Package: tximeta (via r-universe)

May 30, 2026

Version 1.30.0

Title Transcript Quantification Import with Automatic Metadata

Description Transcript quantification import from Salmon and other quantifiers with automatic attachment of transcript ranges and release information, and other associated metadata. De novo transcriptomes can be linked to the appropriate sources with linkedTxomes and shared for computational reproducibility.

Maintainer Michael Love <michaelisaiahlove@gmail.com>

License GPL-2

VignetteBuilder knitr, rmarkdown

Depends R (>= 4.1.0)

Imports SummarizedExperiment (>= 1.39.1), tximport, jsonlite, S4Vectors, IRanges, GenomicRanges (>= 1.61.1), AnnotationDbi, DBI, GenomicFeatures, txdbmaker, ensembldb, BiocFileCache, AnnotationHub, Biostrings, tibble, Seqinfo, tools, utils, methods, Matrix

Suggests knitr, rmarkdown, testthat, tximportData (>= 1.37.5), org.Dm.eg.db, DESeq2, edgeR (>= 4.9.2), limma, devtools, macrophage

URL https://github.com/thelovelab/tximeta

biocViews Annotation, GenomeAnnotation, DataImport, Preprocessing, RNASeq, LongRead, SingleCell, Transcriptomics, Transcription, GeneExpression, FunctionalGenomics, ReproducibleResearch, ReportWriting, ImmunoOncology

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Encoding UTF-8

Config/Needs/website r-lib/pkgdown

Config/pak/sysreqs
make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository https://bioc-release.r-universe.dev

Date/Publication 2026-04-28 12:48:36 UTC

RemoteUrl https://github.com/bioc/tximeta

RemoteRef RELEASE_3_23

RemoteSha fe803f92700131ac603f474754f56945224a5c31

Contents

`tximeta-package` *Import transcript quantification with metadata*

Description

The tximeta package imports abundances (TPM), estimated counts, and effective lengths from quantification tools, and will output a *SummarizedExperiment* (SE) object. For salmon and related quantification tools, `tximeta()` will attempt to identify the correct provenance of the reference transcripts and automatically attach the transcript ranges to the SummarizedExperiment, to facilitate downstream integration with other datasets. The automatic identification of reference transcripts should work out-of-the-box for human or mouse transcriptomes from the sources: GENCODE, Ensembl, or RefSeq. See also `importData()` for importing data when the reference transcripts were derived from a mix of annotated (e.g. GENCODE) and novel or custom transcripts.

Details

The main functions are:

- `tximeta()` - with key argument `coldata` specifying sample information
- `summarizeToGene()` - summarize quantification to gene-level

- `importData()` - import quantification with mixed reference transcript sets

All software-related questions should be posted to the Bioconductor Support Site:

https://support.bioconductor.org

The code can be viewed at the GitHub repository, which also lists the contributor code of conduct:

https://github.com/thelovelab/tximeta

Author(s)

Michael I. Love, Charlotte Soneson, Peter Hickey, Rob Patro

References

- *tximeta* reference:

Michael I. Love, Charlotte Soneson, Peter F. Hickey, Lisa K. Johnson N. Tessa Pierce, Lori Shepherd, Martin Morgan, Rob Patro (2020) *Tximeta: reference sequence checksums for provenance identification in RNA-seq*. PLOS Computational Biology. https://doi.org/10.1371/journal.pcbi.1007664

- *tximport* reference (the effective length GLM offset and counts-from-abundance):

Charlotte Soneson, Michael I. Love, Mark D. Robinson (2015) *Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences*. F1000Research. http://doi.org/10.12688/f1000research.7563

See Also

Useful links:

- https://github.com/thelovelab/tximeta

`addCDS` *Add CDS to rowRanges of a transcript-level SummarizedExperiment*

Description

Working similarly to `addExons`, this function can be used to add information about CDS (coding sequence) to the `SummarizedExperiment` object. As not all transcripts are coding, we have CDS information for only a subset of the rows of the object. For this reason, a logical indicator for whether the transcript is coding, `mcols(se)$coding`, is added as a column to the metadata columns of the `rowRanges` of the object. An additional column, `mcols(se)$cds`, is added to the metadata columns, which is a `GRangesList` with either the CDS regions (if the transcript is coding), or the original transcript/exon ranges (if the transcript is non-coding). This is necessary, as `GRangesList` cannot have NA elements. As with `addExons`, this function is designed only for transcript-level objects.

