Package 'smartid'

May 26, 2026

Title Scoring and Marker Selection Method Based on Modified TF-IDF

Version 1.9.0

Description This package enables automated selection of group specific signature, especially for rare population. The package is developed for generating specifc lists of signature genes based on Term Frequency-Inverse Document Frequency (TF-IDF) modified methods. It can also be used as a new gene-set scoring method or data transformation method. Multiple visualization functions are implemented in this package.

biocViews Software, GeneExpression, Transcriptomics

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Collate 'AllClasses.R' 'tf_idf_iae_wrappers.R' 'score.R' 'AllGenerics.R' 'gs_score-methods.R' 'plot.R' 'scale_mgm.R' 'smartid-package.R' 'score-methods.R' 'select_markers.R' 'top_markers.R' 'top_markers-methods.R'

Depends R (>= 4.4)

Imports dplyr, ggplot2, graphics, Matrix, mclust, methods, mixtools, sparseMatrixStats, stats, SummarizedExperiment, tidyr, utils

Suggests BiocStyle, dbscan, ggpubr, knitr, rmarkdown, scater, splatter, testthat (>= 3.0.0), tibble, tidytext, UpSetR

URL https://davislaboratory.github.io/smartid

BugReports https://github.com/DavisLaboratory/smartid/issues

VignetteBuilder knitr

Language en-US

Config/testthat/edition 3

LazyData false

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

git_branch devel

git_last_commit 931d9df

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-25

Author Jinjin Chen [aut, cre] (ORCID: <https://orcid.org/0000-0001-7923-5723>)

Maintainer Jinjin Chen <chen.j@wehi.edu.au>

Contents

`cal_score` *calculate combined score*

Description

compute TF (term/feature frequency), IDF (inverse document/cell frequency), IAE (inverse average expression of features) and combine the the final score

Usage

```
cal_score(
  data,
  tf = c("logtf", "tf"),
  idf = "prob",
  iae = "prob",
  slot = "counts",
  new.slot = "score",
  par.idf = NULL,
  par.iae = NULL,
  return.intermediate = FALSE
)

## S4 method for signature 'AnyMatrix'
cal_score(
  data,
  tf = c("logtf", "tf"),
  idf = "prob",
  iae = "prob",
  par.idf = NULL,
  par.iae = NULL,
  return.intermediate = FALSE
)

## S4 method for signature 'SummarizedExperiment'
cal_score(
  data,
  tf = c("logtf", "tf"),
  idf = "prob",
  iae = "prob",
  slot = "counts",
  new.slot = "score",
  par.idf = NULL,
  par.iae = NULL,
  return.intermediate = FALSE
)
```

Arguments

data	an expression object, can be matrix or SummarizedExperiment
tf	a character, specify the TF method to use, can be "tf" or "logtf"
idf	a character, specify the IDF method to use. Available methods can be accessed using idf_iae_methods()
iae	a character, specify the IAE method to use. Available methods can be accessed using idf_iae_methods()
slot	a character, specify which slot to use when data is se object, optional, default 'counts'
new.slot	a character, specify the name of slot to save score in se object, optional, default 'score'
par.idf	other parameters for specified IDF methods

par.iae	other parameters for specified IAE methods
return.intermediate	logical, if TRUE also return or store the intermediate tf, idf and iae matrices. Defaults to FALSE since these objects have the same dimension as the input expression matrix and can dominate memory usage on large datasets. Set to TRUE to restore the pre-1.7.3 behavior where intermediates were kept in metadata() of the SummarizedExperiment output.

