Package ‘seeker’

January 22, 2024

Type Package

Title Simplified Fetching and Processing of Microarray and RNA-Seq Data

Version 1.1.5

Description Wrapper around various existing tools and command-line interfaces, providing a standard interface, simple parallelization, and detailed logging. For microarray data, maps probe sets to standard gene IDs, building on 'GEOquery' Davis and Meltzer (2007) <doi:10.1093/bioinformatics/btm254>, 'ArrayExpress' Kauffmann et al. (2009) <doi:10.1093/bioinformatics/btp354>, Robust multi-array average 'RMA' Irizarry et al. (2003) <doi:10.1093/biostatistics/4.2.249>, and 'BrainArray' Dai et al. (2005) <doi:10.1093/nar/gni179>. For RNA-seq data, fetches metadata and raw reads from National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA), performs standard adapter and quality trimming using 'TrimGalore' Krueger <`https://github.com/FelixKrueger/TrimGalore`>, performs quality control checks using 'FastQC' Andrews <`https://github.com/s-andrews/FastQC`>, quantifies transcript abundances using 'salmon' Patro et al. (2017) <doi:10.1038/nmeth.4197> and potentially 'refgenie' Stolarczyk et al. (2020) <doi:10.1093/gigascience/giz149>, aggregates the results using 'MultiQC' Ewels et al. (2016) <doi:10.1093/bioinformatics/btw354>, maps transcripts to genes using 'biomaRt' Durinkck et al. (2009) <doi:10.1038/nprot.2009.97>, and summarizes transcript-level quantifications for gene-level analyses using 'tximport' Soneson et al. (2015) <doi:10.12688/f1000research.7563.2>.

URL `https://seeker.hugheylab.org`, `https://github.com/hugheylab/seeker`

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.5)

Imports affy (>= 1.68.0), AnnotationDbi (>= 1.52.0), BiocManager (>= 1.30.0), biomaRt (>= 2.36.1), checkmate (>= 2.1.0), curl (>= 3.2), data.table (>= 1.11.8), foreach (>= 1.4.4), GEOquery (>= 2.58.0), glue (>= 1.5.0), jsonlite (>= 1.7.2), methods, qs (>=

0.21.2), R.utils (>= 2.11.0), RCurl (>= 1.98), readr (>= 1.4.0), sessioninfo (>= 1.2.0), tximport (>= 1.8.0), withr (>= 2.4.2), yaml (>= 2.2.1)

Suggests ArrayExpress (>= 1.62.0), Biobase, doParallel (>= 1.0.17), knitr, org.Mm.eg.db, rmarkdown, testthat (>= 3.1.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Jake Hughey [aut, cre], Josh Schoenbachler [aut]

Maintainer Jake Hughey <`jakejhughey@gmail.com`>

Repository CRAN

Date/Publication 2024-01-22 18:50:02 UTC

R topics documented:

`checkDefaultCommands` *Check for presence of command-line interfaces*

Description

This function checks whether the command-line tools used by seeker are accessible in the expected places.

Usage

```
checkDefaultCommands(keepIdx = FALSE)
```

Arguments

`keepIdx`	Logical indicating whether to keep the `idx` column of the resulting data.table. For internal use only.

Value

A data.table with columns for command, path, and version.

See Also

`installSysDeps()`

`fastqc` *Run FastQC*

Description

This function calls fastqc using `system2()`. To run in parallel, register a parallel backend, e.g., using `doParallel::registerDoParallel()`.

Usage

```
fastqc(filepaths, outputDir = "fastqc_output", cmd = "fastqc", args = NULL)
```

Arguments

`filepaths`	Paths to fastq files. For single-end reads, each element should be a single filepath. For paired-end reads, each element can be two filepaths separated by ";".
`outputDir`	Directory in which to store output. Will be created if it doesn't exist.
`cmd`	Name or path of the command-line interface.
`args`	Additional arguments to pass to the command-line interface.

Value

A vector of exit codes, invisibly.

See Also

`seeker()`

fastqscreen *Run FastQ Screen*

Description

This function calls fastq_screen using `system2()`. To run in parallel, register a parallel backend, e.g., using `doParallel::registerDoParallel()`.

