Package ‘GSABenchmark’

April 21, 2026

Type Package

Title Tools for benchmarking single-cell gene set analysis methods

Version 0.99.7

Description GSABenchmark is a package designed for benchmarking scRNA-seq gene set analysis (scGSA) methods. It provides both traditional and novel benchmark metrics, as well as visualization tools. Currently, GSABenchmark supports 17 scGSA methods.

License MIT + file LICENSE

Imports abdiv, CSOA, decoupleR, dplyr, escape, fabR, ggplot2, ggrepel, GSVA, hammers, henna, jaccard, lsa, Matrix, MLmetrics, methods, mltools, pagoda2, paletteer, reshape2, rlang, scLang, singscore, SiPSiC, stringr, stats, VAM, withr

Encoding UTF-8

RoxygenNote 7.3.3

Suggests AUCell, BiocStyle, knitr, qs2, ranger, rmarkdown, rpart, scater, scRNAseq, scuttle, Seurat, testthat (>= 3.0.0), UCell

biocViews Software, SingleCell, GeneSetEnrichment, GeneExpression, Visualization

VignetteBuilder knitr

URL https://github.com/andrei-stoica26/GSABenchmark

BugReports https://github.com/andrei-stoica26/GSABenchmark/issues

Config/testthat/edition 3

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

git_branch devel

git_last_commit 1767752

git_last_commit_date 2026-04-12

Repository Bioconductor 3.23

Date/Publication 2026-04-20

Author Andrei-Florian Stoica [aut, cre] (ORCID: <https://orcid.org/0000-0002-5253-0826>)

Maintainer Andrei-Florian Stoica <andreistoica@foxmail.com>

Contents

GSABenchmark-package . . . 3
accuracyBenchmark . . . 4
accuracyBenchmarkMultiple . . . 5
addGlobalAverage . . . 6
addMethodInfo . . . 6
addMetricSummary . . . 7
addModuleScoreHelper . . . 7
aggregateRankPlot . . . 8
aggregateRanks . . . 9
alignmentScore . . . 9
allBenchmarkPlots . . . 10
allBenchmarkResults . . . 10
allGeneSetRanks . . . 12
allMetricRanks . . . 12
allTopRatios . . . 13
benchmarkPlots . . . 13
benchmarkSummary . . . 14
binaryPredMetric . . . 15
binaryPredMetricCore . . . 15
bindSummary . . . 16
boundaryBenchmark . . . 17
boundaryBenchmarkMultiple . . . 17
boundaryMCCBenchmark . . . 18
computeBoundaryMetrics . . . 19
computeGlobalMetrics . . . 19
computeMCC . . . 20
computeMCCMetric . . . 20
computeMetricMeans . . . 21
corrPlots . . . 21
corrSummary . . . 22
directMCCBenchmark . . . 23
directMCCBenchmarkMultiple . . . 23
efficiencyBenchmark . . . 24
extractCellScores . . . 25
geneSetRankPlots . . . 25
geneSetRanks . . . 26
globalBenchmark . . . 27
globalBenchmarkMultiple . . . 28
globalBenchmarkSummary . . . 29
mdsPlots . . . 30
mdsScoreSummary . . . 31
memoryPlot . . . 31
metricRankPlots . . . 32
metricRanks . . . 33
plotList . . . 33
predJaccardPlots . . . 34

predJaccards 35
ratioPlot 36
runAddModuleScore 37
runAUCell 38
runBenchmark 38
runBenchmarkShuffle 39
runDecoupleRMethod 40
runEscapeMethod 41
runGSAMethods 42
runGSVA 43
runGSVAMethod 43
runJASMINE 44
runMDT 45
runMethodShuffle 46
runMLM 47
runORA 48
runPagoda2 48
runPLAGE 49
runSingscore 50
runSiPSiC 50
runssGSEA 51
runUCell 52
runUDT 52
runVAM 53
runZscore 54
scorePlot 55
supportedMethods 56
timeExpr 56
timePlot 57

Index **58**

`GSABenchmark-package` *GSABenchmark: Tools for benchmarking single-cell gene set analysis methods*

Description

GSABenchmark is a package designed for benchmarking scRNA-seq gene set analysis (scGSA) methods. It provides both traditional and novel benchmark metrics, as well as visualization tools. Currently, GSABenchmark supports 17 scGSA methods.

Author(s)

Maintainer: Andrei-Florian Stoica `<andreistoica@foxmail.com>` (ORCID)

See Also

Useful links:

- https://github.com/andrei-stoica26/GSABenchmark
- Report bugs at https://github.com/andrei-stoica26/GSABenchmark/issues

---

`accuracyBenchmark` *Perform an accuracy benchmark on a set of gene set analysis method scores*

Description

This function perform an accuracy benchmark on a set of gene set analysis method scores.

Usage

```
accuracyBenchmark(
  scObj,
  labelCol,
  scoreCol,
  label,
  checkLabel = TRUE,
  computeMetricsFun,
  ...
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`scoreCol`	The metadata column containing the gene set analysis method score.
`label`	The identity assessed from the labelCol column.
`checkLabel`	Whether to check that `label` is found in `labelCol`.
`computeMetricsFun`	Function used to compute metrics. Options are computeBoundaryMetrics and computeDistributionMetrics.
`...`	Additional parameters passed to computeMetricsFun.

Value

A benchmark data frame.

`accuracyBenchmarkMultiple`

Perform accuracy benchmarks for multiple sets of gene set analysis method scores and class labels

---

Description

This function performs accuracy benchmarks for multiple sets of gene set analysis method scores and class labels.

Usage

```
accuracyBenchmarkMultiple(
  scObj,
  labelCol,
  geneSetNames,
  gsaMethods,
  checkLabels = TRUE,
  verbose = FALSE,
  benchmarkFun,
  ...
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSetNames`	Gene set names.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`verbose`	Whether to output a message whenever the benchmarking begins for a gene set.
`benchmarkFun`	Benchmark function. Must be either `boundaryBenchmark` or `distributionBenchmark`.
`...`	Additional paramters passed to benchmarkFun.

Value

A list of lists of benchmark data frames.

`addGlobalAverage` *Add gene set averages to the global summary list*

Description

This function adds gene set averages to the global summary list.

