Package ‘ccfindR’

March 6, 2024

Version 1.22.0
Date 2019-10-1
Title Cancer Clone Finder
Depends R (>= 3.6.0)
Imports stats, S4Vectors, utils, methods, Matrix,
   SummarizedExperiment, SingleCellExperiment, Rtsne, graphics,
   grDevices, gtools, RColorBrewer, ape, Rmpi, irlba, Rcpp, Rdpack
   (> = 0.7)

Rd Macros Rdpack
biocViews Transcriptomics, SingleCell, ImmunoOncology, Bayesian,
   Clustering

Description A collection of tools for cancer genomic data clustering
   analyses, including those for single cell RNA-seq.
   Cell clustering and feature gene selection analysis employ Bayesian
   (and maximum likelihood) non-negative matrix factorization (NMF) algorithm.
   Input data set consists of RNA count matrix, gene, and cell bar code
   annotations. Analysis outputs are factor matrices for multiple ranks and
   marginal likelihood values for each rank. The package includes utilities for
   downstream analyses, including meta-gene identification, visualization, and
   construction of rank-based trees for clusters.

License GPL (> = 2)
NeedsCompilation yes

URL http://dx.doi.org/10.26508/lsa.201900443
RoxygenNote 6.1.1
Encoding UTF-8
Suggests BiocStyle, knitr, rmarkdown
VignetteBuilder knitr
LinkingTo Rcpp, RcppEigen

Author Jun Woo [aut, cre],
   Jinhua Wang [aut]
Maintainer  Jun Woo <jwoo@umn.edu>
git_url  https://git.bioconductor.org/packages/ccfindR
git_branch  RELEASE_3_18
git_last_commit  ac6bcb1
git_last_commit_date  2023-10-24
Repository  Bioconductor 3.18
Date/Publication  2024-03-06

R topics documented:

assignCelltype ........................................... 3
basis ....................................................... 4
basis,scNMFSet-method ................................. 5
basis<- .................................................. 6
basis<-,scNMFSet-method ............................... 6
build_tree ............................................... 7
ccfindR .................................................. 8
cell_map .................................................. 8
cluster_id ............................................... 9
coeff ...................................................... 10
coeff,scNMFSet-method ................................. 10
coeff<- .................................................. 11
coeff<-,scNMFSet-method ............................... 11
colData,scNMFSet-method ............................. 12
colData<-,scNMFSet,ANY-method ....................... 13
counts,scNMFSet-method .............................. 13
counts<-,scNMFSet-method ............................ 14
dbasis ..................................................... 15
dbasis,scNMFSet-method ............................... 15
dbasis<- .................................................. 16
dbasis<-,scNMFSet-method ............................... 16
dcoeff .................................................... 17
dcoeff,scNMFSet-method ............................... 17
dcoeff<- .................................................. 18
dcoeff<-,scNMFSet-method ............................... 18
factorize ............................................... 19
feature_map .............................................. 20
filter_cells ............................................. 21
filter_genes ............................................. 22
gene_map ................................................ 23
measure .................................................. 24
measure,scNMFSet-method ............................. 25
measure<- ............................................... 26
measure<-,scNMFSet-method ........................... 26
meta_gene.cv ............................................ 27
meta_genes .............................................. 28
assignCelltype

Cell type assignment via GSEA

Description

Computes GSEA enrichment score of marker sets in meta gene list

Usage

assignCelltype(obj, rank, gset, gene_names = NULL, p = 0, remove.na = FALSE, p.value = FALSE, nperm = 1000, progress.bar = TRUE, grp.prefix = c("IG"))

Arguments

<table>
<thead>
<tr>
<th>obj</th>
<th>Object of class scNMFSet.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>Rank to examine</td>
</tr>
<tr>
<td>gset</td>
<td>List of gene sets to be used as markers</td>
</tr>
<tr>
<td>gene_names</td>
<td>Names of genes to be used for meta-gene identification</td>
</tr>
<tr>
<td>p</td>
<td>Enrichment score exponent.</td>
</tr>
</tbody>
</table>
basis

remove.na Remove gene sets with no overlap
p.value Estimate p values using permutation
nperm No. of permutation replicates
progress.bar Display progress bar for p value computation
grouprefix Gene name prefix to search for with wildcard matches in query

Details

If `obj` is of class `scNMFSet`, it computes meta gene list using `meta_gene.cv`. Otherwise, `obj` is expected to be a data frame of the same structure as the output of `meta_gene.cv`; the number of rows same as the total number of metagenes per cluster, three columns per each cluster (gene name, meta-gene score, and coefficient of variation). The argument `gset` is a list of gene sets to be checked for enrichment in each cluster meta gene list. The enrichment score is computed using the GSEA algorithm (Subramanian et al. 2005).

Value

Matrix of enrichment score statistics with cell types in rows and clusters in columns

References


Examples

dir <- system.file('extdata',package='ccfindR')
pbmc <- read_10x(dir)
pbmc <- vb_factorize(pbmc, ranks=5)
meta <- meta_gene.cv(pbmc,rank=5, gene_names=rowData(pbmc)[,2])
markers <- list('B cell'='c('CD74','IG','HLA'),
     'CD8+ T'='c('CD8A','CD8B','GZMK','CCR7','LTB'),
     'CD4+ T'='c('CD3D','CD3E','IL7R','LEF1'),
     'NK'='c('GNLY','NKG7','GZMA','GZMH'),
     'Macrophage'='c('S100A8','S100A9','CD14','LYZ','CFD'))
gsea <- assignCelltype(meta, rank=5, gset=markers, grp.prefix=c('IG','HLA'))
gsea

basis

Basis matrices in an Object

Description

Retrieve or set the basis matrices \( W \) from factorization in an object
Usage

basis(object)

Arguments

object Object of class scNMFSet

Details

After factorization, basis matrices corresponding to each rank value are stored as elements of a list, which is in slot basis of object of class scNMFSet. basis(object) will return the list of matrices. basis(object) \( \leftarrow \) value can be used to modify it.

Value

Either NULL or a list of same length as ranks(object), whose elements are basis matrices derived from factorization under each rank value.