Usage

```
fastqscreen(
  filepaths,
  outputDir = "fastqscreen_output",
  cmd = "fastq_screen",
  args = c("--threads", foreach::getDoParWorkers(), "--conf",
    "~/FastQ_Screen_Genomes/fastq_screen.conf")
)
```

Arguments

`filepaths`	Paths to fastq files. For single-end reads, each element should be a single filepath. For paired-end reads, each element can be two filepaths separated by ";".
`outputDir`	Directory in which to store output. Will be created if it doesn't exist.
`cmd`	Name or path of the command-line interface.
`args`	Additional arguments to pass to the command-line interface.

Value

A vector of exit codes, invisibly.

See Also

`seeker()`

fetch *Fetch files*

Description

This function uses the NCBI SRA Toolkit via `system2()` to download files from SRA and convert them to fastq.gz. To process files in parallel, register a parallel backend, e.g., using `doParallel::registerDoParallel()`. Beware that intermediate files created by fasterq-dump are uncompressed and could require hundreds of gigabytes if files are processed in parallel.

Usage

```
fetch(
  accessions,
  outputDir,
  overwrite = FALSE,
  keepSra = FALSE,
  prefetchCmd = "prefetch",
  prefetchArgs = NULL,
  fasterqdumpCmd = "fasterq-dump",
  fasterqdumpArgs = NULL,
  pigzCmd = "pigz",
  pigzArgs = NULL
)
```

Arguments

`accessions`	Character vector of SRA run accessions.
`outputDir`	String indicating the local directory in which to save the files. Will be created if it doesn't exist.
`overwrite`	Logical indicating whether to overwrite files that already exist in `outputDir`.
`keepSra`	Logical indicating whether to keep the ".sra" files.
`prefetchCmd`	String indicating command for prefetch, which downloads ".sra" files.
`prefetchArgs`	Character vector indicating arguments to pass to prefetch.
`fasterqdumpCmd`	String indicating command for fasterq-dump, which uses ".sra" files to create ".fastq" files.
`fasterqdumpArgs`	Character vector indicating arguments to pass to fasterq-dump.
`pigzCmd`	String indicating command for pigz, which converts ".fastq" files to ".fastq.gz" files.
`pigzArgs`	Character vector indicating arguments to pass to pigz.

Value

A list. As the function runs, it updates a tab-delimited log file in `outputDir` called "progress.tsv".

See Also

`seeker()`, `fetchMetadata()`

fetchMetadata *Fetch metadata for a genomic study*

Description

This function can use the API of the European Nucleotide Archive (recommended) or the Sequence Read Archive.

Usage

```
fetchMetadata(
  bioproject,
  host = c("ena", "sra"),
 fields = c("study_accession", "sample_accession", "secondary_sample_accession",
   "sample_alias", "sample_title", "experiment_accession", "run_accession", "fastq_md5",
    "fastq_ftp", "fastq_aspera"),
  file = NULL
)
```

Arguments

`bioproject`	String indicating bioproject accession.
`host`	String indicating from where to fetch the metadata.
`fields`	Character vector indicating which fields to fetch, if `host` is "ena".
`file`	String indicating output file path, if not `NULL`.

Value

A `data.table`.

See Also

`seeker()`, `fetch()`

getPlatforms *Get supported microarray platforms*

Description

Get supported microarray platforms

Usage

```
getPlatforms(type = c("cdf", "mapping"))
```

Arguments

`type`	String indicating whether to get supported platforms for processing raw Affymetrix data using custom CDF or for mapping already processed data from probes to genes.

Value

A `data.table`.

`getSalmonMetadata` *Aggregrate metadata from salmon quantifications*

Description

Aggregrate metadata from salmon quantifications

Usage

```
getSalmonMetadata(inputDir, outputDir = "data")
```

Arguments

`inputDir`	Directory that contains output from salmon.
`outputDir`	Directory in which to save the result, a file named "salmon_meta_info.csv". If `NULL`, no file is saved.

Value

A data.table, invisibly.

#' @seealso `seeker()`, `salmon()`

`getTx2gene` *Get mapping between transcripts and genes*

Description

This function uses the biomaRt package.

Usage

```
getTx2gene(
  organism = "mmusculus",
  version = NULL,
  outputDir = "data",
  checkArgsOnly = FALSE
)
```

Arguments

`organism`	String used to pass `paste0(organism, "_gene_ensembl")` as the `dataset` argument to `biomaRt::useEnsembl()`. To see available datasets, do `mart = biomaRt::useEnsembl("gene`
`version`	Passed to `biomaRt::useEnsembl()`. `NULL` indicates the latest version. To see available versions, do `biomaRt::listEnsemblArchives()`.
`outputDir`	Directory in which to save the result, a file named "tx2gene.csv.gz". If `NULL`, no file is saved.
`checkArgsOnly`	Logical indicating whether to only check function arguments. Used for testing.

