Package 'augere.screen'

November 10, 2026

Version 0.99.3

Date 2026-11-06

Title Automatic Generation of Functional Screen Analyses

Description
Implements pipelines for generating functional screen analysis reports in the augere framework. This uses voom to test for differential abundance of barcodes with consolidation into gene-level statistics.
Each pipeline function generates a self-contained Rmarkdown report with all of the steps required to reproduce the analysis.

License MIT + file LICENSE

Imports stats, limma, edgeR, metapod, S4Vectors, augere.core, augere.de

Suggests testthat, knitr, rmarkdown, BiocStyle, SummarizedExperiment

VignetteBuilder knitr

Encoding UTF-8

URL https://github.com/augere-bioinfo/augere.screen

BugReports https://github.com/augere-bioinfo/augere.screen/issues

biocViews WorkflowManagement, ReportWriting, GeneTarget, FunctionalGenomics

Config/roxygen2/version 8.1.1

git_url https://git.bioconductor.org/packages/augere.screen

git_branch devel

git_last_commit b434d9b

git_last_commit_date 2026-11-05

Repository Bioconductor 3.24

Date/Publication 2026-11-09

Author Aaron Lun [cre, aut] (ORCID: <https://orcid.org/0000-0002-3564-4813>)

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

Contents

augere.screen-package 2
reexports 2
runScreen 3

Index 9

`augere.screen-package` *augere.screen: Automatic Generation of Functional Screen Analyses*

Description

Implements pipelines for generating functional screen analysis reports in the augere framework. This uses voom to test for differential abundance of barcodes with consolidation into gene-level statistics. Each pipeline function generates a self-contained Rmarkdown report with all of the steps required to reproduce the analysis.

Author(s)

Maintainer: Aaron Lun <infinite.monkeys.with.keyboards@gmail.com> (ORCID)

Authors:

- Aaron Lun <infinite.monkeys.with.keyboards@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/augere-bioinfo/augere.screen
- Report bugs at https://github.com/augere-bioinfo/augere.screen/issues

---

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

augere.core `readResult()`, `wrapInput()`

runScreen *Differential abundance for functional screens*

Description

Use voom to test for differential abundance of barcodes in a functional screen.

Usage

```
runScreen(
  x,
  groups,
  comparisons,
  covariates = NULL,
  block = NULL,
  subset.factor = NULL,
  subset.levels = NULL,
  subset.groups = TRUE,
  design = NULL,
  contrasts = NULL,
  dc.block = NULL,
  robust = TRUE,
  trend = FALSE,
  quality = TRUE,
  lfc.threshold = 0,
  filter.reference.factor = NULL,
  filter.reference.levels = NULL,
  filter.num.mads = 3,
  filter.default = TRUE,
  norm.control.barcodes = NULL,
  norm.control.genes = NULL,
  norm.control.type.field = NULL,
  norm.control.types = NULL,
  norm.tmm = TRUE,
  gene.field = NULL,
  report.summary = TRUE,
  summary.threshold = 0.05,
  consolidation = c("simes", "fry", "holm-min"),
  fry.args = list(),
  holm.min.n = 3,
  holm.min.prop = 0.3,
  assay = 1,
  row.data = gene.field,
  gene.data = NULL,
  metadata = NULL,
  output.dir = "voom",
  author = NULL,
  dry.run = FALSE,
  save.results = TRUE,
  suppress.plots = FALSE
)
```

