Package ‘omicsGMF’

October 25, 2025

Type Package

Version 0.99.7

Date 2025-01-14

Title Dimensionality reduction of (single-cell) omics data in R using omicsGMF

Description omicsGMF is a Bioconductor package that uses the sgdGMF-framework of the \code{sgdGMF} package for highly performant
and fast matrix factorization that can be used for dimensionality reduction, visualization and imputation of omics
data. It considers data from the general exponential family as input, and therefore suits the use of both RNA-seq
(Poisson or Negative Binomial data) and proteomics data (Gaussian data). It does not require prior transformation of
counts to the log-scale, because it rather optimizes the deviances from the data family specified. Also, it allows to correct for known sample-level and feature-level covariates, therefore enabling visualization and dimensionality reduction
upon batch correction. Last but not least, it deals with missing values, and allows to impute these after matrix
factorization, useful for proteomics data. This Bioconductor package allows input of SummarizedExperiment,
SingleCellExperiment, and QFeature classes.

biocViews SingleCell, RNASeq, Proteomics, QualityControl, Preprocessing, Normalization, Visualization, DimensionReduction, Transcriptomics, GeneExpression, Sequencing, Software, DataRepresentation, MassSpectrometry

Depends R (>= 4.5.0), sgdGMF, SingleCellExperiment, scuttle, scater

Imports stats, utils, Matrix, S4Vectors, SummarizedExperiment, DelayedArray, MatrixGenerics, BiocSingular, BiocParallel, beachmat, ggplot2, methods, QFeatures

Suggests knitr, dplyr, testthat, BiocGenerics, BiocStyle, graphics, grDevices

License Artistic-2.0

URL `https://github.com/statOmics/omicsGMF`

BugReports `https://github.com/statOmics/omicsGMF/issues`

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Funding This work was supported by grants from Ghent University Special Research Fund [BOF20/GOA/023] (A.S., L.C.), Research Foundation Flanders [FWO G062219N] (A.S., L.C.) and as WOG [W005325N] (L.C.). This work was supported by EU funding within the MUR PNRR ``National Center for HPC, big data and quantum computing'' (Project no. CN00000013 CN1). D.R. was also supported by the National Cancer Institute of the National Institutes of Health (U24CA289073). The views and opinions expressed are only those of the authors and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

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

git_branch devel

git_last_commit 14caf2e

git_last_commit_date 2025-10-21

Repository Bioconductor 3.22

Date/Publication 2025-10-24

Author Alexandre Segers [aut, cre, fnd],
Cristian Castiglione [ctb],
Christophe Vanderaa [ctb],
Davide Risso [ctb, fnd],
Lieven Clement [ctb, fnd]

Maintainer Alexandre Segers <`alexandresegers@outlook.com`>

Contents

`omicsGMF-package` *omicsGMF: Dimensionality reduction of (single-cell) omics data in R using omicsGMF*

Description

omicsGMF is a Bioconductor package that uses the sgdGMF-framework of the `sgdGMF` package for highly performant and fast matrix factorization that can be used for dimensionality reduction, visualization and imputation of omics data. It considers data from the general exponential family as input, and therefore suits the use of both RNA-seq (Poisson or Negative Binomial data) and proteomics data (Gaussian data). It does not require prior transformation of counts to the log-scale, because it rather optimizes the deviances from the data family specified. Also, it allows to correct for known sample-level and feature-level covariates, therefore enabling visualization and dimensionality reduction upon batch correction. Last but not least, it deals with missing values, and allows to impute these after matrix factorization, useful for proteomics data. This Bioconductor package allows input of SummarizedExperiment, SingleCellExperiment, and QFeature classes.