Usage

```
addCDS(se)
```

Arguments

se	the SummarizedExperiment

Value

a SummarizedExperiment

---

addExons *Add exons to rowRanges of a transcript-level SummarizedExperiment*

Description

After running tximeta, the SummarizedExperiment output will have GRanges representing the transcript locations attached as rowRanges to the object. These provide the start and end of the transcript in the genomic coordiantes, and strand information. However, the exonic locations are not provided. This function, addExons, swaps out the GRanges with a GRangesList, essentially a list along the rows of the SummarizedExperiment, where each element of the list is a GRanges providing the locations of the exons for that transcript.

Usage

```
addExons(se)
```

Arguments

se	the SummarizedExperiment

Details

This function is designed only for transcript-level objects. This "lack of a feature" reflects a belief on the part of the package author that it makes more sense to think about exons belonging to transcripts than to genes. For users desiring exonic information alongside gene-level objects, for example, which exons are associated with a particular gene, it is recommended to pull out the relevant GRangesList for the transcripts of this gene, while the object represents transcript-level data, such that the exons are still associated with transcripts.

For an example of addExons, please see the tximeta vignette.

Value

a SummarizedExperiment

addIds *Add IDs to rowRanges of a SummarizedExperiment*

Description

For now this function just works with SummarizedExperiments with Ensembl gene or transcript IDs. See example of usage in tximeta vignette. For obtaining multiple matching IDs for each row of the SummarizedExperiment set `multiVals="list"`. See `select` for documentation on use of `multiVals`.

Usage

```
addIds(se, column, fromDb = FALSE, gene = FALSE, ...)
```

Arguments

`se`	the SummarizedExperiment
`column`	the name of the new ID to add (a `column` of the org package database or of the TxDb/EnsDb is `fromDb=TRUE`)
`fromDb`	logical, whether to use the TxDb/EnsDb that is associated with `se`. Default is FALSE, and an org package is used. Currently only implemented for transcript level (gene=FALSE). Column names can be viewed with `columns(retrieveDb(se))`
`gene`	logical, whether to map by genes or transcripts (default is FALSE). if rows are genes, and easily detected as such (ENSG or ENSMUSG), it will automatically switch to TRUE. if rows are transcripts and `gene=TRUE`, then it will try to use a `gene_id` column to map IDs to `column`
`...`	arguments passed to `mapIds`

Value

a SummarizedExperiment

Examples

```
example(tximeta)
library(org.Dm.eg.db)
se <- addIds(se, "REFSEQ", gene=FALSE)
```

getTximetaBFC *Get or set the directory of the BiocFileCache used by tximeta*

Description

Running `getTximetaBFC` will report the saved directory, if it has been determined, or will return NULL. Running `setTximetaBFC` will ask the user to specify a BiocFileCache directory for accessing and saving TxDb sqlite files. Note that tximeta's BiocFileCache can be set by the environmental variable `TXIMETA_HUB_CACHE`, which will reset the cache location.

Usage

```
getTximetaBFC()

setTximetaBFC(dir, quiet = FALSE)
```

Arguments

`dir`	the location for tximeta's BiocFileCache. can be missing in which case the function will call `file.choose` for choosing location interactively
`quiet`	whether to suppress feedback message

Value

the directory of the BiocFileCache used by tximeta (or nothing, in the case of `setTximetaBFC`)

Examples

```
# getting the BiocFileCache used by tximeta
# (may not be set, which uses BiocFileCache default or temp directory)
getTximetaBFC()

# don't want to actually change user settings so this is not run:
# setTximetaBFC()
```

importData *Import quantification across mixed reference transcripts*

Description

The *oarfish* quantification tools allows a mix of `--annotated` reference transcripts (e.g. GENCODE, Ensembl) and `--novel` or custom transcripts (e.g. de novo assembled transcripts not present in the annotated set) to be used as the index for quantification. `importData()` and associated functions facilitate import, reference identification, and addition of metadata across `annotated` and/or `novel` transcripts. The `importData()` function alone imports the data, while inspection of the recognized digests and updating of transcript metadata is handled by subsequent functions (listed in *See also* section below).

