Package: basicBulk (via r-universe)

November 24, 2026

Title Basic Bulk Workflows

Version 0.99.0

Description Basic functions that leverage the tidyverse for analyses of (pseudo)bulk omic data. Differential expression and gene set enrichment are run with augere.de and augere.gsea, and basicBulk normalizes a SummarizedExperiment and tidies their results. It provides tables and publication-ready plots for the full workflow: PCA and expression QC, volcano, MA, and heatmap displays of differential expression, overlap (UpSet) and logFC correlation summaries across contrasts, and GSEA (Reactome) pathway analysis.

License GPL (>= 3)

Encoding UTF-8

biocViews Software, DifferentialExpression, GeneExpression, GeneSetEnrichment, Pathways, Reactome, RNASeq, Transcriptomics, Normalization, QualityControl, Visualization

URL https://github.com/Genentech/basicBulk

BugReports https://github.com/Genentech/basicBulk/issues

Roxygen list(markdown = TRUE)

Depends R (>= 4.5.0), SummarizedExperiment

Imports AnnotationDbi, augere.de, augere.gsea, circlize, cli, ComplexHeatmap, ComplexUpset, corrr, cowplot, dplyr (>= 1.2.0), DT, edgeR, forcats, gg4way, ggplot2 (>= 3.5.0), ggprism, ggpubr, ggrepel, ggsci, glue, grid, janitor, knitr, limma, magrittr, MetBrewer, msigdbr (>= 26.1.1), PCAtools, purrr (>= 1.0.0), rlang, rstatix, scales, shadowtext, stats, stringr, tibble, tidyr (>= 1.3.0), tidyselect, tidySummarizedExperiment, utils

Suggests rmarkdown, BiocStyle, org.Hs.eg.db, org.Mm.eg.db, BiocManager, MatrixGenerics, airway, ggh4x, furrr, future, parallelly, patchwork, png, pals, sva

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev make libicu-dev libpng-dev libuv1-dev libssl-dev perl libnode-dev zlib1g-dev

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

Date/Publication 2026-11-23 23:02:52 UTC

RemoteUrl https://github.com/BiocStaging/basicBulk

RemoteRef HEAD

RemoteSha da49b461ba4a9e2a1a8936d860ce55a38b137050

Contents

airwayDGE . . . 3
airwayGSEA . . . 4
assert_fresh . . . 4
barStyle . . . 5
basicHeatmap . . . 6
basicMA . . . 8
basicUpset . . . 9
basicVolcano . . . 10
boxPlot . . . 11
cache_key . . . 12
colorRamp3 . . . 13
corTriplot . . . 13
DEGbar . . . 14
dotPlot . . . 15
extractHeatmap . . . 18
extractUpSet . . . 18
fmt_e . . . 19
fromList2 . . . 19
geneHistogram . . . 20
genePlot . . . 21
getHVGs . . . 23
getReactome . . . 24
GSEAplot . . . 25
GSEAplotList . . . 25
joinFeatures . . . 26
load_if_fresh . . . 27
makeDGEdata . . . 28
makePathwayTibble . . . 29
makeSigCountTibble . . . 29
makeSigTibble . . . 30
PCAgrid . . . 31
PCAplot . . . 32
plotPCAloadings . . . 36
prepareCor . . . 37
printCorTibble . . . 38

printGSEAtibble 39
printHeatmaps 39
printPlots 40
printSigTibble 41
runPCA 41
save_cache 42
scaleRows 43
tidyGSEAlist 43
tidySE 44
tidyVoom 46
topGSEAplot 46

Index **48**

`airwayDGE` *airwayDGE example data*

Description

Example `DGEdata` (the tidy differential-expression object produced by `makeDGEdata()`) for the basicBulk workflow, built by testing three cell lines against a shared reference (`N61311`, `N061011`, and `N080611` vs `N052611`) on the expression-filtered genes of the full normalized airway data, then subset to the most variable retained genes plus a few genes used in examples. It is a named list of one tibble per contrast, with columns including `ID`, `symbol`, `LogFC`, `AveExpr`, `t`, `PValue`, `FDR`, and `B`.

Usage

```
data(airwayDGE)
```

Format

A named `list` of three tibbles, one per contrast.

Value

A named list of tibbles.

Source

airway

airwayGSEA *airwayGSEA example data*

Description

Example tidied GSEA results (the object produced by `tidyGSEAlist()`) for the basicBulk workflow, built by running camera on the expression-filtered genes of the full normalized airway data against the Reactome gene sets from `getReactome()` for each of the three contrasts. It is a named list of one tibble per contrast, with columns `Database`, `Term`, `NumGenes`, `Direction`, `PValue`, and `FDR`.

Usage

```
data(airwayGSEA)
```

Format

A named `list` of three tibbles, one per contrast.

Value

A named list of tibbles.

Source

airway

assert_fresh *Assert a cached artifact is present and newer than its source*

Description

A cheap first-line freshness backstop for report/slide caches: stop with an informative error if `artifact` is missing, or if its modification time predates `source`. This catches the common "I re-ran an upstream step and forgot to regenerate the cache" case. It is a *timestamp* check only — for a guarantee that a cache reflects its inputs regardless of file mtimes (e.g. in place edits, out-of-order re-saves), use `cache_key()` + `load_if_fresh()`.

Usage

```
assert_fresh(artifact, source, regen_cmd = NULL)
```

Arguments

artifact	Path to the cached file that must be up to date.
source	Path to the input file the artifact is derived from.
regen_cmd	Optional string shown in the error telling the user how to regenerate the artifact. Defaults to "the step that creates it".

Value

Invisibly TRUE when the artifact is present and fresh; otherwise stops with an error.

Examples

```
se0 <- augere.de::loadExampleDataset()
data(airwayDGE)
settings <- tempfile(fileext = ".RData")
save(se0, file = settings)
results <- tempfile(fileext = ".RData")
save(airwayDGE, file = results)
assert_fresh(results, settings)
```

---

barStyle *barStyle*

Description

Makes plots less baring

Usage

```
barStyle()
```

Value

An improved ggplot2

Examples

```
data(airwayDGE)
airwayDGE %>%
    makeSigTibble() %>%
    makeSigCountTibble() %>%
    ggplot2::ggplot(ggplot2::aes(x = contrast, y = total)) +
    barStyle()
```

basicHeatmap *basicHeatmap*

Description

Create a basic heatmap of DEGs or pathways.

