Package 'mirmisc'

October 26, 2021

Title Non-Modeling Helper Functions For Mirvie R Coders

Version 0.3.0

Maintainer Rory Nolan <rory@mirvie.com>

Description We have random bits of code that are quite useful. This is a home for it.

License file LICENSE

URL https://gitlab.com/Mirvie/mirmisc

BugReports https://gitlab.com/Mirvie/mirmisc/-/issues

Imports arrow (>= 3.0),
checkmate,
DescTools,
detrendr (>= 0.6.12),
dplyr (>= 1.0.0),
forcats,
foreach,
fs,
future,
ggplot2,
ggpmisc,
ggstatsplot,
ggthemes,
glue,
janitor,
magrittr,
methods,
plotly,
png,
pROC,
purrr,
qualV,
readr,
rlang,
rrcov,
rsample,
scales,
strex,
stringr,

tidyr,
utils,
zeallot,
zoo

Suggests datasets,
embed,
knitr,
mirmodels,
mockery,
patchwork,
recipes,
rmarkdown,
spelling,
testthat (>= 3.0),
tidyverse,
withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

R topics documented:

`autoplot.mirvie_cohort_outliers` *Plot a* `mirvie_cohort_outliers` *object.*

Description

A `mirvie_cohort_outliers` object is the output of a call to `get_cohort_outliers()`.

Usage

```
## S3 method for class 'mirvie_cohort_outliers'
autoplot(object, pcx = 1, pcy = 2, plotly = interactive(), ...)
```

Arguments

`object`	A `mirvie_cohort_outliers` object.
`pcx`	An integer between 1 and 5. The principal component that will be on the x axis.
`pcy`	An integer between 1 and 5. The principal component that will be on the y axis.
`plotly`	A flag. Make the plot interactive (with mirvie ID tooltips)?
`...`	Not currently used.

Value

A `ggplot2::ggplot()` or a `plotly::ggplotly()`.

`check_sample_descs` *Check the sample descriptions data frame for an R&D experiment.*

Description

- It must contain a column called `sample_name` with unique values.
- It must contain a column called `condition`.
- If it contains a column called `id_tech_rep`, then for a given `id_tech_rep`, the `sample_names` must have a common substring at least two characters long.

Usage

```
check_sample_descs(sample_descs)
```

Arguments

sample_descs	A data frame of sample descriptions that includes columns called sample_name and condition.

Value

TRUE (invisibly) if the check passes. Otherwise, an error is thrown.

See Also

Other Arkady: collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

---

collapse_tech_reps *Collapse (sum) technical replicates in a set of R&D experiments.*

Description

Technical replicates, identified by the id_tech_rep column, are summed into a single sample. The resulting id_genes_counts is the longest common substring of the id_genes_counts of the input technical replicates.

Usage

```
collapse_tech_reps(prepped_rd_input)
```

Arguments

prepped_rd_input	A data frame. The output of a call to prep_rd_input().

Value

A dataframe with columns id_genes_counts, condition and the gene columns.

See Also

Other Arkady: check_sample_descs(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

`collect_counts` *Collect the count files of several samples into a single data frame.*

Description

This function takes a directory path `dir_path` and searches that directory for files whose names end in '_counts.txt'. It reads those files and concatenates them. Each file is assumed to correspond to a single sample whose name is contained in the first part of the file name (the bit before '_counts.txt'). These sample names are used as column names in the output data frame.

Usage

```
collect_counts(
  dir_path,
  convert_genenames = TRUE,
  cpm = FALSE,
  log2 = FALSE,
  remove_ercc = TRUE,
  remove_rp_4_11 = TRUE,
  remove_metadata = TRUE,
  remove_controls = TRUE,
  write = FALSE
)
```