Author(s)

Maintainer: Alexandre Segers <`alexandresegers@outlook.com`> [funder]

Other contributors:

- Cristian Castiglione <`cristian.castiglione@unipd.it`> [contributor]
- Christophe Vanderaa <`christophe.vanderaa@ugent.be`> [contributor]
- Davide Risso <`davide.risso@unipd.it`> [contributor, funder]
- Lieven Clement <`lieven.clement@ugent.be`> [contributor, funder]

See Also

Useful links:

- `https://github.com/statOmics/omicsGMF`
- Report bugs at `https://github.com/statOmics/omicsGMF/issues`

`calculateCVGMF` *Perform a stochastic gradient descent generalized matrix factorization (sgdGMF) on cells, based on the expression or mass spectrometry data in a SingleCellExperiment, SummarizedExperiment or QFeatures object.*

Description

Perform a stochastic gradient descent generalized matrix factorization (sgdGMF) on cells, based on the expression or mass spectrometry data in a SingleCellExperiment, SummarizedExperiment or QFeatures object.

Usage

```
calculateCVGMF(x, ...)

runCVGMF(x, ...)

## S4 method for signature 'ANY'
calculateCVGMF(
  x,
  family = gaussian(),
  ncomponents = seq(1, 10, 1),
  ntop = NULL,
  X = NULL,
  Z = NULL,
  offset = NULL,
  weights = NULL,
  subset_row = NULL,
  scale = FALSE,
  transposed = FALSE,
  BSPARAM = bsparam(),
  BPPARAM = SerialParam(),
  control.init = list(),
  control.alg = list(),
  control.cv = list(),
  penalty = list(),
  method = "sgd",
  sampling = "block"
)

## S4 method for signature 'SummarizedExperiment'
calculateCVGMF(
  x,
  ...,
  exprs_values = 1,
  assay.type = exprs_values,
  family = gaussian()
)

## S4 method for signature 'SingleCellExperiment'
calculateCVGMF(
  x,
  ...,
  exprs_values = 1,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = exprs_values,
  family = gaussian()
)

## S4 method for signature 'QFeatures'
calculateCVGMF(
  x,
  ...,
```

```
  exprs_values = NULL,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = NULL
)

## S4 method for signature 'SummarizedExperiment'
runCVGMF(x, ...)

## S4 method for signature 'SingleCellExperiment'
runCVGMF(x, ..., altexp = NULL, name = "cv_GMF")

## S4 method for signature 'QFeatures'
runCVGMF(x, ..., exprs_values = NULL, assay.type = NULL)
```

Arguments

`x`	For `calculateCVGMF`, a numeric matrix of expression counts or mass spectrometry intensities where rows are features and columns are cells. Alternatively, a SummarizedExperiment-class, SingleCellExperiment-class or QFeatures object containing such a matrix.
`...`	For the `calculateCVGMF` generic, additional arguments to pass to specific methods. For the SummarizedExperiment and SingleCellExperiment methods, additional arguments to pass to the ANY method. For the QFeatures method, additional arguments to pass to the SingleCellExperiment method. For `runCVGMF`, additional arguments to pass to `calculateCVGMF`.
`family`	The distribution family that is used for the estimation of the parameters.
`ncomponents`	Numeric vector indicating the different number of components used in cross-validation.
`ntop`	Numeric scalar specifying the number of features with the highest variances to use for dimensionality reduction. Default uses all features.
`X`	Sample-level covariate matrix. Defaults to column of ones.
`Z`	Feature-level covariate matrix. Defaults to column of ones.
`offset`	offset matrix with same dimensions as x that is added to the linear predictor. Note that if family = poisson(), this should therefore be on the log-scale.
`weights`	weight matrix with same dimensions as x that determines the weight of each observation.
`subset_row`	Vector specifying the subset of features to use for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector.
`scale`	Logical scalar, should the expression values be standardized? Not recommended for non-Gaussian data.
`transposed`	Logical scalar, is x transposed with cells in rows?
`BSPARAM`	A BiocSingularParam-class object specifying which algorithm should be used to perform the PCA. This is used in `runPCA` to put all information in the sample latent factors.
`BPPARAM`	A BiocParallelParam-class object specifying whether the cross-validation should be parallelized. If BPPARAM$workers > 1 and control.cv$parallel and control.cv$nthreads are not specified, parallelization is enabled with nthreads = BPPARAM$workers.