Examples

```r
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s,ranks=seq(2,4))
basis(s)[[1]]
```
Generics for basis matrix assignment

Description
Access and modify basis matrices

Usage
```r
basis(object) <- value
```

Arguments
- `object`: Object of class `scNMFSet`
- `value`: Basis matrix to be substituted

Value
Input object with updated basis matrices

Examples
```r
set.seed(1)
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=3)
basis(s)[[1]] <- apply(basis(s)[[1]],seq(1,2),round,digits=3)
basis(s)
```

Modify basis matrices

Description
Access and modify basis matrices

Usage
```r
## S4 replacement method for signature 'scNMFSet'
basis(object) <- value
```

Arguments
- `object`: Object of class `scNMFSet`
- `value`: Basis matrix to be substituted
build_tree

Value

Input object with updated basis matrices

Examples

```r
set.seed(1)
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=3)
basis(s)[[1]] <- apply(basis(s)[[1]],c(1,2),round,digits=3)
basis(s)
```

---

**build_tree**  
*Build tree connecting clusters at different ranks*

Description

Build tree connecting clusters at different ranks

Usage

```r
build_tree(object, rmax)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Object of class scNMFSet</td>
</tr>
<tr>
<td>rmax</td>
<td>Maximum rank at which tree branching stops</td>
</tr>
</tbody>
</table>

Value

List containing the tree structure

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSet(x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
tree <- build_tree(s,rmax=5)
tree
```
ccfindR  

**ccfindR: Cancer Clone FindeR**

**Description**

This package contains tools and utilities for cell-type discovery using single-cell transcriptomic data while evaluating significance of the depth of clustering (Woo et al. 2019).

**References**


**cell_map**  

*Plot heatmap of clustering coefficient matrix*

**Description**

Retrieve a coefficient matrix $H$ derived from factorization by rank value and generate heatmap of its elements.

**Usage**

```r
cell_map(object, rank, main = "Cells", ...)
```

**Arguments**

- **object**  
  Object of class `scNMFSet`.
- **rank**  
  Rank value for which the cell map is to be displayed. The object must contain the corresponding slot: one element of `coeff(object)[[k]]` for which `ranks(object)[[k]] == rank`.
- **main**  
  Title of plot.
- **...**  
  Other arguments to be passed to `heatmap`, `image`, and `plot`.

**Value**

`NULL`
**Examples**

```r
set.seed(1)
x <- simulate_data(nfeatures=10, nsamples=c(20,20,60))
rownames(x) <- seq_len(10)
colnames(x) <- seq_len(100)
s <- scNMFSet(count=x, rowData=seq_len(10), colData=seq_len(100))
s <- vb_factorize(s, ranks=seq(2,5))
plot(s)
cell_map(s, rank=3)
```

---

### Description

Use factorization results in an object to assign cells into clusters.

### Usage

```r
cluster_id(object, rank = 2)
```

### Arguments

- **object**: Object of class `scNMFSet`
- **rank**: Rank value whose factor matrices are to be used for assignment.

### Value

Vector of length equal to the number of cells containing cluster ID numbers of each cell.

### Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50, ncol=100, rank=5)
s <- scNMFSet(count=x$x)
s <- vb_factorize(s, ranks=seq(2,8), nrun=5)
cid <- cluster_id(s, rank=5)
table(cid)
```
Description
Retrieve or set the coefficient matrices from factorization in an object

Usage
coeff(object)

Arguments
object Object of class scNMFSet.

Details
After factorization, coefficient matrices \( H \) corresponding to each rank value are stored as elements of a list, which is in slot coeff of object of class scNMFSet. coeff(object) will return the list of matrices. coeff(object) <- value can be used to modify it.

Value
Either NULL or a list of same length as ranks(object), whose elements are coefficient matrices derived from factorization under each rank value.

Examples
s <- scNMFSets(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s,ranks=seq(2,4))
coeff(s)[[1]]
Value
List of coefficient matrices

Description
Access and modify coefficient matrices

Usage
coeff(object) <- value

Arguments
object
Object of class scNMFSet
value
Coefficient matrix to be substituted

Value
Input object with updated coefficient matrices

Examples
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=3)
coeff(s)[[1]] <- apply(coeff(s)[[1]],c(1,2),round,digits=2)
coeff(s)

Description
Modify coefficient matrices

Usage
## S4 replacement method for signature 'scNMFSet'
coeff(object) <- value
Arguments

| object | Object of class scNMFSet |
| value  | Coefficient matrix to be substituted |

Value

Input object with updated coefficient matrices

Examples

```r
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=3)
coeff(s)[[1]] <- apply(coeff(s)[[1]],c(1,2),round,digits=2)
coeff(s)
```

---

Sample annotation accessor

Description

Sample annotation accessor

Usage

```r
## S4 method for signature 'scNMFSet'
colData(x)
```

Arguments

| x         | Object containing sample annotation |

Value

Column annotation DataFrame

Examples

```r
library(S4Vectors)
x <- matrix(rpois(n=12,lambda=3),4,3)
rownames(x) <- seq_len(4)
colnames(x) <- c('a','b','c')
s <- scNMFSet(count=x,rowData=seq_len(4),colData=c('a','b','c'))
cols <- DataFrame(tissue=c('tissue1','tissue1','tissue2'))
rownames(cols) <- c('a','b','c')
colData(s) <- cols
ts
```
Description

Cell annotation assignment

Usage

## S4 replacement method for signature 'scNMFSet,ANY'
colData(x) <- value

Arguments

x Object containing cell annotation
value DataFrame to be substituted

Value

Updated column annotation

Examples

library(S4Vectors)
x <- matrix(rpois(n=12, lambda=3), 4, 3)ownames(x) <- seq_len(4)
colnames(x) <- c('a','b','c')
s <- scNMFSet(count=x, rowData=seq_len(4), colData=c('a','b','c'))
cols <- DataFrame(tissue=c('tissue1','tissue1','tissue2'))
rownames(cols) <- c('a','b','c')
colData(s) <- cols
s

counts,scNMFSet-method

Accessor for count matrix

Description

Accessor for count matrix

Usage

## S4 method for signature 'scNMFSet'
counts(object)
counts<-,scNMFSet-method

Arguments

object

Object containing count matrix

Value

Count matrix

Examples

s <- scNMFSet(count = matrix(rpois(n=12,lambda=3),3,4))
counts(s)

count<-,scNMFSet-method

Assignment of count matrix

Description

Count matrix can be modified

Usage

## S4 replacement method for signature 'scNMFSet'
counts(object) <- value

Arguments

object

Object containing count

value

Matrix-like object for replacement

Value

Object with updated count

Examples

mat <- matrix(rpois(n=12,lambda=3),3,4)
s <- scNMFSet(count = mat)
counts(s) <- mat^2
counts(s)
### dbasis

**Description**

Basis SD matrix accessor

**Usage**

```r
dbasis(object)
```

**Arguments**

- `object` Object containing dbasis matrix

**Value**

List of dbasis matrices

### dbasis,scNMFSet-method

**Description**

Basis SD matrix accessor

**Usage**

