Package ‘tidybulk’

October 17, 2020

Type Package

Title Brings transcriptomics to the tidyverse

Version 1.0.2

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 (>= 4.0.0)

Imports tibble, readr, dplyr, magrittr, tidyr, rlang, purrr,
preprocessCore, stats, parallel, utils, lifecycle

Suggests testthat, AnnotationDbi, BiocManager, Rsubread, e1071, edgeR,
limma, org.Hs.eg.db, sva, GGally, knitr, qpdf, covr, Seurat,
KernSmooth, Rtsne, EGSEA, SummarizedExperiment, S4Vectors,
ggplot2, widyr, clusterProfiler, msigdbr

VignetteBuilder knitr

RdMacros lifecycle

Biarch true

biocViews AssayDomain, Infrastructure

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

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

git_branch RELEASE_3_11

git_last_commit bfa4dd1

git_last_commit_date 2020-06-08

Date/Publication 2020-10-16

Author Stefano Mangiola [aut, cre]

Maintainer Stefano Mangiola <`mangiolastefano@gmail.com`>

R topics documented:

adjust_abundance *Adjust transcript abundance for unwanted variation*

Description

adjust_abundance() takes as imput a 'tbl' formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a 'tbl' with an edditional adjusted abundance column. This method uses scaled counts if present.

Usage

```
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'spec_tbl_df'
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tbl_df'
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tidybulk'
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'SummarizedExperiment'
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'RangedSummarizedExperiment'
adjust_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  log_transform = TRUE,
  action = "add",
  ...
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.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_intrest + batch)

`.sample` The name of the sample column

`.transcript` The name of the transcript/gene column

`.abundance` The name of the transcript/gene abundance column

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

`action` A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

`...` Further parameters passed to the function sva::ComBat

Details

Maturing

This function adjusts the abundance for (known) unwanted variation. At the moment just an unwanted covariated is allowed at a time.

Value

A ‘tbl‘ with additional columns for the adjusted counts as ‘<COUNT COLUMN>_adjusted‘

A ‘tbl‘ with additional columns for the adjusted counts as ‘<COUNT COLUMN>_adjusted‘

A ‘tbl‘ with additional columns for the adjusted counts as ‘<COUNT COLUMN>_adjusted‘

A ‘tbl‘ with additional columns for the adjusted counts as ‘<COUNT COLUMN>_adjusted‘

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
cm = tidybulk::counts_mini
cm$batch = 0
cm$batch[cm$sample %in% c("SRR1740035", "SRR1740043")] = 1

res =
adjust_abundance(
cm,
~ condition + batch,
.sample = sample,
.transcript = transcript,
.abundance = count
)
```

---

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 imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a ‘tbl‘ with aggregated transcripts that were duplicated.

Usage

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

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

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

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

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

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

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.sample`	The name of the sample column

`.transcript` The name of the transcript/gene column

`.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

Details

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.

Value

A ‘tbl‘ object with aggregated transcript abundance and annotation

A ‘tbl‘ object with aggregated transcript abundance and annotation

A ‘tbl‘ object with aggregated transcript abundance and annotation

A ‘tbl‘ object with aggregated transcript abundance and annotation

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
aggregate_duplicates(
tidybulk::counts_mini,
sample,
transcript,
`count`,
aggregation_function = sum
)
```

---

arrange *drplyr-methods*

Description

‘arrange()‘ order the rows of a data frame rows by the values of selected columns.

Unlike other dplyr verbs, ‘arrange()‘ largely ignores grouping; you need to explicit mention grouping variables (or use ‘by_group = TRUE‘) in order to group by them, and functions of variables are evaluated once per data frame, not once per group.

Usage

```
arrange(.data, ..., .by_group = FALSE)

## Default S3 method:
arrange(.data, ..., .by_group = FALSE)

bind_rows(..., .id = NULL)

bind_cols(..., .id = NULL)

ungroup(x, ...)
```

Arguments

`.data` A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.

`...` <[‘tidy-eval‘][dplyr_tidy_eval]> Variables, or functions or variables. Use [desc()] to sort a variable in descending order.

`.by_group` If ‘TRUE‘, will sort first by grouping variable. Applies to grouped data frames only.

`.id` Data frame identifier.

When ‘.id‘ is supplied, a new column of identifiers is created to link each row to its original data frame. The labels are taken from the named arguments to ‘bind_rows()‘. When a list of data frames is supplied, the labels are taken from the names of the list. If no names are found a numeric sequence is used instead.

`x` A [tbl()]

Details

## Locales The sort order for character vectors will depend on the collating sequence of the locale in use: see [locales()].

## Missing values Unlike base sorting with ‘sort()‘, ‘NA‘ are: * always sorted to the end for local data, even when wrapped with ‘desc()‘. * treated differently for remote data, depending on the backend.

Value

A tibble Arrange rows by column values

An object of the same type as ‘.data‘.

* All rows appear in the output, but (usually) in a different place. * Columns are not modified. * Groups are not modified. * Data frame attributes are preserved.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: `filter()`, `mutate()`, `rename()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
arrange(mtcars, cyl, disp)
```

`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`	A character string of the rownames
`do_check`	A boolean

Value

A matrix

Examples

```
as_matrix(head(dplyr::select(tidybulk::counts_mini, transcript, count)), rownames=transcript)
```

`bind` *Efficiently bind multiple data frames by row and column*

Description

This is an efficient implementation of the common pattern of ‘do.call(rbind, dfs)‘ or ‘do.call(cbind, dfs)‘ for binding many data frames into one.

