

Package ‘NewWave’

March 30, 2021

Type Package

Title Negative binomial model for scRNA-seq

Version 1.0.2

Description A model designed for dimensionality reduction and batch effect removal for scRNA-seq data. It is designed to be massively parallelizable using shared objects that prevent memory duplication, and it can be used with different mini-batch approaches in order to reduce time consumption. It assumes a negative binomial distribution for the data with a dispersion parameter that can be both commonwise across gene both genewise.

Depends R (>= 4.0), SummarizedExperiment, SharedObject(>= 1.3.15)

Imports methods, SingleCellExperiment, parallel, irlba, Matrix, DelayedArray, BiocSingular, stats

Suggests testthat, rmarkdown, splatter, mclust, Rtsne, ggplot2, Rcpp, BiocStyle, knitr

License GPL-3

VignetteBuilder knitr

biocViews Software, GeneExpression, Transcriptomics, SingleCell, BatchEffect, Sequencing, Coverage, Regression

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

BugReports <https://github.com/fedeago/NewWave/issues>

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

git_branch RELEASE_3_12

git_last_commit b343548

git_last_commit_date 2020-12-21

Date/Publication 2021-03-29

Author Federico Agostinis [aut, cre],
Chiara Romualdi [aut],
Gabriele Sales [aut],
Davide Risso [aut]

Maintainer Federico Agostinis <federico.agostinis@outlook.com>

R topics documented:

newAIC	2
newAlpha	3
newBeta	4
newBIC	4
newEpsilon_alpha	5
newEpsilon_beta	5
newEpsilon_gamma	6
newEpsilon_W	6
newEpsilon_zeta	7
newFit	7
newGamma	10
newloglik	11
newLogMu	11
newmodel	12
newmodel-class	14
newMu	16
newpenalty	17
newPhi	18
newSim	18
newTheta	19
newV	20
newW	20
newWave	21
newX	23
newZeta	23
numberFactors	24
numberFeatures	24
numberParams	25
numberSamples	25
Index	27

newAIC	<i>Compute the AIC of a model given some data</i>
--------	---

Description

Given a statistical model and some data, this function computes the AIC of the model given the data, i.e., the AIC of the data under the model.

Usage

```
newAIC(model, x)

## S4 method for signature 'newmodel,matrix'
newAIC(model, x)
```

Arguments

model	an object that describes a statistical model.
x	an object that describes data.

Value

the AIC of the model.

Functions

- newAIC, newmodel, matrix-method: returns the AIC of the NB model.

Examples

```
m <- newmodel(n=5, J=10)
x <- newSim(m)
newAIC(m, x$counts)
```

newAlpha

Returns the matrix of parameters alpha

Description

Given an object that describes a matrix of negative binomial distributions, returns the matrix of parameters associated with W for the mean part (μ)

Usage

```
newAlpha(object, ...)
```

Arguments

object an object that describes a matrix of negative binomial distributions.
... Additional parameters.

Value

the matrix of alpha parameters

Examples

```
a <- newmodel(n=5, J=10)
newAlpha(a)
```

newBeta	<i>Returns the matrix of parameters beta</i>
---------	--

Description

Given an object that describes a matrix of negative binomial distributions, returns the matrix of parameters associated with X

Usage

```
newBeta(object, ...)
```

Arguments

object	an object that describes a matrix of negative binomial distributions.
...	Additional parameters.

Value

the matrix of beta parameters

Examples

```
a <- newmodel(n=5, J=10)
newBeta(a)
```

newBIC	<i>Compute the BIC of a model given some data</i>
--------	---

Description

Given a statistical model and some data, this function computes the BIC of the model given the data, i.e., the BIC of the data under the model.

Usage

```
newBIC(model, x)

## S4 method for signature 'newmodel,matrix'
newBIC(model, x)
```

Arguments

model	an object that describes a statistical model.
x	an object that describes data.

Value

the BIC of the model.

Functions

- newBIC, newmodel, matrix-method: returns the BIC of the NB model.

