

Package ‘lute’

November 21, 2024

Version 1.2.0

Title Framework for cell size scale factor normalized bulk transcriptomics deconvolution experiments

Description Provides a framework for adjustment on cell type size when performing bulk transcriptomics deconvolution. The main framework function provides a means of reference normalization using cell size scale factors. It allows for marker selection and deconvolution using non-negative least squares (NNLS) by default. The framework is extensible for other marker selection and deconvolution algorithms, and users may reuse the generics, methods, and classes for these when developing new algorithms.

License Artistic-2.0

Encoding UTF-8

URL <https://github.com/metamaden/lute>

BugReports <https://github.com/metamaden/lute/issues>

LazyData FALSE

Depends R (>= 4.3.0), stats, methods, utils, SummarizedExperiment, SingleCellExperiment, BiocGenerics

Imports S4Vectors, Biobase, scran, dplyr, ggplot2

Suggests nnls, knitr, testthat, rmarkdown, BiocStyle, GenomicRanges, limma, ExperimentHub, AnnotationHub, DelayedMatrixStats, BisqueRNA, DelayedArray

VignetteBuilder knitr

biocViews RNASeq, Sequencing, SingleCell, Coverage, Transcriptomics, Normalization

RoxygenNote 7.3.1

Collate 'lute_generics.R' 'deconvolutionParam-class.R'
'referencebasedParam-class.R' 'independentbulkParam-class.R'
'bisqueParam-class.R' 'typemarkersParam-class.R'
'findmarkersParam-class.R' 'globals.R'
'lute_cellScaleFactors.R' 'lute_classes.R' 'lute_conversions.R'
'lute_framework.R' 'lute_metadata.R' 'lute_randomized-data.R'
'lute_rmse.R' 'lute_rmf.R' 'lute_utilities.R'
'nnlsParam-class.R'

git_url <https://git.bioconductor.org/packages/lute>

git_branch RELEASE_3_20

git_last_commit 566aeec

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-20

Author Sean K Maden [cre, aut] (<<https://orcid.org/0000-0002-2212-4894>>),
Stephanie Hicks [aut] (<<https://orcid.org/0000-0002-7858-0231>>)

Maintainer Sean K Maden <maden.sean@gmail.com>

Contents

| | |
|---|----|
| bisqueParam | 3 |
| bisqueParam-class | 5 |
| cellProportionsPredictions | 6 |
| cellProportionsPredictions-class | 6 |
| deconvolution | 7 |
| deconvolution,bisqueParam-method | 8 |
| deconvolution,deconvolutionParam-method | 9 |
| deconvolution,independentbulkParam-method | 10 |
| deconvolution,nplsParam-method | 10 |
| deconvolution,referencebasedParam-method | 11 |
| deconvolutionParam-class | 12 |
| eset_to_sce | 13 |
| eset_to_se | 13 |
| findmarkersParam | 14 |
| findmarkersParam-class | 15 |
| getDeconvolutionExampleData | 15 |
| getDeconvolutionExampleDataBisque | 16 |
| getDeconvolutionExampleDataSCDC | 17 |
| get_celltypes_from_sce | 17 |
| get_csf_reference | 18 |
| get_eset_from_matrix | 19 |
| independentbulkParam | 19 |
| independentbulkParam-class | 20 |
| lute | 21 |
| luteSupportedDeconvolutionAlgorithms | 22 |
| new_workflow_table | 23 |
| nplsParam | 24 |
| nplsParam-class | 25 |
| parseDeconvolutionPredictionsResults | 26 |
| proportionsVectorsList | 26 |
| randomMarkersVectorsList | 27 |
| randomSingleCellExperiment | 28 |
| referencebasedParam | 29 |
| referencebasedParam-class | 30 |
| referenceFromSingleCellExperiment | 31 |
| rmse | 31 |
| rmseTest | 32 |
| sce_to_eset | 33 |
| sce_to_se | 33 |
| se_to_eset | 34 |

| | |
|--|----|
| se_to_sce | 34 |
| show,bisqueParam-method | 35 |
| show,cellProportionsPredictions-method | 35 |
| show,deconvolutionParam-method | 36 |
| show,findmarkersParam-method | 37 |
| show,independentbulkParam-method | 37 |
| show,nplsParam-method | 38 |
| show,referencebasedParam-method | 39 |
| show,typemarkersParam-method | 39 |
| typemarkers | 40 |
| typemarkers,findmarkersParam-method | 41 |
| typemarkers,typemarkersParam-method | 41 |
| typemarkersParam | 42 |
| typemarkersParam-class | 43 |
| ypb_from_sce | 43 |
| z_matrix_from_sce | 44 |
| [[,deconvolutionParam,ANY,ANY-method | 45 |
| [[,typemarkersParam,ANY,ANY-method | 46 |

Index 47

| | |
|-------------|---|
| bisqueParam | <i>Make new object of class bisqueParam</i> |
|-------------|---|

Description

Main constructor for class [bisqueParam](#).

Usage

```

bisqueParam(
  bulkExpression = NULL,
  bulkExpressionSet = NULL,
  bulkExpressionIndependent = NULL,
  referenceExpression = NULL,
  cellScaleFactors = NULL,
  scData = NULL,
  assayName = "counts",
  batchVariable = "batch.id",
  cellTypeVariable = "celltype",
  useOverlap = FALSE,
  returnInfo = FALSE
)

```

Arguments

`bulkExpression` Bulk expression matrix.

`bulkExpressionSet` ExpressionSet of bulk mixed signals.

`bulkExpressionIndependent` Bulk expression matrix of independent samples.

| | |
|---------------------|---|
| referenceExpression | Signature matrix of cell type-specific signals. If not provided, can be computed from a provided ExpressionSet containing single-cell data. |
| cellScaleFactors | size factor transformations of length equal to the K cell types to deconvolve. |
| scData | SummarizedExperiment-type object of single-cell transcriptomics data. Accepts ExpressionSet, SummarizedExperiment, and SingleCellExperiment object types. |
| assayName | Expression data type (e.g. counts, logcounts, tpm, etc.). |
| batchVariable | Name of variable identifying the batches in scData pData/coldata. |
| cellTypeVariable | Name of cell type labels variable in scData pData/coldata. |
| useOverlap | Whether to deconvolve samples overlapping bulk and sc esets (logical, FALSE). |
| returnInfo | Whether to return metadata and original method outputs with predicted proportions. |

Details

Takes standard inputs for the Bisque method. If user provides matrices, will convert these into ExpressionSet objects compatible with the main bisque method.

Value

New object of class `bisqueParam`.

Examples

```
## get data
exampleList <- getDeconvolutionExampleDataBisque()
bulkExpressionSet <- exampleList[["bulkExpressionSet"]][,seq(10)]
bulkExpression <- exprs(exampleList[["bulkExpressionSet"]])
bulkExpression <- bulkExpression[,c(11:ncol(bulkExpression))]

## get param object
newBisqueParameter <- bisqueParam(bulkExpressionSet=bulkExpressionSet,
                                   bulkExpressionIndependent=bulkExpression,
                                   scData=exampleList[["singleCellExpressionSet"]],
                                   batchVariable="SubjectName",
                                   cellTypeVariable="cellType",
                                   useOverlap=FALSE)

## get predicted proportions
deconvolutionResult <- deconvolution(newBisqueParameter)
```

bisqueParam-class *bisqueParam-class*

Description

Applies the `BisqueRNA::ReferenceBasedDecomposition()` implementation of the Bisque deconvolution algorithm.

