Package 'tidybulk'

October 23, 2026

Type Package

Title Brings transcriptomics to the tidyverse

Version 2.2.0

Description This is a collection of utility functions that allow to perform exploration of and calculations to RNA sequencing data, in a modular, pipe-friendly and tidy fashion.

License GPL-3

Depends R (>= 4.4.0), ttservice (>= 0.3.6)

Imports tibble, dplyr (>= 1.1.0), magrittr, tidyr, stringr, rlang, purrr, tidyselect, stats, parallel, utils, lifecycle, scales, ggplot2, SummarizedExperiment, GenomicRanges, methods, S4Vectors, crayon, Matrix

Suggests BiocStyle, testthat, vctrs, AnnotationDbi, BiocManager, Rsubread, e1071, edgeR, limma, org.Hs.eg.db, org.Mm.eg.db, sva, GGally, knitr, qpdf, covr, Seurat, KernSmooth, Rtsne, widyr, clusterProfiler, msigdbr, DESeq2, broom, survival, boot, betareg, tidyHeatmap, pasilla, ggrepel, devtools, fastmatch, functional, survminer, tidySummarizedExperiment, markdown, uwot, matrixStats, preprocessCore, igraph, EGSEA, IRanges, here, glmmSeq, pbapply, pbmcapply, lme4, glmmTMB, MASS, pkgconfig, enrichplot, patchwork, airway

VignetteBuilder knitr

RdMacros lifecycle

Biarch true

biocViews AssayDomain, Infrastructure, RNASeq, DifferentialExpression, GeneExpression, Normalization, Clustering, QualityControl, Sequencing, Transcription, Transcriptomics

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

LazyDataCompression xz

URL https://github.com/stemangiola/tidybulk

BugReports https://github.com/stemangiola/tidybulk/issues

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

git_branch RELEASE_3_23

git_last_commit 454306a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-10-22

Author Stefano Mangiola [aut, cre],
Maria Doyle [ctb]

Maintainer Stefano Mangiola <mangiolastefano@gmail.com>

Contents

.rotate_dimensions_se . . . 3
adjust_abundance . . . 4
aggregate_duplicates . . . 7
as_matrix . . . 9
as_SummarizedExperiment . . . 10
check_if_counts_is_na . . . 11
check_if_duplicated_genes . . . 12
cluster_elements . . . 12
deconvolve_cellularity . . . 14
describe_transcript . . . 17
fill_missing_abundance . . . 18
get_bibliography . . . 19
get_X_cibersort . . . 20
identify_abundant . . . 21
impute_missing_abundance . . . 24
keep_abundant . . . 26
keep_abundant,RangedSummarizedExperiment-method . . . 28
keep_abundant,SummarizedExperiment-method . . . 29
keep_variable . . . 30
log10_reverse_trans . . . 31
logit_trans . . . 32
pivot_sample . . . 33
pivot_transcript . . . 34
quantile_normalise_abundance . . . 35
reduce_dimensions . . . 37
remove_redundancy . . . 40
resolve_complete_confounders_of_non_interest . . . 43
resolve_complete_confounders_of_non_interest,SummarizedExperiment-method . . . 45
rotate_dimensions . . . 45
scale_abundance . . . 48

scale_x_log10_reverse 50
scale_y_log10_reverse 51
test_differential_abundance 52
test_differential_expression 57
test_gene_enrichment 63
test_gene_overrepresentation 66
test_gene_rank 69
test_stratification_cellularity,SummarizedExperiment-method 71
tximeta_summarizeToGene_object 72
vignette_manuscript_signature_boxplot 73
vignette_manuscript_signature_tsne 73
vignette_manuscript_signature_tsne2 74
X_cibersort 74

Index **75**

`.rotate_dimensions_se` *Rotate two coordinate columns and append the rotated axes*

Description

This internal helper applies a planar rotation to two numeric columns that represent a low-dimensional embedding (for example PCA or UMAP coordinates) stored in either ‘colData()‘ or ‘rowData()‘ of a ‘SummarizedExperiment‘. It returns the original object with two additional columns containing the rotated values. The user specifies the rotation angle in degrees and may provide custom names for the new columns; otherwise sensible defaults are generated.

Usage

```
.rotate_dimensions_se(
  .data,
  dimension_1_column,
  dimension_2_column,
  rotation_degrees,
  .element = NULL,
  of_samples = TRUE,
  dimension_1_column_rotated = NULL,
  dimension_2_column_rotated = NULL
)
```

Arguments

`.data`	A ‘SummarizedExperiment‘ (or derivative) holding the coordinates to be rotated.
`dimension_1_column`	Symbol or bare column name for the first axis (e.g. ‘UMAP_1‘).
`dimension_2_column`	Symbol or bare column name for the second axis (e.g. ‘UMAP_2‘).

rotation_degrees
: Numeric scalar in the closed interval \([-360, 360]\) indicating the anti-clockwise rotation angle.

.element
: Optional quoted column holding sample or feature labels (unused, retained for compatibility).

of_samples
: Logical. If 'TRUE' (default) the function rotates columns in 'colData()'. If 'FALSE' it operates on 'rowData()'.

dimension_1_column_rotated
: Optional symbol to name the new first rotated coordinate column.

dimension_2_column_rotated
: Optional symbol to name the new second rotated coordinate column.

Value

The input 'SummarizedExperiment' with two extra metadata columns containing the rotated axes.

---

adjust_abundance *Adjust transcript abundance for unwanted variation*

Description

adjust_abundance() takes as input A 'tbl' (with at least three columns for sample, feature and transcript abundance) or 'SummarizedExperiment' (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with an additional adjusted abundance column. This method uses scaled counts if present.

Usage

```
adjust_abundance(
  .data,
  .formula = NULL,
  .factor_unwanted = NULL,
  .factor_of_interest = NULL,
  abundance = assayNames(.data)[1],
  .abundance = NULL,
  method = "combat_seq",
  ...,
  log_transform = NULL,
  transform = NULL,
  inverse_transform = NULL
)

## S4 method for signature 'SummarizedExperiment'
adjust_abundance(
  .data,
  .formula = NULL,
```

```
  .factor_unwanted = NULL,
  .factor_of_interest = NULL,
  abundance = assayNames(.data)[1],
  .abundance = NULL,
  method = "combat_seq",
  ...,
  log_transform = NULL,
  transform = NULL,
  inverse_transform = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
adjust_abundance(
  .data,
  .formula = NULL,
  .factor_unwanted = NULL,
  .factor_of_interest = NULL,
  abundance = assayNames(.data)[1],
  .abundance = NULL,
  method = "combat_seq",
  ...,
  log_transform = NULL,
  transform = NULL,
  inverse_transform = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.formula	DEPRECATED - A formula with no response variable, representing the desired linear model where the first covariate is the factor of interest and the second covariate is the unwanted variation (of the kind ~ factor_of_interest + batch)
.factor_unwanted	A tidy select, e.g. column names without double quotation. c(batch, country) These are the factor that we want to adjust for, including unwanted batcheffect, and unwanted biological effects.
.factor_of_interest	A tidy select, e.g. column names without double quotation. c(treatment) These are the factor that we want to preserve.
abundance	The name of the transcript/gene abundance column (character, preferred)
.abundance	DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)
method	A character string. Methods include combat_seq (default), combat and limma_remove_batch_effect.
...	Further parameters passed to the function sva::ComBat

log_transform DEPRECATED - A boolean, whether the value should be log-transformed (e.g., TRUE for RNA sequencing data)

transform DEPRECATED - A function that will tranform the counts, by default it is log1p for RNA sequencing data, but for avoinding tranformation you can use identity

inverse_transform DEPRECATED - A function that is the inverse of transform (e.g. expm1 is inverse of log1p). This is needed to tranform back the counts after analysis.

Details

‘r lifecycle::badge("maturing")‘

This function adjusts the abundance for (known) unwanted variation. At the moment just an unwanted covariate is allowed at a time using Combat (DOI: 10.1093/bioinformatics/bts034)

Underlying method: sva::ComBat(data, batch = my_batch, mod = design, prior.plots = FALSE, ...)

Value

A consistent object (to the input) with additional columns for the adjusted counts as ‘<COUNT COLUMN>_adjusted‘

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Zhang, Y., Parmigiani, G., & Johnson, W. E. (2020). ComBat-seq: batch effect adjustment for RNA-seq count data. NAR Genomics and Bioinformatics, 2(3), lqaa078. doi:10.1093/nargab/lqaa078

Johnson, W. E., Li, C., & Rabinovic, A. (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics, 8(1), 118–127. doi:10.1093/biostatistics/kxj037

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- as.factor(SummarizedExperiment::colData(airway)$dex)
```

```
cm = airway
# Create a balanced two-level batch within each condition to avoid confounding
cond <- SummarizedExperiment::colData(cm)$condition
cm$batch <- rep(NA_character_, ncol(cm))
for (lev in unique(cond)) {
  idx <- which(cond == lev)
  cm$batch[idx] <- rep(c('A','B'), length.out = length(idx))

}
cm$batch <- as.factor(cm$batch)

cm |>
identify_abundant() |>
adjust_abundance( .factor_unwanted = batch, .factor_of_interest =  condition, method="combat_seq" )
```

`aggregate_duplicates` *Aggregates multiple counts from the same samples (e.g., from isoforms), concatenates other character columns, and averages other numeric columns*

Description

aggregate_duplicates() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with aggregated transcripts that were duplicated.