Examples

```
m <- newmodel(n=5, J=10)
x <- newSim(m)
newBIC(m, x$counts)
```

newEpsilon_alpha *Returns the vector of regularization parameter for alpha*

Description

Given an object describing a nb model, returns a vector of size the number of rows in the parameter alpha with the regularization parameters associated to each row.

Usage

```
newEpsilon_alpha(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Value

the regularization parameters for alpha.

Examples

```
a <- newmodel(n=5, J=10)
newEpsilon_alpha(a)
```

newEpsilon_beta *Returns the vector of regularization parameter for beta*

Description

Given an object describing a nb model, returns a vector of size the number of rows in the parameter beta with the regularization parameters associated to each row.

Usage

```
newEpsilon_beta(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Value

the regularization parameters for beta.

Examples

```
a <- newmodel(n=5, J=10)
newEpsilon_beta(a)
```

newEpsilon_gamma	<i>Returns the vector of regularization parameter for gamma</i>
------------------	---

Description

Given an object describing a nb model, returns a vector of size the number of columns in the parameter gamma with the regularization parameters associated to each row.

Usage

```
newEpsilon_gamma(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Value

the regularization parameters for gamma.

Examples

```
a <- newmodel(n=5, J=10)
newEpsilon_gamma(a)
```

newEpsilon_W	<i>Returns the vector of regularization parameter for W</i>
--------------	---

Description

Given an object describing a nb model, returns a vector of size the number of columns in the parameter W with the regularization parameters associated to each column.

Usage

```
newEpsilon_W(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Value

the regularization parameters for W.

Examples

```
a <- newmodel(n=5, J=10)
newEpsilon_W(a)
```

newEpsilon_zeta	<i>Returns the regularization parameter for the dispersion parameter</i>
-----------------	--

Description

The regularization parameter penalizes the variance of zeta, the log of the dispersion parameters across samples.

Usage

```
newEpsilon_zeta(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Value

the regularization parameters for zeta.

Examples

```
a <- newmodel(n=5, J=10)
newEpsilon_zeta(a)
```

newFit	<i>Fit a nb regression model</i>
--------	----------------------------------

Description

Given an object with the data, it fits a nb model.

Usage

```
newFit(Y, ...)  
  
## S4 method for signature 'SummarizedExperiment'  
newFit(  
  Y,  
  X,  
  V,  
  K = 2,  
  which_assay,  
  commondispersion = TRUE,  
  verbose = FALSE,  
  maxiter_optimize = 100,  
  stop_epsilon = 1e-04,  
  children = 1,  
  random_init = FALSE,  
  random_start = FALSE,  
  n_gene_disp = NULL,  
  n_cell_par = NULL,  
  n_gene_par = NULL,  
  ...  
)  
  
## S4 method for signature 'matrix'  
newFit(  
  Y,  
  X,  
  V,  
  K = 2,  
  commondispersion = TRUE,  
  verbose = FALSE,  
  maxiter_optimize = 100,  
  stop_epsilon = 1e-04,  
  children = 1,  
  random_init = FALSE,  
  random_start = FALSE,  
  n_gene_disp = NULL,  
  n_cell_par = NULL,  
  n_gene_par = NULL,  
  ...  
)  
  
## S4 method for signature 'DelayedMatrix'  
newFit(  
  Y,  
  X,  
  V,  
  K = 2,  
  commondispersion = TRUE,  
  verbose = FALSE,  
  maxiter_optimize = 100,  
  stop_epsilon = 1e-04,
```



```

    children = 1,
    random_init = FALSE,
    random_start = FALSE,
    n_gene_disp = NULL,
    n_cell_par = NULL,
    n_gene_par = NULL,
    ...
)