Details

Main constructor for class `bisqueParam`.

Value

New object of class `bisqueParam`.

References

Brandon Jew and Marcus Alvarez (2021). BisqueRNA: Decomposition of Bulk Expression with Single-Cell Sequencing. CRAN, R package version 1.0.5. URL: <https://CRAN.R-project.org/package=BisqueRNA>

Brandon Jew et al. Accurate estimation of cell composition in bulk expression through robust integration of single-cell information. *Nat Commun* 11, 1971 (2020). <https://doi.org/10.1038/s41467-020-15816-6>

See Also

[deconvolutionParam](#), [referencebasedParam](#), [independentbulkParam](#)

Examples

```
## get data
exampleList <- getDeconvolutionExampleDataBisque()
bulkExpressionSet <- exampleList[["bulkExpressionSet"]][,seq(10)]
bulkExpression <- exprs(exampleList[["bulkExpressionSet"]])
bulkExpression <- bulkExpression[,c(11:ncol(bulkExpression))]

## get param object
newBisqueParameter <- bisqueParam(bulkExpressionSet=bulkExpressionSet,
                                   bulkExpressionIndependent=bulkExpression,
                                   scData=exampleList[["singleCellExpressionSet"]],
                                   batchVariable="SubjectName",
                                   cellTypeVariable="cellType",
                                   useOverlap=FALSE)

## get predicted proportions
res <- deconvolution(newBisqueParameter)
```

cellProportionsPredictions

Make new cellProportionsPredictions object.

Description

Make new cellProportionsPredictions object.

Usage

```
cellProportionsPredictions(  
  predictionsTable,  
  cellTypeVector = NULL,  
  sampleIdVector = NULL  
)
```

Arguments

predictionsTable Table of cell type predictions.
cellTypeVector Character vector of cell type labels.
sampleIdVector Character vector of sample id labels.

Value

New cellProportionsPredictions object.
New cellProportionsPredictions object.

Examples

```
exampleData <- getDeconvolutionExampleData()
```

cellProportionsPredictions-class

cellProportionsPredictions-class

Description

Class for cell type predictions.

Arguments

predictionsTable Table containing cell type predictions.
cellTypeVector Character vector of cell type labels.
sampleIdVector Character vector of sample id labels.

Details

Main constructor for class [cellProportionsPredictions](#).

Value

New cellProportionsPredictions object.

Examples

```
new("cellProportionsPredictions")
predictionsTable <- matrix(sample(100,50),nrow=10)
colnames(predictionsTable) <- paste0("cell_type",seq(ncol(predictionsTable)))
rownames(predictionsTable) <- paste0("sample", seq(nrow(predictionsTable)))
cellProportionsPredictions(predictionsTable)
```

deconvolution

deconvolution

Description

Get predicted cell type proportions using a deconvolution method.

Usage

```
deconvolution(object)
```

Arguments

object A [deconvolutionParam](#)-type object (see `?`deconvolutionParam-class``).

Details

This generic maps standard deconvolution inputs to the parameters of the specified deconvolution method for which a subclass of type [deconvolutionParam](#) exists. This generic uses a similar approach to the bluster R/Bioconductor package.

Value

By default, return named numeric vector of predicted proportions for each cell type.

If `returnInfo == TRUE`, instead returns a list including proportions, results object returned from specified method, and additional metadata.

Author(s)

Sean Maden

References

Aaron Lun. bluster: Clustering Algorithms for Bioconductor. (2022) Bioconductor, R package version 1.6.0.

See Also

[deconvolutionParam](#), [referencebasedParam](#), [independentbulkParam](#), [nnlsParam](#), [musicParam](#), [bisqueParam](#)

Examples

```
## get param object
exampleList <- getDeconvolutionExampleData()
param <- nnlsParam(cellScaleFactors=exampleList[["cellScaleFactors"]],
                  bulkExpression=exampleList[["bulkExpression"]],
                  referenceExpression=exampleList[["referenceExpression"]])

## run deconvolution
deconvolution(param)
```

deconvolution,bisqueParam-method

Deconvolution method for bisqueParam

Description

Main method to access the Bisque deconvolution method from the main lute deconvolution generic.

Usage

```
## S4 method for signature 'bisqueParam'
deconvolution(object)
```

Arguments

object Object of type [bisqueParam](#) (see `?bisqueParam`).

Details

Takes an object of class [bisqueParam](#) as input, returning a list.

Value

Either a vector of predicted proportions, or a list containing predictions, metadata, and original outputs.

References

Brandon Jew and Marcus Alvarez (2021). BisqueRNA: Decomposition of Bulk Expression with Single-Cell Sequencing. CRAN, R package version 1.0.5. URL: <https://CRAN.R-project.org/package=BisqueRNA>

Brandon Jew et al. Accurate estimation of cell composition in bulk expression through robust integration of single-cell information. Nat Commun 11, 1971 (2020). <https://doi.org/10.1038/s41467-020-15816-6>

Examples

```
## get data
exampleList <- getDeconvolutionExampleDataBisque()
bulkExpressionSet <- exampleList[["bulkExpressionSet"]][,seq(10)]
bulkExpression <- exprs(exampleList[["bulkExpressionSet"]])
bulkExpression <- bulkExpression[,c(11:ncol(bulkExpression))]

## get param object
newBisqueParameter <- bisqueParam(bulkExpressionSet=bulkExpressionSet,
                                  bulkExpressionIndependent=bulkExpression,
                                  scData=exampleList[["singleCellExpressionSet"]],
                                  batchVariable="SubjectName",
                                  cellTypeVariable="cellType",
                                  useOverlap=FALSE)

## get predicted proportions
deconvolutionResult <- deconvolution(newBisqueParameter)
```

deconvolution,deconvolutionParam-method

Deconvolution generic behavior for object of class [deconvolutionParam](#)

Description

Deconvolution generic behavior for object of class [deconvolutionParam](#)

Usage

```
## S4 method for signature 'deconvolutionParam'
deconvolution(object)
```

Arguments

object An object of class [deconvolutionParam](#) (see ?deconvolutionParam).

Details

Method for behavior of deconvolution generic when called for object of class [deconvolutionParam](#).

Value

Null method.

Examples

```
param <- new("deconvolutionParam")
deconvolution(param)
```

deconvolution,independentbulkParam-method

Deconvolution method for class [independentbulkParam](#)

Description

Function to perform standard operations prior to deconvolution (a.k.a. "deconvolution prep") for an object of class [independentbulkParam](#).

Usage

```
## S4 method for signature 'independentbulkParam'  
deconvolution(object)
```

Arguments

object An object of class [independentbulkParam](#).

Details

Takes an object of [independentbulkParam](#) class as input, and returns a list with the filtered/checked/parsed experiment objects.

Value

Method results.

Examples

```
new("independentbulkParam")
```

deconvolution,nnlsParam-method

Deconvolution method for nnlsParam

Description

Defines the deconvolution method for [nnlsParam](#).

