 100 nucleotides. If the coverage of first 100 nucleotides is lower than coverage_threshold, that transcript will be not considered for further analysis. Default, 5.

Value

A list of list as described below:

background The type of methods for background coverage calculation

z2s Z-score cutoff thresholds for each 3' UTRs

depth.weight A named vector containing depth weight

chr.cov.merge A list of matrice storing condition/sample- specific coverage for 3' UTR and next.exon.gap (if exist)

conn_next_utr3 A logical vector, indicating whether a 3'UTR has a convergent 3' UTR of its downstream transcript

chr.utr3 A GRangesList, storing extracted 3' UTR annotation of transcript on a given chr

Author(s)

Jianhong Ou, Haibo Liu

---

setup_sqlitedb *Create an SQLite database for storing metadata and paths to coverage files*

Description

Create an SQLite database with five tables, "metadata","sample_coverage", "chromosome_coverage", "CPsites", and "utr3_coverage", for storing metadata (sample tag, condition, paths to bedgraph files, and sample total read coverage), sample-then-chromosome-oriented coverage files (sample tag, chromosome, paths to bedgraph files for each chromosome), and paths to chromosome-then-sample-oriented coverage files (chromosome, paths to bedgraph files for each chromosome), CP sites on each chromosome (chromosome, paths to cpsite files), read coverage for 3' UTR and last CDS regions on each chromosome (chromosome, paths to utr3 coverage file), respectively

Usage

```
setup_sqlitedb(metadata, outdir = getInPASOutputDirectory())
```

Arguments

metadata	A path to a tab-delimited file, with columns "tag", "condition", and "bedgraph_file", storing a unique name tag for each sample, a condition name for each sample, such as "treatment" and "control", and a path to the bedgraph file for each sample
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.

Value

A character(1) vector, the path to the SQLite database

Author(s)

Haibo Liu

Examples

```
if (interactive()) {
  bedgraphs <- system.file("extdata", c(
    "Baf3.extract.bedgraph",
    "UM15.extract.bedgraph"
  ),
  package = "InPAS"
  )
  tags <- c("Baf3", "UM15")
  metadata <- data.frame(
    tag = tags,
    condition = c("Baf3", "UM15"),
    bedgraph_file = bedgraphs
  )
  outdir <- tempdir()
  write.table(metadata,
    file = file.path(outdir, "metadata.txt"),
    sep = "\t", quote = FALSE, row.names = FALSE
  )
  sqlite_db <- setup_sqlitedb(
    metadata =
      file.path(outdir, "metadata.txt"),
    outdir
  )
}
```

---

set_globals *Set up global variables for an InPAS analysis*

Description

Set up global variables for an InPAS analysis

Usage

```
set_globals(
  genome = NULL,
  TxDb = NULL,
  EnsDb = NULL,
  outdir = NULL,
  chr2exclude = c("chrM", "MT", "Pltd", "chrPltd"),
  lockfile = tempfile(tmpdir = getInPASOutputDirectory())
)
```

Arguments

genome	An object BSgenome::BSgenome. To make things easy, we suggest users creating a BSgenome::BSgenome instance from the reference genome used for read alignment. For details, see the documentation of `BSgenome::forgeBSgenomeDataPkg()`.
TxDb	An object of GenomicFeatures::TxDb
EnsDb	An object of ensembldb::EnsDb
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
chr2exclude	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.
lockfile	A character(1) vector, specifying a file name used for parallel writing to a SQLite database

---

test_dPDUI *do test for dPDUI*

Description

do test for dPDUI

Usage

```
test_dPDUI(
  eset,
  sqlite_db,
  outdir = getInPASOutputDirectory(),
  method = c("limma", "fisher.exact", "singleSample", "singleGroup"),
  normalize = c("none", "quantiles", "quantiles.robust", "mean", "median"),
  design,
  contrast.matrix,
  coef = 1,
  robust = FALSE,
  ...
)
```

Arguments

eset	An object of UTR3eSet. It is an output of get_UTR3eSet()
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
outdir	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
method	A character(1), indicating the method for testing dPDUI. It can be "limma", "fisher.exact", "singleSample", or "singleGroup"
normalize	A character(1), indicating the normalization method. It can be "none", "quantiles", "quantiles.robust", "mean", or "median"
design	a design matrix of the experiment, with rows corresponding to samples and columns to coefficients to be estimated. Defaults to the unit vector meaning that the samples are treated as replicates. see stats::model.matrix(). Required for limma-based analysis.
contrast.matrix	a numeric matrix with rows corresponding to coefficients in fit and columns containing contrasts. May be a vector if there is only one contrast. see limma::makeContrasts(). Required for limma-based analysis.
coef	column number or column name specifying which coefficient or contrast of the linear model is of interest. see more limma::topTable(). default value: 1
robust	A logical(1) vector, indicating whether the estimation of the empirical Bayes prior parameters should be robustified against outlier sample variances.
...	other arguments are passed to lmFit

Details

if method is "limma", design matrix and contrast is required. if method is "fisher.exact", gp1 and gp2 is required.

