Package 'tximport'

December 2, 2025

Version 1.38.1

Title Import and summarize transcript-level estimates for transcript- and gene-level analysis

Description Imports transcript-level abundance, estimated counts and transcript lengths, and summarizes into matrices for use with downstream gene-level analysis packages. Average transcript length, weighted by sample-specific transcript abundance estimates, is provided as a matrix which can be used as an offset for different expression of gene-level counts.

Author Michael Love [cre,aut], Charlotte Soneson [aut], Mark Robinson [aut], Rob Patro [ctb], Andrew Parker Morgan [ctb], Ryan C. Thompson [ctb], Matt Shirley [ctb], Avi Srivastava [ctb]

Maintainer Michael Love <michaelisaiahlove@gmail.com>

License LGPL (>=2)

VignetteBuilder knitr

Imports utils, stats, methods

Suggests knitr, rmarkdown, testthat, tximportData, TxDb.Hsapiens.UCSC.hg19.knownGene, readr (>= 0.2.2), arrow, limma, edgeR, DESeq2 (>= 1.11.6), rhdf5, jsonlite, matrixStats, Matrix, eds

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

biocViews DataImport, Preprocessing, RNASeq, Transcriptomics, Transcription, GeneExpression, ImmunoOncology

RoxygenNote 7.2.3

NeedsCompilation no

Encoding UTF-8

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

git_branch RELEASE_3_22

git_last_commit 78d15a5

git_last_commit_date 2025-11-05

Repository Bioconductor 3.22

Date/Publication 2025-12-01

Contents

`tximport-package` *Tximport package: import transcript-level quantification data*

Description

The tximport package is designed to simplify import of transcript-level abundances (TPM), estimated counts, and effective lengths from a variety of upstream tools, for downstream transcript-level or gene-level analysis. It has no dependencies beyond R, so as to minimize requirements for downstream packages making use of tximport.

Details

The main function has the same name as the package:

- `tximport` - with key arguments: `files`, `type`, `txOut`, and `tx2gene`

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/mikelove/tximport

Author(s)

Charlotte Soneson, Michael I. Love, Mark D. Robinson

References

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

`makeCountsFromAbundance` *Low-level function to make counts from abundance using matrices*

Description

Simple low-level function used within tximport to generate `scaledTPM` or `lengthScaledTPM` counts, taking as input the original counts, abundance and length matrices. NOTE: This is a low-level function exported in case it is needed for some reason, but the recommended way to generate counts-from-abundance is using tximport with the `countsFromAbundance` argument.

Usage

```
makeCountsFromAbundance(
  countsMat,
  abundanceMat,
  lengthMat,
  countsFromAbundance = c("scaledTPM", "lengthScaledTPM")
)
```

Arguments

`countsMat`	a matrix of original counts
`abundanceMat`	a matrix of abundances (typically TPM)
`lengthMat`	a matrix of effective lengths
`countsFromAbundance`	the desired type of count-from-abundance output

Value

a matrix of count-scale data generated from abundances. for details on the calculation see tximport.

`summarizeToGene` *Summarize estimated quantitites to gene-level*

Description

Summarizes abundances, counts, lengths, (and inferential replicates or variance) from transcript- to gene-level.

Usage

```
summarizeToGene(object, ...)

## S4 method for signature 'list'
summarizeToGene(
  object,
  tx2gene,
  varReduce = FALSE,
```

```
  ignoreTxVersion = FALSE,
  ignoreAfterBar = FALSE,
  countsFromAbundance = c("no", "scaledTPM", "lengthScaledTPM")
)
```

Arguments

object	the list of matrices of trancript-level abundances, counts, lengths produced by tximport, with a countsFromAbundance element that tells how the counts were generated.
...	additional arguments, ignored
tx2gene	see tximport
varReduce	see tximport
ignoreTxVersion	see tximport
ignoreAfterBar	see tximport
countsFromAbundance	see tximport

Value

a list of matrices of gene-level abundances, counts, lengths, (and inferential replicates or variance if inferential replicates are present).

See Also

tximport

---

tximport *Import transcript-level abundances and counts for transcript- and gene-level analysis packages*

Description

tximport imports transcript-level estimates from various external software and optionally summarizes abundances, counts, and transcript lengths to the gene-level (default) or outputs transcript-level matrices (see txOut argument).