Value

If `checkArgsOnly` is `FALSE`, a data.table based on the result from `biomaRt::getBM()`, with an attribute "version". Otherwise `0`.

See Also

`seeker()`, `tximport()`

---

installCustomCdfPackages

Install custom CDF packages

Description

Install Brainarray custom CDFs for processing raw Affymetrix data. See `http://brainarray.mbni.med.umich.edu/Brainarray/Database/CustomCDF/CDF_download.asp`.

Usage

```
installCustomCdfPackages(pkgs, ver = 25, dryRun = FALSE)
```

Arguments

`pkgs`	Character vector of package names, e.g., "hgu133ahsentrezgcdf".
`ver`	Integer version number (25 as of 5 Jan 2021).
`dryRun`	Logical indicating whether to actually install the packages.

Value

A character vector of URLs, invisibly.

installSysDeps *Install seeker's system dependencies*

Description

This function installs and configures the various programs required for seeker to fetch and process RNA-seq data.

Usage

```
installSysDeps(
  sraToolkitDir,
  minicondaDir,
  refgenieDir,
  rprofileDir,
  minicondaEnv = "seeker",
  refgenieGenomes = NULL,
  fastqscreenDir = NULL
)
```

Arguments

`sraToolkitDir`	String indicating directory in which to install the SRA Toolkit. Recommended to use "~", the home directory. If `NULL`, the Toolkit will not be installed.
`minicondaDir`	String indicating directory in which to install Miniconda. Recommended to use "~", the home directory. If `NULL`, Miniconda will not be installed.
`refgenieDir`	String indicating directory in which to store the directory of genome assets from refgenie, which will be named "refgenie_genomes". Recommended to use "~", the home directory. Only used if `minicondaDir` is not `NULL`.
`rprofileDir`	String indicating directory in which to create or modify .Rprofile, which is run by R on startup. Common options are "~" or ".".
`minicondaEnv`	String indicating name of the Miniconda environment in which to install various conda packages (fastq-screen, fastqc, multiqc, pigz, refgenie, salmon, and trim-galore).
`refgenieGenomes`	Character vector indicating genome assets, such as transcriptome indexes for `salmon()`, to pull from refgenomes using refgenie. If `NULL`, no assets are fetched.
`fastqscreenDir`	String indicating directory in which to download the genomes for `fastqscreen()`. This takes a long time. If `NULL`, genomes are not downloaded.

Value

`NULL`, invisibly

See Also

`seeker()`

multiqc *Run MultiQC*

Description

This function calls multiqc using `system2()`.

Usage

```
multiqc(
  parentDir = ".",
  outputDir = "multiqc_output",
  cmd = "multiqc",
  args = NULL
)
```

Arguments

`parentDir`	Directory that contains output to be aggregated.
`outputDir`	Directory in which to store output. Will be created if it doesn't exist.
`cmd`	Name or path of the command-line interface.
`args`	Additional arguments to pass to the command-line interface.

Value

An exit code, invisibly.

See Also

`seeker()`

salmon *Run Salmon*

Description

This function calls salmon using `system2()`. To run in parallel, register a parallel backend, e.g., using `doParallel::registerDoParallel()`.

Usage

```
salmon(
  filepaths,
  samples,
  indexDir,
  outputDir = "salmon_output",
  cmd = "salmon",
  args = c("-l A -q --seqBias --gcBias --no-version-check -p",
    foreach::getDoParWorkers()),
  compress = TRUE
)
```

Arguments

`filepaths`	Paths to fastq files. For single-end reads, each element should be a single filepath. For paired-end reads, each element should be two filepaths separated by ";".
`samples`	Corresponding sample names for fastq files.
`indexDir`	Directory that contains salmon index.
`outputDir`	Directory in which to store output. Will be created if it doesn't exist.
`cmd`	Name or path of the command-line interface.
`args`	Additional arguments to pass to the command-line interface.
`compress`	Logical indicating whether to gzip the quantification file (quant.sf) from salmon. Does not affect downstream analysis.

Value

A vector of exit codes, invisibly.

See Also

`seeker()`, `getSalmonMetadata()`

`seeker` *Process RNA-seq data end to end*

Description

This function selectively performs various steps to process RNA-seq data. See also the vignettes: `browseVignettes('seeker')`.