Value

An object of UTR3eSet, with the last element testRes containing the test results in a matrix.

Author(s)

Jianhong Ou, Haibo Liu

See Also

run_singleSampleAnalysis(), run_singleGroupAnalysis(), run_fisherExactTest(), run_limmaAnalysis()

Examples

```
library(limma)
path <- system.file("extdata", package = "InPAS")
load(file.path(path, "eset.MAQC.rda"))
tags <- colnames(eset@PDUI)
g <- factor(gsub("\\..*$", "", tags))
```

```
design <- model.matrix(~ -1 + g)
colnames(design) <- c("Brain", "UHR")
contrast.matrix <- makeContrasts(
  contrasts = "Brain-UHR",
  levels = design
)
res <- test_dPDUI(
  eset = eset,
  sqlite_db,
  method = "limma",
  normalize = "none",
  design = design,
  contrast.matrix = contrast.matrix
)
```

`trim_seqnames` *Filter sequence names from a BSgenome object*

Description

Filter sequence names for scaffolds from a BSgenome object so that only chromosome-level seqnames are kept.

Usage

```
trim_seqnames(genome = getInPASGenome(), chr2exclude = getChr2Exclude())
```

Arguments

`genome`	An object of BSgenome::BSgenome
`chr2exclude`	A character vector, NA or NULL, specifying chromosomes or scaffolds to be excluded for InPAS analysis. `chrM` and alternative scaffolds representing different haplotypes should be excluded.

Value

An character vector containing filtered seqnames

Author(s)

Jianhong Ou, Haibo Liu

utr3.mm10 *Annotation of 3' UTRs for mouse (mm10)*

Description

A dataset containing the annotation of the 3' UTRs of the mouse

Usage

```
utr3.mm10
```

Format

An object of GenomicRanges::GRanges with 7 metadata columns

feature feature type, utr3, CDS, next.exon.gap
annotatedProximalCP candidate proximal CPsites
exon exon ID
transcript transcript ID
gene gene ID
symbol gene symbol
truncated whether the 3' UTR is trucated

UTR3eSet-class *UTR3eSet-class and its methods*

Description

An object of class UTR3eSet representing the results of 3' UTR usage; methods for constructing, showing, getting and setting attributes of objects; methods for coercing object of other class to UTR3eSet objects.

Objects from the Class

Objects can be created by calls of the form `new("UTR3eSet", ...)`

Objects can be created by calls of the form `new("UTR3eSet", ...)`.

Slots

`usage` Object of class `"GRanges"`
`PDUI` Object of class `"matrix"`
`PDUI.log2` Object of class `"matrix"`
`short` Object of class `"matrix"`
`long` Object of class `"matrix"`
`signals` Object of class `"list"`
`testRes` Object of class `"matrix"`

UTR3eSet-class methods

$ `signature(x = "UTR3eSet")`: ...

$<- `signature(x = "UTR3eSet")`: ...

coerce `signature(from = "UTR3eSet", to = "ExpressionSet")`: ...

coerce `signature(from = "UTR3eSet", to = "GRanges")`: ...

show `signature(object = "UTR3eSet")`: ...

Author(s)

Jianhong Ou

See Also

GRanges

Index

∗ **datasets**
utr3.mm10, 72
∗ **internal**
.onAttach, 3
adjust_distalCPs, 7
adjust_proximalCPs, 8
adjust_proximalCPsByNBC, 9
adjust_proximalCPsByPWM, 10
assign_feature, 13
calculate_mse, 13
compensation, 14
fft.smooth, 17
find_valleyBySpline, 20
gcComp, 21
gcContents, 22
get_depthWeight, 26
get_PAscore, 28
get_PAscore2, 29
get_seqLen, 31
get_totalCov, 34
get_UTR3CDS, 37
get_UTR3region, 40
get_UTR3TotalCov, 41
get_zScoreCutoff, 42
mapComp, 43
polish_CPs, 47
remove_convergentUTR3s, 47
run_fisherExactTest, 51
run_limmaAnalysis, 52
run_singleGroupAnalysis, 53
run_singleSampleAnalysis, 53
search_distalCPs, 59
search_proximalCPs, 60
trim_seqnames, 71
-package *(*InPAS*)*, 43
.onAttach, 3
$,UTR3eSet-method *(*UTR3eSet-class*)*, 72
$<-,UTR3eSet-method *(*UTR3eSet-class*)*, 72

addChr2Exclude, 4
addInPASEnsDb, 4
addInPASGenome, 5
addInPASOutputDirectory, 5
addInPASTxDb, 6
addLockName, 6
adjust_distalCPs, 7
adjust_distalCPs(), *19*
adjust_proximalCPs, 8
adjust_proximalCPs(), *19*, *47*, *57*, *61*
adjust_proximalCPsByNBC, 9
adjust_proximalCPsByNBC(), *9*, *11*, *47*, *57*, *61*
adjust_proximalCPsByPWM, 10
adjust_proximalCPsByPWM(), *9*, *10*, *47*, *57*, *61*
assemble_allCov, 11
assign_feature, 13