Arguments

`dir_path`	A character vector. The path to the directory containing '*_counts.txt' files. To specify several directories, use a list of paths.
`convert_genenames`	A flag. Convert gene names from Ensembl IDs to more widely-used names?
`cpm`	A flag. Convert raw counts to counts-per-million (on a per sample basis)?
`log2`	A flag. Transform the counts using `log2(x + 1)`?
`remove_ercc`	A flag. Remove ERCC counts from the results?
`remove_rp_4_11`	A flag. RP4 and RP11 genes come from a particular donor when building the genome and are mostly not useful. The default (`TRUE`) is to remove them.
`remove_metadata`	A flag. The count files can contain non-gene metadata (e.g. mapping stats). The default (`TRUE`) is to remove them.
`remove_controls`	A flag. The count files can contain positive and negative controls (denoted by having names ending in 'PC_counts.txt' or 'NC_counts.txt'). The default (`TRUE`) is to remove them.
`write`	A flag or string. Write the results to disk as a tab-separated file? If `TRUE`, the file will be written to the working directory with name 'genes_counts.txt', 'genes_cpm.txt', genes_log2.txt' or 'genes_log2_cpm.txt'. To write the file elsewhere, pass the path through this argument as a string.

Value

A data frame object.

Examples

```
## Not run:
collect_counts("path/to/dir/with/count/files")

## End(Not run)
```

`convert_feather_dir_to_csvs` *Make a directory of CSVs from a directory of feathers.*

Description

Take a directory containing feather files and create a sibling directory with corresponding CSVs.

Usage

```
convert_feather_dir_to_csvs(feather_dir_path, new_dir_name = "feather-csvs")
```

Arguments

`feather_dir_path`	The path to the directory containing the feathers.
`new_dir_name`	The name of the new directory. This should *not* be an absolute path and rather just a name; I.e. it should not contain '/'. The created directory will be a sibling of the one at `feather_dir_path`.

Value

The path to the output directory, invisibly.

`convert_gene_names` *Convert gene names to/from Ensembl.*

Description

This function works in a particular way: inputs that don't look like a gene at all are returned as is.

Usage

```
convert_gene_names(x, ensembl = "from")
```

Arguments

`x`	A character vector.
`ensembl`	A string. Either `"from"` or `"to"`. With `"to"` the result is Ensembl gene names. With `"from"` the result is colloquial gene names.

Value

A character vector.

deseq_contrasts *Obtain pairwise contrasts with shrunken log2FC from a DESeq object.*

Description

Obtain pairwise contrasts with shrunken log2FC from a DESeq object.

Usage

```
deseq_contrasts(ddsx, compr, alpha = 0.05, shrink = TRUE)
```

Arguments

ddsx	A DESeq object, output from deseq_wide().
compr	A character vector for pairwise comparisons of interest, eg., c("4","1") from draw 4 with samples from draw 1.
alpha	A number. The significance level.
shrink	A flag. DESeq2::lfcShrink() the result? Default yes.

Value

A dataframe.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

Examples

```
rs_genes <- arrow::read_feather(
  system.file("extdata", "rs_genes_draw_3_4.feather", package = "mirmisc")
)
rs_meta <- arrow::read_feather(
  system.file("extdata", "rs_meta_draw_3_4.feather", package = "mirmisc")
)
ddsx <- deseq_wide(rs_genes, rs_meta, compvar = "meta_draw")
deseq_contrasts(ddsx, compr = c("4", "3"))
```

deseq_pairs *Get the mirmodels::deseq() tables from pairs of conditions.*

Description

Technical replicates in genes_counts should be collapsed with collapse_tech_reps() prior to running this function.

Usage

```
deseq_pairs(prepped_rd_input)
```

Arguments

prepped_rd_input	A data frame. The output of a call to prep_rd_input().

Value

A dataframe. A series of mirmodels::deseq() tables bound together with dplyr::bind_rows(). There are two extra columns cond1 and cond2 to record the conditions being compared.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

deseq_wide *Obtain DESeq object from raw counts wide dataframe.*

Description

Wide is samples in columns, genes in rows.

Usage

```
deseq_wide(data, design, compvar)
```

Arguments

data	A wide datafame with raw counts that contains raw counts for samples of interest (may contain other samples as well, which will be filtered out).
design	A dataframe with all of the variables specified in compvar as column names. Must have the same number of rows as data with samples in the same order.
compvar	• a character vector with the name(s) of the variable(s), which must be identical to the variable name(s) in the design dataframe.

Value

A DESeq2 object.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

Examples

```
rs_genes <- arrow::read_feather(
  system.file("extdata", "rs_genes_draw_3_4.feather", package = "mirmisc")
)
rs_meta <- arrow::read_feather(
  system.file("extdata", "rs_meta_draw_3_4.feather", package = "mirmisc")
)
deseq_wide(rs_genes, rs_meta, compvar = "meta_draw")
```

---

df_fill_missing_genes *Fill missing gene columns in one data frame with those from another.*

Description

The 'gene names' are those returned by get_gene_names(). This function takes two data frames df and df_fill_from and, if there are any columns in df_fill_from with gene names as column names which don't exist in df, they are copied into df.