Usage

```
aggregate_duplicates(
  .data,
  .transcript = NULL,
  feature = NULL,
  .abundance = NULL,
  aggregation_function = sum,
  keep_integer = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
aggregate_duplicates(
  .data,
  .transcript = NULL,
  feature = NULL,
  .abundance = NULL,
  aggregation_function = sum,
```

```
  keep_integer = TRUE,
  ...
)

## S4 method for signature 'RangedSummarizedExperiment'
aggregate_duplicates(
  .data,
  .transcript = NULL,
  feature = NULL,
  .abundance = NULL,
  aggregation_function = sum,
  keep_integer = TRUE,
  ...
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.transcript	DEPRECATED The name of the transcript/gene column (deprecated, use ‘feature‘ instead)
feature	The name of the feature column as a character string
.abundance	The name of the transcript/gene abundance column
aggregation_function	A function for counts aggregation (e.g., sum, median, or mean)
keep_integer	A boolean. Whether to force the aggregated counts to integer
...	Additional arguments passed to the aggregation function

Details

‘r lifecycle::badge("maturing")‘

This function aggregates duplicated transcripts (e.g., isoforms, ensembl). For example, we often have to convert ensembl symbols to gene/transcript symbol, but in doing so we have to deal with duplicates. ‘aggregate_duplicates‘ takes a tibble and column names (as symbols; for ‘sample‘, ‘transcript‘ and ‘count‘) as arguments and returns a tibble with aggregate transcript with the same name. All the rest of the column are appended, and factors and boolean are appended as characters.

Underlying custom method: data |> filter(n_aggr > 1) |> group_by(!!.sample,!!.transcript) |> dplyr::mutate(!!.abundance := !!.abundance |> aggregation_function())

Value

A consistent object (to the input) with aggregated transcript abundance and annotation

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M. T., & Carey, V. J. (2013). Software for computing and annotating genomic ranges. PLoS Computational Biology, 9(8), e1003118. doi:10.1371/journal.pcbi.1003118

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). dplyr: A Grammar of Data Manipulation. R package version 1.1.0. https://CRAN.R-project.org/package=dplyr

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



# Create a aggregation column
airway = airway
SummarizedExperiment::rowData(airway )$gene_name = rownames(airway )

   aggregate_duplicates(
     airway,
   feature = "gene_name"
   )
```

`as_matrix` *Get matrix from tibble*

Description

Get matrix from tibble

Usage

```
as_matrix(tbl, rownames = NULL, do_check = TRUE)
```

Arguments

`tbl`	A tibble
`rownames`	The column name of the input tibble that will become the rownames of the output matrix
`do_check`	A boolean

Value

A matrix

Examples

```
library(tibble)
tibble(.feature = "CD3G", count=1) |> as_matrix(rownames=.feature)
```

`as_SummarizedExperiment` *as_SummarizedExperiment*

Description

as_SummarizedExperiment() creates a 'SummarizedExperiment' object from a 'tbl' or 'tidybulk' tbl formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

Usage

```
as_SummarizedExperiment(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)

## S4 method for signature 'tbl_df'
as_SummarizedExperiment(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)
```

Arguments

`.data`	A tibble
`.sample`	The name of the sample column
`.transcript`	The name of the transcript/gene column
`.abundance`	The name of the transcript/gene abundance column

Value

A 'SummarizedExperiment' object

A 'SummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M. T., & Carey, V. J. (2013). Software for computing and annotating genomic ranges. PLoS Computational Biology, 9(8), e1003118. doi:10.1371/journal.pcbi.1003118

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). dplyr: A Grammar of Data Manipulation. R package version 1.1.0. https://CRAN.R-project.org/package=dplyr

Examples

```
# Convert tibble to SummarizedExperiment
library(tibble)
tibble(.sample = "A", .transcript = "CD3G", count = 1) |>
 as_SummarizedExperiment(.sample = .sample, .transcript = .transcript, .abundance = count)
```

---

check_if_counts_is_na *Check whether there are NA counts*

Description

Check whether there are NA counts

Usage

```
check_if_counts_is_na(.data, abundance, .abundance = NULL)
```

Arguments

.data	A tibble of read counts
abundance	A character name of the read count column
.abundance	A character name of the read count column (DEPRECATED)

Value

A tbl

check_if_duplicated_genes *Check whether there are duplicated genes/transcripts*

Description

Check whether there are duplicated genes/transcripts

Usage

```
check_if_duplicated_genes(
  .data,
  .sample = sample,
  .transcript = transcript,
  .abundance = `read count`
)
```

Arguments

.data	A tibble of read counts
.sample	A character name of the sample column
.transcript	A character name of the transcript/gene column
.abundance	A character name of the read count column

Value

A tbl

cluster_elements *Get clusters of elements (e.g., samples or transcripts)*

Description

cluster_elements() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and identify clusters in the data.

Usage

```
cluster_elements(.data, method, of_samples = TRUE, transform = log1p, ...)

## S4 method for signature 'SummarizedExperiment'
cluster_elements(.data, method, of_samples = TRUE, transform = log1p, ...)

## S4 method for signature 'RangedSummarizedExperiment'
cluster_elements(.data, method, of_samples = TRUE, transform = log1p, ...)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
method	A character string. The cluster algorithm to use, at the moment k-means is the only algorithm included.
of_samples	A boolean. In case the input is a tidybulk object, it indicates Whether the element column will be sample or transcript column
transform	A function that will tranform the counts, by default it is log1p for RNA sequencing data, but for avoinding tranformation you can use identity
...	Further parameters passed to the function kmeans

Details

‘r lifecycle::badge("maturing")‘

identifies clusters in the data, normally of samples. This function returns a tibble with additional columns for the cluster annotation. At the moment only k-means (DOI: 10.2307/2346830) and SNN clustering (DOI:10.1016/j.cell.2019.05.031) is supported, the plan is to introduce more clustering methods.

Underlying method for kmeans do.call(kmeans(.data, iter.max = 1000, ...)

Underlying method for SNN .data |> Seurat::CreateSeuratObject() |> Seurat::ScaleData(display.progress = TRUE,num.cores = 4, do.par = TRUE) |> Seurat::FindVariableFeatures(selection.method = "vst") |> Seurat::RunPCA(npcs = 30) |> Seurat::FindNeighbors() |> Seurat::FindClusters(method = "igraph", ...)

Value

A tbl object with additional columns with cluster labels

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1(14), 281-297. doi:10.1007/978-3-642-05177-7_26

Butler, A., Hoffman, P., Smibert, P., Papalexi, E., & Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature Biotechnology, 36(5), 411-420. doi:10.1038/nbt.4096

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



## Not run:
    cluster_elements(airway, centers = 2, method="kmeans")

## End(Not run)
```

`deconvolve_cellularity` *Get cell type proportions from samples*

Description

deconvolve_cellularity() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with the estimated cell type abundance for each sample

Usage

```
deconvolve_cellularity(
  .data,
  .abundance = NULL,
  reference = NULL,
  method = "cibersort",
  prefix = "",
  feature_column = NULL,
  ...
)

## S4 method for signature 'SummarizedExperiment'
deconvolve_cellularity(
  .data,
  .abundance = NULL,
  reference = NULL,
  method = "cibersort",
  prefix = "",
  feature_column = NULL,
  ...
```

```
)

## S4 method for signature 'RangedSummarizedExperiment'
deconvolve_cellularity(
  .data,
  .abundance = NULL,
  reference = NULL,
  method = "cibersort",
  prefix = "",
  feature_column = NULL,
  ...
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.abundance	The name of the transcript/gene abundance column
reference	A data frame. The methods cibersort and llsr can accept a custom rectangular dataframe with genes as rows names, cell types as column names and gene-transcript abundance as values. If NULL, the default reference for each algorithm will be used. For cibersort and llsr, the default is obtained via ‘get_X_cibersort()‘. For llsr will be LM22.
method	A character string. The method to be used. Available methods: "cibersort", "llsr", "epic", "mcp_counter", "quantiseq", "xcell". If a vector is provided, an error will be thrown. Default is all available methods.
prefix	A character string. The prefix you would like to add to the result columns. It is useful if you want to reshape data.
feature_column	A character string. The name of a column in rowData to use as feature names instead of rownames. If NULL (default), rownames are used.
...	Further parameters passed to the function Cibersort

Details

‘r lifecycle::badge("maturing")‘

This function infers the cell type composition of our samples (with the algorithm Cibersort; Newman et al., 10.1038/nmeth.3337).

Underlying method: CIBERSORT(Y = data, X = reference, ...)

Value

A consistent object (to the input) including additional columns for each cell type estimated

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Newman, A. M., Liu, C. L., Green, M. R., Gentles, A. J., Feng, W., Xu, Y., Hoang, C. D., Diehn, M., & Alizadeh, A. A. (2015). Robust enumeration of cell subsets from tissue expression profiles. Nature Methods, 12(5), 453-457. doi:10.1038/nmeth.3337

Racle, J., de Jonge, K., Baumgaertner, P., Speiser, D. E., & Gfeller, D. (2017). Simultaneous enumeration of cancer and immune cell types from bulk tumor gene expression data. eLife, 6, e26476. doi:10.7554/eLife.26476

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




# Map ENSEMBL rownames to SYMBOLs for compatibility with reference signatures

library(tidySummarizedExperiment)
library(org.Hs.eg.db)

## Not run:
airway |>
mutate(gene_symbol = AnnotationDbi::mapIds(
  org.Hs.eg.db::org.Hs.eg.db,
  keys = .feature,
  keytype = 'ENSEMBL',
  column = 'SYMBOL',
  multiVals = 'first'
)) |>
mutate(gene_symbol = ifelse(is.na(gene_symbol) | gene_symbol == '', .feature, gene_symbol)) |>
deconvolve_cellularity(feature_column = 'gene_symbol', cores = 1)

## End(Not run)

# Alternatively, if you already have a feature column in rowData
# se_with_features <- airway
# rowData(se_with_features)$gene_symbol <- rownames(se_with_features)
# se_with_features |> deconvolve_cellularity(feature_column = 'gene_symbol', cores = 1)

# Using a custom reference matrix
# custom_ref <- get_X_cibersort()  # Get the default Cibersort reference
# se_with_features |> deconvolve_cellularity(reference = custom_ref, feature_column = 'gene_symbol', cores = 1)
```

describe_transcript *Get DESCRIPTION from gene SYMBOL for Human and Mouse*

Description

Get DESCRIPTION from gene SYMBOL for Human and Mouse

describe_transcript

describe_transcript

describe_transcript

Usage

```
describe_transcript(.data)

.describe_transcript_SE(.data)

## S4 method for signature 'SummarizedExperiment'
describe_transcript(.data)

## S4 method for signature 'RangedSummarizedExperiment'
describe_transcript(.data)
```

Arguments

.data	A tt or tbl object.

Value

A tbl

A 'SummarizedExperiment' object

A consistent object (to the input) including additional columns for transcript symbol

A consistent object (to the input) including additional columns for transcript symbol

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Carlson, M. (2019). org.Hs.eg.db: Genome wide annotation for Human. R package version 3.8.2.

Carlson, M. (2019). org.Mm.eg.db: Genome wide annotation for Mouse. R package version 3.8.2.

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



describe_transcript(airway)
```

`fill_missing_abundance` *Fill transcript abundance if missing from sample-transcript pairs*

Description

fill_missing_abundance() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with new observations

Usage

```
fill_missing_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  fill_with
)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.sample`	The name of the sample column
`.transcript`	The name of the transcript column
`.abundance`	The name of the transcript abundance column
`fill_with`	A numerical abundance with which fill the missing data points

Details

[Questioning]

This function fills the abundance of missing sample-transcript pair using the median of the sample group defined by the formula

Value

A consistent object (to the input) non-sparse abundance

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



print("Not run for build time.")

# airway |>  fill_missing_abundance( fill_with = 0)
```

---

get_bibliography *Produces the bibliography list of your workflow*

Description

get_bibliography() takes as input a ‘tidybulk‘

Usage

```
get_bibliography(.data)

## S4 method for signature 'SummarizedExperiment'
get_bibliography(.data)

## S4 method for signature 'RangedSummarizedExperiment'
get_bibliography(.data)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))

Details

‘r lifecycle::badge("maturing")‘

This methods returns the bibliography list of your workflow from the metadata of a tidybulk object (metadata(.)$tidybulk$methods_used) or from the internals for backward compatibility (attr(., "internals"))

Value

NULL. It prints a list of bibliography references for the software used through the workflow.

