Package 'recount'

May 17, 2022

Title Explore and download data from the recount project

Version 1.22.0

Date 2021-11-21

Depends R (>= 3.3.0), SummarizedExperiment

Imports BiocParallel, derfinder, downloader, GEOquery, GenomeInfoDb, GenomicRanges, IRanges, methods, RCurl, rentrez, rtracklayer (>= 1.35.3), S4Vectors, stats, utils

Suggests AnnotationDbi, BiocManager, BiocStyle (>= 2.5.19), DESeq2, sessioninfo, EnsDb.Hsapiens.v79, GenomicFeatures, knitr (>= 1.6), org.Hs.eg.db, RefManageR, regionReport (>= 1.9.4), rmarkdown (>= 0.9.5), testthat (>= 2.1.0), covr, pheatmap, DT, edgeR, ggplot2, RColorBrewer

VignetteBuilder knitr

Description Explore and download data from the recount project available at https://jhubiostatistics.shinyapps.io/recount/. Using the recount package you can download RangedSummarizedExperiment objects at the gene, exon or exon-exon junctions level, the raw counts, the phenotype metadata used, the urls to the sample coverage bigWig files or the mean coverage bigWig file for a particular study. The RangedSummarizedExperiment objects can be used by different packages for performing differential expression analysis. Using http://bioconductor.org/packages/derfinder you can perform annotation-agnostic differential expression analyses with the data from the recount project as described at http://www.nature.com/nbt/journal/v35/n4/full/nbt.3838.html.

License Artistic-2.0

Encoding UTF-8

LazyData true

LazyDataCompression xz

URL https://github.com/leekgroup/recount

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

biocViews Coverage, DifferentialExpression, GeneExpression, RNASeq, Sequencing, Software, DataImport, ImmunoOncology

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

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

git_branch RELEASE_3_15

git_last_commit 514a4a1

git_last_commit_date 2022-04-26

Date/Publication 2022-05-17

Author Leonardo Collado-Torres [aut, cre]
(<https://orcid.org/0000-0003-2140-308X>),
Abhinav Nellore [ctb],
Andrew E. Jaffe [ctb] (<https://orcid.org/0000-0001-6886-1454>),
Margaret A. Taub [ctb],
Kai Kammers [ctb],
Shannon E. Ellis [ctb] (<https://orcid.org/0000-0002-9231-0481>),
Kasper Daniel Hansen [ctb] (<https://orcid.org/0000-0003-0086-0687>),
Ben Langmead [ctb] (<https://orcid.org/0000-0003-2437-1976>),
Jeffrey T. Leek [aut, ths] (<https://orcid.org/0000-0002-2873-2671>)

Maintainer Leonardo Collado-Torres <lcolladotor@gmail.com>

R topics documented:

`abstract_search` *Search the abstracts from the SRA studies available via the recount project*

Description

Given a text query, find the SRA project ids (study accession numbers) that contain the text in their abstract as provided by the SRAdb Bioconductor package.

Usage

```
abstract_search(query, id_only = FALSE, ...)
```

Arguments

`query`	A character vector with the text to search for via grep in the abstract info available at recount_abstract.
`id_only`	Whether to only return the project id or to return summary information for the project(s) that match the query.
`...`	Additional arguments passed to grep.

Details

Both the query and the abstracts are searched in lower case.

For a more powerful search use the recount project website at https://jhubiostatistics.shinyapps.io/recount/.

Value

If `id_only = TRUE` it returns a character vector with the project SRA ids (accession numbers). If `id_only = FALSE` it returns a subset of recount_abstract for the abstracts that contained the query.

Author(s)

Leonardo Collado-Torres

See Also

browse_study, recount_abstract

Examples

```
## Find the Geuvadis consortium project
project_info <- abstract_search("Geuvadis consortium")

## See some summary information for this project
project_info
```

add_metadata *Add additional curated metadata to a recount rse object*

Description

This function appends sample metadata information to a RangedSummarizedExperiment-class from the recount2 project. The sample metadata comes from curated efforts independent from the original recount2 project. Currently the only information comes from the recount_brain project described in more detail at http://lieberinstitute.github.io/recount-brain/.

Usage

```
add_metadata(rse, source = "recount_brain_v2", is_tcga = FALSE, verbose = TRUE)
```

Arguments

rse	A RangedSummarizedExperiment-class object as downloaded with download_study. If this argument is not specified, the function will return the raw metadata table.
source	A valid source name. The only supported options at this moment are recount_brain_v1 and recount_brain_v2.
is_tcga	Set to TRUE only when rse is from TCGA. Otherwise set to FALSE (default).
verbose	If TRUE it will print a message of where the predictions file is being downloaded to.

Details

For source = "recount_brain_v1" and source = "recount_brain_v2", the metadata columns are described at http://lieberinstitute.github.io/recount-brain/. Alternatively, you can explore recount_brain_v2 interactively at https://jhubiostatistics.shinyapps.io/recount-brain/.