Usage

```
importData(coldata, type = "oarfish", quiet = FALSE, ...)
```

Arguments

coldata	data.frame with columns files and names as in tximeta()
type	what quantifier was used (see tximport::tximport()), for now importData() works for "oarfish" files
quiet	whether to suppress printed messages
...	arguments passed to tximport::tximport()

Details

oarfish with mixed reference transcript sets may have been generated with e.g.

```
oarfish --only-index --annotated gencode.v48.transcripts.fa.gz \
  --novel my_novel_txps.fa.gz --seq-tech ont-cdna --threads 32 \
  --index-out gencode_plus_novel
oarfish --reads reads/experiment_rep1.fastq.gz --index gencode_plus_novel \
  --output quants/experiment_rep1 --seq-tech ont-cdna \
  --filter-group no-filters --threads 32
```

Value

an un-ranged SummarizedExperiment (SE) object, for use with subsequent functions described in *See also* section

See Also

inspectDigests() and updateMetadata() for subsequent tasks of inspecting digest matches and updating metadata, respectively. makeLinkedTxome() can be used to add custom metadata into the registry used for inspecting digests and then updating transcript data. A user may follow the workflow importData() > inspectDigests() > makeLinkedTxome() > inspectDigests() > updateMetadata(). See also makeLinkedTxpData() for a lightweight alternative of linking *GRanges* metadata to a digest.

Examples

```
# oarfish files using a mix of --annotated and --novel transcripts
dir <- system.file("extdata/oarfish", package="tximportData")
names <- paste0("rep", 2:4)
files <- file.path(dir, paste0("sgnex_h9_", names, ".quant.gz"))
coldata <- data.frame(files, names)

# returns an un-ranged SE object
se <- importData(coldata, type="oarfish")
```

`inspectDigests` *Inspect digest matches from* `importData()` *imported data*

Description

This function takes as input a *SummarizedExperiment* as output by `importData()` and returns a tibble with information about the digest-match status of two indices (`annotated` and `novel`), with respect to *tximeta* metadata. Inspection of index digests can be run iteratively, checking if the digests used in the mixed reference transcript set have a match against 1) pre-computed digests representing standard annotated sets (e.g. GENCODE, Ensembl, etc.) or 2) digests added by the user to a local registry with `makeLinkedTxome()` (GTF file) or `makeLinkedTxpData` (*GRanges*-based metadata). Optional columns may be added if specified by `fullDigest=TRUE` (include the full digest) and/or `count=TRUE` (add matching transcript ID counts per index). Following inspection, one can run `updateMetadata()` to automatically update the transcript metadata using the sources indicated by this function.

Usage

```
inspectDigests(
  se,
  type = "oarfish",
  prefer = c("txome", "txpdata", "precomputed"),
  fullDigest = FALSE,
  count = FALSE
)
```

Arguments

`se`	the *SummarizedExperiment* output by `importData()`, or alternatively just `metadata(se)$quantInfo`, a list of metadata information from the quantification tool (assuming `annotated` and `novel` indices both used)
`type`	what quantifier was used (see `tximport::tximport()`)
`prefer`	vector of length up to 3, giving the preferred order of *tximeta*'s transcript registries to when finding matches, with elements: `txome`: linkedTxome, `txpdata`: linkedTxpData, `precomputed`: the pre-computed digests in tximeta
`fullDigest`	whether to include the full digest string in the output, in addition to the shortened 6-char version
`count`	whether to count the number of matching transcripts ID to each index (only possible for those indices that have matching metadata). Counting requires loading transcript data, either from locally cached databases or from GTF files.

Value

a 2-row tibble of the `annotated` and `novel` index, their matching information if available (source, organism, release), for matches, whether it is a `linkedTxome` or a `linkedTxpData` (both 'FALSE" for pre-computed) and a small 6 character version of the digest itself.

Examples

```
example(importData)
# now we have an `se` created by importData()...
inspectDigests(se)
# can then update the registry via makeLinkedTxome() and re-run inspection
```

---

linkedTxome *Make and load linked transcriptomes (linked GTF and FASTA)*

Description

`makeLinkedTxome()` reads the digest associated with a salmon index at `indexDir`, and persistently links it to metadata (alternatively the `digest` string itself and an `indexName` can be provided). Linked metadata includes key information about the transcriptome, including the `source`, `organism`, `release`, and `genome` (these are custom character strings), as well as the locations (e.g. local, HTTP, or FTP) for one or more `fasta` files and one `gtf` file. `loadLinkedTxome()` loads this information from a JSON file. See *Details*.

