Package 'BUSpaRse'

April 15, 2020

Type Package

Title kallisto | bustools R utilities

Version 1.0.0

Date 2019-09-11

Description The kallisto | bustools pipeline is a fast and modular set of tools to convert single cell RNA-seq reads in fastq files into gene count or transcript compatibility counts (TCC) matrices for downstream analysis. Central to this pipeline is the barcode, UMI, and set (BUS) file format. This package serves the following purposes: First, this package allows users to manipulate BUS format files as data frames in R and then convert them into gene count or TCC matrices. Furthermore, since R and Rcpp code is easier to handle than pure C++ code, users are encouraged to tweak the source code of this package to experiment with new uses of BUS format and different ways to convert the BUS file into gene count matrix. Second, this package can conveniently generate files required to generate gene count matrices for spliced and unspliced transcripts for RNA velocity. Third, this package implements utility functions to get transcripts and associated genes required to convert BUS files to gene count matrices, to write the transcript to gene information in the format required by bustools, and to read output of bustools into R as sparses matrices.

BugReports https://github.com/BUStools/BUSpaRse/issues

URL https://github.com/BUStools/BUSpaRse

Imports AnnotationDbi, AnnotationFilter, biomaRt, Biostrings, BSgenome, data.table, dplyr, ensembldb, GenomeInfoDb, GenomicFeatures, GenomicRanges, magrittr, Matrix, methods, plyranges, Rcpp, RcppParallel, S4Vectors, stats, stringr, tibble, tidyr, zeallot

LinkingTo Rcpp, RcppArmadillo, RcppProgress, BH, RcppParallel

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown, testthat, BiocStyle, TENxBUSData, DropletUtils, ggplot2, TxDb.Hsapiens.UCSC.hg38.knownGene, BSgenome.Hsapiens.UCSC.hg38, EnsDb.Hsapiens.v86

VignetteBuilder knitr

Collate 'RcppExports.R' 'sparse_matrix.R' 'tr2g.R' 'utils.R' 'velocity.R' 'velocity_methods.R'

Encoding UTF-8

License BSD_2_clause + file LICENSE

biocViews SingleCell, RNASeq, WorkflowStep

SystemRequirements GNU make

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

git_branch RELEASE_3_10

git_last_commit 9b33e52

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Lambda Moses [aut, cre] (<https://orcid.org/0000-0002-7092-9427>),
Lior Pachter [aut, ths] (<https://orcid.org/0000-0002-9164-6231>)

Maintainer Lambda Moses <dlu2@caltech.edu>

R topics documented:

.get_velocity_files *Generate RNA velocity files for GRanges*

Description

Generate RNA velocity files for GRanges

Usage

```
.get_velocity_files(gr, L, Genome, Transcriptome = NULL,
  out_path = ".", style = c("annotation", "genome", "Ensembl", "UCSC",
  "NCBI", "other"), isoform_action = c("separate", "collapse"),
  exon_option = c("full", "junction"), transcript_id = "transcript_id",
  gene_id = "gene_id", transcript_version = "transcript_version",
  gene_version = "gene_version", version_sep = ".",
  compress_fa = FALSE, width = 80L)
```

Arguments

`gr` A `GRanges` object for gene annotation.

`L` Length of the biological read. For instance, 10xv1: 98 nt, 10xv2: 98 nt, 10xv3: 91 nt, Drop-seq: 50 nt. If in doubt check read length in a fastq file for biological reads with the `bash` commands: If the fastq file is gzipped, then do `zcat your_file.fastq.gz | head` on Linux. If on Mac, then `zcat < your_file.fastq.gz | head`. Then you will see lines with nucleotide bases. Copy one of those lines and determine its length with `str_length` in R or `echo -n <the sequence> | wc -c` in bash. Which file corresponds to biological reads depends on the particular technology.

`Genome` Either a `BSgenome` or a `XStringSet` object of genomic sequences, where the intronic sequences will be extracted from. Use `genomeStyles` to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.

`Transcriptome` A `XStringSet`, a path to a fasta file (can be gzipped) of the transcriptome which contains sequences of spliced transcripts, or `NULL`. The transcriptome here will be concatenated with the intronic sequences to give one fasta file. When `NULL`, the transriptome sequences will be extracted from the genome given the gene annotation, so it will be guaranteed that transcript IDs in the transcriptome and in the annotation match. Otherwise, the type of transcript ID in the transcriptome must match that in the gene annotation supplied via argument `X`.

`out_path` Directory to save the outputs written to disk. If this directory does not exist, then it will be created. Defaults to the current working directory.

`style` Formatting of chromosome names. Use `genomeStyles` to check which styles are supported for your organism of interest and what those styles look like. This can also be a style supported for your organism different from the style used by the annotation and the genome. Then this style will be used for both the annotation and the genome. Can take the following values:

- **annotation** If style of the annnotation is different from that of the genome, then the style of the annotation will be used.
- **genome** If style of the annnotation is different from that of the genome, then the style of the genome will be used.
- **other** Custom style, need to manually ensure that the style in annotation matches that of the genome.
- **Ensembl** Or `UCSC` or `NCBI`, whichever is supported by your species of interest.

`isoform_action` Character, indicating action to take with different transcripts of the same gene. Must be one of the following:

- **collapse** First, the union of all exons of different transcripts of a gene will be taken. Then the introns will be inferred from this union. Only the flanked intronic sequences are affected; isoforms will always be taken into account for spliced sequences or exon-exon junctions.
- **separate** Introns from different transcripts will be kept separate.

`exon_option` Character, indicating how exonic sequences should be included in the kallisto index. Must be one of the following:

- **full** The full cDNA sequences, which include the full exonic sequences, will be used. This is the default.
- **junction** Only the exon-exon junctions, with L-1 bases on each side of the junctions, will be used.

`transcript_id` Character vector of length 1. Tag in `attribute` field corresponding to transcript IDs. This argument must be supplied and cannot be `NA` or `NULL`. Will throw error if tag indicated in this argument does not exist.

`gene_id` Character vector of length 1. Tag in `attribute` field corresponding to gene IDs. This argument must be supplied and cannot be `NA` or `NULL`. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

`transcript_version`

Character vector of length 1. Tag in `attribute` field corresponding to *transcript* version number. If your GTF file does not include transcript version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. To decide whether to include transcript version number, check whether version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`gene_version` Character vector of length 1. Tag in `attribute` field corresponding to *gene* version number. If your GTF file does not include gene version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. Unlike transcript version number, it's up to you whether to include gene version number.

`version_sep` Character to separate bewteen the main ID and the version number. Defaults to ".", as in Ensembl.

`compress_fa` Logical, whether to compress the output fasta file of transcriptome and flanked intronic sequenncess. If `TRUE`, then the fasta file will be gzipped.

`width` Maximum number of letters per line of sequence in the output fasta file. Must be an integer.

Value

See get_velocity_files

---

annot_circular — *Transfer information about circular chromosomes between genome and annotation*

Description

Internal use, called after calling subset_annot.

Usage

```
annot_circular(Genome, annot)
```

Arguments

Genome	Either a BSgenome or a XStringSet object of genomic sequences, where the intronic sequences will be extracted from. Use genomeStyles to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.
annot	Genome annotation, an object of a class with a seqlevels method, such as GRanges, TxDb, and EnsDb.

Value

If neither genome nor annotation indicates which chromosome is circular, then the input will be returned unchanged. If only one of genome and annotation has such information, then it will be transferred to the one that does not. If both do have such information, the information from the genome will be transferred to the annotation if they're different.