If you use the recount_brain data please cite the Razmara et al. bioRxiv, 2019 https://www.biorxiv.org/content/10.1101/618025v1. A bib file is available via citation('recount').

Value

A RangedSummarizedExperiment-class object with the sample metadata columns appended to the colData() slot.

Author(s)

Leonardo Collado-Torres

References

Razmara et al, bioRxiv, 2019. https://www.biorxiv.org/content/10.1101/618025v1

Examples

```
## Add the sample metadata to an example rse_gene object
rse_gene <- add_metadata(rse_gene_SRP009615, "recount_brain_v2")

## Explore the metadata
colData(rse_gene)

## For a list of studies present in recount_brain check
## http://lieberinstitute.github.io/recount-brain/.
## recount_brain_v2 includes GTEx and TCGA brain samples in addition to the
## recount_brain_v1 data, plus ontology information.


## Obtain all the recount_brain_v2 metadata if you want to
## explore the metadata manually
recount_brain_v2 <- add_metadata(source = "recount_brain_v2")
```

`add_predictions` *Add predicted phenotypes to a recount rse object*

Description

Shannon Ellis et al (2017) predicted phenotypes based on expression data for the samples in the recount2 project. Using this function you can add the predictions to a RangedSummarizedExperiment-class object to the `colData()` slot.

Usage

```
add_predictions(rse, is_tcga = FALSE, version = "latest", verbose = TRUE)
```

Arguments

`rse`	A RangedSummarizedExperiment-class object as downloaded with download_study. If this argument is not specified, the function will return the full predictions table.
`is_tcga`	Set to `TRUE` only when `rse` is from TCGA. Otherwise set to `FALSE` (default).
`version`	The version number for the predicted phenotypes data. It has to match one of the available numbers at `https://github.com/leekgroup/recount-website/blob/master/predictions/`. Feel free to check if there is a newer version than the default. The version used is printed as part of the file name.
`verbose`	If `TRUE` it will print a message of where the predictions file is being downloaded to.

Details

If you use these predicted phenotypes please cite the Ellis et al bioRxiv pre-print available at https://www.biorxiv.org/content/early/2017/06/03/145656. See citation details with citation('recount').

Value

A RangedSummarizedExperiment-class object with the prediction columns appended to the `colData()` slot. The predicted phenotypes are:

sex male or female,

samplesource cell_line or tissue,

tissue tissue predicted based off of 30 tissues in GTEx,

sequencingstrategy single or paired end sequencing.

For each of the predicted phenotypes there are several columns as described next:

reported_phenotype `NA` when not available,

predicted_phenotype `NA` when we did not predict, "Unassigned" when prediction was ambiguous,

accuracy_phenotype accuracy is assigned per dataset based on comparison to samples for which we had reported phenotype information so there are three distinct values per predictor (GTEx, TCGA, SRA) across all studies.

Author(s)

Leonardo Collado-Torres

References

Ellis et al, bioRxiv, 2017. https://www.biorxiv.org/content/early/2017/06/03/145656

Examples

```
## Add the predictions to an example rse_gene object
rse_gene <- add_predictions(rse_gene_SRP009615)

## Explore the predictions
colData(rse_gene)

## Download all the latest predictions
PredictedPhenotypes <- add_predictions()
```

`all_metadata` *This function downloads the metadata for all projects.*

Description

Download the metadata from all the projects. This can be useful for finding samples of interests across all projects.

Usage

```
all_metadata(subset = "sra", verbose = TRUE)
```

Arguments

subset	Either sra, gtex or tcga. Specifies which metadata file to download.
verbose	If TRUE it will print a message of where the file is being downloaded to.

Details

Note that for subset = 'gtex', there are more variables than the ones we have for 'sra'. This information corresponds to file GTEx_Data_V6_Annotations_SampleAttributesDS.txt available at http://www.gtexportal.org/home/datasets. There you can find the information describing these variables.

For TCGA we acquired metadata information from 3 different sources:

- GDC: via a json query
- CGC: via json queries and a custom script to merge the tables
- TCGAbiolinks: we used to to parse GDC's XML files For more information, check https://github.com/leekgroup/recount-website/tree/master/metadata/tcga_prep.

Value

A DataFrame-class object with the phenotype metadata.

Author(s)

Leonardo Collado-Torres

Examples

```
metadata <- all_metadata()
```

---

browse_study *Open a SRA study id in the SRA website*

Description

Given a SRA study id get the url to browse the study using the SRA website.

Usage

```
browse_study(project, browse = interactive())
```

Arguments

project	A character vector with at least one SRA study id.
browse	Whether to open the resulting URL in the browser.

Value

Returns invisibly the URL for exploring the study in the SRA website.