Usage

```
## S4 method for signature 'nnlsParam'  
deconvolution(object)
```

Arguments

object An object of class [nnlsParam](#) (see ?nnlsParam).

Details

Takes an object of class `nlsParam` as input, returning either a list containing proportions, return info, and metadata, or a vector of predicted cell type proportions.

The key term mappings for this method include: * A : bulkExpression, bulk signals matrix (Y). * b : referenceExpression, signature matrix (Z).

Value

Either a vector of predicted proportions, or a list containing predictions, metadata, and original outputs.

References

Katharine M. Mullen and Ivo H. M. van Stokkum (2012). "nnls: The Lawson-Hanson algorithm for non-negative least squares (NNLS)." CRAN, R package version 1.4. URL: <https://cran.r-project.org/web/packages/nnls/index.html>

Examples

```
exampleList <- getDeconvolutionExampleData()
param <- nlsParam(
  cellScaleFactors=exampleList[["cellScaleFactors"]],
  bulkExpression=exampleList[["bulkExpression"]],
  referenceExpression=exampleList[["referenceExpression"]])

## return only predicted proportions
deconvolution(param)

# return full results
param@returnInfo <- TRUE
names(deconvolution(param))
```

deconvolution,referencebasedParam-method

Deconvolution generic behavior for object of class `referencebasedParam`

Description

Deconvolution generic behavior for object of class `referencebasedParam`

Usage

```
## S4 method for signature 'referencebasedParam'
deconvolution(object)
```

Arguments

`object` An object of class `referencebasedParam` (see `?referencebasedParam`).

Details

Method for behavior of deconvolution generic when called for object of class [referencebasedParam](#).

Value

Method results.

Examples

```
exampleList <- getDeconvolutionExampleData()
referencebasedParam(
  bulkExpression=exampleList$bulkExpression,
  referenceExpression=exampleList$referenceExpression,
  cellScaleFactors=exampleList$cellScaleFactors)
```

deconvolutionParam-class

deconvolutionParam-class

Description

Defines the principal parent class for all deconvolution method parameters.

Details

Defines the parent class for deconvolution method parameters. Since all deconvolution runs require a y signals matrix, whether from experiment data or simulations such as pseudobulking, this parent class manages the bulk signals matrix. For this class, the deconvolution generic performs basic summaries of the bulk signals matrix.

Value

New deconvolutionParam object.

See Also

deconvolution

Examples

```
param <- new("deconvolutionParam")
deconvolution(param)
```

eset_to_sce *eset_to_sce Convert ExpressionSet to SingleCellExperiment.*

Description

eset_to_sce Convert ExpressionSet to SingleCellExperiment.

Usage

```
eset_to_sce(expressionSet, assayName = "counts")
```

Arguments

expressionSet Object of type ExpressionSet (see ?ExpressionSet).
 assayName Name of new assay in new SingleCellExperiment object.

Value

ExpressionSet.

Examples

```
expressionSet <- getDeconvolutionExampleDataBisque()$singleCellExpressionSet
eset_to_sce(expressionSet)
```

eset_to_se *eset_to_se*

Description

Convert ExpressionSet to SummarizedExperiment.

Usage

```
eset_to_se(expressionSet, assayName = "counts")
```

Arguments

expressionSet Object of type ExpressionSet (see ?ExpressionSet).
 assayName Name of assay to store in new SummarizedExperiment object.

Value

New object of type SummarizedExperiment.

Examples

```
expressionSet <- getDeconvolutionExampleDataBisque()$singleCellExpressionSet
eset_to_se(expressionSet, "counts")
```

| | |
|------------------|--|
| findmarkersParam | <i>Make new object of class findmarkersParam</i> |
|------------------|--|

Description

Main constructor for class [findmarkersParam](#).

Usage

```
findmarkersParam(
  singleCellExperiment,
  assayName = "counts",
  cellTypeVariable = "cellType",
  testType = "wilcox",
  markersPerType = 20,
  returnInfo = FALSE
)
```

Arguments

| | |
|----------------------|--|
| singleCellExperiment | Object of type SingleCellExperiment (see ?SingleCellExperiment). |
| assayName | Name of expression matrix in SingleCellExperiment assays (e.g. "counts"). |
| cellTypeVariable | Name of cell type variable in SingleCellExperiment coldata. |
| testType | Test type (see ?findMarkers for options). |
| markersPerType | Number of top markers to get per cell type. |
| returnInfo | Whether to return metadata and original method outputs with predicted proportions. |

Details

Main class for mapping arguments to the findMarkers method implemented as `scan::findMarkers()`.

Value

Object of class [findmarkersParam](#)

See Also

[typemarkersParam](#)

Examples

```
exampleList <- getDeconvolutionExampleData()
singleCellExperimentExample <- randomSingleCellExperiment()
newParam <- findmarkersParam(singleCellExperiment=singleCellExperimentExample,
  cellTypeVariable="celltype", markersPerType=5)
markers <- typemarkers(newParam)
```

```
findmarkersParam-class
      findmarkersParam-class
```

Description

class definition for findmarkersParam, which uses scan::findMarkers()

Arguments

assayName Name of expression matrix in SingleCellExperiment assays (e.g. "counts").
singleCellExperiment Object of type SingleCellExperiment (see ?SingleCellExperiment).
cellTypeVariable Name of cell type variable in SingleCellExperiment coldata.
testType Test type (see ?findMarkers for options).

Details

Main constructor for class [findmarkersParam](#).

Value

New object.

See Also

[typemarkersParam](#)

Examples

```
exampleList <- getDeconvolutionExampleData()
singleCellExperimentExample <- randomSingleCellExperiment()
newParam <- findmarkersParam(singleCellExperiment=singleCellExperimentExample,
cellTypeVariable="celltype", markersPerType=5)
markers <- typemarkers(newParam)
```

```
getDeconvolutionExampleData
      getDeconvolutionExampleData
```

Description

Make example data for deconvolution.

Usage

```
getDeconvolutionExampleData(  
  cellScaleFactors = c(1, 10),  
  numberBulkSamples = 2,  
  numberMarkers = 10,  
  numberTypes = 2  
)
```

Arguments

`cellScaleFactors` Vector of cell scale factors
`numberBulkSamples` Number of bulk samples.
`numberMarkers` Number of cell type markers.
`numberTypes` Number of cell types.

Value

Example data as list.

Examples

```
exampleData <- getDeconvolutionExampleData()
```

`getDeconvolutionExampleDataBisque`
getDeconvolutionExampleDataBisque

Description

Get example data for Bisque algorithm.

Usage

```
getDeconvolutionExampleDataBisque(  
  numberBulkSamples = 100,  
  numberMarkers = 1000,  
  numberCells = 1000,  
  numberTypes = 2  
)
```

Arguments

`numberBulkSamples` Number of bulk samples.
`numberMarkers` Number of cell type markers.
`numberCells` Number of cells.
`numberTypes` Number of cell types.

Value

Example data as list.

Examples

```
exampleData <- getDeconvolutionExampleDataBisque()
```

```
getDeconvolutionExampleDataSCDC  
  getDeconvolutionExampleDataSCDC
```

Description

Get example data for SCDC

Usage

```
getDeconvolutionExampleDataSCDC()
```

Value

Example data as list.