`control.init`	control parameters for the initialization, used in the `sgdGMF` package. See sgdgmf.init and set.control.init.
`control.alg`	control parameters for the estimation, used in the `sgdGMF` package. See sgdgmf.fit and set.control.alg.
`control.cv`	control parameters for the cross-validation, used in the `sgdGMF` package. See sgdgmf.cv and set.control.cv.
`penalty`	ridge penalty added for the estimation of the parameters in the `sgdGMF` package. see sgdgmf.fit.
`method`	estimation algorithm from the `sgdGMF` package used. See sgdgmf.fit.
`sampling`	sub-sampling strategy to use if method = "sgd". See sgdgmf.fit from the `sgdGMF` package.
`exprs_values`	Alias to `assay.type`.
`assay.type`	Integer scalar or string indicating which assay of `x` contains the values of interest.
`dimred`	String or integer scalar specifying the existing dimensionality reduction results to use.
`n_dimred`	Integer scalar or vector specifying the dimensions to use if `dimred` is specified.
`altexp`	String or integer scalar specifying an alternative experiment containing the input data.
`name`	String specifying the name to be used to store the result in the `reducedDims` of the output.

Details

sgdGMF uses sampling of the data to estimate the parameters, which can alter with different seeds. Also, cross-validation puts a random selection of values to missing. This means that the result will change slightly across different runs. For full reproducibility, users should call `set.seed` prior to running `runGMF` with such algorithms. (Note that this includes `BSPARAM=bsparam()`, which uses approximate algorithms by default.)

For feature selection and using alternative Experiments, see `runGMF`.

Value

For `calculateCVGMF`, A table containing the summary statistics of the cross-validation.

For `runCVGMF`, a SingleCellExperiment object is returned containing this table in the metadata of this object.

Author(s)

Alexandre Segers

See Also

`sgdgmf.cv`, for the underlying calculations.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runCVGMF(example_sce,
                        exprs_values="counts",
                        family = poisson(),
                        ncomponents = c(1:5))
head(metadata(example_sce)[["cv_GMF"]])
example_sce <- runGMF(example_sce,
                      exprs_values="counts",
                      family = poisson(),
                      ncomponents = 3)
reducedDimNames(example_sce)
head(reducedDim(example_sce))
```

`calculateGMF` *Perform a stochastic gradient descent generalized matrix factorization (sgdGMF) on cells or bulk samples, based on the expression or mass spectrometry data in a SingleCellExperiment, SummarizedExperiment or QFeatures object.*

Description

Perform a stochastic gradient descent generalized matrix factorization (sgdGMF) on cells or bulk samples, based on the expression or mass spectrometry data in a SingleCellExperiment, SummarizedExperiment or QFeatures object.

Usage

```
calculateGMF(x, ...)

runGMF(x, ...)

## S4 method for signature 'ANY'
calculateGMF(
  x,
  family = gaussian(),
  ncomponents = 50,
  ntop = NULL,
  X = NULL,
  Z = NULL,
  offset = NULL,
  weights = NULL,
  subset_row = NULL,
  scale = FALSE,
  transposed = FALSE,
  BSPARAM = bsparam(),
  BPPARAM = SerialParam(),
  control.init = list(),
  control.alg = list(),
  crossval = FALSE,
  control.cv = list(),
```

```
  penalty = list(),
  method = "sgd",
  sampling = "block"
)

## S4 method for signature 'SummarizedExperiment'
calculateGMF(
  x,
  ...,
  exprs_values = 1,
  assay.type = exprs_values,
  family = gaussian()
)

## S4 method for signature 'SingleCellExperiment'
calculateGMF(
  x,
  ...,
  exprs_values = 1,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = exprs_values,
  family = gaussian()
)

## S4 method for signature 'QFeatures'
calculateGMF(
  x,
  ...,
  exprs_values = NULL,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = NULL,
  family = gaussian()
)