Author(s)

Leonardo Collado-Torres

See Also

abstract_search

Examples

```
## Find the Geuvadis consortium project
id <- abstract_search("Geuvadis consortium", id_only = TRUE)
id

## Explore the Geuvadis consortium project
url <- browse_study(id)

## See the actual URL
url
```

---

coverage_matrix *Given a set of regions for a chromosome, compute the coverage matrix for a given SRA study.*

Description

Given a set of genomic regions as created by expressed_regions, this function computes the coverage matrix for a library size of 40 million 100 bp reads for a given SRA study.

Usage

```
coverage_matrix(
  project,
  chr,
  regions,
  chunksize = 1000,
  bpparam = NULL,
  outdir = NULL,
  chrlen = NULL,
  verbose = TRUE,
  verboseLoad = verbose,
  scale = TRUE,
  round = FALSE,
  ...
)
```

Arguments

project	A character vector with one SRA study id.
chr	A character vector with the name of the chromosome.
regions	A GRanges-class object with regions for chr for which to calculate the coverage matrix.
chunksize	A single integer vector defining the chunksize to use for computing the coverage matrix. Regions will be split into different chunks which can be useful when using a parallel instance as defined by bpparam.
bpparam	A BiocParallelParam-class instance which will be used to calculate the coverage matrix in parallel. By default, SerialParam-class will be used.
outdir	The destination directory for the downloaded file(s) that were previously downloaded with download_study. If the files are missing, but outdir is specified, they will get downloaded first. By default outdir is set to NULL which will use the data from the web. We only recommend downloading the full data if you will use it several times.
chrlen	The chromosome length in base pairs. If it's NULL, the chromosome length is extracted from the Rail-RNA runs GitHub repository. Alternatively check the SciServer section on the vignette to see how to access all the recount data via a R Jupyter Notebook.
verbose	If TRUE basic status updates will be printed along the way.
verboseLoad	If TRUE basic status updates for loading the data will be printed.
scale	If TRUE, the coverage counts will be scaled to read counts based on a library size of 40 million reads. Set scale to FALSE if you want the raw coverage counts. The scaling method is by AUC, as in the default option of scale_counts.
round	If TRUE, the counts are rounded to integers. Set to TRUE if you want to match the defaults of scale_counts.
...	Additional arguments passed to download_study when outdir is specified but the required files are missing.

Details

When using outdir = NULL the information will be accessed from the web on the fly. If you encounter internet access problems, it might be best to first download the BigWig files using download_study. This might be the best option if you are accessing all chromosomes for a given project and/or are thinking of using different sets of regions (for example, from different cutoffs applied to expressed_regions). Alternatively check the SciServer section on the vignette to see how to access all the recount data via a R Jupyter Notebook.

If you have bwtool installed, you can use https://github.com/LieberInstitute/recount.bwtool for faster results. Note that you will need to run scale_counts after running coverage_matrix_bwtool().

Value

A RangedSummarizedExperiment-class object with the counts stored in the assays slot.

Author(s)

Leonardo Collado-Torres

See Also

download_study, findRegions, railMatrix

Examples

```
if (.Platform$OS.type != "windows") {
    ## Reading BigWig files is not supported by rtracklayer on Windows
    ## Define expressed regions for study DRP002835, chrY
    regions <- expressed_regions("DRP002835", "chrY",
        cutoff = 5L,
        maxClusterGap = 3000L
    )

    ## Now calculate the coverage matrix for this study
    rse <- coverage_matrix("DRP002835", "chrY", regions)

    ## One row per region
    identical(length(regions), nrow(rse))
}
```

`download_retry` *Retry multiple times to download a file*

Description

This function is based on the Bioconductor guidelines for querying data from the web at http://bioconductor.org/developers/how-to/web-query/. It will run download a set of `N.TRIES` times before giving up. We implemented this function to reduce the number of Bioconductor build errors due to the occassional errors from our data hosting server.

Usage

```
download_retry(url, destfile = basename(url), mode = "wb", N.TRIES = 3L, ...)
```

Arguments

`url`	The URL to download. Passed to download.
`destfile`	The destination file. Defaults to the base name of the URL. Passed to download.
`mode`	Mode for writing the file. The default `wb` is used for binary files. This value is passed to download which passes it to download.file.
`N.TRIES`	The number of download attempts before giving up; default: 3. Should be an integer of length one with a value greater than 0.
`...`	Additional arguments passed to download.

Value

An invisible integer code as specified in download.file.

Examples

```
## Download the first files_info.tsv file (version 1)
download_retry(
    recount_url$url[which(recount_url$file_name == "files_info.tsv")[1]]
)
```

`download_study` *Download data for a given SRA study id from the recount project*

Description

Download the gene or exon level RangedSummarizedExperiment-class objects provided by the recount project. Alternatively download the counts, metadata or file information for a given SRA study id. You can also download the sample bigWig files or the mean coverage bigWig file.