Examples

```
exampleData <- getDeconvolutionExampleDataSCDC()
```

```
get_celltypes_from_sce  
  get_celltypes_from_sce
```

Description

Extract cell type values from SingleCellExperiment.

Usage

```
get_celltypes_from_sce(singleCellExperiment, cellTypeVariable = "celltype")
```

Arguments

`singleCellExperiment`
A SingleCellExperiment object.

`cellTypeVariable`
Variable containing cell type labels (e.g. "type1", "type2", etc.).

Value

List of cell type variable metadata and values.

Examples

```
exampleList <- getDeconvolutionExampleData()
```

| | |
|-------------------|--------------------------|
| get_csf_reference | <i>get_csf_reference</i> |
|-------------------|--------------------------|

Description

Retrieves the cell scale factors (csf) reference from the cellScaleFactors package.

Usage

```
get_csf_reference(userCellTypesVector = NULL, preferOrthogonal = TRUE)
```

Arguments

userCellTypesVector

Vector of user-specified cell types.

preferOrthogonal

Whether to prefer expression-orthogonal values (if TRUE, removes expression-based values, but only if alternative value types are available).

Details

Returns a table of cell scale factors from various data sources. The cell scale factors reference table has the following columns:

1. cell_type : Label of the cell type for the scale factor (e.g. neuron, T cell, etc.)
2. tissue : Label of the tissue of origin (e.g. brain, blood, etc.)
3. scale.factor.value : Point scale factor value prior to additional normalization
4. scale.factor.type : Label for scale factor type (e.g. cell or nuclear area, etc.)
5. scale.factor.data.source : Label for scale factor source (e.g. osmFISH, housekeeping gene expression, etc.)
6. citation.s : Citation(s) of source studies from which original measures or measure summaries were made.

Further details about the reference table can be found in the cellScaleFactors package.

Value

Table of type "data.frame" or "tibble".

Examples

```
example.data <- getDeconvolutionExampleData()
```

```
get_eset_from_matrix  get_eset_from_matrix
```

Description

Makes an ExpressionSet from a matrix.

Usage

```
get_eset_from_matrix(inputMatrix, batchVariable = "SampleName")
```

Arguments

`inputMatrix` User-specified expression matrix.
`batchVariable` Name of the batch variable.

Value

ExpressionSet.

Examples

```
exampleList <- getDeconvolutionExampleData()
```

```
independentbulkParam  Make a new independentbulkParam object
```

Description

Function to make a new object of class [independentbulkParam](#)

Usage

```
independentbulkParam(  
  bulkExpression = NULL,  
  bulkExpressionIndependent = NULL,  
  referenceExpression = NULL,  
  cellScaleFactors = NULL,  
  returnInfo = FALSE  
)
```

Arguments

| | |
|--|--|
| <code>bulkExpression</code> | Bulk mixed signals matrix of samples, which can be matched to single-cell samples. |
| <code>bulkExpressionIndependent</code> | Bulk mixed signals matrix of independent samples, which should not overlap samples in <code>y</code> . |
| <code>referenceExpression</code> | Signature matrix of cell type-specific signals. If not provided, can be computed from a provided <code>ExpressionSet</code> containing single-cell data. |
| <code>cellScaleFactors</code> | Cell size scale factor transformations of length equal to the <code>K</code> cell types to deconvolve. |
| <code>returnInfo</code> | Whether to return metadata and original method outputs with predicted proportions. |

Value

New object.

Examples

```
new("independentbulkParam")
```

`independentbulkParam-class`

independentbulkParam-class

Description

Class and methods for managing methods requiring independent bulk samples.

Arguments

| | |
|--|--|
| <code>bulkExpressionIndependent</code> | Bulk mixed signals matrix of independent samples, which should not overlap samples in <code>y</code> . |
|--|--|

Details

The main purpose of this class is to compare bulk sample data between the passed objects `y` and `yi`. Since we assume `yi` contains the independent bulk samples, it should not have overlapping sample IDs (`colnames`), and it should have overlapping marker IDs (`rownames`) compared to the reference bulk samples `y`.

Value

New object.

See Also

[deconParam](#), [referencebasedParam](#)

Examples

```
new("independentbulkParam")
```

lute

lute framework

Description

Obtain cell type markers and proportion predictions from various algorithms. Allows flexible data types and standard application of cell size scale factors.

Usage

```
lute(
  singleCellExperiment = NULL,
  referenceExpression = NULL,
  bulkExpression = NULL,
  bulkSummarizedExperiment = NULL,
  cellScaleFactors = NULL,
  returnInfo = FALSE,
  markersPerType = 20,
  assayName = "counts",
  cellTypeVariable = "celltype",
  typemarkerAlgorithm = "findmarkers",
  deconvolutionAlgorithm = "nnls",
  verbose = TRUE
)
```

Arguments

singleCellExperiment Object of type `SingleCellExperiment`. Optional (see argument `z`).

referenceExpression Signature matrix of cell type-specific signals. Optional (see argument `singleCellExperiment`).

bulkExpression Bulk mixed signals matrix of samples, which can be matched to single-cell samples. Optional (see argument `y.se`).

bulkSummarizedExperiment `SummarizedExperiment` or similar data type containing the bulk signals matrix in its assays (e.g. accessible with `assays(y.se)[[assayName]]` using the provided `assayName` argument). Optional (see argument `y`).

cellScaleFactors Cell size factor transformations of length equal to the `K` cell types to deconvolve. Optional, if not provided, uses equal weights for types.

| | |
|------------------------|---|
| returnInfo | Whether to return metadata and original method outputs with predicted proportions. |
| markersPerType | Number of top markers to get per cell type. |
| assayName | Name of expression matrix in singleCellExperiment, and optionally y.se, assays. Optional (e.g. "counts"; see arguments singleCellExperiment, y.se). |
| cellTypeVariable | Name of cell type variable in singleCellExperiment coldata. |
| typemarkerAlgorithm | Which type-specific marker selection algorithm to use. If NULL, skips type marker analyses. |
| deconvolutionAlgorithm | Where deconvolution algorithm to use. If NULL, skips deconvolution. |
| verbose | Whether to show verbose status messages. |

Details

Main function to use the lute deconvolution framework. Manages data conversions and mappings to deconvolution experiment steps, including setup, gene marker identification, and main deconvolution runs.

Support is provided for [SummarizedExperiment](#)-type or matrix-type inputs for the Z signature matrix (see referenceExpression argument) and Y bulk signals matrix (see bulkExpression arguments). Note, both Z and Y need to be provided or derivable in order to run deconvolution.

Value

A list containing results returned from type marker selection and deconvolution runs, with additional information returned if returnInfo == TRUE.

Examples

```
# get example bulk data
bulkExpression <- getDeconvolutionExampleData()$reference

# get example singleCellExperiment
singleCellExperiment <- randomSingleCellExperiment()[seq(10),]

# get framework results
experiment.results <- lute(
  singleCellExperiment=singleCellExperiment,
  bulkExpression=bulkExpression, typemarkerAlgorithm=NULL
)
```

`luteSupportedDeconvolutionAlgorithms`

luteSupportedDeconvolutionAlgorithms

Description

View details about supported deconvolution algorithms.

Usage

```
luteSupportedDeconvolutionAlgorithms()
```

Value

Table of supported deconvolution algorithms.

