Package ‘InPAS’

April 12, 2022

Title A Bioconductor package for identifying novel Alternative PolyAdenylation Sites (PAS) from RNA-seq data

Version 2.2.0

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 RNASeq, Sequencing, AlternativeSplicing, Coverage, DifferentialSplicing, GeneRegulation, Transcription, ImmunoOncology

License GPL (>= 2)

Imports AnnotationDbi, BSgenome, cleanUpdTSeq, preprocessCore, IRanges, GenomeInfoDb, depmixS4, limma, BiocParallel, Biostrings, dplyr, magrittr, plyranges, readr, RSQLite, DBI, purrr, GenomicFeatures, ggplot2, reshape2

Depends R (>= 3.1), methods, Biobase, GenomicRanges, S4Vectors

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

VignetteBuilder knitr

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

LazyData true

Encoding UTF-8

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

git_branch RELEASE_3_14

git_last_commit 84281b7

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

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

R topics documented:

`assemble_allCov` *Assemble coverage files 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, outdir, genome, removeScaffolds = FALSE)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb()
`outdir`	A character(1) vector, a path with write permission for storing the coverage data. If it doesn't exist, it will be created.
`genome`	An object of BSgenome::BSgenome
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.

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()
   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,
                            removeScaffolds = TRUE,
                              BPPARAM = NULL)
    }
    coverage_files <- assemble_allCov(sqlite_db,
                                      outdir,
                                      genome,
                                      removeScaffolds = FALSE)
}
```

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

Description

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

Usage

```
extract_UTR3Anno(
  TxDb = NULL,
  edb = NULL,
  removeScaffolds = FALSE,
  MAX_EXONS_GAP = 10000
)
```

Arguments

`TxDb`	an object of GenomicFeatures::TxDb
`edb`	An object of ensembldb::EnsDb
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.
`MAX_EXONS_GAP`	An integer(1) vector, maximal gap sizes between last known CP sites to downstream exons

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 cubersome.

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.

Author(s)

Jianhong Ou, Haibo Liu

Examples

```
library("EnsDb.Hsapiens.v86")
library("GenomicFeatures")
samplefile <- system.file("extdata",
                          "hg19_knownGene_sample.sqlite",
                           package = "GenomicFeatures")
TxDb <- loadDb(samplefile)
edb <- EnsDb.Hsapiens.v86
utr3 <- extract_UTR3Anno(TxDb, edb,
                 removeScaffolds = TRUE,
                 MAX_EXONS_GAP = 10000)
```

---

filter_testOut *filter 3' UTR usage test results*

Description

filter results of `test_dPDUI()`

Usage

```
filter_testOut(
  res,
  gp1,
  gp2,
  background_coverage_threshold = 2,
  P.Value_cutoff = 0.05,
  adj.P.Val_cutoff = 0.05,
  dPDUI_cutoff = 0.3,
  PDUI_logFC_cutoff
)
```

Arguments

`res`	a UTR3eSet object, output of `test_dPDUI()`
`gp1`	tag names involved in group 1. gp1 and gp2 are used for filtering purpose if both are specified; otherwise only other specified thresholds are used for filtering.
`gp2`	tag names involved in group 2
`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)
```

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

Description

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

Usage

```
get_regionCov(chr.utr3, sqlite_db, outdir, BPPARAM = NULL, phmm = FALSE)
```

Arguments

`chr.utr3`	one element of an output of `extract_UTR3Anno()`
`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`outdir`	A path to a folder for storing coverage data of 3' UTRs and last CDSs on a given chromosome/scaffold. If it doesn't exist, it will be created.
`BPPARAM`	an optional BiocParallel::BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to bplapply. It can be set to NULL or bpparam()
`phmm`	A logical(1) vector, indicating whether data should be prepared for singleSample analysis? By default, FALSE

Value

coverage view in GRanges

Author(s)