Usage

```
makeLinkedTxome(
  digest = NULL,
  indexName,
  indexDir = NULL,
  source,
  organism,
  release,
  genome,
  fasta,
  gtf,
  write = TRUE,
  jsonFile
)

loadLinkedTxome(jsonFile)
```

Arguments

`digest`	the full digest as character string, (this or `indexDir` is required, only one should be specified)
`indexName`	a name for the `index` when storing the linkedTxome, required if providing the `digest` string, suggest using the basename of the FASTA file and the software used, e.g. "gencode.vXX_salmon-0.XX.Y"
`indexDir`	the local path to the salmon index (this or `digest` is required, only one should be specified)

source	the source of transcriptome (e.g. "de-novo"). Note: if you specify "GENCODE" or "Ensembl", this will trigger behavior by tximeta that may not be desired: e.g. attempts to download canonical transcriptome data from AnnotationHub (unless useHub=FALSE when running tximeta) and parsing of Ensembl GTF using ensembldb (which may fail if the GTF file has been modified). For transcriptomes that are defined by local GTF files, it is recommended to use the terms "LocalGENCODE" or "LocalEnsembl". Setting "LocalEnsembl" will also strip version numbers from the FASTA transcript IDs to enable matching with the Ensembl GTF.
organism	organism (e.g. "Homo sapiens")
release	release number (e.g. "27")
genome	genome (e.g. "GRCh38", or "none")
fasta	location(s) for the FASTA transcript sequences (of which the transcripts used to build the index is equal or a subset). This can be a local path, or an HTTP or FTP URL
gtf	location for the GTF/GFF file (of which the transcripts used to build the index is equal or a subset). This can be a local path, or an HTTP or FTP URL While the `fasta` argument can take a vector of length greater than one (more than one FASTA file containing transcripts used in indexing), the `gtf` argument has to be a single GTF/GFF file. This can also be a serialized GRanges object (location of a .rds file) imported with rtracklayer. If transcripts were added to a standard set of reference transcripts (e.g. fusion genes, or pathogen transcripts), it is recommended that the tximeta user would manually add these to the GTF/GFF file, and post the modified GTF/GFF publicly, such as on Zenodo. This enables consistent annotation and downstream annotation tasks, such as by `summarizeToGene()`.
write	logical, should a JSON file be written out which documents the transcriptome digest and metadata? (default is TRUE)
jsonFile	the path to the json file for the linkedTxome

Details

`makeLinkedTxome()` links the information about the transcriptome used for quantification in two ways:

1. the function will store a record in tximeta's cache such that future import of quantification data will automatically access and parse the GTF as if the transcriptome were one of those automatically detected by tximeta. Then all features of tximeta (e.g. summarization to gene, programmatic adding of IDs or metadata) will be available;
2. it will by default write out a JSON file that can be shared, or posted online, and which can be read by `loadLinkedTxome()` which will store the information in tximeta's cache. This should make the full quantification-import pipeline computationally reproducible / auditable even for transcriptomes which differ from those provided by references (GENCODE, Ensembl, RefSeq).

For further details please see the "Linked transcriptomes" section of the tximeta vignette.

This function can be used in combination with `inspectDigests()` and oarfish data from `importData()`, when multiple reference transcript sets have been indexed. See also `makeLinkedTxpData()`.

Value

nothing, the function is run for its side effects

Examples

```
# point to a salmon quantification file with an additional artificial transcript
dir <- system.file("extdata/salmon_dm", package="tximportData")
file <- file.path(dir, "SRR1197474.plus", "quant.sf")
coldata <- data.frame(files=file, names="SRR1197474", sample="1",
                      stringsAsFactors=FALSE)

# now point to the salmon index itself to create a linkedTxome
# as the index will not match a known txome
indexDir <- file.path(dir, "Dm.BDGP6.22.98.plus_salmon-0.14.1")

# point to the source FASTA and GTF:
baseFTP <- "ftp://ftp.ensembl.org/pub/release-98/fasta/drosophila_melanogaster/"
fastaFTP <- c(
  paste0(baseFTP,
    c("cdna/Drosophila_melanogaster.BDGP6.22.cdna.all.fa.gz",
      "ncrna/Drosophila_melanogaster.BDGP6.22.ncrna.fa.gz")),
  "extra_transcript.fa.gz"
)
gtfPath <- file.path(dir, "Drosophila_melanogaster.BDGP6.22.98.plus.gtf.gz")

# now create a linkedTxome, linking the salmon index to its FASTA and GTF sources
makeLinkedTxome(indexDir=indexDir, source="LocalEnsembl", organism="Drosophila melanogaster",
              release="98", genome="BDGP6.22", fasta=fastaFTP, gtf=gtfPath, write=FALSE)

# to clear the entire linkedTxome table
# (don't run unless you want to clear this table!)
# bfcloc <- getTximetaBFC()
# bfc <- BiocFileCache(bfcloc)
# bfcremove(bfc, bfcquery(bfc, "linkedTxomeTbl")$rid)
```

---

linkedTxpData *Make linked transcript data (linked GRanges)*

Description

`linkedTxpData` allows the user to save relevant *GRanges* transcript data for identifying and updating transcript metadata in a persistent manner across R sessions. It can be used in combination with `inspectDigests()` and `updateMetadata()`. This is a lightweight version of `linkedTxome` (see `makeLinkedTxome()`), which requires specifying a GTF file for building a *TxDb* and optionally a FASTA file for sequence retrieval.)