Usage

```
df_fill_missing_genes(df, df_fill_from)
```

Arguments

df	A data frame.
df_fill_from	A data frame with the same number of rows as df.

Details

If the gene names in df are contiguously located, the copied genes are inserted right after those. Otherwise, they are inserted on the end.

Value

A data frame.

Examples

```
if (require("mirmodels")) {
  st_data <- get_st_data()
  st_data_median0 <- get_st_data(gene_predicate = ~ median(.) == 0)
  dim(st_data)
  dim(st_data_median0)
  dim(df_fill_missing_genes(st_data_median0, st_data))
}
```

---

downsample_count_vec *Downsample a vector of counts.*

Description

Uniformly downsample a vector of non-negative integers to have a specified sum. That is, keep randomly subtracting 1 from nonzero elements of the vector until it has the desired sum. Each count is equally likely to be taken. That is, an element with value 8 is 4 times more likely to be decremented than an element with value 2.

Usage

```
downsample_count_vec(vec, end_sum)

downsample_count_mat_rows(mat, end_sum)

downsample_count_mat_cols(mat, end_sum)

downsample_gene_counts(df, end_sum)
```

Arguments

vec	A vector of non-negative integers.
end_sum	The number that you would like vector to sum to after downsampling. This must be less than the initial sum.
mat	A matrix of non-negative integers.
df	A data frame with gene names as columns.

Details

`downsample_count_mat_rows()` and `downsample_count_mat_cols()` just do `downsample_vec()` to all rows and columns of a matrix using `apply()`.

`downsample_gene_counts()` downsamples on the subset of columns in the data frame that have names in `get_gene_names()`.

If `end_sum > sum(vec)`, `vec` is returned unchanged.

Value

A vector of non-negative integers.

Examples

```
downsample_count_vec(1:24, 24)
mat <- matrix(sample.int(100, size = 6^2, replace = TRUE), nrow = 6)
downsample_count_mat_rows(mat, end_sum = 6)
downsample_count_mat_cols(mat, end_sum = 6)
if (rlang::is_installed("mirmodels")) {
  ms_data <- mirmodels::get_ms_data(gene_predicate = ~ median(.) > 0)
  downsampled_ms <- downsample_gene_counts(ms_data, end_sum = 1e6)
}
```

`fgsea_basic` *Get results from fast Gene Set Enrichment Analysis (GSEA)*

Description

Uses `fgsea::fgsea()` defaults.

Usage

```
fgsea_basic(prepped_gsea_input_lfc, prepped_gsea_input_gene_set)
```

Arguments

`prepped_gsea_input_lfc`	Output of `prep_gsea_input_lfc()`.
`prepped_gsea_input_gene_set`	Output of `prep_gsea_input_gene_set()`.

Value

A dataframe.

See Also

Other Arkady: `check_sample_descs()`, `collapse_tech_reps()`, `deseq_contrasts()`, `deseq_pairs()`, `deseq_wide()`, `plot_cond_paired_pearson()`, `plot_down()`, `plot_metrics_global()`, `plot_var_mean_ratio()`, `prep_gsea_input_gene_set()`, `prep_gsea_input_lfc()`, `prep_rd_input()`, `reconcile_names()`

Examples

```
rs_genes <- arrow::read_feather(
  system.file("extdata", "rs_genes_draw_3_4.feather", package = "mirmisc")
)
rs_meta <- arrow::read_feather(
  system.file("extdata", "rs_meta_draw_3_4.feather", package = "mirmisc")
)
ddsx <- deseq_wide(rs_genes, rs_meta, compvar = "meta_draw")
dsq_contrasts <- deseq_contrasts(ddsx, compr = c("4", "3"))
prepped_gsea_input_lfc <- prep_gsea_input_lfc(dsq_contrasts)
prepped_gsea_input_gene_set <- prep_gsea_input_gene_set("C8")
fgsea_basic(prepped_gsea_input_lfc, prepped_gsea_input_gene_set)
```

filter_genes *Filter genes using a predicate function.*

Description

Gene names are elements of `get_gene_names()`. Genes for whom the predicate function `f()` evaluates to `FALSE` are dropped.