Jianhong Ou, Haibo Liu

`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,
  sqlite_db,
  outdir,
  BATCH_SIZE = 10L,
  removeScaffolds = FALSE,
  BPPARAM = NULL
)
```

Arguments

`bedgraph`	A path to a bedGraph file
`tag`	A character(1) vector, a name tag used to label the bedgraph file. It must match the tag specified in the metadata file used to setup the SQLite database
`genome`	an object BSgenome::BSgenome. To make things easy, we suggest users creating a BSgenome::BSgenome instance from the reference genome used for read alignment. For details, see the documentation of `BSgenome::forgeBSgenomeDataPkg()`.
`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`outdir`	A character(1) vector, a path with write permission for storing the coverage data. If it doesn't exist, it will be created.
`BATCH_SIZE`	A integer(1) vector, indicating the number of parallel jobs run at the same time per batch. Default, 10. You may adjust this number based based on the available computing resource: CPUs and RAM. For BATCH_SIZE of 10, 15-20G RAM is needed. This parameter affects the time for converting coverage from bedgraph to Rle.
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.
`BPPARAM`	an optional BiocParallel::BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to bplapply. It can be set to NULL or bpparam()

Value

A list of lists containing read coverage as Rle instances of S4Vectors::Rle representing read coverage for each chromosome of a given sample, as described below.

tag the sample tag

- **chr1** coverage as Rle instance for chr1
- **chr2** coverage as Rle instance for chr2
- **chrN** coverage as Rle instance for chrN

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)
   coverage <- get_ssRleCov(bedgraph = bedgraphs[1],
                          tag = tags[1],
                          genome = genome,
                          sqlite_db = sqlite_db,
                          outdir = outdir,
                          removeScaffolds = TRUE,
                          BPPARAM = NULL)
   # check read coverage depth
   db_connect <- dbConnect(drv = RSQLite::SQLite(), dbname = sqlite_db)
   dbReadTable(db_connect, "metadata")
   dbDisconnect(db_connect)
}
```

`get_usage4plot` *prepare coverage data and fitting data for plot*

Description

prepare coverage data and fitting data for plot

Usage

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

Arguments

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

Value

An object of GenomicRanges::GRanges with metadata:

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

Author(s)

Jianhong Ou, Haibo Liu

Examples

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

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

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)
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,
                         removeScaffolds = TRUE,
                         BPPARAM = NULL)
}
coverage_files <- assemble_allCov(sqlite_db,
                                  outdir,
                                  genome,
                                  removeScaffolds = TRUE)

data4CPsSearch <- setup_CPsSearch(sqlite_db,
                                  genome,
                                  utr3,
                                  background = "10K",
                                  TxDb = TxDb,
                                  removeScaffolds = TRUE,
                                  BPPARAM = NULL,
                                  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_UTR3eSet *prepare 3' UTR coverage data for usage test*

Description

generate a UTR3eSet object with PDUI information for statistic tests

Usage

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

Arguments

sqlite_db	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().
normalize	A character(1) vector, spcifying the normalization method. It can be "none", "quantiles", "quantiles.robust", "mean", or "median"

`...`	parameter can be passed into `preprocessCore::normalize.quantiles.robust()`
`singleSample`	A logical(1) vector, indicating whether data is prepared for analysis in a single-Sample mode? Default, FALSE

Value

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

Author(s)

Jianhong Ou, Haibo Liu

Examples

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

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

   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()
   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,
                            removeScaffolds = TRUE,
                            BPPARAM = NULL)}
   coverage_files <- assemble_allCov(sqlite_db,
                                    outdir,
                                    genome,
                                    removeScaffolds = TRUE)
   data4CPsSearch <- setup_CPsSearch(sqlite_db,
                                     genome,
                                     utr3,
                                     background = "10K",
                                     TxDb = TxDb,
                                     removeScaffolds = TRUE,
                                     BPPARAM = NULL,
                                     hugeData = TRUE,
                                     outdir = outdir)
   ## 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,
                     utr3 = utr3,
                     background = data4CPsSearch$background,
                     z2s = data4CPsSearch$z2s,
                     depth.weight = data4CPsSearch$depth.weight,
                     genome = genome,
                     MINSIZE = 10,
                     window_size = 100,
                     search_point_START =50,
                     search_point_END = NA,
                     cutStart = 10,
                     cutEnd = 0,
                     adjust_distal_polyA_end = TRUE,
                     coverage_threshold = 5,
                     long_coverage_threshold = 2,
                     PolyA_PWM = pwm,
                     classifier = classifier,
                     classifier_cutoff = 0.8,
                     shift_range = 100,
                     step = 5,
                     two_way = FALSE,
                     hugeData = TRUE,
                     outdir = outdir)

utr3_cds <- InPAS:::get_UTR3CDS(sqlite_db,
                        chr.utr3 = utr3[["chr6"]],
                        BPPARAM = NULL)