Usage

```
addGlobalAverage(globalSmr, gsaMethods)
```

Arguments

`globalSmr`	Global summary.
`gsaMethods`	Character vector of gene set analysis methods.

Value

Extended summary list with an additional data frame showing the average results obtained for each gene set.

`addMethodInfo` *Add method information to a summary data frame*

Description

This function adds gene set analysis method information to a summary data frame. It sets the names of the gene set analysis method as the rownames of the data frame, and optionally computes the mean score for each method and decreasingly sorts the data frame by these scores.

Usage

```
addMethodInfo(df, gsaMethods, doAverage = TRUE)
```

Arguments

`df`	A data frame where the values represent the scores obtained by a gene set analysis method (row) on a gene set (column) for a metric.
`gsaMethods`	Character vector of gene set analysis methods.
`doAverage`	Whether to add an average column to each data frame, sorting each data frame decreasingly by the average column in the process.

Value

A summary data frame with added method information.

`addMetricSummary` *Add metric information to the summary list*

Description

This function adds metric information to the summary list and optionally extends the list by addding overall results for each metric.

Usage

```
addMetricSummary(smr, metrics, gsaMethods, doSummarize = TRUE)
```

Arguments

`smr`	List of result data frames for each metrics. Each data frame contains the results for each tested gene set analysis method for each gene set for the corresponding method.
`metrics`	Metrics.
`gsaMethods`	Character vector of gene set analysis methods.
`doSummarize`	Whether to add a metric summary. Must be set to `FALSE` when summarizing a MCC benchmark list of lists.

Value

Extended summary list with an additional data frame showing the average results obtained for each metric.

`addModuleScoreHelper` *Helper used to run AddModuleScore*

Description

This function is used to run `AddModuleScore`.

Usage

```
addModuleScoreHelper(
  scObj,
  geneSets,
  slot = "data",
  pool = rownames(scObj),
  nbin = 24,
  ctrl = 100
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`pool`	The set from which features to be compared with signature genes are selected. Defaults to all features.
`nbin`	Number of bins of aggregate expression levels for pool features.
`ctrl`	Number of control features chosen from the same bin for each feature.

Value

A single-cell expression object with the results saved as a metadata column.

`aggregateRankPlot` *Create an aggregate rank plot from a summary object*

Description

This function creates an aggregate rank plot from a summary object.

Usage

```
aggregateRankPlot(
  smr,
  title = NULL,
  xLab = "Method",
  sigDigits = 2,
  rankMethod = c("min", "average", "first", "last", "random", "max"),
  ...
)
```

Arguments

`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`title`	Plot title.
`xLab`	x axis label.
`sigDigits`	Number of significant digits used when displaying mean ranks. If `NULL`, the mean ranks will not be displayed.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.
`...`	Additional parameters passed to other functions.

Value

A ggplot object.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
p <- aggregateRankPlot(smr)
```

`aggregateRanks` *Compute aggregate ranks*

Description

This function computes aggregate ranks from a list of summary data frames.

Usage

```
aggregateRanks(
  smr,
  rankMethod = c("min", "average", "first", "last", "random", "max")
)
```

Arguments

`smr`	List containing boundary, MCC and global lists of data frames.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.

Value

A data frame of aggregate summary results.

`alignmentScore` *Calculate the alignment between two numeric vectors of the same length*

Description

This function calculates the alignment between two numeric vectors of the same lenght.

Usage

```
alignmentScore(v, w)
```

Arguments

`v`	A vector.
`w`	A vector of the same length as

Value

Alignment score.

`allBenchmarkPlots` *Plot the complete list of benchmark summaries*

Description

This function plots the complete list of benchmark summaries.

Usage

```
allBenchmarkPlots(smr, titleSuffix = NULL, ...)
```

Arguments

`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`...`	Additional parameters passed to other functions.

Value

A list of ggplot objects.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- allBenchmarkPlots(smr)
```

`allBenchmarkResults` *Generate all benchmark results with some precomputed arguments. This function generates all benchmark results by using precomputed values for* `normSilDF`, `dimMat` *and* `maxDist`.

Description

Generate all benchmark results with some precomputed arguments. This function generates all benchmark results by using precomputed values for `normSilDF`, `dimMat` and `maxDist`.

Usage

```
allBenchmarkResults(
  scObj,
  labelCol,
  geneSets,
  gsaMethods,
  checkLabels = TRUE,
  normSilDF = NULL,
  dimMat = NULL,
  maxDist = NULL,
  efBenchmark = NULL,
  runEFBenchmark = TRUE,
  verbose = TRUE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A list of gene sets. If not `NULL` while `efBenchmark` is `NULL`, the efficiency benchmark will be run.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`normSilDF`	Data frame of normalized silhouettes.
`dimMat`	UMAP dimensionality reduction matrix of the single-cell expression object. Unused if `normSilDF` is `NULL`.
`maxDist`	Maximum UMAP distance in the single-cell expression object. Unused if `normSilDF` or `dimMat` is `NULL`.
`efBenchmark`	A list of data frames generated with efficiencyBenchmark.
`runEFBenchmark`	Whether to run efficiency benchmark.
`verbose`	Whether the output of the efficiency benchmark should be verbose. Ignored if `runEFBenchmark` is `FALSE`.

Details

This function is designed to save some computational time when multiple runs are sequentially performed for the same single-cell expression object and cell identity classes, but different choices of gene sets (so long as they still describe the same classes) or methods. If a single run is planned rather than multiple ones, `runBenchmark` is a more straightforward choice, as it takes care of generating `normSilDF` and `dimMat`.

