Package 'pairedGSEA'

May 8, 2024

Title Paired DGE and DGS analysis for gene set enrichment analysis

Version 1.4.0

Description pairedGSEA makes it simple to run a paired Differential Gene Expression (DGE) and Differencital Gene Splicing (DGS) analysis. The package allows you to store intermediate results for further investiation, if desired.
pairedGSEA comes with a wrapper function for running an Over-Representation Analysis (ORA) and functionalities for plotting the results.

License MIT + file LICENSE

Imports DESeq2, DEXSeq, limma, fgsea, sva, SummarizedExperiment, S4Vectors, BiocParallel, ggplot2, aggregation, stats, utils, methods

Suggests writexl, readxl, readr, rhdf5, msigdbr, plotly, testthat (>= 3.0.0), knitr, rmarkdown, covr, BiocStyle

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

VignetteBuilder knitr

biocViews DifferentialExpression, AlternativeSplicing, DifferentialSplicing, GeneExpression, ImmunoOncology, GeneSetEnrichment, Pathways, RNASeq, Software, Transcription,

URL https://github.com/shdam/pairedGSEA

BugReports https://github.com/shdam/pairedGSEA/issues

Depends R (>= 4.3.0)

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

git_branch RELEASE_3_19

git_last_commit 438a425

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-07

Author Søren Helweg Dam [cre, aut] (<https://orcid.org/0000-0002-9895-0930>),
Lars Rønn Olsen [aut] (<https://orcid.org/0000-0002-6725-7850>),
Kristoffer Vitting-Seerup [aut]
(<https://orcid.org/0000-0002-6450-0608>)

Maintainer Søren Helweg Dam <sohdam@dtu.dk>

Contents

`example_diff_result` *Output of running paired_diff on example_se.*

Description

This example result is used primarily to do package tests and for function man pages

Usage

```
data("example_diff_result")
```

Format

A 'DataFrame' with 954 rows and 7 columns.

Value

A 'DataFrame'.

example_gene_sets *MSigDB gene sets from humans, category C5 with ensemble gene IDs*

Description

This example gene set list is used primarily to do package tests and for function man pages.

Usage

```
data("example_gene_sets")
```

Format

A list of 77 human gene sets

Value

A list of gene sets

example_ora_results *Output of running paired_ora on example_diff_result and gene sets extracted from MSigDB*

Description

This example result is used primarily to do package tests and for function man pages.

Usage

```
data("example_ora_results")
```

Format

A 'DataFrame' with 559 rows and 18 columns.

Value

A 'DataFrame'

example_se *A small subset of the GEO:GSE61220 data set.*

Description

The subset is used in the vignettes and function man pages. The subset was created by extracting genes belonging to Telomere-related gene sets and randomly selecting 900 other genes from the original dataset.

Usage

```
data("example_se")
```

Format

A 'SummarizedExperiment'

assay Count matrix with 5611 transcripts and 6 samples

colData The metadata associated with the count matrix

Value

A 'SummarizedExperiment'

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE61220

paired_diff *Run paired DESeq2 and DEXSeq analyses*

Description

With `paired_diff` you can run a paired differential gene expression and splicing analysis. The function expects a counts matrix or a `SummarizedExperiment` or `DESeqDataSet` object as input. A preliminary prefiltering step is performed to remove genes with a summed count lower than the provided threshold. Likewise, genes with counts in only one sample are removed. This step is mostly to speed up differential analyses, as `DESeq2` will do a stricter filtering. Surrogate Variable Analysis is recommended to allow the analyses to take batch effects etc. into account. After the two differential analyses, the transcript-level p-values will be aggregated to gene-level to allow subsequent Gene-Set Enrichment Analysis. Transcript-level results can be extracted by setting `store_results = TRUE`.

Usage

```
paired_diff(
    object,
    group_col,
    sample_col,
    baseline,
    case,
    metadata = NULL,
    covariates = NULL,
    experiment_title = NULL,
    store_results = FALSE,
    run_sva = TRUE,
    use_limma = FALSE,
    prefilter = 10,
    test = "LRT",
    fit_type = "local",
    quiet = FALSE,
    parallel = FALSE,
    BPPARAM = BiocParallel::bpparam(),
    expression_only = FALSE,
    custom_design = FALSE,
    ...
    )
```

Arguments

`object`	A data object of the types matrix, `SummarizedExperiment`, or `DESeqDataSet`. If a matrix is used, please also provide metadata.
`group_col`	The metadata column specifying the what group each sample is associated with
`sample_col`	The column in the metadata that specifies the sample IDs (should correspond to column names in `object`). Set to `"rownames"` if the rownames should be used.
`baseline`	Group value of baseline samples
`case`	Group value of case samples
`metadata`	(Default: `NULL`) A metadata file or `data.frame` object
`covariates`	Name of column(s) in the `metadata` that indicate(s) covariates. E.g., c("gender", "tissue_type")
`experiment_title`	Title of your experiment. Your results will be stored in `paste0("results/", experiment_title, "_pairedGSEA.RDS")`.
`store_results`	(Default: `FALSE`) A logical indicating if results should be stored in the folder `"results/"`.
`run_sva`	(Default: `TRUE`) A logical stating whether SVA should be run.
`use_limma`	(Default: `FALSE`) A logical determining if `limma+voom` or `DESeq2 + DEXSeq` should be used for the analysis