Usage

```
makeLinkedTxpData(
  digest,
  digestType = "sha256",
  indexName,
  txpData,
  source,
  organism,
  release,
  genome
)
```

Arguments

digest	character string of the full digest of the reference transcripts, see `inspectDigests()` with `fullDigest=TRUE`
digestType	character string of the digest, default `"sha256"`
indexName	a name for the `index` when storing the linkedTxpData,
txpData	*GRanges* providing information about ranges representing the transcript sequences linked to `digest`
source	the source of transcriptome, e.g. `denovo`. See `makeLinkedTxome()` for more information on specifying source
organism	organism (e.g. "Homo sapiens")
release	release number (e.g. "27")
genome	genome (e.g. "GRCh38", or "none")

Details

The `txpData` object is saved in the `getTximetaBFC()` location, appended with a 32-character substring of `digest`. The tibble listing all `linkedTxpData` is named `linkedTxpDataTbl` and is listed in the same location.

Value

nothing, the function is run for its side effects

Examples

```
novel <- data.frame(seqnames = paste0("chr", rep(1:22, each=500)),
  start = 1e6 + 1 + 0:499 * 1000, end = 1e6 + 1 + 0:499 * 1000 + 1000 - 1,
  strand = "+", tx_name = paste0("novel", 1:(22*500)),
  gene_id = paste0("novel_gene", rep(1:(22*10), each=50)),
type = "protein_coding")
novel_gr <- as(novel, "GRanges")
names(novel_gr) <- novel$tx_name

makeLinkedTxpData(
```

```
 digest = "43158f2c8e88e3acd77c22aee557625a6f1b6a5038cfc7deb5e64903892d8070",
 digestType = "sha256",
 indexName = "my_novel_txps",
 txpData = novel_gr,
 source = "novel", organism="Homo sapiens",
 release="v1", genome="GRCh38"
)

# to clear the entire linkedTxome table
# (don't run unless you want to clear this table!)
# bfcloc <- getTximetaBFC()
# bfc <- BiocFileCache(bfcloc)
# bfcremove(bfc, bfcquery(bfc, "linkedTxpDataTbl")$rid)
```

`makeDGEList` *Make a DGEList from tximeta output*

Description

A simple wrapper function for constructing a DGEList for use with edgeR. See vignette for an example. Requires installation of the edgeR package from Bioconductor.

Usage

```
makeDGEList(se)
```

Arguments

`se`	a SummarizedExperiment produced by tximeta

Value

a DGEList

`retrieveCDNA` *Retrieve the cDNA transcript sequence for a SummarizedExperiment*

Description

This helper function retrieves the cDNA sequence of the transcripts used for expression quantification. This function either downloads or loads the transcript sequence from cache, it does not re-order or check against the rows of the SummarizedExperiment (which could be already summarized to genes for example).

Usage

```
retrieveCDNA(se, quiet = FALSE)
```

Arguments

se	the SummarizedExperiment
quiet	logical, suppress messages

Value

a DNAStringSet object

Examples

```
## Not run:
# this example is not run because it requires access to Ensembl ftp
example(tximeta)
cdna <- retrieveCDNA(se)

## End(Not run)
```

---

retrieveDb *Retrieve the TxDb or EnsDb associated with a SummarizedExperiment*

Description

SummarizedExperiment objects returned by `tximeta` have associated TxDb or EnsDb databases which are cached locally and used to perform various metadata related tasks. This helper function retrieves the database itself for the user to perform any additional operations.

Usage

```
retrieveDb(se)
```

Arguments

se	the SummarizedExperiment

Value

a database object

Examples

```
example(tximeta)
edb <- retrieveDb(se)
```

splitSE *Split SummarizedExperiment by gene categories*

Description

Construct a new SummarizedExperiment by splitting one of the assays into a list of assays, each of which contains features of a given 'type'. It is assumed that there is a one-to-one correspondence between feature sets of different types; for example, these can be spliced and unspliced variants of the same transcripts. The type of each feature in the original SummarizedExperiment, and the correspondence between the features of different types, are given in a `data.frame`.