Arguments

`...` Data frames to combine.

Each argument can either be a data frame, a list that could be a data frame, or a list of data frames.

When row-binding, columns are matched by name, and any missing columns will be filled with NA.

When column-binding, rows are matched by position, so all data frames must have the same number of rows. To match by value, not position, see [mutate-joins].

`.id` Data frame identifier.

When ‘.id‘ is supplied, a new column of identifiers is created to link each row to its original data frame. The labels are taken from the named arguments to ‘bind_rows()‘. When a list of data frames is supplied, the labels are taken from the names of the list. If no names are found a numeric sequence is used instead.

Details

The output of ‘bind_rows()‘ will contain a column if that column appears in any of the inputs.

Value

‘bind_rows()‘ and ‘bind_cols()‘ return the same type as the first input, either a data frame, ‘tbl_df‘, or ‘grouped_df‘.

Examples

```
`%>%` = magrittr::`%>%`
one <- mtcars[1:4, ]
two <- mtcars[11:14, ]

# You can supply data frames as arguments:
bind_rows(one, two)
```

`breast_tcga_mini` *Data set*

Description

Data set

Usage

```
breast_tcga_mini
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 125500 rows and 5 columns.

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

Description

cluster_elements() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and identify clusters in the data.

Usage

```
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'spec_tbl_df'
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tbl_df'
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tidybulk'
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'SummarizedExperiment'
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'RangedSummarizedExperiment'
cluster_elements(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  of_samples = TRUE,
  log_transform = TRUE,
  action = "add",
  ...
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.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 cluster algorithm to use, ay 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

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

`action` A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

`...` Further parameters passed to the function kmeans

Details

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 clustering is supported, the plan is to introduce more clustering methods.

Value

A tbl object with additional columns with cluster labels

A tbl object with additional columns with cluster labels

A tbl object with additional columns with cluster labels

A tbl object with additional columns with cluster labels

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
cluster_elements(tidybulk::counts_mini, sample, transcript, count,centers = 2, method="kmeans")
```

`counts` *Example data set*

Description

Example data set

Usage

```
counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 938112 rows and 8 columns.

`counts_ensembl` *Counts with ensembl annotation*

Description

Counts with ensembl annotation

Usage

```
counts_ensembl
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 119 rows and 6 columns.

`counts_mini` *Example data set reduced*

Description

Example data set reduced

Usage

```
counts_mini
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 2635 rows and 6 columns.

`deconvolve_cellularity` *Get cell type proportions from samples*

Description

deconvolve_cellularity() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a ‘tbl‘ with the estimated cell type abundance for each sample

Usage

```
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)

## S4 method for signature 'spec_tbl_df'
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)

## S4 method for signature 'tbl_df'
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)

## S4 method for signature 'tidybulk'
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)

## S4 method for signature 'SummarizedExperiment'
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)

## S4 method for signature 'RangedSummarizedExperiment'
deconvolve_cellularity(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  reference = X_cibersort,
  method = "cibersort",
  action = "add",
  ...
)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.sample`	The name of the sample column
`.transcript`	The name of the transcript/gene column
`.abundance`	The name of the transcript/gene abundance column
`reference`	A data frame. The transcript/cell_type data frame of integer transcript abundance
`method`	A character string. The method to be used. At the moment Cibersort (default) and llsr (linear least squares regression) are available.
`action`	A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).
`...`	Further parameters passed to the function Cibersort

Details

Maturing

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

Value

A ‘tbl‘ object including additional columns for each cell type estimated

A ‘tbl‘ object including additional columns for each cell type estimated

A ‘tbl‘ object including additional columns for each cell type estimated

A ‘tbl‘ object including additional columns for each cell type estimated

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
deconvolve_cellularity(tidybulk::counts, sample, transcript, `count`, cores = 2)
```

---

distinct *distinct*

Description

distinct

Usage

```
distinct(.data, ..., .keep_all = FALSE)
```

Arguments

`.data`	A tbl. (See dplyr)
`...`	Data frames to combine (See dplyr)
`.keep_all`	If TRUE, keep all variables in .data. If a combination of ... is not distinct, this keeps the first row of values. (See dplyr)

Value

A tt object

Examples

```
distinct(tidybulk::counts_mini)
```

---

ensembl_symbol_mapping *Data set*

Description

Data set

Usage

```
ensembl_symbol_mapping
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 291249 rows and 3 columns.

ensembl_to_symbol *Add transcript symbol column from ensembl id for human and mouse data*

Description

ensembl_to_symbol() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <ENSEMBL_ID> | <COUNT> | <...> | and returns a ‘tbl‘ with the additional transcript symbol column

Usage

```
ensembl_to_symbol(.data, .ensembl, action = "add")