Examples

```
luteSupportedDeconvolutionAlgorithms()
```

```
new_workflow_table     new_workflow_table
```

Description

Makes a new experiment table for r-nf_deconvolution runs.

Usage

```
new_workflow_table(  
  singleCellExperimentNames = NULL,  
  dataDirectory = "data",  
  trueProportionsFilenameStem = "true_proportions_",  
  cellTypeVariable = "celltype",  
  tableDirectory = ".",  
  tableFileName = "workflow_table.csv",  
  save = TRUE,  
  overwrite = TRUE,  
  verbose = FALSE  
)
```

Arguments

```
singleCellExperimentNames     Names of SingleCellExperiment files to load.  
dataDirectory     Directory containing datasets to load.  
trueProportionsFilenameStem     File name stem of true proportions values.  
cellTypeVariable     Name of variable containing cell type labels.  
tableDirectory     Directory to write table.  
tableFileName     The file name of the new table to write.  
save     Whether to save the new table.  
overwrite     Whether to overwrite old table files.  
verbose     Whether to show verbose messages (T/F).
```

Details

Makes and returns/saves a r-nf_deconvolution experiment table. Checks for existence of provided files.

Value

New r-nf_deconvolution compatible table of experiment/run metadata.

Examples

```
new_workflow_table(save=FALSE)
```

| | |
|----------|--|
| nlsParam | <i>Make new object of class nlsParam</i> |
|----------|--|

Description

Main constructor for class [nlsParam](#).

Usage

```
nlsParam(
  bulkExpression,
  referenceExpression,
  cellScaleFactors,
  returnInfo = FALSE
)
```

Arguments

bulkExpression Bulk mixed signals matrix of samples, which can be matched to single-cell samples.

referenceExpression Signature matrix of cell type-specific signals. If not provided, can be computed from a provided [ExpressionSet](#) containing single-cell data.

cellScaleFactors Cell size factor transformations of length equal to the K cell types to deconvolve.

returnInfo Whether to return metadata and original method outputs with predicted proportions.

Details

Main parameter class for mapping inputs to the non-negative least squares (NNLS) deconvolution algorithm, implemented as `nls:::nls()`.

Value

Object of class [nlsParam](#)

See Also

[referencebasedParam](#), [deconvolutionParam](#)

Examples

```
exampleList <- getDeconvolutionExampleData()
param <- nlsParam(cellScaleFactors=exampleList[["cellScaleFactors"]],
  bulkExpression=exampleList[["bulkExpression"]],
  referenceExpression=exampleList[["referenceExpression"]])

## return only predicted proportions
deconvolution(param)

# return full results
param@returnInfo <- TRUE
names(deconvolution(param))
```

nlsParam-class

nlsParam-class

Description

Uses `nls::nls()`.

Details

Main constructor for class [nlsParam](#).

Value

New object.

See Also

[deconParam](#)

Examples

```
exampleList <- getDeconvolutionExampleData()
param <- nlsParam(cellScaleFactors=exampleList[["cellScaleFactors"]],
  bulkExpression=exampleList[["bulkExpression"]],
  referenceExpression=exampleList[["referenceExpression"]])

## return only predicted proportions
deconvolution(param)

# return full results
param@returnInfo <- TRUE
names(deconvolution(param))
```

```
parseDeconvolutionPredictionsResults
  parseDeconvolutionPredictionsResults
```

Description

Gets formatted predicted cell type proportions table from deconvolution results list.

Usage

```
parseDeconvolutionPredictionsResults(listPred, columnLabels, rowLabels)
```

Arguments

`listPred` List of cell type proportions predictions.
`columnLabels` Vector of cell type labels (e.g. "type1", "type2", etc.).
`rowLabels` Vector of sample id labels (e.g. "sample1", "sample2", etc.).

Value

Example data as list.

Examples

```
exampleData <- getDeconvolutionExampleData()
```

```
proportionsVectorsList
  proportionsVectorsList
```

Description

Get complementary proportions for k types. The first type `k1` is the vector of proportions for the first type. The remaining types up to `totalCellTypesK` are based on the reverse of `k1`. Types `k > 1` are assumed to have equal proportions complementary to `k1`.

Usage

```
proportionsVectorsList(totalCellTypesK = 2, firstCellTypeProportions = NULL)
```

Arguments

`totalCellTypesK` Total number of cell types to simulate.
`firstCellTypeProportions` Vector of first cell type proportions. If NULL, uses `seq(1e-3, 1-1e-3, 1e-3)`.

Details

For $k1=c(0, 0.5, 1)$, $totalCellTypesK=2$ will generate an additional type with proportions $c(1, 0.5, 0)$.

For the same $k1$ above, $totalCellTypesK=3$, will generate 2 types with the same proportions as $c(0.5, 0.25, 0)$.

Value

`lpv`, a list of proportions vectors for simulation iterations.

Examples

```
proportionsVectorsList(firstCellTypeProportions=c(0, 0.5, 1))
```

```
randomMarkersVectorsList
      randomMarkersVectorsList
```

Description

Get randomized markers using Poisson distribution sampling. For a given K , we assume "positive" markers have higher values than for non- K types, and thus we sample from 2 different Poisson distributions defined by different λ values (e.g. arguments `lambdaMean`, `lambdaMeanNegative`). WE also use argument `markerIndexVector` to define total markers as `length(markerIndexVector)` and the marker balance as relative counts of each type index.

Usage

```
randomMarkersVectorsList(
  markerIndexVector,
  numberIterations = 1,
  lambdaMean = 25,
  lambdaMeanNegative = 2,
  method = "nbinom",
  gammaSize = 10,
  gammaSizeNegative = 10
)
```

Arguments

| | |
|--------------------------------|---|
| <code>markerIndexVector</code> | Vector of marker indices. Index values correspond to the k types, and each index position represents a marker (e.g. $c(1,2,2)$ means two markers for the second type, etc.). |
| <code>numberIterations</code> | Total simulation iterations. |
| <code>lambdaMean</code> | Value of λ (Poisson dist. mean) for "positive" marker status (e.g. mean of dist. for k when marker is positive for k , negative for not- k). This is passed to the argument <code>mu</code> when method is "nbinom". |

| | |
|--------------------|--|
| lambdaMeanNegative | Value of lambda (Poisson dist. mean) for "negative" marker status (e.g. mean of dist. for k when marker is positive for not-k, negative for k). This is passed to the argument mu when method is "nbinom". |
| method | Type of randomization method to use. Accepts either "poisson" for poisson distribution (see '?rpois' for details), or "nbinom" for the negative binomial (a.k.a. gamm poisson) distribution (see '?nbinom' for details). |
| gammaSize | The gamma distribution magnitude for "positive" markers. This is applied when the "nbinom" method is used. |
| gammaSizeNegative | The gamma distribution magnitude for "negative" markers. This is applied when the "nbinom" method is used. |

Details

For example, if gindex is c(1, 1, 2), we define 3 total markers, 2 positive markers for type 1 (negative for type 2) and a single positive marker for type 2 (negative for type 1).

Value

Listed lgv object containing the randomized marker values across types.

Examples

```
randomMarkersVectorsList(markerIndexVector=c(rep(1, 10), rep(2, 5)))
```

```
randomSingleCellExperiment
      randomSingleCellExperiment
```

Description

Make a random object of type SingleCellExperiment. Uses the negative binomial distribution to randomly generate gene expression data for simulated cells.