## S4 method for signature 'SummarizedExperiment'
runGMF(x, ...)

## S4 method for signature 'SingleCellExperiment'
runGMF(x, ..., altexp = NULL, name = "GMF")

## S4 method for signature 'QFeatures'
runGMF(x, ..., exprs_values = NULL, assay.type = NULL)
```

Arguments

`x`	For `calculateGMF`, a numeric matrix of expression counts or mass spectrometry intensities where rows are features and columns are cells. Alternatively, a SummarizedExperiment-class, SingleCellExperiment-class or QFeatures object containing such a matrix.
`...`	For the `calculateGMF` generic, additional arguments to pass to specific meth-

	ods. For the SummarizedExperiment and SingleCellExperiment methods, additional arguments to pass to the ANY method. For the QFeatures method, additional arguments to pass to the SingleCellExperiment method. For `runGMF`, additional arguments to pass to `calculateGMF`.
`family`	The distribution family that is used for the estimation of the parameters.
`ncomponents`	Numeric scalar indicating the number of principal components to estimate.
`ntop`	Numeric scalar specifying the number of features with the highest variances to use for dimensionality reduction. Default uses all features.
`X`	Sample-level covariate matrix. Defaults to column of ones.
`Z`	Feature-level covariate matrix. Defaults to column of ones.
`offset`	offset matrix with same dimensions as `x` that is added to the linear predictor. Note that if family = poisson(), this should therefore be on the log-scale.
`weights`	weight matrix with same dimensions as `x` that determines the weight of each observation.
`subset_row`	Vector specifying the subset of features to use for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector.
`scale`	Logical scalar, should the expression values be standardized? Not recommended for non-Gaussian data.
`transposed`	Logical scalar, is `x` transposed with cells in rows?
`BSPARAM`	A BiocSingularParam-class object specifying which algorithm should be used to perform the PCA. This is used in `runPCA` to put all information in the sample latent factors.
`BPPARAM`	A BiocParallelParam-class object specifying whether the initialization and cross-validation should be parallelized.
`control.init`	control parameters for the initialization, used in the `sgdGMF` package. See sgdgmf.init and set.control.init.
`control.alg`	control parameters for the estimation, used in the `sgdGMF` package. See sgdgmf.fit and set.control.alg.
`crossval`	if TRUE, performs cross-validation followed by fitting a final model with the optimal number of components. Generally not recommended, as no quality control of the cross-validation is done before the final fit. See calculateCVGMF for cross-validation.
`control.cv`	control parameters for the cross-validation, used in the `sgdGMF` package. See sgdgmf.cv and set.control.cv.
`penalty`	ridge penalty added for the estimation of the parameters in the `sgdGMF` package. see sgdgmf.fit.
`method`	estimation algorithm from the `sgdGMF` package used. See sgdgmf.fit. Defaults to 'sgd' for a stochastic gradient descent optimization.
`sampling`	sub-sampling strategy to use if method = "sgd". See sgdgmf.fit from the `sgdGMF` package. Defaults to 'block' for a block-wise stochastic gradient descent optimization.
`exprs_values`	Alias to `assay.type`.
`assay.type`	Integer scalar or string indicating which assay of `x` contains the values of interest.
`dimred`	String or integer scalar specifying the existing dimensionality reduction results to use.

`n_dimred`	Integer scalar or vector specifying the dimensions to use if `dimred` is specified.
`altexp`	String or integer scalar specifying an alternative experiment containing the input data.
`name`	String specifying the name to be used to store the result in the `reducedDims` of the output.

Details

sgdGMF uses sampling of the data to estimate the parameters, which can alter with different seeds. This means that the result will change slightly across different runs. For full reproducibility, users should call `set.seed` prior to running `runGMF` with such algorithms. (Note that this includes `BSPARAM=bsparam()`, which uses approximate algorithms by default.)

Value

This section is adapted from the `scater` package manual.

For `calculateGMF`, a numeric matrix of coordinates for each cell (row) in each of `ncomponents` PCs (column).

For `runGMF`, a SingleCellExperiment object is returned containing this matrix in `reducedDims(..., name)`.

In both cases, the attributes of the PC coordinate matrix contain the following elements:

- `"rotation"`, the rotation matrix containing loadings for all features used in the analysis and for each PC.
- `"X"`, the known sample-level covariate matrix.
