

# Package ‘InPAS’

February 3, 2025

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

**Version** 2.14.1

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

**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, stats, S4Vectors, utils

**Depends** R (>= 3.1)

**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** 7.1.2

**Roxygen** list(markdown = TRUE)

**LazyData** true

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/InPAS>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** 448b02b

**git\_last\_commit\_date** 2024-12-19

**Repository** Bioconductor 3.20

**Date/Publication** 2025-02-03

**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 . . . . .	4
addInPASOutputDirectory . . . . .	5
addInPASTxDB . . . . .	5
addLockName . . . . .	5
adjust_distalCPs . . . . .	6
adjust_proximalCPs . . . . .	7
adjust_proximalCPsByNBC . . . . .	8
adjust_proximalCPsByPWM . . . . .	9
assemble_allCov . . . . .	10
assign_feature . . . . .	11
calculate_mse . . . . .	12
compensation . . . . .	13
extract_UTR3Anno . . . . .	13
fft.smooth . . . . .	15
filter_testOut . . . . .	16
find_minMSEdistr . . . . .	17
find_valleyBySpline . . . . .	18
gcComp . . . . .	19
gcContents . . . . .	20
getChr2Exclude . . . . .	21
getInPASEnsDb . . . . .	21
getInPASGenome . . . . .	21
getInPASOutputDirectory . . . . .	22
getInPASSQLiteDb . . . . .	22
getInPASTxDB . . . . .	22
getLockName . . . . .	23
get_chromosomes . . . . .	23
get_depthWeight . . . . .	24
get_lastCDSUTR3 . . . . .	25
get_PAscore . . . . .	26
get_PAscore2 . . . . .	26
get_regionCov . . . . .	27
get_seqLen . . . . .	28
get_ssRleCov . . . . .	29
get_totalCov . . . . .	31
get_usage4plot . . . . .	32
get_UTR3CDS . . . . .	33
get_UTR3eSet . . . . .	34
get_UTR3region . . . . .	37

get\_UTR3TotalCov . . . . . 37

get\_zScoreCutoff . . . . . 38

InPAS . . . . . 39

mapComp . . . . . 39

parse\_TxDb . . . . . 40

plot\_utr3Usage . . . . . 42

polish\_CPs . . . . . 43

remove\_convergentUTR3s . . . . . 43

run\_coverageQC . . . . . 44

run\_fisherExactTest . . . . . 46

run\_limmaAnalysis . . . . . 47

run\_singleGroupAnalysis . . . . . 48

run\_singleSampleAnalysis . . . . . 49

search\_CPs . . . . . 50

search\_distalCPs . . . . . 54

search\_proximalCPs . . . . . 55

setup\_CPsSearch . . . . . 56

setup\_GSEA . . . . . 59

setup\_parCPsSearch . . . . . 60

setup\_sqlitedb . . . . . 62

set\_globals . . . . . 63

test\_dPDUI . . . . . 64

trim\_seqnames . . . . . 65

utr3.mm10 . . . . . 66

UTR3eSet-class . . . . . 66

**Index** **68**

---

.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	<i>Add a globally-applied requirement for filtering out scaffolds from all analysis</i>
----------------	---

---

**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	<i>Add a globally defined EnsDb to some InPAS functions.</i>
---------------	--

---

**Description**

Add a globally defined EnsDb to some InPAS functions.

**Usage**

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

**Arguments**

EnsDb	An object of <a href="#">ensemldb::EnsDb</a>
-------	--

---

addInPASGenome	<i>Add a globally defined genome to all InPAS functions.</i>
----------------	--

---

**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**

<code>outdir</code>	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**

<code>TxDB</code>	An object of <a href="#">GenomicFeatures::TxDb</a>
-------------------	--

**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**

<code>filename</code>	A character(1) vector, specifying 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 <a href="#">search_distalCPs()</a>
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 <a href="#">BSgenome::BSgenome</a> object
seqname	A character(1) vector, specifying a chromosome/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 <a href="#">search_proximalCPs()</a>
PolyA_PWM	PolyA position weight matrix
genome	a <a href="#">BSgenome::BSgenome</a> 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 <a href="#">BSgenome::BSgenome</a> 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**

<code>idx</code>	the offset of positions of CP sites
<code>PolyA_PWM</code>	polyA PWM
<code>seqnames</code>	a character(n) vector, the chromosome/scaffolds' names
<code>starts</code>	start position in the genome
<code>strands</code>	strands
<code>genome</code>	an <a href="#">BSgenome::BSgenome</a> object
<code>shift_range</code>	the shift range of PWM hits
<code>search_point_START</code>	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**