Value

A list of matrices or se object containing combined score

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
cal_score(
  data,
  par.idf = list(label = sample(c("A", "B"), 10, replace = TRUE)),
  par.iae = list(label = sample(c("A", "B"), 10, replace = TRUE))
)
```

---

cal_score_init *Calculate score for each feature in each cell*

Description

Calculate score for each feature in each cell

Usage

```
cal_score_init(
  expr,
  tf = c("logtf", "tf"),
  idf = "prob",
  iae = "prob",
  par.idf = NULL,
  par.iae = NULL,
  return.intermediate = FALSE
)
```

Arguments

expr	a count matrix, features in row and cells in column
tf	a character, specify the TF method to use, can be "tf" or "logtf"
idf	a character, specify the IDF method to use. Available methods can be accessed using idf_iae_methods()
iae	a character, specify the IAE method to use. Available methods can be accessed using idf_iae_methods()
par.idf	other parameters for specified IDF methods
par.iae	other parameters for specified IAE methods

return.intermediate
: logical, if TRUE the returned list also contains the intermediate `tf`, `idf` and `iae` objects. Default `FALSE` keeps only the combined `score` to avoid the memory overhead of three extra feature-by-cell matrices on large inputs.

Value

a list always containing `score`; when `return.intermediate = TRUE` the list additionally contains `tf`, `idf` and `iae`.

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
label <- sample(c("A", "B"), 10, replace = TRUE)
smartid:::cal_score_init(data,
  par.idf = list(label = label),
  par.iae = list(label = label)
)
```

---

gs_score *compute overall score based on the given marker list*

Description

compute overall score based on the given marker list

Usage

```
gs_score(data, features = NULL, slot = "score", suffix = "score")

## S4 method for signature 'AnyMatrix,ANY'
gs_score(data, features = NULL)

## S4 method for signature 'AnyMatrix,list'
gs_score(data, features = NULL, suffix = "score")

## S4 method for signature 'SummarizedExperiment,ANY'
gs_score(data, features = NULL, slot = "score", suffix = "score")
```

Arguments

`data`	an expression object, can be matrix or SummarizedExperiment
`features`	vector or named list, feature names to compute score
`slot`	a character, specify which slot to use when data is se object, optional, default 'score'
`suffix`	a character, specify the name suffix to save score when features is a named list

Value

A vector of overall score for each sample

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(seq_len(10)))
gs_score(data, features = seq_len(3))
```

`gs_score_init` *Calculate scores of each cell on given features*

Description

Calculate scores of each cell on given features

Usage

```
gs_score_init(score, features = NULL)
```

Arguments

`score`	matrix, features in row and samples in column
`features`	vector, feature names to compute score

Value

a vector of score

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
gs_score_init(data, 1:5)
```

`iae` *standard inverse average expression*

Description

standard inverse average expression

Usage

```
iae(expr, features = NULL, thres = 0)
```

Arguments

`expr`	a matrix, features in row and cells in column
`features`	vector, feature names or indexes to compute
`thres`	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IAE_i} = log(1 + \frac{n}{\hat{N}_{i,j} + 1})$$

where n is the total number of cells, $N_{i,j}$ is the counts of feature i in cell j.

Value

a vector of inverse average expression score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae(data)
```

---

iae_hdb *inverse average expression using hdbscan cluster as label*

Description

inverse average expression using hdbscan cluster as label

Usage

```
iae_hdb(expr, features = NULL, multi = TRUE, thres = 0, minPts = 2, ...)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
multi	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
thres	numeric, cell only counts when expr > threshold, default 0
minPts	integer, minimum size of clusters, default 2. Details in dbscan::hdbscan().
...	parameters for dbscan::hdbscan()

Details

Details as iae_prob().

Value

a matrix of inverse average expression score

Examples

```
set.seed(123)
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_hdb(data)
```

iae_igm *labeled inverse average expression: IGM*

Description

labeled inverse average expression: IGM

Usage

```
iae_igm(expr, features = NULL, label, lambda = 7, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
label	vector, group label of each cell
lambda	numeric, hyperparameter for IGM
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IGM_i} = log(1 + \lambda \frac{max(mean(N_{i,j \in D})_k)}{\sum_k^K (mean(N_{i,j \in D})_k * r_k) + e^{-8}})$$

where λ is the hyper parameter, $N_{i,j \in D}$ is the counts of feature i in cell j within class D, and r_k is the rank of $mean(N_{i,j \in D})$.

Value

a vector of inverse gravity moment score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_igm(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

iae_m *inverse average expression: max*

Description

inverse average expression: max

Usage

```
iae_m(expr, features = NULL, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IAE_{i,j}} = log(1 + \frac{max_{\{i' \in j\}}(n_{i'})}{\sum_{j=1}^{n} max(0, N_{i,j} - threshold) + 1})$$

where i is the feature i and $i^{'}$ is the feature except i, $N_{i,j}$ is the counts of feature i in cell j, and $n_{i'}$ is $\sum_{j=1}^{n} sign(N_{i,j} > threshold)$.