---

check_char1 — *Check that an object is a character vector of length 1*

Description

Just in case the user passes something with length more than 1 and messes up everything thanks to vectorization.

Usage

```
check_char1(x)
```

Arguments

x	Named vector of arguments to be checked.

Value

Error if x is not a character vector with length 1.

check_genome *Check for chromosomes in genome but not annotation*

Description

Check for chromosomes in genome but not annotation

Usage

```
check_genome(chrs_use, Genome)
```

Arguments

chrs_use	Character vector of names of chromosomes present in both the annotation and the genome.
Genome	Either a BSgenome or a XStringSet object of genomic sequences, where the intronic sequences will be extracted from. Use genomeStyles to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.

Value

Nothing. Will emit message if the genome contains chromosomes absent from the annotation.

check_gff *Check inputs to tr2g_gtf and tr2g_gff3*

Description

This function validates inputs to tr2g_gtf and tr2g_gff3 and throws error early if some inputs are wrong.

Usage

```
check_gff(format, file, transcript_id, gene_id)
```

Arguments

format	Whether it's gtf or gff3.
file	Path to a GTF file to be read. The file can remain gzipped.
transcript_id	Character vector of length 1. Tag in `attribute` field corresponding to transcript IDs. This argument must be supplied and cannot be `NA` or `NULL`. Will throw error if tag indicated in this argument does not exist.
gene_id	Character vector of length 1. Tag in `attribute` field corresponding to gene IDs. This argument must be supplied and cannot be `NA` or `NULL`. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

Value

Nothing, will throw error if there's a problem.

`check_tag_present` *Check that a tag is present in attribute field of GTF/GFF*

Description

The attribute field of GTF/GFF files are very complicated and is very inconsistent between sources. This function is to make sure that transcript and gene IDs can be extracted properly.

Usage

```
check_tag_present(tags_use, tags, error = TRUE)
```

Arguments

`tags_use` The tags to be checked.

`tags` The tags present in attribute field.

`error` Whether to throw an error for absent tags. If `FALSE`, then a warning will be given.

Value

Error or warning if tag is absent.

`check_tx` *Check if transcript ID in transcriptome and annotation match*

Description

This function throws an error if transcript IDs in transcriptome and annotation do not overlap. If they do overlap, this function will give a message about transcript IDs that do not agree in the transcriptome and the annotation

Usage

```
check_tx(tx_annot, tx)
```

Arguments

`tx_annot` Character vector of transcript IDs from the annotation.

`tx` Character vector of transcript IDs from the transcriptome.

Value

Character vector of the overlapping transcript IDs.

EC2gene *Map EC Index to Genes Compatible with the EC*

Description

In the output file `output.bus`, equivalence classes (EC) are denoted by an index, which is related to the set of transcripts the EC is compatible to in the output file `matrix.ec`. This function further relates the set of transcripts to the set of genes the EC is compatible to. This function first reads in `matrix.ec`, and then translates the transcripts into genes.

Usage

```
EC2gene(tr2g, kallisto_out_path, ncores = 0, verbose = TRUE)
```

Arguments

`tr2g` A Data frame with columns `gene` and `transcript`, in the same order as in the transcriptome index for `kallisto`.

`kallisto_out_path` Path to the `kallisto` bus output directory.

`ncores` Number of cores to use, defaults to 0, which means the system will automatically determine the number of cores as it sees fit. Negative numbers are interpreted as 0. Positive numbers will limit the number of cores used. This might not speed up `EC2gene` very much unless there are many genes or ECs detected.

`verbose` Whether to display progress. Defaults to `TRUE`.

Details

The data frame passed to `tr2g` can be generated from function `transcript2gene` in this package for any organism that has gene and transcript ID on Ensembl, or from the `tr2g_*` family of function. You no longer need to use this function before running `make_sparse_matrix`; the purpose of this function is to query which genes equivalence classes map to.

Calling this function is unnessary when working with gene count matrices. However, this function is useful for finding genes the ECs map to in TCC matrices, such as when finding species-specific ECs in mixed species datasets and identifying ECs mapped to known marker genes of cell types.

Value

A data frame with 3 columns:

EC_ind Index of the EC as appearing in the `matrix.ec` file.

EC A list column each element of which is a numeric vector of the transcripts in the EC corresponding to the EC index. To learn more about list columns, see the relevant section in the R for Data Science book.

gene A list column each element of which is a character vector of genes the EC maps to.

See Also

`transcript2gene`

Examples

```
# Load toy example for testing
toy_path <- system.file("testdata", package = "BUSpaRse")
load(paste(toy_path, "toy_example.RData", sep = "/"))
EC2gene(tr2g_toy, toy_path, verbose = FALSE, ncores = 1)
```

`get_intron_flanks` *Get flanked intronic ranges*

Description

Get flanked intronic ranges

Usage

```
get_intron_flanks(grl, L, get_junctions)
```

Arguments

`grl`	A `CompressedGRangesList` for exonic ranges, each element for one transcript.
`L`	Read length.
`get_junctions`	Logical, whether to also return exon-exon junctions.

Value

If `get_junctions` is `FALSE`, then a `GRanges` object with ranges for flanked intronic regions. If `get_junctions` is `TRUE`, then in addition to the flanked intronic ranges, a `CompressedGRangesList` with exon-exon junction ranges and ranges for transcripts without introns.

`get_velocity_files` *Get files required for RNA velocity with bustools*

Description

Computation of RNA velocity requires the number of unspliced transcripts, which can be quantified with reads containing intronic sequences. This function extracts intronic sequences flanked by L-1 bases of exonic sequences where L is the biological read length of the single cell technology of interest. The flanking exonic sequences are included for reads partially mapping to an intron and an exon.

Usage

```
get_velocity_files(X, L, Genome, Transcriptome = NULL, out_path = ".",
  style = c("annotation", "genome", "Ensembl", "UCSC", "NCBI", "other"),
  isoform_action = c("separate", "collapse"), exon_option = c("full",
  "junction"), compress_fa = FALSE, width = 80L, ...)

## S4 method for signature 'GRanges'
get_velocity_files(X, L, Genome,
  Transcriptome = NULL, out_path = ".", style = c("annotation",
  "genome", "Ensembl", "UCSC", "NCBI", "other"),
  isoform_action = c("separate", "collapse"), exon_option = c("full",
  "junction"), compress_fa = FALSE, width = 80L,
  transcript_id = "transcript_id", gene_id = "gene_id",
  transcript_version = "transcript_version",
  gene_version = "gene_version", version_sep = ".")

## S4 method for signature 'character'
get_velocity_files(X, L, Genome,
  Transcriptome = NULL, out_path = ".", style = c("annotation",
  "genome", "Ensembl", "UCSC", "NCBI", "other"),
  isoform_action = c("separate", "collapse"), exon_option = c("full",
  "junction"), compress_fa = FALSE, width = 80L, is_circular = NULL,
  transcript_id = "transcript_id", gene_id = "gene_id",
  transcript_version = "transcript_version",
  gene_version = "gene_version", version_sep = ".")

## S4 method for signature 'TxDb'
get_velocity_files(X, L, Genome, Transcriptome, out_path,
  style = c("annotation", "genome", "Ensembl", "UCSC", "NCBI", "other"),
  isoform_action = c("separate", "collapse"), exon_option = c("full",
  "junction"), compress_fa = FALSE, width = 80L)

## S4 method for signature 'EnsDb'
get_velocity_files(X, L, Genome, Transcriptome, out_path,
  style = c("annotation", "genome", "Ensembl", "UCSC", "NCBI", "other"),
  isoform_action = c("separate", "collapse"), exon_option = c("full",
  "junction"), compress_fa = FALSE, width = 80L,
  use_transcript_version = TRUE, use_gene_version = TRUE)
```

Arguments

X — Gene annotation with transcript and exon information. It can be a path to a GTF file with annotation of exon coordinates of transcripts, preferably from Ensembl. In the metadata, the following fields are required: type (e.g. whether the range of interest is a gene or transcript or exon or CDS), gene ID, and transcript ID. These fields need not to have standard names, as long as their names are specified in arguments of this function. It can also be a TxDb object, such as from the Bioconductor package TxDb.Hsapiens.UCSC.hg38.knownGene. It can also be a EnsDb object.