Value

A list of benchmark results.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- hammers::computeSilhouette(scObj, 'label')
normSilDF <- hammers::normalizeSilhouette(scObj, 'label')
dimMat <- scLang::scPCAMat(scObj)
maxDist <- max(dist(dimMat))
smr <- allBenchmarkResults(scObj, 'label', geneSets, c('CSOA', 'Zscore'),
TRUE, normSilDF, dimMat, maxDist, runEFBenchmark=FALSE)
```

`allGeneSetRanks` *Extract all gene set ranks list of summary data frames*

Description

This function extracts all gene set ranks list from a list of summary data frames.

Usage

```
allGeneSetRanks(smr, rankMethod = "min")
```

Arguments

`smr`	List of summary data frames.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.

Value

List of gene set ranks.

`allMetricRanks` *Extract all metric ranks list of summary data frames*

Description

This function extracts metric ranks from a list of summary data frames.

Usage

```
allMetricRanks(smr, rankMethod = "min")
```

Arguments

`smr`	List of summary data frames.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.

Value

List of metric ranks.

`allTopRatios` *Extract all ratio ranks list of summary data frames*

Description

This function extracts all ratio ranks list from a list of summary data frames.

Usage

```
allTopRatios(smr, nItems = 25)
```

Arguments

`smr`	List of summary data frames.
`nItems`	Number of retained iterms.

Value

List of gene set ranks.

`benchmarkPlots` *Plot a list of summary data frames*

Description

This function plots a list of summary data frames with methods as rows, gene sets and the average of scores across all gene sets as columns.

Usage

```
benchmarkPlots(smr, titleSuffix = NULL, lastPlotLegendLab = "Metric", ...)
```

Arguments

`smr`	List of summary data frames, whether boundary, MCC or global.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`lastPlotLegendLab`	The legend label of the last plot.
`...`	Additional arguments passed to `scorePlot`.

Value

A list of ggplot objects.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- benchmarkPlots(smr[[1]])
```

`benchmarkSummary` *Compares the identification of different labels through different gene set analysis methods*

Description

This function compares the identification of different labels through different gene set analysis methods.

Usage

```
benchmarkSummary(benchmarkLL, doAverage = TRUE, doSummarize = TRUE)
```

Arguments

`benchmarkLL`	A list of lists of benchmark data frames.
`doAverage`	Whether to add an average column to each data frame, sorting each data frame decreasingly by the average column in the process.
`doSummarize`	Whether to add a metric summary. Must be set to `FALSE` when summarizing a MCC benchmark list of lists.

Value

A list of summary data frames.

`binaryPredMetric` *Generate metric results using binary predictions*

Description

This function generates metric results using binary predictions.

Usage

```
binaryPredMetric(
  globalSmr,
  geneSetNames,
  gsaMethods,
  predList,
  smrCol,
  metricFun
)
```

Arguments

`globalSmr`	Global summary.
`gsaMethods`	Character vector of gene set analysis methods.
`predList`	A list of data frames of the same length as the number of gene sets, in which predictions made by each gene set analysis method are represented as binary columns.
`smrCol`	Summary column to be added
`metricFun`	Metric function. Choose between `jaccard` and `numCosine`.

Value

A global summary object with an added binary prediction data frame.

`binaryPredMetricCore` *Generate metric results for a gene set using binary predictions*

Description

This function generates metric results for a gene set using binary predictions.

Usage

```
binaryPredMetricCore(gsName, gsaMethods, predList, metricFun)
```

Arguments

`gsName`	Gene set name.
`gsaMethods`	Character vector of gene set analysis methods.
`predList`	A list of data frames of the same length as the number of gene sets, in which predictions made by each gene set analysis method are represented as binary columns.
`metricFun`	Metric function. Choose between `jaccard` and `numCosine`.

Value

A numeric vector.

`bindSummary` *Extract results list of summary data frames*

Description

This function extracts results from a list of summary data frames.

Usage

```
bindSummary(smr, nAggDFs = 0, nAvgCols = 0)
```

Arguments

`smr`	List of summary data frames.
`nAggDFs`	Number of data frames of aggregate results in the list of summary data frames.
`nAvgCols`	Number of average columns for each data frame in the list.

Value

A data frame of aggregate summary results.

`boundaryBenchmark` *Perform a class boundary accuracy benchmark on a set of GSA method scores*

Description

This function performs a class boundary accuracy benchmark on a set of GSA method scores.

Usage

```
boundaryBenchmark(scObj, labelCol, scoreCol, label, checkLabel = TRUE)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`scoreCol`	The metadata column containing the gene set analysis method score.
`label`	The identity assessed from the labelCol column.
`checkLabel`	Whether to check that `label` is found in `labelCol`.

Value

A benchmark data frame with cells as row names, and labels, GSA scores, and class boundary metric scores as columns.

`boundaryBenchmarkMultiple` *Perform boundary accuracy benchmarks for multiple sets of gene set analysis method scores and class labels*

Description

This function performs boundary accuracy benchmarks for multiple sets of gene set analysis method scores and class labels.

Usage

```
boundaryBenchmarkMultiple(
  scObj,
  labelCol,
  geneSetNames,
  gsaMethods,
  checkLabels = TRUE,
  verbose = FALSE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSetNames`	Gene set names.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`verbose`	Whether to output a message whenever the benchmarking begins for a gene set.

Value

A list of lists of boundary benchmark data frames.

`boundaryMCCBenchmark` *Compute the MCC based on the previously identified boundary cutoff*

Description

This function computes the MCC based on the previously identified boundary cutoff.

Usage

```
boundaryMCCBenchmark(boundaryBenchmarkLL)
```

Arguments

`boundaryBenchmarkLL`	A list of lists of data frames generated with `boundaryBenchmarkMultiple`.

Value

A benchmark data frame with cells as row names, and labels, GSA scores, and MCC scores as columns.

`computeBoundaryMetrics` *Compute the class boundary metrics*

Description

This function computes the class boundary metrics.

Usage

```
computeBoundaryMetrics(df)
```

Arguments

`df`	A data frame consisting of cells with a binary label column (1) and a column containing the scores obtained by a gene set analysis method (2).

Value

A benchmark data frame with cells as row names, labels in the first column, gene set analysis method scores in the second column, and computed metric scores in the remaining columns.

`computeGlobalMetrics` *Compute the global evaluation metrics*

Description

This function computes the global evaluation metrics.

Usage

```
computeGlobalMetrics(df, normSilDF = NULL, dimMat = NULL, maxDist = NULL)
```

Arguments

`df`	A data frame consisting of cells with a binary label column (1) and a column containing the scores obtained by a gene set analysis method (2).
`normSilDF`	Data frame of normalized silhouettes.
`dimMat`	UMAP dimensionality reduction matrix of the single-cell expression object. Unused if `normSilDF` is `NULL`.
`maxDist`	Maximum UMAP distance in the single-cell expression object. Unused if `normSilDF` or `dimMat` is `NULL`.