Value

a matrix of inverse average expression score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_m(data)
```

---

iae_prob *labeled inverse average expression: probability based*

Description

labeled inverse average expression: probability based

Usage

```
iae_prob(expr, features = NULL, label, multi = TRUE, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
label	vector, group label of each cell
multi	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IAE_{i,j}} = log(1 + \frac{mean(N_{i,j \in D})}{max(mean(N_{i,j \in \hat{D}})) + e^{-8}} * mean(N_{i,j \in D}))$$

where $N_{i,j \in D}$ is the counts of feature i in cell j within class D, and $\hat{D}$ is the class except D.

Value

a matrix of inverse average expression score

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_prob(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

`iae_rf` *labeled inverse average expression: relative frequency*

Description

labeled inverse average expression: relative frequency

Usage

```
iae_rf(expr, features = NULL, label, multi = TRUE, thres = 0)
```

Arguments

`expr`	a matrix, features in row and cells in column
`features`	vector, feature names or indexes to compute
`label`	vector, group label of each cell
`multi`	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
`thres`	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IAE} = log(1 + \frac{mean(N_{i,j \in D})}{max(mean(N_{i,j \in \hat{D}})) + e^{-8}})$$

where $N_{i,j \in D}$ is the counts of feature i in cell j within class D, and $\hat{D}$ is the class except D.

Value

a matrix of inverse average expression score

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_rf(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

iae_sd *inverse average expression using standard deviation (SD)*

Description

inverse average expression using standard deviation (SD)

Usage

```
iae_sd(expr, features = NULL, log = FALSE, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
log	logical, if to do log-transformation
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IAE} = log(1 + sd(tf_i) * \frac{n}{\sum_{j=1}^{n} max(0, N_{i,j}) + 1})$$

where tf_i is the term frequency of feature i, see details in `tf()`, n is the total number of cells and $N_{i,j}$ is the counts of feature i in cell j.

Value

a vector of inverse average expression score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::iae_sd(data)
```

idf *standard inverse cell frequency*

Description

standard inverse cell frequency

Usage

```
idf(expr, features = NULL, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IDF_i} = log(1 + \frac{n}{n_i + 1})$$

where n is the total number of cells, n_i is the number of cells containing feature i.

Value

a vector of inverse cell frequency score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf(data)
```

---

idf_hdb *inverse document frequency using hdbscan cluster as label*

Description

inverse document frequency using hdbscan cluster as label

Usage

```
idf_hdb(expr, features = NULL, multi = TRUE, thres = 0, minPts = 2, ...)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
multi	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
thres	numeric, cell only counts when expr > threshold, default 0
minPts	integer, minimum size of clusters, default 2. Details in dbscan::hdbscan().
...	parameters for dbscan::hdbscan()

Details

Details as idf_prob().

Value

a matrix of inverse cell frequency score

Examples

```
set.seed(123)
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_hdb(data)
```

`idf_iae_methods` *Get names of available IDF and IAE methods*

Description

Returns a named vector of IDF/IAE methods

Usage

```
idf_iae_methods()
```

Value

names of methods implemented

Examples

```
idf_iae_methods()
```

`idf_igm` *labeled inverse cell frequency: IGM*

Description

labeled inverse cell frequency: IGM

Usage