```r
## S4 method for signature 'scNMFSet'

dbasis(object)
```

**Arguments**

- `object` Object containing basis standard deviation (SD) matrix

**Value**

List of dbasis matrices
dbasis<-,scNMFSets-method

Basis SD matrix assignment

Description

Basis SD matrix assignment

Usage

dbasis(object) <- value

Arguments

object Object containing dbasis matrix
value List for assignment

Value

Updated object

dbasis<-,scNMFSets-method

Modify dbasis matrices

Description

Access and modify dbasis matrices

Usage

## S4 replacement method for signature 'scNMFSets'
dbasis(object) <- value

Arguments

object Object of class scNMFSets
value Basis SD matrix to be substituted

Value

Modified object
**dcoeff**  

*Coefficient SD matrix accessor*

---

**Description**

Coeff SD matrix accessor

**Usage**

dcoeff(object)

**Arguments**

- object: Object containing dcoeff matrix

**Value**

List of dcoeff matrices

---

**dcoeff,scNMFSSet-method**  

*Coefficient SD matrix accessor*

---

**Description**

Coefficient SD matrix accessor

**Usage**

```
## S4 method for signature 'scNMFSet'

dcoeff(object)
```

**Arguments**

- object: Object containing coefficient standard deviation (SD) matrix

**Value**

List of dcoeff matrices
**dcoeff<-,scNMFS<-.method**

---

**dcoeff<-**  
_Coeff SD matrix assignment_

### Description

Coeff SD matrix assignment

### Usage

```r
dcoeff(object) <- value
```

### Arguments

- **object**  
  Object containing dcoeff matrix

- **value**  
  List for assignment

### Value

Updated object

---

**dcoeff<-,scNMFS<-.method**  
_Modify dcoeff matrices_

### Description

Access and modify dcoeff matrices

### Usage

```r
## S4 replacement method for signature 'scNMFSSet'
dcoeff(object) <- value
```

### Arguments

- **object**  
  Object of class `scNMFSSet`

- **value**  
  Coeff SD matrix to be substituted

### Value

Updated object
factorize

Maximum likelihood factorization

Description
Performs single or multiple rank NMF factorization of count matrix using maximum likelihood

Usage
factorize(object, ranks = 2, nrun = 20, randomize = FALSE,
          nsmpl = 1, verbose = 2, progress.bar = TRUE, Itmax = 10000,
          ncnn.step = 40, criterion = "likelihood", linkage = "average",
          Tol = 1e-05, store.connectivity = FALSE)

Arguments
- object: scNMFSet object containing count matrix.
- ranks: Rank for factorization; can be a vector of multiple values.
- nrun: No. of runs with different initial guess.
- randomize: Boolean; if TRUE, input matrix is randomized.
- nsmpl: No. of randomized samples to average over.
- verbose: The verbosity level: 3, each iteration output printed; 2, each run output printed; 1, each randomized sample output printed; 0, silent.
- progress.bar: Display progress bar when nrun > 1 and verbose = 1.
- Itmax: Maximum no. of iteration.
- ncnn.step: Minimum no. of steps with no change in connectivity matrix to achieve convergence.
- criterion: If 'likelihood', iteration stops when fractional changes in likelihood is below tolerance Tol. If criterion = 'connectivity', iteration stops when connectivity matrix does not change for at least ncnn.step steps.
- linkage: Method to be sent to hclust in calculating cophenetic correlation.
- Tol: Tolerance for checking convergence with criterion = 'likelihood'.
- store.connectivity: Returns a list also containing connectivity data.

Details
The main input is the scNMFSet object with count matrix. This function performs non-negative factorization and fills in the empty slots basis, coeff, and ranks.

When run with multiple values of ranks, factorization is repeated for each rank and the slot measure contains quality measures of the ranks. The quality measure likelihood is negative the KL distance of the fit to the target. With nrun > 1, the likelihood is the maximum among all runs.
The quality measure dispersion is the scalar measure of how far the connectivity matrix is from 0, 1. With increasing \( n_{\text{run}} \), dispersion decreases from 1. \( n_{\text{run}} \) should be chosen such that dispersion does not change appreciably. With randomization, count matrix of object is shuffled. \( n_{\text{smpl}} \) can be used to average over multiple permutations. This averaging applies to each quality measure under a given rank.

**Value**

Object of class \texttt{scNMFSet} with factorization slots filled.

**Examples**

```r
cat
set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60,40,30))
s <- scNMFSet(count=x)
s <- factorize(s,ranks=seq(2,8),nrun=5)
plot(s)
```

---

**feature_map**

*Plot heatmap of basis matrix*

**Description**

Generate heatmap of features derived from factorization of count data.

**Usage**

```r
feature_map(object, basis.matrix = NULL, rank, markers = NULL,
            subtract.mean = TRUE, log = TRUE, max.per.cluster = 10,
            feature.names = NULL, perm = NULL, main = "Feature map",
            cscale = NULL, cex.cluster = 1, cex.feature = 0.5, mar = NULL,
            ...)```

**Arguments**

- **object** Object of class \texttt{scNMFSet}.
- **basis.matrix** Basis matrix can be supplied instead of \texttt{object}.
- **rank** Rank value for which the gene map is to be displayed. The object must contain the corresponding slot (one element of \texttt{basis(object)[[k]]} for which \texttt{ranks(object)[[k]]==rank}.
- **markers** Vector of gene names containing markers to be included in addition to the metagenes. All entries of \texttt{rowData(object)} matching them will be added to the metagene list.
- **subtract.mean** Process each rows of basis matrix \( W \) by standardization using the mean of elements within the row.
- **log** If TRUE, \texttt{subtract.mean} uses geometric mean and division. Otherwise, use arithmetic mean and subtraction.
filter_cells

max.per.cluster  Maximum number of metagenes per cluster.
feature.names Names to be used in the plot for features.
perm Permutation of cluster IDs.
main Main title.
cscale Colors for heatmap.
cex.cluster Cluster ID label size.
cex.feature Feature ID label size.
mar Margins for graphics::par.
... Other arguments to be passed to image, and plot.

Details

This function uses image() and is more flexible than gene_map.
If object contains multiple ranks, only the requested rank’s basis matrix W will be displayed. As in gene_map, the features displayed in rows are selected by "max" scheme

Value

NULL

Examples

set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60))
rownames(x) <- seq_len(10)

set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60))
rownames(x) <- seq_len(10)
colnames(x) <- seq_len(100)
s <- scNMFSet(count=x, rowData=seq_len(10), colData=seq_len(100))
s <- vb_factorize(s,ranks=seq(2,5))
plot(s)
feature_map(s, rank=3)

filter_cells  Filter cells with quality control criteria

Description

Remove low quality cell entries from object

Usage

filter_cells(object, umi.min = 0, umi.max = Inf, plot = TRUE, remove.zeros = TRUE)
filter_genes

Filter genes with quality control criteria

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>scNMFSet object</td>
</tr>
<tr>
<td>umi.min</td>
<td>Minimum UMI count for cell filtering</td>
</tr>
<tr>
<td>umi.max</td>
<td>Maximum UMI count for cell filtering</td>
</tr>
<tr>
<td>plot</td>
<td>If TRUE, the UMI count distribution of all cells will be displayed. Cells selected are colored red.</td>
</tr>
<tr>
<td>remove.zeros</td>
<td>Remove rows/columns containing zeros only</td>
</tr>
</tbody>
</table>

Details

Takes as input scNMFSet object and plots histogram of UMI counts for each cell. Optionally, cells are filtered using minimum and maximum UMI counts. The resulting object is returned after removing empty rows and columns, if any.