L — Length of the biological read. For instance, 10xv1: 98 nt, 10xv2: 98 nt, 10xv3: 91 nt, Drop-seq: 50 nt. If in doubt check read length in a fastq file for biological reads with the bash commands: If the fastq file is gzipped, then do zcat your_file.fastq.gz | head on Linux. If on Mac, then zcat < your_file.fastq.gz | head. Then you will see lines with nucleotide bases. Copy one of those lines and determine its length with str_length in R or echo -n <the sequence> | wc -c in bash. Which file corresponds to biological reads depends on the particular technology.

Genome
: Either a BSgenome or a XStringSet object of genomic sequences, where the intronic sequences will be extracted from. Use genomeStyles to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.

Transcriptome
: A XStringSet, a path to a fasta file (can be gzipped) of the transcriptome which contains sequences of spliced transcripts, or NULL. The transcriptome here will be concatenated with the intronic sequences to give one fasta file. When NULL, the transriptome sequences will be extracted from the genome given the gene annotation, so it will be guaranteed that transcript IDs in the transcriptome and in the annotation match. Otherwise, the type of transcript ID in the transcriptome must match that in the gene annotation supplied via argument X.

out_path
: Directory to save the outputs written to disk. If this directory does not exist, then it will be created. Defaults to the current working directory.

style
: Formatting of chromosome names. Use genomeStyles to check which styles are supported for your organism of interest and what those styles look like. This can also be a style supported for your organism different from the style used by the annotation and the genome. Then this style will be used for both the annotation and the genome. Can take the following values:

annotation If style of the annnotation is different from that of the genome, then the style of the annotation will be used.

genome If style of the annnotation is different from that of the genome, then the style of the genome will be used.

other Custom style, need to manually ensure that the style in annotation matches that of the genome.

Ensembl Or UCSC or NCBI, whichever is supported by your species of interest.

isoform_action
: Character, indicating action to take with different transcripts of the same gene. Must be one of the following:

collapse First, the union of all exons of different transcripts of a gene will be taken. Then the introns will be inferred from this union. Only the flanked intronic sequences are affected; isoforms will always be taken into account for spliced sequences or exon-exon junctions.

separate Introns from different transcripts will be kept separate.

exon_option
: Character, indicating how exonic sequences should be included in the kallisto index. Must be one of the following:

full The full cDNA sequences, which include the full exonic sequences, will be used. This is the default.

junction Only the exon-exon junctions, with L-1 bases on each side of the junctions, will be used.

compress_fa
: Logical, whether to compress the output fasta file of transcriptome and flanked intronic sequenncess. If TRUE, then the fasta file will be gzipped.

width
: Maximum number of letters per line of sequence in the output fasta file. Must be an integer.

`...` Extra arguments for methods.

`transcript_id` Character vector of length 1. Tag in `attribute` field corresponding to transcript IDs. This argument must be supplied and cannot be `NA` or `NULL`. Will throw error if tag indicated in this argument does not exist.

`gene_id` Character vector of length 1. Tag in `attribute` field corresponding to gene IDs. This argument must be supplied and cannot be `NA` or `NULL`. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

`transcript_version`
Character vector of length 1. Tag in `attribute` field corresponding to *transcript* version number. If your GTF file does not include transcript version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. To decide whether to include transcript version number, check whether version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`gene_version` Character vector of length 1. Tag in `attribute` field corresponding to *gene* version number. If your GTF file does not include gene version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. Unlike transcript version number, it's up to you whether to include gene version number.

`version_sep` Character to separate bewteen the main ID and the version number. Defaults to ".", as in Ensembl.

`is_circular` Logical vector of the same length as the number of sequences in the annotation and with the same names as the sequences, indicating whether the sequence is circular. If `NULL`, then all sequences will be assumed to be linear.

`use_transcript_version`
Logical, whether to include version number in the Ensembl transcript ID.

`use_gene_version`
Logical, whether to include version number in the Ensembl gene ID. Unlike transcript version number, it's up to you whether to include gene version number.

Value

The following files will be written to disk in the directory `out_path`:

cDNA_introns.fa A fasta file containing both the spliced transcripts and the flanked intronic sequences. The intronic sequences are flanked by L-1 nt of exonic sequences to capture reads from nascent transcript partially mapping to exons. If the exon is shorter than 2*(L-1) nt, then the entire exon will be included in the intronic sequence. This will be used to build the `kallisto` index.

cDNA_tx_to_capture.txt A text file of transcript IDs of spliced transcripts. If `exon_option == "junction"`, then IDs of the exon-exon junctions. These IDs will have the pattern <transcript ID>-Jx, where x is a number differentiating between different junctions of the same transcript. Here x will always be ordered from 5' to 3' as on the plus strand.

introns_tx_to_capture.txt A text file of IDs of introns. The names will have the pattern <transcript ID>-Ix, where x is a number differentiating between introns of the same transcript. If all transcripts of the same gene are collapsed before inferring intronic sequences, gene ID will be used in place of transcript ID. Here x will always be ordered from 5' to 3' as on the plus strand.

tr2g.txt A text file with two columns matching transcripts and introns to genes. The first column is transcript or intron ID, and the second column is the corresponding gene ID. The part for transcripts are generated from the gene annotation supplied.

Nothing is returned into the R session.

Examples

```
# Use toy example
toy_path <- system.file("testdata", package = "BUSpaRse")
file <- paste0(toy_path, "/velocity_annot.gtf")
genome <- Biostrings::readDNAStringSet(paste0(toy_path, "/velocity_genome.fa"))
transcriptome <- paste0(toy_path, "/velocity_tx.fa")
get_velocity_files(file, 11, genome, transcriptome, ".",
  gene_version = NULL, transcript_version = NULL)
```

make_sparse_matrix *Convert the Output of* kallisto bus *into Gene by Gell Matrix*

Description

This function takes the output file of kallisto bus, after being sorted and converted into text with bustools. See vignettes on the website of this package for a tutorial. The bustools output has 4 columns: barcode, UMI, equivalence class, and counts. This function converts that file into a sparse matrix that can be used in downstream analyses.

Usage

```
make_sparse_matrix(bus_path, tr2g, est_ncells, est_ngenes,
  whitelist = NULL, gene_count = TRUE, TCC = TRUE,
  single_gene = TRUE, ncores = 0, verbose = TRUE,
  progress_unit = 5e+06)
```

Arguments

`bus_path` Path to the sorted text bus output file.