Usage

```
seeker(params, parentDir = ".", dryRun = FALSE)
```

Arguments

`params` Named list of parameters with components:

- `study`: String used to name the output directory within `parentDir`.
- `metadata`: Named list with components:
  - `run`: Logical indicating whether to fetch metadata. See `fetchMetadata()`. If `TRUE`, saves a file `parentDir`/study/metadata.csv. If `FALSE`, expects that file to already exist. The unmodified fetched or found metadata is saved to a file `parentDir`/study/metadata_original.csv. Following components are only checked if `run` is `TRUE`.
  - `bioproject`: String indicating the study's bioproject accession.
  - `include`: Optional named list for specifying which rows of metadata to include for further processing, with components:
    - `colname`: String indicating column in metadata
    - `values`: Vector indicating values within `colname`
  - `exclude`: Optional named list for specifying which rows of metadata to exclude from further processing (superseding `include`), with components:
    - `colname`: String indicating column in metadata
    - `values`: Vector indicating values within `colname`
- `fetch`: Named list with components:
  - `run`: Logical indicating whether to fetch files from SRA. See `fetch()`. If `TRUE`, saves files to `parentDir`/study/fetch_output. Whether `TRUE` or `FALSE`, expects metadata to have a column "run_accession", and updates metadata with column "fastq_fetched" containing paths to files in `parentDir`/study/fetch_output. Following components are only checked if `run` is `TRUE`.
  - `keep`: Logical indicating whether to keep fastq.gz files when all processing steps have completed. `NULL` indicates `TRUE`.
  - `overwrite`: Logical indicating whether to overwrite files that already exist. `NULL` indicates to use the default in `fetch()`.
  - `keepSra`: Logical indicating whether to keep the ".sra" files. `NULL` indicates to use the default in `fetch()`.
  - `prefetchCmd`: String indicating command for prefetch, which downloads ".sra" files. `NULL` indicates to use the default in `fetch()`.
  - `prefetchArgs`: Character vector indicating arguments to pass to prefetch. `NULL` indicates to use the default in `fetch()`.
  - `fasterqdumpCmd`: String indicating command for fasterq-dump, which uses ".sra" files to create ".fastq" files. `NULL` indicates to use the default in `fetch()`.
  - `prefetchArgs`: Character vector indicating arguments to pass to fasterq-dump. `NULL` indicates to use the default in `fetch()`.
  - `pigzCmd`: String indicating command for pigz, which converts ".fastq" files to ".fastq.gz" files. `NULL` indicates to use the default in `fetch()`.
  - `pigzArgs`: Character vector indicating arguments to pass to pigz. `NULL` indicates to use the default in `fetch()`.

- `trimgalore`: Named list with components:
  - `run`: Logical indicating whether to perform quality/adapter trimming of reads. See `trimgalore()`. If `TRUE`, expects metadata to have a column "fastq_fetched" containing paths to fastq files in `parentDir`/study/fetch_output, saves trimmed files to `parentDir`/study/trimgalore_output, and updates metadata with column "fastq_trimmed". If `FALSE`, expects and does nothing. Following components are only checked if `run` is `TRUE`.
  - `keep`: Logical indicating whether to keep trimmed fastq files when all processing steps have completed. `NULL` indicates `TRUE`.
  - `cmd`: Name or path of the command-line interface. `NULL` indicates to use the default in `trimgalore()`.
  - `args`: Additional arguments to pass to the command-line interface. `NULL` indicates to use the default in `trimgalore()`.
  - `pigzCmd`: String indicating command for pigz, which converts ".fastq" files to ".fastq.gz" files. `NULL` indicates to use the default in `trimgalore()`.
- `fastqc`: Named list with components:
  - `run`: Logical indicating whether to perform QC on reads. See `fastqc()`. If `TRUE` and `trimgalore$run` is `TRUE`, expects metadata to have a column "fastq_trimmed" containing paths to fastq files in `parentDir`/study/trimgalore_output. If `TRUE` and `trimgalore$run` is `FALSE`, expects metadata to have a column "fastq_fetched" containing paths to fastq files in `parentDir`/study/fetch_output. If `TRUE`, saves results to `parentDir`/study/fastqc_output. If `FALSE`, expects and does nothing. Following components are only checked if `run` is `TRUE`.
  - `keep`: Logical indicating whether to keep fastqc files when all processing steps have completed. `NULL` indicates `TRUE`.
  - `cmd`: Name or path of the command-line interface. `NULL` indicates to use the default in `fastqc()`.
  - `args`: Additional arguments to pass to the command-line interface. `NULL` indicates to use the default in `fastqc()`.
- `salmon`: Named list with components:
  - `run`: Logical indicating whether to quantify transcript abundances. See `salmon()`. If `TRUE` and `trimgalore$run` is `TRUE`, expects metadata to have a column "fastq_trimmed" containing paths to fastq files in `parentDir`/study/trimgalore_output. If `TRUE` and `trimgalore$run` is `FALSE`, expects metadata to have a column "fastq_fetched" containing paths to fastq files in `parentDir`/study/fetch_output. If `TRUE`, saves results to `parentDir`/study/salmon_output and `parentDir`/study/salmon_meta_info.csv. If `FALSE`, expects and does nothing. Following components are only checked if `run` is `TRUE`.
  - `indexDir`: Directory that contains salmon index.
  - `sampleColname`: String indicating column in metadata containing sample ids. `NULL` indicates "sample_accession", which should work for data from SRA and ENA.
  - `keep`: Logical indicating whether to keep quantification results when all processing steps have completed. `NULL` indicates `TRUE`.

