Package: rsemverse (via r-universe)

November 15, 2026

Type Package

Title From RSEM Gene Results to Analysis-Ready Gene Tables

Version 0.2.0

Description Turns RSEM '*.genes.results' files into analysis-ready gene tables for human and mouse bulk RNA-seq. The files listed in a sample table are read into one table of expected counts or TPM with one row per gene symbol and one column per sample, using the Bioconductor annotation packages 'org.Hs.eg.db' and 'org.Mm.eg.db'. The same table is then filtered (the 'edgeR' filterByExpr rule, detection thresholds, most variable genes), reduced to built-in gene groups (protein-coding, mitochondrial, ribosomal, hemoglobin, sex-chromosome, spike-in), normalized between samples (CPM, TMM, upper quartile, DESeq2 median of ratios, quantile), transformed (log2, z-score, ranks), and returned as a long table with the sample information. No gene list or sample grouping is needed, and filtering and normalization give the same results as 'edgeR', 'DESeq2' and 'limma' without depending on them.

License MIT + file LICENSE

URL https://daikikumakura.github.io/rsemverse/, https://github.com/DaikiKumakura/rsemverse

BugReports https://github.com/DaikiKumakura/rsemverse/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

biocViews RNASeq, Transcriptomics

Depends R (>= 4.1.0)

Imports AnnotationDbi, data.table, stats, utils

Suggests org.Hs.eg.db, org.Mm.eg.db, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/pak/sysreqs libpng-dev libssl-dev zlib1g-dev

Repository https://daikikumakura.r-universe.dev

Date/Publication 2026-11-15 07:00:18 UTC

RemoteUrl https://github.com/DaikiKumakura/rsemverse

RemoteRef HEAD

RemoteSha 8b5077f32760c349c951f668df9b1f0c63f309de

Contents

rsem_filter 2
rsem_long 4
rsem_normalize 5
rsem_read 6
rsem_subset 8
rsem_transform 9

Index **11**

`rsem_filter` *Remove lowly expressed or uninformative genes*

Description

Keeps the genes that pass one of four common filters that need no sample groups. The table keeps its values and row order; only rows are removed. A message reports the rule and how many genes were kept.

Usage

```
rsem_filter(
  x,
  expr,
  method = NULL,
  min_value = NULL,
  min_samples = 0.2,
  n_top = 2000
)
```

Arguments

`x`	The gene by sample table, as returned by `rsem_read()`: gene names as row names (or in the first column) and one numeric column per sample.
`expr`	`"count"` or `"tpm"`: what the values are. Required.
`method`	`"edger"`, `"detected"`, `"nonzero"` or `"variable"`. The default is `"edger"` for counts and `"detected"` for TPM.

`min_value`	For `"edger"`, the count needed in a sample of median size (default 10). For `"detected"`, the value a gene must reach (default 1 for TPM, 10 for counts).
`min_samples`	For `"detected"`: in how many samples the gene must reach `min_value`. A number below 1 is read as a proportion of the samples (`0.5` = half). Default 0.2.
`n_top`	For `"variable"`: how many genes to keep. Default 2000.

Value

A data frame with the kept genes, in the same layout as `x` (genes as row names, samples as columns).

Methods

- `"edger"` (counts only; the default for counts): the rule of `edgeR::filterByExpr()` with all samples treated as one group. A gene is kept when its count per million (CPM) reaches `min_value / median library size * 1e6` in at least k samples and its total count is at least 15. k is the number of samples, or `10 + (n - 10) * 0.7` when there are more than 10. `min_value` is the count a gene needs in a sample of median size (default 10, the edgeR default).
- `"detected"` (the default for TPM): a gene is kept when its value is at least `min_value` in at least `min_samples` samples. The defaults are TPM >= 1 or count >= 10, in at least 20% of the samples (at least one sample). This is the form used for bulk RNA-seq (for example, GTEx keeps genes above a TPM and a count threshold in at least 20% of samples) and for single-cell data ("expressed in at least n cells").
- `"nonzero"`: removes genes that are zero in every sample.
- `"variable"`: keeps the `n_top` genes with the largest variance of `log2(value + 1)` across samples (default 2000), as used before PCA, clustering or heatmaps. Genes that are zero everywhere are never kept. Rows stay in their original order, not in the order of variance.

Filters can be chained, for example `"edger"` and then `"variable"`.

References

Chen Y, Lun ATL, Smyth GK (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. *F1000Research* 5:1438.

