Package 'ODER'

May 19, 2022

Title Optimising the Definition of Expressed Regions

Version 1.3.0

Date 2021-04-13

Description The aim of ODER is to identify previously unannotated expressed regions (ERs) using RNA-sequencing data. For this purpose, ODER defines and optimises the definition of ERs, then connected these ERs to genes using junction data. In this way, ODER improves gene annotation. Gene annotation is a staple input of many bioinformatic pipelines and a more complete gene annotation can enable more accurate interpretation of disease associated variants.

License Artistic-2.0

URL https://github.com/eolagbaju/ODER

BugReports https://support.bioconductor.org/t/ODER

biocViews Software, GenomeAnnotation, Transcriptomics, RNASeq, GeneExpression, Sequencing, DataImport

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Suggests BiocStyle, covr, knitr, recount, RefManageR, rmarkdown, sessioninfo, SummarizedExperiment, testthat (>= 3.0.0), GenomicFeatures, xfun

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

Imports BiocGenerics, BiocFileCache, dasper, derfinder, dplyr, IRanges, GenomeInfoDb, GenomicRanges, ggplot2, ggpubr, ggrepel, magrittr, rtracklayer, S4Vectors, stringr, data.table, megadepth, methods, plyr, purrr, tibble, utils

Depends R (>= 4.1)

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

git_branch master

git_last_commit a22c045

git_last_commit_date 2022-04-26

Date/Publication 2022-05-19

Author Emmanuel Olagbaju [aut],
David Zhang [aut, cre] (<https://orcid.org/0000-0003-2382-8460>),
Sebastian Guelfi [ctb],
Siddharth Sethi [ctb]

Maintainer David Zhang <david.zhang.12@ucl.ac.uk>

R topics documented:

`add_expressed_genes` *Adding the nearest expressed genes*

Description

Updating expressed regions with the expressed gene that is closest to it. After entering the tissue that has been sequenced, the nearest gene and nearest expressed gene will be added to the metadata columns of the annotated ERs.

Usage

```
add_expressed_genes(
  input_file = NULL,
  tissue,
  gtf,
  species = "Homo_sapiens",
  annot_ers,
  type_col_name = "type"
)
```

Arguments

input_file	GTEX median expression file, if left as NULL the default file will be used.
tissue	Tissue to filter for. See tissue_options for options
gtf	Either a string containg the path to a .gtf file or a pre-imported gtf using rtracklayer::import . Provides gene data to help determine the nearest gene and nearest expressed gene.
species	character string containing the species to filter for, Homo sapiens is the default
annot_ers	annotated ERs i.e. the product of annotatERs, should have an mcols column called "annotation"
type_col_name	column name in the gtf file to filter on genes. Default is "type

Value

Granges with annotated ERs and details of their nearest expressed genes

Examples

```
gtf_url <- paste0(
    "http://ftp.ensembl.org/pub/release-103/gtf/",
    "homo_sapiens/Homo_sapiens.GRCh38.103.chr.gtf.gz"
)
gtf_path <- file_cache(gtf_url)
gtf_gr <- rtracklayer::import(gtf_path)

ex_opt_ers <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr21", "chr22"), c(2, 2)),
    ranges = IRanges::IRanges(
        start = c(5116369, 5118691, 5125879, 5128214),
        end = c(5117231, 5118847, 5125988, 5128403)
    )
)

ex_opt_ers_w_exp_genes <- add_expressed_genes(
    tissue = "lung", gtf = gtf_gr,
    annot_ers = ex_opt_ers
)

ex_opt_ers_w_exp_genes
```

annotatERs *Connects ERs to genes using junction data, then classifies ERs into "exonic", "intronic", "intergenic", or a combination of these categories*

Description

Finds the overlap between junctions and ERs, then adds gene info and junction info as metadata columns. Then, uses a `gtf` file or a `Txdb` passed in to generate a genomic state used to label each ER as to whether they are exonic, intronic, intergenic or none.