prefilter	(Default: 10) The prefilter threshold, where rowSums lower than the prefilter threshold will be removed from the count matrix. Set to 0 or FALSE to prevent prefiltering
test	either "Wald" or "LRT", which will then use either Wald significance tests (defined by nbinomWaldTest), or the likelihood ratio test on the difference in deviance between a full and reduced model formula (defined by nbinomLRT)
fit_type	(Default: "local") Either "parametric", "local", "mean", or "glmGamPoi" for the type of fitting of dispersions to the mean intensity.
quiet	(Default: FALSE) Whether to print messages
parallel	(Default: FALSE) If FALSE, no parallelization. If TRUE, parallel execution using BiocParallel, see next argument BPPARAM.
BPPARAM	(Default: bpparam()) An optional parameter object passed internally to bplapply when parallel = TRUE. If not specified, the parameters last registered with register will be used.
expression_only	(Default: FALSE) A logical that indicates whether to only run DESeq2 analysis. Not generally recommended. The setting was implemented to make the SVA impact analysis easier
custom_design	(Default: FALSE) A logical or formula. Can be used to apply a custom design formula for the analysis. Generally not recommended, as pairedGSEA will make its own design formula from the group and covariate columns
...	Additional parameters passed to DESeq()

Value

A DFrame of aggregated pvalues

See Also

Other paired: paired_ora()

Examples

```
# Run analysis on included example data
data("example_se")

diff_results <- paired_diff(
    object = example_se[1:15, ],
    group_col = "group_nr",
    sample_col = "id",
    baseline = 1,
    case = 2,
    experiment_title = "Example",
    store_results = FALSE
)
```

paired_ora *Paired Over-Representation Analysis*

Description

paired_ora uses fora to run the over-representation analysis. First the aggregated pvalues are adjusted using the Benjamini & Hochberg method. The analysis is run on all significant genes found by DESeq2 and DEXSeq individually. I.e., two runs of fora are executed and subsequently joined into a single object. You can use prepare_msigdb to create a list of gene_sets.

Usage

```
paired_ora(
    paired_diff_result,
    gene_sets,
    cutoff = 0.05,
    min_size = 25,
    experiment_title = NULL,
    expression_only = FALSE,
    quiet = FALSE
    )
```

Arguments

paired_diff_result	The output of paired_diff
gene_sets	List of gene sets to analyse
cutoff	(Default: 0.05) Adjusted p-value cutoff for selecting significant genes
min_size	(Default: 25) Minimal size of a gene set to test. All pathways below the threshold are excluded.
experiment_title	Title of your experiment. Your results will be stored in paste0("results/", experiment_title, "_fora.RDS").
expression_only	(Default: FALSE) A logical that indicates whether to only run DESeq2 analysis. Not generally recommended.
quiet	(Default: FALSE) Whether to print messages

Value

A data.table of merged ORA results

See Also

Other paired: paired_diff()

Examples

```
data("example_diff_result")
data("example_gene_sets")

ora <- paired_ora(
    example_diff_result,
    example_gene_sets)
```

---

plot_ora *Scatter plot of Over-Representation Analysis results*

Description

Scatter plot of Over-Representation Analysis results

Usage

```
plot_ora(
    ora,
    pattern = NULL,
    paired = TRUE,
    plotly = FALSE,
    cutoff = 0.05,
    lines = TRUE,
    colors = c("darkgray", "purple", "lightblue", "maroon")
    )
```

Arguments

ora	Output of paired_ora
pattern	Highlight pathways containing a specific regex pattern
paired	(Default: TRUE) New plotting mode for paired ora analysis
plotly	(Default: FALSE) Logical on whether to return plot as an interactive plotly plot or a simple ggplot.
cutoff	(Default: 0.2) Adjusted p-value cutoff for pathways to include
lines	(Default: TRUE) Whether to show dashed lines
colors	(Default: c("darkgray", "purple", "navy")) Colors to use in plot. The colors are ordered as "Both", "DGS", and "DGE"

Value

A ggplot

Note

Suggested: importFrom plotly ggplotly

Examples

```
data(example_ora_results)

plot_ora(example_ora_results, pattern = "Telomer")
```

---

prepare_msigdb *Load MSigDB and convert to names list of gene sets*

Description

This function is wrapper around `msigdbr()`. Please see their manual for details on its use. The function extracts the gene set name and a user-defined gene id type (Default: "ensembl_gene"). Please make sure the gene IDs match those from your DE analysis. This function will format the gene sets such that they can be directly used with `paired_ora()`.

Usage

```
prepare_msigdb(
    gene_id_type = "ensembl_gene",
    species = "Homo sapiens",
    category = "C5",
    subcategory = NULL
    )
```

Arguments

`gene_id_type`	(Default: "ensemble_gene") The gene ID type to extract. The IDs should match the gene IDs from your DE analysis.
`species`	Species name, such as Homo sapiens or Mus musculus.
`category`	MSigDB collection abbreviation, such as H or C1.
`subcategory`	MSigDB sub-collection abbreviation, such as CGP or BP.

Value

A list of gene sets

Note

Suggested: importFrom msigdbr msigdbr

Examples

```
gene_sets <- prepare_msigdb(species = "Homo sapiens")
```

Index