## S4 method for signature 'spec_tbl_df'
ensembl_to_symbol(.data, .ensembl, action = "add")

## S4 method for signature 'tbl_df'
ensembl_to_symbol(.data, .ensembl, action = "add")

## S4 method for signature 'tidybulk'
ensembl_to_symbol(.data, .ensembl, action = "add")
```

Arguments

`.data`	A ‘tbl‘ formatted as	<SAMPLE>	<ENSEMBL_ID>	<COUNT>	<...>	
`.ensembl`	A character string. The column that is represents ensembl gene id
`action`	A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

Details

Maturing

This is useful since different resources use ensembl IDs while others use gene symbol IDs. At the moment this work for human (genes and transcripts) and mouse (genes) data.

Value

A ‘tbl‘ object including additional columns for transcript symbol

A ‘tbl‘ object including additional columns for transcript symbol

A ‘tbl‘ object including additional columns for transcript symbol

A ‘tbl‘ object including additional columns for transcript symbol

Examples

```
ensembl_to_symbol(tidybulk::counts_ensembl, ens)
```

`filter` *Subset rows using column values*

Description

‘filter()‘ retains the rows where the conditions you provide a ‘TRUE‘. Note that, unlike base subsetting with ‘[‘, rows where the condition evaluates to ‘NA‘ are dropped.

Usage

```
filter(.data, ..., .preserve = FALSE)
```

Arguments

`.data` A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.

`...` <[‘tidy-eval‘][dplyr_tidy_eval]> Logical predicates defined in terms of the variables in ‘.data‘. Multiple conditions are combined with ‘&‘. Only rows where the condition evaluates to ‘TRUE‘ are kept.

`.preserve` when ‘FALSE‘ (the default), the grouping structure is recalculated based on the resulting data, otherwise it is kept as is.

Details

dplyr is not yet smart enough to optimise filtering optimisation on grouped datasets that don’t need grouped calculations. For this reason, filtering is often considerably faster on [ungroup()]ed data.

Value

An object of the same type as ‘.data‘.

* Rows are a subset of the input, but appear in the same order. * Columns are not modified. * The number of groups may be reduced (if ‘.preserve‘ is not ‘TRUE‘). * Data frame attributes are preserved.

Useful filter functions

* [‘==‘], [‘>‘], [‘>=‘] etc * [‘&‘], [‘|‘], [‘¡], [xor()] * [is.na()] * [between()], [near()]

Grouped tibbles

Because filtering expressions are computed within groups, they may yield different results on grouped tibbles. This will be the case as soon as an aggregating, lagging, or ranking function is involved. Compare this ungrouped filtering:

The former keeps rows with ‘mass‘ greater than the global average whereas the latter keeps rows with ‘mass‘ greater than the gender

average.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

[filter_all()], [filter_if()] and [filter_at()].

Other single table verbs: `arrange()`, `mutate()`, `rename()`, `summarise()`

Examples

```
# Learn more in ?dplyr_tidy_eval
```

`flybaseIDs` *flybaseIDs*

Description

flybaseIDs

Usage

```
flybaseIDs
```

Format

An object of class `character` of length 14599.

`full_join` *Full join datasets*

Description

Full join datasets

Usage

```
full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

`x` tbls to join. (See dplyr)

`y` tbls to join. (See dplyr)

`by` A character vector of variables to join by. (See dplyr)

`copy` If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)

`suffix` If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)

`...` Data frames to combine (See dplyr)

Value

A tt object

Examples

```
`%>%` = magrittr::`%>%`
annotation = tidybulk::counts %>% distinct(sample) %>% mutate(source = "AU")
tidybulk::counts %>% full_join(annotation)
```

`group_by` *Group by one or more variables*

Description

Most data operations are done on groups defined by variables. ‘group_by()‘ takes an existing tbl and converts it into a grouped tbl where operations are performed "by group". ‘ungroup()‘ removes grouping.

Usage

```
group_by(.data, ..., .add = FALSE, .drop = group_by_drop_default(.data))
```

Arguments

`.data` A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.

`...` In ‘group_by()‘, variables or computations to group by. In ‘ungroup()‘, variables to remove from the grouping.

`.add` When ‘FALSE‘, the default, ‘group_by()‘ will override existing groups. To add to the existing groups, use ‘.add = TRUE‘.

This argument was previously called ‘add‘, but that prevented creating a new grouping variable called ‘add‘, and conflicts with our naming conventions.

`.drop` When ‘.drop = TRUE‘, empty groups are dropped. See [group_by_drop_default()] for what the default value is for this argument.

Value

A [grouped data frame][grouped_df()], unless the combination of '...' and 'add' yields a non empty set of grouping columns, a regular (ungrouped) data frame otherwise.

Methods

These function are **generic**s, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

Methods available in currently loaded packages:

Examples

```
`%>%` = magrittr::`%>%`
by_cyl <- mtcars %>% group_by(cyl)
```

impute_abundance *Impute transcript abundance if missing from sample-transcript pairs*

Description

impute_abundance() takes as imput a 'tbl' formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a 'tbl' with an edditional adjusted abundance column. This method uses scaled counts if present.