Usage

```
annotatERs(opt_ers, junc_data, genom_state, gtf, txdb)
```

Arguments

`opt_ers`	optimally defined ERs (the product of the ODER function)
`junc_data`	junction data that should match the ERs passed into opt_ers
`genom_state`	a genomic state object
`gtf`	gtf in a GRanges object, pre-imported using `rtracklayer::import` . This is used to provide the gene information for annotation.
`txdb`	TxDb-class (txdb object) to create genomic state. This is used to annotate the expressed regions as exonic, intronic or intergenic.

Value

annotated ERs

Examples

```
gtf_url <- paste0(
    "http://ftp.ensembl.org/pub/release-103/gtf/",
    "homo_sapiens/Homo_sapiens.GRCh38.103.chr.gtf.gz"
)
# file_cache is an internal function to download a bigwig file from a link
# if the file has been downloaded recently, it will be retrieved from a cache
gtf_path <- file_cache(gtf_url)

gtf_gr <- rtracklayer::import(gtf_path)

ex_opt_ers <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr21"), c(5)),
    ranges = IRanges::IRanges(
        start = c(5032176, 5033408, 5034717, 5035188, 5036577),
        end = c(5032217, 5033425, 5034756, 5035189, 5036581)
    )
)
```

```
junctions <- SummarizedExperiment::rowRanges(dasper::junctions_example)
chrs_to_keep <- c("21", "22")

#### preparing the txdb and genomstate object(s)

hg38_chrominfo <- GenomeInfoDb::getChromInfoFromUCSC("hg38")
new_info <- hg38_chrominfo$size[match(
    chrs_to_keep,
    GenomeInfoDb::mapSeqlevels(hg38_chrominfo$chrom, "Ensembl")
)]
names(new_info) <- chrs_to_keep
gtf_gr_tx <- GenomeInfoDb::keepSeqlevels(gtf_gr,
    chrs_to_keep,
    pruning.mode = "tidy"
)
GenomeInfoDb::seqlengths(gtf_gr_tx) <- new_info
GenomeInfoDb::seqlevelsStyle(gtf_gr_tx) <- "UCSC"
rtracklayer::genome(gtf_gr_tx) <- "hg38"

ucsc_txdb <- GenomicFeatures::makeTxDbFromGRanges(gtf_gr_tx)
genom_state <- derfinder::makeGenomicState(txdb = ucsc_txdb)
ens_txdb <- ucsc_txdb
GenomeInfoDb::seqlevelsStyle(ens_txdb) <- "Ensembl"

annot_ers1 <- annotatERs(
    opt_ers = ex_opt_ers, junc_data = junctions,
    gtf = gtf_gr, txdb = ens_txdb, genom_state = genom_state
)

annot_ers1
```

`file_cache` *Cache a file if it is not found locally*

Description

`file_cache` will use: BiocFileCache and will then cache the file for faster repeated retrival, if it is not found locally (i.e. a URL).

Usage

```
file_cache(file_path)
```

Arguments

`file_path`	a path to file of interest.

Value

file_path of cached file or unchanged file_path if found locally.

Examples

```
rec_url <- recount::download_study(
    project = "SRP012682",
    type = "samples",
    download = FALSE
)

eg_bwfile <- file_cache(rec_url[1])
eg_bwfile
```

`get_chr_info` *Get information from UCSC about the chromosomes passed in*

Description

Download information about each of the chromosomes passed in, most importantly the size.

Usage

```
get_chr_info(chrs, genome)
```

Arguments

`chrs`	chromosomes to look up (must match UCSC format)
`genome`	the UCSC genome to look at see https://genome.ucsc.edu/.