Usage

```
basicHeatmap(
  se,
  sigTibble = NULL,
  geneSet = NULL,
  assay = "logCPM",
  contrast = NULL,
  pathways = FALSE,
  cluster_columns = FALSE,
  cluster_rows = TRUE,
  column_title = NULL,
  show_row_names = FALSE,
  show_column_names = FALSE,
  heatmap_legend_side = "bottom",
  annotation_legend_side = "right",
  verbose = FALSE,
  legendLimits = NULL,
  top_annotation = NULL,
  right_annotation = NULL,
  row_split = NULL,
  draw = TRUE,
  ...
)
```

Arguments

se	A SummarizedExperiment object with "Group" in the colData.
sigTibble	A tibble of significant genes.
geneSet	A gene set of Ensembl IDs or symbols to use instead of the sigTibble; with cluster_rows = FALSE the rows keep its order.
assay	Character specifying the name of the assay from the se.
contrast	Character specifying the name of a contrast ("X vs Y") to subset by if not NULL: keeps the samples of its two groups and, for a sigTibble of DEGs, only that contrast's genes.
pathways	Logical indicating whether sigTibble is a pathwayTibble.
cluster_columns	Whether make cluster on columns? Same settings as cluster_rows.

cluster_rows	If the value is a logical, it controls whether to make cluster on rows. The value can also be a hclust or a dendrogram which already contains clustering. Check https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#clustering .
column_title	Title on the column.
show_row_names	Whether show row names.
show_column_names	Whether show column names.
heatmap_legend_side	Character specifying the side for the heatmap legend.
annotation_legend_side	Character specifying the side for the annotation legend.
verbose	Logical indicating whether to print range of Z-scores
legendLimits	Manually override the legend limits
top_annotation	A HeatmapAnnotation object.
right_annotation	it should be specified by rowAnnotation.
row_split	Same as split.
draw	Logical indicating whether to call draw.
...	Additional arguments passed onto Heatmap.

Value

A HeatmapList object.

References

Gu Z, Eils R, Schlesner M (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics*. doi:10.1093/bioinformatics/btw313

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
basicHeatmap(se, sigTibble = makeSigTibble(airwayDGE))
```

basicMA *basicMA*

Description

Create a grid of MA plots to examinine the LogFC (M) and average logCPM (A)

Usage

```
basicMA(
  DGEdata,
  nrow = NULL,
  ncol = NULL,
  FDRcutoff = 0.05,
  logFCcutoff = 1,
  pointShape = 19,
  pointSize = 0.1
)
```

Arguments

DGEdata	A list of DGE results
nrow	Number of rows
ncol	Number of columns
FDRcutoff	Numeric for the FDR cut-off for DEGs, default is 0.05
logFCcutoff	Numeric for the absolute Log2FC cut-off for DEGs, default is 1
pointShape	Numeric (or ".") specifying shape of points
pointSize	Numeric specifying size of points

Value

A ggplot2 wrap

Examples

```
data(airwayDGE)
basicMA(airwayDGE)
```

basicUpset *basicUpset*

Description

A wrapper for upset

Usage

```
basicUpset(
  DGEdata,
  directionSplit = TRUE,
  FDRcutoff = 0.05,
  logFCcutoff = 1,
  n_intersections = 40,
  width_ratio = 0.1,
  height_ratio = 0.95,
  counts = TRUE,
  sort_sets = "descending",
  fontSize = 12,
  fontSizeOverall = 9
)
```

Arguments

DGEdata	A list of DGE results
directionSplit	Logical indicating whether gene lists should be split by direction, default is TRUE
FDRcutoff	Numeric for the FDR cut-off for DEGs, default is 0.05
logFCcutoff	Numeric for the absolute Log2FC cut-off for DEGs, default is 1
n_intersections	the exact number of the intersections to be displayed; n largest intersections that meet the size and degree criteria will be shown
width_ratio	ratio of the overall set size width to intersection matrix width
height_ratio	ratio of the intersection matrix to intersection size height
counts	whether to display count number labels above the bars
sort_sets	whether to sort the rows in the intersection matrix (descending sort by default); one of: 'ascending', 'descending', FALSE
fontSize	Integer specifying font size
fontSizeOverall	Integer specifying font size for overall set size axis labels

Value

A ggplot2

References

Lex A, Gehlenborg N, Strobelt H, Vuillemot R, Pfister H (2014). UpSet: visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), 1983-1992. doi:10.1109/TVCG.2014.2346248

Krassowski M (2020). ComplexUpset. doi:10.5281/zenodo.3700590

Examples

```
data(airwayDGE)
basicUpset(airwayDGE)
```

`basicVolcano` *basicVolcano*

Description

A wrap of basic volcanos with DEG counts

Usage

```
basicVolcano(
  DGEdata,
  FDRcutoff = 0.05,
  logFCcutoff = 1,
  lab = NULL,
  pointSize = 0.1,
  nrow = NULL,
  ncol = NULL,
  labSize = 9,
  labFace = "italic",
  style = c("basic", "enhanced")
)
```

Arguments

`DGEdata`	A list of DGE results
`FDRcutoff`	Numeric for the FDR cut-off for DEGs, default is 0.05
`logFCcutoff`	Numeric for the absolute Log2FC cut-off for DEGs, default is 1
`lab`	TRUE adds labels for all significant genes, while a character vector of gene symbols will only label that selection
`pointSize`	Numeric specifying the size of the points
`nrow`	Number of rows
`ncol`	Number of columns
`labSize`	Numeric for font size of labels (in pt)
`labFace`	Character specifying font face of gene labels (i.e bold, italic, bolditalic)
`style`	Character specifying either a "basic" or "enhanced" style.

Value

A ggplot2 wrap

Examples

```
data(airwayDGE)
basicVolcano(airwayDGE)
```

`boxPlot` *boxPlot*

Description

Makes a ggboxplot

Usage

```
boxPlot(data, x, y, color, size = 1.25, plotly = FALSE, ...)
```

Arguments

`data`	The data
`x`	Column for the x-axis
`y`	Column for the y-axis
`color`	Column for the color
`size`	Numeric specifying size of the dots
`plotly`	Logical indicating whether to add the text aes for ggplotly
`...`	Additional arguments for ggboxplot

Value

A stylish ggppubr boxplot

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
se %>%
    tibble::as_tibble() %>%
    dplyr::filter(symbol == "FKBP5") %>%
    boxPlot(x = "Group", y = "logCPM", color = "Group")
```

cache_key *Content hash of a cache's invalidating inputs*

Description

Computes a stable digest of the objects a cached computation depends on, for use as the freshness key with `save_cache()` / `load_if_fresh()`. Pass the *actual input objects* (e.g. the assay matrix and colData of the normalized `SummarizedExperiment`, the design matrix, the contrast list, the gene sets), so that any change to a hashed input — an outlier sample removed, a covariate / block / `dc.block` changed, a contrast or gene set edited — changes the key and forces a recompute. Key on the inputs only, not on package versions or function code: when an upgrade could change the results, force a recompute instead. Because the *objects* are hashed (not a hand-maintained list of field names), a newly added input is captured automatically.

Usage

```
cache_key(...)
```

Arguments

...	The invalidating inputs to hash (any R objects and/or strings).

Details

Hash data objects, not environments or closures, which do not hash stably. Key on the inputs that determine the result. The sample `colData`, the gene `rownames`, the model/contrast specification and the gene-set collection catch a dropped sample or a changed covariate/block/`dc.block`, but not revised counts under the same sample and gene names: add the count values when the input counts can change. Avoid hashing a DelayedArray / HDF5-backed assay directly — it may hash on its backing pointer rather than its contents; realize it with `as.matrix()` first.

Value

A single character string (the digest).