Value

scNMFSet object with cells filtered.

Examples

```r
set.seed(1)
s <- scNMFSet(matrix(stats::rpois(n=1200,lambda=3),40,30))
s <- filter_cells(s, umi.min=10^2.0, umi.max=10^2.1)
```

Description

Select genes with high relative variance in count data for further analysis

Usage

```r
filter_genes(object, markers = NULL, vmr.min = 0, min.cells.expressed = 0, max.cells.expressed = Inf, rescue.genes = FALSE, progress.bar = TRUE, save.memory = FALSE, plot = TRUE, log = "xy", cex = 0.5)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>scNMFSet object</td>
</tr>
<tr>
<td>markers</td>
<td>A vector containing marker genes to be selected. All rows in rowData that contain columns matching this set will be selected.</td>
</tr>
<tr>
<td>vmr.min</td>
<td>Minimum variance-to-mean ratio for gene filtering.</td>
</tr>
<tr>
<td>min.cells.expressed</td>
<td>Minimum no. of cells expressed for gene filtering.</td>
</tr>
</tbody>
</table>
gene_map  

max.cells.expressed
Maximum no. of cells expressed for gene filtering.

rescue.genes
Selected additional genes whose (non-zero) count distributions have at least one mode.

progress.bar
Display progress of mode-gene scan or VMR calculation with save.memory = TRUE.

save.memory
For a very large number of cells, calculate VMR row by row while avoiding calls to as.matrix(). Progress bar will be displayed unless progress.bar=FALSE.

plot
Plot the distribution of no. of cells expressed vs. VMR.

log
Axis in log-scale, c('x','y','xy').

cex
Symbol size for each gene in the plot.

Details
Takes as input scNMFSet object and scatterplot no. of cells expressed versus VMR (variance-to-mean ratio) for each gene. Optionally, genes are filtered using minimum VMR together with a range of no. of cells expressed.

Value
Object of class scNMFSet.

Examples

```r
set.seed(1)
s <- scNMFSet(matrix(stats::rpois(n=1200,lambda=3),40,30))
s <- filter_genes(s,vmr.min=1.0,min.cells.expressed=28, rescue.genes=FALSE)
```

---

**gene_map**

Plot heatmap of metagene matrix

---

**Description**

Generate heatmap of metagenes derived from factorization of count data.

**Usage**

gene_map(object, rank, markers = NULL, subtract.mean = TRUE, log = TRUE, max.per.cluster = 10, Colv = NA, gene.names = NULL, main = "Genes", col = NULL, ...)
gene_map

Arguments

object Object of class scNMFSet.
rank Rank value for which the gene map is to be displayed. The object must contain the corresponding slot (one element of basis(object)[[k]] for which ranks(object)[[k]]==rank.
markers Vector of gene names containing markers to be included in addition to the metagenes. All entries of rowData(object) matching them will be added to the metagene list.
subtract.mean Process each rows of basis matrix $W$ by standardization using the mean of elements within the row.
log If TRUE, subtract.mean uses geometric mean and division. Otherwise, use arithmetic mean and subtraction.
max.per.cluster Maximum number of metagenes per cluster.
Colv NA suppresses reordering and dendrogram of clusters along the column. See heatmap.
gene.names Names to be used in the plot for genes.
main Title of plot.
col Colors for the cluster panels on the left and top.
... Other arguments to be passed to heatmap, image, and plot.

Details

Wrapper for heatmap to display metagenes and associated basis matrix element magnitudes. Factorization results inside an object specified by its rank value will be retrieved, and metagene sets identified from clusters.

If object contains multiple ranks, only the requested rank’s basis matrix $W$ will be displayed. The genes displayed in rows are selected by "max" scheme [Carmona-Saez, BMC Bioinformatics (2006), https://doi.org/10.1186/1471-2105-7-54]: for each cluster (k in 1:ncol), rows of $W$ are sorted by decreasing order of $W[,k]$. Marker genes for k are those among the top nmarker for which $W[,k]$ is maximum within each row.

Value

NULL

Examples

set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60))rownames(x) <- seq_len(10)colnames(x) <- seq_len(100)s <- scNMFSet(count=x,rowData=seq_len(10), colData=seq_len(100))s <- vb_factorize(s,ranks=seq(2,5))plot(s)gene_map(s, rank=3)
**measure**

*Factorization measures in an Object*

**Description**

Retrieve or set factorization measures in an object

**Usage**

```
measure(object)
```

**Arguments**

- `object` Object of class `scNMFSet`.

**Details**

Factorization under multiple rank values lead to measures stored in a data frame inside a slot `measure`. In maximum likelihood using `factorize`, this set of quality measures include dispersion and cophenetic coefficients for each rank. In Bayesian factorization using `vb_factorize`, log evidence for each rank is stored. `measure(object)` will return the data frame. `measure(object) <- value` can be used to modify it.

**Value**

Either `NULL` or a data frame containing measures.

**Examples**

```r
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s,ranks=seq(2,4))
measure(s)
```

**measure,scNMFSet-method**

*Rank measure accessor*

**Description**

Rank measure accessor

**Usage**