Value

a dataframe with data on the passed in chromosomes

Examples

```
eg_info <- get_chr_info(chrs = c("chr21", "chr22"), genome = "hg38")

eg_info
```

`get_count_matrix` *Generate the count matrix*

Description

Scores the mean coverage of the expressed regions as a count matrix

Usage

```
get_count_matrix(bw_paths, annot_ers, cols = NULL)
```

Arguments

`bw_paths`	Vector containing the bigwig file paths to read in
`annot_ers`	GRangesList containing the annotated ERs (product of `annotatERs`)
`cols`	A dataframe containing the information to be used as colData for the output. If NULL then the bw_paths will be used for the colData

Value

A Ranged Summarized Experiment containing the gene counts as an assay

Examples

```
megadepth::install_megadepth()

rec_url <- recount::download_study(
    project = "SRP012682",
    type = "samples",
    download = FALSE
)
# file_cache is an internal function to download a bigwig file from a link
# if the file has been downloaded recently, it will be retrieved from a cache
bw_path <- file_cache(rec_url[1])

ex_opt_ers <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr1", "chr2"), c(4, 1)),
    ranges = IRanges::IRanges(
        start = c(1:5),
        end = seq(100, 500, 100)
    )
)

example_cm <- get_count_matrix(
    bw_paths = c(bw_path, bw_path),
    annot_ers = ex_opt_ers
)
example_cm
```

get_coverage *Obtain the mean coverage across multiple BigWig files*

Description

get_coverage returns the mean coverage of the BigWig files passed in. Internally, this operates through derfinder::loadCoverage.

Usage

```
get_coverage(
  bw_paths,
  auc_raw,
  auc_target,
  chrs = "",
  genome = "hg38",
  bw_chr = "chr"
)
```

Arguments

bw_paths	path(s) to bigwig file(s) with the RNA-seq data that you want the #' coverage of.
auc_raw	vector containing AUCs(Area Under Coverage) matching the order of bigwig path(s).
auc_target	total AUC to normalise all samples to e.g. 40e6 * 100 would be the estimated total auc for sample sequenced to 40 million reads of 100bp in length.
chrs	chromosomes to obtain mean coverage for, default is "" giving every chromosome. Can take UCSC format(chrs = "chr1") or just the chromosome i.e. chrs = c(1,X)
genome	the UCSC genome you want to use, the default is hg38.
bw_chr	specifies whether the bigwig files has the chromosomes labelled with a "chr" preceding the chromosome i.e. "chr1" vs "1". Can be either "chr" or "nochr" with "chr" being the default.

Value

a list of Rles detailing the mean coverage per chromosome passed in.

Examples

```
rec_url <- recount::download_study(
    project = "SRP012682",
    type = "samples",
    download = FALSE
)
```

```
bw_path <- file_cache(rec_url[1])
# As of rtracklayer 1.25.16, BigWig is not supported on Windows.
if (!xfun::is_windows()) {
    eg_coverage <- get_coverage(
        bw_paths = bw_path,
        auc_raw = 11872688252,
        auc_target = 40e6 * 100,
        chrs = c("chr21", "chr22")
    )
    eg_coverage
}
```

`get_ers` *Define sets of ERs*

Description

`get_ers` defines expressed regions across an inputted range of mean coverage cut-offs (MCCs) and max region gaps (MRGs) from the coverage.

`get_strand_ers` defines ERs across an inputted range of mean coverage cut-offs (MCCs) and max region gaps (MRGs) from the coverage.

Usage

```
get_ers(coverage, mccs, mrgs)

get_strand_ers(
  bw_pos,
  bw_neg,
  auc_raw_pos,
  auc_raw_neg,
  auc_target,
  chrs,
  mccs,
  mrgs,
  bw_chr = "chr"
)
```

Arguments

`coverage`	the coverage of the bigwig files passed into `get_coverage`.
`mccs`	mean coverage cut-offs to apply.
`mrgs`	max region gaps to apply.
`bw_pos`	positive strand bigwig file
`bw_neg`	negative strand bigwig file
`auc_raw_pos`	vector containing AUCs(Area Under Coverage) matching the order of the positive bigwig paths.