Usage

```
tximport(
  files,
  type = c("none", "salmon", "sailfish", "alevin", "piscem", "oarfish", "kallisto",
    "rsem", "stringtie"),
  txIn = TRUE,
  txOut = FALSE,
  countsFromAbundance = c("no", "scaledTPM", "lengthScaledTPM", "dtuScaledTPM"),
  tx2gene = NULL,
  varReduce = FALSE,
  dropInfReps = FALSE,
```

```
  infRepStat = NULL,
  ignoreTxVersion = FALSE,
  ignoreAfterBar = FALSE,
  geneIdCol,
  txIdCol,
  abundanceCol,
  countsCol,
  lengthCol,
  importer = NULL,
  existenceOptional = FALSE,
  sparse = FALSE,
  sparseThreshold = 1,
  readLength = 75,
  alevinArgs = NULL
)
```

Arguments

`files` a character vector of filenames for the transcript-level abundances

`type` character, the type of software used to generate the abundances. Options are "salmon", "sailfish", "alevin", "piscem", "oarfish", "kallisto", "rsem", "stringtie", or "none". This argument is used to autofill the arguments below (geneIdCol, etc.) "none" means that the user will specify these columns. Be aware that specifying `type` other than "none" will ignore the arguments below (geneIdCol, etc.)

`txIn` logical, whether the incoming files are transcript level (default TRUE)

`txOut` logical, whether the function should just output transcript-level (default FALSE)

`countsFromAbundance`

character, either "no" (default), "scaledTPM", "lengthScaledTPM", or "dtuScaledTPM". Whether to generate estimated counts using abundance estimates:

- scaled up to library size (scaledTPM),
- scaled using the average transcript length over samples and then the library size (lengthScaledTPM), or
- scaled using the median transcript length among isoforms of a gene, and then the library size (dtuScaledTPM).

dtuScaledTPM is designed for DTU analysis in combination with `txOut=TRUE`, and it requires specifing a `tx2gene` data.frame. dtuScaledTPM works such that within a gene, values from all samples and all transcripts get scaled by the same fixed median transcript length. If using scaledTPM, lengthScaledTPM, or geneLengthScaledTPM, the counts are no longer correlated across samples with transcript length, and so the length offset matrix should not be used.

`tx2gene` a two-column data.frame linking transcript id (column 1) to gene id (column 2). the column names are not relevant, but this column order must be used. this argument is required for gene-level summarization, and the tximport vignette describes how to construct this data.frame (see Details below). An automated solution to avoid having to create `tx2gene` if one has quantified with Salmon or alevin with human or mouse transcriptomes is to use the `tximeta` function from the tximeta Bioconductor package.

`varReduce` whether to reduce per-sample inferential replicates information into a matrix of sample variances `variance` (default FALSE). alevin computes inferential

variance by default for bootstrap inferential replicates, so this argument is ignored/not necessary

`dropInfReps` whether to skip reading in inferential replicates (default FALSE). For alevin, `tximport` will still read in the inferential variance matrix if it exists

`infRepStat` a function to re-compute counts and abundances from the inferential replicates, e.g. `matrixStats::rowMedians` to re-compute counts as the median of the inferential replicates. The order of operations is: first counts are re-computed, then abundances are re-computed. Following this, if `countsFromAbundance` is not "no", `tximport` will again re-compute counts from the re-computed abundances. `infRepStat` should operate on rows of a matrix. (default is NULL)

`ignoreTxVersion`
logical, whether to split the tx id on the '.' character to remove version information to facilitate matching with the tx id in `tx2gene` (default FALSE)

`ignoreAfterBar` logical, whether to split the tx id on the '|' character to facilitate matching with the tx id in `tx2gene` (default FALSE). if `txOut=TRUE` it will strip the text after '|' on the rownames of the matrices

`geneIdCol` name of column with gene id. if missing, the `tx2gene` argument can be used. Note that this argument and the other four "...Col" arguments below are ignored unless `type="none"`

`txIdCol` name of column with tx id

`abundanceCol` name of column with abundances (e.g. TPM or FPKM)

`countsCol` name of column with estimated counts

`lengthCol` name of column with feature length information

`importer` a function used to read in the files

`existenceOptional`
logical, should tximport not check if files exist before attempting import (default FALSE, meaning files must exist according to `file.exists`)

`sparse` logical, whether to try to import data sparsely (default is FALSE). Initial implementation for `txOut=TRUE`, `countsFromAbundance="no"` or `"scaledTPM"`, no inferential replicates. Only counts matrix is returned (and abundance matrix if using `"scaledTPM"`)

`sparseThreshold`
the minimum threshold for including a count as a non-zero count during sparse import (default is 1)

`readLength` numeric, the read length used to calculate counts from StringTie's output of coverage. Default value (from StringTie) is 75. The formula used to calculate counts is: `cov * transcript length / read length`

`alevinArgs` named list, with logical elements `filterBarcodes`, `tierImport`, `forceSlow`, `dropMeanVar`. See Details for definitions.