```
idf_igm(expr, features = NULL, label, lambda = 7, thres = 0)
```

Arguments

`expr`	a matrix, features in row and cells in column
`features`	vector, feature names or indexes to compute
`label`	vector, group label of each cell
`lambda`	numeric, hyperparameter for IGM
`thres`	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IGM_i} = log(1 + \lambda \frac{max(n_{i,j \in D})_k}{\sum_k^K ((n_{i,j \in D})_k * r_k) + e^{-8}})$$

where λ is the hyper parameter, $n_{i,j \in D}$ is the number of cells containing feature i in class D, r_k is the rank of $n_{i,j \in D}$.

Value

a vector of inverse gravity moment score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_igm(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

---

idf_m *inverse cell frequency: max*

Description

inverse cell frequency: max

Usage

```
idf_m(expr, features = NULL, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IDF_{i,j}} = log(\frac{max_{\{i' \in j\}}(n_{i'})}{n_i + 1})$$

where i is the feature i and $i^{'}$ is the feature except i, n_i is the number of cells containing feature i, and $n_{i'}$ is the number of cells containing feature $i^{'}$.

Value

a matrix of inverse cell frequency score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_m(data)
```

idf_prob *labeled inverse cell frequency: probability based*

Description

labeled inverse cell frequency: probability based

Usage

```
idf_prob(expr, features = NULL, label, multi = TRUE, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
label	vector, group label of each cell
multi	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IDF_{i,j}} = log(1 + \frac{\frac{n_{i,j\in D}}{n_{j\in D}}}{max(\frac{n_{i,j\in \hat{D}}}{n_{j\in \hat{D}}}) + e^{-8}} \frac{n_{i,j\in D}}{n_{j\in D}})$$

where $n_{i,j\in D}$ is the number of cells containing feature i in class D, $n_{j\in D}$ is the total number of cells in class D, $\hat{D}$ is the class except D.

Value

a matrix of inverse cell frequency score

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_prob(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

idf_rf *labeled inverse cell frequency: relative frequency*

Description

labeled inverse cell frequency: relative frequency

Usage

```
idf_rf(expr, features = NULL, label, multi = TRUE, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
label	vector, group label of each cell
multi	logical, if to compute based on binary (FALSE) or multi-class (TRUE)
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IDF_{i,j}} = log(1 + \frac{\frac{n_{i,j \in D}}{n_{j \in D}}}{max(\frac{n_{i,j \in \hat{D}}}{n_{j \in \hat{D}}}) + e^{-8}})$$

where $n_{i,j \in D}$ is the number of cells containing feature i in class D, $n_{j \in D}$ is the total number of cells in class D, $\hat{D}$ is the class except D.

Value

a matrix of inverse cell frequency score

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_rf(data, label = sample(c("A", "B"), 10, replace = TRUE))
```

---

idf_sd *inverse cell frequency using standard deviation (SD)*

Description

inverse cell frequency using standard deviation (SD)

Usage

```
idf_sd(expr, features = NULL, log = FALSE, thres = 0)
```

Arguments

expr	a matrix, features in row and cells in column
features	vector, feature names or indexes to compute
log	logical, if to do log-transformation
thres	numeric, cell only counts when expr > threshold, default 0

Details

$$\mathbf{IDF_i} = log(1 + sd(tf_i) * \frac{n}{n_i + 1})$$

where tf_i is the term frequency of feature i, see details in `tf()`, n is the total number of cells and n_i is the number of cells containing feature i.

Value

a vector of inverse cell frequency score for each feature

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::idf_sd(data)
```

`markers_hdbscan` *select markers using HDBSCAN method*

Description

select markers using HDBSCAN method

Usage

```
markers_hdbscan(
  top_markers,
  column = ".dot",
  s_thres = NULL,
  method = c("max.one", "remove.min"),
  minPts = 5,
  plot = FALSE,
  ...
)
```

Arguments

`top_markers`	output of `top_markers()`
`column`	character, specify which column used as group label
`s_thres`	NULL or numeric, only features with score > threshold will be returned, if NULL will use 2 * average probability as threshold
`method`	can be "max.one" or "remove.min", if to only keep features in 1st component or return features not in the last component
`minPts`	integer, minimum size of clusters for `dbscan::hdbscan()`
`plot`	logical, if to plot mixture density and hist
`...`	other params for `dbscan::hdbscan()`

Value

a list of markers for each group

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
top_n <- top_markers(data, label = rep(c("A", "B"), 5))
markers_hdbscan(top_n, minPts = 2)
```