## S4 method for signature 'dgCMatrix'
newFit(Y, ...)

```

Arguments

Y	The matrix with the data
...	Additional parameters to describe the model, see newmodel .
X	The design matrix containing sample-level covariates, one sample per row. If missing, X will contain only an intercept.
V	The design matrix containing gene-level covariates, one gene per row. If missing, V will contain only an intercept.
K	integer. Number of latent factors(default 2).
which_assay	numeric or character. Which assay of Y to use. If missing, if 'assayNames(Y)' contains "counts" then that is used. Otherwise, the first assay is used.
commondispersion	Whether or not a single dispersion for all features is estimated (default TRUE).
verbose	Print helpful messages(default FALSE).
maxiter_optimize	maximum number of iterations for the optimization step (default 100).
stop_epsilon	stopping criterion in the optimization step, when the relative gain in likelihood is below epsilon (default 0.0001).
children	number of cores of the used cluster(default 1)
random_init	if TRUE no initializations is done(default FALSE)
random_start	if TRUE the setup of parameters is a random samplig(default FALSE)
n_gene_disp	number of genes used in mini-batch dispersion estimation approach(default NULL > all genes are used)
n_cell_par	number of cells used in mini-batch cell's related parameters estimation approach(default NULL > all cells are used)
n_gene_par	number of genes used in mini-batch gene's related parameters estimation approach(default NULL > all genes are used)

Details

By default, i.e., if no arguments other than Y are passed, the model is fitted with an intercept for the regression across-samples and one intercept for the regression across genes.

If Y is a Summarized experiment, the function uses the assay named "counts", if any, or the first assay.

Currently, if Y is a sparseMatrix, this calls the newFit method on as.matrix(Y)

Value

An object of class `newmodel` that has been fitted by penalized maximum likelihood on the data.

Methods (by class)

- `SummarizedExperiment`: Y is a `SummarizedExperiment`.
- `matrix`: Y is a matrix of counts (genes in rows).
- `DelayedMatrix`: Y is a `DelayedMatrix` of counts (genes in rows).
- `dgCMatrix`: Y is a sparse matrix of counts (genes in rows).

See Also

[model.matrix](#).

Examples

```
se <- SummarizedExperiment(matrix(rpois(60, lambda=5), nrow=10, ncol=6),
                             colData = data.frame(bio = gl(2, 3)))

m <- newFit(se, X=model.matrix(~bio, data=colData(se)))
bio <- gl(2, 3)
m <- newFit(matrix(rpois(60, lambda=5), nrow=10, ncol=6),
             X=model.matrix(~bio))
```

`newGamma`

Returns the matrix of parameters gamma

Description

Given an object that describes a matrix of negative binomial distributions, returns the matrix of parameters associated with V

Usage

```
newGamma(object, ...)
```

Arguments

`object` an object that describes a matrix of negative binomial distributions.
`...` Additional parameters.

Value

the matrix of gamma parameters

Examples

```
a <- newmodel(n=5, J=10)
newGamma(a)
```

 newloglik

Compute the log-likelihood of a model given some data

Description

Given a statistical model and some data, this function computes the log-likelihood of the model given the data, i.e., the log-probability of the data under the model.

Usage

```
newloglik(model, x, ...)

## S4 method for signature 'newmodel,matrix'
newloglik(model, x)
```

Arguments

model	an object that describes a statistical model.
x	an object that describes data.
...	additional arguments.

Value

The log-likelihood of the model given the data.

Methods (by class)

- model = newmodel, x = matrix: return the log-likelihood of the nb model.

Examples

```
m <- newmodel(n=5, J=10)
x <- newSim(m)
newloglik(m, x$counts)
```

 newLogMu

Returns the matrix of logarithm of mean parameters

Description

Given an object that describes a matrix of negative binomial distributions, returns the matrix of logarithm of mean parameters.

Usage

```
newLogMu(object)
```

Arguments

object	an object that describes a matrix of negative binomial distributions.
--------	---

Details

Note that although the user interface of `newFit` requires a $J \times n$ matrix, internally this is stored as a $n \times J$ matrix (i.e., samples in row and genes in column). Hence the parameter matrix returned by this function is of $n \times J$ dimensions.

Value

the matrix of logarithms of mean parameters

Examples

```
a <- newmodel(n=5, J=10)
newLogMu(a)
```

<code>newmodel</code>	<i>Initialize an object of class newmodel</i>
-----------------------	---

Description

Initialize an object of class newmodel

Usage