Usage

```
filter_genes(df, f)
```

Arguments

`df`	A data frame some of whose column names are gene names.
`f`	A predicate function or a formula coercible to a function by `rlang::as_function()`.

Value

A data frame.

get_cohort_outliers *Detect the outlying samples in a cohort.*

Description

This function wraps `mirmodels::compute_pcas()` and hence uses `rrcov::PcaGrid()` to do robust PCA analysis and detect outliers.

Usage

```
get_cohort_outliers(cohort)
```

Arguments

`cohort`	A two-character string, e.g. `"BW"`.

Details

Prior to PCA calculation (and outlier detection), a call to `mirmodels::linear_correct()` is made to regress away the effect of the total number of counts on gene expression levels, with care taken to not regress away the effect of gestational age.

There's an Easter egg. You can pass a data frame directly as the `cohort` argument and then the function will use that rather than having to call get_*_data() to get the data. I advise get_*_data(log2 = TRUE, tot_counts = TR
dian(.) > 0).

Value

An object of class `mirvie_cohort_outliers`. This is a data frame with 5 principal components named `PC1`, `PC2`, . . ., `PC5`. It also has columns `meta_collectionga`, `mirvie_id` and `outlier` which is a boolean column where `TRUE` indicates an outlier. This object has attributes `var_exp` and `loadings`. Read the documentation of `mirmodels::compute_pcas()` for more on those.

See Also

`autoplot.mirvie_cohort_outliers()`

Examples

```
if (require("mirmodels")) {
  ga_outliers <- get_cohort_outliers("ga")
  autoplot(ga_outliers)
}
```

---

get_df_gene_names *Which gene names are also column names?*

Description

This is just `intersect(names(df),get_gene_names())`.

Usage

```
get_df_gene_names(df)
```

Arguments

`df`	A data frame.

Value

A character vector.

---

get_feather_path *Get the path to the folder containing the feather files.*

Description

This function requires you to have set the environment variable `MIRVIE_FEATHER_PATH`, which you can do in the ~/.Rprofile file. It should have a line like `Sys.setenv(MIRVIE_FEATHER_PATH = "path/to/mirvie/feathers/dir")`. If the file doesn't exist, create it and make this the only line in the file. If this is done correctly, this function then forms a path with `MIRVIE_FEATHER_PATH` as the root directory.

Usage

```
get_feather_path(..., use_dotenv = TRUE, verify = TRUE)
```

Arguments

...	Character vectors. Elements of the path. Mostly, you'll leave this blank.
use_dotenv	A flag. If the MIRVIE_FEATHER_PATH environment variable isn't found by the usual R means, check the ~/.env file used by python-dotenv.
verify	A flag. Check that MIRVIE_FEATHER_PATH exists and contains at least one *.feather file. Error if check fails. Default TRUE.

Details

There's a whole vignette explaining this function. To find it, run browseVignettes(package = "mirmisc").

Value

An fs::path.

Examples

```
## Not run:
get_feather_path()

## End(Not run)
```

---

get_gene_names *Get the names of all of the genes that we use.*

Description

This includes the Ensembl and colloquial names (so in that sense there is duplication).

Usage

```
get_gene_names()
```

Value

A character vector.

Examples

```
get_gene_names()
```

get_htseq_paths *Get the paths to* htseq/ *directories for a given cohort.*

Description

The *_counts.txt files live in directories called htseq/. This function helps you to find all such directories for a given cohort.

Usage

```
get_htseq_paths(base_dir = "/mnt/storage/Cohorts", cohort_code)
```

Arguments

base_dir	A string. The path to a directory that the cohort directories live under. The cohort directories have name structure ###_XY where # is a digit and XY is the cohort code. For example, 007_RS.
cohort_code	A string with exactly two characters. E.g. "RS".

Value

A character vector of paths.

Examples

```
## Not run:
get_htseq_paths(cohort_code = "RS")

## End(Not run)
```

mutate_genes *Apply the same function to all columns whose names are gene names.*

Description

Gene names are elements of `get_gene_names()`.

Usage

```
mutate_genes(df, f)
```

Arguments

df	A data frame.
f	A function or a formula coercible to a function by `rlang::as_function()`.