`markers_mclust` *select markers using mclust EM method*

Description

select markers using mclust EM method

Usage

```
markers_mclust(
  top_markers,
  column = ".dot",
  prob = 0.99,
  s_thres = NULL,
  method = c("max.one", "remove.min"),
  plot = FALSE,
  ...
)
```

Arguments

`top_markers`	output of `top_markers()`
`column`	character, specify which column used as group label
`prob`	numeric, probability cutoff for 1st component classification
`s_thres`	NULL or numeric, only features with score > threshold will be returned, if NULL will use 2 * average probability as threshold
`method`	can be "max.one" or "remove.min", if to only keep features in 1st component or return features not in the last component
`plot`	logical, if to plot mixture density and hist
`...`	other params for `mclust::densityMclust()`

Value

a list of markers for each group

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
top_n <- top_markers(data, label = rep(c("A", "B"), 5))
markers_mclust(top_n)
```

`markers_mixmdl` *select markers using mixtools EM method*

Description

select markers using mixtools EM method

Usage

```
markers_mixmdl(
  top_markers,
  column = ".dot",
  prob = 0.99,
  k = 3,
  ratio = 2,
  dist = c("norm", "gamma"),
  maxit = 1e+05,
  plot = FALSE,
  ...
)
```

Arguments

`top_markers`	output of `top_markers()`
`column`	character, specify which column used as group label
`prob`	numeric, probability cutoff for 1st component classification
`k`	integer, number of components of mixtures
`ratio`	numeric, ratio cutoff of 1st component mu to 2nd component mu, only when ratio > cutoff will return markers for the group
`dist`	can be one of "norm" and "gamma", specify if to use `mixtools::normalmixEM()` or `mixtools::gammamixEM()`
`maxit`	integer, maximum number of iterations for EM
`plot`	logical, if to plot mixture density and hist
`...`	other params for `mixtools::normalmixEM()` or `mixtools::gammamixEM()`

Value

a list of markers for each group

Examples

```
set.seed(1000)
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
top_n <- top_markers(data, label = rep(c("A", "B"), 5))
markers_mixmdl(top_n, k = 3)
```

ova_score_boxplot *boxplot of features overall score*

Description

boxplot of features overall score

Usage

```
ova_score_boxplot(data, features, ref.group, label, method = "t.test")
```

Arguments

data	matrix, features in row and samples in column
features	vector, feature names to plot
ref.group	character, reference group name
label	vector, group labels
method	character, statistical test to use, details in ggpubr::stat_compare_means()

Value

ggplot object

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
ova_score_boxplot(data, 1:5, ref.group = "A", label = rep(c("A", "B"), 5))
```

scale_mgm *scale by mean of group mean for imbalanced data*

Description

scale by mean of group mean for imbalanced data

Usage

```
scale_mgm(expr, label, pooled.sd = FALSE)
```

Arguments

expr	matrix
label	a vector of group label
pooled.sd	logical, if to use pooled SD for scaling

Details

$$z = \frac{x - \frac{\sum_k^{n_D}(\mu_k)}{n_D}}{s}$$

where μ_k is the mean of x in k^{th} class, and n_D is the number of classes, s is the standard deviation of x, when `pooled.sd` is set to be TRUE, s will be replaced with s_{pooled}, $s_{pooled} = \sqrt{\frac{\sum_k^{n_D}(n_k-1){s_k}^2}{\sum_k^{n_D} n_k - k}}$

Value

scaled matrix

Examples

```
scale_mgm(matrix(rnorm(100), 10), label = rep(letters[1:2], 5))
```