```
newmodel(
  X,
  V,
  W,
  beta,
  gamma,
  alpha,
  zeta,
  epsilon,
  epsilon_beta,
  epsilon_gamma,
  epsilon_W,
  epsilon_alpha,
  epsilon_zeta,
  n,
  J,
  K
)
```

Arguments

<code>X</code>	matrix. The design matrix containing sample-level covariates, one sample per row.
<code>V</code>	matrix. The design matrix containing gene-level covariates, one gene per row.
<code>W</code>	matrix. The factors of sample-level latent factors.
<code>beta</code>	matrix or NULL. The coefficients of <code>X</code> in the regression of <code>mu</code> .
<code>gamma</code>	matrix or NULL. The coefficients of <code>V</code> in the regression of <code>mu</code> .

alpha	matrix or NULL. The coefficients of W in the regression of μ .
zeta	numeric. A vector of log of inverse dispersion parameters.
epsilon	nonnegative scalar. Regularization parameter.
epsilon_beta	nonnegative scalar. Regularization parameter for beta.
epsilon_gamma	nonnegative scalar. Regularization parameter for gamma.
epsilon_W	nonnegative scalar. Regularization parameter for W .
epsilon_alpha	nonnegative scalar. Regularization parameter for alpha
epsilon_zeta	nonnegative scalar. Regularization parameter for zeta.
n	integer. Number of samples.
J	integer. Number of genes.
K	integer. Number of latent factors.

Details

This is a wrapper around the `new()` function to create an instance of class `newmodel`. Rarely, the user will need to create a `newmodel` object from scratch, as typically this is the result of `newFit`.

If any of X , V , W matrices are passed, n , J , and K are inferred. Alternatively, the user can specify one or more of n , J , and K .

The regularization parameters can be set by a unique parameter `epsilon` or specific values for the different regularization parameters can also be provided. If only `epsilon` is specified, the other parameters take the following values:

- `epsilon_beta = epsilon/J`
- `epsilon_gamma = epsilon/n`
- `epsilon_W = epsilon/n`
- `epsilon_alpha = epsilon/J`
- `epsilon_zeta = epsilon`

We empirically found that large values of `epsilon` provide a more stable estimation of W .

A call with no argument has the following default values: $n = 50$, $J = 100$, $K = 0$, `epsilon=J`.

Although it is possible to create new instances of the class by calling this function, this is not the most common way of creating `newmodel` objects. The main use of the class is within the `newFit` function.

Value

an object of class `newmodel`.

Examples

```
a <- newmodel()
numberSamples(a)
numberFeatures(a)
numberFactors(a)
```

`newmodel-class`*Class newmodel*

Description

Objects of this class store all the values needed to work with a negative binomial model, as described in the vignette. They contain all information to fit a model by penalized maximum likelihood or simulate data from a model.

Usage

```
## S4 method for signature 'newmodel'  
show(object)  
  
## S4 method for signature 'newmodel'  
numberSamples(x)  
  
## S4 method for signature 'newmodel'  
numberFeatures(x)  
  
## S4 method for signature 'newmodel'  
numberFactors(x)  
  
## S4 method for signature 'newmodel'  
newX(object)  
  
## S4 method for signature 'newmodel'  
newV(object)  
  
## S4 method for signature 'newmodel'  
newLogMu(object)  
  
## S4 method for signature 'newmodel'  
newMu(object)  
  
## S4 method for signature 'newmodel'  
newZeta(object)  
  
## S4 method for signature 'newmodel'  
newPhi(object)  
  
## S4 method for signature 'newmodel'  
newTheta(object)  
  
## S4 method for signature 'newmodel'  
newEpsilon_beta(object)  
  
## S4 method for signature 'newmodel'  
newEpsilon_gamma(object)  
  
## S4 method for signature 'newmodel'
```

```

newEpsilon_W(object)