Usage

```
download_study(
  project,
  type = "rse-gene",
  outdir = project,
  download = TRUE,
  version = 2,
  ...
)
```

Arguments

`project`	A character vector with one SRA study id.
`type`	Specifies which files to download. The options are:
	rse-gene the gene-level RangedSummarizedExperiment-class object in a file named rse_gene.Rdata.
	rse-exon the exon-level RangedSummarizedExperiment-class object in a file named rse_exon.Rdata.
	rse-jx the exon-exon junction level RangedSummarizedExperiment-class object in a file named rse_jx.Rdata.
	rse-tx the transcript level RangedSummarizedExperiment-class object in a file named rse_tx.RData.
	counts-gene the gene-level counts in a tsv file named counts_gene.tsv.gz.
	counts-exon the exon-level counts in a tsv file named counts_exon.tsv.gz.
	counts-jx the exon-exon junction level counts in a tsv file named counts_jx.tsv.gz.

phenotype the phenotype data for the study in a tsv file named `project`.tsv.

files-info the files information for the given study (including md5sum hashes) in a tsv file named files_info.tsv.

samples one bigWig file per sample in the study.

mean one mean bigWig file for the samples in the study, with each sample normalized to a 40 million 100 bp library using the total coverage sum (area under the coverage curve, AUC) for the given sample.

all Downloads all the above types. Note that it might take some time if the project has many samples. When using `type = 'all'` a small delay will be added before each download request to avoid request issues.

rse-fc Downloads the FANTOM-CAT/recount2 rse file described in Imada, Sanchez, et al., bioRxiv, 2019.

`outdir` The destination directory for the downloaded file(s). Alternatively check the `SciServer` section on the vignette to see how to access all the recount data via a R Jupyter Notebook.

`download` Whether to download the files or just get the download urls.

`version` A single integer specifying which version of the files to download. Valid options are 1 and 2, as described in https://jhubiostatistics.shinyapps.io/recount/ under the documentation tab. Briefly, version 1 are counts based on reduced exons while version 2 are based on disjoint exons. This argument mostly just matters for the exon counts. Defaults to version 2 (disjoint exons). Use `version = 1` for backward compatability with exon counts prior to version 1.5.3 of the package.

`...` Additional arguments passed to download.

Details

Check http://stackoverflow.com/a/34383991 if you need to find the effective URLs. For example, http://duffel.rail.bio/recount/DRP000366/bw/mean_DRP000366.bw points to a link from SciServer.

Transcript quantifications are described in Fu et al, bioRxiv, 2018. https://www.biorxiv.org/content/10.1101/247346v2

FANTOM-CAT/recount2 quantifications are described in Imada, Sanchez, et al., bioRxiv, 2019. https://www.biorxiv.org/content/10.1101/659490v1

Value

Returns invisibly the URL(s) for the files that were downloaded.

Author(s)

Leonardo Collado-Torres

Examples

```
## Find the URL to download the RangedSummarizedExperiment for the
## Geuvadis consortium study.
url <- download_study("ERP001942", download = FALSE)

## See the actual URL
url
## Not run:
## Download the example data included in the package for study SRP009615

url2 <- download_study("SRP009615")
url2

## Load the data
load(file.path("SRP009615", "rse_gene.Rdata"))

## Compare the data
library("testthat")
expect_equivalent(rse_gene, rse_gene_SRP009615)

## End(Not run)
```

`expressed_regions` *Identify expressed regions from the mean coverage for a given SRA project*

Description

This function uses the pre-computed mean coverage for a given SRA project to identify the expressed regions (ERs) for a given chromosome. It returns a GRanges-class object with the expressed regions as defined by findRegions.

Usage

```
expressed_regions(
  project,
  chr,
  cutoff,
  outdir = NULL,
  maxClusterGap = 300L,
  chrlen = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

project	A character vector with one SRA study id.
chr	A character vector with the name of the chromosome.
cutoff	The base-pair level cutoff to use.
outdir	The destination directory for the downloaded file(s) that were previously downloaded with download_study. If the files are missing, but outdir is specified, they will get downloaded first. By default outdir is set to NULL which will use the data from the web. We only recommend downloading the full data if you will use it several times.
maxClusterGap	This determines the maximum gap between candidate ERs.
chrlen	The chromosome length in base pairs. If it's NULL, the chromosome length is extracted from the Rail-RNA runs GitHub repository. Alternatively check the SciServer section on the vignette to see how to access all the recount data via a R Jupyter Notebook.
verbose	If TRUE basic status updates will be printed along the way.
...	Additional arguments passed to download_study when outdir is specified but the required files are missing.

Value

A GRanges-class object as created by findRegions.

Author(s)

Leonardo Collado-Torres

See Also

download_study, findRegions, railMatrix

Examples