  - `cmd`: Name or path of the command-line interface. `NULL` indicates to use the default in `salmon()`.
  - `args`: Additional arguments to pass to the command-line interface. `NULL` indicates to use the default in `salmon()`.
- `multiqc`: Named list with components:
  - `run`: Logical indicating whether to aggregrate results of various processing steps. See `multiqc()`. If `TRUE`, saves results to `parentDir/study/`multiqc_output. If `FALSE`, expects and does nothing. Following components are only checked if `run` is `TRUE`.
  - `cmd`: Name or path of the command-line interface. `NULL` indicates to use the default in `multiqc()`.
  - `args`: Additional arguments to pass to the command-line interface. `NULL` indicates to use the default in `multiqc()`.
- `tximport`: Named list with components:
  - `run`: Logical indicating whether to summarize transcript- or gene-level estimates for downstream analysis. See `tximport()`. If `TRUE`, expects metadata to have a column `sampleColname` of sample ids, and expects a directory `parentDir/study/`salmon_output containing directories of quantification results, and saves results to `parentDir/study/`tximport_output.qs. If `FALSE`, expects and does nothing. Following components are only checked if `run` is `TRUE`.
  - `tx2gene`: Optional named list with components:
    * `organism`: String indicating organism and thereby ensembl gene dataset. See `getTx2gene()`.
    * `version`: Optional number indicating ensembl version. `NULL` indicates the latest version. See `getTx2gene()`.
    * `filename`: Optional string indicating name of pre-existing text file in `parentDir/params$study` containing mapping between transcripts (first column) and genes (second column), with column names in the first row. If `filename` is specified, `organism` and `version` must not be specified.

    If not `NULL`, saves a file `parentDir/study/`tx2gene.csv.gz.
  - `countsFromAbundance`: String indicating whether or how to estimate counts using estimated abundances. See `tximport::tximport()`.
  - `ignoreTxVersion`: Logical indicating whether to the version suffix on transcript ids. `NULL` indicates to use `TRUE`. See `tximport::tximport()`.

`params` can be derived from a yaml file, see `vignette("introduction", package = "seeker")`. The yaml representation of `params` will be saved to `parentDir/params$study/`params.yml

`parentDir`	Directory in which to store the output, which will be a directory named according to `params$study`.
`dryRun`	Logical indicating whether to check the validity of inputs without actually fetching or processing any data.

Value

Path to the output directory `parentDir/params$study`, invisibly.

See Also

fetchMetadata(), fetch(), trimgalore(), fastqc(), salmon(), multiqc(), tximport(), installSysDeps(), seekerArray()

Examples

```
## Not run:
doParallel::registerDoParallel()
params = yaml::read_yaml('my_params.yaml')
seeker(params)

## End(Not run)
```

---

seekerArray *Process microarray data end to end*

Description

This function fetches data and metadata from NCBI GEO and ArrayExpress, processes raw Affymetrix data using RMA and custom CDFs from Brainarray, and maps probes to genes. See also the vignettes: `browseVignettes('seeker')`.