Examples

```
se0 <- augere.de::loadExampleDataset()
cache_key(SummarizedExperiment::colData(se0), rownames(se0))
```

colorRamp3 *colorRamp3*

Description

Generate a function for a color palette for a heatmap

Usage

```
colorRamp3(
  mat,
  heatColors = MetBrewer::met.brewer("Troy", n = 10, type = "continuous") %>%
    as.character() %>% rev() %>% append("#f6f2ee", after = 5),
  legendLimits = NULL
)
```

Arguments

mat	A Matrix
heatColors	Character vector of colors that has an odd number of colors with a midpoint; the default is MetBrewer's Troy palette (10 colors, blue to red) around a white midpoint
legendLimits	Manually override the legend limits

Value

A function

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
SummarizedExperiment::assay(se, "logCPM")[getHVGs(se, n = 20, symbols = FALSE), ] %>%
    scaleRows() %>%
    colorRamp3()
```

corTriplot *corTriplot*

Description

Plot the lower triangle of logFC correlations from prepareCor

Usage

```
corTriplot(corRes)
```

Arguments

corRes	cor_df from prepareCor

Value

A ggplot

References

Laufer BI, Friedman BA (2026). gg4way: 4way Plots of Differential Expression. R package. doi:10.18129/B9.bioc.gg4way

Kuhn M, Jackson S, Cimentada J (2025). corrr: Correlations in R. R package. doi:10.32614/CRAN.package.corrr

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
corRes <- tidyr::crossing(x = names(airwayDGE), y = names(airwayDGE)) %>%
    dplyr::filter(x != y) %>%
    dplyr::mutate(name = paste(stringr::str_remove(x, " vs .*"), "vs",
                               stringr::str_remove(y, " vs .*"))) %>%
    prepareCor(airwayDGE, cd = SummarizedExperiment::colData(se))
corTriplot(corRes)
```

DEGbar *DEGbar*

Description

Bar plot of total DEGs

Usage

```
DEGbar(
  sigTabyl,
  x = "contrast",
  addCounts = TRUE,
  flip = TRUE,
  linewidth = 0.1
)
```

Arguments

sigTabyl	A tibble from makeSigCountTibble or a sigTibble run through janitor::tabyl(contrast, direction) and mutated with other variables of interest
x	Character specifying the x-axis variable
addCounts	Logical indicating whether to add text with totals
flip	Logical indicating whether to flip the coordinates
linewidth	Numeric specifying linewidth for black outline

Value

A ggplot2

Examples

```
data(airwayDGE)
airwayDGE %>%
    makeSigTibble() %>%
    makeSigCountTibble() %>%
    DEGbar()
```

---

dotPlot *dotPlot*

Description

Makes a ggstripchart

Usage

```
dotPlot(
  data,
  x,
  y,
  color,
  shape = 21,
  size = 1.25,
  alpha = 0.6,
  jitter = 0,
  plotly = FALSE,
  ...
)
```

Arguments

data	a data frame
x	character string containing the name of x variable.
y	character vector containing one or more variables to plot
color	outline color.
shape	point shape
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
alpha	Numeric specifying the alpha
jitter	the amount of jitter.
plotly	Logical indicating whether to add the text aes for ggplotly
...	Arguments passed on to ggpubr::ggstripchart

`combine` logical value. Default is FALSE. Used only when y is a vector containing multiple variables to plot. If TRUE, create a multi-panel plot by combining the plot of y variables.

`merge` logical or character value. Default is FALSE. Used only when y is a vector containing multiple variables to plot. If TRUE, merge multiple y variables in the same plotting area. Allowed values include also "asis" (TRUE) and "flip". If merge = "flip", then y variables are used as x tick labels and the x variable is used as grouping variable.

`palette` the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

`title` plot main title.

`xlab` character vector specifying x axis labels. Use xlab = FALSE to hide xlab.

`ylab` character vector specifying y axis labels. Use ylab = FALSE to hide ylab.

`facet.by` character vector, of length 1 or 2, specifying grouping variables for faceting the plot into multiple panels. Should be in the data.

`panel.labs` a list of one or two character vectors to modify facet panel labels. For example, panel.labs = list(sex = c("Male", "Female")) specifies the labels for the "sex" variable. For two grouping variables, you can use for example panel.labs = list(sex = c("Male", "Female"), rx = c("Obs", "Lev", "Lev2")).

`short.panel.labs` logical value. Default is TRUE. If TRUE, create short labels for panels by omitting variable names; in other words panels will be labelled only by variable grouping levels.

`select` character vector specifying which items to display.

`remove` character vector specifying which items to remove from the plot.

`order` character vector specifying the order of items.

`add` character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_hilow", "median_q1q3", "median_mad", "median_range"; see ?desc_statby for more details.

`add.params` parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").

`error.plot` plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...

`label` the name of the column containing point labels. Can be also a character vector with length = nrow(data).

`font.label` a list which can contain the combination of the following elements: the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of labels. For example font.label = list(size = 14, face = "bold", color ="red"). To specify only the size and the style, use font.label = list(size = 14, face = "plain").

`label.select` can be of two formats:

- a character vector specifying some labels to show.
- a list containing one or the combination of the following components:
  - `top.up` and `top.down`: to display the labels of the top up/down points. For example, `label.select = list(top.up = 10, top.down = 4)`.
  - `criteria`: to filter, for example, by x and y variables values, use this: `label.select = list(criteria = "`y` > 2 & `y` < 5 & `x` %in% c('A', 'B')")`.

`repel` a logical value, whether to use ggrepel to avoid overplotting text labels or not.

`label.rectangle` logical value. If TRUE, add rectangle underneath the text, making it easier to read.

`show.n` logical. If TRUE, displays the number of observations ("n = <count>") at the top of each group. Off by default. When the groups are dodged (a `color`/`fill` grouping with a dodging `position`), one count is shown per group; otherwise a single count is shown per x-axis tick. Counts respect `select`/`remove` and are computed per facet.

Value

A stylish ggppubr strip chart

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
se %>%
    tibble::as_tibble() %>%
    dplyr::filter(symbol == "FKBP5") %>%
    dotPlot(x = "Group", y = "logCPM", color = "Group")
```

extractHeatmap *extractHeatmap*

Description

Extract the clustered matrix and relevant metadata from a heatmap for figure source data

Usage

```
extractHeatmap(se, heatmap)
```

Arguments

se	A SummarizedExperiment object
heatmap	A drawn heatmap from basicHeatmap

Value

A tibble

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
heatmap <- basicHeatmap(se, sigTibble = makeSigTibble(airwayDGE))
extractHeatmap(se, heatmap)
```

extractUpSet *extractUpSet*

Description

Extract gene IDs from an intersection in an UpSet plot

Usage

```
extractUpSet(DGEcontrasts, DGEdata, FDRcutoff = 0.05, logFCcutoff = 1)
```

Arguments

DGEcontrasts	A character vector of the UpSet sets to intersect: DGEdata names followed by " Up" or " Down"
DGEdata	A list of DGE results
FDRcutoff	Numeric for the FDR cut-off for DEGs, default is 0.05
logFCcutoff	Numeric for the absolute Log2FC cut-off for DEGs, default is 1

Value

A character vector of the gene IDs in the requested overlap.