```r
## S4 method for signature 'scNMFSet'
measure(object)
```
Arguments

object Object containing measure

Value

Data frame of measure

Description

Can be used to access and modify factorization measure

Usage

\[
\text{measure}(\text{object}) \leftarrow \text{value}
\]

Arguments

object Object of class `scNMFSet`
value Measure to be substituted

Value

Input object with updated measure

Examples

\[
s <- \text{scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))}
s <- \text{vb_factorize(s, ranks=3)}
\text{measure}(s)[,-1] \leftarrow \text{apply(measure}(s)[,-1], c(1,2), \text{round(digits=3)})
\text{measure}(s)
\]
Arguments

object  Object of class scNMFSet
value  Measure to be substituted

Value

Input object with updated measure

Examples

s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=3)
measure(s)[,-1] <- apply(measure(s)[,-1], c(1,2), round,digits=3)
measure(s)

Description

Generates meta gene table with coefficient of variation

Usage

meta_gene.cv(object = NULL, rank, basis.matrix = NULL, dbasis = NULL,
max.per.cluster = 100, gene_names = NULL, subtract.mean = TRUE,
log = TRUE, cv.max = Inf)

Arguments

object  Main object containing factorization outcome
rank  Rank for which meta gene is to be found
basis.matrix  Basis matrix to work with. Only necessary when object is NULL.
dbasis  Variance of basis matrix. Only necessary when object is NULL.
max.per.cluster  Maximum meta genes per cluster.
gene_names  Name of genes. If NULL, will be taken from row names.
subtract.mean  Standardize magnitudes of basis elements by subtracting mean
log  Use geometric mean.
cv.max  Upper bound for CV in selecting meta genes.

Value

Data frame with meta genes and their CV in each column.
meta_genes

Find metagenes from basis matrix

Description
Retrieve a basis matrix from an object and find metagenes.

Usage
meta_genes(object, rank, basis.matrix = NULL, max.per.cluster = 10,
gene_names = NULL, subtract.mean = TRUE, log = TRUE)

Arguments
object Object of class scNMFSet.
rank Rank value for which metagenes are to be found.
basis.matrix Instead of an object containing basis matrices, the matrix itself can be provided.
max.per.cluster Maximum number of metagenes per cluster.
gene_names Names of genes to replace row names of basis matrix.
subtract.mean Standardize the matrix elements with means within each row.
log Use geometric mean and division instead of arithmetic mean and subtraction with subtract.mean.

Value
List of vectors each containing metagene names of clusters.

Examples
set.seed(1)
x <- simulate_data(nfeatures=10, nsamples=c(20,20,60))
rownames(x) <- seq_len(10)
colnames(x) <- seq_len(100)
s <- scNMFSet(count=x, rowData=seq_len(10), colData=seq_len(100))
s <- vb_factorize(s, ranks=seq(2,5))
meta_genes(s, rank=4)
newick

Generate Newick format tree string from tree list object

Description
Generate Newick format tree string from tree list object

Usage

newick(tree, parent = "1.1", string = "")

Arguments

tree Tree list object from build_tree
parent Parent ID
string Newick string of parent tree

Value
String of newick tree

Examples

set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSet(x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
tree <- build_tree(s,rmax=5)
nw <- newick(tree=tree)
nw

normalize_count

Normalize count data

Description
Rescale count matrix entries such that all cells have the same library size.

Usage

normalize_count(object)

Arguments

object scNMFSet object.
Details

For analysis purposes, it is sometimes useful to rescale integer count data into floats such that all cells have the same median counts. This function will calculate the median of all UMI counts of cells (total number of RNAs derived from each cell). All count data are then rescaled such that cells have uniform UMI count equal to the median.

Value

scNMFSet object with normalized count data.

Examples

```r
library(Matrix)
set.seed(1)
s <- scNMFSet(count=matrix(rpois(n=1200,lambda=3),40,30))
colMeans(counts(s))
s <- normalize_count(s)
colMeans(counts(s))
```

<table>
<thead>
<tr>
<th>optimal_rank</th>
<th>Determine optimal rank</th>
</tr>
</thead>
</table>

Description

Takes as main argument scNMFSet object containing factorized output and estimate the optimal rank.

Usage

```r
optimal_rank(object, df = 10, BF.threshold = 3, type = NULL, m = NULL)
```

Arguments

- **object** scNMFSet object containing factorization output, or data frame containing the rank-evidence profile.
- **df** Degrees of freedom for split fit. Upper bound is the total number of data points (number of rank values scanned).
- **BF.threshold** Bayes factor threshold for statistical threshold.
- **type** c(1, 2). Type 1 is where there is a clear maximum. Type 2 is where marginal likelihood reaches a maximal level and stays constant. If omitted, the type will be inferred from data.
- **m** Number of features (e.g., genes) in the count matrix. Only necessary when object is of type data.frame.
Details

The input object is used along with Bayes factor threshold to determine the heterogeneity type (1 or 2) and the optimal rank. If evidence(rank 1)/evidence(rank2) > BF.threshold, rank 1 is favorable than rank 2.

Value

List containing type and ropt (optimal rank).

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50, ncol=100, rank=5)
s <- scNMFSet(x$x)
s <- vb_factorize(s, ranks=seq(2,8), nrun=5)
plot(s)
optimal_rank(s)
```

Description

Gene variance to mean ratio and the number of expressing cells are plotted.

Usage

```r
plot_genes(object, vmr = NULL, ncexpr = NULL, selected_genes = NULL,
variable_genes = NULL, mode_genes = NULL, marker_genes = NULL,
save.memory = FALSE, progress.bar = TRUE, log = "xy", cex = 0.5)
```

Arguments

- **object**: Object containing count data
- **vmr**: Variance to mean ratio (VMR)
- **ncexpr**: Number of cells expressing each gene
- **selected_genes**: Logical vector specifying genes selected
- **variable_genes**: Logical vector specifying genes with high VMR
- **mode_genes**: Logical vector specifying genes with nonzero modes
- **marker_genes**: Logical vector specifying marker genes
- **save.memory**: If TRUE, calculate VMR using slower method to save memory. Not used when gene lists are supplied.
- **progress.bar**: Display progress bar for VMR calculation. Not used when gene lists are supplied.
- **log**: Axis in log-scale, c('x', 'y', 'xy').
- **cex**: Symbol size for genes (supplied to plot()).
Details

This function can be called separately or is also called within `filter_genes` by default. In the latter case, parameters other than `object` will have been already filled. If called separately with `NULL` gene lists, VMR is recalculated but gene selection is not done.

Value

NULL

Examples

```r
set.seed(1)
s <- scNMFSets(matrix(stats::rpois(n=1200,lambda=3),40,30))
plot_genes(s)
```

---

plot_tree  
Plot cluster tree

Description

Visualize the output of `build_tree` as a dendrogram.

Usage

```r
plot_tree(tree, direction = "rightwards", cex = 0.7, ...)
```

Arguments

- `tree`: List containing tree structure. Output from `build_tree`
- `direction`: c('rightwards', 'downwards'); the direction of dendrogram
- `cex`: Font size of edge/tip labels
- `...`: Other parameters to `plot.phylo`

Details

Uses `plot.phylo` to visualize cluster tree.

Value

NULL

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSets(x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
tree <- build_tree(s,rmax=5)
plot_tree(tree)
```
ranks

| ranks | Rank values in an Object |

**Description**

Retrieve or set the rank values in an object

**Usage**

```r
ranks(object)
```

**Arguments**

- `object` Object of class `scNMFSet`.

**Details**

Ranks for which factorization has been performed are stored in slot `ranks` of `scNMFSet` object. `ranks(object)` will return the rank vector. `ranks(object) <- value` can be used to modify it.

**Value**

Either `NULL` or vector.

**Examples**

```r
s <- scNMFSet(matrix(rpois(n=12, lambda=3), 4, 3))
s <- vb_factorize(s, ranks=seq(2,4))
ranks(s)
```

---

**ranks, scNMFSet-method  Rank accessor**

**Description**

Rank accessor

**Usage**

```r
## S4 method for signature 'scNMFSet'
ranks(object)
```

**Arguments**

- `object` Object containing rank values

**Value**

Vector of rank values
ranks<-,scNMFSet-method

**Generics for ranks assignment**

**Description**

Replace ranks slot of scNMFSet object

**Usage**