Value

A benchmark data frame with one row and computed metric scores in the columns.

`computeMCC` *Compute the Matthews correlation coefficient*

Description

This function computes the Matthews correlation coefficient for an input data frame generated through class boundary determination benchmarking. The first column contains labels while the second is binarized based on the previously determined cutoff (the first element in the second column).

Usage

```
computeMCC(boundaryResDF, threshold = boundaryResDF[1, 2])
```

Arguments

`boundaryResDF`	A data frame generated with the boundary benchmark.
`threshold`	Threshold that determines class boundary.

Value

The Matthews correlation coefficient.

`computeMCCMetric` *Compute the MCC at each threshold*

Description

This function computes the MCC at each threshold.

Usage

```
computeMCCMetric(df)
```

Arguments

`df`	A data frame consisting of cells with a binary label column (1) and a column containing the scores obtained by a gene set analysis method (2).

Value

A benchmark data frame with cells as row names, labels in the first column, gene set analysis method scores in the second column, and MCC scores in the third column.

`computeMetricMeans` *Add overall scores to a benchmark data frame*

Description

This function adds overlap scores to a benchmark data frame.

Usage

```
computeMetricMeans(df, startCol)
```

Arguments

`df`	Benchmark data frame.
`startCol`	Column where metric scores start.

Value

The input data frame sorted decreasingly by the newly added overall column.

`corrPlots` *Create correlation plots for method results*

Description

This function creates correlation plots for method results.

Usage

```
corrPlots(
  scObj,
  smr,
  titlePrefix = "Correlation plot",
  titleInfixes = NULL,
  titleSuffix = NULL,
  ...
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.

`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`...`	Additional arguments passed to `henna::correlationPlot`.

Value

A named list of ggplot objects.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- corrPlots(scObj, smr)
```

`corrSummary` *Calculate correlation matrix for method results*

Description

This function calculates the correlation matrix for all the methods for an identity class.

Usage

```
corrSummary(scObj, smr, corMethod = "spearman")
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`smr`	List containing boundary, MCC and global lists of data frames.
`corMethod`	Correlation method.

Value

A list of correlation matrices.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
m <- corrSummary(scObj, smr)
```

`directMCCBenchmark` *Perform a direct MCC benchmark on a set of gene set analysis method scores*

Description

This function performs a direct MCC benchmark on a set of gene set analysis method scores.

Usage

```
directMCCBenchmark(scObj, labelCol, scoreCol, label, checkLabel = TRUE)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`scoreCol`	The metadata column containing the gene set analysis method score.
`label`	The identity assessed from the labelCol column.
`checkLabel`	Whether to check that `label` is found in `labelCol`.

Value

A benchmark data frame with cells as row names, and labels, gene set analysis scores, and MCC scores as columns.

`directMCCBenchmarkMultiple` *Perform direct MCC benchmarks for multiple sets of gene set analysis method scores and class labels*

Description

This function performs direct MCC benchmarks for multiple sets of gene set analysis method scores and class labels.

Usage

```
directMCCBenchmarkMultiple(
  scObj,
  labelCol,
  geneSetNames,
  gsaMethods,
  checkLabels = TRUE,
  verbose = FALSE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSetNames`	Gene set names.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`verbose`	Whether to output a message whenever the benchmarking begins for a gene set.

Value

A list of lists of boundary benchmark data frames.

`efficiencyBenchmark` *Extract running times and peak memory usage for gene set analysis methods*

Description

This function extracts running times and peak memory usage for gene set analysis methods.

Usage

```
efficiencyBenchmark(
  scObj,
  labelCol,
  geneSets,
  gsaMethods,
  checkLabels = TRUE,
  verbose = TRUE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A list of gene sets. If not `NULL` while `efBenchmark` is `NULL`, the efficiency benchmark will be run.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`verbose`	Whether output should be verbose.

Value

A list of two data frames, the first comprising running times in seconds, the second comprising peak memory usage in mebibytes.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
ef <- efficiencyBenchmark(scObj, 'label', geneSets[1], 'CSOA',
verbose=FALSE)
```

`extractCellScores` *Extract gene set analysis method scores and truth labels from single-cell expression object*

Description

This function extracts gene set analysis method scores and truth labels from a single-cell expression object.

Usage

```
extractCellScores(scObj, labelCol, scoreCol, label)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`scoreCol`	The metadata column containing the gene set analysis method score.
`label`	The identity assessed from the labelCol column.

Value

A data frame with two columns: truth labels (1 or 0) and gene set analysis method scores.

`geneSetRankPlots` *Create gene set rank plots for the method results*

Description

This function creates gene set rank plots for method results.

Usage

```
geneSetRankPlots(
  smr,
  rankMethod = "min",
  titlePrefix = "Distribution of ranks",
  titleInfixes = NULL,
  titleSuffix = NULL,
  xLab = "Method",
  ...
)
```

Arguments

`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`xLab`	x axis label.
`...`	Additional arguments passed to `henna::rankPlot`.

Value

A named list of ggplot objects.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- geneSetRankPlots(smr)
```

`geneSetRanks` *Extract gene set ranks list from a list of summary data frames*

Description

This function extracts gene set ranks list from a list of summary data frames.

Usage

```
geneSetRanks(
  smr,
  nAggDFs = 0,
  nAvgCols = 0,
  rankMethod = c("min", "average", "first", "last", "random", "max")
)
```

Arguments

`smr`	List of summary data frames.
`nAggDFs`	Number of data frames of aggregate results in the list of summary data frames.
`nAvgCols`	Number of average columns for each data frame in the list.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.

Value

List of gene set ranks.

`globalBenchmark` *Perform a global evaluation benchmark on a set of gene set analysis method scores*

Description

This function performs a global evaluation benchmark on a set of gene set analysis method scores.