Usage

```
impute_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)

## S4 method for signature 'spec_tbl_df'
impute_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)

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

## S4 method for signature 'tidybulk'
impute_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)

## S4 method for signature 'SummarizedExperiment'
impute_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)

## S4 method for signature 'RangedSummarizedExperiment'
impute_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL
)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.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_intrest + batch)
`.sample`	The name of the sample column
`.transcript`	The name of the transcript/gene column
`.abundance`	The name of the transcript/gene abundance column

Details

Maturing

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

Value

A ‘tbl‘ non-sparse abundance

A ‘tbl‘ with imputed abundnce

A ‘tbl‘ with imputed abundnce

A ‘tbl‘ with imputed abundnce

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
res =
impute_abundance(
tidybulk::counts_mini,
~ condition,
.sample = sample,
.transcript = transcript,
.abundance = count
)
```

inner_join *Inner join datasets*

Description

Inner join datasets

Usage

```
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tt object

Examples

```
`%>%` = magrittr::`%>%`
annotation = tidybulk::counts %>% distinct(sample) %>% mutate(source = "AU")
tidybulk::counts %>% inner_join(annotation)
```

keep_abundant *Keep abundant transcripts*

Description

keep_abundant() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a ‘tbl‘ with additional columns for the statistics from the hypothesis test.

Usage

```
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)

## S4 method for signature 'spec_tbl_df'
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)

## S4 method for signature 'tbl_df'
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)

## S4 method for signature 'tidybulk'
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)

## S4 method for signature 'SummarizedExperiment'
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)

## S4 method for signature 'RangedSummarizedExperiment'
keep_abundant(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.sample` The name of the sample column

`.transcript` The name of the transcript/gene column

`.abundance` The name of the transcript/gene abundance column

`factor_of_interest` The name of the column of the factor of interest. This is used for defining sample groups for the filtering process.

`minimum_counts` A real positive number. It is the threshold of count per million that is used to filter transcripts/genes out from the scaling procedure.

`minimum_proportion` A real positive number between 0 and 1. It is the threshold of proportion of samples for each transcripts/genes that have to be characterised by a cmp bigger than the threshold to be included for scaling procedure.

Details

Maturing

At the moment this function uses edgeR only, but other inference algorithms will be added in the near future.

Value

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

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

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

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

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
keep_abundant(
tidybulk::counts_mini,
    sample,
    transcript,
    `count`
)
```

`keep_variable` *Keep variable transcripts*

Description

keep_variable() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a ‘tbl‘ with additional columns for the statistics from the hypothesis test.

Usage

```
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)

## S4 method for signature 'spec_tbl_df'
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)
```

```
## S4 method for signature 'tbl_df'
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)

## S4 method for signature 'tidybulk'
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)

## S4 method for signature 'SummarizedExperiment'
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)

## S4 method for signature 'RangedSummarizedExperiment'
keep_variable(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  top = 500,
  log_transform = TRUE
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.sample` The name of the sample column

`.transcript` The name of the transcript/gene column

`.abundance` The name of the transcript/gene abundance column

`top` Integer. Number of top transcript to consider

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

Details

Maturing

At the moment this function uses edgeR only, but other inference algorithms will be added in the near future.

Value

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

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

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

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

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
keep_variable(
tidybulk::counts_mini,
    sample,
    transcript,
    `count`,
    top = 500
)
```

---

left_join *Left join datasets*

Description

Left join datasets

Usage

```
left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tt object

Examples

```
`%>%` = magrittr::`%>%`
annotation = tidybulk::counts %>% distinct(sample) %>% mutate(source = "AU")
tidybulk::counts %>% left_join(annotation)
```

mutate *Create, modify, and delete columns*

Description

'mutate()' adds new variables and preserves existing ones; 'transmute()' adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to 'NULL'.

Usage

```
mutate(.data, ...)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	<['tidy-eval'][dplyr_tidy_eval]> Name-value pairs. The name gives the name of the column in the output. The value can be: * A vector of length 1, which will be recycled to the correct length. * A vector the same length as the current group (or the whole data frame if ungrouped). * 'NULL', to remove the column. * A data frame or tibble, to create multiple columns in the output.

Value

An object of the same type as ‘.data‘.

For ‘mutate()‘:

* Rows are not affected. * Existing columns will be preserved unless explicitly modified. * New columns will be added to the right of existing columns. * Columns given value ‘NULL‘ will be removed * Groups will be recomputed if a grouping variable is mutated. * Data frame attributes are preserved.

For ‘transmute()‘:

* Rows are not affected. * Apart from grouping variables, existing columns will be remove unless explicitly kept. * Column order matches order of expressions. * Groups will be recomputed if a grouping variable is mutated. * Data frame attributes are preserved.

Useful mutate functions

* [‘+‘], [‘-‘], [log()], etc., for their usual mathematical meanings

* [lead()], [lag()]

* [dense_rank()], [min_rank()], [percent_rank()], [row_number()], [cume_dist()], [ntile()]

* [cumsum()], [cummean()], [cummin()], [cummax()], [cumany()], [cumall()]

* [na_if()], [coalesce()]

* [if_else()], [recode()], [case_when()]

Grouped tibbles

Because mutating expressions are computed within groups, they may yield different results on grouped tibbles. This will be the case as soon as an aggregating, lagging, or ranking function is involved. Compare this ungrouped mutate:

With the grouped equivalent:

The former normalises ‘mass‘ by the global average whereas the latter normalises by the averages within gender levels.

Methods

These function are **generic**s, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

Methods available in currently loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `rename()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
# Newly created variables are available immediately
mtcars %>% as_tibble() %>% mutate(
  cyl2 = cyl * 2,
  cyl4 = cyl2 * 2
)
```

nest *nest*

Description

nest

Usage

```
nest(.data, ...)

## Default S3 method:
nest(.data, ...)

## S3 method for class 'tidybulk'
nest(.data, ...)
```