```r
ranks(object) <- value
```

**Arguments**

- **object**: Object of class scNMFSet
- **value**: Rank values (vector) to be substituted

**Value**

Input object with updated ranks

**Examples**

```r
s <- scNMFSet(count=matrix(rpois(n=12,lambda=3),4,3))
s <- vb_factorize(s, ranks=seq(2,3))
ranks(s) <- c('two', 'three')
ranks(s)
```

Modify ranks

**Description**

Replace ranks slot of scNMFSet object

**Usage**

```r
## S4 replacement method for signature 'scNMFSet'
ranks(object) <- value
```

**Arguments**

- **object**: Object of class scNMFSet
- **value**: Rank values (vector) to be substituted
Value

Input object with updated ranks

Examples

```r
s <- scNMFSet(count=matrix(rpois(n=12, lambda=3), 4, 3))
s <- vb_factorize(s, ranks=seq(2, 3))
ranks(s) <- c('two', 'three')
ranks(s)
```

Description

Read count, gene, and barcode annotation data in 10x format and create an object of class `scNMFSet`.

Usage

```r
read_10x(dir, count = "matrix.mtx", genes = "genes.tsv", barcodes = "barcodes.tsv", remove.zeros = TRUE)
```

Arguments

- `dir`: Name of directory containing data files.
- `count`: Name of count matrix file.
- `genes`: Name of gene annotation file.
- `barcodes`: Name of cell annotation file.
- `remove.zeros`: If TRUE, empty rows/columns are removed.

Details

Files for count, genes, and barcodes are assumed to be present in `dir`. Count data are in sparse "Matrix Market" format (https://math.nist.gov/MatrixMarket/formats.html).

Value

Object of class `scNMFSet`

Examples

```r
library(S4Vectors)
s <- scNMFSet(count=matrix(rpois(n=12, lambda=3), 4, 3))
rowData(s) <- DataFrame(seq_len(4))
colData(s) <- DataFrame(seq_len(3))
write_10x(s, dir='.')
s <- read_10x(dir='.')
s
```
**remove_zeros**

Remove rows or columns that are empty from an object

**Description**

Remove rows or columns that are empty from an object

**Usage**

```
remove_zeros(object)
```

**Arguments**

- `object` Object containing data

**Value**

Object with empty rows/columns removed

**Examples**

```r
s2 <- remove_zeros(s)
s2
```

---

**rename_tips**

Rename tips of trees with cell types

**Description**

Rename tips of trees with cell types

**Usage**

```
rename_tips(tree, rank, tip.labels)
```

**Arguments**

- `tree` List containing tree
- `rank` Rank value of which tip names are to be replaced
- `tip.labels` Vector of new names for tips

**Value**

List containing tree with updated tip labels
Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSet(x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
tree <- build_tree(s,rmax=5)
tree <- rename_tips(tree,rank=5,tip.labels=letters[seq_len(5)])
tree
```

rowData,scNMFSet-method

### Description

Feature annotation accessor

#### Usage

```r
## S4 method for signature 'scNMFSet'
rowData(x)
```

#### Arguments

- `x` Object containing data

#### Value

DataFrame of feature annotation

#### Examples

```r
x <- matrix(rpois(n=12,lambda=3),4,3)
rownames(x) <- seq_len(4)
colnames(x) <- seq_len(3)
s <- scNMFSet(count=x,rowData=seq_len(4),colData=seq_len(3))
rowData(s)
```
rowData<-,scNMFS method

**Gene annotation assignment**

**Description**

Gene annotation assignment

**Usage**

```r
## S4 replacement method for signature 'scNMFS'
rowData(x) <- value
```

**Arguments**

- `x`: Object containing data
- `value`: DataFrame of row annotation to be substituted

**Value**

Row annotation DataFrame

---

**scNMFS**

*Create scNMFS object*

**Description**

Object derived from `SingleCellExperiment`

**Usage**

```r
scNMFS(count = NULL, ..., remove.zeros = TRUE)
```

**Arguments**

- `count`: Count matrix
- `...`: Other parameters of `SingleCellExperiment`
- `remove.zeros`: Remove empty rows and columns

**Value**

Object of class `scNMFS`.

**Examples**

```r
count <- matrix(rpois(n=12,lambda=2),4,3)
s <- scNMFS(count=count)
s
```
Description

S4 class derived from SingleCellExperiment that can store single-cell count matrix, gene and cell annotation data frames, and factorization factors as well as quality measures for rank determination.

Usage

## S4 method for signature 'scNMFSet,ANY'
plot(x)

Arguments

x Object containing measure

Value

Object of class scNMFSet
NULL

Methods (by generic)

- plot: Plot measures of an object. For quality measures derived from maximum likelihood inference, dispersion and cophenetic will be plotted separately.
  For measure derived from Bayesian inference, log evidence as a function of rank values will be plotted.

Slots

- assays Named list for count matrix counts.
- rowData DataFrame for gene (feature) names and annotations in columns.
- colData DataFrame for cell IDs and other annotations in columns (e.g., barcodes, cell types).
- ranks Vector for rank values for which factorization has been performed.
- basis List (of length equal to that of ranks) of basis matrices W from factorization; dimension nrow x rank, where nrow is no. of rows in count.
- coeff List (of length equal to that of ranks) of coefficient matrices H from factorization; dimension rank x ncol, where ncol is no. of columns in count.
- measure Data frame of factorization quality measures for each rank (likelihood and dispersion).

Other slots inherited from SingleCellExperiment class are not explicitly used.
Examples

```r
library(S4Vectors)
# toy matrix
ngenes <- 8
ncells <- 5
mat <- matrix(rpois(n=ngenes*ncells,lambda=3),ngenes,ncells)

abc <- letters[seq_len(ngenes)]
ABC <- LETTERS[seq_len(ncells)]
gen <- DataFrame(gene_id=abc)
cell <- DataFrame(cell_id=ABC)
rownames(mat) <- rownames(gen) <- abc
colnames(mat) <- rownames(cell) <- ABC

# create scNMFS object
s <- scNMFS(count=mat,rowData=gen,colData=cell)
# alternative ways
s2 <- scNMFS(count=mat)
s2 <- scNMFS(assays=list(counts=mat))

# show dimensions
dim(s)

# show slots
rowData(s)

# modify slots
colData(s) <- DataFrame(cell_id=seq_len(ncells),
                        cell_type=c(rep('tissue1',2),
                                   rep('tissue2',ncells-2)))
colData(s)
```

**show,scNMFS-method**  
Display object

Description

Display the class and dimension of an object

Object name itself on command line or (show(object)) will display class and dimensionality

Usage

```r
## S4 method for signature 'scNMFS'
show(object)
```

Arguments

- **object**: Object of class scNMFS
simulate_data

Value

NULL

Examples

s <- scNMFSet(matrix(rpois(n=12,lambda=3),4,3))
show(s)

simulate_data Generate simulated data for factorization

Description

Use one of two schemes to generate simulated data suitable for testing factorization.