Details

Inferential replicates: `tximport` will also load in information about inferential replicates – a list of matrices of the Gibbs samples from the posterior, or bootstrap replicates, per sample – if these data are available in the expected locations relative to the `files`. The inferential replicates, stored in `infReps` in the output list, are on estimated counts, and therefore follow `counts` in the output list. By setting `varReduce=TRUE`, the inferential replicate matrices will be replaced by a single matrix with the sample variance per transcript/gene and per sample.

summarizeToGene: While `tximport` summarizes to the gene-level by default, the user can also perform the import and summarization steps manually, by specifing `txOut=TRUE` and then using the function `summarizeToGene`. Note however that this is equivalent to `tximport` with `txOut=FALSE` (the default).

Solutions on summarization: regarding `"tximport failed at summarizing to the gene-level"`:

1. provide a `tx2gene` data.frame linking transcripts to genes (more below)
2. avoid gene-level summarization by specifying `txOut=TRUE`

See `vignette('tximport')` for example code for generating a `tx2gene` data.frame from a `TxDb` object. The `tx2gene` data.frame should exactly match and be derived from the same set of transcripts used for quantifying (the set of transcript used to create the transcriptome index).

Tximeta: One automated solution for Salmon/alevin/piscem/oarfish quantification data is to use the `tximeta` function in the tximeta Bioconductor package which builds upon and extends `tximport`; this solution should work out-of-the-box for human and mouse transcriptomes downloaded from GENCODE, Ensembl, or RefSeq. For other cases, the user should create the `tx2gene` manually as shown in the tximport vignette.

On tx2gene construction: Note that the `keys` and `select` functions used to create the `tx2gene` object are documented in the man page for AnnotationDb-class objects in the AnnotationDbi package (TxDb inherits from AnnotationDb). For further details on generating TxDb objects from various inputs see `vignette('GenomicFeatures')` from the GenomicFeatures package.

alevin: The `alevinArgs` argument includes some alevin-specific arguments. This optional argument is a list with any or all of the following named logical variables: `filterBarcodes`, `tierImport`, and `forceSlow`. The variables are described as follows (with default values in parens): `filterBarcodes` (FALSE) import only cell barcodes listed in `whitelist.txt`; `tierImport` (FALSE) import the tier information in addition to counts; `forceSlow` (FALSE) force the use of the slower import R code even if `eds` is installed; `dropMeanVar` (FALSE) don't import inferential mean and variance matrices even if they exist (also skips inferential replicates) For `type="alevin"` all arguments other than `files`, `dropInfReps`, and `alevinArgs` are ignored. Note that `files` should point to a single `quants_mat.gz` file, in the directory structure created by the alevin software (e.g. do not move the file or delete the other important files). Note that importing alevin quantifications will be much faster by first installing the `eds` package, which contains a C++ importer for alevin's EDS format. For alevin, `tximport` is importing the gene-by-cell matrix of counts, as `txi$counts`, and effective lengths are not estimated. `txi$mean` and `txi$variance` may also be imported if inferential replicates were used, as well as inferential replicates if these were output by alevin. Length correction should not be applied to datasets where there is not an expected correlation of counts and feature length.

Value

A simple list containing matrices: abundance, counts, length. Another list element 'countsFromAbundance' carries through the character argument used in the tximport call. The length matrix contains the average transcript length for each gene which can be used as an offset for gene-level analysis. If detected, and `txOut=TRUE`, inferential replicates for each sample will be imported and stored as a list of matrices, itself an element `infReps` in the returned list. An exception is alevin, in which the `infReps` are a list of bootstrap replicate matrices, where each matrix has genes as rows and cells as columns. If `varReduce=TRUE` the inferential replicates will be summarized according to the sample variance, and stored as a matrix `variance`. alevin already computes the variance of the bootstrap inferential replicates and so this is imported without needing to specify `varReduce=TRUE`.

References

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

Examples

```
# load data for demonstrating tximport
# note that the vignette shows more examples
# including how to read in files quickly using the readr package

library(tximportData)
dir <- system.file("extdata", package="tximportData")
samples <- read.table(file.path(dir,"samples.txt"), header=TRUE)
files <- file.path(dir,"salmon", samples$run, "quant.sf.gz")
names(files) <- paste0("sample",1:6)

# tx2gene links transcript IDs to gene IDs for summarization
tx2gene <- read.csv(file.path(dir, "tx2gene.gencode.v27.csv"))

txi <- tximport(files, type="salmon", tx2gene=tx2gene)
```

Index