Usage

```
splitSE(se, splitDf, assayName)
```

Arguments

`se`	A `SummarizedExperiment` object.
`splitDf`	A `data.frame` with feature IDs. Each column represents a separate feature type, and the features in a given row are considered representatives of the same feature (and will be represented as one feature in the output object).
`assayName`	A character scalar, indicating the assay of `se` that will be split. Must be one of `assayNames(se)`.

Value

A `SummarizedExperiment` object with the same columns as the input object, and the same number of assays as the number of columns in `splitDf`. The assays will be named by the column names of `splitDf`. The `colData` and `metadata` of the input `SummarizedExperiment` object are copied to the output object. The row names are set to the feature IDs in the first column of `splitDf`.

Examples

```
se <- SummarizedExperiment::SummarizedExperiment(
  assays = S4Vectors::SimpleList(
    counts = as(matrix(1:15, nrow = 5), "sparseMatrix"),
    logcounts = log2(matrix(1:15, nrow = 5))
  ),
  colData = S4Vectors::DataFrame(sID = paste0("S", 1:3),
                                 condition = c("A", "A", "B")),
  metadata = list(md1 = "annotation")
)
rownames(se) <- paste0("G", 1:5)
colnames(se) <- paste0("P", 1:3)
splitDf <- data.frame(spliced = c("G1", "G2", "G6"),
                      unspliced = c("G3", "G5", "G4"),
                      stringsAsFactors = FALSE)
```

```
splse <- splitSE(se = se, splitDf = splitDf, assayName = "counts")
```

---

summarizeToGene,SummarizedExperiment-method
Summarize estimated quantitites to gene-level

Description

Summarizes abundances, counts, lengths, (and inferential replicates or variance) from transcript- to gene-level. Transcript IDs are stored as a CharacterList in the `mcols` of the output object. This function operates on SummarizedExperiment objects, and will automatically access the relevant TxDb (by either finding it in the BiocFileCache or by building it from an ftp location). This function uses the tximport package to perform summarization, where a method is defined that works on simple lists.

Usage

```
## S4 method for signature 'SummarizedExperiment'
summarizeToGene(
  object,
  assignRanges = c("range", "abundant"),
  varReduce = FALSE,
  skipRanges = FALSE,
  ...
)
```

Arguments

`object`	a SummarizedExperiment produced by `tximeta`
`assignRanges`	`"range"` or `"abundant"`, this argument controls the way that the `rowRanges` of the output object are assigned (note that this argument does not affect data aggregation at all). The default is to just output the entire `range` of the gene, i.e. the leftmost basepair to the rightmost basepair across all isoforms. Alternatively, for expressed genes, one can obtain the start and end of the most `abundant` isoform (averaging over all samples). Non-expressed genes will have `range`-based positions. For `abundant`, for expressed genes, the name of the range-assigned `isoform`, `max_prop` (maximum isoform proportion), and `iso_prop` (numeric values for isoform proportions) are also returned in `mcols`
`varReduce`	whether to reduce per-sample inferential replicates information into a matrix of sample variances `variance` (default FALSE)
`skipRanges`	whether to skip making use of, or outputting, rowRanges (default FALSE)
`...`	arguments passed to `tximport`

Value

a SummarizedExperiment with summarized quantifications and transcript IDs as a CharacterList in the `mcols`

Examples

```
example(tximeta)
gse <- summarizeToGene(se)
```

`tximeta` *Import transcript quantification with metadata*

Description

`tximeta` leverages the digest of the reference transcripts that were indexed in order to identify metadata from the output of quantification tools. A computed digest (a hash value) can be used to uniquely identify the collection of reference sequences, and associate the dataset with other useful metadata. After identification, tximeta uses a number of core Bioconductor packages (GenomicFeatures, ensembldb, AnnotationHub, Seqinfo, BiocFileCache) to automatically populate metadata for the user.