Usage

```
randomSingleCellExperiment(
  numberGenes = 20,
  numberCells = 12,
  numberTypes = 2,
  fractionTypes = NULL,
  dispersion = NULL,
  expressionMean = 10,
  naInclude = FALSE,
  naFraction = 0.2,
  zeroInclude = FALSE,
  zeroFraction = 0.2,
  verbose = FALSE,
  seedNumber = 0
)
```

Arguments

| | |
|----------------|---|
| numberGenes | Number of genes to randomize. |
| numberCells | Number of cells to randomize. |
| numberTypes | Number of cell types to annotate. |
| fractionTypes | Vector of fractions by type. |
| dispersion | Dispersion of gene expression. If NULL, uses the mean from expressionMean |
| expressionMean | Poisson dist mean for random expression data. |
| naInclude | Whether to include random NA values. |
| naFraction | Fraction of NA values to include. |
| zeroInclude | Whether to include random zero-count values. |
| zeroFraction | Fraction of zero-count values to include. |
| verbose | Whether to show verbose status messages. |
| seedNumber | Seed value for randomization of expression data. |

Value

New randomized SingleCellExperiment object.

Examples

```
singleCellExperiment <- randomSingleCellExperiment()
```

referencebasedParam *Make new object of class referencebasedParam*

Description

Main constructor for class [referencebasedParam](#).

Usage

```
referencebasedParam(
  bulkExpression,
  referenceExpression,
  cellScaleFactors,
  returnInfo = FALSE
)
```

Arguments

| | |
|---------------------|---|
| bulkExpression | Bulk mixed signals matrix of samples, which can be matched to single-cell samples. |
| referenceExpression | Signature matrix of cell type-specific signals. If not provided, can be computed from a provided ExpressionSet containing single-cell data. |
| cellScaleFactors | Cell size factor transformations of length equal to the K cell types to deconvolve. |
| returnInfo | Whether to return metadata and original method outputs with predicted proportions. |

Details

Takes standard inputs for reference-based deconvolution algorithms.

Value

New object of class [referencebasedParam](#).

New object.

Examples

```
exampleList <- getDeconvolutionExampleData()
referencebasedParam(
  bulkExpression=exampleList$bulkExpression,
  referenceExpression=exampleList$referenceExpression,
  cellScaleFactors=exampleList$cellScaleFactors
)
```

referencebasedParam-class

referencebasedParam-class

Description

Class and methods for managing reference-based deconvolution methods.

Details

This is a parent class to manage reference-based deconvolution algorithms.

Child/sub-classes of this are distinguished by their use of either an explicit or implied z signature matrix (i.e. $Z[G,K]$ of dimensions G markers by K cell types). These also have an implied cell size term for biases from systematic cell size differences. If no cell size transformation is intended, this is the equivalent of passing equal size scales, (e.g. a K-length vector of equal values). See ‘vignette(package="lute")’ for details about experiment terms.

Value

New object.

Examples

```
exampleList <- getDeconvolutionExampleData()
referencebasedParam(
  bulkExpression=exampleList$bulkExpression,
  referenceExpression=exampleList$referenceExpression,
  cellScaleFactors=exampleList$cellScaleFactors)
```

```
referenceFromSingleCellExperiment
      referenceFromSingleCellExperiment
```

Description

Makes the Z cell atlas reference from a SingleCellExperiment.

Usage

```
referenceFromSingleCellExperiment(
  singleCellExperiment,
  assayName = "counts",
  cellTypeVariable = "celltype"
)
```

Arguments

`singleCellExperiment`
A SingleCellExperiment object.

`assayName` Name of expression assay type (e.g. "counts").

`cellTypeVariable`
Name of variable containing cell type labels (e.g. "type1", "type2", etc.).

Value

Matrix of cell summary values (Z reference atlas).

Examples

```
exampleList <- getDeconvolutionExampleData()
```

```
rmse          rmse
```

Description

Takes 2 vectors of numerics

Usage

```
rmse(proportionsTrue, proportionsPred, summaryType = "mean")
```

Arguments

`proportionsTrue`
cell type proportions taken as true

`proportionsPred`
cell type proportions taken as false

`summaryType` Toggle summary type (either "mean" or "median")

Details

Calculates the root mean squared error (RMSE) for specified true and predicted cell type proportions.

Function does not distinguish between true and predicted status, variable labels provided for convenience.

Value

single numeric

Examples

```
proportionsVectorPred <- seq(1e-10,2e-10,1e-11)
proportionsVectorTrue <- rev(proportionsVectorPred)
rmse(proportionsVectorTrue, proportionsVectorPred)
```

rmseTest

rmseTest

Description

Takes 2 vectors of numerics

Usage

```
rmseTest(firstVector, secondVector)
```

Arguments

firstVector First numeric vector.
secondVector Second numeric vector.

Details

Tests the rmse function for rounding imprecision.

Function to test RMSE values ('./unitTests/test_rmse.R').

Value

Single numeric value

Examples

```
proportionsVectorPred <- seq(1e-10,2e-10,1e-11)
proportionsVectorTrue <- rev(proportionsVectorPred)
rmseTest(proportionsVectorTrue, proportionsVectorPred)
```

| | |
|-------------|---|
| sce_to_eset | <i>sce_to_eset Convert SingleCellExperiment to ExpressionSet.</i> |
|-------------|---|

Description

sce_to_eset Convert SingleCellExperiment to ExpressionSet.

Usage

```
sce_to_eset(singleCellExperiment, assayName = "counts")
```

Arguments

singleCellExperiment
Object of type SingleCellExperiment (see ?SingleCellExperiment).

assayName
Name of assay to store in new eset.

Value

ExpressionSet.

Examples

```
sce <- randomSingleCellExperiment()  
sce_to_eset(sce, "counts")
```

| | |
|-----------|--|
| sce_to_se | <i>sce_to_se Convert SingleCellExperiment to SummarizedExperiment.</i> |
|-----------|--|

Description

sce_to_se Convert SingleCellExperiment to SummarizedExperiment.

Usage

```
sce_to_se(singleCellExperiment)
```

Arguments

singleCellExperiment
Object of type SingleCellExperiment (see ?SingleCellExperiment).

Value

SummarizedExperiment.

Examples

```
sce <- randomSingleCellExperiment()  
sce_to_se(sce)
```

| | |
|------------|-------------------|
| se_to_eset | <i>se_to_eset</i> |
|------------|-------------------|

Description

Convert SummarizedExperiment to ExpressionSet.

Usage

```
se_to_eset(summarizedExperiment, assayName = "counts")
```

Arguments

summarizedExperiment
Object of type SummarizedExperiment (see ?SummarizedExperiment).

assayName
Name of assay to store in new ExpressionSet object.

Value

New object of type ExpressionSet.

Examples

```
summarizedExperiment <- sce_to_se(randomSingleCellExperiment())
se_to_eset(summarizedExperiment)
```

| | |
|-----------|------------------|
| se_to_sce | <i>se_to_sce</i> |
|-----------|------------------|

Description

Convert SummarizedExperiment to SingleCellExperiment.