## S4 method for signature 'newmodel'
newEpsilon_alpha(object)

## S4 method for signature 'newmodel'
newEpsilon_zeta(object)

## S4 method for signature 'newmodel'
newW(object)

## S4 method for signature 'newmodel'
newBeta(object)

## S4 method for signature 'newmodel'
newGamma(object)

## S4 method for signature 'newmodel'
newAlpha(object)

```

Arguments

object an object of class newmodel.
x an object of class newmodel.

Details

For the full description of the model see the model vignette. Internally, the slots are checked so that the matrices are of the appropriate dimensions: in particular, X, O and W need to have n rows, V needs to have J rows, zeta must be of length J.

Value

numberSamples returns the number of samples; numberFeatures returns the number of features; numberFactors returns the number of latent factors.

Methods (by generic)

- show: show useful info on the object.
- numberSamples: returns the number of samples.
- numberFeatures: returns the number of features.
- numberFactors: returns the number of latent factors.
- newX: returns the sample-level design matrix for mu.
- newV: returns the gene-level design matrix for mu.
- newLogMu: returns the logarithm of the mean of the non-zero component.
- newMu: returns the mean of the non-zero component.
- newZeta: returns the log of the inverse of the dispersion parameter.
- newPhi: returns the dispersion parameter.
- newTheta: returns the inverse of the dispersion parameter.
- newEpsilon_beta: returns the regularization parameters for beta.

- newEpsilon_gamma: returns the regularization parameters for gamma.
- newEpsilon_W: returns the regularization parameters for W.
- newEpsilon_alpha: returns the regularization parameters for alpha.
- newEpsilon_zeta: returns the regularization parameters for zeta.
- newW: returns the matrix W of inferred sample-level covariates.
- newBeta: returns the matrix beta of inferred parameters.
- newGamma: returns the matrix gamma of inferred parameters.
- newAlpha: returns the matrix alpha of inferred parameters.

Slots

X matrix. The design matrix containing sample-level covariates, one sample per row.

V matrix. The design matrix containing gene-level covariates, one gene per row.

X_intercept logical. TRUE if X contains an intercept.

V_intercept logical. TRUE if V contains an intercept.

W matrix. The factors of sample-level latent factors.

beta matrix or NULL. The coefficients of X in the regression.

gamma matrix or NULL. The coefficients of V in the regression.

alpha matrix. The weight of sample-level latent factors.

zeta numeric. A vector of log of inverse dispersion parameters.

epsilon_beta nonnegative scalar. Regularization parameter for beta

epsilon_gamma nonnegative scalar. Regularization parameter for gamma

epsilon_W nonnegative scalar. Regularization parameter for W

epsilon_alpha nonnegative scalar. Regularization parameter for alpha

epsilon_zeta nonnegative scalar. Regularization parameter for zeta

newMu

Returns the matrix of mean parameters

Description

Given an object that describes a matrix of negative binomial distributions, returns the matrix of mean parameters.

Usage

```
newMu(object)
```

Arguments

object an object that describes a matrix of negative binomial distributions.

Details

Note that although the user interface of `newFit` requires a J x n matrix, internally this is stored as a n x J matrix (i.e., samples in row and genes in column). Hence the parameter matrix returned by this function is of n x J dimensions.

Value

the matrix of mean parameters

Examples

```
a <- newmodel(n=5, J=10)
newMu(a)
```

newpenalty

Compute the penalty of a model

Description

Given a statistical model with regularization parameters, compute the penalty.

Usage

```
newpenalty(model)

## S4 method for signature 'newmodel'
newpenalty(model)
```

Arguments

model an object that describes a statistical model with regularization parameters.

Value

The penalty of the model.

Methods (by class)

- newmodel: return the penalization.

Examples

```
m <- newmodel(K=2)
newpenalty(m)
```

newPhi	<i>Returns the vector of dispersion parameters</i>
--------	--

Description

Given an object that describes a matrix of negative binomial negative binomial distributions, returns the vector of dispersion parameters phi.

Usage

```
newPhi(object)
```

Arguments

object an object that describes a matrix of negative binomial. distributions.

Value

the vector of dispersion parameters

Examples

```
a <- newmodel(n=5, J=10)
newPhi(a)
```

newSim	<i>Simulate counts from a negative binomial model</i>
--------	---

Description

Given an object that describes negative binomial distribution, simulate counts from the distribution.

Usage

```
newSim(object, seed, ...)

