Package 'LineagePulse'

March 30, 2021

Type Package

Title Differential expression analysis and model fitting for single-cell RNA-seq data

Version 1.10.0

Date 2017-10-09

Description LineagePulse is a differential expression and expression model fitting package tailored to single-cell RNA-seq data (scRNA-seq). LineagePulse accounts for batch effects, dropout and variable sequencing depth. One can use LineagePulse to perform longitudinal differential expression analysis across pseudotime as a continuous coordinate or between discrete groups of cells (e.g. pre-defined clusters or experimental conditions). Expression model fits can be directly extracted from LineagePulse.

License Artistic-2.0

Encoding UTF-8

Imports BiocParallel, circlize, compiler, ComplexHeatmap, ggplot2, gplots, grDevices, grid, knitr, Matrix, methods, RColorBrewer, SingleCellExperiment, splines, stats, SummarizedExperiment, utils

biocViews ImmunoOncology, Software, StatisticalMethod, TimeCourse, Sequencing, DifferentialExpression, GeneExpression, CellBiology, CellBasedAssays, SingleCell

Collate 'srcLineagePulse_classUnions.R' 'main_LineagePulse.R' 'srcLineagePulse_calcNormConst.R' 'srcLineagePulse_calcPostDrop.R' 'srcLineagePulse_classLineagePulseObject.R' 'srcLineagePulse_decompressParameters.R' 'srcLineagePulse_evalDropoutModel.R' 'srcLineagePulse_evalImpulseModel.R' 'srcLineagePulse_evalLogLik.R' 'srcLineagePulse_fitMeanDispersion.R' 'srcLineagePulse_fitDropout.R' 'srcLineagePulse_fitWrapperLP.R' 'srcLineagePulse_fitModel.R' 'srcLineagePulse_getFits.R' 'srcLineagePulse_initialiseDispModel.R' 'srcLineagePulse_initialiseDropModel.R' 'srcLineagePulse_initialiseImpulseParameters.R' 'srcLineagePulse_initialiseMuModel.R' 'srcLineagePulse_plotGene.R' 'srcLineagePulse_processSCData.R' 'srcLineagePulse_runHypothesisTests.R'

'srcLineagePulse_simulateDataSet.R'
'srcLineagePulse_sortGeneTrajectories.R'

RoxygenNote 6.0.1

VignetteBuilder knitr

BugReports https://github.com/YosefLab/LineagePulse/issues

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

git_branch RELEASE_3_12

git_last_commit 9ccb5a6

git_last_commit_date 2020-10-27

Date/Publication 2021-03-29

Author David S Fischer [aut, cre],
Fabian Theis [ctb],
Nir Yosef [ctb]

Maintainer David S Fischer <david.fischer@helmholtz-muenchen.de>

R topics documented:

`accessors` *LineagePulseObject accession methods*

Description

Get internal data of LineagePulse output object.

Usage

```
dfAnnotationProc(objLP)

dfResults(objLP)

lsMuModelH0(objLP)

lsMuModelH1(objLP)

lsMuModelConst(objLP)

lsMuModelH0_NB(objLP)
```

```
lsMuModelH1_NB(objLP)

lsDispModelH0(objLP)

lsDispModelH1(objLP)

lsDispModelConst(objLP)

lsDispModelH0_NB(objLP)

lsDispModelH1_NB(objLP)

lsDropModel(objLP)

lsFitConvergence(objLP)

matCountsProc(objLP)

matWeights(objLP)

scaDFSplinesDisp(objLP)

scaDFSplinesMu(objLP)

strReport(objLP)

vecAllGenes(objLP)

vecConfoundersDisp(objLP)

vecConfoundersMu(objLP)

scaOmega(objLP)

boolFixedPopulations(objLP)

vecH0Pop(objLP)

vecNormConst(objLP)

strVersion(objLP)

strReport(objLP)
```

Arguments

objLP	(LineagePulse-Object) A LineagePulse output object to extract from.