Arguments

x	A SummarizedExperiment object containing read counts for each barcode (row) and sample (column). Alternatively, the output of wrapInput that refers to a SummarizedExperiment.
groups	String specifying the colData(x) column containing the grouping factor of interest, see processSimpleDesignMatrix for more details. This may be NULL for experimental designs with no groups, e.g., covariates only. Ignored if design and contrasts are provided.
comparisons	Character vector specifying two or more groups to compare from groups, or the covariate to be tested from covariates. Alternatively, or a list of such character vectors specifying multiple comparisons to perform. See processSimpleComparisons for more details. Ignored if design and contrasts are provided.
covariates	Character vector specifying the colData(x) columns containing continuous covariates of interest, see processSimpleDesignMatrix for more details. Ignored if design and contrasts are provided.
block	Character vector specifying the colData(x) columns containing additional (uninteresting) blocking factors, see processSimpleDesignMatrix for more details. Ignored if design and contrasts are provided.
subset.factor	String specifying the colData(x) column containing the factor to use for subsetting.
subset.levels	Vector containing the levels of the subset.factor to be retained.
subset.groups	Boolean indicating whether to automatically subset the dataset to only those samples assigned to groups in comparisons. Setting this to TRUE sacrifices some residual degrees of freedom for greater robustness against variability in irrelevant groups. Ignored if design and contrasts are provided. Also ignored if covariates is provided, as all samples are informative for a continuous covariate.
design	Matrix, function, or formula specifying the experimental design, see processCustomDesignMatrix for details. If this and contrasts are specified, groups, block, covariates, comparisons and subset.groups are ignored.
contrasts	String, function or matrix specifying a custom contrast, or a list of such objects; see processCustomContrasts for details. If this and design are specified, groups, block, covariates, comparisons and subset.groups are ignored.
dc.block	String specifying the blocking factor to use in duplicateCorrelation. Typically used for uninteresting factors that cannot be used in block as they are confounded with the factors of interest. No additional blocking is performed if NULL.
robust	Boolean indicating whether robust empirical Bayes shrinkage should be used in eBayes. Setting this to TRUE sacrifices some precision for improved robustness against genes with extreme variances.
trend	Boolean indicating whether variances should be shrunk towards a trend in eBayes. Usually unnecessary as the observation weights already account for the mean-variance relationship.
quality	Boolean indicating whether quality weighting should be performed. This reduces the influence of low-quality samples at the cost of more computational work.
lfc.threshold	Number specifying a threshold on the log-fold change in treat. If zero, eBayes will be used instead.

`filter.reference.factor`
: String specifying `colData(se)` column that indicates whether a sample is a reference. This is used to filter out barcodes that are not present in the original barcode library.

`filter.reference.levels`
: Character vector containing the reference levels in the `filter.reference.factor` column. Only used if `filter.reference.factor` is supplied.

`filter.num.mads`
: Number specifying the number of median absolute deviations below the median to define an outlier threshold for filtering barcodes. Only used if `filter.reference.factor` is supplied.

`filter.default`
: Boolean indicating whether to filter barcodes using the default **edgeR** method, i.e., `filterByExpr`. If `TRUE` and `filter.reference.factor` is supplied, the default filtering is applied in addition to the reference-based filtering. Otherwise, it is applied alone.

`norm.control.barcodes`
: A named list of character vectors containing one or more sets of control barcodes. Each vector should contain the names of barcodes, corresponding to the row names of `se`. The names of the vectors will be stored in the metadata.

`norm.control.genes`
: A named list containing one or more sets of control genes. Each vector should contain the names of genes, corresponding to the entries of the `gene.field` column. The names of the vectors will be stored in the metadata. Only used if `gene.field` is supplied.

`norm.control.type.field`
: String specifying the `rowData(se)` column containing the feature type for each barcode.

`norm.control.types`
: Character vector containing the control feature types in the `norm.type.field` column. Only used if `norm.type.field` is supplied.

`norm.tmm`
: Boolean indicating whether TMM normalization should be used. If `FALSE`, normalization is instead performed using the library size for each sample.

`gene.field`
: String specifying the `rowData(se)` column that contains the gene identity for each barcode. If `NULL`, consolidation of barcode-level statistics into per-gene inferences will not be performed.

`report.summary`
: Boolean indicating whether to report barcode-level summaries within the per-gene data frames. Only used if `gene.field` is supplied.

`summary.threshold`
: Number specifying the FDR threshold when summarizing the number of significant barcodes per gene. Only used if `report.summary=TRUE`.