`tr2g` A Data frame with columns gene and transcript, in the same order as in the transcriptome index for kallisto. This argument can be missing or is ignored if only the TCC matrix, not the gene count matrix, is made.

`est_ncells` Estimated number of cells; providing this argument will speed up computation as it minimizes memory reallocation as vectors grow.

`est_ngenes` Estimated number of genes or equivalence classes.

`whitelist` A character vector with valid cell barcodes. This is an optional argument, that defaults to NULL. When it is NULL, all cell barcodes present that have some UMI assignable to genes or ECs will be included in the sparse matrix whether they are known to be valid or not. Barcodes with only UMIs that are not assignable to genes or ECs will still be excluded.

`gene_count` Logical, whether the gene count matrix should be returned.

`TCC` Logical, whether the TCC matrix should be returned.

`single_gene` Logical, whether to use single gene mode. In single gene mode, only UMIs that can be uniquely mapped to one gene are kept. Without single gene mode, UMIs mapped to multiple genes will be evenly distributed to those genes.

`ncores` Number of cores to use, defaults to 0, which means the system will automatically determine the number of cores as it sees fit. Negative numbers are interpreted as 0. Positive numbers will limit the number of cores used. This might not speed up `EC2gene` very much unless there are many genes or ECs detected.

`verbose` Whether to display progress.

`progress_unit` How many iteration to print one progress update when reading in the `kallisto bus` file.

Details

This function can generate both the gene count matrix and the transcript compatibility count (TCC) matrix. The TCC matrix has barcodes in the columns and equivalence classes in the rows. See Ntranos et al. 2016 for more information about the RCC matrix.

For 10x data sets, you can find a barcode whitelist file that comes with CellRanger installation. You don't need to run CellRanger to get that. An example path to get the whitelist file is `cellranger-2.1.0/cellranger-cs/` for v2 chemistry.

Value

If both gene count and TCC matrices are returned, then this function returns a list with two matrices, each with genes/equivalence classes in the rows and barcodes in the columns. If only one of gene count and TCC matrices is returned, then a `dgCMatrix` with genes/equivalence classes in the rows and barcodes in the columns. These matrices are unfiltered. Please filter the empty droplets before downstream analysis.

See Also

`EC2gene`

Examples

```
# Load toy example for testing
toy_path <- system.file("testdata", package = "BUSpaRse")
load(paste(toy_path, "toy_example.RData", sep = "/"))
out_fn <- paste0(toy_path, "/output.sorted.txt")
# With whitelist
m <- make_sparse_matrix(out_fn, tr2g_toy, 10, 3, whitelist = whitelist,
  gene_count = TRUE, TCC = FALSE, single_gene = TRUE,
  verbose = FALSE)
```

match_style — *Match chromosome naming styles of annotation and genome*

Description

Internal use. This function matches chromosome naming styles. It will also give the genome and the annotation the same genome slot. This function assumes that the annotation and the genome refer to the same version of genome. If more than one style, then the first element will be used.

Usage

```
match_style(Genome, annot, style)
```

Arguments

Genome	Either a BSgenome or a XStringSet object of genomic sequences, where the intronic sequences will be extracted from. Use genomeStyles to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.
annot	Genome annotation, an object of a class with a seqlevels method, such as GRanges, TxDb, and EnsDb.
style	Formatting of chromosome names. Use genomeStyles to check which styles are supported for your organism of interest and what those styles look like. This can also be a style supported for your organism different from the style used by the annotation and the genome. Then this style will be used for both the annotation and the genome. Can take the following values: **annotation** If style of the annnotation is different from that of the genome, then the style of the annotation will be used. **genome** If style of the annnotation is different from that of the genome, then the style of the genome will be used. **other** Custom style, need to manually ensure that the style in annotation matches that of the genome. **Ensembl** Or UCSC or NCBI, whichever is supported by your species of interest.

Value

A list of two. The first element is the genome with the proper style, and the second element is the annotation with the proper style.

read_count_output *Read matrix along with barcode and gene names*

Description

This function takes in a directory and name and reads the mtx file, genes, and barcodes from the output of bustools to return a sparse matrix with column names and row names.

Usage

```
read_count_output(dir, name, tcc = TRUE)
```

Arguments

dir	Directory with the bustools count outputs.
name	The files in the output directory should be <name>.mtx, <name>.genes.txt, and <name>.barcodes.txt.
tcc	Logical, whether the matrix of interest is a TCC matrix. Defaults to FALSE.

Value

A dgCMatrix with barcodes as column names and genes as row names.

Examples

```
# Internal toy data used for unit testing
toy_path <- system.file("testdata", package = "BUSpaRse")
m <- read_count_output(toy_path, name = "genes", tcc = FALSE)
```

read_velocity_output *Read intronic and exonic matrices into R*

Description

Read intronic and exonic matrices into R

Usage

```
read_velocity_output(spliced_dir, unspliced_dir, spliced_name,
  unspliced_name)
```

Arguments

spliced_dir	Directory with mtx file for UMI counts of spliced transcripts.
unspliced_dir	Directory with mtx file for UMI counts of unspliced transcripts.
spliced_name	The files in the splicedd directory should be <spliced_name>.mtx, <spliced_name>.genes.txt, and <spliced_name>.barcodes.txt.
unspliced_name	The files in the unsplicedd directory should be <unspliced_name>.mtx, <unspliced_name>.genes.txt, and <unspliced_name>.barcodes.txt.

Value

A list of two dgCMatrix with barcodes as column names and genes as row names. The elements of the list will be `spliced` and `unspliced`.

Examples

```
# Internal toy data used for unit testing
toy_path <- system.file("testdata", package = "BUSpaRse")
m <- read_velocity_output(toy_path, toy_path,
  spliced_name = "genes",
  unspliced_name = "genes")
```

`save_tr2g_bustools` *Save transcript to gene file for use in* `bustools`

Description

This function saves the transcript to gene data frame generated by this package in whatever means in a format required by `bustools`. In order to use `bustools` to generate the gene count or TCC matrix, a file that maps transcripts to genes is required. This should be a tsv file with 2 columns: the first column for transcript ID and the second for gene ID. The order of transcripts in this file must be the same as the order in the kallisto index, and this ordering can be ensured by the function `sort_tr2g`. There must also be no headers. All columns other than `transcript` and gene will be discarded. To save a file with those columns, directly save the transcript to gene data frame with function like `write.table`, `readr::write_delim`, and `fwrite`.

Usage

```
save_tr2g_bustools(tr2g, file_save = "./tr2g.tsv", ...)
```

Arguments

`tr2g`	The data frame output from the `tr2g_*` family of functions.
`file_save`	File name of the file to be saved. The directory in which the file is to be saved must exist.
`...`	Other arguments passed to `fwrite`, such as `sep`, `quote`, and `col.names`.

Value

Nothing is returned into the R session. A tsv file of the format required by `bustools` with the name and directory specified will be written to disk.

Examples

```
toy_path <- system.file("testdata", package = "BUSpaRse")
file_use <- paste(toy_path, "gtf_test.gtf", sep = "/")
tr2g <- tr2g_gtf(file = file_use, verbose = FALSE)
save_tr2g_bustools(tr2g, file_save = "./tr2g.tsv")
```

sort_tr2g *Sort transcripts to the same order as in kallisto index*

Description

This function takes the data frame output from the tr2g_* family of functions in this package as the input, and sorts it so the transcripts are in the same order as in the kallisto index used to generate the bus file. Sorting is vital to obtain the correct sparse matrix from the bus file as equivalence class notations are based on the index of transcripts in the kallisto index.