- `"Beta"`, the estimated parameters related to the known sample-level covariate matrix.
- `"Z"`, the known feature-level covariate matrix.
- `"Gamma"`, the estimated parameters related to the known feature-level covariate matrix.
- `"family"`, the distribution family used for the estimation of the parameters.
- `"trace"`, a trace matrix recording the optimization history of sgdGMF.
- `"summary.cv"`, only if cross-validation was performed, a summary table of the cross-validation.
- `"offset"`, only if offset is not NULL, a matrix containing the offsets.

Feature selection

This section is adapted from the `scater` package manual.

This section is relevant if `x` is a numeric matrix with features in rows and cells in columns; or if `x` is a SingleCellExperiment-class and `dimred=NULL`. In the latter, the expression values are obtained from the assay specified by `assay.type`.

The `subset_row` argument specifies the features to use for dimensionality reduction. The aim is to allow users to specify highly variable features to improve the signal/noise ratio, or to specify genes in a pathway of interest to focus on particular aspects of heterogeneity.

If `subset_row=NULL`, the `ntop` features with the largest variances are used instead. We literally compute the variances from the expression values without considering any mean-variance trend, nor considering missing values, so often a more considered choice of genes is possible, e.g., with **scran** functions. Note that the value of `ntop` is ignored if `subset_row` is specified.

If `scale=TRUE`, the expression values for each feature are standardized so that their variance is unity. This will also remove features with standard deviations below 1e-8. This is not recommended when using non-Gaussian family distributions.

Using reduced dimensions

This section is adapted from the `scater` package manual.

If `x` is a SingleCellExperiment-class, the method can be applied on existing dimensionality reduction results in `x` by setting the `dimred` argument.

The matrix of existing reduced dimensions is taken from `reducedDim(x, dimred)`. By default, all dimensions are used to compute the second set of reduced dimensions. If `n_dimred` is also specified, only the first `n_dimred` columns are used. Alternatively, `n_dimred` can be an integer vector specifying the column indices of the dimensions to use.

When `dimred` is specified, no additional feature selection or standardization is performed. This means that any settings of `ntop`, `subset_row` and `scale` are ignored.

If `x` is a numeric matrix, setting `transposed=TRUE` will treat the rows as cells and the columns as the variables/dimensions. This allows users to manually pass in dimensionality reduction results without needing to wrap them in a SingleCellExperiment-class. As such, no feature selection or standardization is performed, i.e., `ntop`, `subset_row` and `scale` are ignored.

Using alternative Experiments

This section is adapted from the `scater` package manual.

This section is relevant if `x` is a SingleCellExperiment-class and `altexp` is not `NULL`. In such cases, the method is run on data from an alternative SummarizedExperiment-class nested within `x`. This is useful for performing dimensionality reduction on other features stored in `altExp(x, altexp)`, e.g., antibody tags.

Setting `altexp` with `assay.type` will use the specified assay from the alternative SummarizedExperiment-class. If the alternative is a SingleCellExperiment, setting `dimred` will use the specified dimensionality reduction results from the alternative. This option will also interact as expected with `n_dimred`.

Note that the output is still stored in the `reducedDims` of the output SingleCellExperiment. It is advisable to use a different `name` to distinguish this output from the results generated from the main experiment's assay values.

Author(s)

Alexandre Segers

See Also

`sgdgmf.fit`, for the underlying calculations. `plotGMF`, to conveniently visualize the results. `imputeGMF`, to conveniently impute missing values.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runCVGMF(example_sce,
                         exprs_values="counts",
                         family = poisson(),
                         ncomponents = c(1:5))
example_sce <- runGMF(example_sce,
                      exprs_values="counts",
                      family = poisson(),
                      ncomponents = 3)
reducedDimNames(example_sce)
head(reducedDim(example_sce))
```

`calculateRankGMF` *Perform an eigendecomposition for model selection based on a screeplot.*

Description

Perform an eigendecomposition for model selection based on a screeplot.