```
## Define expressed regions for study SRP009615, chrY
if (.Platform$OS.type != "windows") {
    ## Reading BigWig files is not supported by rtracklayer on Windows
    regions <- expressed_regions("SRP009615", "chrY",
        cutoff = 5L,
        maxClusterGap = 3000L
    )
}
## Not run:
## Define the regions for multiple chrs
regs <- sapply(chrs, expressed_regions, project = "SRP009615", cutoff = 5L)

## You can then combine them into a single GRanges object if you want to
library("GenomicRanges")
single <- unlist(GRangesList(regs))
```

```
## End(Not run)
```

`find_geo` *Find the GEO accession id for a given SRA run*

Description

Given a SRA run id, this function will retrieve the GEO accession id (starting with GSM) if it's available. Otherwise it will return `NA`.

Usage

```
find_geo(run, verbose = FALSE, sleep = 1/2)
```

Arguments

`run`	A character vector of length 1 with the SRA run accession id.
`verbose`	Whether to print a message for the run. Useful when looping over a larger number of SRA run ids.
`sleep`	The number of seconds (or fraction) to wait before downloading data using getGEO. This is important if you are looking over `geo_info()` given the constraints published at `https://www.ncbi.nlm.nih.gov/books/NBK25497/`.

Details

Although the phenotype information already includes the GEO accession ids, not all projects had GEO entries at the time these tables were created. This function will then be useful to check if there is a GEO accession id for a given sample (run). If there is, you can then retrieve the information using geo_info.

Value

The GEO accession id for the corresponding sample.

Author(s)

Leonardo Collado-Torres

Examples

```
## Find the GEO accession id for for SRX110461
find_geo("SRX110461")
```

geo_characteristics *Build a data.frame from GEO's charactersitics for a given sample*

Description

This function builds a data.frame from the GEO characteristics extracted for a given sample. The names of the of columns correspond to the field names. For a given SRA project, this information can be combined for all samples as shown in the examples section.

Usage

```
geo_characteristics(pheno)
```

Arguments

pheno	A DataFrame-class as created by geo_info.

Value

A 1 row data.frame with the characteristic fields as column names and the values as the entries on the first row. If the authors of the study used the same names for all samples, you can then combine them using rbind.

Author(s)

Leonardo Collado-Torres

Examples

```
## Load required library
library("SummarizedExperiment")

## Get the GEO accession ids
# geoids <- sapply(colData(rse_gene_SRP009615)$run[1:2], find_geo)
## The previous example code works nearly all the time but it
## can occassionally fail depending on how rentrez is doing.
## This code makes sure that the example code runs.
geoids <- tryCatch(
    sapply(colData(rse_gene_SRP009615)$run[1:2], find_geo),
    error = function(e) {
        c(
            "SRR387777" = "GSM836270",
            "SRR387778" = "GSM836271"
        )
    }
)

## Get the data from GEO
geodata <- do.call(rbind, sapply(geoids, geo_info))
```

```
## Add characteristics in a way that we can access easily later on
geodata <- cbind(geodata, geo_characteristics(geodata))

## Explore the original characteristics and the result from
## geo_characteristics()
geodata[, c("characteristics", "cells", "shrna.expression", "treatment")]
```

`geo_info` *Extract information from GEO for a given sample*

Description

This function uses GEOquery to extract information for a given sample. The GEO accession ids for the sample can be found in the study phenotype table.

Usage

```
geo_info(
  geoid,
  verbose = FALSE,
  sleep = 1/2,
  getGPL = FALSE,
  destdir = tempdir(),
  ...
)
```

Arguments

`geoid`	A character vector of length 1 with the GEO accession id for a given sample.
`verbose`	If `TRUE` the `geoid` will be shown.
`sleep`	The number of seconds (or fraction) to wait before downloading data using getGEO. This is important if you are looking over `geo_info()` given the constraints published at `https://www.ncbi.nlm.nih.gov/books/NBK25497/`.
`getGPL`	This argument is passed to getGEO and is set to `FALSE` by default to speed up the process.
`destdir`	This argument is passed to getGEO.
`...`	Additional arguments passed to getGEO.

Value

Returns a DataFrame-class with the information from GEO available for the given sample.

Author(s)

Leonardo Collado-Torres, Andrew Jaffe

Examples

```
geo_info("GSM836270")
```

`getRPKM` *Compute an RPKM matrix based on a RangedSummarizedExperiment object*

Description

For some analyses you might be interested in transforming the counts into RPKMs which you can do with this function.

Usage