auc_raw_neg	vector containing AUCs(Area Under Coverage) matching the order of the negative bigwig paths.
auc_target	total AUC to normalise all samples to. E.g. 40e6 * 100 would be the estimated total auc for sample sequenced to 40 million reads of 100bp in length.
chrs	chromosomes to obtain mean coverage for, default is "" giving every chromosome. Can take UCSC format(chrs = "chr1") or just the chromosome i.e. chrs = c(1,X)
bw_chr	specifies whether the bigwig files has the chromosomes labelled with a "chr" preceding the chromosome i.e. "chr1" vs "1". Can be either "chr" or "nochr" with "chr" being the default.

Value

list containing sets of ERs, each generated using a particular combination of MCC and MRG.

list containing sets of stranded ERs, each generated using a particular combination of MCC and MRG.

Functions

- `get_strand_ers`: Method for getting ers from stranded BigWig files

Examples

```
data(gtex_SRP012682_SRX222703_lung_coverage_1, package = "ODER")

eg_ers <- get_ers(
    coverage = gtex_SRP012682_SRX222703_lung_coverage_1,
    mccs = c(5, 10),
    mrgs = c(10, 20)
)

eg_ers
library("magrittr")
gtex_metadata <- recount::all_metadata("gtex")
gtex_metadata <- gtex_metadata %>%
    as.data.frame() %>%
    dplyr::filter(project == "SRP012682")

rec_url <- recount::download_study(
    project = "SRP012682",
    type = "samples",
    download = FALSE
)
# file_cache is an internal function to download a bigwig file from a link
# if the file has been downloaded recently, it will be retrieved from a cache
bw_plus <- file_cache(rec_url[58])
bw_minus <- file_cache(rec_url[84])

# As of rtracklayer 1.25.16, BigWig is not supported on Windows.
if (!xfun::is_windows()) {
```

```
    stranded_ers <- get_strand_ers(
        bw_pos = bw_plus, bw_neg = bw_minus,
        auc_raw_pos = gtex_metadata[["auc"]][58],
        auc_raw_neg = gtex_metadata[["auc"]][84], auc_target = 40e6 * 100,
        chrs = "chr21", mccs = c(5, 10), mrgs = c(10, 20)
    )
    stranded_ers
}
```

`get_exons` *Obtain set of non-overlapping exons*

Description

Downloads a well-defined set of exons to be used in obtaining the optimum set of Expressed regions. These exons are used in calculating the exon deltas.

Calculates the median exon delta and the number of ERs with an exon delta of 0 by comparing each combination of MCC and MRG with the optimum exons from the ensembl database.

Uses a delta calculating function and a well defined set of exons to find which combination of MCC and MRG gives the best definition of the Expressed regions.

Usage

```
get_exons(gtf, ucsc_chr, ignore.strand = TRUE, biotype = "Non-overlapping")

get_ers_delta(ers, opt_exons, delta_fun = NULL)

get_opt_ers(ers, ers_delta)
```

Arguments

`gtf`	Either a string containg the path to a .gtf file or a pre-imported gtf using `rtracklayer::import` .
`ucsc_chr`	logical scalar, determining whether to add "chr" prefix to the seqnames of non-overlapping exons and change "chrMT" -> "chrM". Note, if set to TRUE and seqnames already have "chr", it will not add another.
`ignore.strand`	logical value for input into `findOverlaps`, default is True.
`biotype`	Filters the GTF file passed in to what would be considered the "Gold Standard" exons. The Default is "Non-overlapping" but the options are: "Non-overlapping" (exons that don't intersect each other), "Three Prime" (3' UTR), "Five Prime" (5' UTR), "Internal" (Internal coding), "lncRNA" (Long Non-Coding RNA), "ncRNA" (Non-Coding RNA) and "Pseudogene"
`ers`	Sets of ERs across various MCCs/MRGs - output of `get_ers`.
`opt_exons`	GRanges object that contains the regions that ideally, you want the ER definitions to match - output of `get_exons`.

delta_fun	Function that calculates the delta between ERs and opt_exons. Takes as input a set of ERs from ers and opt_exons. Then outputs a tibble/dataframe containing the summarised delta scores for that set of one set of ERs.
ers_delta	tibble/dataframe containing summarised delta values. One row per set of ERs.