Usage

```
calculateRankGMF(x, ...)

runRankGMF(x, ...)

## S4 method for signature 'ANY'
calculateRankGMF(
  x,
  family = gaussian(),
  maxcomp = 100,
  ntop = NULL,
  X = NULL,
  Z = NULL,
  offset = NULL,
  weights = NULL,
  subset_row = NULL,
  scale = FALSE,
  transposed = FALSE,
  BSPARAM = bsparam(),
  BPPARAM = SerialParam(),
  method = "oht",
  normalize = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
calculateRankGMF(
  x,
  ...,
  exprs_values = 1,
  assay.type = exprs_values,
  family = gaussian()
)

## S4 method for signature 'SingleCellExperiment'
calculateRankGMF(
  x,
  ...,
  exprs_values = 1,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = exprs_values,
```

```
  family = gaussian()
)

## S4 method for signature 'QFeatures'
calculateRankGMF(
  x,
  ...,
  exprs_values = NULL,
  dimred = NULL,
  n_dimred = NULL,
  assay.type = NULL,
  family = gaussian()
)

## S4 method for signature 'SummarizedExperiment'
runRankGMF(x, ...)

## S4 method for signature 'SingleCellExperiment'
runRankGMF(x, ..., altexp = NULL, name = "rank_GMF")

## S4 method for signature 'QFeatures'
runRankGMF(x, ..., exprs_values = NULL, assay.type = NULL)
```

Arguments

`x`	For `calculateRankGMF`, a numeric matrix of expression counts or mass spectrometry intensities where rows are features and columns are cells. Alternatively, a SummarizedExperiment-class or SingleCellExperiment-class containing such a matrix. For `runRankGMF`, a SummarizedExperiment-class, SingleCellExperiment-class or QFeatures object containing such a matrix.
`...`	For the `calculateRankGMF` generic, additional arguments to pass to specific methods such as `sgdgmf.rank`. For the SummarizedExperiment and SingleCellExperiment methods, additional arguments to pass to the ANY method. For `runRankGMF`, additional arguments to pass to `calculateRankGMF`.
`family`	The distribution family that is used for the estimation of the parameters.
`maxcomp`	Scalar indicating the maximal number of eigenvalues to compute.
`ntop`	Numeric scalar specifying the number of features with the highest variances to use for dimensionality reduction. Default uses all features.
`X`	Sample-level covariate matrix. Defaults to column of ones.
`Z`	Feature-level covariate matrix. Defaults to column of ones.
`offset`	offset matrix with same dimensions as x that is added to the linear predictor. Note that if family = poisson(), this should therefore be on the log-scale
`weights`	weight matrix with same dimensions as x that determines the weight of each observation.
`subset_row`	Vector specifying the subset of features to use for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector.
`scale`	Logical scalar, should the expression values be standardized? Not recommended for non-Gaussian data.

`transposed`	Logical scalar, is x transposed with cells in rows?
`BSPARAM`	A BiocSingularParam-class object specifying which algorithm should be used to perform the PCA. This is used in `runPCA` to put all information in the sample latent factors.
`BPPARAM`	A BiocParallelParam-class object specifying whether the cross-validation should be parallelized. If BPPARAM$workers > 1 and control.cv$parallel and control.cv$nthreads are not specified, parallelization is enabled with nthreads = BPPARAM$workers.
`method`	rank selection method, see `sgdgmf.rank`.
`normalize`	if TRUE, standardize the residual matrix for each feature.
`exprs_values`	Alias to `assay.type`.
`assay.type`	Integer scalar or string indicating which assay of x contains the values of interest.
`dimred`	String or integer scalar specifying the existing dimensionality reduction results to use.
`n_dimred`	Integer scalar or vector specifying the dimensions to use if `dimred` is specified.
`altexp`	String or integer scalar specifying an alternative experiment containing the input data.
`name`	String specifying the name to be used to store the result in the metadata of the output.