## S4 method for signature 'newmodel'
newSim(object, seed)
```

Arguments

object an object that describes a matrix of negative binomial.
seed an optional integer to specify how the random number generator should be initialized with a call to `set.seed`. If missing, the random generator state is not changed.
... additional arguments.

Value

A list with the following elements.

- `counts` the matrix with the simulated counts.
- `dataNB` the data simulated from the negative binomial.
- `dataDropout` the data simulated from the binomial process.
- `zeroFraction` the fraction of zeros.

Methods (by class)

- `newmodel`: simulate from a nb distribution.

Examples

```
a <- newmodel(n=5, J=10)
newSim(a)
```

`newTheta`

Returns the vector of inverse dispersion parameters

Description

Given an object that describes a matrix of negative binomial negative binomial distributions, returns the vector of inverse dispersion parameters `theta`.

Usage

```
newTheta(object)
```

Arguments

`object` an object that describes a matrix of negative binomial distributions.

Value

the vector of inverse dispersion parameters `theta`

Examples

```
a <- newmodel(n=5, J=10)
newTheta(a)
```

newV	<i>Returns the gene-level design matrix for mu</i>
------	--

Description

Given an object that describes a matrix of negative binomial distributions, returns the gene-level design matrix for mu

Usage

```
newV(object, ...)
```

Arguments

object	an object that describes a matrix of negative binomial distributions.
...	Additional parameters.

Value

the gene-level design matrix for mu

Examples

```
a <- newmodel(n=5, J=10)
newV(a)
```

newW	<i>Returns the low-dimensional matrix of inferred sample-level covariates W</i>
------	---

Description

Given an object that contains the fit of a nb-WaVE model, returns the matrix W of low-dimensional matrix of inferred sample-level covariates.

Usage

```
newW(object)
```

Arguments

object	a <code>newmodel</code> object, typically the result of <code>newFit</code> .
--------	---

Value

the matrix W of inferred sample-level covariates.

Examples

```
a <- newmodel(n=5, J=10)
newW(a)
```

newWave	<i>Perform dimensionality reduction using a nb regression model with gene and cell-level covariates.</i>
---------	--

Description

Given an object with the data, it performs dimensionality reduction using a nb regression model with gene and cell-level covariates.

Usage

```
newWave(Y, ...)