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

**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	<i>Calculate mean squared errors (MSE)</i>
---------------	--

---

**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	<i>Compensate the coverage with GC-content or mappability</i>
--------------	---

---

**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	<i>extract 3' UTR information from a <a href="#">GenomicFeatures::TxDb</a> object</i>
------------------	---

---

**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 <code>setup_sqlitedb()</code> .
TxDB	An object of <code>GenomicFeatures::TxDb</code>
edb	An object of <code>ensemblDb::EnsDb</code>
genome	An object of <code>BSgenome::BSgenome</code>
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_sqllitedb(
    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 <a href="#">stats::fft()</a>
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 <a href="#">UTR3eSet</a> object, output of <a href="#">test_dPDUI()</a>
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 <code>search_proximalCPs()</code> or <code>adjust_distalCPs()</code> or <code>adjust_proximalCPs()</code>
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 <code>splinestats::smooth.spline()</code> . 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 <code>cutEnd/search_point_END</code> 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	<i>Calculate weights for GC composition</i>
--------	---

---

**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 <a href="#">BSgenome::BSgenome</a>
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 <a href="#">BSgenome:BSgenome</a>
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	<i>Get a globally-applied requirement for filtering scaffolds.</i>
----------------	--

---

**Description**

This function will get the default requirement of filtering scaffolds.

**Usage**

```
getChr2Exclude()
```

---

getInPASEnsDb	<i>Get the globally defined EnsDb.</i>
---------------	--

---

**Description**

Get the globally defined EnsDb.

**Usage**

```
getInPASEnsDb()
```

**Value**

An object of [ensemblDb::EnsDb](#)

---

getInPASGenome	<i>Get the globally defined genome</i>
----------------	--

---

**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	<i>Get the path to a file for locking the SQLite database</i>
-------------	---

---

**Description**

Get the path to a file for locking the SQLite database

**Usage**

```
getLockName()
```

**Value**

A path to a file for locking

---

get_chromosomes	<i>Identify chromosomes/scaffolds for CP site discovery</i>
-----------------	---

---

**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 <code>GenomicRanges::GRangesList</code> . An output of <code>extract_UTR3Anno()</code> .
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of <code>setup_sqlitedb()</code> .

**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	<i>Calculate the depth weight for each sample or each experimental condition</i>
-----------------	--

---

### Description

Calculate the depth weight for each sample of non-hugeData or each experimental condition for hugeData:  $\text{depth}/\text{mean}(\text{depth})$

### 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 <a href="#">setup_sqlitedb()</a>
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 <a href="#">GenomicFeatures::TxDb</a>
genome	An object of <a href="#">BSgenome::BSgenome</a>
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	<i>Calculate the CP score</i>
-------------	-------------------------------

---

### 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 <a href="#">BSgenome::BSgenome</a>
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	<i>calculate the CP score</i>
--------------	-------------------------------

---

### 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 <a href="#">BSgenome::BSgenome</a>
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	<i>Get coverage for 3' UTR and last CDS regions on a single chromosome</i>
---------------	--

---

**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 <a href="#">GenomicRanges::GRanges</a> , one element of an output of <a href="#">extract_UTR3Anno()</a>
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of <a href="#">setup_sqlitedb()</a> .
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 <a href="#">BSgenome::BSgenome</a>
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**[GenomeInfoDb::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 <a href="#">BSgenome::BSgenome</a> . To make things easy, we suggest users creating a <a href="#">BSgenome::BSgenome</a> instance from the reference genome used for read alignment. For details, see the documentation of <a href="#">BSgenome:::forgeBSgenomeDataPkg()</a> .
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of <a href="#">setup_sqlitedb()</a> .
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.

<code>outdir</code>	A character(1) vector, a path with write permission for storing InPAS analysis results. If it doesn't exist, it will be created.
<code>chr2exclude</code>	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 <a href="#">setup_sqlitedb()</a> .
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 <a href="#">setup_sqlitedb()</a>
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

---

<code>get_usage4plot</code>	<i>prepare coverage data and fitting data for plot</i>
-----------------------------	--

---

**Description**

prepare coverage data and fitting data for plot

**Usage**

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

**Arguments**

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

**Value**

An object of [GenomicRanges::GRanges](#) with metadata:

<code>dat</code>	A data.frame, first column is the position, the other columns are Coverage and value
<code>offset</code>	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 <code>setup_sqlitedb()</code> .
normalize	A character(1) vector, specifying the normalization method. It can be "none", "quantiles", "quantiles.robust", "mean", or "median"
...	parameter can be passed into <code>preprocessCore::normalize.quantiles.robust()</code>
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 <a href="#">GenomicRanges::GRanges</a> . It must be an element of the output of <a href="#">extract_UTR3Anno()</a> for a given chromosome.
chr.totalCov	total coverage for each condition of a given chromosome. It must be an output of <a href="#">get_totalCov()</a>
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 <a href="#">GenomicRanges::GRanges</a> for introns of a give chromosome/scaffold
chr.totalCov	total coverage for a given chromosome/scaffold, an output from <a href="#">get_totalCov()</a> for a given chromosome/scaffold
chr.utr3	An object of <a href="#">GenomicRanges::GRanges</a> , an element of the output of <a href="#">extract_UTR3Anno()</a> 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 <a href="#">setup_sqlitedb()</a> . It can be NULL.
TxDb	An object of <a href="#">GenomicFeatures::TxDb</a>
edb	An object of <a href="#">ensemblDb::EnsDb</a>
genome	An object of <a href="#">BSgenome::BSgenome</a>
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 <code>GenomicRanges::GRanges</code> , an output from <code>get_usage4plot()</code> .
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 <code>ggplot2</code> <a href="#">linetype</a> .

### 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 <a href="#">search_proximalCPs()</a> or <a href="#">adjust_proximalCPs()</a>
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 <i>output.all</i> 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\_cvgRate set the cutoff value for the coverage rate of each gene; cutoff\_expdGene\_sampleRate set 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  $\text{mean}(c(0.05,0.12, 0.2, 0.17)) > \text{cutoff\_expdGene\_cvgRate}) > \text{cutoff\_expdGene\_sampleRate}$  if the coverage rates by base are: 0.05, 0.12, 0.07, 0.17, this gene will be count as un-expressed gene because  $\text{mean}(c(0.05, 0.12, 0.07, 0.17)) > \text{cutoff\_expdGene\_cvgRate}) \leq \text{cutoff\_expdGene\_sampleRate}$

`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	<i>Run Fisher Exact Test for differential usage of 3' UTRs for a two-group experimental design</i>
---------------------	--

---

**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 <a href="#">UTR3eSet</a> , output of <a href="#">get_UTR3eSet()</a>
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 <code>UTR3eSet</code> , output of <code>get_UTR3eSet()</code>
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 <code>stats::model.matrix()</code>
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 <code>limma::makeContrasts()</code>
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 <code>limma::topTable()</code> . 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 <code>limma::lmFit()</code>

**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 <code>UTR3eSet</code> , output of <code>get_UTR3eSet()</code>
----------	--

**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 chromosome/scaffold name
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of <a href="#">setup_sqlitedb()</a> .
genome	A <a href="#">BSgenome::BSgenome</a> 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 <a href="#">cleanUpdTSeq::cleanUpdTSeq-package</a>
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 <a href="#">batchtools vignette</a>
template_file	A character(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 <a href="#">batchtools</a> , and for <a href="#">parallel::mclapply()</a>
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 <code>cluster_type</code> is set to "torque", "slurm", "sge", "lsf", or "openlava". See <a href="#">batchtools vignette</a>
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 == "windows") {
  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 <a href="#">remove_convergentUTR3s()</a> )
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 <a href="#">setup_CPsSearch()</a> for coverage depth weight, which is the output of <a href="#">get_depthWeight()</a>
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 <a href="#">setup_CPsSearch()</a> for Z-score cutoff values, which is the output of <a href="#">get_zScoreCutoff()</a>

**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 <a href="#">search_distalCPs()</a>
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	<i>prepare data for predicting cleavage and polyadenylation (CP) sites</i>
-----------------	--

---

**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 <code>setup_sqlitedb()</code> .
genome	An object of <code>BSgenome::BSgenome</code>
chr.utr3	An object of <code>GenomicRanges::GRanges</code> , an element of the output of <code>extract_UTR3Anno()</code>
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 <code>GenomicFeatures::TxDb</code>
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 <code>coverage_threshold</code> , 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_CPSearch(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 log<sub>2</sub> 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 <a href="#">UTR3eSet</a> object, output of <code>test_dPDUI()</code>
groupList	A list of grouped sample tag names, with the group names as the list's name, such as <code>list(groupA = c("sample_1", "sample_2", "sample_3"), groupB = c("sample_4", "sample_5", "sample_6"))</code>
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](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	<i>Prepare data for predicting cleavage and polyadenylation (CP) sites using parallel computing</i>
--------------------	---