Usage

```
globalBenchmark(
  scObj,
  labelCol,
  scoreCol,
  label,
  checkLabel = TRUE,
  normSilDF = NULL,
  dimMat = NULL,
  maxDist = NULL
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`scoreCol`	The metadata column containing the gene set analysis method score.
`label`	The identity assessed from the labelCol column.
`checkLabel`	Whether to check that `label` is found in `labelCol`.
`normSilDF`	Data frame of normalized silhouettes.
`dimMat`	UMAP dimensionality reduction matrix of the single-cell expression object. Unused if `normSilDF` is `NULL`.
`maxDist`	Maximum UMAP distance in the single-cell expression object. Unused if `normSilDF` or `dimMat` is `NULL`.

Value

A one-row benchmark data frame with global metric scores as columns.

`globalBenchmarkMultiple` *Perform distribution accuracy benchmarks for multiple sets of gene set analysis method scores and class labels*

Description

This function performs distribution accuracy benchmarks for multiple sets of gene set analysis method scores and class labels.

Usage

```
globalBenchmarkMultiple(
  scObj,
  labelCol,
  geneSetNames,
  gsaMethods,
  checkLabels = TRUE,
  verbose = FALSE,
  normSilDF = NULL,
  dimMat = NULL,
  maxDist = NULL
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSetNames`	Gene set names.
`gsaMethods`	Character vector of gene set analysis methods.
`checkLabels`	Whether to check that `geneSetNames` is a subset of the values in `labelCol`.
`verbose`	Whether to output a message whenever the benchmarking begins for a gene set.
`normSilDF`	Data frame of normalized silhouettes.
`dimMat`	UMAP dimensionality reduction matrix of the single-cell expression object. Unused if `normSilDF` is `NULL`.
`maxDist`	Maximum UMAP distance in the single-cell expression object. Unused if `normSilDF` or `dimMat` is `NULL`.

Value

A list of lists of distribution benchmark data frames.

`globalBenchmarkSummary` *Compares the identification of different labels through different gene set analysis methods*

Description

This function compares the identification of different labels through different gene set analysis methods.

Usage

```
globalBenchmarkSummary(globalBenchmarkLL, geneSetNames, gsaMethods, predList)
```

Arguments

`globalBenchmarkLL`	A list of lists of benchmark data frames.

Value

A list of summary data frames.

`mdsPlots` *Create MDS plots for method results*

Description

This function creates MDS plots for method results.

Usage

```
mdsPlots(
  scObj,
  smr,
  titlePrefix = "MDS plot",
  titleInfixes = NULL,
  titleSuffix = NULL,
  drawScores = TRUE,
  ...
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`drawScores`	Whether to render scores on the plot. If set to `TRUE`, the third column of the input data frame will be numeric and scores will be taken from there.
`...`	Additional arguments passed to `henna::densityPlot`.

Value

A named list of ggplot objects.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- mdsPlots(scObj, smr)
```

`mdsScoreSummary` *Compute the scored MDS summary*

Description

This function computes the score MDS summary. For each gene set, MDS is performed for the cell-level method predicted scores. The resulting two-column data frame is appended the average accuracy benchmark score of each method, obtained on the gene set.

Usage

```
mdsScoreSummary(scObj, smr)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`smr`	List containing boundary, MCC and global lists of data frames.

Value

A list containing a data frame with three columns (two MDS coordinates and method score) for each gene set.

`memoryPlot` *Plot a data frame consisting of gene set analysis method peak memory usage This function plots data frame consisting of method peak memory usages with methods as rows, gene sets and the gene set average as columns.*

Description

Plot a data frame consisting of gene set analysis method peak memory usage This function plots data frame consisting of method peak memory usages with methods as rows, gene sets and the gene set average as columns.

Usage

```
memoryPlot(efBenchmark, title = NULL, xLab = "Peak memory usage (MiB)", ...)
```

Arguments

`efBenchmark`	A list of dataframes generated with `efficiencyBenchmark`.
`title`	Plot title.
`xLab`	x axis label.
`...`	Additional parameters to be passed to `scorePlot`.

Value

A ggplot object.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
memoryPlot(smr[[5]])
```

`metricRankPlots` *Create metric rank plots for the method results*

Description

This function creates metric rank plots for method results.

Usage

```
metricRankPlots(
  smr,
  titlePrefix = "Distribution of ranks",
  titleInfixes = metricNames(),
  titleSuffix = NULL,
  rankMethod = "min",
  xLab = "Method",
  ...
)
```

Arguments

`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.
`xLab`	x axis label.
`...`	Additional arguments passed to `henna::rankPlot`.

Value

A named list of ggplot objects.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- metricRankPlots(smr)
```

`metricRanks` *Extract metric ranks list from a list of summary data frames*

Description

This function extracts metric ranks from a list of summary data frames.

Usage

```
metricRanks(smr, nAggDFs = 0, nAvgCols = 0, rankMethod = "min")
```

Arguments

`smr`	List of summary data frames.
`nAggDFs`	Number of data frames of aggregate results in the list of summary data frames.
`nAvgCols`	Number of average columns for each data frame in the list.
`rankMethod`	Rank method. Choose between 'min', 'average', 'first', 'last', 'random' and 'max'. Default in 'min'.

Value

List of metric ranks.

`plotList` *Create a list of plots from a list of arguments*

Description

This function creates a list of plots from a list of arguments by first converting the arguments into results that can be plotted using `calcFun`, then plotting these results using `plotFun`.

Usage

```
plotList(
  args,
  calcFun,
  plotFun,
  unlistPlotArgs = FALSE,
  titlePrefix = NULL,
  titleInfixes = NULL,
  titleSuffix = NULL,
  ...
)
```

Arguments

`args`	A list of arguments passed to `calcFun`.
`calcFun`	The function used for generating the results that will be plotted
`plotFun`	The function used for plotting.
`unlistPlotArgs`	Whether the additional arguments passed to `plotFun` have to be unlisted (that is, they are provided as a list).
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`...`	Additional arguments passed to `plotFun`.

Value

A named list of ggplot objects.

`predJaccardPlots` *Create Jaccard tile plots for method binary predictions*

Description

This function creates Jaccard tile plots for method binary predictions.

Usage

```
predJaccardPlots(
  predictionsSmr,
  titlePrefix = "Binary prediction Jaccard plot",
  titleInfixes = NULL,
  titleSuffix = NULL,
  ...
)
```

Arguments

`predictionsSmr`	Binary predictions summary.
`titlePrefix`	Prefix used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleInfixes`	A character vector of title infixes used to create a title with `subplotTitle`. Ignored if `titleSuffix` is `NULL`.
`titleSuffix`	Suffix used to create plot titles If `NULL`, the plots will have no titles.
`...`	Additional arguments passed to `henna::tilePlot`.

Value

A named list of ggplot objects.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
plots <- predJaccardPlots(smr$predictions)
```

`predJaccards` *Calculate the Jaccard scores of method binary predictions*

Description

This function calculates the Jaccard scores of method binary predictions.