Examples

```
data(airwayDGE)
names(airwayDGE) %>%
    paste("Up") %>%
    extractUpSet(airwayDGE)
```

`fmt_e` *Format a number in one-significant-digit scientific notation*

Description

Formats a p-value (or any numeric) as e.g. `"1.2e-04"`. Raw numeric p-values render inconsistently inside inline knitr `r` expressions; pre-formatting with `fmt_e()` keeps slide/report values stable.

Usage

```
fmt_e(x)
```

Arguments

`x`	A numeric vector.

Value

A character vector in scientific notation with one decimal place.

Examples

```
fmt_e(0.000123)
```

`fromList2` *fromList2*

Description

Convert a named list of sets to a data.frame of set membership with the elements as rownames

Usage

```
fromList2(input)
```

Arguments

input	A named list of character vectors

Value

A data.frame of 0/1 set membership, with one row per element (rownames) and one column per set.

Examples

```
data(airwayDGE)
airwayDGE %>%
    purrr::map(~ dplyr::filter(.x, FDR < 0.05) %>% dplyr::pull(ID)) %>%
    fromList2()
```

---

geneHistogram *geneHistogram*

Description

Histograms of mean gene expression for each group

Usage

```
geneHistogram(se, geneSymbol = NULL, assay = "logCPM", palette = "category20")
```

Arguments

se	A SummarizedExperiment object
geneSymbol	An optional character vector specifying the gene to draw a mean expression line for
assay	Assay from se to use
palette	Character specifying palette name for scale_color_d3

Value

A ggplot

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
geneHistogram(se, geneSymbol = "FKBP5")
```

genePlot *genePlot*

Description

Faceted plots for genes of interest

Usage

```
genePlot(
  se,
  genes,
  DGEdata = NULL,
  assay = "logCPM",
  addStats = !is.null(DGEdata),
  ref.group = NULL,
  logFCcutoff = 1,
  plotType = c("dot", "box"),
  size = 2,
  jitter = 0.1,
  nrow = NULL,
  ncol = NULL,
  nrow_legend = NULL,
  tip.length = 0,
  ggprism = FALSE,
  nudge_y = ifelse(is.null(ref.group), 0, 0.1),
  ...
)
```

Arguments

se	A SummarizedExperiment object, or a tibble of gene set scores from joinFeatures with score = TRUE stacked with the gene set names in a symbol column
genes	A character vector specifying the genes (or gene sets) to plot
DGEdata	A list of DGE results, which requires a SummarizedExperiment se
assay	Assay from se to use
addStats	Logical indicating whether to add significance annotations from DGEdata
ref.group	Character specifying reference group for statistics
logFCcutoff	Numeric for the absolute Log2FC cut-off for DEGs, default is 1
plotType	A character specifying a "box" or "dot" plot
size	Numeric specifying size of the dots
jitter	Numeric specifying jitter of the dots (for dot plot only)
nrow	Number of rows
ncol	Number of columns

nrow_legend Number of rows for the legend

tip.length Numeric specifying length of bracket tips

ggprism Logical indicating whether a prism theme should be applied

nudge_y Numeric specifying the multiplier to increase spacing between statistics results and max y value; the default is 0.1 with a ref.group (its asterisks start at the max y value) and 0 without (the brackets are already spaced above it)

... Arguments passed on to ggprism::add_pvalue

data A data.frame with the statistics to plot. Default format has the following columns: group1 | group2 | p.adj | y.position | etc. group1 and group2 are the two groups that were compared. p.adj is the adjusted p-value. y.position is the y coordinate that specifies where on the plot the p-value should go. The column names can differ from the default as long as their are specified when calling the function.

label string. Name of column in data that contains the text to plot (e.g. label = "p.adj"). Can also be an expression that can be formatted by glue (e.g. label = "p = {p.adj}").

xmin string. Name of column in data that contains the position of the left side of the brackets. Default is "group1".

xmax Optional. string. Name of column in data that contains the position of the right side of the brackets. Default is "group2". If NULL, the p-values are plotted as text only.

x string or numeric. x coordinate of the p-value text. Only use when plotting p-value as text only (no brackets).

y.position string. Name of column in data containing the y coordinates (numeric) of each p-value to be plotted. Can also be a single number to plot all p-values at the same height or a numeric vector that will override data.

parse logical. Default is FALSE. If TRUE the text labels will be parsed into expressions and displayed as described in ?plotmath.

label.size numeric. Size of text.

colour,color string. Colour of text.

bracket.size numeric. Line width of bracket.

bracket.colour,bracket.color string. Colour of bracket. Default is NULL which causes brackets to inherit the colour of the text.

bracket.shorten numeric. Shortens the brackets slightly to allow them to be plotted side-by-side at the same y position.

bracket.nudge.y numeric. Changes the y position of p-values. Useful for slightly adjusting p-values if the text is cut off.

step.increase numeric. Changes the space between brackets.

step.group.by string. Variable to group brackets by.

remove.bracket logical. If TRUE all brackets are removed and p-value is shown as text only.

coord.flip logical. If TRUE p-values are rotated by 90 degrees. Should be used with coord_flip

position string or call to position function such as position_dodge. Typically used for adjusting x position of p-values to be in line with dodged data.

Value

A ggplot

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
genePlot(se, genes = c("RARRES1", "TMEM176B", "GBP3"))

## Gene set scores
gene.sets <- getReactome("Homo sapiens")
gene.sets[c("REACTOME_CELL_SURFACE_INTERACTIONS_AT_THE_VASCULAR_WALL",
            "REACTOME_MUSCLE_CONTRACTION")] %>%
    purrr::map(~ joinFeatures(se, features = .x, score = TRUE)) %>%
    purrr::list_rbind(names_to = "symbol") %>%
    genePlot(genes = unique(.$symbol))
```

---

getHVGs *getHVGs*

Description

Get highly variable genes.

Usage

```
getHVGs(se, assay = "logCPM", n = 100, symbols = TRUE)
```

Arguments

se	A SummarizedExperiment object
assay	Character specifying the name of the assay from the se
n	Integer specifying the number of genes.
symbols	Logical indicating to return gene symbols instead of Ensembl IDs.