Arguments

`.data`	A tbl. (See tidyr)
`...`	Name-variable pairs of the form new_col = c(col1, col2, col3) (See tidyr)

Value

A tt object

Examples

```
nest(tidybulk(tidybulk::counts_mini, sample, transcript, count), data = -transcript)
```

pivot_sample *Extract sample-wise information*

Description

pivot_sample() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <ENSEMBL_ID> | <COUNT> | <...> | and returns a ‘tbl‘ with only sample-related columns

Usage

```
pivot_sample(.data, .sample = NULL)

## S4 method for signature 'spec_tbl_df'
pivot_sample(.data, .sample = NULL)

## S4 method for signature 'tbl_df'
pivot_sample(.data, .sample = NULL)

## S4 method for signature 'tidybulk'
pivot_sample(.data, .sample = NULL)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.sample`	The name of the sample column

Details

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‘ object

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

Examples

```
pivot_sample(
tidybulk::counts_mini,
.sample = sample
)
```

`pivot_transcript` *Extract transcript-wise information*

Description

pivot_transcript() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <ENSEMBL_ID> | <COUNT> | <...> | and returns a ‘tbl‘ with only sample-related columns

Usage

```
pivot_transcript(.data, .transcript = NULL)

## S4 method for signature 'spec_tbl_df'
pivot_transcript(.data, .transcript = NULL)

## S4 method for signature 'tbl_df'
pivot_transcript(.data, .transcript = NULL)

## S4 method for signature 'tidybulk'
pivot_transcript(.data, .transcript = NULL)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.transcript` The name of the transcript column

Details

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‘ object

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

Examples

```
pivot_transcript(
tidybulk::counts_mini,
.transcript = transcript
)
```

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

Description

reduce_dimensions() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | 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,
  log_transform = TRUE,
  scale = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'spec_tbl_df'
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  log_transform = TRUE,
  scale = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tbl_df'
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  log_transform = TRUE,
  scale = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'tidybulk'
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  log_transform = TRUE,
  scale = TRUE,
  action = "add",
  ...
)

## S4 method for signature 'SummarizedExperiment'
```

```
reduce_dimensions(
  .data,
  .element = NULL,
  .feature = NULL,
  .abundance = NULL,
  method,
  .dims = 2,
  top = 500,
  of_samples = TRUE,
  log_transform = TRUE,
  scale = TRUE,
  action = "add",
  ...
)

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

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.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` A list of integer vectors corresponding to principal components of interest (e.g., list(1:2, 3:4, 5:6))

`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

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

`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.

`action` A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

`...` Further parameters passed to the function prcomp if you choose method="PCA" or Rtsne if you choose method="tSNE"

Details

Maturing

This function reduces the dimensions of the transcript abundances. It can use multi-dimensional scaling (MDS) of principal component analysis (PCA).

Value

A tbl object with additional columns for the reduced dimensions

A tbl object with additional columns for the reduced dimensions

A tbl object with additional columns for the reduced dimensions

A tbl object with additional columns for the reduced dimensions

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
counts.MDS =  reduce_dimensions(tidybulk::counts_mini, sample, transcript, count, method="MDS", .dims = 3)


counts.PCA =  reduce_dimensions(tidybulk::counts_mini, sample, transcript, count, method="PCA", .dims = 3)
```

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

Description

remove_redundancy() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | for correlation method or | <DIMENSION 1> | <DIMENSION 2> | <...> | for reduced_dimensions method, and returns a ‘tbl‘ 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,
  log_transform = FALSE,
  Dim_a_column,
  Dim_b_column
)

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

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

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

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.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 cluster algorithm to use, ay 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

`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

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

`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

Details

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 recundant 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.

Value

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

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

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

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

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
   remove_redundancy(
    tidybulk::counts_mini,
   .element = sample,
   .feature = transcript,
    .abundance =  count,
    method = "correlation"
    )

counts.MDS =  reduce_dimensions(tidybulk::counts_mini, sample, transcript, count, method="MDS", .dims = 3)

remove_redundancy(
counts.MDS,
Dim_a_column = `Dim1`,
Dim_b_column = `Dim2`,
.element = sample,
  method = "reduced_dimensions"
)
```

`rename` *Rename columns*

Description

Rename individual variables using ‘new_name = old_name‘ syntax.