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




get_bibliography(airway)
```

`get_X_cibersort` *Get Cibersort reference data*

Description

This function loads and returns the X_cibersort reference matrix used for cell type deconvolution with the Cibersort and LLSR methods. The reference matrix contains gene expression signatures for 22 immune cell types.

Usage

```
get_X_cibersort()
```

Value

The X_cibersort reference matrix with genes as rows and cell types as columns

References

Newman, A. M., Liu, C. L., Green, M. R., Gentles, A. J., Feng, W., Xu, Y., Hoang, C. D., Diehn, M., & Alizadeh, A. A. (2015). Robust enumeration of cell subsets from tissue expression profiles. Nature Methods, 12(5), 453-457. doi:10.1038/nmeth.3337

Examples

```
# Get the default Cibersort reference matrix
ref_matrix <- get_X_cibersort()

# Use with deconvolve_cellularity
# se |> deconvolve_cellularity(reference = get_X_cibersort(), method = "cibersort")
```

---

identify_abundant *Identify abundant transcripts/genes*

Description

Identifies transcripts/genes that are consistently expressed above a threshold across samples. This function adds a logical column '.abundant' to indicate which features pass the filtering criteria.

Usage

```
identify_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
identify_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
```

```
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
identify_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ or ‘SummarizedExperiment‘ object containing transcript/gene abundance data
abundance	The name of the transcript/gene abundance column (character, preferred)
design	A design matrix for more complex experimental designs. If provided, this is passed to filterByExpr instead of factor_of_interest.
formula_design	...
minimum_counts	...
minimum_proportion	...
minimum_count_per_million	Minimum CPM cutoff to use for filtering (passed to CPM.Cutoff in filterByExpr). If provided, this will override the minimum_counts parameter. Default is NULL (uses edgeR default).
factor_of_interest	The name of the column containing groups/conditions for filtering. Used by edgeR’s filterByExpr to define sample groups. **DEPRECATED:** Use ’design’ or ’formula_design’ instead. This argument will be removed in a future release.
...	Further arguments.
.abundance	DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

‘r lifecycle::badge("maturing")‘

This function uses edgeR’s filterByExpr() function to identify consistently expressed features. A feature is considered abundant if it has CPM > minimum_counts in at least minimum_proportion of samples in at least one experimental group (defined by factor_of_interest or design).

Value

Returns the input object with an additional logical column ‘.abundant‘ indicating which features passed the abundance threshold criteria.

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

McCarthy, D. J., Chen, Y., & Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Research, 40(10), 4288-4297. DOI: 10.1093/bioinformatics/btp616

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex


# Basic usage
airway |> identify_abundant()

# With custom thresholds
airway |> identify_abundant(
  minimum_counts = 5,
  minimum_proportion = 0.5
)

# Using a factor of interest
airway |> identify_abundant(factor_of_interest = condition)
```

impute_missing_abundance *impute transcript abundance if missing from sample-transcript pairs*

Description

impute_missing_abundance() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with additional sample-transcript pairs with imputed transcript abundance.

Usage

```
impute_missing_abundance(
  .data,
  .formula,
  suffix = "",
  force_scaling = FALSE,
  ...,
  abundance = assayNames(.data)[1],
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
impute_missing_abundance(
  .data,
  .formula,
  suffix = "",
  force_scaling = FALSE,
  ...,
  abundance = assayNames(.data)[1],
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
impute_missing_abundance(
  .data,
  .formula,
  suffix = "",
  force_scaling = FALSE,
  ...,
  abundance = assayNames(.data)[1],
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.formula	A formula with no response variable, representing the desired linear model where the first covariate is the factor of interest and the second covariate is the unwanted variation (of the kind ~ factor_of_interest + batch)
suffix	A character string. This is added to the imputed count column names. If empty the count column are overwritten
force_scaling	A boolean. In case a abundance-containing column is not scaled (columns with _scale suffix), setting force_scaling = TRUE will result in a scaling by library size, to compensating for a possible difference in sequencing depth.
...	Further arguments.
abundance	The name of the transcript/gene abundance column (character, preferred)
.abundance	DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

‘r lifecycle::badge("maturing")‘

This function imputes the abundance of missing sample-transcript pair using the median of the sample group defined by the formula

Value

A consistent object (to the input) non-sparse abundance

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex


library(airway)
data(airway)
```

```
airway <- airway[1:100, 1:5]

airway |>
  impute_missing_abundance(.formula = ~ dex)
```

---

keep_abundant *Filter to keep only abundant transcripts/genes*

Description

Filters the data to keep only transcripts/genes that are consistently expressed above a threshold across samples. This is a filtering version of identify_abundant() that removes low-abundance features instead of just marking them.

This function is similar to identify_abundant() but instead of adding an .abundant column, it filters out the low-abundance features directly.

Usage

```
keep_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ or ‘SummarizedExperiment‘ object containing transcript/gene abundance data
abundance	The name of the transcript/gene abundance column (character, preferred)
design	A design matrix for more complex experimental designs. If provided, this is passed to filterByExpr instead of factor_of_interest.
formula_design	A formula for creating the design matrix
minimum_counts	The minimum count threshold for a feature to be considered abundant
minimum_proportion	The minimum proportion of samples in which a feature must be abundant
minimum_count_per_million	The minimum count per million threshold

factor_of_interest
: The name of the column containing groups/conditions for filtering. DEPRECATED: Use 'design' or 'formula_design' instead.

...
: Further arguments.

.abundance
: DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

Filter to keep only abundant transcripts/genes

[Questioning]

This function uses edgeR's filterByExpr() function to identify and keep consistently expressed features. A feature is kept if it has CPM > minimum_counts in at least minimum_proportion of samples in at least one experimental group (defined by factor_of_interest or design).

This function is similar to identify_abundant() but instead of adding an .abundant column, it filters out the low-abundance features directly.

Value

Returns a filtered version of the input object containing only the features that passed the abundance threshold criteria.

Returns a filtered version of the input object containing only the features that passed the abundance threshold criteria.

References

McCarthy, D. J., Chen, Y., & Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Research, 40(10), 4288-4297. DOI: 10.1093/bioinformatics/btp616

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex


# Basic usage
airway |> keep_abundant()

# With custom thresholds
airway |> keep_abundant(
  minimum_counts = 5,
  minimum_proportion = 0.5
)
```

```
# Using a factor of interest
airway |> keep_abundant(factor_of_interest = condition)
```

---

keep_abundant,RangedSummarizedExperiment-method
keep_abundant

Description

keep_abundant

Usage

```
## S4 method for signature 'RangedSummarizedExperiment'
keep_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)
```

Arguments

`.data`	A ‘tbl‘ or ‘SummarizedExperiment‘ object containing transcript/gene abundance data
`abundance, .abundance`	The name of the transcript/gene abundance column (character, preferred)
`design`	A design matrix for more complex experimental designs. If provided, this is passed to filterByExpr instead of factor_of_interest.
`formula_design`	A formula for creating the design matrix
`minimum_counts`	The minimum count threshold for a feature to be considered abundant
`minimum_proportion`	The minimum proportion of samples in which a feature must be abundant
`minimum_count_per_million`	The minimum count per million threshold
`factor_of_interest`	The name of the column containing groups/conditions for filtering. DEPRECATED: Use ’design’ or ’formula_design’ instead.
`...`	Further arguments.

Value

A 'SummarizedExperiment' object

`keep_abundant,SummarizedExperiment-method`
keep_abundant

Description

keep_abundant

Usage

```
## S4 method for signature 'SummarizedExperiment'
keep_abundant(
  .data,
  abundance = assayNames(.data)[1],
  design = NULL,
  formula_design = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  minimum_count_per_million = NULL,
  factor_of_interest = NULL,
  ...,
  .abundance = NULL
)
```

Arguments

`.data`	A 'tbl' or 'SummarizedExperiment' object containing transcript/gene abundance data
`abundance, .abundance`	The name of the transcript/gene abundance column (character, preferred)
`design`	A design matrix for more complex experimental designs. If provided, this is passed to filterByExpr instead of factor_of_interest.
`formula_design`	A formula for creating the design matrix
`minimum_counts`	The minimum count threshold for a feature to be considered abundant
`minimum_proportion`	The minimum proportion of samples in which a feature must be abundant
`minimum_count_per_million`	The minimum count per million threshold
`factor_of_interest`	The name of the column containing groups/conditions for filtering. DEPRECATED: Use 'design' or 'formula_design' instead.
`...`	Further arguments.

Value

A ‘SummarizedExperiment‘ object

`keep_variable` *Keep variable transcripts*

Description

keep_variable() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with additional columns for the statistics from the hypothesis test.

Usage

```
keep_variable(
  .data,
  .abundance = NULL,
  top = 500,
  transform = log1p,
  log_transform = TRUE
)

## S4 method for signature 'SummarizedExperiment'
keep_variable(.data, top = 500, transform = log1p)

## S4 method for signature 'RangedSummarizedExperiment'
keep_variable(.data, top = 500, transform = log1p)
```

Arguments

`.data`	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
`.abundance`	The name of the transcript/gene abundance column
`top`	Integer. Number of top transcript to consider
`transform`	A function that will tranform the counts, by default it is log1p for RNA sequencing data, but for avoinding tranformation you can use identity
`log_transform`	DEPRECATED. Use transform instead.

Details

‘r lifecycle::badge("maturing")‘

At the moment this function uses edgeR https://doi.org/10.1093/bioinformatics/btp616

Value

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

Underlying method: s <- rowMeans((x - rowMeans(x)) ^ 2) o <- order(s, decreasing = TRUE) x <- x[o[1L:top], , drop = FALSE] variable_trancripts = rownames(x)

A 'SummarizedExperiment' object

A 'SummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




 keep_variable(airway, top = 500)
```

`log10_reverse_trans` *Log10 reverse transformation for ggplot2*

Description

Creates a transformation that applies -log10(x) to data, useful for visualizing p-values or other values where smaller values should be displayed larger.