utr3_cds_cov <- get_regionCov(chr.utr3 = utr3[["chr6"]],
                              sqlite_db,
                              outdir,
                              BPPARAM = NULL,
                              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)
}
```

`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, assemble_allCov, get_ssRleCov, run_coverageQC, get_UTR3eSet, test_dPDUI, run_singleSampleAnalysis, run_singleGroupAnalysis, run_limmaAnalysis, filter_testOut, get_usage4plot, setup_GSEA

functions for retrieving 3' UTR annotation

parse_TxDb, extract_UTR3Anno

functions for processing read coverage data

assemble_allCov, get_ssRleCov, run_coverageQC

functions for alternative polyadenylation site analysis

test_dPDUI, run_singleSampleAnalysis, run_singleGroupAnalysis, run_limmaAnalysis, filter_testOut, get_usage4plot

---

`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(TxDb = NULL, edb = NULL, removeScaffolds = FALSE)
```

Arguments

`TxDb`	An object of GenomicFeatures::TxDb
`edb`	An object of ensembldb::EnsDb
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.

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

parsed_Txdb <- parse_TxDb(TxDb, edb,
                          removeScaffolds = TRUE)
```

`plot_utr3Usage` *Visualize the dPDUI events using ggplot2*

Description

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

Usage

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

Arguments

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

Value

A ggplot object for refined plotting

Author(s)

Haibo Liu

See Also

For example, see `get_usage4plot()`.

`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,
  edb,
  genome,
  cutoff_readsNum = 1,
  cutoff_expdGene_cvgRate = 0.1,
  cutoff_expdGene_sampleRate = 0.5,
  removeScaffolds = FALSE,
  BPPARAM = NULL,
  which = NULL,
  ...
)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`TxDb`	An object of GenomicFeatures::TxDb
`edb`	An object of ensembldb::EnsDb
`genome`	An object of BSgenome::BSgenome
`cutoff_readsNum`	cutoff reads number. If the coverage in the location is greater than cutoff_readsNum,the location will be treated as covered by signal
`cutoff_expdGene_cvgRate`	cutoff_expdGene_cvgRate and cutoff_expdGene_sampleRate are the parameters used to calculate which gene is expressed in all input dataset. cutoff_expdGene_cvgRateset the cutoff value for the coverage rate of each gene; cutoff_expdGene_sampleRateset the cutoff value for ratio of numbers of expressed and all samples for each gene. for example, by default, cutoff_expdGene_cvgRate=0.1 and cutoff_expdGene_sampleRate=0.5,suppose there are 4 samples, for one gene, if the coverage rates by base are:0.05, 0.12, 0.2, 0.17, this gene will be count as expressed gene because mean(c(0.05,0.12, 0.2, 0.17) > cutoff_expdGene_cvgRate) > cutoff_expdGene_sampleRate if the coverage rates by base are: 0.05, 0.12, 0.07, 0.17, this gene will be count as unexpressed gene because mean(c(0.05, 0.12, 0.07, 0.17) > cutoff_expdGene_cvgRate)<= cutoff_expdGene_sampleRate
`cutoff_expdGene_sampleRate`	See cutoff_expdGene_cvgRate
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.
`BPPARAM`	an optional BiocParallel::BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to bplapply. It can be set to NULL or bpparam()
`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.
`...`	Not used yet

Value

A data frame with colnames: gene.coverage.rate: coverage 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.c
rate: coverage per base for 3' UTRs of expressed genes. and 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)
   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,
                            removeScaffolds = TRUE,
                            BPPARAM = NULL)
   }
   coverage_files <- assemble_allCov(sqlite_db,
                                    outdir,
                                    genome,
                                    removeScaffolds = FALSE)
   run_coverageQC(sqlite_db, TxDb, edb, genome,
                  removeScaffolds = TRUE,
                  which = GRanges("chr6",
                     ranges = IRanges(98013000, 140678000)))
}
```