Value

A character vector

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
getHVGs(se, n = 20)
```

`getReactome` *getReactome*

Description

Get the Reactome pathway collection (MSigDB C2:CP:REACTOME) for GSEA as a named list of Ensembl gene IDs, via msigdbr. Set names follow the `REACTOME_*` convention expected by tidyGSEAlist and makePathwayTibble, and the Ensembl IDs match the row names used by runContrast.

Usage

```
getReactome(organism)
```

Arguments

`organism`	Organism of interest c("Homo sapiens", "Mus musculus")

Value

A named list of character vectors of Ensembl gene IDs

References

Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43), 15545-15550. doi:10.1073/pnas.0506580102

Liberzon A, Birger C, Thorvaldsdóttir H, Ghandi M, Mesirov JP, Tamayo P (2015). The Molecular Signatures Database hallmark gene set collection. *Cell Systems*, 1(6), 417-425. doi:10.1016/j.cels.2015.12.004

Ragueneau E, Gong C, Sinquin P, Sevilla C, Beavers D, Grentner A, Griss J, Hogue GFJ, Li NT, Matthews L, May B, Milacic M, Mohammadi H, Petryszak R, Rothfels K, Shamovsky V, Stephan R, Tiwari K, Weiser J, Wright A, Gillespie M, Wu G, Stein L, Hermjakob H, D'Eustachio P (2025). The Reactome Knowledgebase 2026. *Nucleic Acids Research*, 54(D1), D673-D681. doi:10.1093/nar/gkaf1223

Examples

```
gene.sets <- getReactome("Homo sapiens")
```

GSEAplot *GSEAplot*

Description

Plot GSEA results

Usage

```
GSEAplot(GSEA, n_terms = 5)
```

Arguments

GSEA	A tibble of GSEA results for one contrast, an element of the tidyGSEAlist output
n_terms	Numeric specifying number of terms per a facet (Up and Down)

Value

A ggplot2

Examples

```
data(airwayGSEA)
GSEAplot(airwayGSEA[[1]])
```

GSEAplotList *GSEAplotList*

Description

Create a list of GSEA plots for all comparisons

Usage

```
GSEAplotList(GSEAlist, n_terms = 5)
```

Arguments

GSEAlist	A named list of GSEA result tibbles from tidyGSEAlist
n_terms	Numeric specifying number of terms per a facet (Up and Down)

Value

A list of ggplots

Examples

```
data(airwayGSEA)
GSEAplotList(airwayGSEA)
```

`joinFeatures` *joinFeatures*

Description

Join features in the tidy transcriptomics style

Usage

```
joinFeatures(
  se,
  features,
  symbol = FALSE,
  score = FALSE,
  long = TRUE,
  assay = "logCPM"
)
```

Arguments

`se`	A SummarizedExperiment object
`features`	A character vector of features
`symbol`	Logical indicating whether the feature character vector contains gene symbols, FALSE indicates it contains Ensembl IDs
`score`	Logical indicating whether a gene set score should be returned
`long`	Logical indicating whether long format should be returned (FALSE for wide matrix style)
`assay`	Assay from se to use

Value

A tibble

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
joinFeatures(se, c("RARRES1", "TMEM176B", "GBP3"), symbol = TRUE)
```

load_if_fresh *Load a cache only if its stored key matches the current inputs*

Description

The safe-by-default guarded loader. Loads `file` (written by `save_cache()`) into `envir` **only** when its stored freshness key is identical to `key` (the current inputs, from `cache_key()`). In every other case — the file is missing, unreadable, carries no stored key (an old or foreign cache), or its key differs from `key` — it returns `FALSE` **without loading anything**, so the caller recomputes. This makes stale caches fail safe: a change to any hashed input forces a recompute rather than silently reusing an old result.

Usage

```
load_if_fresh(file, key, envir = parent.frame())
```

Arguments

`file`	Path to the `.RData` cache written by `save_cache()`.
`key`	The current freshness key from `cache_key()`.
`envir`	Environment to load the cached objects into on a hit. Defaults to the caller's environment.

Details

The guard is enforced wherever this runs — a report chunk, the R console, an unattended re-render — so freshness does not depend on any external process being "aware" of the change. Chain it across dependent caches (each keyed on its freshly-computed upstream) so staleness cannot propagate through a cache-of-a-cache.

Value

`TRUE` (and injects the cached objects into `envir`) on a fresh hit; otherwise `FALSE` and nothing is loaded.

Examples

```
se0 <- augere.de::loadExampleDataset()
data(airwayGSEA)
key <- cache_key(SummarizedExperiment::colData(se0), rownames(se0))
file <- tempfile(fileext = ".RData")
save_cache(file, key, GSEAlist = airwayGSEA)
load_if_fresh(file, key)
```

makeDGEdata *makeDGEdata*

Description

Create a named list of DGE stats tibbles for downstream functions

Usage

```
makeDGEdata(se, voomList)
```

Arguments

se	A SummarizedExperiment object
voomList	A list of DataFrames of voom results from runVoom

Value

A list of tibbles

References

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

Law CW, Chen Y, Shi W, Smyth GK (2014). voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2), R29. doi:10.1186/gb2014152r29

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
voomList <- se %>%
    augere.de::runVoom(groups = "cell",
                       block = "dex",
                       comparisons = list(c("N61311", "N052611")),
                       assay = "counts",
                       output.dir = tempfile(),
                       save.results = FALSE) %>%
    magrittr::extract2("results")
makeDGEdata(se, voomList)
```

makePathwayTibble *makePathwayTibble*

Description

Make a tibble of pathways for basicHeatmap

Usage

```
makePathwayTibble(pathways, gene.sets)
```

Arguments

pathways	A character vector of pathway terms from printGSEAtibble
gene.sets	Gene sets from getReactome

Value

A tibble

Examples

```
data(airwayGSEA)
gene.sets <- getReactome("Homo sapiens")
airwayGSEA[[1]] %>%
    dplyr::slice(1:2) %>%
    dplyr::pull(Term) %>%
    stringr::str_replace_all(" ", "_") %>%
    paste0("REACTOME_", .) %>%
    makePathwayTibble(gene.sets)
```

makeSigCountTibble *makeSigCountTibble*

Description

Prepare the sigTibble for DEGbar

Usage

```
makeSigCountTibble(sigTibble)
```

Arguments

sigTibble	A tibble from makeSigTibble. Every contrast level gets a row, with zero counts for contrasts without DEGs; to count only some contrasts, subset the DGEdata before makeSigTibble

Value

A tibble

Examples

```
data(airwayDGE)
airwayDGE %>%
    makeSigTibble() %>%
    makeSigCountTibble()
```

`makeSigTibble` *makeSigTibble*

Description

Make the sigTibble

Usage

```
makeSigTibble(DGEdata, FDRcutoff = 0.05, logFCcutoff = 1)
```

Arguments

`DGEdata`	A list of tibbles with tidied voom results
`FDRcutoff`	Numeric for the FDR cut-off for DEGs, default is 0.05
`logFCcutoff`	Numeric for the absolute Log2FC cut-off for DEGs, default is 1

Value

A tibble of tidied significant DGE results, with `contrast` a factor whose levels are all the `DGEdata` names

Examples

```
data(airwayDGE)
makeSigTibble(airwayDGE)
```

PCAgrid *PCAgrid*

Description

A grid with a PCA biplot and top loadings for PCs 1 and 2

Usage

```
PCAgrid(se, assay = "logCPM", removeVar = 0.1, pal = NULL, ...)
```

Arguments

se	A SummarizedExperiment object
assay	Assay from se to use
removeVar	Argument to remove lower percent of variables for pca
pal	Named character vector of colors for the PCA biplot
...	Additional arguments for PCAplot()

Value

A ggplot (a cowplot grid of the PCA biplot and loadings).

References

Blighe K, Andrews J, Lun A (2026). PCAtools: Everything Principal Components Analysis. R package. doi:10.18129/B9.bioc.PCAtools

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
PCAgrid(se, colby = "dex", shape = "cell")
```