Usage

```
predJaccards(predictionsSmr)
```

Arguments

`predictionsSmr`	Binary predictions summary.

Value

A list of numeric matrices.

`ratioPlot` *Create a ratio rank plot for the method results*

Description

This function creates a ratio rank plot for the method results.

Usage

```
ratioPlot(
  smr,
  title = NULL,
  nItems = 25,
  xLab = "Maximum over mean ratio",
  yLab = "Metric",
  legendLab = "Method",
  ...
)
```

Arguments

`smr`	Complete summary list generated with `allBenchmarkResults` or `runBenchmark`.
`title`	Plot title.
`nItems`	Number of retained iterms.
`xLab`	x axis label.
`yLab`	y axis label.
`legendLab`	Legend label.
`...`	Additional arguments passed to `henna::classRank`.

Value

A ggplot object.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
p <- ratioPlot(smr)
```

`runAddModuleScore` *Run AddModuleScore*

Description

This function runs `AddModuleScore`.

Usage

```
runAddModuleScore(
  scObj,
  geneSets,
  slot = "data",
  pool = rownames(scObj),
  nbin = 24,
  ctrl = 100,
  seed = 1
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`pool`	The set from which features to be compared with signature genes are selected. Defaults to all features.
`nbin`	Number of bins of aggregate expression levels for pool features.
`ctrl`	Number of control features chosen from the same bin for each feature.
`seed`	Random seed.

Details

Reimplemented from `https://github.com/satijalab`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runAddModuleScore(scObj, geneSets)
```

`runAUCell` *Run AUCell*

Description

This function runs `AUCell` using `escape`.

Usage

```
runAUCell(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runEscapeMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
if (requireNamespace("AUCell", quietly=TRUE)){
    scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
    scObj <- qs2::qs_read(scoPath)
    gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
    geneSets <- qs2::qs_read(gsPath)
    scObj <- runAUCell(scObj, geneSets)
    }
```

`runBenchmark` *Generate all benchmark results This function performs the entire* `GSABenchmark` *pipeline.*

Description

Generate all benchmark results This function performs the entire `GSABenchmark` pipeline.

Usage

```
runBenchmark(
  scObj,
  labelCol,
  geneSets,
  gsaMethods,
  runEFBenchmark = TRUE,
  verbose = TRUE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A list of gene sets. If not `NULL` while `efBenchmark` is `NULL`, the efficiency benchmark will be run.
`gsaMethods`	Character vector of gene set analysis methods.
`runEFBenchmark`	Whether to run efficiency benchmark.
`verbose`	Whether the output of the efficiency benchmark should be verbose. Ignored if `runEFBenchmark` is `FALSE`.

Details

A wrapper around `allBenchmarkResults`. Slower for repeated runs, but it does not require users to manually generate `normSilDF` and `dimMat`.

Value

A list of benchmark results.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
smr <- runBenchmark(scObj, 'label', geneSets, c('CSOA', 'Zscore'), FALSE)
```

`runBenchmarkShuffle` *Generate all benchmark results for shuffled gene sets This function generates all benchmark results for shuffled gene sets.*

Description

Generate all benchmark results for shuffled gene sets This function generates all benchmark results for shuffled gene sets.

Usage

```
runBenchmarkShuffle(
  scObj,
  labelCol,
  geneSets,
  gsaMethod,
  runEFBenchmark = TRUE
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A list of gene sets. If not `NULL` while `efBenchmark` is `NULL`, the efficiency benchmark will be run.
`gsaMethod`	Name of the gene set analysis method.
`runEFBenchmark`	Whether to run efficiency benchmark.

Value

A list of benchmark results.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runMethodShuffle(scObj, 'label', geneSets, 'CSOA', 0.2, 0.2)
smr <- runBenchmarkShuffle(scObj, 'label', geneSets, 'CSOA', FALSE)
```

`runDecoupleRMethod` *Run a gene set analysis method using decoupleR*

Description

This function runs one of the gene set analysis methods supported by `decoupleR`.

Usage

```
runDecoupleRMethod(scObj, geneSets, method, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`method`	Gene set analysis method.
`slot`	Gene expression slot.
`...`	Additional arguments passed to gene set analysis method.

Value

A single-cell expression object with the results saved as a metadata column.

`runEscapeMethod` *Run a gene set analysis method using escape*

Description

This function runs one of the gene set analysis methods supported by `escape`.

Usage

```
runEscapeMethod(scObj, geneSets, method, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`method`	Gene set analysis method.
`slot`	Gene expression slot.
`...`	Additional arguments passed to gene set analysis method.

Value

A single-cell expression object with the results saved as a metadata column.

`runGSAMethods` *Run gene set analysis methods*

Description

This function runs the gene set analysis methods.

Usage

```
runGSAMethods(
  scObj,
  labelCol,
  geneSets,
  gsaMethods,
  infix = NULL,
  outputFun = silently_run
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A named list of gene sets.
`gsaMethods`	Character vector of gene set analysis methods.
`infix`	Infix to add between method name and gene set name in single-cell expression object. The string consisting of the method name and the infix is separated by the gene set name with a '_' character.
`outputFun`	Choose between silently_run (suppress all warnings and messages) or identity (do not suppress them). Default is silently_run.