Usage

```
rename(.data, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<[‘tidy-select‘][dplyr_tidy_select]> Use ‘new_name = old_name‘ to rename selected variables.

Value

An object of the same type as ‘.data‘. * Rows are not affected. * Column names are changed; column order is preserved * Data frame attributes are preserved. * Groups are updated to reflect new names.

Scoped selection and renaming

Use the three scoped variants ([rename_all()], [rename_if()], [rename_at()]) to renaming a set of variables with a function.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
iris <- as_tibble(iris) # so it prints a little nicer
rename(iris, petal_length = Petal.Length)
```

right_join *Right join datasets*

Description

Right join datasets

Usage

```
right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tt object

Examples

```
`%>%` = magrittr::`%>%`
annotation = tidybulk::counts %>% distinct(sample) %>% mutate(source = "AU")
tidybulk::counts %>% right_join(annotation)
```

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

Description

rotate_dimensions() takes as imput 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,
  action = "add"
)

## S4 method for signature 'spec_tbl_df'
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,
  action = "add"
)

## S4 method for signature 'tbl_df'
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,
  action = "add"
)

## S4 method for signature 'tidybulk'
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,
  action = "add"
)
```

```
## 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,
  action = "add"
)

## 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,
  action = "add"
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`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)

`action` A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

Details

Maturing

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

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 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 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 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

Examples

```
counts.MDS =  reduce_dimensions(tidybulk::counts_mini, sample, transcript, count, method="MDS", .dims = 3)

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

`rowwise` *Group input by rows*

Description

See [this repository](https://github.com/jennybc/row-oriented-workflows) for alternative ways to perform row-wise operations.

Usage

```
rowwise(.data)
```

Arguments

`.data`	Input data frame.

Details

‘rowwise()‘ is used for the results of [do()] when you create list-variables. It is also useful to support arbitrary complex operations that need to be applied to each row.

Currently, rowwise grouping only works with data frames. Its main impact is to allow you to work with list-variables in [summarise()] and [mutate()] without having to use `[[1]]`. This makes ‘summarise()‘ on a rowwise tbl effectively equivalent to [plyr::ldply()].

Value

A ‘tbl‘

A ‘tbl‘

Examples

```
`%>%` = magrittr::`%>%`
df <- expand.grid(x = 1:3, y = 3:1)
df_done <- df %>% rowwise() %>% do(i = seq(.$x, .$y))
```

scale_abundance *Scale the counts of transcripts/genes*

Description

scale_abundance() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | 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,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
)

## S4 method for signature 'spec_tbl_df'
scale_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
)

## S4 method for signature 'tbl_df'
scale_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
)

## S4 method for signature 'tidybulk'
scale_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
)

## S4 method for signature 'SummarizedExperiment'
scale_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
)

## S4 method for signature 'RangedSummarizedExperiment'
scale_abundance(
  .data,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  factor_of_interest = NULL,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  method = "TMM",
  reference_selection_function = median,
  action = "add"
```

```
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.sample` The name of the sample column

`.transcript` The name of the transcript/gene column

`.abundance` The name of the transcript/gene abundance column

`factor_of_interest` The name of the column of the factor of interest. This is used for identifying lowly abundant transcript, to be ignored for calculating scaling fators.

`minimum_counts` A real positive number. It is the threshold of count per million that is used to filter transcripts/genes out from the scaling procedure. The scaling inference is then applied back to all unfiltered data.

`minimum_proportion` A real positive number between 0 and 1. It is the threshold of proportion of samples for each transcripts/genes that have to be characterised by a cmp bigger than the threshold to be included for scaling procedure.

`method` A character string. The scaling method passed to the backend function (i.e., edgeR::calcNormFactors; "TMM","TMMwsp","RLE","upperquartile")

`reference_selection_function` A fucntion that is used to selecting the reference sample for scaling. It could be max (default), which choose the sample with maximum library size; or median, which chooses the sample with median library size.

`action` A character string between "add" (default) and "only". "add" joins the new information to the input tbl (default), "only" return a non-redundant tbl with the just new information.

Details

Maturing

Scales transcript abundance compansating 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.

Value

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

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

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

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

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
scale_abundance(tidybulk::counts_mini,  sample, transcript, `count`)
```

se *SummarizedExperiment*

Description

SummarizedExperiment

Usage

```
se
```

Format

An object of class `RangedSummarizedExperiment` with 100 rows and 8 columns.

se_mini *SummarizedExperiment mini for vignette*

Description

SummarizedExperiment mini for vignette

Usage

```
se_mini
```

Format

An object of class `SummarizedExperiment` with 527 rows and 5 columns.

summarise *Summarise each group to fewer rows*

Description

‘summarise()‘ creates a new data frame. It will have one (or more) rows for each combination of grouping variables; if there are no grouping variables, the output will have a single row summarising all observations in the input. It will contain one column for each grouping variable and one column for each of the summary statistics that you have specified.

‘summarise()‘ and ‘summarize()‘ are synonyms.

Usage

```
summarise(.data, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<[‘tidy-eval‘][dplyr_tidy_eval]> Name-value pairs of summary functions. The name will be the name of the variable in the result. The value can be: * A vector of length 1, e.g. ‘min(x)‘, ‘n()‘, or ‘sum(is.na(y))‘. * A vector of length ‘n‘, e.g. ‘quantile()‘. * A data frame, to add multiple columns from a single expression.

Value

An object _usually_ of the same type as ‘.data‘.