Biostrings::matchPWM(), *11*
BSgenome::BSgenome, *7–11*, *15*, *21*, *22*, *27–29*, *31*, *32*, *44*, *48*, *55*, *62*, *66*, *69*, *71*
BSgenome::forgeBSgenomeDataPkg(), *32*, *69*

calculate_mse, 13
cleanUpdTSeq::cleanUpdTSeq-package, *55*
coerce,UTR3eSet,ExpressionSet-method *(*UTR3eSet-class*)*, 72
coerce,UTR3eSet,GRanges-method *(*UTR3eSet-class*)*, 72
compensation, 14

depmixS4::depmix(), *54*

ensembldb::EnsDb, *4*, *15*, *23*, *44*, *48*, *69*
extract_UTR3Anno, 15
extract_UTR3Anno(), *25*, *30*, *37*, *41*, *42*, *62*, *66*

fft.smooth, 17

filter_testOut, 17
find_minMSEDistr, 19
find_valleyBySpline, 20

gcComp, 21
gcContents, 22
GenomicFeatures::TxDb, *6*, *15*, *24*, *27*, *44*, *48*, *62*, *66*, *69*
GenomicRanges::GRanges, *13*, *18*, *30*, *35*, *37, 38*, *41, 42*, *45, 46*, *49*, *57*, *62*, *72*
GenomicRanges::GRangesList, *15*, *25*, *66*
get_chromosomes, 25
get_depthWeight, 26
get_depthWeight(), *59*
get_lastCDSUTR3, 27
get_PAscore, 28
get_PAscore(), *9*, *11*, *29*, *57*, *61*
get_PAscore2, 29
get_PAscore2(), *7*, *9, 10*, *28*, *47*, *57*, *60, 61*
get_regionCov, 30
get_seqLen, 31
get_ssRleCov, 32
get_totalCov, 34
get_totalCov(), *41, 42*
get_usage4plot, 35
get_usage4plot(), *46*
get_UTR3CDS, 37
get_UTR3eSet, 38
get_UTR3eSet(), *51–54*, *70*
get_UTR3region, 40
get_UTR3TotalCov, 41
get_zScoreCutoff, 42
get_zScoreCutoff(), *59*
getChr2Exclude, 22
getInPASEnsDb, 23
getInPASGenome, 23
getInPASOutputDirectory, 23
getInPASSQLiteDb, 24
getInPASTxDb, 24
getLockName, 25

InPAS, 43

limma::eBayes(), *52*
limma::lmFit(), *52*
limma::makeContrasts(), *52*, *70*
limma::MArrayLM, *52*
limma::topTable(), *52*, *70*

mapComp, 43

parallel::mclapply(), *56*
parse_TxDb, 44
plot_utr3Usage, 46
polish_CPs, 47
polish_CPs(), *8, 9*, *61*
preprocessCore::normalize.quantiles.robust(), *38*

remove_convergentUTR3s, 47
remove_convergentUTR3s(), *59*
run_coverageQC, 48
run_fisherExactTest, 51
run_fisherExactTest(), *52*, *70*
run_limmaAnalysis, 52
run_limmaAnalysis(), *51*, *70*
run_singleGroupAnalysis, 53
run_singleGroupAnalysis(), *51, 52*, *70*
run_singleSampleAnalysis, 53
run_singleSampleAnalysis(), *51, 52*, *70*

search_CPs, 54
search_CPs(), *41*
search_distalCPs, 59
search_distalCPs(), *7*, *60*
search_proximalCPs, 60
search_proximalCPs(), *7–9*, *19*, *47*, *57*
Seqinfo::Seqinfo, *31*
set_globals, 68
setup_CPsSearch, 61
setup_CPsSearch(), *59*
setup_GSEA, 64
setup_parCPsSearch, 65
setup_sqlitedb, 67
setup_sqlitedb(), *15*, *25*, *27*, *30*, *32*, *34, 35*, *38*, *44*, *48*, *55*, *62*, *66*
show,UTR3eSet-method *(*UTR3eSet-class*)*, 72
stats::fft(), *17*
stats::model.matrix(), *52*, *70*
stats::smooth.spline(), *20*

test_dPDUI, 69
test_dPDUI(), *17, 18*, *64*
trim_seqnames, 71

utr3.mm10, 72
UTR3eSet, *18*, *38*, *51–54*, *64*, *70*, *72*
UTR3eSet-class, 72