Usage

```
log10_reverse_trans()
```

Value

A transformation object that can be used with ggplot2's scale functions

Examples

```
## Not run:
library(ggplot2)
# Example usage with p-values
ggplot(data, aes(x = pvalue)) +
  geom_histogram() +
  scale_x_continuous(trans = log10_reverse_trans())

## End(Not run)
```

`logit_trans` *logit scale*

Description

it perform logit scaling with right axis formatting. To not be used directly but with ggplot (e.g. scale_y_continuous(trans = "log10_reverse"))

Usage

```
logit_trans()
```

Details

‘r lifecycle::badge("maturing")‘

Value

A scales object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. https://ggplot2.tidyverse.org

Examples

```
library(ggplot2)
library(tibble)

tibble(pvalue = c(0.001, 0.05, 0.1), fold_change = 1:3) |>
 ggplot(aes(fold_change , pvalue)) +
 geom_point() +
 scale_y_continuous(trans = "log10_reverse")
```

pivot_sample *Extract sample-wise information*

Description

pivot_sample() takes as input a ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a ‘tbl‘ with only sample-related columns

Usage

```
pivot_sample(.data)

## S4 method for signature 'SummarizedExperiment'
pivot_sample(.data)

## S4 method for signature 'RangedSummarizedExperiment'
pivot_sample(.data)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))

Details

‘r lifecycle::badge("maturing")‘

This functon extracts only sample-related information for downstream analysis (e.g., visualisation). It is disruptive in the sense that it cannot be passed anymore to tidybulk function.

Value

A ‘tbl‘ with transcript-related information

A consistent object (to the input)

A consistent object (to the input)

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex
```

```
library(airway)
data(airway)
airway <- airway[1:100, 1:5]

 pivot_sample(airway )
```

`pivot_transcript` *Extract transcript-wise information*

Description

pivot_transcript() takes as input a ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a ‘tbl‘ with only transcript-related columns

Usage

```
pivot_transcript(.data)

## S4 method for signature 'SummarizedExperiment'
pivot_transcript(.data)

## S4 method for signature 'RangedSummarizedExperiment'
pivot_transcript(.data)
```

Arguments

`.data`	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))

Details

‘r lifecycle::badge("maturing")‘

This functon extracts only transcript-related information for downstream analysis (e.g., visualisation). It is disruptive in the sense that it cannot be passed anymore to tidybulk function.

Value

A ‘tbl‘ with transcript-related information

A consistent object (to the input)

A consistent object (to the input)

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex


library(airway)
data(airway)
airway <- airway[1:100, 1:5]

 pivot_transcript(airway  )
```

---

quantile_normalise_abundance

Normalise by quantiles the counts of transcripts/genes

Description

quantile_normalise_abundance() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and Scales transcript abundance compansating for sequencing depth (e.g., with TMM algorithm, Robinson and Oshlack doi.org/10.1186/gb-2010-11-3-r25).

Usage

```
quantile_normalise_abundance(
  .data,
  .abundance = NULL,
  method = "limma_normalize_quantiles",
  target_distribution = NULL
)

## S4 method for signature 'SummarizedExperiment'
quantile_normalise_abundance(
  .data,
  .abundance = NULL,
  method = "limma_normalize_quantiles",
```

```
  target_distribution = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
quantile_normalise_abundance(
  .data,
  .abundance = NULL,
  method = "limma_normalize_quantiles",
  target_distribution = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.abundance	The name of the transcript/gene abundance column
method	A character string. Either "limma_normalize_quantiles" for limma::normalizeQuantiles or "preprocesscore_normalize_quantiles_use_target" for preprocessCore::normalize.quantiles.use.target for large-scale datasets.
target_distribution	A numeric vector. If NULL the target distribution will be calculated by preprocessCore. This argument only affects the "preprocesscore_normalize_quantiles_use_target" method.

Details

‘r lifecycle::badge("maturing")‘

Tranform the feature abundance across samples so to have the same quantile distribution (using preprocessCore).

Underlying method

If ‘limma_normalize_quantiles‘ is chosen

.data |>limma::normalizeQuantiles()

If ‘preprocesscore_normalize_quantiles_use_target‘ is chosen

.data |> preprocessCore::normalize.quantiles.use.target(target = preprocessCore::normalize.quantiles.determine.target(.data))

Value

A tbl object with additional columns with scaled data as ‘<NAME OF COUNT COLUMN>_scaled‘

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




 airway |>
   quantile_normalise_abundance()
```

`reduce_dimensions` *Dimension reduction of the transcript abundance data*

Description

reduce_dimensions() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and calculates the reduced dimensional space of the transcript abundance.

Usage

```
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  transform = log1p,
  scale = TRUE,
```

```
  ...,
  log_transform = NULL
)

## S4 method for signature 'SummarizedExperiment'
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  transform = log1p,
  scale = TRUE,
  ...,
  log_transform = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  transform = log1p,
  scale = TRUE,
  ...,
  log_transform = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.element	The name of the element column (normally samples).
.feature	The name of the feature column (normally transcripts/genes)
.abundance	The name of the column including the numerical value the clustering is based on (normally transcript abundance)
method	A character string. The dimension reduction algorithm to use (PCA, MDS, tSNE).

.dims	An integer. The number of dimensions your are interested in (e.g., 4 for returning the first four principal components).
top	An integer. How many top genes to select for dimensionality reduction
of_samples	A boolean. In case the input is a tidybulk object, it indicates Whether the element column will be sample or transcript column
transform	A function that will tranform the counts, by default it is log1p for RNA sequencing data, but for avoinding tranformation you can use identity
scale	A boolean for method="PCA", this will be passed to the ‘prcomp‘ function. It is not included in the ... argument because although the default for ‘prcomp‘ if FALSE, it is advisable to set it as TRUE.
...	Further parameters passed to the function prcomp if you choose method="PCA" or Rtsne if you choose method="tSNE", or uwot::tumap if you choose method="umap"
log_transform	DEPRECATED - A boolean, whether the value should be log-transformed (e.g., TRUE for RNA sequencing data)

Details

‘r lifecycle::badge("maturing")‘

This function reduces the dimensions of the transcript abundances. It can use multi-dimensional scaling (MDS; DOI.org/10.1186/gb-2010-11-3-r25), principal component analysis (PCA), or tSNE (Jesse Krijthe et al. 2018)

Underlying method for PCA: prcomp(scale = scale, ...)

Underlying method for MDS: limma::plotMDS(ndim = .dims, plot = FALSE, top = top)

Underlying method for tSNE: Rtsne::Rtsne(data, ...)

Underlying method for UMAP:

df_source = .data |>

Filter NA symbol filter(!!.feature |> is.na() |> not()) |>

Prepare data frame distinct(!!.feature,!!.element,!!.abundance) |>

Filter most variable genes keep_variable_transcripts(top) |> reduce_dimensions(method="PCA", .dims = calculate_for_pca_dimensions) |> as_matrix(rownames = quo_name(.element)) |> uwot::tumap(...)

Value

A tbl object with additional columns for the reduced dimensions

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Krijthe, J. H. (2015). Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation. R package version 0.15.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv preprint arXiv:1802.03426.

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex
```

```
counts.MDS =
 airway |>
 identify_abundant() |>
 reduce_dimensions( method="MDS", .dims = 3)
```

```
counts.PCA =
 airway |>
 identify_abundant() |>
 reduce_dimensions(method="PCA", .dims = 3)
```

---

remove_redundancy *Drop redundant elements (e.g., samples) for which feature (e.g., transcript/gene) abundances are correlated*

Description

remove_redundancy() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) for correlation method or | <DIMENSION 1> | <DIMENSION 2> | <...> | for reduced_dimensions method, and returns a consistent object (to the input) with dropped elements (e.g., samples).

Usage

```
remove_redundancy(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
```

```
  correlation_threshold = 0.9,
  top = Inf,
  transform = identity,
  Dim_a_column,
  Dim_b_column,
  log_transform = NULL
)

## S4 method for signature 'SummarizedExperiment'
remove_redundancy(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  correlation_threshold = 0.9,
  top = Inf,
  transform = identity,
  Dim_a_column = NULL,
  Dim_b_column = NULL,
  log_transform = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
remove_redundancy(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  correlation_threshold = 0.9,
  top = Inf,
  transform = identity,
  Dim_a_column = NULL,
  Dim_b_column = NULL,
  log_transform = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.element	The name of the element column (normally samples).
.feature	The name of the feature column (normally transcripts/genes)

.abundance	The name of the column including the numerical value the clustering is based on (normally transcript abundance)
method	A character string. The method to use, correlation and reduced_dimensions are available. The latter eliminates one of the most proximar pairs of samples in PCA reduced dimensions.
of_samples	A boolean. In case the input is a tidybulk object, it indicates Whether the element column will be sample or transcript column
correlation_threshold	A real number between 0 and 1. For correlation based calculation.
top	An integer. How many top genes to select for correlation based method
transform	A function that will tranform the counts, by default it is log1p for RNA sequencing data, but for avoinding tranformation you can use identity
Dim_a_column	A character string. For reduced_dimension based calculation. The column of one principal component
Dim_b_column	A character string. For reduced_dimension based calculation. The column of another principal component
log_transform	DEPRECATED - A boolean, whether the value should be log-transformed (e.g., TRUE for RNA sequencing data)

Details

‘r lifecycle::badge("maturing")‘

This function removes redundant elements from the original data set (e.g., samples or transcripts). For example, if we want to define cell-type specific signatures with low sample redundancy. This function returns a tibble with dropped redundant elements (e.g., samples). Two redundancy estimation approaches are supported: (i) removal of highly correlated clusters of elements (keeping a representative) with method="correlation"; (ii) removal of most proximal element pairs in a reduced dimensional space.

Underlying method for correlation: widyr::pairwise_cor(sample, transcript,count, sort = TRUE, diag = FALSE, upper = FALSE)

Underlying custom method for reduced dimensions: select_closest_pairs = function(df) couples <- df |> head(n = 0) while (df |> nrow() > 0) pair <- df |> arrange(dist) |> head(n = 1) couples <- couples |> bind_rows(pair) df <- df |> filter(¡sample 1‘

couples

Value

A tbl object with with dropped redundant elements (e.g., samples).