PCAplot *PCAplot*

Description

Wrapper for generating a PCA biplot with PCAtools via pca and biplot

Usage

```
PCAplot(
  se,
  PCA = NULL,
  assay = "logCPM",
  colby = "Group",
  lab = "Sample",
  pointSize = 6,
  labSize = 4,
  alpha = 1,
  legendPosition = "bottom",
  legendLabSize = 16,
  legendIconSize = 6,
  palette = "category20",
  x = "PC1",
  y = "PC2",
  ...
)
```

Arguments

se	A SummarizedExperiment object
PCA	A pca object from pca
assay	Assay from se to use
colby	colData variable to color points by
lab	colData variable to label points by, or NULL for no labels
pointSize	Size of plotted points.
labSize	Size of labels.
alpha	Numeric specifying alpha for points
legendPosition	Position of legend ('top', 'bottom', 'left', 'right', 'none').
legendLabSize	Size of plot legend text.
legendIconSize	Size of plot legend icons / symbols.
palette	Palette type. There are four available options: • "category10" (10-color palette). • "category20" (20-color palette).

- "category20b" (20-color palette).
- "category20c" (20-color palette).

x A principal component to plot on x-axis. All principal component names are stored in pcaobj$label.

y A principal component to plot on y-axis. All principal component names are stored in pcaobj$label.

... Arguments passed on to PCAtools::biplot

pcaobj Object of class 'pca' created by [pca()].

showLoadings Logical, indicating whether or not to overlay variable loadings.

ntopLoadings If showLoadings == TRUE, select this many variables based on absolute ordered variable loading for each PC in the biplot. As a result of looking across 2 PCs, it can occur whereby greater than this number are actually displayed.

LoadingsNames Names of variable loadings to display. If provided, this overrides 'ntopLoadings'.

showLoadingsNames Logical, indicating to show variable loadings names or not.

colLoadingsNames If 'showLoadings == TRUE', colour of text labels.

sizeLoadingsNames If 'showLoadings == TRUE', size of text labels.

boxedLoadingsNames Logical, if 'showLoadings == TRUE', draw text labels in boxes.

fillBoxedLoadings When 'boxedLoadingsNames == TRUE', this controls the background fill of the boxes. To control both the fill and transparency, user can specify a value of the form 'alpha(<colour>, <alpha>)'.

drawConnectorsLoadings If 'showLoadings == TRUE', draw line connectors to the variable loadings arrows in order to fit more labels in the plot space.

widthConnectorsLoadings If 'showLoadings == TRUE', width of the line connectors drawn to the variable loadings arrows.

colConnectorsLoadings If 'showLoadings == TRUE', colour of the line connectors drawn to the variable loadings arrows.

lengthLoadingsArrowsFactor If 'showLoadings == TRUE', multiply the internally-determined length of the variable loadings arrows by this factor.

colLoadingsArrows If showLoadings == TRUE, colour of the variable loadings arrows.

widthLoadingsArrows If showLoadings == TRUE, width of the variable loadings arrows.

alphaLoadingsArrow If showLoadings == TRUE, colour transparency of the variable loadings arrows.

colkey Vector of name-value pairs relating to value passed to 'col', e.g., c(A='forestgreen', B='gold').

colLegendTitle Title of the legend for the variable specified by 'colby'.

singlecol If specified, all points will be shaded by this colour. Overrides 'col'.

shape If NULL, all points will be have the same shape. If not NULL, value is assumed to be a column name in pcaobj$metadata relating to some grouping/categorical variable.

shapekey Vector of name-value pairs relating to value passed to 'shape', e.g., c(A=10, B=21).

shapeLegendTitle Title of the legend for the variable specified by 'shape'.

legendTitleSize Size of plot legend title text.

encircle Logical, indicating whether to draw a polygon around the groups specified by 'colby'.

encircleFill Logical, if 'encircle == TRUE', this determines whether to fill the encircled region or not.

encircleFillKey Vector of name-value pairs relating to value passed to 'encircleFill', e.g., c(A='forestgreen', B='gold'). If NULL, the fill is controlled by whatever has already been used for 'colby' / 'colkey'.

encircleAlpha Alpha for purposes of controlling colour transparency of the encircled region. Used when 'encircle == TRUE'.

encircleLineSize Line width of the encircled line when 'encircle == TRUE'.

encircleLineCol Colour of the encircled line when 'encircle == TRUE'.

ellipse Logical, indicating whether to draw a data ellipse around the groups specified by 'colby'.

ellipseType [paraphrased from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The type of ellipse. "t" assumes a multivariate t-distribution, while "norm" assumes a multivariate normal distribution. "euclid" draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless coord_fixed() is applied.

ellipseLevel [paraphrased from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The level at which to draw an ellipse, or, if ellipseType="euclid", the radius of the circle to be drawn.

ellipseSegments [from https://ggplot2.tidyverse.org/reference/stat_ellipse.html] The number of segments to be used in drawing the ellipse.

ellipseFill Logical, if 'ellipse == TRUE', this determines whether to fill the region or not.

ellipseFillKey Vector of name-value pairs relating to value passed to 'ellipseFill', e.g., c(A='forestgreen', B='gold'). If NULL, the fill is controlled by whatever has already been used for 'colby' / 'colkey'.

ellipseAlpha Alpha for purposes of controlling colour transparency of the ellipse region. Used when 'ellipse == TRUE'.

ellipseLineSize Line width of the ellipse line when 'ellipse == TRUE'.

ellipseLineCol Colour of the ellipse line when 'ellipse == TRUE'.

xlim Limits of the x-axis.

ylim Limits of the y-axis.

boxedLabels Logical, draw text labels in boxes.

selectLab A vector containing a subset of lab to plot.

drawConnectors Logical, indicating whether or not to connect plot labels to their corresponding points by line connectors.

widthConnectors Line width of connectors.

colConnectors Line colour of connectors.

`max.overlaps` Equivalent of max.overlaps in ggrepel. Set to 'Inf' to always display all labels when drawConnectors = TRUE.

`maxoverlapsConnectors` See max.overlaps.

`min.segment.length` When drawConnectors = TRUE, specifies the minimum length of the connector line segments.

`directionConnectors` direction in which to draw connectors. 'both', 'x', or 'y'.

`xlab` Label for x-axis.

`xlabAngle` Rotation angle of x-axis labels.

`xlabhjust` Horizontal adjustment of x-axis labels.

`xlabvjust` Vertical adjustment of x-axis labels.

`ylab` Label for y-axis.

`ylabAngle` Rotation angle of y-axis labels.

`ylabhjust` Horizontal adjustment of y-axis labels.

`ylabvjust` Vertical adjustment of y-axis labels.

`axisLabSize` Size of x- and y-axis labels.

`title` Plot title.

`subtitle` Plot subtitle.

`caption` Plot caption.

`titleLabSize` Size of plot title.

`subtitleLabSize` Size of plot subtitle.

`captionLabSize` Size of plot caption.

`hline` Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).

`hlineType` Line type for hline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').

`hlineCol` Colour of hline.

`hlineWidth` Width of hline.

`vline` Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., c(60,90).