Value

A data frame.

`plot_cond_paired_pearson` *Violin plots for within-condition gene Pearson correlation coefficients.*

Description

Violin plots for within-condition gene Pearson correlation coefficients.

Usage

```
plot_cond_paired_pearson(prepped_rd_input, gene_subset = NULL, qntl = NULL)
```

Arguments

`prepped_rd_input`	A data frame. The output of a call to `prep_rd_input()`.
`gene_subset`	A character vector of genes in a subset of interest.
`qntl`	A number between 0 and 1. A quantile-based threshold below which the data is subset. The purpose of this is to improve the sensitivity of the reproducibility analysis by focusing on subsets of data with lower counts.

Value

A ggplot.

See Also

Other Arkady: `check_sample_descs()`, `collapse_tech_reps()`, `deseq_contrasts()`, `deseq_pairs()`, `deseq_wide()`, `fgsea_basic()`, `plot_down()`, `plot_metrics_global()`, `plot_var_mean_ratio()`, `prep_gsea_input_gene_set()`, `prep_gsea_input_lfc()`, `prep_rd_input()`, `reconcile_names()`

`plot_down` *Plot top* `n` *most significantly enriched pathways in the upward or downward direction.*

Description

Plot top `n` most significantly enriched pathways in the upward or downward direction.

Usage

```
plot_down(fgsea_out, padj_thresh = 0.05, n = 20)

plot_up(fgsea_out, padj_thresh = 0.05, n = 20)
```

Arguments

`fgsea_out`	Output of `fgsea::fgsea()` or `fgsea_basic()`.
`padj_thresh`	Adjusted p-value threshold.
`size_thresh`	number of genes to show.

Value

A ggplot2::ggplot().

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

Examples

```
rs_genes <- arrow::read_feather(
  system.file("extdata", "rs_genes_draw_3_4.feather", package = "mirmisc")
)
rs_meta <- arrow::read_feather(
  system.file("extdata", "rs_meta_draw_3_4.feather", package = "mirmisc")
)
ddsx <- deseq_wide(rs_genes, rs_meta, compvar = "meta_draw")
dsq_contrasts <- deseq_contrasts(ddsx, compr = c("4", "3"))
prepped_gsea_input_lfc <- prep_gsea_input_lfc(dsq_contrasts)
prepped_gsea_input_gene_set <- prep_gsea_input_gene_set("C8")
fgs <- fgsea_basic(prepped_gsea_input_lfc, prepped_gsea_input_gene_set)
plot_down(fgs)
plot_up(fgs)
```

---

plot_metrics_global *Plots comparisons between conditions in an experiment.*

Description

Actual experimental samples and positive controls, using all genes in the count table or a subset of genes.

Usage

```
plot_metrics_global(prepped_rd_input, gene_subset = NULL)
```

Arguments

prepped_rd_input	A data frame. The output of a call to prep_rd_input().
gene_subset	A character vector of genes in a subset of interest.

Value

A list of 6 ggplots.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

plot_var_mean_ratio *Plot median and other quantile values for variance-to-mean ratios obtained across all genes.*

Description

Useful in comparison to a reference: if median and other quantile ratios are higher or lower than in the reference, then one can draw a conclusion about a relative variability among the samples in a given experimental condition.

Usage

```
plot_var_mean_ratio(prepped_rd_input)
```

Arguments

prepped_rd_input	A data frame. The output of a call to prep_rd_input().

Value

A ggplot.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), prep_gsea_input_gene_set(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

prep_gsea_input_gene_set *Prepare gene set input for fgsea::fgsea().*

Description

Categories of interest from http://www.gsea-msigdb.org/gsea/msigdb/index.jsp.

Usage

```
prep_gsea_input_gene_set(category)
```

Arguments

cetegory	A string. Category of interest.

Value

A list object for fgsea.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_lfc(), prep_rd_input(), reconcile_names()

Examples

```
prep_gsea_input_gene_set("C8")
```

---

prep_gsea_input_lfc *Prep ranking metric gene input for* fgsea::fgsea().

Description

Prep ranking metric gene input for fgsea::fgsea().

Usage

```
prep_gsea_input_lfc(x)
```

Arguments

x	The dataframe with the contrasts from DESeq, results generated by deseq_contrasts().

Value

A data frame.