Usage

```
se_to_sce(summarizedExperiment)
```

Arguments

summarizedExperiment
Object of type SummarizedExperiment (see ?SummarizedExperiment).

Value

New SingleCellExperiment object.

Examples

```
se_to_sce(SummarizedExperiment())
```

 show,bisqueParam-method

Show generic behavior for object of class bisqueParam

Description

Show generic behavior for object of class bisqueParam

Usage

```
## S4 method for signature 'bisqueParam'
show(object)
```

Arguments

object Object of class [bisqueParam](#) (see ?bisqueParam).

Value

Prints data summary messages to console.

Examples

```
## get data
exampleList <- getDeconvolutionExampleDataBisque()
bulkExpressionSet <- exampleList[["bulkExpressionSet"]][,seq(10)]
bulkExpression <- exprs(exampleList[["bulkExpressionSet"]])
bulkExpression <- bulkExpression[,c(11:ncol(bulkExpression))]

## get param object
newBisqueParameter <- bisqueParam(bulkExpressionSet=bulkExpressionSet,
                                   bulkExpressionIndependent=bulkExpression,
                                   scData=exampleList[["singleCellExpressionSet"]],
                                   batchVariable="SubjectName",
                                   cellTypeVariable="cellType",
                                   useOverlap=FALSE)

## show
newBisqueParameter
```

 show,cellProportionsPredictions-method

Inspect cellProportionsPredictions object.

Description

Inspect cellProportionsPredictions object.

Usage

```
## S4 method for signature 'cellProportionsPredictions'
show(object)
```

Arguments

object Object of type cellProportionsPredictions (see ?cellProportionsPredictions).

Details

Method behavior for show.

Value

Shows object summaries.

Examples

```
exampleData <- getDeconvolutionExampleData()
```

show,deconvolutionParam-method

Show generic behavior for object of class [deconvolutionParam](#)

Description

Show generic behavior for object of class [deconvolutionParam](#)

Usage

```
## S4 method for signature 'deconvolutionParam'  
show(object)
```

Arguments

object An object of class [deconvolutionParam](#) (see ?deconvolutionParam).

Details

Method for behavior of show generic when called for object of class [deconvolutionParam](#)

Value

Shows object summaries.

Examples

```
param <- new("deconvolutionParam")  
deconvolution(param)
```

show,findmarkersParam-method

Show generic behavior for object of class [findmarkersParam](#)

Description

Show generic behavior for object of class [findmarkersParam](#)

Usage

```
## S4 method for signature 'findmarkersParam'  
show(object)
```

Arguments

object An object of class [findmarkersParam](#) (see `?findmarkersParam`).

Details

Method for behavior of show generic when called for object of class [findmarkersParam](#)

Value

Shows object summaries.

Examples

```
exampleList <- getDeconvolutionExampleData()  
singleCellExperimentExample <- randomSingleCellExperiment()  
newParam <- findmarkersParam(singleCellExperiment=singleCellExperimentExample,  
cellTypeVariable="celltype", markersPerType=5)  
markers <- typemarkers(newParam)
```

show,independentbulkParam-method

Method for [independentbulkParam](#)

Description

Method for [independentbulkParam](#)

Usage

```
## S4 method for signature 'independentbulkParam'  
show(object)
```

Arguments

object An object of class [independentbulkParam](#) (see `?independentbulkParam`).

Details

Display data summaries for an object of class [independentbulkParam](#).

Value

Shows object summaries.

Examples

```
new("independentbulkParam")
```

show,nlsParam-method *Show generic behavior for object of class nlsParam*

Description

Show generic behavior for object of class nlsParam

Usage

```
## S4 method for signature 'nlsParam'  
show(object)
```

Arguments

object Object of class [nlsParam](#) (see ?nlsParam).

Value

Prints data summary messages to console.

Examples

```
exampleList <- getDeconvolutionExampleData()  
referencebasedParam(  
  bulkExpression=exampleList$bulkExpression,  
  referenceExpression=exampleList$referenceExpression,  
  cellScaleFactors=exampleList$cellScaleFactors)
```

show,referencebasedParam-method

Show generic behavior for object of class [referencebasedParam](#)

Description

Show generic behavior for object of class [referencebasedParam](#)

Usage

```
## S4 method for signature 'referencebasedParam'  
show(object)
```

Arguments

object Object of class [referencebasedParam](#) (see `?referencebasedParam`).

Value

Prints data summary messages to console.

Examples

```
exampleList <- getDeconvolutionExampleData()  
referencebasedParam(  
  bulkExpression=exampleList$bulkExpression,  
  referenceExpression=exampleList$referenceExpression,  
  cellScaleFactors=exampleList$cellScaleFactors)
```

show,typemarkersParam-method

Show generic behavior for object of class [typemarkersParam](#)

Description

Show generic behavior for object of class [typemarkersParam](#)

Usage

```
## S4 method for signature 'typemarkersParam'  
show(object)
```

Arguments

object An object of class [typemarkersParam](#) (see `?typemarkersParam`).

Details

Method for behavior of show generic when called for object of class [typemarkersParam](#)

Value

Shows object summaries.

Examples

```
exampleList <- getDeconvolutionExampleData()
```

typemarkers

typemarkers

Description

Get cell type gene markers using standard accessors to supported functions.

Usage

```
typemarkers(object)
```

Arguments

object A [typemarkersParam](#)-type object (see `?typemarkersParam`).

Details

This generic manages tasks for marker gene identification. In particular, it takes a specified amount of marker genes to return per type.

Value

By default, return a vector of marker genes.

If `returnInfo == TRUE`, provides detailed results, including original outputs.

Author(s)

Sean Maden

See Also

[typemarkersParam](#)

Examples

```
exampleList <- getDeconvolutionExampleData()
```

typemarkers, findmarkersParam-method

Cell type markers method for findmarkersParam

Description

Defines the typemarkers method for [findmarkersParam](#).

Usage

```
## S4 method for signature 'findmarkersParam'  
typemarkers(object)
```

Arguments

object An object of class [findmarkersParam](#) (see ?findmarkersParam).

Details

Takes an object of class [findmarkersParam](#) as input, returning either a vector of cell type gene markers, or (if returnInfo == TRUE) a list containing such a vector along with original function outputs.

Value

Returns the top available markers, with type-specific marker filters, as either a vector of marker IDs or a results list.

Examples

```
exampleList <- getDeconvolutionExampleData()  
singleCellExperimentExample <- randomSingleCellExperiment()  
newParam <- findmarkersParam(singleCellExperiment=singleCellExperimentExample,  
cellTypeVariable="celltype", markersPerType=5)  
markers <- typemarkers(newParam)
```

typemarkers, typemarkersParam-method

Method for class typemarkersParam

Description

Method for class [typemarkersParam](#)

Usage

```
## S4 method for signature 'typemarkersParam'  
typemarkers(object)
```

Arguments

object An object of class [typemarkersParam](#).

Value

Info related to gene markers for cell types.

Examples

```
example.data <- getDeconvolutionExampleData()
```

| | |
|------------------|--|
| typemarkersParam | <i>Make new object of class typemarkersParam</i> |
|------------------|--|

Description

Main constructor for class [typemarkersParam](#).

Usage

```
typemarkersParam(markersPerType = 20, returnInfo = FALSE)
```

Arguments

markersPerType Bulk mixed signals matrix of samples, which can be matched to single-cell samples.

returnInfo Whether to return metadata and original marker selection method outputs with predicted proportions.