GTEx Consortium (2020). The GTEx Consortium atlas of genetic regulatory effects across human tissues. *Science* 369:1318-1330.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  counts <- rsem_read(file.path(ex, "data"), file.path(ex, "metadata.csv"),
                      species = "human", expr = "count")
  kept <- rsem_filter(counts, expr = "count")                # edgeR rule
  tpm <- rsem_read(file.path(ex, "data"), file.path(ex, "metadata.csv"),
                   species = "human", expr = "tpm")
  rsem_filter(tpm, expr = "tpm", min_value = 1, min_samples = 2)
  rsem_filter(tpm, expr = "tpm", method = "variable", n_top = 10)
}
```

`rsem_long` *Turn the table into long format with the sample information*

Description

Returns one row per gene and sample, with the columns `gene`, `sample_id` and `value`, followed by every column of the sample table. This is the layout that ggplot2 and dplyr work with: a group or a continuous variable in the sample table can go straight to colour, facets or the x axis.

Usage

```
rsem_long(x, metadata = NULL)
```

Arguments

`x`	The gene by sample table, as returned by `rsem_read()` or the other functions of this package.
`metadata`	The sample table: a path to a CSV file, or a data frame, with a `sample_id` column. Optional; without it only `gene`, `sample_id` and `value` are returned.

Value

A data frame with `nrow(x) * ncol(x)` rows.

Sample table

The sample table is the same `metadata.csv` that `rsem_read()` uses (or a data frame). It needs a `sample_id` column; every other column is added as it is, with no setting per column: columns of numbers stay numeric (for example age or dose) and other columns stay text (for example group or batch). `sample_id` is always read as text, so `001` stays `001`.

Every sample in `x` must be in the sample table. Rows of the sample table whose sample is not in `x` (for example samples removed earlier) are left out, and a message says so.

`sample_id` is a factor in the column order of `x`, so plots show the samples in that order. Each gene's rows are next to each other.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  meta <- file.path(ex, "metadata.csv")
  tpm <- rsem_read(file.path(ex, "data"), meta, species = "human", expr = "tpm")
  long <- rsem_long(rsem_transform(tpm, "log2"), metadata = meta)
  head(long)
  # ggplot2::ggplot(subset(long, gene == "TP53"),
  #                 ggplot2::aes(sample_id, value)) + ggplot2::geom_col()
}
```

`rsem_normalize` *Normalize between samples*

Description

Makes samples comparable by correcting for sequencing depth and, for some methods, RNA composition. The genes and samples stay the same; only the values change. A message reports the method and the per-sample factors.

Usage

```
rsem_normalize(x, expr, method)
```

Arguments

`x`	The gene by sample table, as returned by `rsem_read()`: gene names as row names (or in the first column) and one numeric column per sample.
`expr`	`"count"` or `"tpm"`: what the values are. Required.
`method`	For counts: `"cpm"`, `"tmm"`, `"upper_quartile"`, `"deseq2"` or `"quantile"`. For TPM: `"quantile"`. Required.

Value

A data frame of normalized values in the same layout as `x`. The per-sample factors are in the attribute `"factors"`: library size for `"cpm"`, effective library size (library size times the normalization factor) for `"tmm"` and `"upper_quartile"`, and size factors for `"deseq2"`.

Methods

For counts (`expr = "count"`):

- `"cpm"`: counts per million, `count / library size * 1e6`. Corrects for depth only.
- `"tmm"`: CPM with TMM normalization factors (trimmed mean of M values; Robinson and Oshlack 2010), as `edgeR::calcNormFactors()` then `edgeR::cpm()`. Also corrects for differences in RNA composition, for example when a few genes take up a large part of one sample.
- `"upper_quartile"`: CPM with upper-quartile factors (Bullard et al. 2010), as `edgeR::calcNormFactors(method = "upperquartile")`.
- `"deseq2"`: counts divided by the DESeq2 median-of-ratios size factors (Anders and Huber 2010), the values of `DESeq2::counts(dds, normalized = TRUE)`. Needs genes that are nonzero in every sample.
- `"quantile"`: quantile normalization, as `limma::normalizeQuantiles()`.

For TPM (`expr = "tpm"`), TPM already corrects for depth and gene length, so only `"quantile"` is offered. It makes every sample have the same distribution of values, which is a strong assumption: use it only when the samples are expected to differ in a few genes.