* The rows come from the underlying ‘group_keys()‘. * The columns are a combination of the grouping keys and the summary expressions that you provide. * If ‘x‘ is grouped by more than one variable, the output will be another [grouped_df] with the right-most group removed. * If ‘x‘ is grouped by one variable, or is not grouped, the output will be a [tibble]. * Data frame attributes are **not** preserved, because ‘summarise()‘ fundamentally creates a new data frame.

Useful functions

* Center: [mean()], [median()] * Spread: [sd()], [IQR()], [mad()] * Range: [min()], [max()], [quantile()] * Position: [first()], [last()], [nth()], * Count: [n()], [n_distinct()] * Logical: [any()], [all()]

Backend variations

The data frame backend supports creating a variable and using it in the same summary. This means that previously created summary variables can be further transformed or combined within the summary, as in [mutate()]. However, it also means that summary variables with the same names as previous variables overwrite them, making those variables unavailable to later summary variables.

This behaviour may not be supported in other backends. To avoid unexpected results, consider using new names for your summary variables, especially when creating multiple summaries.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `rename()`

Examples

```
`%>%` = magrittr::`%>%`
# A summary applied to ungrouped tbl returns a single row
mtcars %>%
  summarise(mean = mean(disp))
```

`symbol_to_entrez` *Get ENTREZ id from gene SYMBOL*

Description

Get ENTREZ id from gene SYMBOL

Usage

```
symbol_to_entrez(.data, .transcript = NULL, .sample = NULL)
```

Arguments

`.data`	A tt or tbl object.
`.transcript`	A character. The name of the ene symbol column.
`.sample`	The name of the sample column

Value

A tbl

Examples

```
symbol_to_entrez(tidybulk::counts_mini, .transcript = transcript, .sample = sample)
```

test_differential_abundance *Add differential transcription information to a tbl using edgeR.*

Description

test_differential_abundance() takes as imput a 'tbl' formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> | and returns a 'tbl' with additional columns for the statistics from the hypothesis test.

Usage

```
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)

## S4 method for signature 'spec_tbl_df'
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)

## S4 method for signature 'tbl_df'
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)

## S4 method for signature 'tidybulk'
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)

## S4 method for signature 'SummarizedExperiment'
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)

## S4 method for signature 'RangedSummarizedExperiment'
test_differential_abundance(
  .data,
  .formula,
  .sample = NULL,
  .transcript = NULL,
  .abundance = NULL,
  .contrasts = NULL,
  method = "edgeR_quasi_likelihood",
  significance_threshold = 0.05,
  minimum_counts = 10,
  minimum_proportion = 0.7,
  fill_missing_values = FALSE,
  scaling_method = "TMM",
  omit_contrast_in_colnames = FALSE,
  action = "add"
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.formula` A formula with no response variable, representing the desired linear model

`.sample` The name of the sample column

`.transcript` The name of the transcript/gene column

`.abundance` The name of the transcript/gene abundance column

`.contrasts` A character vector. See edgeR makeContrasts specification for the parameter ‘contrasts‘. If contrasts are not present the first covariate is the one the model is tested against (e.g., ~ factor_of_interest)

`method` A string character. Either "edgeR_quasi_likelihood" (i.e., QLF), "edgeR_likelihood_ratio" (i.e., LRT)

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

`minimum_counts` A real positive number. It is the threshold of count per million that is used to filter transcripts/genes out from the scaling procedure.

`minimum_proportion`
A real positive number between 0 and 1. It is the threshold of proportion of samples for each transcripts/genes that have to be characterised by a cmp bigger than the threshold to be included for scaling procedure.

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

`scaling_method` A character string. The scaling method passed to the backend function (i.e., edgeR::calcNormFactors; "TMM","TMMwsp","RLE","upperquartile")

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

`action` A character string. Whether to join the new information to the input tbl (add), or just get the non-redundant tbl with the new information (get).

Details

Maturing

At the moment this function uses edgeR only, but other inference algorithms will be added in the near future.

Value

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

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

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

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

A ‘SummarizedExperiment‘ object

A ‘SummarizedExperiment‘ object

Examples

```
test_differential_abundance(
 tidybulk::counts_mini,
    ~ condition,
    sample,
    transcript,
    `count`
)

# The functon `test_differential_abundance` operated with contrasts too

 test_differential_abundance(
    tidybulk::counts_mini,
    ~ 0 + condition,
    sample,
    transcript,
    `count`,
    .contrasts = c( "conditionTRUE - conditionFALSE")
 )
```

`test_gene_enrichment` *analyse gene enrichment with EGSEA*

Description

test_gene_enrichment() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <ENSEMBL_ID> | <COUNT> | <...> | and returns a ‘tbl‘ with the additional transcript symbol column

Usage

```
test_gene_enrichment(
  .data,
  .formula,
  .sample = NULL,
  .entrez,
  .abundance = NULL,
  .contrasts = NULL,
  species,
  cores = 10
)

## S4 method for signature 'spec_tbl_df'
test_gene_enrichment(
  .data,
  .formula,
  .sample = NULL,
  .entrez,
  .abundance = NULL,
  .contrasts = NULL,
  species,
  cores = 10
)