---

score_barplot *barplot of processed score*

Description

barplot of processed score

Usage

```
score_barplot(top_markers, column = ".dot", f_list, n = 30)
```

Arguments

`top_markers`	output of `top_markers()`
`column`	character, specify which column used as group label
`f_list`	a named list of markers
`n`	numeric, number of returned top genes for each group

Value

ggplot object

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
top_n <- top_markers(data, label = rep(c("A", "B"), 5))
score_barplot(top_n)
```

sim_sce_test *scRNA-seq test data of 4 groups simulated by* splatter.

Description

A SingleCellExperiment object containing 4 groups with each group up-regulated DEGs saved in metadata.

Usage

```
data(sim_sce_test)
```

Format

A SingleCellExperiment object of 100genes * 400 cells.

Value

SingleCellExperiment

Source

splatter::splatSimulate()

sin_score_boxplot *boxplot of split single feature score*

Description

boxplot of split single feature score

Usage

```
sin_score_boxplot(data, features = NULL, ref.group, label, method = "t.test")
```

Arguments

data	matrix, features in row and samples in column
features	vector, feature names to plot
ref.group	character, reference group name
label	vector, group labels
method	character, statistical test to use, details in ggpubr::stat_compare_means()

Value

faceted ggplot object

Examples

```
data <- matrix(rnorm(100), 10, dimnames = list(1:10))
sin_score_boxplot(data, 1:2, ref.group = "A", label = rep(c("A", "B"), 5))
```

smartid_Package *Scoring and Marker Selection method based on modified TF-IDF*

Description

smartid This package enables automated selection of group specific signature, especially for rare population. The package is developed for generating specific lists of signature genes based on TF-IDF modified methods. It can also be used as a new gene-set scoring method or data transformation method. Multiple visualization functions are implemented in this package.

Value

Marker list and scores

Author(s)

Jinjin Chen <chen.j@wehi.edu.au>

See Also

Useful links:

- https://davislaboratory.github.io/smartid
- Report bugs at https://github.com/DavisLaboratory/smartid/issues

tf *compute term/feature frequency within each cell*

Description

compute term/feature frequency within each cell

Usage

```
tf(expr, log = FALSE)
```

Arguments

expr	a count matrix, features in row and cells in column
log	logical, if to do log-transformation

Details

$$\mathbf{TF_{i,j}} = \frac{N_{i,j}}{\sum_j N_{i,j}}$$

where $N_{i,j}$ is the counts of feature i in cell j.

Value

a matrix of term/gene frequency. For a `dgCMatrix` input the returned object preserves sparsity (`log1p(0) == 0`); dense input returns a dense matrix.

Examples

```
data <- matrix(rpois(100, 2), 10, dimnames = list(1:10))
smartid:::tf(data)
```

`top_markers` *scale score and return top markers*

Description

scale and transform score and output top markers for groups

Usage

```
top_markers(
  data,
  label,
  n = 10,
  use.glm = TRUE,
  batch = NULL,
  scale = TRUE,
  use.mgm = TRUE,
  softmax = TRUE,
  slot = "score",
  ...
)

## S4 method for signature 'AnyMatrix'
top_markers(
  data,
  label,
  n = 10,
  use.glm = TRUE,
  batch = NULL,
  scale = TRUE,
  use.mgm = TRUE,
  softmax = TRUE,
  slot = "score",
  ...
)

## S4 method for signature 'SummarizedExperiment'
top_markers(
  data,
  label,
  n = 10,
  use.glm = TRUE,
```

```
  batch = NULL,
  scale = TRUE,
  use.mgm = TRUE,
  softmax = TRUE,
  slot = "score",
  ...
)
```

Arguments

data	an expression object, can be matrix or SummarizedExperiment
label	a vector of group label
n	integer, number of returned top genes for each group
use.glm	logical, if to use stats::glm() to compute group mean score, if TRUE, also compute mean score difference as output
batch	a vector of batch labels, default NULL
scale	logical, if to scale data by row
use.mgm	logical, if to scale data using scale_mgm()
softmax	logical, if to apply softmax transformation on output
slot	a character, specify which slot to use when data is se object, optional, default 'score'
...	params for top_markers_abs() or top_markers_glm()

Value

A tibble with top n feature names, group labels and ordered scores

Examples