---

**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 <code>setup_sqlitedb()</code> .
genome	An object of <code>BSgenome::BSgenome</code>
utr3	An object of <code>GenomicRanges::GRangesList</code> , the output of <code>extract_UTR3Anno()</code>
seqnames	A character(1), the names of all chromosomes/scaffolds with both coverage and 3' UTR annotation. Users can get this by calling the <code>get_chromosomes()</code> .
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 <code>GenomicFeatures::TxDb</code>
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 <code>future.scheduling = 1</code> is used by default. Users can set <code>future.chunk.size = total number of elements/number of cores set for the backend</code> . See the <code>future.apply</code> 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 <code>coverage_threshold</code> , 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	<i>Create an SQLite database for storing metadata and paths to coverage files</i>
----------------	---

---

### 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 <code>BSgenome::BSgenome</code> . To make things easy, we suggest users creating a <code>BSgenome::BSgenome</code> instance from the reference genome used for read alignment. For details, see the documentation of <code>BSgenome::forgeBSgenomeDataPkg()</code> .
TxDb	An object of <code>GenomicFeatures::TxDb</code>
EnsDb	An object of <code>ensembldb::EnsDb</code>
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	<i>do test for dPDUI</i>
------------	--------------------------

---

## 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 <a href="#">UTR3eSet</a> . It is an output of <a href="#">get_UTR3eSet()</a>
sqlite_db	A path to the SQLite database for InPAS, i.e. the output of <a href="#">setup_sqlitedb()</a> .
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 <a href="#">stats::model.matrix()</a> . 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 <a href="#">limma::makeContrasts()</a> . 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 <a href="#">limma::topTable()</a> . 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 <a href="#">lmFit</a>

## 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 seq-names 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 truncated

---

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  
Jianhong Ou

**See Also**

[GRanges](#)

# Index

- \* **datasets**
  - utr3.mm10, 66
- \* **internal**
  - .onAttach, 3
  - adjust\_distalCPs, 6
  - adjust\_proximalCPs, 7
  - adjust\_proximalCPsByNBC, 8
  - adjust\_proximalCPsByPWM, 9
  - assign\_feature, 11
  - calculate\_mse, 12
  - compensation, 13
  - fft.smooth, 15
  - find\_valleyBySpline, 18
  - gcComp, 19
  - gcContents, 20
  - get\_depthWeight, 24
  - get\_PAscore, 26
  - get\_PAscore2, 26
  - get\_seqLen, 28
  - get\_totalCov, 31
  - get\_UTR3CDS, 33
  - get\_UTR3region, 37
  - get\_UTR3TotalCov, 37
  - get\_zScoreCutoff, 38
  - mapComp, 39
  - polish\_CPs, 43
  - remove\_convergentUTR3s, 43
  - run\_fisherExactTest, 47
  - run\_limmaAnalysis, 47
  - run\_singleGroupAnalysis, 48
  - run\_singleSampleAnalysis, 49
  - search\_distalCPs, 54
  - search\_proximalCPs, 55
  - trim\_seqnames, 65
- .onAttach, 3
- \$.UTR3eSet-method (UTR3eSet-class), 66
- \$<-, UTR3eSet-method (UTR3eSet-class), 66
- addChr2Exclude, 4
- addInPASEnsDb, 4
- addInPASGenome, 4
- addInPASOutputDirectory, 5
- addInPASTxDB, 5
- addLockName, 5
- adjust\_distalCPs, 6
- adjust\_distalCPs(), 18
- adjust\_proximalCPs, 7
- adjust\_proximalCPs(), 18, 43, 52, 56
- adjust\_proximalCPsByNBC, 8
- adjust\_proximalCPsByNBC(), 7, 10, 43, 52, 56
- adjust\_proximalCPsByPWM, 9
- adjust\_proximalCPsByPWM(), 7, 9, 43, 52, 56
- assemble\_allCov, 10
- assign\_feature, 11
- Biostrings::matchPWM(), 9
- BSgenome::BSgenome, 6–10, 14, 19, 20, 25–29, 40, 44, 50, 57, 61, 63, 65
- BSgenome::forgeBSgenomeDataPkg(), 29, 63
- calculate\_mse, 12
- cleanUpdTSeq::cleanUpdTSeq-package, 51
- coerce, UTR3eSet, ExpressionSet-method (UTR3eSet-class), 66
- coerce, UTR3eSet, GRanges-method (UTR3eSet-class), 66
- compensation, 13
- depmixS4::depmix(), 49
- ensemblDb::EnsDb, 4, 14, 21, 40, 44, 63
- extract\_UTR3Anno, 13
- extract\_UTR3Anno(), 23, 28, 34, 38, 57, 61
- fft.smooth, 15
- filter\_testOut, 16
- find\_minMSEdistr, 17
- find\_valleyBySpline, 18
- gcComp, 19
- gcContents, 20
- GenomeInfoDb::Seqinfo, 29
- GenomicFeatures::TxDb, 5, 13, 14, 22, 25, 40, 44, 57, 61, 63
- GenomicRanges::GRanges, 12, 16, 28, 32, 34, 35, 37, 38, 41, 42, 45, 52, 57, 66