## S4 method for signature 'tbl_df'
test_gene_enrichment(
  .data,
  .formula,
  .sample = NULL,
  .entrez,
  .abundance = NULL,
  .contrasts = NULL,
  species,
  cores = 10
)

## S4 method for signature 'tidybulk'
test_gene_enrichment(
  .data,
  .formula,
  .sample = NULL,
  .entrez,
  .abundance = NULL,
  .contrasts = NULL,
  species,
  cores = 10
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.formula` A formula with no response variable, representing the desired linear model

`.sample`	The name of the sample column
`.entrez`	The ENTREZ ID of the transcripts/genes
`.abundance`	The name of the transcript/gene abundance column
`.contrasts`	= NULL,
`species`	A character. For example, human or mouse
`cores`	An integer. The number of cores available

Details

Maturing

This wrapper execute gene enrichment analyses of the dataset

Value

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

Examples

```
## Not run:

df_entrez = symbol_to_entrez(tidybulk::counts_mini, .transcript = transcript, .sample = sample)
df_entrez = aggregate_duplicates(df_entrez, aggregation_function = sum, .sample = sample, .transcript = entrez

library("EGSEA")

test_gene_enrichment(
df_entrez,
~ condition,
.sample = sample,
.entrez = entrez,
.abundance = count,
species="human",
cores = 1
)


## End(Not run)
```

`test_gene_overrepresentation`

analyse gene over-representation with GSEA

Description

test_gene_overrepresentation() takes as imput a ‘tbl‘ formatted as | <SAMPLE> | <ENSEMBL_ID> | <COUNT> | <...> | and returns a ‘tbl‘ with the GSEA statistics

Usage

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

## S4 method for signature 'spec_tbl_df'
test_gene_overrepresentation(.data, .sample = NULL, .entrez, .do_test, species)

## S4 method for signature 'tbl_df'
test_gene_overrepresentation(.data, .sample = NULL, .entrez, .do_test, species)

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

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.sample` The name of the sample column

`.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\")

Details

Maturing

This wrapper execute gene enrichment analyses of the dataset using a list of transcripts and GSEA. This wrapper uses clusterProfiler on the backend.

Value

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

A ‘tbl‘ object

Examples

```
df_entrez = symbol_to_entrez(tidybulk::counts_mini, .transcript = transcript, .sample = sample)
df_entrez = aggregate_duplicates(df_entrez, aggregation_function = sum, .sample = sample, .transcript = entrez
df_entrez = mutate(df_entrez, do_test = transcript %in% c("TNFRSF4", "PLCH2", "PADI4", "PAX7"))

test_gene_overrepresentation(
df_entrez,
.sample = sample,
.entrez = entrez,
.do_test = do_test,
species="Homo sapiens"
)
```

tidybulk *Creates a ‘tt‘ object from a ‘tbl“*

Description

tidybulk() creates a ‘tt‘ object from a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

Usage

```
tidybulk(.data, .sample, .transcript, .abundance, .abundance_scaled = NULL)

## S4 method for signature 'spec_tbl_df'
tidybulk(.data, .sample, .transcript, .abundance, .abundance_scaled = NULL)

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

## S4 method for signature 'SummarizedExperiment'
tidybulk(.data, .sample, .transcript, .abundance, .abundance_scaled = NULL)

## S4 method for signature 'RangedSummarizedExperiment'
tidybulk(.data, .sample, .transcript, .abundance, .abundance_scaled = NULL)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.sample`	The name of the sample column
`.transcript`	The name of the transcript/gene column
`.abundance`	The name of the transcript/gene abundance column
`.abundance_scaled`	The name of the transcript/gene scaled abundance column

Details

Maturing

This function created a tidybulk object and is useful if you want to avoid to specify .sample, .transcript and .abundance arguments all the times. The tidybulk object have an attribute called internals where these three arguments are stored as metadata. They can be extracted as attr(<object>, "internals").

Value

A ‘tidybulk‘ object

A ‘tidybulk‘ object

A ‘tidybulk‘ object

A ‘tidybulk‘ object

A ‘tidybulk‘ object

Examples

```
my_tt =  tidybulk(tidybulk::counts_mini, sample, transcript, count)
```

---

tidybulk_SAM_BAM *Creates a ‘tt‘ object from a list of file names of BAM/SAM*

Description

tidybulk_SAM_BAM() creates a ‘tt‘ object from a ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

Usage

```
tidybulk_SAM_BAM(file_names, genome = "hg38", ...)

## S4 method for signature 'character,character'
tidybulk_SAM_BAM(file_names, genome = "hg38", ...)
```

Arguments

`file_names`	A character vector
`genome`	A character string
`...`	Further parameters passed to the function Rsubread::featureCounts

Details

Maturing

This function is based on FeatureCounts package. This function created a tidybulk object and is useful if you want to avoid to specify .sample, .transcript and .abundance arguments all the times. The tidybulk object have an attribute called internals where these three arguments are stored as metadata. They can be extracted as attr(<object>, "internals").

Value

A ‘tidybulk‘ object

A ‘tidybulk‘ object

`X_cibersort` *Cibersort reference*

Description

Cibersort reference

Usage

```
X_cibersort
```

Format

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

Index