Usage

```
sort_tr2g(tr2g, file, kallisto_out_path, verbose = TRUE)
```

Arguments

tr2g The data frame output from the tr2g_* family of functions.

file Character vector of length 1, path to a tsv file with transcript IDs and the corresponding gene IDs, in the format required for bustools, or written by save_tr2g_bustools.

kallisto_out_path Character vector of length 1, path to the directory for the outputs of kallisto bus.

verbose Whether to display progress.

Details

Since the attribute field of GTF and GFF3 files varies across sources, output from tr2g_gtf and tr2g_gff3 may need further clean up. You may also supply gene and transcript IDs from other sources. This function should be used after the clean up, when the transcript IDs in the cleaned up data frame have the same format as those in transcript

Value

A data frame with columns transcript and gene and the other columns present in tr2g or the data frame in file, with the transcript IDs sorted to be in the same order as in the kallisto index.

See Also

Other functions to retrieve transcript and gene info: tr2g_EnsDb, tr2g_TxDb, tr2g_ensembl, tr2g_fasta, tr2g_gff3, tr2g_gtf, transcript2gene

Examples

```
toy_path <- system.file("testdata", package = "BUSpaRse")
file_use <- paste(toy_path, "gtf_test.gtf", sep = "/")
tr2g <- tr2g_gtf(file = file_use, verbose = FALSE,
  transcript_version = NULL)
tr2g <- sort_tr2g(tr2g, kallisto_out_path = toy_path, verbose = FALSE)
```

species2dataset *Convert Latin species name to dataset name*

Description

This function converts Latin species name to a dataset name in biomart to query gene and transcript ID.

Usage

```
species2dataset(species, type = c("vertebrate", "metazoa", "plant",
  "fungus", "protist"))
```

Arguments

species	A character vector of Latin names of species present in this scRNA-seq dataset. This is used to retrieve Ensembl information from biomart.
type	A character vector indicating the type of each species. Each element must be one of "vertebrate", "metazoa", "plant", "fungus", and "protist". If length is 1, then this type will be used for all species specified here. Can be missing if `fasta_file` is specified.

Value

The appropriate dataset name for biomart.

Examples

```
species2dataset(species = "Homo sapiens")
```

standardize_tags *Standardize GRanges field names*

Description

To avoid introducing rlang as another dependency for tidyeval. This function will also convert exon numbers to integer.

Usage

```
standardize_tags(gr, gene_id, transcript_id)
```

Arguments

gr	A `GRanges` object.
gene_id	Name of the metadata field for gene ID.
transcript_id	Name of the metadata field for transcript ID.

Value

A `GRanges` object with standardized names: gene ID as `gene_id`, and transcript ID as `transcript_id`.

subset_annot *Subset genome annotation*

Description

Exclude chromosomes present in the annotation but absent from the genome and add information about circular chromosomes.

Usage

```
subset_annot(Genome, annot)

## S4 method for signature 'DNAStringSet'
subset_annot(Genome, annot)

## S4 method for signature 'BSgenome'
subset_annot(Genome, annot)
```

Arguments

Genome	Either a BSgenome or a XStringSet object of genomic sequences, where the intronic sequences will be extracted from. Use genomeStyles to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.
annot	Either a GRanges object or a TxDb object for gene annotation.

Value

A subsetted genome annotation of the same type ofo the input genome annotation.

sub_annot *Remove chromosomes in anotation absent from genome*

Description

Remove chromosomes in anotation absent from genome

Usage

```
sub_annot(chrs_use, annot)
```

Arguments

chrs_use	Character vector of names of chromosomes present in both the annotation and the genome.
annot	Either a GRanges object or a TxDb object for gene annotation.

Value

A subsetted genome annotation of the same type ofo the input genome annotation.

---

tr2g_EnsDb — *Get transcript and gene info from EnsDb objects*

Description

Bioconductor provides Ensembl genome annotation in AnnotationHub; older versions of Ensembl annotation can be obtained from packages like EnsDb.Hsapiens.v86. This is an alternative to querying Ensembl with biomart; Ensembl's server seems to be less stable than that of Bioconductor. However, more information and species are available on Ensembl biomart than on AnnotationHub.

Usage

```
tr2g_EnsDb(ensdb, other_attrs = NULL, use_gene_name = TRUE,
  use_transcript_version = TRUE, use_gene_version = TRUE)
```

Arguments

ensdb
: Ann EnsDb object, such as from AnnotationHub or EnsDb.Hsapiens.v86.

other_attrs
: Character vector. Other attributes to get from the EnsDb object, such as gene symbol and position on the genome. Use columns to see which attributes are available.

use_gene_name
: Logical, whether to get gene names.

use_transcript_version
: Logical, whether to include version number in the Ensembl transcript ID. To decide whether to include transcript version number, check whether version numbers are included in the transcripts.txt in the kallisto output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

use_gene_version
: Logical, whether to include version number in the Ensembl gene ID. Unlike transcript version number, it's up to you whether to include gene version number.

Value

A data frame with at least 2 columns: gene for gene ID, transcript for transcript ID, and optionally gene_name for gene names. If other_attrs has been specified, then those will also be columns in the data frame returned.

See Also

Other functions to retrieve transcript and gene info: sort_tr2g, tr2g_TxDb, tr2g_ensembl, tr2g_fasta, tr2g_gff3, tr2g_gtf, transcript2gene

Examples

```
library(EnsDb.Hsapiens.v86)
tr2g_EnsDb(EnsDb.Hsapiens.v86, use_transcript_version = FALSE,
  use_gene_version = FALSE)
```

`tr2g_ensembl` *Get transcript and gene info from Ensembl*

Description

This function queries Ensembl biomart to convert transcript IDs to gene IDs.

Usage

```
tr2g_ensembl(species, type = c("vertebrate", "metazoa", "plant",
  "fungus", "protist"), other_attrs = NULL, use_gene_name = TRUE,
  use_transcript_version = TRUE, use_gene_version = TRUE,
  ensembl_version = NULL, verbose = TRUE, ...)
```

Arguments

`species` Character vector of length 1, Latin name of the species of interest.

`type` Character, must be one of "vertebrate", "metazoa", "plant", "fungus" and "protist". Passing "vertebrate" will use the default www.ensembl.org host. Gene annotation of some common invertebrate model organisms, such as *Drosophila melanogaster*, are available on www.ensembl.org so for these invertebrate model organisms, "vertebrate" can be used for this argument. Passing values other than "vertebrate" will use other Ensembl hosts. For animals absent from www.ensembl.org, try "metazoa".

`other_attrs` Character vector. Other attributes to get from Ensembl, such as gene symbol and position on the genome. Use `listAttributes` to see which attributes are available.

`use_gene_name` Logical, whether to get gene names.

`use_transcript_version` Logical, whether to include version number in the Ensembl transcript ID. To decide whether to include transcript version number, check whether version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`use_gene_version` Logical, whether to include version number in the Ensembl gene ID. Unlike transcript version number, it's up to you whether to include gene version number.

`ensembl_version` Integer version number of Ensembl (e.g. 94 for the October 2018 release). This argument defaults to `NULL`, which will use the current release of Ensembl. Use `listEnsemblArchives` to see the version number corresponding to the Ensembl release of a particular date. The version specified here must match the version of Ensembl where the transcriptome used to build the kallisto index was downloaded.

`verbose` Whether to display progress.