`vlineType` Line type for vline ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').

`vlineCol` Colour of vline.

`vlineWidth` Width of vline.

`gridlines.major` Logical, indicating whether or not to draw major gridlines.

`gridlines.minor` Logical, indicating whether or not to draw minor gridlines.

`borderWidth` Width of the border on the x and y axes.

`borderColour` Colour of the border on the x and y axes.

`returnPlot` Logical, indicating whether or not to return the plot object.

`flip_axes` Logical, indicating whether to swap the x- and y-axes.

Value

A ggplot2

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
PCAplot(se)
```

`plotPCAloadings` *plotPCAloadings*

Description

Plot of results from PCA loadings from getLoadings

Usage

```
plotPCAloadings(PCA, se)
```

Arguments

`PCA`	A pca object from pca
`se`	A SummarizedExperiment object

Value

A ggplot of the PC1 and PC2 loadings.

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
plotPCAloadings(runPCA(se), se)
```

prepareCor *prepareCor*

Description

Calculate pairwise logFC correlations with getCor, returned in long or wide (corrr) format

Usage

```
prepareCor(
  corTibble,
  DGEdata,
  cd,
  format = c("wide", "long"),
  .progress = TRUE
)
```

Arguments

corTibble	A dataframe used to specify correlations through the following columns: x, y, name
DGEdata	A list of DGE results
cd	colData from a SummarizedExperiment object
format	Character specifying the format of the returned data frame, "wide" (the default) or "long"
.progress	Whether to show a progress bar. Use TRUE to turn on a basic progress bar, use a string to give it a name, or see progress_bars for more details.

Value

A dataframe

References

Laufer BI, Friedman BA (2026). gg4way: 4way Plots of Differential Expression. R package. doi:10.18129/B9.bioc.gg4way

Kuhn M, Jackson S, Cimentada J (2025). corrr: Correlations in R. R package. doi:10.32614/CRAN.package.corrr

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
corRes <- tidyr::crossing(x = names(airwayDGE), y = names(airwayDGE)) %>%
```

```
    dplyr::filter(x != y) %>%
    dplyr::mutate(name = paste(stringr::str_remove(x, " vs .*"), "vs",
                               stringr::str_remove(y, " vs .*"))) %>%
    prepareCor(airwayDGE, cd = SummarizedExperiment::colData(se))
```

---

printCorTibble *printCorTibble*

Description

Print the corTibble

Usage

```
printCorTibble(corTibble, BiocStyle = FALSE)
```

Arguments

corTibble	A tibble of tidied correlations
BiocStyle	Logical indicating whether the table is rendered in a BiocStyle vignette, where the per-column filter boxes misalign, so they are dropped in favour of the global search

Value

A datatable

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
corRes <- tidyr::crossing(x = names(airwayDGE), y = names(airwayDGE)) %>%
    dplyr::filter(x != y) %>%
    dplyr::mutate(name = paste(stringr::str_remove(x, " vs .*"), "vs",
                               stringr::str_remove(y, " vs .*"))) %>%
    prepareCor(airwayDGE, cd = SummarizedExperiment::colData(se),
               format = "long")
printCorTibble(corRes)
```

printGSEAtibble *printGSEAtibble*

Description

Print a table of significant GSEA results

Usage

```
printGSEAtibble(GSEAlist, BiocStyle = FALSE)
```

Arguments

GSEAlist	A named list of GSEA result tibbles from tidyGSEAlist
BiocStyle	Logical indicating whether the table is rendered in a BiocStyle vignette, where the per-column filter boxes misalign, so they are dropped in favour of the global search

Value

A datatable

Examples

```
data(airwayGSEA)
printGSEAtibble(airwayGSEA)
```

printHeatmaps *Print heatmaps*

Description

Create tabs of heatmaps in R Markdown

Usage

```
printHeatmaps(.x, .y, depth = 2)
```

Arguments

.x	plot
.y	title
depth	Number of # to use

Details

printHeatmaps

Value

NULL; called for its side effect of printing a markdown header followed by the heatmap.

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
data(airwayDGE)
list(DEGs = basicHeatmap(se, sigTibble = makeSigTibble(airwayDGE))) %>%
    purrr::iwalk(printHeatmaps)
```

---

printPlots *Print ggplots*

Description

Create tabs of ggplots in R Markdown

Usage

```
printPlots(.x, .y, depth = 2)
```

Arguments

.x	plot
.y	title
depth	Number of # to use

Details

printPlots

Value

NULL; called for its side effect of printing a markdown header followed by the plot.

Examples

```
data(airwayGSEA)
airwayGSEA %>%
    GSEAplotList() %>%
    purrr::iwalk(printPlots)
```

printSigTibble *printSigTibble*

Description

Print the sigTibble

Usage

```
printSigTibble(sigTibble, rearrange = TRUE, BiocStyle = FALSE)
```

Arguments

sigTibble	A tibble of tidied significant DGE results
rearrange	Logical indicating whether to sort by contrast and FDR
BiocStyle	Logical indicating whether the table is rendered in a BiocStyle vignette, where the per-column filter boxes misalign, so they are dropped in favour of the global search

Value

A datatable

Examples

```
data(airwayDGE)
airwayDGE %>%
    makeSigTibble() %>%
    printSigTibble()
```

runPCA *runPCA*

Description

Wrapper for generating a PCA with PCAtools via pca

Usage

```
runPCA(se, assay = "logCPM", removeVar = 0.1)
```

Arguments

se	A SummarizedExperiment object
assay	Assay from se to use
removeVar	Argument to remove lower percent of variables for the pca

Value

A pca object.

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
runPCA(se)
```

---

save_cache *Save a cache stamped with its freshness key*

Description

Saves named objects to `file` together with the content-hash `key` produced by `cache_key()`, so `load_if_fresh()` can later verify the cache matches its current inputs before loading it.

Usage

```
save_cache(file, key, ...)
```

Arguments

`file`	Path to write the `.RData` cache to.
`key`	The freshness key from `cache_key()`.
`...`	Named objects to store in the cache (e.g. `GSEAlist = GSEAlist`).

Value

Invisibly, the `file` path.

Examples

```
se0 <- augere.de::loadExampleDataset()
data(airwayGSEA)
key <- cache_key(SummarizedExperiment::colData(se0), rownames(se0))
file <- tempfile(fileext = ".RData")
save_cache(file, key, GSEAlist = airwayGSEA)
```

scaleRows *scaleRows*

Description

Scale a matrix by rows to generate Z-scores for a heatmap

Usage

```
scaleRows(mat, verbose = TRUE)
```

Arguments

mat	A Matrix
verbose	Logical indicating whether to print range of Z-scores

Value

A matrix of row z-scores (the rows of mat centred and scaled).