Value

GRanges object containing non-overlapping exons.

tibble/dataframe containing summarised delta values. One row per set of ERs.

list containing optimised ERs, optimal pair of MCC/MRGs and delta_df

Functions

- get_exons: Filter for the exons to calculate the deltas against
- get_ers_delta: Method to get ers delta to help determine the optimum ers

Examples

```
gtf_url <- paste0(
    "http://ftp.ensembl.org/pub/release-103/gtf/",
    "homo_sapiens/Homo_sapiens.GRCh38.103.chr.gtf.gz"
)
gtf_path <- file_cache(gtf_url)

gtf_gr <- rtracklayer::import(gtf_path)

eg_opt_exons <- get_exons(
    gtf = gtf_gr,
    ucsc_chr = TRUE,
    ignore.strand = TRUE
)


eg_opt_exons
data(gtex_SRP012682_SRX222703_lung_ers_1, package = "ODER")

eg_ers_delta <- get_ers_delta(
    ers = gtex_SRP012682_SRX222703_lung_ers_1,
    opt_exons = eg_opt_exons
)

eg_ers_delta
data(gtex_SRP012682_SRX222703_lung_ers_1, package = "ODER")
opt_ers <- get_opt_ers(
    ers = gtex_SRP012682_SRX222703_lung_ers_1,
    ers_delta = eg_ers_delta
)
opt_ers
```

gtex_SRP012682_SRX222703_lung_auc_1 *An example AUC value*

Description

An Area Under Coverage (AUC) value for a user to try out the package and to pass in for tests. From the GTEX data set and project SRP012682, the actual value is 11872688252.

Usage

```
data(gtex_SRP012682_SRX222703_lung_auc_1)
```

Format

A numeric value

Source

See example.R in data-raw

gtex_SRP012682_SRX222703_lung_coverage_1 *An example object containing coverage*

Description

Coverage generated for a user to try out the package and to pass in for tests. Coverage of chromosomes 21 and 22 from the project SRP012682.

Usage

```
data(gtex_SRP012682_SRX222703_lung_coverage_1)
```

Format

A list of length 2 containing 2 Rles for chromosomes 21 and 22 respectively

Source

See example.R in data-raw

`gtex_SRP012682_SRX222703_lung_erdelta_1`

An example set of ER deltas

Description

This set of deltas was calculated using gtex_lung_ers_1 and exons from ensembl.

Usage

```
data(gtex_SRP012682_SRX222703_lung_erdelta_1)
```

Format

A tibble/dataframe with the sums, means, medians, n_eq_0 and propor_eq_0 for each combination of mccs (5 & 10) and mrgs (10 & 20)

Source

See example.R in data-raw

`gtex_SRP012682_SRX222703_lung_ers_1`

An example set of Expressed Regions

Description

An example set of Expressed Regions generated for a user to try out the package and to pass in for tests. Generated using gtex_SRP012682_SRX222703_lung_coverage_1 and MCCs of 5 and 10 and MRGs of 10 and 20.

Usage

```
data(gtex_SRP012682_SRX222703_lung_ers_1)
```

Format

A list containing two lists (for each mcc) each with a set of genomic ranges for the different combinations of mcc and mrg

Source

See example.R in data-raw

lung_junc_21_22 *Junction data of chromosomes 21 and 22 from a lung tissue sample*

Description

These junctions were sampled from a local junction file.