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




 airway |>
 identify_abundant() |>
   remove_redundancy(
    .element = sample,
    .feature = transcript,
     .abundance =  count,
     method = "correlation"
     )
```

`resolve_complete_confounders_of_non_interest` *Resolve Complete Confounders of Non-Interest*

Description

This function identifies and resolves complete confounders among specified factors of non-interest within a 'SummarizedExperiment' object. Complete confounders occur when the levels of one factor are entirely predictable based on the levels of another factor. Such relationships can interfere with downstream analyses by introducing redundancy or collinearity.

Usage

```
resolve_complete_confounders_of_non_interest(se, ...)
```

Arguments

`se`	A 'SummarizedExperiment' object. This object contains assay data, row data (e.g., gene annotations), and column data (e.g., sample annotations).
`...`	Factors of non-interest (column names from 'colData(se)') to examine for complete confounders.

Details

The function systematically examines pairs of specified factors and determines whether they are completely confounded. If a pair of factors is found to be confounded, one of the factors is adjusted or removed to resolve the issue. The adjusted 'SummarizedExperiment' object is returned, preserving all assays and metadata except the resolved factors.

Complete confounders of non-interest can create dependencies between variables that may bias statistical models or violate their assumptions. This function systematically addresses this by: 1. Creating new columns with the suffix "___altered" for each specified factor to preserve original values 2. Identifying pairs of factors in the specified columns that are fully confounded 3. Resolving confounding by adjusting one of the factors in the "___altered" columns

The function creates new columns with the "___altered" suffix to store the modified values while preserving the original data. This allows users to compare the original and adjusted values if needed.

The resolution strategy depends on the analysis context and can be modified in the helper function ‘resolve_complete_confounders_of_non_interest_pair_SE()‘. By default, the function adjusts one of the confounded factors in the "___altered" columns.

Value

A ‘SummarizedExperiment‘ object with resolved confounders. The object retains its structure, including assays and metadata, but the column data (‘colData‘) is updated with new "___altered" columns containing the resolved factors.

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

library(SummarizedExperiment) library(dplyr)

Sample annotations sample_annotations <- data.frame(sample_id = paste0("Sample", seq(1, 9)), factor_of_interest = c(rep("treated", 4), rep("untreated", 5)), A = c("a1", "a2", "a1", "a2", "a1", "a2", "a1", "a2", "a3"), B = c("b1", "b1", "b2", "b1", "b1", "b1", "b2", "b1", "b3"), C = c("c1", "c1", "c1", "c1", "c1", "c1", "c1", "c1", "c3"), stringsAsFactors = FALSE)

Simulated assay data assay_data <- matrix(rnorm(100 * 9), nrow = 100, ncol = 9)

Row data (e.g., gene annotations) row_data <- data.frame(gene_id = paste0("Gene", seq_len(100)))

Create SummarizedExperiment object se <- SummarizedExperiment(assays = list(counts = assay_data), rowData = row_data, colData = DataFrame(sample_annotations))

Apply the function to resolve confounders se_resolved <- resolve_complete_confounders_of_non_interest(se, A, B, C)

View the updated column data colData(se_resolved)

See Also

SummarizedExperiment for creating and handling ‘SummarizedExperiment‘ objects.

Examples

```
# Load necessary libraries
```

resolve_complete_confounders_of_non_interest,SummarizedExperiment-method *resolve_complete_confounders_of_non_interest*

Description

resolve_complete_confounders_of_non_interest

resolve_complete_confounders_of_non_interest

Usage

```
## S4 method for signature 'SummarizedExperiment'
resolve_complete_confounders_of_non_interest(se, ...)

## S4 method for signature 'RangedSummarizedExperiment'
resolve_complete_confounders_of_non_interest(se, ...)
```

Arguments

se	A 'SummarizedExperiment' object
...	Factors of non-interest (column names from 'colData(se)') to examine for complete confounders

Value

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

rotate_dimensions *Rotate two dimensions (e.g., principal components) of an arbitrary angle*

Description

rotate_dimensions() takes as input a 'tbl' formatted as | <DIMENSION 1> | <DIMENSION 2> | <...> | and calculates the rotated dimensional space of the transcript abundance.

Usage

```
rotate_dimensions(
  .data,
  dimension_1_column,
  dimension_2_column,
  rotation_degrees,
  .element = NULL,
  of_samples = TRUE,
  dimension_1_column_rotated = NULL,
  dimension_2_column_rotated = NULL
)

## S4 method for signature 'SummarizedExperiment'
rotate_dimensions(
  .data,
  dimension_1_column,
  dimension_2_column,
  rotation_degrees,
  .element = NULL,
  of_samples = TRUE,
  dimension_1_column_rotated = NULL,
  dimension_2_column_rotated = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
rotate_dimensions(
  .data,
  dimension_1_column,
  dimension_2_column,
  rotation_degrees,
  .element = NULL,
  of_samples = TRUE,
  dimension_1_column_rotated = NULL,
  dimension_2_column_rotated = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
dimension_1_column	A character string. The column of the dimension 1
dimension_2_column	A character string. The column of the dimension 2
rotation_degrees	A real number between 0 and 360

`.element` The name of the element column (normally samples).

`of_samples` A boolean. In case the input is a tidybulk object, it indicates Whether the element column will be sample or transcript column

`dimension_1_column_rotated`
A character string. The column of the rotated dimension 1 (optional)

`dimension_2_column_rotated`
A character string. The column of the rotated dimension 2 (optional)

Details

‘r lifecycle::badge("maturing")‘

This function to rotate two dimensions such as the reduced dimensions.

Underlying custom method: rotation = function(m, d) r = d * pi / 180 ((bind_rows(c(‘1‘ = cos(r), ‘2‘ = -sin(r)), c(‘1‘ = sin(r), ‘2‘ = cos(r))) |> as_matrix())

Value

A tbl object with additional columns for the reduced dimensions. additional columns for the rotated dimensions. The rotated dimensions will be added to the original data set as ‘<NAME OF DIMENSION> rotated <ANGLE>‘ by default, or as specified in the input arguments.

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



counts.MDS =
 airway |>
 identify_abundant() |>
 reduce_dimensions( method="MDS", .dims = 3)

counts.MDS.rotated =  rotate_dimensions(counts.MDS, `Dim1`, `Dim2`, rotation_degrees = 45, .element = sample)
```

scale_abundance *Scale the counts of transcripts/genes*

Description

scale_abundance() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and Scales transcript abundance compansating for sequencing depth (e.g., with TMM algorithm, Robinson and Oshlack doi.org/10.1186/gb-2010-11-3-r25).

Usage

```
scale_abundance(
  .data,
  abundance = assayNames(.data)[1],
  method = "TMM",
  reference_sample = NULL,
  .subset_for_scaling = NULL,
  suffix = "_scaled",
  reference_selection_function = NULL,
  ...,
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
scale_abundance(
  .data,
  abundance = assayNames(.data)[1],
  method = "TMM",
  reference_sample = NULL,
  .subset_for_scaling = NULL,
  suffix = "_scaled",
  reference_selection_function = NULL,
  ...,
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
scale_abundance(
  .data,
  abundance = assayNames(.data)[1],
  method = "TMM",
  reference_sample = NULL,
  .subset_for_scaling = NULL,
  suffix = "_scaled",
  reference_selection_function = NULL,
  ...,
```

```
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
abundance	The name of the transcript/gene abundance column (character, preferred)
method	A character string. The scaling method passed to the back-end function (i.e., edgeR::calcNormFactors; "TMM","TMMwsp","RLE","upperquartile")
reference_sample	A character string. The name of the reference sample. If NULL the sample with highest total read count will be selected as reference.
.subset_for_scaling	A gene-wise quosure condition. This will be used to filter rows (features/genes) of the dataset. For example
suffix	A character string to append to the scaled abundance column name. Default is "_scaled".
reference_selection_function	DEPRECATED. please use reference_sample.
...	Further arguments.
.abundance	DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

‘r lifecycle::badge("maturing")‘

Scales transcript abundance compensating for sequencing depth (e.g., with TMM algorithm, Robinson and Oshlack doi.org/10.1186/gb-2010-11-3-r25). Lowly transcribed transcripts/genes (defined with minimum_counts and minimum_proportion parameters) are filtered out from the scaling procedure. The scaling inference is then applied back to all unfiltered data.

Underlying method edgeR::calcNormFactors(.data, method = c("TMM","TMMwsp","RLE","upperquartile"))

Value

A tbl object with additional columns with scaled data as ‘<NAME OF COUNT COLUMN>_scaled‘

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Robinson, M. D., & Oshlack, A. (2010). A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biology, 11(3), R25. doi:10.1186/gb-2010-11-3-r25

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex




 airway |>
   identify_abundant() |>
   scale_abundance()
```

`scale_x_log10_reverse` *scale_x_log10_reverse*

Description

A wrapper function that provides evenly spaced ticks with scientific notation for log10 reverse transformed x-axis. This is particularly useful for plots showing p-values or other values where smaller values should be displayed larger. The function applies a log10 transformation and reverses the axis to better visualize p-values without hard transforming the data, while maintaining the original p-value scale for interpretation. This allows you to see the full range of p-values with proper scaling while keeping the original values readable.

Usage

```
scale_x_log10_reverse(breaks = 5, digits = 2, ...)
```

Arguments

`breaks`	Number of breaks to display (default: 5)
`digits`	Number of digits for scientific notation (default: 2)
`...`	Additional arguments passed to scale_x_continuous

Details

‘r lifecycle::badge("maturing")‘

Value

A ggplot2 scale object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. https://ggplot2.tidyverse.org

Examples

```
library(ggplot2)
library(tibble)

# Create test data
test_data <- tibble(
  pvalue = c(0.0001, 0.001, 0.01, 0.05, 0.1, 0.5),
  fold_change = 1:6
)

# Use the wrapper function
test_data |>
  ggplot(aes(pvalue, fold_change)) +
  geom_point() +
  scale_x_log10_reverse()
```

`scale_y_log10_reverse` *scale_y_log10_reverse*

Description

A wrapper function that provides evenly spaced ticks with scientific notation for log10 reverse transformed y-axis. This is particularly useful for volcano plots and other plots showing p-values. The function applies a log10 transformation and reverses the axis to better visualize p-values without hard transforming the data, while maintaining the original p-value scale for interpretation. This allows you to see the full range of p-values with proper scaling while keeping the original values readable.

Usage

```
scale_y_log10_reverse(breaks = 5, digits = 2, ...)
```

Arguments

`breaks`	Number of breaks to display (default: 5)
`digits`	Number of digits for scientific notation (default: 2)
`...`	Additional arguments passed to scale_y_continuous

Details

‘r lifecycle::badge("maturing")‘

Value

A ggplot2 scale object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. https://ggplot2.tidyverse.org

Examples

```
library(ggplot2)
library(tibble)

# Create test data
test_data <- tibble(
  pvalue = c(0.0001, 0.001, 0.01, 0.05, 0.1, 0.5),
  fold_change = 1:6
)

# Use the wrapper function
test_data |>
  ggplot(aes(fold_change, pvalue)) +
  geom_point() +
  scale_y_log10_reverse()
```

`test_differential_abundance`

Perform differential transcription testing using edgeR quasi-likelihood (QLT), edgeR likelihood-ratio (LR), limma-voom, limma-voom-with-quality-weights or DESeq2

Description

test_differential_abundance() takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with additional columns for the statistics from the hypothesis test.

Usage

```
test_differential_abundance(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
test_differential_abundance(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
test_differential_abundance(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
```

```
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.formula	A formula representing the desired linear model. If there is more than one factor, they should be in the order factor of interest + additional factors.
abundance	The name of the transcript/gene abundance column (character, preferred)
contrasts	This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)
method	A character vector. Available methods are "edgeR_quasi_likelihood" (i.e., QLF), "edgeR_likelihood_ratio" (i.e., LRT), "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights", "glmmseq_lme4", "glmmseq_glmmtmb". Only one method can be specified at a time.
test_above_log2_fold_change	A positive real value. This works for edgeR and limma_voom methods. It uses the ‘treat‘ function, which tests that the difference in abundance is bigger than this threshold rather than zero https://pubmed.ncbi.nlm.nih.gov/19176553.
scaling_method	A character string. The scaling method passed to the back-end functions: edgeR and limma-voom (i.e., edgeR::calcNormFactors; "TMM","TMMwsp","RLE","upperquartile"). Setting the parameter to \"none\" will skip the compensation for sequencing-depth for the method edgeR or limma_voom.
omit_contrast_in_colnames	If just one contrast is specified you can choose to omit the contrast label in the colnames.
prefix	A character string. The prefix you would like to add to the result columns. It is useful if you want to compare several methods.
...	Further arguments passed to some of the internal experimental functions. For example for glmmSeq, it is possible to pass .dispersion, and .scaling_factor column tidyeval to skip the caluclation of dispersion and scaling and use precalculated values. This is helpful is you want to calculate those quantities on many

genes and do DE testing on fewer genes. .scaling_factor is the TMM value that can be obtained with tidybulk::scale_abundance.