`...` Othe arguments to be passed to `useEnsembl`, such as mirror. Note that setting mirrors other than the default, e.g. uswest, does not work for archived versions.

Value

A data frame with at least 2 columns: gene for gene ID, `transcript` for transcript ID, and optionally `gene_name` for gene names. If `other_attrs` has been specified, then those will also be columns in the data frame returned.

See Also

Other functions to retrieve transcript and gene info: `sort_tr2g`, `tr2g_EnsDb`, `tr2g_TxDb`, `tr2g_fasta`, `tr2g_gff3`, `tr2g_gtf`, `transcript2gene`

Examples

```
tr2g <- tr2g_ensembl(species = "Felis catus", other_attrs = "description")
# This will use plants.ensembl.org as host instead of www.ensembl.org
tr2g <- tr2g_ensembl(species = "Arabidopsis thaliana", type = "plant")
```

`tr2g_fasta` *Get transcript and gene info from names in FASTA files*

Description

FASTA files, such as those for cDNA and ncRNA from Ensembl, might have genome annotation information in the name of each sequence entry. This function extracts the transcript and gene IDs from such FASTA files.

Usage

```
tr2g_fasta(file, use_gene_name = TRUE, use_transcript_version = TRUE,
  use_gene_version = TRUE, verbose = TRUE)
```

Arguments

`file` Path to the FASTA file to be read. The file can remain gzipped.

`use_gene_name` Logical, whether to get gene names.

`use_transcript_version`
Logical, whether to include version number in the Ensembl transcript ID. To decide whether to include transcript version number, check whether version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`use_gene_version`
Logical, whether to include version number in the Ensembl gene ID. Unlike transcript version number, it's up to you whether to include gene version number.

`verbose` Whether to display progress.

Details

At present, this function only works with FASTA files from Ensembl, and uses regex to extract vertebrate Ensembl IDs. Sequence names should be formatted as follows:

```
ENST00000632684.1 cdna chromosome:GRCh38:7:142786213:142786224:1
gene:ENSG00000282431.1 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene
gene_symbol:TRBD1 description:T cell receptor beta diversity 1
[Source:HGNC Symbol;Acc:HGNC:12158]
```

If your FASTA file sequence names are formatted differently, then you must extract the transcript and gene IDs by some other means. The Bioconductor package `Biostrings` is recommended; after reading the FASTA file into R, the sequence names can be accessed by the `names` function.

While normally, you should call `sort_tr2g` to sort the transcript IDs from the output of the `tr2g_*` family of functions, If the FASTA file supplied here is the same as the one used to build the kallisto index, then the transcript IDs in the output of this function are in the same order as in the kallisto index, so you can skip `sort_tr2g` and proceed directly to `EC2gene` with the output of this function.

Value

A data frame with at least 2 columns: `gene` for gene ID, `transcript` for transcript ID, and optionally `gene_name` for gene names.

See Also

Other functions to retrieve transcript and gene info: `sort_tr2g`, `tr2g_EnsDb`, `tr2g_TxDb`, `tr2g_ensembl`, `tr2g_gff3`, `tr2g_gtf`, `transcript2gene`

Examples

```
toy_path <- system.file("testdata", package = "BUSpaRse")
file_use <- paste(toy_path, "fasta_test.fasta", sep = "/")
tr2g <- tr2g_fasta(file = file_use, verbose = FALSE)
```

---

tr2g_gff3 *Get transcript and gene info from GFF3 file*

Description

This function reads a GFF3 file and extracts the transcript ID and corresponding gene ID. This function assumes that the GFF3 file is properly formatted. See http://gmod.org/wiki/GFF3 for a detailed description of proper GFF3 format. Note that GTF files have a somewhat different and simpler format in the attribute field, which this function does not support. See http://mblab.wustl.edu/GTF2.html for a detailed description of proper GTF format. To extract transcript and gene information from GTF files, see the function `tr2g_gtf` in this package. Some files bearing the `.gff3` are in fact more like the GTF format. If this is so, then change the extension to `.gtf` and use the function `tr2g_gtf` in this package instead.

Usage

```
tr2g_gff3(file, type_use = "mRNA", transcript_id = "transcript_id",
  gene_id = "gene_id", gene_name = "Name",
  transcript_version = "version", gene_version = "version",
  version_sep = ".", verbose = TRUE)
```

Arguments

`file` Path to a GTF file to be read. The file can remain gzipped.

`type_use` Character vector, the values taken by the `type` field in the GTF file that denote the desired transcripts. This can be "exon", "transcript", "mRNA", and etc.

`transcript_id` Character vector of length 1. Tag in `attribute` field corresponding to transcript IDs. This argument must be supplied and cannot be `NA` or `NULL`. Will throw error if tag indicated in this argument does not exist.

`gene_id` Character vector of length 1. Tag in `attribute` field corresponding to gene IDs. This argument must be supplied and cannot be `NA` or `NULL`. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

`gene_name` Character vector of length 1. Tag in `attribute` field corresponding to gene symbols. This argument can be `NA` or `NULL` if you are fine with non-human readable gene IDs and do not wish to extract human readable gene symbols.

`transcript_version` Character vector of length 1. Tag in `attribute` field corresponding to *transcript* version number. If your GTF file does not include transcript version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. To decide whether to include transcript version number, check whether version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`gene_version` Character vector of length 1. Tag in `attribute` field corresponding to *gene* version number. If your GTF file does not include gene version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. Unlike transcript version number, it's up to you whether to include gene version number.

`version_sep` Character to separate bewteen the main ID and the version number. Defaults to ".", as in Ensembl.

`verbose` Whether to display progress.

Details

Transcript and gene versions may not be present in all GTF files, so these arguments are optional. This function has arguments for transcript and gene version numbers because Ensembl IDs have version numbers. For Ensembl IDs, we recommend including the version number, since a change in version number signals a change in the entity referred to by the ID after reannotation. If a version is used, then it will be appended to the ID, separated by `version_sep`.

The transcript and gene IDs are The `attribute` field (the last field) of GTF files can be complicated and inconsistent across different sources. Please check the `attribute` tags in your GTF file and consider the arguments of this function carefully. The defaults are set according to Ensembl GTF files; defaults may not work for files from other sources. Due to the general lack of standards for the `attribute` field, you may need to further clean up the output of this function.

Value

A data frame at least 2 columns: `gene` for gene ID, `transcript` for transcript ID, and optionally, `gene_name` for gene names.

See Also

Other functions to retrieve transcript and gene info: sort_tr2g, tr2g_EnsDb, tr2g_TxDb, tr2g_ensembl, tr2g_fasta, tr2g_gtf, transcript2gene

Examples

```
toy_path <- system.file("testdata", package = "BUSpaRse")
file_use <- paste(toy_path, "gff3_test.gff3", sep = "/")
# Default
tr2g <- tr2g_gff3(file = file_use, verbose = FALSE)
# Excluding version numbers
tr2g <- tr2g_gff3(file = file_use, transcript_version = NULL,
  gene_version = NULL)
```

---

tr2g_GRanges *Get transcript and gene info from GRanges*

Description

Internal use, for GRanges from GTF files

Usage

```
tr2g_GRanges(gr, type_use = "exon", transcript_id = "transcript_id",
  gene_id = "gene_id", gene_name = "gene_name",
  transcript_version = "transcript_version",
  gene_version = "gene_version", version_sep = ".")
```

Arguments

`gr` A GRanges object. The metadata columns should be atomic vectors, not lists.