Usage

```
data(lung_junc_21_22)
```

Format

A dataframe with the junction ID, chromosome, start and ends, strand, number of samples, acceptor and donor

Source

GTEx

ODER *ODER: Optimising the Definition of Expressed Regions*

Description

The aim of ODER is to identify previously unannotated expressed regions (ERs) using RNA-sequencing data. For this purpose, ODER defines and optimises the definition of ERs, then connected these ERs to genes using junction data. In this way, ODER improves gene annotation. Gene annotation is a staple input of many bioinformatic pipelines and a more complete gene annotation can enable more accurate interpretation of disease associated variants.

Returns the optimum definition of the expressed regions by finding the ideal MCC (Mean Coverage Cutoff) and MRG (Max Region Gap). The combination of MCC and MRG that returns the expressed region with the smallest exon delta is the most ideal.

Usage

```
ODER(
  bw_paths,
  auc_raw,
  auc_target,
  chrs = "",
  genome = "hg38",
  mccs,
  mrgs,
  gtf = NULL,
```

```
  ucsc_chr,
  ignore.strand,
  exons_no_overlap = NULL,
  biotype = "Non-overlapping",
  bw_chr = "chr",
  file_type = "non-stranded",
  bw_pos = NULL,
  bw_neg = NULL,
  auc_raw_pos = NULL,
  auc_raw_neg = NULL
)
```

Arguments

bw_paths	path(s) to bigwig file(s) with the RNA-seq data that you want the #' coverage of.
auc_raw	vector containing AUCs(Area Under Coverage) matching the order of bigwig path(s).
auc_target	total AUC to normalise all samples to e.g. 40e6 * 100 would be the estimated total auc for sample sequenced to 40 million reads of 100bp in length.
chrs	chromosomes to obtain mean coverage for, default is "" giving every chromosome. Can take UCSC format(chrs = "chr1") or just the chromosome i.e. chrs = c(1,X)
genome	the UCSC genome you want to use, the default is hg38.
mccs	mean coverage cut-offs to apply.
mrgs	max region gaps to apply.
gtf	Either a string containg the path to a .gtf file or a pre-imported gtf using `rtracklayer::import` .
ucsc_chr	logical scalar, determining whether to add "chr" prefix to the seqnames of non-overlapping exons and change "chrMT" -> "chrM". Note, if set to TRUE and seqnames already have "chr", it will not add another.
ignore.strand	logical value for input into `findOverlaps`, default is True.
exons_no_overlap	Optimum set of exons to help calculate deltas
biotype	Filters the GTF file passed in to what would be considered the "Gold Standard" exons. The Default is "Non-overlapping" but the options are: "Non-overlapping" (exons that don't intersect each other), "Three Prime" (3' UTR), "Five Prime" (5' UTR), "Internal" (Internal coding), "lncRNA" (Long Non-Coding RNA), "ncRNA" (Non-Coding RNA) and "Pseudogene"
bw_chr	specifies whether the bigwig files has the chromosomes labelled with a "chr" preceding the chromosome i.e. "chr1" vs "1". Can be either "chr" or "nochr" with "chr" being the default.
file_type	Describes if the BigWigs are stranded or not. Either "stranded" or non-stranded
bw_pos	positive strand bigwig file

bw_neg	negative strand bigwig file
auc_raw_pos	vector containing AUCs(Area Under Coverage) matching the order of the positive bigwig paths.
auc_raw_neg	vector containing AUCs(Area Under Coverage) matching the order of the negative bigwig paths.