`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,
  utr3,
  background,
  z2s,
  depth.weight,
  genome,
  MINSIZE = 10,
  window_size = 100,
  search_point_START = 50,
  search_point_END = NA,
  cutStart = 10,
  cutEnd = 0,
  adjust_distal_polyA_end = TRUE,
  coverage_threshold = 5,
  long_coverage_threshold = 2,
  PolyA_PWM = NA,
  classifier = NA,
  classifier_cutoff = 0.8,
  shift_range = window_size,
  step = 1,
  two_way = FALSE,
  hugeData = TRUE,
  outdir,
  silence = FALSE
)
```

Arguments

`seqname`	A character(1) vector, specifying a chromososome/scaffold name
`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`utr3`	An object of GenomicRanges::GRanges. Output of `extract_UTR3Anno()` for 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".
`z2s`	one element of an output of `setup_CPsSearch()` for Z-score cutoff values, which is the output of `get_zScoreCutoff()`
`depth.weight`	A named vector. One element of an output of `setup_CPsSearch()` for coverage depth weight, which is the output of `get_depthWeight()`
`genome`	A BSgenome::BSgenome object
`MINSIZE`	A integer(1) vector, specifying the minimal length in bp of a short/proximal 3' UTR. Default, 10

`window_size`	An integer(1) vector, the window size for novel distal or proximal CP site searching. default: 100.
`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
`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
`cutEnd`	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
`adjust_distal_polyA_end`	A logical(1) vector. If true, distal CP sites are subject to adjustment by the Naive Bayes classifier from the cleanUpdTSeq::cleanUpdTSeq-package
`coverage_threshold`	An integer(1) vector, specifying the cutoff threshold of coverage for first 100 nucleotides. If the coverage of first 100 nucleotides is lower than coverage_threshold, that transcript will be not considered for further analysis. Default, 5.
`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, 100. 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.
`two_way`	A logical (1), indicating whether the proximal CP sites are searched from both directions or not.
`hugeData`	A logical(1), indicating whether it is huge data

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

Value

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

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))

   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()
   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,
                             removeScaffolds = TRUE,
                             BPPARAM = NULL)}
   coverage_files <- assemble_allCov(sqlite_db,
                                     outdir,
                                     genome,
                                     removeScaffolds = TRUE)
   data4CPsSearch <- setup_CPsSearch(sqlite_db,
                                      genome,
                                      utr3,
                                      background = "10K",
                                      TxDb = TxDb,
                                      removeScaffolds = TRUE,
                                      BPPARAM = NULL,
                                      hugeData = TRUE,
                                      outdir = outdir)
   ## 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,
                     utr3 = utr3,
                     background = data4CPsSearch$background,
                     z2s = data4CPsSearch$z2s,
                     depth.weight = data4CPsSearch$depth.weight,
                     genome = genome,
                     MINSIZE = 10,
                     window_size = 100,
                     search_point_START =50,
                     search_point_END = NA,
                     cutStart = 10,
                     cutEnd = 0,
                     adjust_distal_polyA_end = TRUE,
                     coverage_threshold = 5,
                     long_coverage_threshold = 2,
                     PolyA_PWM = pwm,
                     classifier = classifier,
                     classifier_cutoff = 0.8,
                     shift_range = 100,
                     step = 5,
                     two_way = FALSE,
                     hugeData = TRUE,
                     outdir = outdir)
}
```

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

Description

prepare data for predicting cleavage and polyadenylation (CP) sites

Usage

```
setup_CPsSearch(
  sqlite_db,
  genome,
  utr3,
  background = c("same_as_long_coverage_threshold", "1K", "5K", "10K", "50K"),
  TxDb = NA,
  removeScaffolds = FALSE,
  hugeData = TRUE,
  outdir,
  BPPARAM = NULL,
  silence = FALSE
)
```

Arguments

`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of `setup_sqlitedb()`.
`genome`	An object of BSgenome::BSgenome
`utr3`	An object of GenomicRanges::GRangesList, output of `extract_UTR3Anno()`
`background`	A character(1) vector, the range for calculating cutoff threshold of local background. It can be "same_as_long_coverage_threshold", "1K", "5K","10K", or "50K".
`TxDb`	an object of GenomicFeatures::TxDb
`removeScaffolds`	A logical(1) vector, whether the scaffolds should be removed from the genome If you use a TxDb containing alternative scaffolds, you'd better to remove the scaffolds.
`hugeData`	A logical(1) vector, indicating whether it is huge data
`outdir`	A character(1) vector, a path with write permission for storing the coverage data. If it doesn't exist, it will be created.
`BPPARAM`	an optional BiocParallel::BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to bplapply. It can be set to NULL or bpparam()
`silence`	report progress or not. By default it doesn't report progress.