The results are the same as those of the packages named above (checked in the tests), computed here without them. Low-count genes affect TMM and the size factors, so filter first (`rsem_filter()`). Methods for differential expression (edgeR, DESeq2) need the raw counts, not these values. For a log scale, pass the result to `rsem_transform()`.

References

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 11:R25.

Anders S, Huber W (2010). Differential expression analysis for sequence count data. *Genome Biology* 11:R106.

Bullard JH, Purdom E, Hansen KD, Dudoit S (2010). Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics* 11:94.

Bolstad BM, Irizarry RA, Astrand M, Speed TP (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* 19:185-193.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  counts <- rsem_read(file.path(ex, "data"), file.path(ex, "metadata.csv"),
                      species = "human", expr = "count")
  tmm <- rsem_normalize(counts, expr = "count", method = "tmm")
  attr(tmm, "factors")
  rsem_transform(tmm, "log2")    # log2(TMM-CPM + 1)
}
```

`rsem_read` *Read RSEM gene-level results as a gene symbol by sample table*

Description

Reads the `*.genes.results` files listed in a metadata CSV and returns one expression value per gene symbol and sample, either the RSEM expected count or the TPM.

Usage

```
rsem_read(data = "data", metadata = "metadata.csv", species, expr)
```

Arguments

`data`	Directory that contains the RSEM files.
`metadata`	Path to the metadata CSV.
`species`	`"human"` or `"mouse"`. Required. The function stops if the gene IDs are of the other species.
`expr`	`"count"` (`expected_count`) or `"tpm"` (`TPM`). Required.

Value

A data frame with one row per gene symbol (as row names) and one column per sample, in the order of the metadata. A message reports how many genes received a symbol.

Input

`metadata` is a CSV with the columns `sample_id` and `ref`; other columns are ignored. `ref` is the name of the sample's RSEM file inside `data` (that is, the file read is `file.path(data, ref)`).

```
sample_id,ref
S1,S1.genes.results
S2,S2.genes.results
```

Every file needs the columns `gene_id` and `expected_count` or `TPM` (whichever `expr` selects); other columns are ignored. All files must contain the same genes in the same order, without duplicates.

Gene symbols

Symbols come from `org.Hs.eg.db` (human) or `org.Mm.eg.db` (mouse). The version suffix of an Ensembl gene ID (`ENSG00000141510.18`) is removed before the lookup, because the OrgDb keys have no version. The rules are:

- An Ensembl ID with several symbols gets the symbol of its lowest Entrez ID.
- Rows that end up with the same symbol are **summed** (counts and TPM are both additive).
- A gene without a symbol (not in the OrgDb, or not an Ensembl ID, such as a spike-in or an ID ending in `_PAR_Y`) keeps its `gene_id` from the file, so no expression is lost. If such an ID equals a gene symbol, the function stops instead of merging the two.

Rows appear in the order in which they first occur in the files. Values are not rounded and TPM is not renormalized.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  tpm <- rsem_read(
    data = file.path(ex, "data"),
    metadata = file.path(ex, "metadata.csv"),
    species = "human",
    expr = "tpm"
```

```
  )
  head(tpm)
}
```

`rsem_subset` *Keep or remove groups of genes*

Description

Selects genes by built-in gene groups, so that no gene list has to be prepared. Gene types and chromosomes come from the same annotation package as `rsem_read()` (`org.Hs.eg.db` or `org.Mm.eg.db`); the other groups are fixed symbol patterns. Values are not changed. A message reports how many genes each group matched.

Usage

```
rsem_subset(x, species, keep = NULL, remove = NULL)
```

Arguments

`x`	The gene by sample table, as returned by `rsem_read()`: gene names as row names (or in the first column) and one numeric column per sample.
`species`	`"human"` or `"mouse"`. Required.
`keep`, `remove`	Gene groups: one or more of `"protein_coding"`, `"mito"`, `"ribo"`, `"hb"`, `"sex"` and `"ercc"`. At least one of the two is required.

Value

A data frame with the selected genes, in the same layout and order as `x`.