- GenomicRanges::GRangesList, [14](#), [23](#), [61](#)
- get\_chromosomes, [23](#)
- get\_depthWeight, [24](#)
- get\_depthWeight(), [54](#)
- get\_lastCDSUTR3, [25](#)
- get\_PAScore, [26](#)
- get\_PAScore(), [7](#), [10](#), [27](#), [52](#), [56](#)
- get\_PAScore2, [26](#)
- get\_PAScore2(), [6](#), [7](#), [9](#), [26](#), [43](#), [52](#), [55](#), [56](#)
- get\_regionCov, [27](#)
- get\_seqLen, [28](#)
- get\_ssRleCov, [29](#)
- get\_totalCov, [31](#)
- get\_totalCov(), [38](#)
- get\_usage4plot, [32](#)
- get\_usage4plot(), [42](#)
- get\_UTR3CDS, [33](#)
- get\_UTR3eSet, [34](#)
- get\_UTR3eSet(), [47–49](#), [64](#)
- get\_UTR3region, [37](#)
- get\_UTR3TotalCov, [37](#)
- get\_zScoreCutoff, [38](#)
- get\_zScoreCutoff(), [54](#)
- getChr2Exclude, [21](#)
- getInPASEnsDb, [21](#)
- getInPASGenome, [21](#)
- getInPASOutputDirectory, [22](#)
- getInPASSQLiteDb, [22](#)
- getInPASTxDB, [22](#)
- getLockName, [23](#)
- GRanges, [67](#)
- InPAS, [39](#)
- limma::eBayes(), [48](#)
- limma::lmFit(), [48](#)
- limma::makeContrasts(), [48](#), [64](#)
- limma::MArrayLM, [48](#)
- limma::topTable(), [48](#), [64](#)
- mapComp, [39](#)
- parallel::mclapply(), [51](#)
- parse\_TxDB, [40](#)
- plot\_utr3Usage, [42](#)
- polish\_CPs, [43](#)
- polish\_CPs(), [7](#), [56](#)
- preprocessCore::normalize.quantiles.robust(), [35](#)
- remove\_convergentUTR3s, [43](#)
- remove\_convergentUTR3s(), [54](#)
- run\_coverageQC, [44](#)
- run\_fisherExactTest, [46](#)
- run\_fisherExactTest(), [48](#), [65](#)
- run\_limmaAnalysis, [47](#)
- run\_limmaAnalysis(), [47](#), [65](#)
- run\_singleGroupAnalysis, [48](#)
- run\_singleGroupAnalysis(), [47](#), [48](#), [65](#)
- run\_singleSampleAnalysis, [49](#)
- run\_singleSampleAnalysis(), [47](#), [48](#), [65](#)
- search\_CPs, [50](#)
- search\_CPs(), [37](#)
- search\_distalCPs, [54](#)
- search\_distalCPs(), [6](#), [55](#)
- search\_proximalCPs, [55](#)
- search\_proximalCPs(), [6](#), [7](#), [18](#), [43](#), [52](#)
- set\_globals, [63](#)
- setup\_CPsSearch, [56](#)
- setup\_CPsSearch(), [54](#)
- setup\_GSEA, [59](#)
- setup\_parCPsSearch, [60](#)
- setup\_sqlitedb, [62](#)
- setup\_sqlitedb(), [14](#), [23](#), [24](#), [28](#), [29](#), [31](#), [32](#), [35](#), [40](#), [44](#), [50](#), [57](#), [61](#)
- show, UTR3eSet-method (UTR3eSet-class), [66](#)
- stats::fft(), [15](#)
- stats::model.matrix(), [48](#), [64](#)
- stats::smooth.spline(), [18](#)
- test\_dPDUI, [64](#)
- test\_dPDUI(), [16](#), [17](#), [59](#)
- trim\_seqnames, [65](#)
- utr3.mm10, [66](#)
- UTR3eSet, [16](#), [35](#), [47–49](#), [59](#), [64–66](#)
- UTR3eSet-class, [66](#)