```
getRPKM(rse, length_var = "bp_length", mapped_var = NULL)
```

Arguments

`rse`	A RangedSummarizedExperiment-class object as downloaded with download_study.
`length_var`	A length 1 character vector with the column name from `rowData(rse)` that has the coding length. For gene level objects from recount this is `bp_length`. If `NULL`, then it will use `width(rowRanges(rse))` which should be used for exon RSEs.
`mapped_var`	A length 1 character vector with the column name from `colData(rse)` that has the number of reads mapped. For recount RSE object this would be `mapped_read_count`. If `NULL` (default) then it will use the column sums of the counts matrix. The results are different because not all mapped reads are mapped to exonic segments of the genome.

Details

For gene RSE objects, you will want to specify the `length_var` because otherwise you will be adjusting for the total gene length instead of the total exonic sequence length of the gene.

Value

A matrix with the RPKM values.

Author(s)

Andrew Jaffe, Leonardo Collado-Torres

See Also

scale_counts

Examples

```
## get RPKM matrix
rpkm <- getRPKM(rse_gene_SRP009615)

## You can also get an RPKM matrix after running scale_counts()
## with similar RPKM values
rpkm2 <- getRPKM(scale_counts(rse_gene_SRP009615))
rpkm3 <- getRPKM(scale_counts(rse_gene_SRP009615, by = "mapped_reads"))

summary(rpkm - rpkm2)
summary(rpkm - rpkm3)
summary(rpkm2 - rpkm3)
```

---

getTPM *Compute a TPM matrix based on a RangedSummarizedExperiment object*

Description

For some analyses you might be interested in transforming the counts into TPMs which you can do with this function. This function uses the gene-level RPKMs to derive TPM values (see Details).

Usage

```
getTPM(rse, length_var = "bp_length", mapped_var = NULL)
```

Arguments

`rse`	A RangedSummarizedExperiment-class object as downloaded with download_study.
`length_var`	A length 1 character vector with the column name from `rowData(rse)` that has the coding length. For gene level objects from recount this is `bp_length`. If `NULL`, then it will use `width(rowRanges(rse))` which should be used for exon RSEs.
`mapped_var`	A length 1 character vector with the column name from `colData(rse)` that has the number of reads mapped. For recount RSE object this would be `mapped_read_count`. If `NULL` (default) then it will use the column sums of the counts matrix. The results are different because not all mapped reads are mapped to exonic segments of the genome.

Details

For gene RSE objects, you will want to specify the `length_var` because otherwise you will be adjusting for the total gene length instead of the total exonic sequence length of the gene.

As noted in https://support.bioconductor.org/p/124265/, Sonali Arora et al computed TPMs in https://www.biorxiv.org/conten using the formula: TPM = FPKM / (sum of FPKM over all genes/transcripts) * 10^6

Arora et al mention in their code that the formula comes from https://doi.org/10.1093/bioinformatics/btp692; specifically `1.1.1 Comparison to RPKM estimation` where they mention an important assumption: Under the assumption of uniformly distributed reads, we note that RPKM measures are estimates of ...

There's also a blog post by Harold Pimentel explaining the relationship between FPKM and TPM: https://haroldpimentel.wordpress.com/2014/05/08/what-the-fpkm-a-review-rna-seq-expression-units/.

Value

A matrix with the TPM values.

Author(s)

Sonali Arora, Leonardo Collado-Torres

References

https://www.biorxiv.org/content/10.1101/445601v2 https://arxiv.org/abs/1104.3889

See Also

getRPKM

Examples

```
## Compute the TPM matrix from the raw gene-level base-pair counts.
tpm <- getTPM(rse_gene_SRP009615)
```

---

read_counts *Compute read counts*

Description

As described in the recount workflow, the counts provided by the recount2 project are base-pair counts. You can scale them using scale_counts or compute the read counts using the area under coverage information (AUC). We use the AUC because Rail-RNA soft clips some reads.

Usage

```
read_counts(rse, use_paired_end = TRUE, round = FALSE)
```

Arguments

`rse`	A RangedSummarizedExperiment-class object as downloaded with download_study.
`use_paired_end`	A logical vector. When `TRUE` it uses the paired-end flag (`colData(rse)$paired_end`) to determine whether the sample is paired-end or not. If it's paired-end, then it divides the counts by 2 to return paired-end read counts instead of fragment counts. When `FALSE`, this information is ignored.
`round`	Whether to round the counts to integers or not.

Details

Check the recount workflow for details about the counts provided by the recount2 project. Note that this function should not be used after scale_counts or it will return non-sensical results.

Value

Returns a RangedSummarizedExperiment-class object with the read counts.

Author(s)

Leonardo Collado-Torres

References

Collado-Torres L, Nellore A and Jaffe AE. recount workflow: Accessing over 70,000 human RNA-seq samples with Bioconductor version 1; referees: 1 approved, 2 approved with reservations. F1000Research 2017, 6:1558 doi: 10.12688/f1000research.12223.1.

See Also

scale_counts

Examples

```
## Difference between read counts and reads downloaded by Rail-RNA
colSums(assays(read_counts(rse_gene_SRP009615))$counts) / 1e6 -
    colData(rse_gene_SRP009615)$reads_downloaded / 1e6