Usage

```
tximeta(
  coldata,
  type = NULL,
  txOut = TRUE,
  skipMeta = FALSE,
  skipSeqinfo = FALSE,
  useHub = TRUE,
  markDuplicateTxps = FALSE,
  cleanDuplicateTxps = FALSE,
  customMetaInfo = NULL,
  skipFtp = FALSE,
  ...
)
```

Arguments

`coldata`	a data.frame with at least two columns (others will propogate to object): • `files` - character, paths of quantification files • `names` - character, sample names if `coldata` is a vector, it is assumed to be the paths of quantification files and unique sample names are created
`type`	what quantifier was used, see `tximport::tximport()`

txOut	whether to output transcript-level data. tximeta is designed to have transcript-level output with salmon, so default is TRUE, and it's recommended to use summarizeToGene following tximeta for gene-level summarization. For an alevin file, tximeta will import the gene level counts ignoring this argument (alevin produces only gene-level quantification).
skipMeta	whether to skip metadata generation (e.g. to avoid errors if not connected to internet). This calls tximport directly and so either txOut=TRUE or tx2gene should be specified.
skipSeqinfo	whether to skip the addition of Seqinfo, which requires an internet connection to download the relevant chromosome information table from UCSC
useHub	whether to first attempt to download a TxDb/EnsDb object from AnnotationHub, rather than creating from a GTF file from FTP (default is TRUE). If FALSE, it will force tximeta to download and parse the GTF
markDuplicateTxps	whether to mark the status (hasDuplicate) and names of duplicate transcripts (duplicates) in the rowData of the SummarizedExperiment output. Subsequent summarization to gene level will keep track of the number of transcripts sets per gene (numDupSets). see Details
cleanDuplicateTxps	whether to "clean" duplicate transcripts (exact sequence duplicates) by replacing, when possible, transcript names that do not appear in the GTF with those that do appear in the GTF. see Details
customMetaInfo	the relative path to a custom metadata information JSON file, relative to the paths in files of coldata. For example, customMetaInfo="meta_info.json" would indicate that in the same directory as the quantification files in files, there are custom metadata information JSON files. These should contain the SHA-256 hash of the reference transcripts with the index_seq_hash tag (see details in vignette).
skipFtp	whether to avoid ftp:// in case of firewall, default is FALSE
...	arguments passed to tximport

Details

Most of the code in tximeta works to add metadata and transcript ranges when the quantification was performed with salmon or related tools. However, tximeta can be used with any quantification type that is supported by `tximport::tximport()`, where it will return an non-ranged SummarizedExperiment. For other quantification tools see also the customMetaInfo argument below. This behavior can also be triggered with skipMeta=TRUE.

tximeta performs a lookup of the digest (or hash value) of the index stored in an auxilary information directory of the quantification tool's output against a database of known transcriptomes, which is stored within the tximeta package (extdata/hashtable.csv) and is continually updated to match Ensembl and GENCODE releases, with updates pushed to Bioconductor current release branch. In addition, tximeta performs a lookup of the digest against a locally stored table of linkedTxome references, see `makeLinkedTxome()`. If tximeta detects a match in either source, it will automatically populate the transcript locations, the transcriptome release, the genome with correct chromosome lengths, and connect the SE object to locally cached derived metadata. tximeta

also facilitates automatic summarization of transcript-level quantifications to the gene-level via `summarizeToGene`` without the need to manually build the correct tx2gene‘ table for the reference used for indexing.

tximeta on the first run will ask where the BiocFileCache::BiocFileCache() location for this package (*tximeta*) should be kept, either using a default location or a temporary directory. At any point, the user can specify a location using setTximetaBFC() and this choice will be saved for future sessions. Multiple users can point to the same BiocFileCache, such that transcript databases (TxDb or EnsDb) associated with certain salmon indices and linkedTxomes can be accessed by different users without additional effort or time spent downloading and building the relevant TxDb / EnsDb. In order to allow that multiple users can read and write to the same location, one should set the BiocFileCache directory to have group write permissions (g+w). Note that, if the TxDb or EnsDb is present in AnnotationHub, tximeta will use this object instead of downloading and building a TxDb/EnsDb from GTF (to disable this set `useHub=FALSE`).

Regarding `markDuplicateTxps` and `cleanDuplicateTxps`: these functions may be used when salmon has been run in default mode (where it reduces exact sequence duplicates in the index by replacing them with a single representative). These won't make sense (and the latter will give an error) when `--keepDuplicates` has been used by salmon during indexing. They help by either adding metadata ("marking") regarding transcript duplicates to the rowData, or changing names of transcripts ("cleaning") where possible to provide better alignment with GTF.