Value

A list as described below:

utr3TotalCov chromosome-wise 3' UTR coverage in summarized View format

- **chr1** A filename for chr1 3' UTR coverage in summarized View format

chr2 A filename for chr2 3' UTR coverage in summarized View format

chrN A filename for chrN 3' UTR coverage in summarized View format

background The type of methods for bckground coverage calculation

z2s Z-score cutoff thresholds for each 3' UTRs

depth.weight A named vector containing depth weight

Author(s)

Jianhong Ou, Haibo Liu

@examples if (interactive()) library(BSgenome.Mmusculus.UCSC.mm10) library("TxDb.Mmusculus.UCSC.mm10.knownC

```
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))

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()
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,
                         removeScaffolds = TRUE)
}
coverage_files <- assemble_allCov(sqlite_db,
                                 outdir,
                                 genome,
                                 removeScaffolds = TRUE)
data4CPsitesSearch <- setup_CPsSearch(sqlite_db,
                                      genome,
                                      utr3,
                                     background = "10K",
                                     TxDb = TxDb,
                                     removeScaffolds = TRUE,
                                     hugeData = TRUE,
                                     outdir = outdir)
```

---

setup_GSEA *prepare files for GSEA analysis*

Description

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

Usage

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

Arguments

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

Author(s)

Jianhong Ou, Haibo Liu

See Also

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

Examples

```
if (interactive()) {
  file <- system.file("extdata", "eset.MAQC.rda", package = "InPAS")
  load(file)
  gp1 <- c("Brain.auto", "Brain.phiX")
  gp2 <- c("UHR.auto", "UHR.phiX")
  groupList <- list(Brain = gp1, UHR = gp2)
  prepare4GSEA(eset,
               groupList = groupList,
               outdir = tempdir(),
               preranked = TRUE,
               rankBy = "logFC")
}
```

---

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

Description

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

Usage

```
setup_sqlitedb(metadata, outdir)
```

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 the SQLite database. 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)
}
```

`test_dPDUI` *do test for dPDUI*

Description

do test for dPDUI

Usage

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

Arguments

`eset`	An object of UTR3eSet. It is an output of `get_UTR3eSet()`
`method`	A character(1), indicating the method for testing dPDUI. It can be "limma", "fisher.exact", "singleSample", or "singleGroup"
`normalize`	A character(1), indicating the normalization method. It can be "none", "quantiles", "quantiles.robust", "mean", or "median"
`design`	a design matrix of the experiment, with rows corresponding to samples and columns to coefficients to be estimated. Defaults to the unit vector meaning that the samples are treated as replicates. see `stats::model.matrix()`. Required for limma-based analysis.
`contrast.matrix`	a numeric matrix with rows corresponding to coefficients in fit and columns containing contrasts. May be a vector if there is only one contrast. see `limma::makeContrasts()`. Required for limma-based analysis.
`coef`	column number or column name specifying which coefficient or contrast of the linear model is of interest. see more `limma::topTable()`. default value: 1
`robust`	logical, should the estimation of the empirical Bayes prior parameters be robustified against outlier sample variances?
`...`	other arguments are passed to lmFit
`sqlite_db`	A path to the SQLite database for InPAS, i.e. the output of setup_sqlitedb().

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,
                  method = "limma",
                  normalize = "none",
                  design = design,
                  contrast.matrix = contrast.matrix)
```

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

Description

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

Usage

```
utr3.mm10
```

Format

An object of GenomicRanges::GRanges with 7 metadata columns

feature feature type, utr3, CDS, next.exon.gap

annotatedProximalCP candidate proximal CPsites

exon exon ID

transcript transcript ID

gene gene ID

symbol gene symbol

truncated whether the 3' UTR is trucated

`UTR3eSet-class` *UTR3eSet-class and its methods*

Description

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

Objects from the Class

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

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

Slots

`usage`**:** Object of class "`GRanges`"

`PDUI`**:** Object of class "`matrix`"

`PDUI.log2`**:** Object of class "`matrix`"

`short`**:** Object of class "`matrix`"

`long`**:** Object of class "`matrix`"

`signals`**:** Object of class "`list`"

`testRes`**:** Object of class "`matrix`"

UTR3eSet-class methods

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

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

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

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

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

Author(s)

Jianhong Ou

Jianhong Ou

See Also

`GRanges`

Index