Details

sgdGMF uses sampling of the data to estimate the parameters, which can alter with different seeds. Also, cross-validation puts a random selection of values to missing. This means that the result will change slightly across different runs. For full reproducibility, users should call `set.seed` prior to running `runRankGMF` with such algorithms. (Note that this includes `BSPARAM=bsparam()`, which uses approximate algorithms by default.)

For feature selection and using alternative Experiments, see `runGMF`.

Value

A list containing the eigenvalues. If a SummarizedExperiment-class or SingleCellExperiment-class was given as input, this is stored in the metadata of this object.

Author(s)

Alexandre Segers

See Also

`sgdgmf.rank`, for the underlying calculations.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runRankGMF(example_sce,
                          exprs_values="counts",
                          family = poisson(),
                          maxcomp = 10)
head(metadata(example_sce)[["rank_GMF"]])
plotRank(example_sce)
```

`imputeGMF` *Impute missing values based on the results of runGMF.*

Description

Impute missing values based on the results of runGMF.

Usage

```
imputeGMF(x, ...)

## S4 method for signature 'ANY'
imputeGMF(x, sgdGMF_reducedDims)

## S4 method for signature 'SummarizedExperiment'
imputeGMF(
  x,
  sgdGMF_reducedDims,
  exprs_values = 1,
  assay.type = exprs_values,
  name = "imputedAssay"
)

## S4 method for signature 'SingleCellExperiment'
imputeGMF(
  x,
  reducedDimName = "GMF",
  sgdGMF_reducedDims = reducedDim(x, reducedDimName),
  exprs_values = 1,
  assay.type = exprs_values,
  name = "imputedAssay"
)

## S4 method for signature 'QFeatures'
imputeGMF(
  x,
  ...,
  reducedDimName = "GMF",
  exprs_values = NULL,
  assay.type = NULL,
  name = "imputedAssay"
)
```

Arguments

`x`	a numeric matrix of expression counts or mass spectrometry intensities containing missing values and with features in the rows and samples in columns. Alternatively, a SummarizedExperiment-class, SingleCellExperiment-class or QFeatures object containing such a matrix.
`...`	For the `imputeGMF` generic, additional arguments to pass to specific methods.

sgdGMF_reducedDims	the output obtained by runGMF or calculateGMF. If x is a SingleCellExperiment-class, sgdGMF_reducedDims is taken from reducedDim(x, reducedDimName).
exprs_values	Alias to assay.type.
assay.type	Integer scalar or string indicating which assay of x contains the values of interest.
name	New assay name included for the matrix with imputed values.
reducedDimName	the name of the reducedDim slot corresponding to the dimensionality reduction obtained with runGMF when x is a SingleCellExperiment-class or QFeatures object.

Details

Imputation is only possible after running `runGMF` using all features. Therefore, `subset_row` or `ntop` should be set to NULL when performing the matrix factorization.

Value

For SummarizedExperiment-class, SingleCellExperiment-class or QFeatures, a similar object now containing an extra assay with the imputed values.

For a matrix, a matrix with missing values imputed.

Author(s)

Alexandre Segers

See Also

`runGMF`, to conveniently obtain the matrix factorization.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- logNormCounts(example_sce)
assay(example_sce, 'logcounts')[assay(example_sce, 'logcounts') == 0] <- NA
example_sce <- runGMF(example_sce,
                      exprs_values="logcounts",
                      family = gaussian(),
                      ncomponents = 3)
example_sce <- imputeGMF(example_sce)
```

---

plotGMF

Wrapper functions to create plots for specific types of reduced dimension results in a SingleCellExperiment object, similar as the `scater` *package.*

Description

Wrapper functions to create plots for specific types of reduced dimension results in a SingleCellExperiment object, similar as the `scater` package.