Value

A `Seurat` or `SingleCellExpression` object with the results of the runs stored as metadata columns.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runGSAMethods(scObj, 'label', geneSets, c('CSOA', 'Zscore'))
```

`runGSVA` *Run GSVA*

Description

This function runs `GSVA` using `GSVA`.

Usage

```
runGSVA(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runGSVAMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runGSVA(scObj, geneSets)
```

`runGSVAMethod` *Run a gene set analysis method using* `GSVA`

Description

This function runs one of the gene set analysis methods supported by `GSVA`.

Usage

```
runGSVAMethod(
  scObj,
  geneSets,
  method,
  slot = "data",
  invert = FALSE,
  filter = FALSE,
  ...
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`method`	Gene set analysis method.
`slot`	Gene expression slot.
`invert`	Whether to transform the scores from x to 1 - x.
`filter`	Whether to filter the expression matrix as to contain only signature genes.
`...`	Additional parameters passed to other functions.

Value

A single-cell expression object with the results saved as a metadata column.

`runJASMINE` *Run JASMINE*

Description

This function runs `JASMINE`.

Usage

```
runJASMINE(
  scObj,
  geneSets,
  slot = "data",
  method = c("oddsratio", "likelihood")
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`method`	One of 'oddsratio' and 'likelihood'.

Details

Reimplemented from `https://github.com/NNoureen/JASMINE`. The method's paper can be found at `https://doi.org/10.7554/eLife.71994`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runJASMINE(scObj, geneSets)
```

`runMDT` *Run MDT using decoupleR*

Description

This function runs `MDT` using `decoupleR`.

Usage

```
runMDT(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runDecoupleRMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
if (requireNamespace("ranger", quietly=TRUE)){
    scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
    scObj <- qs2::qs_read(scoPath)
    gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
    geneSets <- qs2::qs_read(gsPath)
    scObj <- runMDT(scObj, geneSets)
}
```

`runMethodShuffle` *Run gene set analysis method on shuffled gene sets*

Description

This function runs a gene set analysis method on shuffled gene sets.

Usage

```
runMethodShuffle(
  scObj,
  labelCol,
  geneSets,
  gsaMethod,
  loss = c(0, 0.2),
  noise = c(0, 0.2),
  doGrid = TRUE,
  averageReplicates = TRUE,
  seeds = c(1, 2, 3),
  outputFun = identity
)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`labelCol`	The metadata column containing the ground truth annotation.
`geneSets`	A named list of gene sets.
`gsaMethod`	Name of the gene set analysis method.
`loss`	A numeric vector of gene loss values. Must be in [0, 1).
`noise`	A numeric vector of noise values. Must be in [0, 1).
`doGrid`	Whether to run the methods for each loss-noise combination.
`averageReplicates`	Whether to average replicates. If `TRUE`, a single set of scores will be returned for each loss-noise combination for which the method is run. If `FALSE`, distinct sets of scores will be returned for each replicate.
`seeds`	A numeric vector of random seeds passed to `hammers::shuffleGenes`. Its length determines the number of replicates.
`outputFun`	Choose between silently_run (suppress all warnings and messages) or identity (do not suppress them). Default is silently_run.

Value

A `Seurat` or `SingleCellExpression` object with the results of the runs stored as metadata columns.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runMethodShuffle(scObj, 'label', geneSets, 'CSOA', 0.2, 0.2)
```

`runMLM` *Run MLM using decoupleR*

Description

This function runs `MLM` using `decoupleR`.

Usage

```
runMLM(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runDecoupleRMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runMLM(scObj, geneSets)
```

`runORA` *Run ORA using decoupleR*

Description

This function runs `ORA` using `decoupleR`.

Usage

```
runORA(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runDecoupleRMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runORA(scObj, geneSets)
```

`runPagoda2` *Run pagoda2*

Description

This function runs `pagoda2`.

Usage

```
runPagoda2(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional arguments passed to `pagoda2::score.cells.puram`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runPagoda2(scObj, geneSets)
```

`runPLAGE` *Run PLAGE*

Description

This function runs `PLAGE` using `GSVA`.

Usage

```
runPLAGE(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runGSVAMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runPLAGE(scObj, geneSets)
```

`runSingscore` *Run Singscore*

Description

This function runs `Singscore`.

Usage

```
runSingscore(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional arguments passed to `singscore::simpleScore`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runSingscore(scObj, geneSets)
```

`runSiPSiC` *Run SiPSiC*

Description

This function runs `SiPSiC`.

Usage

```
runSiPSiC(scObj, geneSets, slot = "counts", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional arguments passed to `SiPSiC::getPathwayScores`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runSiPSiC(scObj, geneSets)
```

`runssGSEA` *Run ssGSEA*

Description

This function runs `ssGSEA` using `escape`.

Usage

```
runssGSEA(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runEscapeMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runssGSEA(scObj, geneSets)
```

`runUCell` *Run UCell*

Description

This function runs `UCell` using `escape`.

Usage

```
runUCell(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runEscapeMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
if (requireNamespace("UCell", quietly=TRUE)){
    scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
    scObj <- qs2::qs_read(scoPath)
    gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
    geneSets <- qs2::qs_read(gsPath)
    scObj <- runUCell(scObj, geneSets)
    }
```

`runUDT` *Run UDT using decoupleR*

Description

This function runs `UDT` using `decoupleR`.

Usage

```
runUDT(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runDecoupleRMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
if (requireNamespace("rpart", quietly=TRUE)){
    scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
    scObj <- qs2::qs_read(scoPath)
    gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
    geneSets <- qs2::qs_read(gsPath)
    scObj <- runUDT(scObj, geneSets)
    }
```

`runVAM` *Run VAM*

Description

This function runs `VAM`.

Usage

```
runVAM(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional arguments passed to `VAM::vam`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runVAM(scObj, geneSets)
```

`runZscore` *Run Zscore*

Description

This function runs `Zscore` using `GSVA`.

Usage

```
runZscore(scObj, geneSets, slot = "data", ...)
```

Arguments

`scObj`	A `Seurat` or `SingleCellExperiment` object.
`geneSets`	A named list of gene sets.
`slot`	Gene expression slot.
`...`	Additional parameters passed to `runGSVAMethod`.

Value

A single-cell expression object with the results saved as a metadata column.