Value

list containing optimised ERs, optimal pair of MCC/MRGs and `delta_df`

Examples

```
rec_url <- recount::download_study(
    project = "SRP012682",
    type = "samples",
    download = FALSE
)

# file_cache is an internal function to download a bigwig file from a link
# if the file has been downloaded recently, it will be retrieved from a cache
bw_path <- file_cache(rec_url[1])
gtf_url <- paste0(
    "http://ftp.ensembl.org/pub/release-103/gtf/",
    "homo_sapiens/Homo_sapiens.GRCh38.103.chr.gtf.gz"
)
gtf_path <- file_cache(gtf_url)

# As of rtracklayer 1.25.16, BigWig is not supported on Windows.
data(gtex_SRP012682_SRX222703_lung_auc_1, package = "ODER")
if (!xfun::is_windows()) {
    opt_ers <- ODER(
        bw_paths = bw_path,
        auc_raw = gtex_SRP012682_SRX222703_lung_auc_1,
        auc_target = 40e6 * 100, chrs = c("chr21", "chr22"),
        genome = "hg38", mccs = c(5, 10), mrgs = c(10, 20),
        gtf = gtf_path, ucsc_chr = TRUE, ignore.strand = TRUE,
        exons_no_overlap = NULL, bw_chr = "chr"
    )

    opt_ers
}
```

---

plot_ers *Plot Expressed regions*

Description

Plots the median deltas and the number of ERs with a delta of 0 against the MCCs on two separate graphs with a line for each of the various MRGs.

Usage

```
plot_ers(ers_delta, opt_mcc_mrg)
```

Arguments

ers_delta	tibble/dataframe containing summarised delta values. One row per set of ERs.
opt_mcc_mrg	vector containing the optimum mcc and mrg, in that order

Value

Plot of MCC against median delta and number of ERS with a delta of 0

Examples

```
data(gtex_SRP012682_SRX222703_lung_erdelta_1, package = "ODER")

eg_plots <- plot_ers(
    ers_delta = gtex_SRP012682_SRX222703_lung_erdelta_1, opt_mcc_mrg = c(
        "mcc_10",
        "mrg_20"
    )
)

eg_plots
```

---

pseudogene *Different transcript biotypes that count as pseudogene*

Description

These are the various transcript biotypes typically found in the transcript biotype column of a gtf file.

Usage

```
data(pseudogene)
```

Format

A character vector with all of the different pseudogene categories `get_exons` function.

Source

See exon_biotypes.R in data-raw

refine_ERs *Refines the ERs start and end points*

Description

Uses the junctions added by annotatERs to modify the starts and ends of the expressed regions. When a junction intersects an expressed region depending on whether it is the start or end or both, the regions corresponding starts and ends will be modified.

Usage

```
refine_ERs(annot_ers)
```

Arguments

annot_ers	ERs that have been annotated (result of annotatER)

Details

As junctions mark intron boundaries, the expressed region will be changed to either being one less or one more than the junction end.

Value

Genomic ranges with refined base pair starts and ends

Examples

```
# create example set of ers to save runtime
ex_annot_ers <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr21"), c(3)),
    ranges = IRanges::IRanges(
        start = c(5093576, 5097663, 5162182),
        end = c(5093833, 5097762, 5162257)
    ),
    grl = GenomicRanges::GRangesList(
        GenomicRanges::GRangesList(
            GenomicRanges::GRanges(
                seqnames = S4Vectors::Rle(c("chr21"), c(1)),
                ranges = IRanges::IRanges(
                    start = c(5093712),
                    end = c(5093744)
                )
            ),
            GenomicRanges::GRanges(
                seqnames = S4Vectors::Rle(c("chr21"), c(1)),
                ranges = IRanges::IRanges(
                    start = c(5097642),
                    end = c(5097669)
```

```
                )
            ),
            GenomicRanges::GRanges(
                seqnames = S4Vectors::Rle(c("chr21"), c(1)),
                ranges = IRanges::IRanges(
                    start = c(5162249),
                    end = c(5162287)
                )
            )
        )
    ),
    annotation = c("intron", "intron", "intron")
)

refined_ers <- refine_ERs(ex_annot_ers)

refined_ers
```

`tissue_options` *The different tissues that can be filtered on for gene expression*

Description

These options were derived from the contents of the GTEx analysis gene median RPKM file.

Usage

```
data(tissue_options)
```

Format

A character vector with all of the tissue options available to filter on. These are to be used in conjunction with the `add_expressed_genes` function.

Source

local data

Index