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
SummarizedExperiment::assay(se, "logCPM")[getHVGs(se, n = 20, symbols = FALSE), ] %>%
    scaleRows()
```

tidyGSEAlist *tidyGSEAlist*

Description

Create a named list of GSEA tibbles for downstream functions

Usage

```
tidyGSEAlist(GSEAlist, GSEAmethod = "camera")
```

Arguments

GSEAlist	A named list of runContrast results, one per contrast
GSEAmethod	Character specifying name of GSEA method to extract

Value

A list of tibbles

References

Wu D, Smyth GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research*, 40(17), e133. doi:10.1093/nar/gks461

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
gene.sets <- getReactome("Homo sapiens")
list(N61311 = se %>%
         augere.gsea::runContrast(sets = gene.sets,
                                  groups = "cell",
                                  block = "dex",
                                  comparison = c("N61311", "N052611"),
                                  methods = "camera",
                                  assay = "counts",
                                  output.dir = tempfile(),
                                  save.results = FALSE)) %>%
    tidyGSEAlist()
```

`tidySE` *tidySE*

Description

Tidy and normalize a SummarizedExperiment by adding assays for downstream analyses and visualization. Genes are filtered with filterByExpr, TMM normalization factors are calculated with normLibSizes, and the expression (`E`) matrix from voomWithQualityWeights (the quality weights are not stored) becomes the `logCPM` assay, with `NA` for the filtered genes. When `adjustBatch = TRUE`, a `ComBat` assay holds voom values from ComBat_seq-adjusted counts.

Usage

```
tidySE(
  se,
  group = "Group",
  covariates = NULL,
  block = NULL,
  groups = se %>% SummarizedExperiment::colData() %>% as.data.frame() %>%
    purrr::pluck(group) %>% forcats::as_factor() %>% levels() %>%
    purrr::set_names(),
  adjustBatch = FALSE
)
```

Arguments

se	A SummarizedExperiment with a counts assay
group	Character specifying the colData column to tidy into the Group factor
covariates	Character vector of colData columns with numeric covariates
block	Character vector of colData columns with blocking factors
groups	Named character vector for group factor recoding
adjustBatch	Logical indicating whether a Batch colData column should be adjusted for with ComBat_seq in an additional ComBat assay

Details

The input se must have a counts assay. Downstream basicBulk functions expect a rowData ID column and, for symbol-based plots and labelled heatmaps, a symbol column. The logCPM/ComBat assays are NA for genes that did not pass filterByExpr.

Value

The input se with an added logCPM (and ComBat) assay, a retained rowData column, and Group/Sample/lib.size/norm.fa in the colData

References

Chen Y, Lun ATL, Smyth GK (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. *F1000Research*, 5, 1438. doi:10.12688/f1000research.8987.2

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(3), R25. doi:10.1186/gb2010113r25

Chen Y, Chen L, Lun ATL, Baldoni P, Smyth GK (2025). edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets. *Nucleic Acids Research*, 53(2), gkaf018. doi:10.1093/nar/gkaf018

Law CW, Chen Y, Shi W, Smyth GK (2014). voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2), R29. doi:10.1186/gb2014152r29

Liu R, Holik AZ, Su S, Jansz N, Chen K, Leong HS, Blewitt ME, Asselin-Labat ML, Smyth GK, Ritchie ME (2015). Why weight? Modelling sample and observational level variability improves power in RNA-seq analyses. *Nucleic Acids Research*, 43(15), e97. doi:10.1093/nar/gkv412

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

See Also

runVoom

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
tidySE(se0, group = "cell", block = "dex")
```

`tidyVoom` *tidyVoom*

Description

Tidy a single contrast's DGE stats into a tibble for downstream functions

Usage

```
tidyVoom(contrast, se)
```

Arguments

`contrast`	A DataFrame of voom results for one contrast from runVoom
`se`	A SummarizedExperiment object

Value

A tibble

Examples

```
se0 <- augere.de::loadExampleDataset()
SummarizedExperiment::rowData(se0)$ID <- rownames(se0)
se <- tidySE(se0, group = "cell", block = "dex")
se %>%
    augere.de::runVoom(groups = "cell",
                       block = "dex",
                       comparisons = list(c("N61311", "N052611")),
                       assay = "counts",
                       output.dir = tempfile(),
                       save.results = FALSE) %>%
    magrittr::extract2("results") %>%
    magrittr::extract2(1) %>%
    tidyVoom(se)
```

`topGSEAplot` *topGSEAplot*

Description

A plot of the top GSEA terms from each contrast

Usage

```
topGSEAplot(GSEAlist, n_terms = 2)
```

Arguments

GSEAlist	A list of results from GSEA
n_terms	Numeric indicating number of terms per a direction from each contrast

Value

A ggplot2

Examples

```
data(airwayGSEA)
topGSEAplot(airwayGSEA)
```

Index

∗ **datasets**
    airwayDGE, 3
    airwayGSEA, 4

airway, *3, 4*
airwayDGE, 3
airwayGSEA, 4
assert_fresh, 4

barStyle, 5
basicHeatmap, 6, *18*, *29*
basicMA, 8
basicUpset, 9
basicVolcano, 10
biplot, *32*
boxPlot, 11

cache_key, 12
cache_key(), *4*, *27*, *42*
colorRamp3, 13
ComBat_seq, *44, 45*
coord_flip, *22*
corrr, *37*
corTriplot, 13

DataFrame, *28*, *46*
DEGbar, 14, *29*
dendrogram, *7*
dotPlot, 15
draw, *7*

extractHeatmap, 18
extractUpSet, 18

filterByExpr, *44, 45*
fmt_e, 19
fromList2, 19

geneHistogram, 20
genePlot, 21
getCor, *37*
getHVGs, 23
getLoadings, *36*
getReactome, 24, *29*
getReactome(), *4*
ggboxplot, *11*
ggplotly, *11*, *16*
ggprism::add_pvalue, *22*
ggpubr::ggstripchart, *16*
ggstripchart, *15*
glue, *22*
GSEAplot, 25
GSEAplotList, 25

hclust, *7*
Heatmap, *7*
HeatmapAnnotation, *7*
HeatmapList, *7*

joinFeatures, *21*, 26

load_if_fresh, 27
load_if_fresh(), *4*, *12*, *42*

makeDGEdata, 28
makeDGEdata(), *3*
makePathwayTibble, *24*, 29
makeSigCountTibble, *15*, 29
makeSigTibble, *29*, 30
msigdbr, *24*

normLibSizes, *44*

pca, *31, 32*, *36*, *41, 42*
PCAgrid, 31
PCAplot, 32
PCAtools::biplot, *33*
plotPCAloadings, 36
position_dodge, *23*
prepareCor, *13, 14*, 37
printCorTibble, 38
printGSEAtibble, *29*, 39

printHeatmaps, 39
printPlots, 40
printSigTibble, 41
progress_bars, *37*

rowAnnotation, *7*
runContrast, *24*, *43*
runPCA, 41
runVoom, *28*, *45, 46*

save_cache, 42
save_cache(), *12*, *27*
scale_color_d3, *20*
scaleRows, 43
SummarizedExperiment, *6*, *18*, *20, 21*, *23*, *26*, *28*, *31, 32*, *36, 37*, *41*, *44–46*

tidyGSEAlist, *24, 25*, *39*, 43
tidyGSEAlist(), *4*
tidySE, 44
tidyVoom, 46
topGSEAplot, 46

upset, *9*

voomWithQualityWeights, *44*