`consolidation`
: Character vector specifying the consolidation method(s) to convert per-barcode statistics into per-gene results. This can be zero, one or more of:

  - `"simes"`, which uses `groupedSimes` to combine barcode-level p-values into a single per-gene p-value. This is the most sensitive approach for detecting a small number of barcodes (possibly just 1) that are differentially abundant.
  - `"holm-min"`, which uses `groupedHolmMin` to combine barcode-level p-values into a single per-gene p-value. This should be used when a minimum number/proportion of barcodes for a gene is expected to exhibit differential abundance.

- `"fry"`, which uses `fry` to test for differential abundance across all barcodes for a gene. This favors consistent differences across the majority of barcodes for a given gene. However, it is not compatible with ANOVA-like contrasts and will ignore any `lfc.threshold > 0`.

Only used if `gene.field` is supplied.

`fry.args` Named list of additional arguments to pass to `fry`.

`holm.min.n` Integer specifying the minimum number of significant barcodes per gene when `consolidation="holm"`, see the `min.sig.n=` in `?groupedHolmMin` for details.

`holm.min.prop` Integer specifying the minimum proportion of significant barcodes per gene when `consolidation="holm"`, see the `min.sig.prop=` in `?groupedHolmMin` for details.

`assay` String or integer specifying the assay of `x` containing the count matrix.

`row.data` Character vector specifying the `rowData(x)` columns containing extra gene annotations to include in the result DataFrames.

`gene.data` Character vector containing the names of `rowData(se)` columns to add to the per-gene result tables. If a gene has multiple barcodes, the row corresponding to the first barcode is used.

`metadata` Named list of additional metadata to store alongside each result.

`output.dir` String containing the path to an output directory in which to write the Rmarkdown file and save results.

`author` Character vector containg the names of the authors. If `NULL`, defaults to the current user.

`dry.run` Boolean indicating whether to perform a dry run. This generates the Rmarkdown report in `output.dir` but does not execute the analysis.

`save.results` Boolean indicating whether the results should be saved to file.

`suppress.plots` Boolean indicating whether plots should be suppressed. This can be set to `TRUE` for faster execution.

Value

A Rmarkdown report named `report.Rmd` is written inside `output.dir` that contains the analysis commands.

If `dry.run=FALSE`, a list is returned containing:

- `barcode`, a named list of DataFrames containing barcode-level result tables. Each DataFrame corresponds to a comparison/contrast where each row corresponds to a barcode (i.e., row) in `se`. Each DataFrame contains the following columns:
  - `AveAb`, the average abundance.
  - `t`, the t-statistic. (For non-ANOVA-like contrasts only.)
  - `F`, the F-statistic. (For ANOVA-like contrasts only.)
  - `LogFC`, the log-fold change. (For non-ANOVA-like contrasts only.)
  - `LogFC.<COLUMN>`, the log-fold change corresponding to each column of the contrast matrix. (For ANOVA-like contrasts only.)
  - `PValue`, the p-value;
  - `FDR`, the Benjamini-Hochberg-adjusted p-value.
- `gene`, a named list of lists containing one entry per method in `consolidation`. Each inner list contains one DataFrame per contrast where each row corresponds to a gene (not barcode). Each DataFrame contains at least the following columns:

  - NumBarcodes, the number of barcodes per gene (possibly after filtering).
  - PValue, the p-value for differential abundance across all barcodes for the gene.
  - FDR, the Benjamini-Hochberg-adjusted p-value.
  - Direction, the overall direction of change for all barcodes of the gene. (For non-ANOVA-like contrasts only.)
  - NumUp, the number of barcodes with a significant increase in abundance. (For non-ANOVA-like contrasts only, when report.summary=TRUE.)
  - NumDown, the number of barcodes with a significant decrease in abundance. (For non-ANOVA-like contrasts only, when report.summary=TRUE.)
  - LogFC, the log-fold change of the barcode with the lowest p-value. (For non-ANOVA-like contrasts only, when report.summary=TRUE.)
  - NumSig, the number of significant barcodes. (For ANOVA-like contrasts only, when report.summary=TRUE.)
  - AveAb, the average abundance of the barcode with the lowest p-value.

  Only reported if gene.field is not NULL.