## Paired-end read counts vs fragment counts (single-end counts)
download_study("DRP000499")
load("DRP000499/rse_gene.Rdata")
colSums(assays(read_counts(rse_gene, use_paired_end = FALSE))$counts) /
    colSums(assays(read_counts(rse_gene))$counts)

## Difference between paired-end read counts vs paired-end reads downloaded
colSums(assays(read_counts(rse_gene))$counts) / 1e6 -
    colData(rse_gene)$reads_downloaded / 1e6 / 2
```

---

recount_abstract *Summary information at the project level for the recount project*

Description

A data.frame with summary information at the project level for the studies analyzed in the recount project.

Format

A data.frame with 4 columns.

number_samples the number of samples in the study,

species the species of the study,

abstract the abstract text as provided by the SRAdb Bioconductor package,

project the SRA project id.

References

https://jhubiostatistics.shinyapps.io/recount/

See Also

download_study

`recount_exons` *Exon annotation used in recount*

Description

Exon annotation extracted from Gencode v25 (GRCh38.p7) used in recount. Data extraced on October 12th, 2017.

Format

A GRangesList-class with one element per gene. The names match the gene Gencode v25 ids.

References

https://jhubiostatistics.shinyapps.io/recount/

See Also

reproduce_ranges, recount_genes

recount_genes *Gene annotation used in recount*

Description

Gene annotation extracted from Gencode v25 (GRCh38.p7) used in recount. Data extraced on October 12th, 2017. The symbols were updated compared to the version from January 17th, 2017. It includes the sum of the width of the disjoint exons which can be used for normalizing the counts provided in the RangedSummarizedExperiment-class objects.

Format

A GRanges-class with one range per gene. The names match their Gencode v25 ids. The GRanges-class has three metadata columns.

gene_id the Gencode v25 ids, identical to the names.

bp_length the sum of the width of the disjoint exons for that given gene.

symbol a CharacterList with the corresponding gene symbols.

References

https://jhubiostatistics.shinyapps.io/recount/

See Also

reproduce_ranges, recount_exons

recount_url *Files and URLs hosted by the recount project*

Description

Files and URLs as provided by the recount project. This information is used internally in download_study.

Format

A data.frame with 6 columns.

path the original path to the file before being uploaded,

file_name the file name,

project the SRA project id,

version1 A logical vector indicating whether the file was part of version 1 (reduced exons).

version 2 A logical vector indicating whether the file was updated in version 2 (disjoint exons). Further details in the recount website documentation tab.

url the public URL for the given file.

References

https://jhubiostatistics.shinyapps.io/recount/

See Also

download_study

`reproduce_ranges` *Reproduce the gene or exons used in the RangedSummarizedExperiment objects*

Description

This function reproduces the gene or exon level information used for creating the RangedSummarizedExperiment-class objects provided by recount. The annotation is based on Gencode v25 with the gene-level information extracted with `genes()` (see transcripts with default arguments.

Usage

```
reproduce_ranges(level = "gene", db = "Gencode.v25")
```

Arguments

`level`	Either `genes` or `exon`. It specifies whether to return Gene or exon level information as a GRanges-class or GRangesList-class object respectively. The gene level information contains the width of the disjoint exons for that given gene which can be used to normalize the counts provided by recount. Can also be `both` in which case a 2 element list with the exon and the gene output is returned.
`db`	Either `Gencode.v25` (default) or `EnsDb.Hsapiens.v79`. The default option reproduces the annotation used when creating recount. EnsDb.Hsapiens.v79 can be used for an alternative annotation as showcased in the recount vignette.

Details

For Gencode.v25, we use the comprehensive gene annotation (regions: `CHR`) from https://www.gencodegenes.org/releases/25.html (GRCh38.p7).

Note that gene symbols have changed over time. This answer in the Bioconductor support forum details how to obtain the latest gene symbol mappings: https://support.bioconductor.org/p/126148/#126173.

Value

Either a GRanges-class object like recount_genes or a GRangesList-class object like recount_exons.

Author(s)

Leonardo Collado-Torres

See Also

recount_genes, recount_exons, https://github.com/nellore, https://jhubiostatistics.shinyapps.io/recount/

Examples

```
## Not run:
## Reproduce gene level information
genes <- reproduce_ranges()

## Compare against recount_genes
length(genes)
length(recount_genes)

## End(Not run)
```

---

rse_gene_SRP009615 *RangedSummarizedExperiment at the gene level for study SRP009615*

Description

RangedSummarizedExperiment-class at the gene level for study SRP009615. Used as an example in scale_counts. Matches the version2 file (details at https://jhubiostatistics.shinyapps.io/recount/) under the documentation tab.

Format

A RangedSummarizedExperiment-class as created by the recount project for study with SRA id (accession number) SRP009615.

References

https://jhubiostatistics.shinyapps.io/recount/

See Also

scale_counts, download_study

scale_counts *Scale the raw counts provided by the recount project*

Description

In preparation for a differential expression analysis, you will have to choose how to scale the raw counts provided by the recount project. Note that the raw counts are the sum of the base level coverage so you have to take into account the read length or simply the total coverage for the given sample (default option). You might want to do some further scaling to take into account the gene or exon lengths. If you prefer to calculate read counts without scaling check the function read_counts.

Usage