`type_use` Character vector, the values taken by the type field in the GTF file that denote the desired transcripts. This can be "exon", "transcript", "mRNA", and etc.

`transcript_id` Character vector of length 1. Tag in attribute field corresponding to transcript IDs. This argument must be supplied and cannot be NA or NULL. Will throw error if tag indicated in this argument does not exist.

`gene_id` Character vector of length 1. Tag in attribute field corresponding to gene IDs. This argument must be supplied and cannot be NA or NULL. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

`gene_name` Character vector of length 1. Tag in attribute field corresponding to gene symbols. This argument can be NA or NULL if you are fine with non-human readable gene IDs and do not wish to extract human readable gene symbols.

`transcript_version`
Character vector of length 1. Tag in attribute field corresponding to *transcript* version number. If your GTF file does not include transcript version numbers, or if you do not wish to include the version number, then use NULL for this argument. To decide whether to include transcript version number, check whether

version numbers are included in the `transcripts.txt` in the `kallisto` output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.

`gene_version` Character vector of length 1. Tag in `attribute` field corresponding to *gene* version number. If your GTF file does not include gene version numbers, or if you do not wish to include the version number, then use `NULL` for this argument. Unlike transcript version number, it's up to you whether to include gene version number.

`version_sep` Character to separate bewteen the main ID and the version number. Defaults to ".", as in Ensembl.

Value

A data frame at least 2 columns: `gene` for gene ID, `transcript` for transcript ID, and optionally, `gene_name` for gene names.

`tr2g_gtf` *Get transcript and gene info from GTF file*

Description

This function reads a GTF file and extracts the transcript ID and corresponding gene ID. This function assumes that the GTF file is properly formatted. See `http://mblab.wustl.edu/GTF2.html` for a detailed description of proper GTF format. Note that GFF3 files have a somewhat different and more complicated format in the attribute field, which this function does not support. See `http://gmod.org/wiki/GFF3` for a detailed description of proper GFF3 format. To extract transcript and gene information from GFF3 files, see the function `tr2g_gff3` in this package.

Usage

```
tr2g_gtf(file, type_use = "exon", transcript_id = "transcript_id",
  gene_id = "gene_id", gene_name = "gene_name",
  transcript_version = "transcript_version",
  gene_version = "gene_version", version_sep = ".", verbose = TRUE)
```

Arguments

`file` Path to a GTF file to be read. The file can remain gzipped.

`type_use` Character vector, the values taken by the `type` field in the GTF file that denote the desired transcripts. This can be "exon", "transcript", "mRNA", and etc.

`transcript_id` Character vector of length 1. Tag in `attribute` field corresponding to transcript IDs. This argument must be supplied and cannot be `NA` or `NULL`. Will throw error if tag indicated in this argument does not exist.

`gene_id` Character vector of length 1. Tag in `attribute` field corresponding to gene IDs. This argument must be supplied and cannot be `NA` or `NULL`. Note that this is different from gene symbols, which do not have to be unique. This can be Ensembl or Entrez IDs. However, if the gene symbols are in fact unique for each gene, you may supply the tag for human readable gene symbols to this argument. Will throw error if tag indicated in this argument does not exist.

gene_name	Character vector of length 1. Tag in attribute field corresponding to gene symbols. This argument can be NA or NULL if you are fine with non-human readable gene IDs and do not wish to extract human readable gene symbols.
transcript_version	Character vector of length 1. Tag in attribute field corresponding to *transcript* version number. If your GTF file does not include transcript version numbers, or if you do not wish to include the version number, then use NULL for this argument. To decide whether to include transcript version number, check whether version numbers are included in the transcripts.txt in the kallisto output directory. If that file includes version numbers, then trannscript version numbers must be included here as well. If that file does not include version numbers, then transcript version numbers must not be included here.
gene_version	Character vector of length 1. Tag in attribute field corresponding to *gene* version number. If your GTF file does not include gene version numbers, or if you do not wish to include the version number, then use NULL for this argument. Unlike transcript version number, it's up to you whether to include gene version number.
version_sep	Character to separate bewteen the main ID and the version number. Defaults to ".", as in Ensembl.
verbose	Whether to display progress.

Details

Transcript and gene versions may not be present in all GTF files, so these arguments are optional. This function has arguments for transcript and gene version numbers because Ensembl IDs have version numbers. For Ensembl IDs, we recommend including the version number, since a change in version number signals a change in the entity referred to by the ID after reannotation. If a version is used, then it will be appended to the ID, separated by version_sep.

The transcript and gene IDs are The attribute field (the last field) of GTF files can be complicated and inconsistent across different sources. Please check the attribute tags in your GTF file and consider the arguments of this function carefully. The defaults are set according to Ensembl GTF files; defaults may not work for files from other sources. Due to the general lack of standards for the attribute field, you may need to further clean up the output of this function.

Value

A data frame at least 2 columns: gene for gene ID, transcript for transcript ID, and optionally, gene_name for gene names.

See Also

Other functions to retrieve transcript and gene info: sort_tr2g, tr2g_EnsDb, tr2g_TxDb, tr2g_ensembl, tr2g_fasta, tr2g_gff3, transcript2gene

Examples

```
toy_path <- system.file("testdata", package = "BUSpaRse")
file_use <- paste(toy_path, "gtf_test.gtf", sep = "/")
# Default
tr2g <- tr2g_gtf(file = file_use, verbose = FALSE)
# Excluding version numbers
tr2g <- tr2g_gtf(file = file_use, transcript_version = NULL,
  gene_version = NULL)
```

`tr2g_junction` *tr2g for exon-exon junctions*

Description

tr2g for exon-exon junctions

Usage

```
tr2g_junction(tr2g_cdna, junction_names)
```

Arguments

`tr2g_cdna`	The original `tr2g_cdna`.
`junction_names`	Names of junctions internally generated.

Value

A `tr2g` data frame where "transcripts" are the exon-exon junctions and genes are the corresponding genes.

`tr2g_TxDb` *Get transcript and gene info from TxDb objects*

Description

The genome and gene annotations of some species can be conveniently obtained from Bioconductor packages. This is more convenient than downloading GTF files from Ensembl and reading it into R. In these packages, the gene annotation is stored in a `TxDb` object, which has standardized names for gene IDs, transcript IDs, exon IDs, and so on, which are stored in the metadata fields in GTF and GFF3 files, which are not standardized. This function extracts transcript and corresponding gene information from gene annotation stored in a `TxDb` object.

Usage

```
tr2g_TxDb(txdb)
```

Arguments

`txdb`	A `TxDb` object with gene annotation.

Value

A data frame with 3 columns: `gene` for gene ID, `transcript` for transcript ID, and `tx_id` for internal transcript IDs used to avoid duplicate transcript names. For TxDb packages from Bioconductor, gene ID is Entrez ID, while transcript IDs are Ensembl IDs with version numbers for `TxDb.Hsapiens.UCSC.hg38.knownGene`. In some cases, the transcript ID have duplicates, and this is resolved by adding numbers to make the IDs unique.

A data frame with 3 columns: `gene` for gene ID, `transcript` for transcript ID, and `gene_name` for gene names. If `other_attrs` has been specified, then those will also be columns in the data frame returned.