- normalized, a copy of x with normalized abundance values, possibly subsetted by sample. This contains:
  - lib.size and norm.factors columns in its colData, containing the library sizes and normalization factors, respectively.
  - a retained column in its rowData, indicating whether a gene was retained after filtering.
  - a logcounts assay, containing the log-normalized counts.

  normalized may be subsetted by sample, depending on subset.factor, subset.group, etc.

If save.results=TRUE, the results are saved in a results directory inside output.

If dry.run=TRUE, NULL is returned. Only the Rmarkdown report is saved to file.

Reference filtering

A reference sample should represent the barcode distribution before any perturbation, e.g., time zero in a time course. All barcodes should have similar abundances in the reference samples as they should be present at the same molarity in the original barcode library. If a barcode has very low abundance, it was probably missing from the original library due to some manufacturing defect.

To identify these low outliers, we compute the average abundance across reference samples for each barcode. We define a filtering threshold based on the median absolute deviation (MAD) below the median of the average abundances across barcodes. Barcodes with average abundances below this threshold are discarded.

We also use filterByExpr to remove low-abundance barcodes. This ensures that all remaining barcodes have sufficiently large counts for voom, especially if the reference samples themselves are not used in the rest of the analysis.

Control normalization

Control barcodes might not target any gene (non-targeting controls, or NTCs) or they might target non-essential genes (NEGs). We assume that such barcodes do not genuinely change in abundance across conditions, allowing us to use them to compute normalization factors. The choice of control barcodes can be made by one or more of the norm.barcodes, norm.genes and norm.types arguments. If more than one of these modes is provided, the union of barcodes is used.

By default, we use TMM normalization (see calcNormFactors) on the control barcodes. This avoids composition biases and provides some robustness against changes in abundance due to inadvertent biological activity. If norm.tmm=FALSE, we instead use the total sum of counts across the

controls to derive normalization factors. This can be more precise but is more sensitive to genuine changes in abundance.

If no control barcodes are specified, normalization uses all barcodes that remain after filtering. If `norm.tmm=TRUE`, TMM normalization is performed on all (filtered) barcodes. Otherwise, the total sum of counts across all (filtered) barcodes is used.

Author(s)

Aaron Lun, Jean-Philippe Fortin

Examples

```
# Creating an example dataset.
mu <- 2^runif(1000, 0, 5)
grouping <- rep(c("A", "B", "ctrl"), each=3)
counts <- matrix(rnbinom(length(mu)* length(grouping), mu=mu, size=20), ncol=length(grouping))
rownames(counts) <- sprintf("BARCODE-%s", seq_along(mu))

library(SummarizedExperiment)
se <- SummarizedExperiment(list(counts=counts))
rowData(se)$type <- sample(c("NTC", "NEG", "other"), length(mu), replace=TRUE)
rowData(se)$gene <- paste0("GENE-", sample(LETTERS, length(mu), replace=TRUE))
colData(se)$group <- grouping

output <- tempfile()
res <- runScreen(
    se,
    groups="group",
    comparisons=c("A", "B"),
    norm.control.type.field="type",
    norm.control.types="NEG",
    filter.reference.factor="group",
    filter.reference.levels="ctrl",
    gene.field="gene",
    output.dir=output
)

list.files(output, recursive=TRUE)
res$barcode[[1]]
res$gene$simes[[1]]
res$normalized
```

Index

∗ **internal**
reexports, 2

augere.screen *(*augere.screen-package*)*, 2
augere.screen-package, 2

calcNormFactors, *7*
colData, *4, 5*, *7*

DataFrame, *6*
duplicateCorrelation, *4*

eBayes, *4*

filterByExpr, *5*, *7*
fry, *6*

groupedHolmMin, *5, 6*
groupedSimes, *5*

processCustomContrasts, *4*
processCustomDesignMatrix, *4*
processSimpleComparisons, *4*
processSimpleDesignMatrix, *4*

readResult *(*reexports*)*, 2
readResult(), *2*
reexports, 2
rowData, *5–7*
runScreen, 3

SummarizedExperiment, *4*

treat, *4*

voom, *3*, *7*

wrapInput, *4*
wrapInput *(*reexports*)*, 2
wrapInput(), *2*