See Also

Other Arkady: check_sample_descs(), collapse_tech_reps(), deseq_contrasts(), deseq_pairs(), deseq_wide(), fgsea_basic(), plot_cond_paired_pearson(), plot_down(), plot_metrics_global(), plot_var_mean_ratio(), prep_gsea_input_gene_set(), prep_rd_input(), reconcile_names()

Examples

```
rs_genes <- arrow::read_feather(
  system.file("extdata", "rs_genes_draw_3_4.feather", package = "mirmisc")
)
rs_meta <- arrow::read_feather(
  system.file("extdata", "rs_meta_draw_3_4.feather", package = "mirmisc")
)
ddsx <- deseq_wide(rs_genes, rs_meta, compvar = "meta_draw")
dsq_contrasts <- deseq_contrasts(ddsx, compr = c("4", "3"))
prep_gsea_input_lfc(dsq_contrasts)
```

prep_rd_input — *Join the sample descriptions and genes counts data frames into a wide format.*

Description

This function calls `reconcile_names()` and `check_sample_descs()` internally.

Usage

```
prep_rd_input(sample_descs, genes_counts)
```

Arguments

`sample_descs`	A data frame of sample descriptions. The `sampleName` column holds the sample names.
`genes_counts`	A data frame of gene counts. The first column is `gene` and the rest are sample names.

Value

A data frame. The gene counts table with columns corresponding to the desired subset of data.

See Also

Other Arkady: `check_sample_descs()`, `collapse_tech_reps()`, `deseq_contrasts()`, `deseq_pairs()`, `deseq_wide()`, `fgsea_basic()`, `plot_cond_paired_pearson()`, `plot_down()`, `plot_metrics_global()`, `plot_var_mean_ratio()`, `prep_gsea_input_gene_set()`, `prep_gsea_input_lfc()`, `reconcile_names()`

reconcile_names — *Match sample description sheet names to count file names.*

Description

Given a data frame of sample descriptions from the lab team and a data frame from a `genes_counts.txt` file from a sequencing run, match the sample names and return a data frame of sample descriptions detailing the matching.

Usage

```
reconcile_names(sample_descs, genes_counts)
```

Arguments

`sample_descs`	A data frame of sample descriptions. The `sampleName` column holds the sample names.
`genes_counts`	A data frame of gene counts. The first column is `gene` and the rest are sample names.

Details

Name matching is done by first replacing all whitespace and underscores with hyphens and then asserting that the `sampleName` must a substring of the column name. The matching is done via a pair of columns `id_sample_descs` and `id_genes_counts` in the output.

Value

A data frame.

See Also

Other Arkady: `check_sample_descs()`, `collapse_tech_reps()`, `deseq_contrasts()`, `deseq_pairs()`, `deseq_wide()`, `fgsea_basic()`, `plot_cond_paired_pearson()`, `plot_down()`, `plot_metrics_global()`, `plot_var_mean_ratio()`, `prep_gsea_input_gene_set()`, `prep_gsea_input_lfc()`, `prep_rd_input()`

`repsim_gene_cond` *Repeatedly simulate gene counts for cases and controls of a condition.*

Description

Given a number of gene repetitions, a number of samples, the condition prevalence and the theoretical mean of a gene for cases and controls, repeatedly simulate gene counts for that gene across the condition.

Usage

```
repsim_gene_cond(
  n_gene_reps,
  n_samples,
  cond_prevalence,
  control_mean,
  case_mean
)
```

Arguments

`n_gene_reps`	The number of times to repeatedly simulate the gene counts for that gene.
`n_samples`	The number of samples in the simulation.
`cond_prevalence`	A number in (0, 1). The condition prevalence.
`control_mean`	Theoretical mean of the gene for controls.
`case_mean`	Theoretical mean of the gene for cases.

Details

Poisson gene counts are assumed.

Value

A tibble with `n_gene_reps` + 1 columns called `generep_1`, `generep_2`, . . ., generep_{n_gene_reps}, `cond` and `n_samples` columns.

Examples

```
if (rlang::is_installed("mirmodels")) {
  rsgc5000 <- repsim_gene_cond(15000, 5000, 1 / 10, 0.01, 0.05)
  sde <- mirmodels::cor_de(rsgc5000, "cond", head(names(rsgc5000), -1))
}
```

Index