## S4 method for signature 'SummarizedExperiment'
newWave(
  Y,
  X,
  V,
  K = 2,
  which_assay,
  commondispersion = TRUE,
  verbose = FALSE,
  maxiter_optimize = 100,
  stop_epsilon = 1e-04,
  children = 1,
  random_init = FALSE,
  random_start = FALSE,
  n_gene_disp = NULL,
  n_cell_par = NULL,
  n_gene_par = NULL,
  ...
)
```

Arguments

Y	The SummarizedExperiment with the data
...	Additional parameters to describe the model, see newmodel .
X	The design matrix containing sample-level covariates, one sample per row. If missing, X will contain only an intercept. If Y is a SummarizedExperiment object, X can be a formula using the variables in the colData slot of Y.
V	The design matrix containing gene-level covariates, one gene per row. If missing, V will contain only an intercept. If Y is a SummarizedExperiment object, V can be a formula using the variables in the rowData slot of Y.
K	integer. Number of latent factors(default 2).
which_assay	numeric or character. Which assay of Y to use. If missing, if 'assayNames(Y)' contains "counts" then that is used. Otherwise, the first assay is used.
commondispersion	Whether or not a single dispersion for all features is estimated (default TRUE).
verbose	Print helpful messages(default FALSE).

<code>maxiter_optimize</code>	maximum number of iterations for the optimization step (default 100).
<code>stop_epsilon</code>	stopping criterion in the optimization step, when the relative gain in likelihood is below epsilon (default 0.0001).
<code>children</code>	number of cores of the used cluster(default 1)
<code>random_init</code>	if TRUE no initializations is done(default FALSE)
<code>random_start</code>	if TRUE the setup of parameters is a random samplig (default FALSE)
<code>n_gene_disp</code>	number of genes used in mini-batch dispersion estimation approach(default NULL > all genes are used)
<code>n_cell_par</code>	number of cells used in mini-batch cells related parameters estimation approach(default NULL > all cells are used)
<code>n_gene_par</code>	number of genes used in mini-batch genes related parameters estimation approach(default NULL > all genes are used)

Details

For visualization (heatmaps, ...), please use the normalized values. It corresponds to the deviance residuals when the W is not included in the model but the gene and cell-level covariates are. As a results, when W is not included in the model, the deviance residuals should capture the biology. Note that we do not recommend to use the normalized values for any downstream analysis (such as clustering, or differential expression), but only for visualization.

If one has already fitted a model using `newmodel`, the object containing such model can be used as input of `newWave` to save the resulting W into a `SummarizedExperiment` and optionally compute residuals and normalized values, without the need for re-fitting the model.

By default `newWave` uses all genes to estimate W . However, we recommend to use the top 1,000 most variable genes for this step. In general, a user can specify any custom set of genes to be used to estimate W , by specifying either a vector of gene names, or a single character string corresponding to a column of the `rowData`.

Note that if both `which_genes` is specified and at least one among `observationalWeights`, `imputedValues`, `residuals`, and `normalizedValues` is TRUE, the model needs to be fit twice.

Value

An object of class `SingleCellExperiment`; the dimensionality reduced matrix is stored in the `reducedDims` slot and optionally normalized values and residuals are added in the list of assays.

Methods (by class)

- `SummarizedExperiment`: Y is a `SummarizedExperiment`.

Examples

```
se <- SummarizedExperiment(matrix(rpois(60, lambda=5), nrow=10, ncol=6),
                           colData = data.frame(bio = gl(2, 3)))

m <- newWave(se, X="~bio")
```

newX	<i>Returns the sample-level design matrix for mu</i>
------	--

Description

Given an object that describes a matrix of negative binomial distributions, returns the sample-level design matrix for mu

Usage

```
newX(object, ...)
```

Arguments

object	an object that describes a matrix of negative binomial distributions.
...	Additional parameters.

Value

the sample-level design matrix for mu

Examples

```
a <- newmodel(n=5, J=10)
newX(a)
```

newZeta	<i>Returns the vector of log of inverse dispersion parameters</i>
---------	---

Description

Given an object that describes a matrix of negative binomial negative binomial distributions, returns the vector zeta of log of inverse dispersion parameters

Usage

```
newZeta(object)
```

Arguments

object	an object that describes a matrix of negative binomial distributions.
--------	---

Value

the vector zeta of log of inverse dispersion parameters

Examples

```
a <- newmodel(n=5, J=10)
newZeta(a)
```

numberFactors	<i>Generic function that returns the number of latent factors</i>
---------------	---

Description

Given an object that describes a dataset or a model involving latent factors, this function returns the number of latent factors.

Usage

```
numberFactors(x)
```

Arguments

x an object that describes a dataset or a model involving latent factors

Value

the number of latent factors

Examples

```
a <- newmodel(n=5, J=10)
numberFactors(a)
```

numberFeatures	<i>Generic function that returns the number of features</i>
----------------	---

Description

Given an object that describes a dataset or a model involving features, this function returns the number of features

Given an object that describes a dataset or a model, it returns the number of features.

Usage

```
numberFeatures(x)
```

```
numberFeatures(x)
```

Arguments

x an object that describes a dataset or a model.

Value

the number of features

the number of features.

Examples

```
a <- newmodel(n=5, J=10)
numberFeatures(a)
a <- newmodel(n=5, J=10)
numberFeatures(a)
```

numberParams	<i>Generic function that returns the total number of parameters of the model</i>
--------------	--

Description

Given an object that describes a model or a dataset, it returns total number of parameters of the model.

Usage