`significance_threshold`
: DEPRECATED - A real between 0 and 1 (usually 0.05).

`fill_missing_values`
: DEPRECATED - A boolean. Whether to fill missing sample/transcript values with the median of the transcript. This is rarely needed.

`.contrasts`
: DEPRECATED - This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)

`.abundance`
: DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

‘r lifecycle::badge("maturing")‘

This function provides the option to use edgeR https://doi.org/10.1093/bioinformatics/btp616, limma-voom https://doi.org/10.1186/gb-2014-15-2-r29, limma_voom_sample_weights https://doi.org/10.1093/nar/gkv412 or DESeq2 https://doi.org/10.1186/s13059-014-0550-8 to perform the testing. All methods use raw counts, irrespective of if scale_abundance or adjust_abundance have been calculated, therefore it is essential to add covariates such as batch effects (if applicable) in the formula.

Underlying method for edgeR framework:

.data |>

Filter keep_abundant(factor_of_interest = !!(as.symbol(parse_formula(.formula)[1])), minimum_counts = minimum_counts, minimum_proportion = minimum_proportion) |>

Format select(!!.transcript,!!.sample,!!.abundance) |> spread(!!.sample,!!.abundance) |> as_matrix(rownames = !!.transcript) |>

edgeR edgeR::DGEList(counts = .) |> edgeR::calcNormFactors(method = scaling_method) |> edgeR::estimateDisp(design) |>

Fit edgeR::glmQLFit(design) |> // or glmFit according to choice edgeR::glmQLFTest(coef = 2, contrast = my_contrasts) // or glmLRT according to choice

Underlying method for DESeq2 framework:

keep_abundant(factor_of_interest = !!as.symbol(parse_formula(.formula)[[1]]), minimum_counts = minimum_counts, minimum_proportion = minimum_proportion) |>

DESeq2 DESeq2::DESeqDataSet(design = .formula) |> DESeq2::DESeq() |> DESeq2::results()

Underlying method for glmmSeq framework:

counts = .data |> assay(my_assay)

Create design matrix for dispersion, removing random effects design = model.matrix(object = .formula |> lme4::nobars(), data = metadata)

dispersion = counts |> edgeR::estimateDisp(design = design)

glmmSeq(.formula, countdata = counts , metadata = metadata |> as.data.frame(), dispersion = dispersion, progress = TRUE, method = method |> str_remove("(?i)^glmmSeq_"),)

Value

A consistent object (to the input) with additional columns for the statistics from the test (e.g., log fold change, p-value and false discovery rate).

A 'SummarizedExperiment' object

A 'SummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

McCarthy, D. J., Chen, Y., & Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Research, 40(10), 4288-4297. doi:10.1093/nar/gks042

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15(12), 550. doi:10.1186/s13059-014-0550-8

Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biology, 15(2), R29. doi:10.1186/gb-2014-15-2-r29

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



 # edgeR (default method)

 airway |>
 identify_abundant() |>
 test_differential_abundance( ~ condition, method = "edgeR_quasi_likelihood" )

 # You can also explicitly specify the method
 airway |>
 identify_abundant() |>
 test_differential_abundance( ~ condition, method = "edgeR_quasi_likelihood" )

 # The function `test_differential_abundance` operates with contrasts too

 airway |>
 identify_abundant(factor_of_interest = condition) |>
 test_differential_abundance(
     ~ 0 + condition,
     contrasts = c( "conditiontrt - conditionuntrt"),
```

```
    method = "edgeR_quasi_likelihood"
 )

 # DESeq2 - equivalent for limma-voom

my_se_mini = airway
my_se_mini$condition  = factor(my_se_mini$condition)

# demontrating with `fitType` that you can access any arguments to DESeq()
my_se_mini  |>
   identify_abundant(factor_of_interest = condition) |>
       test_differential_abundance( ~ condition, method="deseq2", fitType="local")

# testing above a log2 threshold, passes along value to lfcThreshold of results()
res <- my_se_mini  |>
   identify_abundant(factor_of_interest = condition) |>
        test_differential_abundance( ~ condition, method="deseq2",
            fitType="local",
            test_above_log2_fold_change=4 )

# Use random intercept and random effect models

 airway[1:50,] |>
  identify_abundant(factor_of_interest = condition) |>
  test_differential_abundance(
    ~ condition + (1 + condition | cell),
    method = "glmmseq_lme4", cores = 1
  )

# confirm that lfcThreshold was used
# Not run to keep the example fast
# res |>
#     mcols() |>
#     DESeq2::DESeqResults() |>
#     DESeq2::plotMA()


# The function `test_differential_abundance` operates with contrasts too

 my_se_mini |>
 identify_abundant() |>
 test_differential_abundance(
     ~ 0 + condition,
     contrasts = list(c("condition", "trt", "untrt")),
     method="deseq2",
     fitType="local"
 )
```

test_differential_expression *Perform differential expression testing using edgeR quasi-likelihood (QLT), edgeR likelihood-ratio (LR), limma-voom, limma-voom-with-quality-weights or DESeq2*

Description

test_differential_expression() is an alias for test_differential_abundance() that takes as input A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a consistent object (to the input) with additional columns for the statistics from the hypothesis test.

Usage

```
test_differential_expression(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
test_differential_expression(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
```

```
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
test_differential_expression(
  .data,
  .formula,
  abundance = assayNames(.data)[1],
  contrasts = NULL,
  method = c("edgeR_quasi_likelihood", "edgeR_likelihood_ratio",
   "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights",
    "glmmseq_lme4", "glmmseq_glmmtmb"),
  test_above_log2_fold_change = NULL,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  prefix = "",
  ...,
  significance_threshold = NULL,
  fill_missing_values = NULL,
  .contrasts = NULL,
  .abundance = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.formula	A formula representing the desired linear model. If there is more than one factor, they should be in the order factor of interest + additional factors.
abundance	The name of the transcript/gene abundance column (character, preferred)
contrasts	This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)
method	A character vector. Available methods are "edgeR_quasi_likelihood" (i.e., QLF), "edgeR_likelihood_ratio" (i.e., LRT), "edger_robust_likelihood_ratio", "DESeq2", "limma_voom", "limma_voom_sample_weights", "glmmseq_lme4", "glmmseq_glmmtmb". Only one method can be specified at a time.
test_above_log2_fold_change	A positive real value. This works for edgeR and limma_voom methods. It uses the ‘treat‘ function, which tests that the difference in abundance is bigger than this threshold rather than zero https://pubmed.ncbi.nlm.nih.gov/19176553.

`scaling_method` A character string. The scaling method passed to the back-end functions: edgeR and limma-voom (i.e., edgeR::calcNormFactors; "TMM","TMMwsp","RLE","upperquartile"). Setting the parameter to \"none\" will skip the compensation for sequencing-depth for the method edgeR or limma_voom.

`omit_contrast_in_colnames` If just one contrast is specified you can choose to omit the contrast label in the colnames.

`prefix` A character string. The prefix you would like to add to the result columns. It is useful if you want to compare several methods.

`...` Further arguments passed to some of the internal experimental functions. For example for glmmSeq, it is possible to pass .dispersion, and .scaling_factor column tidyeval to skip the caluclation of dispersion and scaling and use precalculated values. This is helpful is you want to calculate those quantities on many genes and do DE testing on fewer genes. .scaling_factor is the TMM value that can be obtained with tidybulk::scale_abundance.

`significance_threshold` DEPRECATED - A real between 0 and 1 (usually 0.05).

`fill_missing_values` DEPRECATED - A boolean. Whether to fill missing sample/transcript values with the median of the transcript. This is rarely needed.

`.contrasts` DEPRECATED - This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)

`.abundance` DEPRECATED. The name of the transcript/gene abundance column (symbolic, for backward compatibility)

Details

‘r lifecycle::badge("maturing")‘

This function provides the option to use edgeR https://doi.org/10.1093/bioinformatics/btp616, limma-voom https://doi.org/10.1186/gb-2014-15-2-r29, limma_voom_sample_weights https://doi.org/10.1093/nar/gkv412 or DESeq2 https://doi.org/10.1186/s13059-014-0550-8 to perform the testing. All methods use raw counts, irrespective of if scale_abundance or adjust_abundance have been calculated, therefore it is essential to add covariates such as batch effects (if applicable) in the formula.

Underlying method for edgeR framework:

.data |>

Filter keep_abundant(factor_of_interest = !!(as.symbol(parse_formula(.formula)[1])), minimum_counts = minimum_counts, minimum_proportion = minimum_proportion) |>

Format select(!!.transcript,!!.sample,!!.abundance) |> spread(!!.sample,!!.abundance) |> as_matrix(rownames = !!.transcript) |>

edgeR edgeR::DGEList(counts = .) |> edgeR::calcNormFactors(method = scaling_method) |> edgeR::estimateDisp(design) |>

Fit edgeR::glmQLFit(design) |> // or glmFit according to choice edgeR::glmQLFTest(coef = 2, contrast = my_contrasts) // or glmLRT according to choice

Underlying method for DESeq2 framework:

keep_abundant(factor_of_interest = !!as.symbol(parse_formula(.formula)[[1]]), minimum_counts = minimum_counts, minimum_proportion = minimum_proportion) |>

DESeq2 DESeq2::DESeqDataSet(design = .formula) |> DESeq2::DESeq() |> DESeq2::results()

Underlying method for glmmSeq framework:

counts = .data |> assay(my_assay)

Create design matrix for dispersion, removing random effects design = model.matrix(object = .formula |> lme4::nobars(), data = metadata)

dispersion = counts |> edgeR::estimateDisp(design = design)

glmmSeq(.formula, countdata = counts , metadata = metadata |> as.data.frame(), dispersion = dispersion, progress = TRUE, method = method |> str_remove("(?i)^glmmSeq_"),)

Value

A consistent object (to the input) with additional columns for the statistics from the test (e.g., log fold change, p-value and false discovery rate).