Examples

```
scoPath <- system.file('extdata', 'scObj.qs2', package='GSABenchmark')
scObj <- qs2::qs_read(scoPath)
gsPath <- system.file('extdata', 'geneSets.qs2', package='GSABenchmark')
geneSets <- qs2::qs_read(gsPath)
scObj <- runZscore(scObj, geneSets)
```

`scorePlot` *Plot a data frame consisting of gene set analysis method scores This function plots a data frame consisting of method scores with methods as rows, gene sets and the gene set average as columns.*

Description

Plot a data frame consisting of gene set analysis method scores This function plots a data frame consisting of method scores with methods as rows, gene sets and the gene set average as columns.

Usage

```
scorePlot(
  scoreDF,
  title = NULL,
  xLab = "Score",
  yLab = "Method",
  isDecreasing = FALSE,
  palette = "grDevices::Plasma",
  legendLab = "Gene set",
  pointSize = 1.5
)
```

Arguments

`scoreDF`	A summary data frame.
`title`	Plot title.
`xLab`	x axis label.
`yLab`	y axis label.
`isDecreasing`	Logical; whether the methods should be displayed on the plot in decreasing order of the obtained average scores. If `FALSE` (as default), the methods will be displayed in increasing order of the average scores.
`palette`	Color palette.
`legendLab`	Legend label.
`pointSize`	Point size.

Value

A ggplot object.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
scorePlot(smr[[1]][[1]])
```

`supportedMethods` *Show supported methods*

Description

This function shows the methods currently supported by GSABenchmark.

Usage

```
supportedMethods()
```

Value

A character vector of supported methods.

Examples

```
supportedMethods()
```

`timeExpr` *Time the evaluation of an expression*

Description

This function computes the time required to evaluate an expression.

Usage

```
timeExpr(expr, verbose = FALSE)
```

Arguments

`expr`	An expression to be evaluated.
`verbose`	Whether to display the elapsed time as output.

Value

A numeric value representing elapsed time.

`timePlot` *Plot a data frame consisting of gene set analysis method running times This function plots data frame consisting of method running times with methods as rows, gene sets and the gene set average as columns.*

Description

Plot a data frame consisting of gene set analysis method running times This function plots data frame consisting of method running times with methods as rows, gene sets and the gene set average as columns.

Usage

```
timePlot(efBenchmark, title = NULL, xLab = "Running time (s)", ...)
```

Arguments

`efBenchmark`	A list of dataframes generated with `efficiencyBenchmark`.
`title`	Plot title.
`xLab`	x axis label.
`...`	Additional parameters to be passed to `scorePlot`.

Value

A ggplot object.

Examples

```
sPath <- system.file('extdata', 'smr.qs2', package='GSABenchmark')
smr <- qs2::qs_read(sPath)
timePlot(smr[[5]])
```

Index

∗ **internal**
  accuracyBenchmark, 4
  accuracyBenchmarkMultiple, 5
  addGlobalAverage, 6
  addMethodInfo, 6
  addMetricSummary, 7
  addModuleScoreHelper, 7
  aggregateRanks, 9
  alignmentScore, 9
  allGeneSetRanks, 12
  allMetricRanks, 12
  allTopRatios, 13
  benchmarkSummary, 14
  binaryPredMetric, 15
  binaryPredMetricCore, 15
  bindSummary, 16
  boundaryBenchmark, 17
  boundaryBenchmarkMultiple, 17
  boundaryMCCBenchmark, 18
  computeBoundaryMetrics, 19
  computeGlobalMetrics, 19
  computeMCC, 20
  computeMCCMetric, 20
  computeMetricMeans, 21
  directMCCBenchmark, 23
  directMCCBenchmarkMultiple, 23
  extractCellScores, 25
  geneSetRanks, 26
  globalBenchmark, 27
  globalBenchmarkMultiple, 28
  globalBenchmarkSummary, 29
  GSABenchmark-package, 3
  mdsScoreSummary, 31
  metricRanks, 33
  plotList, 33
  predJaccards, 35
  runDecoupleRMethod, 40
  runEscapeMethod, 41
  runGSVAMethod, 43
  timeExpr, 56

accuracyBenchmark, 4
accuracyBenchmarkMultiple, 5
addGlobalAverage, 6
addMethodInfo, 6
addMetricSummary, 7
addModuleScoreHelper, 7
aggregateRankPlot, 8
aggregateRanks, 9
alignmentScore, 9
allBenchmarkPlots, 10
allBenchmarkResults, 10
allGeneSetRanks, 12
allMetricRanks, 12
allTopRatios, 13

benchmarkPlots, 13
benchmarkSummary, 14
binaryPredMetric, 15
binaryPredMetricCore, 15
bindSummary, 16
boundaryBenchmark, 17
boundaryBenchmarkMultiple, 17
boundaryMCCBenchmark, 18

computeBoundaryMetrics, 19
computeGlobalMetrics, 19
computeMCC, 20
computeMCCMetric, 20
computeMetricMeans, 21
corrPlots, 21
corrSummary, 22

directMCCBenchmark, 23
directMCCBenchmarkMultiple, 23

efficiencyBenchmark, 24
extractCellScores, 25

geneSetRankPlots, 25

geneSetRanks, 26
globalBenchmark, 27
globalBenchmarkMultiple, 28
globalBenchmarkSummary, 29
GSABenchmark *(GSABenchmark-package)*, 3
GSABenchmark-package, 3

mdsPlots, 30
mdsScoreSummary, 31
memoryPlot, 31
metricRankPlots, 32
metricRanks, 33

plotList, 33
predJaccardPlots, 34
predJaccards, 35

ratioPlot, 36
runAddModuleScore, 37
runAUCell, 38
runBenchmark, 38
runBenchmarkShuffle, 39
runDecoupleRMethod, 40
runEscapeMethod, 41
runGSAMethods, 42
runGSVA, 43
runGSVAMethod, 43
runJASMINE, 44
runMDT, 45
runMethodShuffle, 46
runMLM, 47
runORA, 48
runPagoda2, 48
runPLAGE, 49
runSingscore, 50
runSiPSiC, 50
runssGSEA, 51
runUCell, 52
runUDT, 52
runVAM, 53
runZscore, 54

scorePlot, 55
supportedMethods, 56

timeExpr, 56
timePlot, 57