```
numberParams(model)

## S4 method for signature 'newmodel'
numberParams(model)
```

Arguments

model an object that describes a dataset or a model.

Value

the total number of parameters of the model.

Functions

- numberParams, newmodel-method: returns the total number of parameters in the model.

Examples

```
a <- newmodel(n=5, J=10)
numberParams(a)
```

numberSamples	<i>Generic function that returns the number of samples</i>
---------------	--

Description

Given an object that describes a dataset or a model involving samples, this function returns the number of samples.

Given an object that describes a model or a dataset, it returns the number of samples.

Usage

```
numberSamples(x)
```

```
numberSamples(x)
```

Arguments

x an object that describes a dataset or a model.

Value

the number of samples

the number of samples.

Examples

```
a <- newmodel(n=5, J=10)
numberSamples(a)
a <- newmodel(n=5, J=10)
numberSamples(a)
```

Index

model.matrix, [10](#)

newAIC, [2](#)

newAIC, newmodel, matrix-method (newAIC), [2](#)

newAlpha, [3](#)

newAlpha, newmodel-method (newmodel-class), [14](#)

newBeta, [4](#)

newBeta, newmodel-method (newmodel-class), [14](#)

newBIC, [4](#)

newBIC, newmodel, matrix-method (newBIC), [4](#)

newEpsilon_alpha, [5](#)

newEpsilon_alpha, newmodel-method (newmodel-class), [14](#)

newEpsilon_beta, [5](#)

newEpsilon_beta, newmodel-method (newmodel-class), [14](#)

newEpsilon_gamma, [6](#)

newEpsilon_gamma, newmodel-method (newmodel-class), [14](#)

newEpsilon_W, [6](#)

newEpsilon_W, newmodel-method (newmodel-class), [14](#)

newEpsilon_zeta, [7](#)

newEpsilon_zeta, newmodel-method (newmodel-class), [14](#)

newFit, [7](#), [12](#), [13](#), [16](#), [20](#)

newFit, DelayedMatrix-method (newFit), [7](#)

newFit, dgCMatrix-method (newFit), [7](#)

newFit, matrix-method (newFit), [7](#)

newFit, SummarizedExperiment-method (newFit), [7](#)

newGamma, [10](#)

newGamma, newmodel-method (newmodel-class), [14](#)

newloglik, [11](#)

newloglik, newmodel, matrix-method (newloglik), [11](#)

newLogMu, [11](#)

newLogMu, newmodel-method (newmodel-class), [14](#)

newmodel, [9](#), [12](#), [13](#), [20–22](#)

newmodel-class, [14](#)

newMu, [16](#)

newMu, newmodel-method (newmodel-class), [14](#)

newpenalty, [17](#)

newpenalty, newmodel-method (newpenalty), [17](#)

newPhi, [18](#)

newPhi, newmodel-method (newmodel-class), [14](#)

newSim, [18](#)

newSim, newmodel-method (newSim), [18](#)

newTheta, [19](#)

newTheta, newmodel-method (newmodel-class), [14](#)

newV, [20](#)

newV, newmodel-method (newmodel-class), [14](#)

newW, [20](#)

newW, newmodel-method (newmodel-class), [14](#)

newWave, [21](#)

newWave, SummarizedExperiment-method (newWave), [21](#)

newX, [23](#)

newX, newmodel-method (newmodel-class), [14](#)

newZeta, [23](#)

newZeta, newmodel-method (newmodel-class), [14](#)

numberFactors, [24](#)

numberFactors, newmodel-method (newmodel-class), [14](#)

numberFeatures, [24](#)

numberFeatures, newmodel-method (newmodel-class), [14](#)

numberParams, [25](#)

numberParams, newmodel-method (numberParams), [25](#)

numberSamples, [25](#)

numberSamples, newmodel-method (newmodel-class), [14](#)

show, `newmodel-method (newmodel-class)`,
[14](#)