A 'SummarizedExperiment' object

A 'SummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

McCarthy, D. J., Chen, Y., & Smyth, G. K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Research, 40(10), 4288-4297. doi:10.1093/nar/gks042

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15(12), 550. doi:10.1186/s13059-014-0550-8

Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biology, 15(2), R29. doi:10.1186/gb-2014-15-2-r29

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex
```

```
 # edgeR (default method)

 airway |>
 identify_abundant() |>
 test_differential_expression( ~ condition, method = "edgeR_quasi_likelihood" )

 # You can also explicitly specify the method
 airway |>
 identify_abundant() |>
 test_differential_expression( ~ condition, method = "edgeR_quasi_likelihood" )

 # The function `test_differential_expression` operates with contrasts too

 airway |>
 identify_abundant(factor_of_interest = condition) |>
 test_differential_expression(
     ~ 0 + condition,
     contrasts = c( "conditiontrt - conditionuntrt"),
    method = "edgeR_quasi_likelihood"
 )

 # DESeq2 - equivalent for limma-voom

my_se_mini = airway
my_se_mini$condition  = factor(my_se_mini$condition)

# demontrating with `fitType` that you can access any arguments to DESeq()
my_se_mini  |>
   identify_abundant(factor_of_interest = condition) |>
       test_differential_expression( ~ condition, method="deseq2", fitType="local")

# testing above a log2 threshold, passes along value to lfcThreshold of results()
res <- my_se_mini  |>
   identify_abundant(factor_of_interest = condition) |>
        test_differential_expression( ~ condition, method="deseq2",
            fitType="local",
            test_above_log2_fold_change=4 )

# Use random intercept and random effect models

 airway[1:50,] |>
  identify_abundant(factor_of_interest = condition) |>
  test_differential_expression(
    ~ condition + (1 + condition | cell),
    method = "glmmseq_lme4", cores = 1
  )

# confirm that lfcThreshold was used

## Not run:
    res |>
        mcols() |>
        DESeq2::DESeqResults() |>
```

```
        DESeq2::plotMA()

## End(Not run)

# The function `test_differential_expression` operates with contrasts too

 my_se_mini |>
 identify_abundant() |>
 test_differential_expression(
     ~ 0 + condition,
     contrasts = list(c("condition", "trt", "untrt")),
     method="deseq2",
         fitType="local"
 )
```

`test_gene_enrichment` *analyse gene enrichment with EGSEA*

Description

test_gene_enrichment() takes as input a ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a ‘tbl‘ of gene set information

Usage

```
test_gene_enrichment(
  .data,
  .formula,
  .entrez,
  .abundance = NULL,
  contrasts = NULL,
  methods = c("camera", "roast", "safe", "gage", "padog", "globaltest", "ora"),
  gene_sets = c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease",
    "kegg_metabolism", "kegg_signaling"),
  species,
  cores = parallel::detectCores(),
  method = NULL,
  .contrasts = NULL
)

## S4 method for signature 'SummarizedExperiment'
test_gene_enrichment(
  .data,
  .formula,
  .entrez,
  .abundance = NULL,
```

```
  contrasts = NULL,
  methods = c("camera", "roast", "safe", "gage", "padog", "globaltest", "ora"),
  gene_sets = c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease",
    "kegg_metabolism", "kegg_signaling"),
  species,
  cores = parallel::detectCores(),
  method = NULL,
  .contrasts = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
test_gene_enrichment(
  .data,
  .formula,
  .entrez,
  .abundance = NULL,
  contrasts = NULL,
  methods = c("camera", "roast", "safe", "gage", "padog", "globaltest", "ora"),
  gene_sets = c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease",
    "kegg_metabolism", "kegg_signaling"),
  species,
  cores = parallel::detectCores(),
  method = NULL,
  .contrasts = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.formula	A formula with no response variable, representing the desired linear model
.entrez	The ENTREZ ID of the transcripts/genes
.abundance	The name of the transcript/gene abundance column
contrasts	This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)
methods	A character vector. One or 3 or more methods to use in the testing (currently EGSEA errors if 2 are used). Type EGSEA::egsea.base() to see the supported GSE methods.
gene_sets	A character vector or a list. It can take one or more of the following built-in collections as a character vector: c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease", "kegg_metabolism", "kegg_signaling"), to be used with EGSEA buildIdx. c1 is human specific. Alternatively, a list of user-supplied gene sets can be provided, to be used with EGSEA buildCustomIdx. In that case, each gene set is a character vector of Entrez IDs and the names of the list are the gene set names.

species	A character. It can be human, mouse or rat.
cores	An integer. The number of cores available
method	DEPRECATED. Please use methods.
.contrasts	DEPRECATED - This parameter takes the format of the contrast parameter of the method of choice. For edgeR and limma-voom is a character vector. For DESeq2 is a list including a character vector of length three. The first covariate is the one the model is tested against (e.g., ~ factor_of_interest)

Details

‘r lifecycle::badge("maturing")‘

This wrapper executes ensemble gene enrichment analyses of the dataset using EGSEA (DOI:0.12688/f1000research.12544.1

dge = data |> keep_abundant(factor_of_interest = !!as.symbol(parse_formula(.formula)[[1]]), !!.sample, !!.entrez, !!.abundance) |>

Make sure transcript names are adjacent [...] |> as_matrix(rownames = !!.entrez) |> edgeR::DGEList(counts = .)

idx = buildIdx(entrezIDs = rownames(dge), species = species, msigdb.gsets = msigdb.gsets, kegg.exclude = kegg.exclude)

dge |>

Calculate weights limma::voom(design, plot = FALSE) |>

Execute EGSEA egsea(contrasts = my_contrasts, baseGSEAs = methods, gs.annots = idx, sort.by = "med.rank", num.threads = cores, report = FALSE)

Value

A consistent object (to the input)

A consistent object (to the input)

A consistent object (to the input)

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Alhamdoosh, M., Ng, M., Wilson, N. J., Sheridan, J. M., Huynh, H., Wilson, M. J., & Ritchie, M. E. (2017). Combining multiple tools outperforms individual methods for gene set enrichment analysis in single-cell RNA-seq data. Genome Biology, 18(1), 174. doi:10.1186/s13059-017-1279-y

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex
```

```
library(tidySummarizedExperiment)

library("EGSEA")

## Not run:
df_entrez <- airway |>
  mutate(entrez = .feature) |>
  aggregate_duplicates(.transcript = entrez )

 test_gene_enrichment(
  df_entrez,
  ~ condition,
  .entrez = entrez,
  .abundance = count,
  methods = c("roast" , "safe", "gage"  ,  "padog" , "globaltest", "ora" ),
 gene_sets = c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease", "kegg_metabolism", "kegg_signaling")
  species="human",
  cores = 2
 )

## End(Not run)
```

`test_gene_overrepresentation` *analyse gene over-representation with GSEA*

Description

test_gene_overrepresentation() takes as input a ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a ‘tbl‘ with the GSEA statistics

Usage

```
test_gene_overrepresentation(
  .data,
  .entrez,
  .do_test,
  species,
  .sample = NULL,
  gene_sets = NULL,
  gene_set = NULL
)
```

```
## S4 method for signature 'SummarizedExperiment'
test_gene_overrepresentation(
  .data,
  .entrez,
  .do_test,
  species,
  .sample = NULL,
  gene_sets = NULL,
  gene_set = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
test_gene_overrepresentation(
  .data,
  .entrez,
  .do_test,
  species,
  .sample = NULL,
  gene_sets = NULL,
  gene_set = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.entrez	The ENTREZ ID of the transcripts/genes
.do_test	A boolean column name symbol. It indicates the transcript to check
species	A character. For example, human or mouse. MSigDB uses the latin species names (e.g., \"Mus musculus\", \"Homo sapiens\")
.sample	The name of the sample column
gene_sets	A character vector. The subset of MSigDB datasets you want to test against (e.g. \"C2\"). If NULL all gene sets are used (suggested). This argument was added to avoid time overflow of the examples.
gene_set	DEPRECATED. Use gene_sets instead.

Details

‘r lifecycle::badge("maturing")‘

This wrapper execute gene enrichment analyses of the dataset using a list of transcripts and GSEA. This wrapper uses clusterProfiler (DOI: doi.org/10.1089/omi.2011.0118) on the back-end.

Get MSigDB data msigdb_data = msigdbr::msigdbr(species = species)

Filter for specific gene collections if provided

msigdb_data = filter(msigdb_data, gs_collection

Process the data msigdb_data |> nest(data = -gs_collection) |> mutate(test = map(data, ~ clusterProfiler::enricher(my_entrez_rank, TERM2GENE=.x |> select(gs_name, ncbi_gene), pvalueCutoff = 1) |> as_tibble()))

Value

A consistent object (to the input)

A 'SummarizedExperiment' object

A 'RangedSummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Yu, G., Wang, L. G., Han, Y., & He, Q. Y. (2012). clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology, 16(5), 284-287. doi:10.1089/omi.2011.0118

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex



print("Not run for build time.")

# airway = airway[!rowData(airway)$entrez |> is.na(),] |> aggregate_duplicates(.transcript = entrez)
# df_entrez = mutate(df_entrez, do_test = feature %in% c("TNFRSF4", "PLCH2", "PADI4", "PAX7"))

## Not run:
 test_gene_overrepresentation(
  df_entrez,
  .sample = sample,
  .entrez = entrez,
  .do_test = do_test,
  species="Homo sapiens",
   gene_sets =c("C2")
 )

## End(Not run)
```

test_gene_rank *analyse gene rank with GSEA*

Description

test_gene_rank() takes as input a ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment)) and returns a ‘tbl‘ with the GSEA statistics

Usage

```
test_gene_rank(
  .data,
  .entrez,
  .arrange_desc,
  species,
  gene_sets = NULL,
  gene_set = NULL
)

## S4 method for signature 'SummarizedExperiment'
test_gene_rank(
  .data,
  .entrez,
  .arrange_desc,
  species,
  gene_sets = NULL,
  gene_set = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
test_gene_rank(
  .data,
  .entrez,
  .arrange_desc,
  species,
  gene_sets = NULL,
  gene_set = NULL
)
```

Arguments

.data	A ‘tbl‘ (with at least three columns for sample, feature and transcript abundance) or ‘SummarizedExperiment‘ (more convenient if abstracted to tibble with library(tidySummarizedExperiment))
.entrez	The ENTREZ ID of the transcripts/genes

.arrange_desc	A column name of the column to arrange in decreasing order
species	A character. For example, human or mouse. MSigDB uses the latin species names (e.g., \"Mus musculus\", \"Homo sapiens\")
gene_sets	A character vector or a list. It can take one or more of the following built-in collections as a character vector: c("h", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "kegg_disease", "kegg_metabolism", "kegg_signaling"), to be used with EGSEA buildIdx. c1 is human specific. Alternatively, a list of user-supplied gene sets can be provided, to be used with EGSEA buildCustomIdx. In that case, each gene set is a character vector of Entrez IDs and the names of the list are the gene set names.
gene_set	DEPRECATED. Use gene_sets instead.

Details

[Maturing]

This wrapper execute gene enrichment analyses of the dataset using a list of transcripts and GSEA. This wrapper uses clusterProfiler (DOI: doi.org/10.1089/omi.2011.0118) on the back-end.

Undelying method: my_gene_collection <- msigdbr::msigdbr(species = species)

my_gene_collection <- filter(my_gene_collection, gs_collection

Execute calculation nest(data = -gs_collection) |> mutate(fit = map(data, ~ clusterProfiler::GSEA(my_entrez_rank, TERM2GENE=.x |> select(gs_name, ncbi_gene), pvalueCutoff = 1)

))

Value

A consistent object (to the input)

A 'SummarizedExperiment' object

A 'RangedSummarizedExperiment' object

References

Mangiola, S., Molania, R., Dong, R., Doyle, M. A., & Papenfuss, A. T. (2021). tidybulk: an R tidy framework for modular transcriptomic data analysis. Genome Biology, 22(1), 42. doi:10.1186/s13059-020-02233-7

Yu, G., Wang, L. G., Han, Y., & He, Q. Y. (2012). clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology, 16(5), 284-287. doi:10.1089/omi.2011.0118

Examples

```
## Load airway dataset for examples

  data('airway', package = 'airway')
  # Ensure a 'condition' column exists for examples expecting it

   SummarizedExperiment::colData(airway)$condition <- SummarizedExperiment::colData(airway)$dex
```

```
print("Not run for build time.")