```
data <- matrix(rgamma(100, 2), 10, dimnames = list(1:10))
top_markers(data, label = rep(c("A", "B"), 5))
```

---

top_markers_abs *calculate group median, MAD or mean score and order genes based on scores*

Description

calculate group median, MAD or mean score and order genes based on scores

Usage

```
top_markers_abs(
  data,
  label,
  n = 10,
  pooled.sd = FALSE,
  method = c("median", "mad", "mean"),
  scale = TRUE,
```

```
  use.mgm = TRUE,
  softmax = TRUE,
  tau = 1
)
```

Arguments

data	matrix, features in row and samples in column
label	a vector of group label
n	integer, number of returned top genes for each group
pooled.sd	logical, if to use pooled SD for scaling
method	character, specify metric to compute, can be one of "median", "mad", "mean"
scale	logical, if to scale data by row
use.mgm	logical, if to scale data using scale_mgm()
softmax	logical, if to apply softmax transformation on output
tau	numeric, hyper parameter for softmax

Value

a tibble with feature names, group labels and ordered processed scores

Examples

```
data <- matrix(rgamma(100, 2), 10, dimnames = list(1:10))
top_markers_abs(data, label = rep(c("A", "B"), 5))
```

---

top_markers_glm *calculate group mean score using glm and order genes based on scores difference*

Description

calculate group mean score using glm and order genes based on scores difference

Usage

```
top_markers_glm(
  data,
  label,
  n = 10,
  family = gaussian(),
  batch = NULL,
  scale = TRUE,
  use.mgm = TRUE,
  pooled.sd = FALSE,
  softmax = TRUE,
  tau = 1
)
```

Arguments

data	matrix, features in row and samples in column
label	a vector of group label
n	integer, number of returned top genes for each group
family	family for glm, details in stats::glm()
batch	a vector of batch labels, default NULL
scale	logical, if to scale data by row
use.mgm	logical, if to scale data using scale_mgm()
pooled.sd	logical, if to use pooled SD for scaling
softmax	logical, if to apply softmax transformation on output
tau	numeric, hyper parameter for softmax

Details

When family is gaussian() with the identity link (the default) and the design matrix is full-rank, top_markers_glm() computes all per- gene label coefficients in a single closed-form least-squares solve via Matrix::solve(crossprod(X), crossprod(X, t(data))), avoiding the per-gene glm() loop. For any other family, or a rank-deficient design, the function automatically falls back to the legacy apply(data, 1, glm(...)) path, so results are unchanged for users who pass e.g. family = Gamma() or family = poisson().

Value

a tibble with feature names, group labels and ordered processed scores

Examples

```
data <- matrix(rgamma(100, 2), 10, dimnames = list(1:10))
top_markers_glm(data, label = rep(c("A", "B"), 5))
```

---

top_markers_init *compute group summarized score and order genes based on processed scores*

Description

compute group summarized score and order genes based on processed scores

Usage

```
top_markers_init(
  data,
  label,
  n = 10,
  use.glm = TRUE,
  batch = NULL,
  scale = TRUE,
  use.mgm = TRUE,
  softmax = TRUE,
  ...
)
```

Arguments

data	matrix, features in row and samples in column
label	a vector of group label
n	integer, number of returned top genes for each group
use.glm	logical, if to use stats::glm() to compute group mean score, if TRUE, also compute mean score difference as output
batch	a vector of batch labels, default NULL
scale	logical, if to scale data by row
use.mgm	logical, if to scale data using scale_mgm()
softmax	logical, if to apply softmax transformation on output
...	params for top_markers_abs() or top_markers_glm()

Value

a tibble with feature names, group labels and ordered processed scores

Examples

```
data <- matrix(rgamma(100, 2), 10, dimnames = list(1:10))
top_markers_init(data, label = rep(c("A", "B"), 5))
```

Index