Details

This is the main parent class for cell type gene marker identification methods. Currently supported methods and their child classes include:

1. Mean Ratios: The method `DeconvoBuddies::get_mean_ratios2()`, supported by the class `mean-ratiosParam`.

Value

New object of class [typemarkersParam](#).

Examples

```
example.data <- getDeconvolutionExampleData()
```

 typemarkersParam-class

typemarkersParam-class

Description

Main constructor for class to manage mappings to the `typemarkers()` generic.

Arguments

`markersPerType` Number of top markers to get per cell type.
`returnInfo` Whether to return metadata and original method outputs with predicted proportions.

Details

Main constructor for class [typemarkersParam](#).

Value

New object.

See Also

`meanratiosParam`

Examples

```
exampleList <- getDeconvolutionExampleData()
```

 ypb_from_sce

ypb_from_sce

Description

Get pseudobulk from a `SingleCellExperiment` object.

Usage

```
ypb_from_sce(
  singleCellExperiment,
  assayName = "counts",
  cellTypeVariable = "celltype",
  sampleIdVariable = NULL,
  cellScaleFactors = NULL
)
```

Arguments

singleCellExperiment
 An object of type [SingleCellExperiment](#).

assayName
 Name of expression matrix in singleCellExperiment assays.

cellTypeVariable
 Variable name for cell type labels in singleCellExperiment coldata.

sampleIdVariable
 Variable name for sample/group ID labels in singleCellExperiment coldata.

cellScaleFactors
 Vector of cell type size scale factors. Optional.

Value

Matrix of simulated bulk convoluted signals.

Examples

```
singleCellExperimentExample <- randomSingleCellExperiment()
ypb_from_sce(singleCellExperimentExample)
```

z_matrix_from_sce *z_matrix_from_sce*

Description

Calculate a Z signature matrix (referenceExpression) from object of type [SingleCellExperiment](#).

Usage

```
z_matrix_from_sce(
  singleCellExperiment,
  cellTypeVariable = "celltype",
  summaryMethod = "mean",
  assayName = "counts"
)
```

Arguments

singleCellExperiment
 An object of type [SingleCellExperiment](#).

cellTypeVariable
 Variable name for cell type labels in singleCellExperiment coldata (e.g. "type1", "type2", etc.).

summaryMethod
 Summary statistic function to use.

assayName
 Name of expression matrix in singleCellExperiment assays (e.g. "counts").

Details

Calculate a Z signature matrix from object of type [SingleCellExperiment](#).

Value

New Z signature matrix.

Examples

```
singleCellExperiment.example <- randomSingleCellExperiment()
z_matrix_from_sce(singleCellExperiment.example)
```

[[,deconvolutionParam,ANY,ANY-method
Inspect slot in [deconvolutionParam](#) object

Description

Inspect slot in [deconvolutionParam](#) object

Usage

```
## S4 method for signature 'deconvolutionParam,ANY,ANY'  
x[[i]]
```

Arguments

| | |
|---|-------------------|
| x | Object to access. |
| i | Slot to access. |

Details

Inspect slot in [deconvolutionParam](#) object

Value

Contents of specified slot.
Object slot contents.

Examples

```
param <- new("deconvolutionParam")  
deconvolution(param)
```

[[, typemarkersParam, ANY, ANY-method

Inspect slot in [typemarkersParam](#) object

Description

Inspect slot in [typemarkersParam](#) object

Usage

```
## S4 method for signature 'typemarkersParam,ANY,ANY'  
x[[i]]
```

Arguments

| | |
|---|-------------------|
| x | Object to access. |
| i | Slot to access. |

Details

Inspect slot in [typemarkersParam](#) object

Value

Contents of specified slot.

Examples

```
example.data <- getDeconvolutionExampleData()
```

Index

[[, deconvolutionParam, ANY, ANY-method, 45
[[, typemarkersParam, ANY, ANY-method, 46
bisqueParam, 3, 3, 4, 5, 8, 35
BisqueParam-class (bisqueParam-class), 5
bisqueParam-class, 5
cellProportionsPredictions, 6, 7
cellProportionsPredictions-class, 6
deconParam, 21, 25
deconParam-class
 (deconvolutionParam-class), 12
DeconParam-class,
 (deconvolutionParam-class), 12
Deconvolute (deconvolution), 7
deconvolute, (deconvolution), 7
deconvolution, 7
Deconvolution, (deconvolution), 7
deconvolution, bisqueParam-method, 8
deconvolution, deconvolutionParam-method, 9
deconvolution, independentbulkParam-method, 10
deconvolution, nnlsParam-method, 10
deconvolution, referencebasedParam-method, 11
deconvolutionParam, 5, 7–9, 25, 36, 45
deconvolutionParam-class, 12
DeconvolutionParam-class,
 (deconvolutionParam-class), 12
eset_to_sce, 13
eset_to_se, 13
ExpressionSet, 24
findmarkersParam, 14, 14, 15, 37, 41
findMarkersParam-class
 (findmarkersParam-class), 15
findmarkersParam-class, 15
FindmarkersParam-class,
 (findmarkersParam-class), 15
get_celltypes_from_sce, 17
get_csf_reference, 18
get_eset_from_matrix, 19
getDeconvolutionExampleData, 15
getDeconvolutionExampleDataBisque, 16
getDeconvolutionExampleDataSCDC, 17
independentbulkParam, 5, 8, 10, 19, 19, 37, 38
independentbulkParam-class, 20
lute, 21
luteSupportedDeconvolutionAlgorithms, 22
new_workflow_table, 23
nnlsParam, 8, 10, 11, 24, 24, 25, 38
NNLSParam-class (nnlsParam-class), 25
nnlsParam-class, 25
parseDeconvolutionPredictionsResults, 26
proportionsVectorsList, 26
randomMarkersVectorsList, 27
randomSingleCellExperiment, 28
referencebasedParam, 5, 8, 11, 12, 21, 25, 29, 29, 30, 39
referencebasedParam-class, 30
referenceFromSingleCellExperiment, 31
rmse, 31
rmseTest, 32
sce_to_eset, 33
sce_to_se, 33
se_to_eset, 34
se_to_sce, 34
show, bisqueParam-method, 35
show, cellProportionsPredictions-method, 35
show, deconvolutionParam-method, 36
show, findmarkersParam-method, 37
show, independentbulkParam-method, 37
show, nnlsParam-method, 38
show, referencebasedParam-method, 39
show, typemarkersParam-method, 39

SingleCellExperiment, [44](#)
SummarizedExperiment, [22](#)

TypeMarkers (typemarkers), [40](#)
typemarkers, [40](#)
Typemarkers, (typemarkers), [40](#)
typemarkers, findmarkersParam-method,
[41](#)
typemarkers, typemarkersParam-method,
[41](#)
typemarkersParam, [14](#), [15](#), [39–42](#), [42](#), [43](#), [46](#)
TypeMarkersParam-class
(typemarkersParam-class), [43](#)
typemarkersParam-class, [43](#)
TypemarkersParam-class,
(typemarkersParam-class), [43](#)

y pb_from_sce, [43](#)

z_matrix_from_sce, [44](#)