Usage

```
seekerArray(
  study,
  geneIdType,
  platform = NULL,
  parentDir = ".",
  metadataOnly = FALSE
)
```

Arguments

`study`	String indicating the study accession and used to name the output directory within `parentDir`. Must start with "GSE", "E-", or "LOCAL". If starts with "GSE", data are fetched using `GEOquery::getGEO()`. If starts with "E-", data are fetched using `ArrayExpress::getAE()`. If starts with "LOCAL", data in the form of cel(.gz) files must in the directory `parentDir`/study/raw, and `parentDir`/study must contain a file "sample_metadata.csv" that has a column `sample_id` containing the names of the cel(.gz) files without the file extension.
`geneIdType`	String indicating whether to map probes to gene IDs from Ensembl ("ensembl") or Entrez ("entrez").
`platform`	String indicating the GEO-based platform accession for the raw data. See https://www.ncbi.nlm.nih.gov/geo/browse/?view=platforms. Only necessary if `study` starts with "LOCAL", or starts with "GSE" and the study uses multiple platforms.

`parentDir`	Directory in which to store the output, which will be a directory named according to `study`.
`metadataOnly`	Logical indicating whether to only process the sample metadata, and skip processing the expression data.

Details

The standard output:

- naive_expression_set.qs: Initial `ExpresssionSet` generated by GEOquery::getGEO or `ArrayExpress::ae2bioc()`. Should generally *not* be used if sample_metadata.csv and gene_expression_matrix.qs are available.
- sample_metadata.csv: Table of sample metadata. Column `sample_id` matches colnames of the gene expression matrix.
- gene_expression_matrix.qs: Rows correspond to genes, columns to samples. Expression values are log2-transformed.
- custom_cdf_name.txt: Name of custom CDF package used by `affy::justRMA()` to process and normalize raw Affymetrix data and map probes to genes.
- feature_metadata.qs: `GPL` object, if gene expression matrix was generated from processed data.
- probe_gene_mapping.csv.gz: Table of probes and genes, if gene expression matrix was generated from processed data.
- "raw" directory: Contains raw Affymetrix files.
- params.yml: Parameters used to process the dataset.
- session.log: R session information.

The output may include other files from NCBI GEO or ArrayExpress. Files with extension "qs" can be read into R using `qs::qread()`.

Value

Path to the output directory `parentDir/study`, invisibly.

See Also

`seeker()`

Examples

```
## Not run:
seekerArray('GSE25585', 'entrez')

## End(Not run)
```

trimgalore *Run Trim Galore!*

Description

This function calls trim_galore using `system2()`, and is only designed to handle standard adapter/quality trimming. To run in parallel, register a parallel backend, e.g., using `doParallel::registerDoParallel()`.

Usage

```
trimgalore(
  filepaths,
  outputDir = "trimgalore_output",
  cmd = "trim_galore",
  args = NULL,
  pigzCmd = "pigz"
)
```

Arguments

`filepaths`	Paths to fastq files. For single-end reads, each element should be a single filepath. For paired-end reads, each element should be two filepaths separated by ";".
`outputDir`	Directory in which to store output. Will be created if it doesn't exist.
`cmd`	Name or path of the command-line interface.
`args`	Additional arguments to pass to the command-line interface. Output files will always be compressed. Arguments "–gzip", "–cores", "-j", and "–basename" are not allowed. Arguments "-o" and "–paired" should not be specified here.
`pigzCmd`	String for pigz command, which will gzip the output files.

Value

A vector of exit codes, invisibly.

See Also

`seeker()`

tximport *Run tximport on RNA-seq quantifications*

Description

This function uses the tximport package.

Usage

```
tximport(
  inputDir,
  tx2gene,
  samples = NULL,
  outputDir = "data",
  type = c("salmon", "kallisto"),
  countsFromAbundance = "lengthScaledTPM",
  ignoreTxVersion = TRUE,
  ...
)
```

Arguments

`inputDir`	Directory that contains the quantification directories.
`tx2gene`	`NULL` or data.frame of mapping between transcripts and genes, as returned by `getTx2gene()`, passed to `tximport::tximport()`.
`samples`	Names of quantification directories to include. `NULL` indicates all.
`outputDir`	Directory in which to save the result, a file named "tximport_output.qs", using `qs::qsave()`. If `NULL`, no file is saved.
`type`	Passed to `tximport::tximport()`.
`countsFromAbundance`	Passed to `tximport::tximport()`.
`ignoreTxVersion`	Passed to `tximport::tximport()`.
`...`	Additional arguments passed to `tximport::tximport()`.

Value

A list, as returned by `tximport::tximport()`, invisibly.

See Also

`seeker()`, `getTx2gene()`

Index