Usage

simulate_data(nfeatures, nsamples, generate.factors = FALSE,
               nfactor = 10, alpha0 = 0.5, shuffle = TRUE)

Arguments

nfeatures  Number of features m (e.g., genes).
nsamples  Vector of sample sizes in each cluster. Rank r is equal to the length of this vector. Sum of elements is the total sample size n.
generate.factors  Generate factor matrices W and H, each with dimension n x r and r x n. If FALSE, factor matrices are not used and count data are generated directly from r multinomials for m genes.
nfactor  Total RNA count of multinomials for each cluster with generate.factors = FALSE. Small nfactor will yield sparse count matrix.
alpha0  Variance parameter of Dirichlet distribution from which multinomial probabilities are sampled with generate.factors = FALSE.
shuffle  Randomly permute rows and columns of count matrix.

Details

In one scheme (generate.factors = TRUE), simulated factor matrices W and H are used to build count data X = WH. In the second scheme, factor matrices are not used and X is sampled directly from r (rank requested) sets of multinomial distributions.

Value

If generate.factors = TRUE, list of components w (basis matrix, nfeatures x rank), h (coefficient matrix, rank x ncells, where ncells is equal to n, the sum of nsamples), and x, a matrix of Poisson deviates with mean W x H. If generate.factors = FALSE, only the count matrix x is in the list.
simulate_whx

Examples

```r
set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60,40,30))
s <- scNMFSet(x)
s
```

<table>
<thead>
<tr>
<th>simulate_whx</th>
<th>Simulate factor matrices and data using priors</th>
</tr>
</thead>
</table>

Description

Under Bayesian formulation, use prior distributions of factor matrices and generate simulated data.

Usage

```r
simulate_whx(nrow, ncol, rank, aw = 0.1, bw = 1, ah = 0.1, bh = 1)
```

Arguments

- `nrow`: Number of features (genes).
- `ncol`: Number of cells (samples).
- `rank`: Rank (ncol of W, nrow of H).
- `aw`: Shape parameter of basis prior.
- `bw`: Mean of basis prior. Scale parameter is equal to `aw/bw`.
- `ah`: Shape parameter of coefficient prior.
- `bh`: Mean of coefficient prior. Scale parameter is equal to `ah/bh`.

Details

Basis W and coefficient matrices H are sampled from gamma distributions (priors) with shape (aw, ah) and mean (bw, bh) parameters. Count data X are sampled from Poisson distribution with mean values given by WH.

Value

List with elements w, h, and x, each containing basis, coefficient, and count matrices.

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSet(count=x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
plot(s)
```
vb_factorize

Bayesian NMF inference of count matrix

Description

Perform variational Bayes NMF and store factor matrices in object

Usage

vb_factorize(object, ranks = 2, nrun = 1, verbose = 2,
             progress.bar = TRUE, initializer = "random", Itmax = 10000,
             hyper.update = rep(TRUE, 4), gamma.a = 1, gamma.b = 1,
             Tol = 1e-05, hyper.update.n0 = 10, hyper.update.dn = 1,
             connectivity = TRUE, fudge = NULL, ncores = 1, useC = TRUE,
             unif.stop = TRUE)

Arguments

object scNMFSet object containing count matrix.
ranks Rank for factorization; can be a vector of multiple values.
nrun No. of runs with different initial guesses.
verbose The verbosity level: 3, each iteration output printed; 2, each run output printed;
           1, each randomized sample output printed; 0, silent.
progress.bar Display progress bar with verbose = 1 for multiple runs.
initializer If 'random', randomized initial conditions; 'svd2' for singular value decom-
               posed initial condition.
Itmax Maximum no. of iteration.
hyper.update Vector of four logicals, each indicating whether hyperparameters c(aw, bw, ah,
                bh) should be optimized.
gamma.a Gamma distribution shape parameter.
gamma.b Gamma distribution mean. These two parameters are used for fixed hyperpara-
            meters with hyper.update elements FALSE.
Tol Tolerance for terminating iteration.
hyper.update.n0 Initial number of steps in which hyperparameters are fixed.
hyper.update.dn Step intervals for hyperparameter updates.
connectivity If TRUE, connectivity and dispersion will be calculated after each run. Can be
                turned off to save memory.
fudge Small positive number used as lower bound for factor matrix elements to avoid
        singularity. If fudge = NULL (default), it will be replaced by .Machine$double.eps.
        Can be set to 0 to skip regularization.
ncores Number of processors (cores) to run. If ncores > 1, parallelization is attempted.
useC Use C++ version of updates for speed.
unif.stop Terminate if any of columns in basis matrix is uniform.
Details

The main input is the scNMFSet object with count matrix. This function performs non-negative factorization using Bayesian algorithm and gamma priors. Slots basis, coeff, and ranks are filled.

When run with multiple values of ranks, factorization is repeated for each rank and the slot measure contains log evidence and optimal hyperparameters for each rank. With \( \text{nrun} > 1 \), the solution with the maximum log evidence is stored for a given rank.

Value

Object of class scNMFSet with factorization slots filled.

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50,ncol=100,rank=5)
s <- scNMFSet(x$x)
s <- vb_factorize(s,ranks=seq(2,8),nrun=5)
plot(s)
```

Description

Use tSNE to generate two-dimensional map of coefficient matrix.

Usage

```r
visualize_clusters(object, rank, verbose = FALSE, cex = 1,
cex.names = 0.7, ...)
```

Arguments

- `object` scNMF object.
- `rank` Rank value to extract from object.
- `verbose` Print tSNE messages.
- `cex` Symbol size in tSNE plot.
- `cex.names` Font size of labels in count barplot.
- `...` Other parameters to send to Rtsne.

Details

It retrieves a coefficient matrix \( H \) from an object and use its elements to assign each cell into clusters. t-Distributed Stochastic Neighbor Embedding (t-SNE; [https://lvdmaaten.github.io/tsne/](https://lvdmaaten.github.io/tsne/)) is used to visualize the clustering in 2D. Also plotted is the distribution of cell counts for all clusters.
Value

NULL

Examples

set.seed(1)
x <- simulate_data(nfeatures=10,nsamples=c(20,20,60,40,30))
rownames(x) <- seq_len(10)
colnames(x) <- seq_len(170)
s <- scNMFSet(count=x,rowData=seq_len(10),colData=seq_len(170))
s <- vb_factorize(s,ranks=seq(2,5))
visualize_clusters(s,rank=5)

write_10x

Write 10x data files

Description

Use an object and write count and annotation files in 10x format.