```
scale_counts(
  rse,
  by = "auc",
  targetSize = 4e+07,
  L = 100,
  factor_only = FALSE,
  round = TRUE
)
```

Arguments

rse	A RangedSummarizedExperiment-class object as downloaded with download_study.
by	Either auc or mapped_reads. If set to auc it will scale the counts by the total coverage of the sample. That is, the area under the curve (AUC) of the coverage. If set to mapped_reads it will scale the counts by the number of mapped reads, whether the library was paired-end or not, and the desired read length (L).
targetSize	The target library size in number of single end reads.
L	The target read length. Only used when by = 'mapped_reads' since it cancels out in the calculation when using by = 'auc'.
factor_only	Whether to only return the numeric scaling factor or to return a RangedSummarizedExperiment-class object with the counts scaled. If set to TRUE, you have to multiply the sample counts by this scaling factor.
round	Whether to round the counts to integers or not.

Details

Rail-RNA http://rail.bio uses soft clipping when aligning which is why we recommed using by = 'auc'.

If the reads are from a paired-end library, then the avg_read_length is the average fragment length. This is taken into account when using by = 'mapped_reads'.

Value

If `factor_only = TRUE` it returns a numeric vector with the scaling factor for each sample. If `factor_only = FALSE` it returns a RangedSummarizedExperiment-class object with the counts already scaled.

Author(s)

Leonardo Collado-Torres

See Also

download_study, read_counts

Examples

```
## Load an example rse_gene object
rse_gene <- rse_gene_SRP009615

## Scale counts
rse <- scale_counts(rse_gene)

## Find the project used as an example
project_info <- abstract_search("GSE32465")

## See some summary information for this project
project_info

## Use the following code to re-download this file
## Not run:
## Download
download_study(project_info$project)

## Load file
load(file.path(project_info$project, "rse_gene.Rdata"))
identical(rse_gene, rse_gene_SRP009615)

## End(Not run)
```

`snaptron_query` *Query Snaptron to get data from exon-exon junctions present in Intropolis*

Description

This function uses the Snaptron API to query specific exon-exon junctions that are available via Intropolis as described in the vignette.

Usage

```
snaptron_query(junctions, version = "srav1", verbose = TRUE, async = TRUE)
```

Arguments

junctions	A GRanges-class object with the exon-exon junctions of interest. The chromosome names should be in UCSC format, such as 'chr1'. The strand information is ignored in the query.
version	Either srav1, srav2, gtex or tcga. SRA Version 1 of Intropolis has the exon-exon junctions from about 20 thousand RNA-seq samples in hg19 coordinates. SRA Version 2 has the data from about 50 thousand RNA-seq samples aligned to hg38. GTEx has about 30 million junctions from about 10 thousand samples from the GTEx consortium on hg38 coordinates. Finally, TCGA has about 36 million junctions from about 11 thousand samples from the TCGA consortium on hg38 coordinates.
verbose	If TRUE status updates will be printed.
async	Defaults to TRUE but in some situations it might be preferrable to set it to FALSE. This argument gets passed to getURL. Check https://github.com/ChristopherWilks/snaptron/issues/11 for more details.

Value

A GRanges-class object with the results from the Snaptron query. For information on the different columns please see http://snaptron.cs.jhu.edu.

Author(s)

Leonardo Collado-Torres

References

Please cite http://snaptron.cs.jhu.edu if you use this function as Snaptron is a separate project from recount. Thank you!

Examples

```
library("GenomicRanges")
## Define some exon-exon junctions (hg19 coordinates)
junctions <- GRanges(seqnames = "chr2", IRanges(
    start = c(28971710:28971712, 29555081:29555083, 29754982:29754984),
    end = c(29462417:29462419, 29923338:29923340, 29917714:29917716)
))

## Check against Snaptron SRA version 1 (hg19 coordinates)
snaptron_query(junctions)
## Not run:
## Check another set of junctions against SRA version 2 (more data, hg38
## coordinates)
junctions_v2 <- GRanges(seqnames = "chr2", IRanges(
```

```
    start = 29532116:29532118, end = 29694848:29694850
))
snaptron_query(junctions_v2, version = "srav2")

## Check these junctions in GTEx and TCGA data
snaptron_query(junctions_v2, version = "gtex")
snaptron_query(junctions_v2, version = "tcga")

## End(Not run)
```

Index