Usage

```
plotGMF(object, ..., ncomponents = 2, dimred = "GMF")
```

Arguments

`object`	A SingleCellExperiment-class object.
`...`	Additional arguments to pass to `plotReducedDim` from the `scater` package.
`ncomponents`	Numeric scalar indicating the number of dimensions components to (calculate and) plot. This can also be a numeric vector, see `plotReducedDim` for details
`dimred`	A string or integer scalar indicating the reduced dimension result in `reducedDims`(object) to plot.

Details

This is a wrapper around `plotReducedDim` that uses the "GMF" slot from the `reducedDims` to obtain a dimensionality reduction plot.

Value

A `ggplot` object.

Author(s)

Alexandre Segers

See Also

`plotReducedDim`, for the underlying calculations. `plotPCA`, for a similar wrapper.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runCVGMF(example_sce,
                        exprs_values="counts",
                        family = poisson(),
                        ncomponents = c(1:5))
example_sce <- runGMF(example_sce,
                      exprs_values="counts",
                      family = poisson(),
                      ncomponents = 3)
plotGMF(example_sce)
```

`plot_cv` *Functions to create a scree plot for model selection.*

Description

Functions to create a scree plot for model selection.

Usage

```
plot_cv(x, ...)

## S4 method for signature 'ANY'
plot_cv(x, method = "mean", criteria = "dev")

plotCV(x, ..., name = "cv_GMF")
```

Arguments

`x`	Output of `sgdgmf.cv`, `calculateCVGMF` or `runCVGMF`.
`...`	For the `plot_cv` generic, additional arguments to pass to specific methods.
`method`	Function for summarization of the cross-validation over multiple folds. Default is mean.
`criteria`	The model selection criteria that is plotted. Default is 'dev' (deviance residuals), but 'mae', 'mse', 'aic' and 'bic are possible.
`name`	String specifying the name to be used to obtain the cross-validation table object in the metadata. It is possible to specify multiple names if multiple cross-validations have been ran.

Details

This function plots a screeplot based on the output of `runCVGMF`, `calculateCVGMF` or `sgdgmf.cv`.

Value

A `plot` object.

Author(s)

Alexandre Segers

See Also

`runCVGMF`, to perform the cross-validation.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runCVGMF(example_sce,
                        exprs_values="counts",
                        family = poisson(),
                        ncomponents = c(1:5))
plotCV(example_sce)
```

`screeplot_rank` *Functions to create a scree plot for model selection.*

Description

Functions to create a scree plot for model selection.

Usage

```
screeplot_rank(x, ...)

## S4 method for signature 'ANY'
screeplot_rank(
  x,
  maxcomp = length(x$lambdas),
  type = c("point", "barplot", "lines"),
  ...
)

plotRank(x, ..., name = "rank_GMF")
```

Arguments

`x`	Output of `sgdgmf.rank`, `calculateRankGMF` or `runRankGMF`.
`...`	For the `screeplot_rank` generic, additional arguments to pass to specific methods.
`maxcomp`	Numeric scalar indicating the number of eigenvalues to plot.
`type`	Type of scree plot to make: choose between 'point', 'barplot' or 'lines.
`name`	String specifying the name to be used to obtain the rank object in the metadata.

Details

This function plots a screeplot based on the output of `runRankGMF` or `sgdgmf.rank`.

Value

A plot object.

Author(s)

Alexandre Segers

See Also

`runRankGMF`, to calculate the eigenvalues.

Examples

```
example_sce <- mockSCE(ncells = 200, ngenes = 100)
example_sce <- runRankGMF(example_sce,
                         exprs_values="counts",
                         family = poisson(),
                         maxcomp = 10)
head(metadata(example_sce)[["rank_GMF"]])
plotRank(example_sce)
```

Index