Usage

write_10x(object, dir, count = "matrix.mtx", genes = "genes.tsv",
barcodes = "barcodes.tsv", quote = FALSE)

Arguments

object Object of class scNMFSet containing count data
dir Directory where files are to be written.
count File name for count matrix.
genesis File name for gene annotation.
barcodes File name for cell annotation.
quote Suppress quotation marks in output files.

Value

NULL

Examples

set.seed(1)
x <- matrix(rpois(n=12,lambda=3),4,3)
rownames(x) <- seq_len(4)
colnames(x) <- seq_len(3)
s <- scNMFSet(count=x,rowData=seq_len(4),colData=seq_len(3))
write_10x(s,dir='.'
write_meta

Write meta genes to a file

Description

Write a csv file of meta gene lists from input list

Usage

write_meta(meta, file)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta</td>
<td>List of meta genes output from meta_genes</td>
</tr>
<tr>
<td>file</td>
<td>Output file name</td>
</tr>
</tbody>
</table>

Value

NULL

Examples

```r
set.seed(1)
x <- simulate_whx(nrow=50, ncol=100, rank=5)
s <- scNMFSets(x$x)
s <- vb_factorize(s, ranks=seq(2,8), nrun=5)
plot(s)
m <- meta_genes(s, rank=5)
write_meta(m, file='meta.csv')
```

[.scNMFSets,ANY,ANY,ANY-method

Subsetting scNMFSets object

Description

Subsetting scNMFSets object

Usage

```r
## S4 method for signature 'scNMFSets,ANY,ANY,ANY'
x[i, j]
```
Arguments

- **x**: Object to be subsetted
- **i**: row index
- **j**: column index

Value

Subsetted object
## Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>[,scNMFSet,ANY,ANY,ANY-method</td>
<td>46</td>
</tr>
<tr>
<td>[,scNMFSet-method</td>
<td>46</td>
</tr>
<tr>
<td>([,scNMFSet,ANY,ANY,ANY-method)</td>
<td></td>
</tr>
<tr>
<td>assignCelltype</td>
<td>3</td>
</tr>
<tr>
<td>basis</td>
<td>4</td>
</tr>
<tr>
<td>basis,scNMFSet-method</td>
<td>5</td>
</tr>
<tr>
<td>basis&lt;-,scNMFSet-method</td>
<td>6</td>
</tr>
<tr>
<td>build_tree</td>
<td>7, 29, 32</td>
</tr>
<tr>
<td>ccfindR</td>
<td>8</td>
</tr>
<tr>
<td>ccfindR-package (ccfindR)</td>
<td>8</td>
</tr>
<tr>
<td>cell_map</td>
<td>8</td>
</tr>
<tr>
<td>cluster_id</td>
<td>9</td>
</tr>
<tr>
<td>coeff</td>
<td>10</td>
</tr>
<tr>
<td>coeff,scNMFSet-method</td>
<td>10</td>
</tr>
<tr>
<td>coeff&lt;-,scNMFSet-method</td>
<td>11</td>
</tr>
<tr>
<td>colData,scNMFSet-method</td>
<td>12</td>
</tr>
<tr>
<td>colData&lt;-,scNMFSet,ANY-method,13</td>
<td></td>
</tr>
<tr>
<td>counts,scNMFSet-method</td>
<td>13</td>
</tr>
<tr>
<td>counts&lt;-,scNMFSet-method</td>
<td>14</td>
</tr>
<tr>
<td>dbasis</td>
<td>15</td>
</tr>
<tr>
<td>dbasis,scNMFSet-method</td>
<td>15</td>
</tr>
<tr>
<td>dbasis&lt;-,scNMFSet-method</td>
<td>16</td>
</tr>
<tr>
<td>dcoeff,scNMFSet-method</td>
<td>16</td>
</tr>
<tr>
<td>dcoeff&lt;-,scNMFSet-method</td>
<td>17</td>
</tr>
<tr>
<td>dcoeff&lt;-,scNMFSet-method</td>
<td>18</td>
</tr>
<tr>
<td>factorize</td>
<td>19, 25</td>
</tr>
<tr>
<td>feature_map</td>
<td>20</td>
</tr>
<tr>
<td>filter_cells</td>
<td>21</td>
</tr>
<tr>
<td>filter_genes</td>
<td>22, 32</td>
</tr>
<tr>
<td>gene_map</td>
<td>23</td>
</tr>
<tr>
<td>heatmap</td>
<td>8, 24</td>
</tr>
<tr>
<td>image</td>
<td>8, 21, 24</td>
</tr>
<tr>
<td>measure</td>
<td>25</td>
</tr>
<tr>
<td>measure,scNMFSet-method</td>
<td>25</td>
</tr>
<tr>
<td>measure&lt;-,scNMFSet-method</td>
<td>26</td>
</tr>
<tr>
<td>meta_gene.cv</td>
<td>4, 27</td>
</tr>
<tr>
<td>meta_genes</td>
<td>28</td>
</tr>
<tr>
<td>newick</td>
<td>29</td>
</tr>
<tr>
<td>normalize_count</td>
<td>29</td>
</tr>
<tr>
<td>optimal_rank</td>
<td>30</td>
</tr>
<tr>
<td>plot</td>
<td>8, 21, 24</td>
</tr>
<tr>
<td>plot,scNMFSet,ANY-method</td>
<td></td>
</tr>
<tr>
<td>(scNMFSet-class)</td>
<td>39</td>
</tr>
<tr>
<td>plot.phylo</td>
<td>32</td>
</tr>
<tr>
<td>plot_genes</td>
<td>31</td>
</tr>
<tr>
<td>plot_tree</td>
<td>32</td>
</tr>
<tr>
<td>ranks</td>
<td>33</td>
</tr>
<tr>
<td>ranks,scNMFSet-method</td>
<td>33</td>
</tr>
<tr>
<td>ranks&lt;-,scNMFSet-method</td>
<td>34</td>
</tr>
<tr>
<td>read_10x</td>
<td>35</td>
</tr>
<tr>
<td>remove_zeros</td>
<td>36</td>
</tr>
<tr>
<td>rename_tips</td>
<td>36</td>
</tr>
<tr>
<td>rowData,scNMFSet-method</td>
<td>37</td>
</tr>
<tr>
<td>rowData&lt;-,scNMFSet-method</td>
<td>38</td>
</tr>
<tr>
<td>scNMFSet,scNMFSet-class</td>
<td>38, 39</td>
</tr>
<tr>
<td>simulate_data</td>
<td>40</td>
</tr>
<tr>
<td>simulate_whx</td>
<td>42</td>
</tr>
<tr>
<td>SingleCellExperiment</td>
<td>38, 39</td>
</tr>
<tr>
<td>vb_factorize</td>
<td>25, 43</td>
</tr>
</tbody>
</table>
INDEX

visualize_clusters, 44

write_10x, 45
write_meta, 46