Gene groups

- `"protein_coding"`: genes whose type is `protein-coding` in the annotation package (NCBI Gene). Many analyses use only these genes, for example to compare with proteomics, to reduce noise from lowly expressed non-coding RNAs and pseudogenes, or because a signature was built on them.
- `"mito"`: the 37 genes of the mitochondrial genome (chromosome MT; symbols `MT-` in human and `mt-` in mouse). The rRNA and tRNA genes have no Ensembl link in the OrgDb, so `rsem_read()` keeps them under their Ensembl IDs; a built-in list of the 37 Ensembl IDs (Ensembl release 116) finds them as well, together with the symbols the OrgDb gives to these IDs (the ID of `MT-TS1` maps to `MIR12136`, a microRNA inside it). A high share of these reads is a sign of damaged cells or degraded RNA.
- `"ribo"`: cytosolic ribosomal protein genes (`RPL*`, `RPS*`, `RPLP0-2`, `RPSA`; for mouse `Rpl*`, `Rps*`), protein-coding only. Mitochondrial ribosomal genes (`MRPL*`, `MRPS*`), kinases such as `RPS6KA1` and pseudogenes such as `RPL10P1` are not included. They are highly expressed and often removed before clustering.

- "hb": hemoglobin genes (HBA1, HBA2, HBB, HBD, HBE1, HBG1, HBG2, HBM, HBQ1, HBZ; for mouse Hba-*, Hbb-*, Hbq1*). protein-coding only. They signal blood contamination. HBEGF, hemoglobin pseudogenes and the locus control regions are not included.
- "sex": genes on chromosome Y and XIST. They separate samples by sex and are often removed when sex is not of interest. Pseudoautosomal genes (on both X and Y, expressed in both sexes) are not included.
- "ercc": ERCC spike-in controls (IDs starting with ERCC-).

keep is applied first (a gene is kept when it is in any of the listed groups), then remove. Genes are matched by symbol; rows that kept their Ensembl ID in rsem_read() are matched by that ID. Genes that the annotation package does not know are not in "protein_coding", "ribo", "hb" or "sex"; for example, chromosome Y genes that rsem_read() kept under their Ensembl ID are not found by "sex".

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  tpm <- rsem_read(file.path(ex, "data"), file.path(ex, "metadata.csv"),
                   species = "human", expr = "tpm")
  pc <- rsem_subset(tpm, "human", keep = "protein_coding")
  rsem_subset(tpm, "human", keep = "protein_coding", remove = c("mito", "ribo"))
}
```

---

rsem_transform *Transform values for plots, clustering and scores*

Description

Applies one or more transformations in the order given. The genes and samples stay the same. A message reports what was done.

Usage

```
rsem_transform(x, method, pseudocount = 1)
```

Arguments

x	The gene by sample table, as returned by rsem_read(), rsem_filter() or rsem_normalize().
method	One or more of "log2", "zscore", "center" and "rank", applied in this order. Required.
pseudocount	For "log2": the value added before the logarithm. Default 1.

Value

A data frame of transformed values in the same layout as x. The attribute "transform" lists the methods applied.

Methods

- `"log2"`: `log2(value + pseudocount)`, with `pseudocount = 1` by default. Compresses the range of TPM, CPM or normalized counts so that highly expressed genes do not dominate plots, distances and PCA. Needs values that are not negative.
- `"zscore"`: for each gene, subtracts its mean over samples and divides by its standard deviation, so every gene has mean 0 and standard deviation
    1. Used for heatmaps and for comparing genes on one scale. Apply it after `"log2"`. A gene with the same value in every sample gets 0. Needs at least two samples.
- `"center"`: subtracts each gene's mean over samples, without dividing by the standard deviation.
- `"rank"`: within each sample, the rank of each gene divided by the number of genes, from near 0 (lowest) to 1 (highest); tied values share their average rank. It does not depend on the scale of the values, so samples measured or normalized differently can be compared. Rank-based scores such as singscore start from this.

For example, `rsem_transform(tpm, c("log2", "zscore"))` gives the usual heatmap scale.

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly = TRUE)) {
  ex <- system.file("extdata", "example", package = "rsemverse")
  tpm <- rsem_read(file.path(ex, "data"), file.path(ex, "metadata.csv"),
                   species = "human", expr = "tpm")
  log_tpm <- rsem_transform(tpm, "log2")               # log2(TPM + 1)
  heat <- rsem_transform(tpm, c("log2", "zscore"))     # heatmap scale
  ranks <- rsem_transform(tpm, "rank")
}
```

Index

rsem_filter, 2
rsem_filter(), *6*, *9*
rsem_long, 4
rsem_normalize, 5
rsem_normalize(), *9*
rsem_read, 6
rsem_read(), *2*, *4*, *5*, *8*, *9*
rsem_subset, 8
rsem_transform, 9
rsem_transform(), *6*