Value

a SummarizedExperiment with metadata on the `rowRanges`. (if the hashed digest in the salmon or Sailfish index does not match any known transcriptomes, or any locally saved `linkedTxome`, `tximeta` will just return a non-ranged SummarizedExperiment)

Examples

```
# point to a salmon quantification file:
dir <- system.file("extdata/salmon_dm", package="tximportData")
files <- file.path(dir, "SRR1197474", "quant.sf")
coldata <- data.frame(files, names="SRR1197474", condition="A", stringsAsFactors=FALSE)

# normally we would just run the following which would download the appropriate metadata
# se <- tximeta(coldata)

# for this example, we instead point to a local path where the GTF can be found
# by making a linkedTxome:
indexDir <- file.path(dir, "Dm.BDGP6.22.98_salmon-0.14.1")
dmFTP <- "ftp://ftp.ensembl.org/pub/release-98/fasta/drosophila_melanogaster/"
fastaFTP <- paste0(
  dmFTP,
  c("cdna/Drosophila_melanogaster.BDGP6.22.cdna.all.fa.gz",
    "ncrna/Drosophila_melanogaster.BDGP6.22.ncrna.fa.gz")
)
gtfPath <- file.path(dir, "Drosophila_melanogaster.BDGP6.22.98.gtf.gz")
makeLinkedTxome(indexDir=indexDir, source="LocalEnsembl", organism="Drosophila melanogaster",
                release="98", genome="BDGP6.22", fasta=fastaFTP, gtf=gtfPath, write=FALSE)
se <- tximeta(coldata)
```

```
# to clear the entire linkedTxome table
# (don't run unless you want to clear this table!)
# bfcloc <- getTximetaBFC()
# bfc <- BiocFileCache(bfcloc)
# bfcremove(bfc, bfcquery(bfc, "linkedTxomeTbl")$rid)
```

`updateMetadata` *Update transcript metadatda for* `importData()` *imported data*

Description

This function takes as input a *SummarizedExperiment* as output by `importData()`, and will update the metadata on the transcripts when possible (updating `rowData` and/or `rowRanges` depending on the value of `ranges`). `importData()` uses metadata pulled from digest matches in registries used by *tximeta* (`linkedTxome`, `linkedTxpData`, and the pre-computed digests). Additionally, *GRanges* or *data.frame*-type data can be provided on a one-time basis via the argument `txpData`, which will annotate transcripts with `index="user"`. See `inspectDigests()` for how to inspect which indices have matching digests, and how to link data to local metadata in a persistent manner.

Usage

```
updateMetadata(
  se,
  txpData = NULL,
  ranges = FALSE,
  prefer = c("txome", "txpdata", "precomputed"),
  order = c("annotated", "novel", "user"),
  key = c(annotated = "tx_name", novel = "tx_name", user = "tx_name")
)
```

Arguments

`se`	the *SummarizedExperiment* (SE) output by `importData()`
`txpData`	either *GRanges* or *data.frame*-type object to use if there is not a match based on digest. This is used on a one-time basis, and transcripts will be marked in metadata columns as `index = "user"`` . See makeLinkedTxome()ormakeLinkedTxpData()‘ for persistent metadata storage/retrieval
`ranges`	logical, whether to add `rowRanges` (or just `rowData`)
`prefer`	vector of length up to 3, giving the preferred order of *tximeta*'s transcript registries to when finding matches, with elements: `txome`: linkedTxome, `txpdata`: linkedTxpData, `precomputed`: the pre-computed digests in tximeta
`order`	order of index, in which to update the metadata, by default the order is `annotation`, then `novel`, then `user`, info supplied here as `txpData`
`key`	a named character vector of length 3. For each index (annotated, novel, and user) `key` is the name of the column to use for merging metadata with `rownames(se)`. The `user` index corresponds to data provided here as `txpData` Defaults to `"tx_name"` which often matches the transcript names in GENCODE

Value

a *SummarizedExperiment* with new `rowData`, or a *RangedSummarizedExperiment* with new metadata

Examples

```
example(importData)

# build custom novel GRanges data
library(GenomicRanges)
novel <- data.frame(
  seqnames = paste0("chr", rep(1:22, each=500)),
  start = 1e6 + 1 + 0:499 * 1000, end = 1e6 + 1 + 0:499 * 1000 + 1000 - 1,
  strand = "+", tx_name = paste0("novel", 1:(22*500)),
  gene_id = paste0("novel_gene", rep(1:(22*10), each=50)), type = "protein_coding"
)
novel_gr <- as(novel, "GRanges")
names(novel_gr) <- novel$tx_name

# now update the metadata + ranges:
## Not run:
# this requires connection to internet (will download GENCODE GTF via FTP)
se_with_ranges <- updateMetadata(
  se, txpData=novel_gr, ranges=TRUE
)
mcols(se_with_ranges)

## End(Not run)
```

Index