## Not run:

df_entrez = airway
df_entrez = mutate(df_entrez, do_test = .feature %in% c("TNFRSF4", "PLCH2", "PADI4", "PAX7"))
df_entrez  = df_entrez |> test_differential_abundance(~ condition)


test_gene_rank(
 df_entrez,
  .sample = .sample,
 .entrez = entrez,
  species="Homo sapiens",
   gene_sets =c("C2"),
 .arrange_desc = logFC
 )



## End(Not run)
```

---

test_stratification_cellularity,SummarizedExperiment-method
test_stratification_cellularity

Description

test_stratification_cellularity

test_stratification_cellularity

Usage

```
## S4 method for signature 'SummarizedExperiment'
test_stratification_cellularity(
  .data,
  .formula,
  .abundance = NULL,
  method = c("cibersort", "llsr", "epic", "mcp_counter", "quantiseq", "xcell"),
  reference = X_cibersort,
  ...
)

## S4 method for signature 'RangedSummarizedExperiment'
test_stratification_cellularity(
```

```
  .data,
  .formula,
  .abundance = NULL,
  method = c("cibersort", "llsr", "epic", "mcp_counter", "quantiseq", "xcell"),
  reference = X_cibersort,
  ...
)
```

Arguments

.data	A 'SummarizedExperiment' object
.formula	A formula representing the desired linear model
.abundance	The name of the transcript/gene abundance column
method	A character string naming the deconvolution method
reference	Reference matrix or method-specific handle
...	Additional arguments passed through to the underlying deconvolution function

Value

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

A consistent object (to the input) with additional columns for the statistics from the hypothesis test (e.g., log fold change, p-value and false discovery rate).

---

tximeta_summarizeToGene_object

Needed for tests tximeta_summarizeToGene_object, It is SummarizedExperiment from tximeta

---

Description

Needed for tests tximeta_summarizeToGene_object, It is SummarizedExperiment from tximeta

Usage

```
tximeta_summarizeToGene_object
```

Format

An object of class `RangedSummarizedExperiment` with 10 rows and 1 columns.

vignette_manuscript_signature_boxplot

Needed for vignette vignette_manuscript_signature_boxplot

Description

Needed for vignette vignette_manuscript_signature_boxplot

Usage

```
vignette_manuscript_signature_boxplot
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 899 rows and 12 columns.

vignette_manuscript_signature_tsne

Needed for vignette vignette_manuscript_signature_tsne

Description

Needed for vignette vignette_manuscript_signature_tsne

Usage

```
vignette_manuscript_signature_tsne
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 283 rows and 10 columns.

vignette_manuscript_signature_tsne2 *Needed for vignette vignette_manuscript_signature_tsne2*

Description

Needed for vignette vignette_manuscript_signature_tsne2

Usage

```
vignette_manuscript_signature_tsne2
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 283 rows and 9 columns.

X_cibersort *Cibersort reference*

Description

Cibersort reference

Usage

```
X_cibersort
```

Format

An object of class `data.frame` with 547 rows and 22 columns.

References

Newman, A. M., Liu, C. L., Green, M. R., Gentles, A. J., Feng, W., Xu, Y., Hoang, C. D., Diehn, M., & Alizadeh, A. A. (2015). Robust enumeration of cell subsets from tissue expression profiles. Nature Methods, 12(5), 453–457. https://doi.org/10.1038/nmeth.3337

Index

∗ **datasets**
txiimeta_summarizeToGene_object, 72
vignette_manuscript_signature_boxplot, 73
vignette_manuscript_signature_tsne, 73
vignette_manuscript_signature_tsne2, 74
X_cibersort, 74
∗ **internal**
.rotate_dimensions_se, 3
check_if_counts_is_na, 11
check_if_duplicated_genes, 12
.describe_transcript_SE *(describe_transcript)*, 17
.rotate_dimensions_se, 3

adjust_abundance, 4
adjust_abundance,RangedSummarizedExperiment-method *(adjust_abundance)*, 4
adjust_abundance,SummarizedExperiment-method *(adjust_abundance)*, 4
aggregate_duplicates, 7
aggregate_duplicates,RangedSummarizedExperiment-method *(aggregate_duplicates)*, 7
aggregate_duplicates,SummarizedExperiment-method *(aggregate_duplicates)*, 7
as_matrix, 9
as_SummarizedExperiment, 10
as_SummarizedExperiment,tbl_df-method *(as_SummarizedExperiment)*, 10

check_if_counts_is_na, 11
check_if_duplicated_genes, 12
cluster_elements, 12
cluster_elements,RangedSummarizedExperiment-method *(cluster_elements)*, 12
cluster_elements,SummarizedExperiment-method *(cluster_elements)*, 12

deconvolve_cellularity, 14
deconvolve_cellularity,RangedSummarizedExperiment-method *(deconvolve_cellularity)*, 14
deconvolve_cellularity,SummarizedExperiment-method *(deconvolve_cellularity)*, 14
describe_transcript, 17
describe_transcript,RangedSummarizedExperiment-method *(describe_transcript)*, 17
describe_transcript,SummarizedExperiment-method *(describe_transcript)*, 17

fill_missing_abundance, 18

get_bibliography, 19
get_bibliography,RangedSummarizedExperiment-method *(get_bibliography)*, 19
get_bibliography,SummarizedExperiment-method *(get_bibliography)*, 19
get_X_cibersort, 20

identify_abundant, 21
identify_abundant,RangedSummarizedExperiment-method *(identify_abundant)*, 21
identify_abundant,SummarizedExperiment-method *(identify_abundant)*, 21
impute_missing_abundance, 24
impute_missing_abundance,RangedSummarizedExperiment-method *(impute_missing_abundance)*, 24
impute_missing_abundance,SummarizedExperiment-method *(impute_missing_abundance)*, 24

keep_abundant, 26
keep_abundant,RangedSummarizedExperiment-method, 28
keep_abundant,SummarizedExperiment-method, 29
keep_variable, 30
keep_variable,RangedSummarizedExperiment-method *(keep_variable)*, 30
keep_variable,SummarizedExperiment-method *(keep_variable)*, 30

log10_reverse_trans, 31
logit_trans, 32

pivot_sample, 33
pivot_sample,RangedSummarizedExperiment-method
(pivot_sample), 33
pivot_sample,SummarizedExperiment-method
(pivot_sample), 33
pivot_transcript, 34
pivot_transcript,RangedSummarizedExperiment-method
(pivot_transcript), 34
pivot_transcript,SummarizedExperiment-method
(pivot_transcript), 34

quantile_normalise_abundance, 35
quantile_normalise_abundance,RangedSummarizedExperiment-method
(quantile_normalise_abundance), 35
quantile_normalise_abundance,SummarizedExperiment-method
(quantile_normalise_abundance), 35

reduce_dimensions, 37
reduce_dimensions,RangedSummarizedExperiment-method
(reduce_dimensions), 37
reduce_dimensions,SummarizedExperiment-method
(reduce_dimensions), 37
remove_redundancy, 40
remove_redundancy,RangedSummarizedExperiment-method
(remove_redundancy), 40
remove_redundancy,SummarizedExperiment-method
(remove_redundancy), 40
resolve_complete_confounders_of_non_interest, 43
resolve_complete_confounders_of_non_interest,RangedSummarizedExperiment-method
(resolve_complete_confounders_of_non_interest), 45
resolve_complete_confounders_of_non_interest,SummarizedExperiment-method
(resolve_complete_confounders_of_non_interest), 45
rotate_dimensions, 45
rotate_dimensions,RangedSummarizedExperiment-method
(rotate_dimensions), 45
rotate_dimensions,SummarizedExperiment-method
(rotate_dimensions), 45

scale_abundance, 48
scale_abundance,RangedSummarizedExperiment-method
(scale_abundance), 48
scale_abundance,SummarizedExperiment-method
(scale_abundance), 48
scale_x_log10_reverse, 50
scale_y_log10_reverse, 51
SummarizedExperiment, *44*

test_differential_abundance, 52
test_differential_abundance,RangedSummarizedExperiment-method
(test_differential_abundance), 52
test_differential_abundance,SummarizedExperiment-method
(test_differential_abundance), 52
test_differential_expression, 57
test_differential_expression,RangedSummarizedExperiment-method
(test_differential_expression), 58
test_differential_expression,SummarizedExperiment-method
(test_differential_expression), 58
test_gene_enrichment, 63
test_gene_enrichment,RangedSummarizedExperiment-method
(test_gene_enrichment), 63
test_gene_enrichment,SummarizedExperiment-method
(test_gene_enrichment), 63
test_gene_overrepresentation, 66
test_gene_overrepresentation,RangedSummarizedExperiment-method
(test_gene_overrepresentation), 66
test_gene_overrepresentation,SummarizedExperiment-method
(test_gene_overrepresentation), 66
test_gene_rank, 69
test_gene_rank,RangedSummarizedExperiment-method
(test_gene_rank), 69
test_gene_rank,SummarizedExperiment-method
(test_gene_rank), 69
test_stratification_cellularity,RangedSummarizedExperiment-method
(test_stratification_cellularity), 71
test_stratification_cellularity,SummarizedExperiment-method
(test_stratification_cellularity), 71
tximeta_summarizeToGene_object, 72

vignette_manuscript_signature_boxplot, 73
vignette_manuscript_signature_tsne, 73
vignette_manuscript_signature_tsne2, 74

X_cibersort, 74