See Also

Other functions to retrieve transcript and gene info: `sort_tr2g`, `tr2g_EnsDb`, `tr2g_ensembl`, `tr2g_fasta`, `tr2g_gff3`, `tr2g_gtf`, `transcript2gene`

Other functions to retrieve transcript and gene info: `sort_tr2g`, `tr2g_EnsDb`, `tr2g_ensembl`, `tr2g_fasta`, `tr2g_gff3`, `tr2g_gtf`, `transcript2gene`

Examples

```
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
tr2g_TxDb(TxDb.Hsapiens.UCSC.hg38.knownGene)
```

`transcript2gene` *Map Ensembl transcript ID to gene ID*

Description

This function is a shortcut to get the correctly sorted data frame with transcript IDs and the corresponding gene IDs from Ensembl biomart or Ensembl transcriptome FASTA files. For biomart query, it calls `tr2g_ensembl` and then `sort_tr2g`. For FASTA files, it calls `tr2g_fasta` and then `sort_tr2g`. Unlike in `tr2g_ensembl` and `tr2g_fasta`, multiple species can be supplied if cells from different species were sequenced together. This function should only be used if the kallisto inidex was built with transcriptomes from Ensembl. Also, if querying biomart, please make sure to set `ensembl_version` to match the version where the transcriptomes were downloaded.

Usage

```
transcript2gene(species, fasta_file, kallisto_out_path,
  type = "vertebrate", verbose = TRUE, ...)
```

Arguments

`species` A character vector of Latin names of species present in this scRNA-seq dataset. This is used to retrieve Ensembl information from biomart.

`fasta_file` Character vector of paths to the transcriptome FASTA files used to build the kallisto index. Exactly one of `species` and `fasta_file` can be missing.

`kallisto_out_path` Path to the `kallisto` bus output directory.

`type` A character vector indicating the type of each species. Each element must be one of "vertebrate", "metazoa", "plant", "fungus", and "protist". If length is 1, then this type will be used for all species specified here. Can be missing if `fasta_file` is specified.

`verbose` Whether to display progress. Defaults to `TRUE`.

`...` Other arguments passed to `tr2g_ensembl` such as `other_attrs`, `ensembl_version`, and arguments passed to `useEnsembl`. If `fasta_files` is supplied instead of `species`, then this will be extra argumennts to `tr2g_fasta`, such as `use_transcript_version` and `use_gene_version`.

Value

A data frame with two columns: `gene` and `transcript`, with Ensembl gene and transcript IDs (with version number), in the same order as in the transcriptome index used in `kallisto`.

See Also

Other functions to retrieve transcript and gene info: `sort_tr2g`, `tr2g_EnsDb`, `tr2g_TxDb`, `tr2g_ensembl`, `tr2g_fasta`, `tr2g_gff3`, `tr2g_gtf`

Examples

```
# Download dataset already in BUS format
library(TENxBUSData)
TENxBUSData(".", dataset = "retina")
tr2g <- transcript2gene("Mus musculus", type = "vertebrate",
  ensembl_version = 94, kallisto_out_path = "./out_retina")
```

`validate_velocity_input` *Validate input to get_velocity_files*

Description

Validate input to get_velocity_files

Usage

```
validate_velocity_input(L, Genome, Transcriptome, out_path, compress_fa,
  width, exon_option)
```

Arguments

`L` Length of the biological read. For instance, 10xv1: 98 nt, 10xv2: 98 nt, 10xv3: 91 nt, Drop-seq: 50 nt. If in doubt check read length in a fastq file for biological reads with the bash commands: If the fastq file is gzipped, then do `zcat your_file.fastq.gz | head` on Linux. If on Mac, then `zcat < your_file.fastq.gz | head`. Then you will see lines with nucleotide bases. Copy one of those lines and determine its length with `str_length` in R or `echo -n <the sequence> | wc -c` in bash. Which file corresponds to biological reads depends on the particular technology.

`Genome` Either a `BSgenome` or a `XStringSet` object of genomic sequences, where the intronic sequences will be extracted from. Use `genomeStyles` to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.

`Transcriptome` A `XStringSet`, a path to a fasta file (can be gzipped) of the transcriptome which contains sequences of spliced transcripts, or `NULL`. The transcriptome here will be concatenated with the intronic sequences to give one fasta file. When `NULL`, the transriptome sequences will be extracted from the genome given the gene annotation, so it will be guaranteed that transcript IDs in the transcriptome and in the annotation match. Otherwise, the type of transcript ID in the transcriptome must match that in the gene annotation supplied via argument `X`.

`out_path` Directory to save the outputs written to disk. If this directory does not exist, then it will be created. Defaults to the current working directory.

`compress_fa` Logical, whether to compress the output fasta file of transcriptome and flanked intronic sequenncess. If `TRUE`, then the fasta file will be gzipped.

`width` Maximum number of letters per line of sequence in the output fasta file. Must be an integer.

`exon_option` Character, indicating how exonic sequences should be included in the kallisto index. Must be one of the following:

- **full** The full cDNA sequences, which include the full exonic sequences, will be used. This is the default.
- **junction** Only the exon-exon junctions, with L-1 bases on each side of the junctions, will be used.

Value

Will throw error if validation fails. Returns a named list whose first element is the normalized path to output directory, and whose second element is the normalized path to the transcriptome file if specified.

`write_velocity_output` *Write the files for RNA velocity to disk*

Description

Write the files for RNA velocity to disk, in the specified output directory.

Usage

```
write_velocity_output(out_path, introns, Genome, Transcriptome,
  isoform_action, exon_option, tr2g_cdna, compress_fa, width)
```

Arguments

`out_path` Directory to save the outputs written to disk. If this directory does not exist, then it will be created.

`introns` Intronic ranges plus flanking region, returned by `get_intron_flanks`.

`Genome` Either a `BSgenome` or a `XStringSet` object of genomic sequences, where the intronic sequences will be extracted from. Use `genomeStyles` to check which styles are supported for your organism of interest; supported styles can be interconverted. If the style in your genome or annotation is not supported, then the style of chromosome names in the genome and annotation should be manually set to be consistent.

`Transcriptome` A `XStringSet`, a path to a fasta file (can be gzipped) of the transcriptome which contains sequences of spliced transcripts, or `NULL`. The transcriptome here will be concatenated with the intronic sequences to give one fasta file. When `NULL`, the transriptome sequences will be extracted from the genome given the gene annotation, so it will be guaranteed that transcript IDs in the transcriptome and in the annotation match. Otherwise, the type of transcript ID in the transcriptome must match that in the gene annotation supplied via argument `X`.

`isoform_action` Character, indicating action to take with different transcripts of the same gene. Must be one of the following:

- **collapse** First, the union of all exons of different transcripts of a gene will be taken. Then the introns will be inferred from this union. Only the flanked intronic sequences are affected; isoforms will always be taken into account for spliced sequences or exon-exon junctions.
- **separate** Introns from different transcripts will be kept separate.

`exon_option` Character, indicating how exonic sequences should be included in the kallisto index. Must be one of the following:

- **full** The full cDNA sequences, which include the full exonic sequences, will be used. This is the default.
- **junction** Only the exon-exon junctions, with L-1 bases on each side of the junctions, will be used.

`tr2g_cdna` A data frame with columns `transcript` and `gene` that maps transcripts to genes for spliced transcripts.

`compress_fa` Logical, whether to compress the output fasta file of transcriptome and flanked intronic sequenncess. If `TRUE`, then the fasta file will be gzipped.

`width` Maximum number of letters per line of sequence in the output fasta file. Must be an integer.

Value

Nothing into the R session. The files are written to disk.

Index