Value

The internal data object specified by the function.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 10,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# get hidden objects within LineagePulse object
dfAnnotationProc <- dfAnnotationProc(objLP)
dfResults <- dfResults(objLP)
lsMuModelH0 <- lsMuModelH0(objLP)
lsMuModelH1 <- lsMuModelH1(objLP)
lsMuModelConst <- lsMuModelConst(objLP)
lsMuModelH0_NB <- lsMuModelH0_NB(objLP)
lsMuModelH1_NB <- lsMuModelH1_NB(objLP)
lsDispModelH0 <- lsDispModelH0(objLP)
lsDispModelH1 <- lsDispModelH1(objLP)
lsDispModelConst <- lsDispModelConst(objLP)
lsDropModel <- lsDropModel(objLP)
lsFitConvergence <- lsFitConvergence(objLP)
matCountDataProc <- matCountsProc(objLP)
matWeights <- matWeights(objLP)
scaDFSplinesDisp <- scaDFSplinesDisp(objLP)
scaDFSplinesMu <- scaDFSplinesMu(objLP)
strReport <- strReport(objLP)
vecAllGenes <- vecAllGenes(objLP)
vecConfoundersDisp <- vecConfoundersDisp(objLP)
vecConfoundersMu <- vecConfoundersMu(objLP)
scaOmega <- scaOmega(objLP)
boolFixedPopulations <- boolFixedPopulations(objLP)
vecH0Pop <- vecH0Pop(objLP)
vecNormConst <- vecNormConst(objLP)
strVersion <- strVersion(objLP)
```

`calcNormConst` *Compute size factors for a LineagePulse-object*

Description

Either use externally supplied normalisation constants or set these to one.

Usage

```
calcNormConst(objLP, vecNormConstExternal)
```

Arguments

objLP	(LineagePulse-object) Object to fit normalization constants on.
vecNormConstExternal	(numeric vector number of cells) Model scaling factors supplied by user, one per cell.

Value

objLP (LineagePulse-object) Object with fit normalization constants.

Author(s)

David Sebastian Fischer

See Also

Called by `runLineagePulse`.

---

calcPostDrop_Matrix *Calculate posterior of drop-out*

Description

Calculates posterior of observation being a drop-out for a matrix.

Usage

```
calcPostDrop_Matrix(matCounts, lsMuModel, lsDispModel, lsDropModel,
  vecIDs = NULL)
```

Arguments

matCounts	(count matrix genes x cells) Observed read counts, not observed are NA.
lsMuModel	(list) Object containing description of gene-wise mean parameter models.
lsDispModel	(list) Object containing description of gene-wise dispersion parameter models.
lsDropModel	(list) Object containing description of cell-wise drop-out parameter models.
vecIDs	(vector of strings) [Default NULL] Gene IDs for which posteriors of drop-out are to be computed.

Value

matZ (numeric matrix genes x cells) Posterior probability of observation not being generated by drop-out.

Author(s)

David Sebastian Fischer

See Also

Called by `plotGene`.

`calcPostDrop_Vector` *Calculate posterior of drop-out*

Description

Calculates posterior of observation being a drop-out for a vector.

Usage

```
calcPostDrop_Vector(vecMu, vecDisp, vecDrop, vecboolZero, vecboolNotZero)
```

Arguments

`vecMu`	(numeric vector samples) Negative binomial mean parameters of samples.
`vecDisp`	(numeric vector samples) Negative binomial mean parameters of samples.
`vecDrop`	(numeric vector samples) Drop out rates of samples.
`vecboolZero`	(bool vector samples) Whether observation is zero.
`vecboolNotZero`	(bool vector samples) Whether observation is real and non-zero.

Value

vecZ (numeric vector samples) Posterior probability of observation not being generated by drop-out.

Author(s)

David Sebastian Fischer

See Also

For matrices `calcPostDrop_Matrix`.

`decompressDispByGene` *Compute dispersion parameter estimates from mean parameter model for a gene*

Description

Takes the model type and computes one dispersion parameter for each cell for one gene.

Usage

```
decompressDispByGene(vecDispModel, lsvecBatchModel = NULL, lsDispModelGlobal,
  vecInterval = NULL)
```

Arguments

vecDispModel	(numerical vector number of model parameters) Parameters of dispersion model for given gene.
lsvecBatchModel	(list) [Defaul NULL] List of vectors of batch correction models for dispersion parameter.
lsDispModelGlobal	(list) Object containing meta-data of gene-wise dispersion parameter models.
vecInterval	(integer vector length target cells) [Default NULL] Positions of cells in ordering, for which parameters are to be computed. Default all cells.

Value

vecDisp (numerical vector number of cells) Dispersion parameter estimates for given gene (one per cell for given gene).

Author(s)

David Sebastian Fischer

See Also

Called by fitZINB.

---

decompressDispByGeneMM

Compute mean parameter estimate matrix from mean parameter model for a gene (strDispModel == "MM")

Description

Compute mean parameter estimate matrix from mean parameter model for a gene (strDispModel == "MM")

Usage

```
decompressDispByGeneMM(vecDispModel, lsvecBatchModel = NULL,
  lsDispModelGlobal, vecInterval = NULL)
```

Arguments

vecDispModel	(numerical vector number of model parameters) Parameters of dispersion model for given gene.
lsvecBatchModel	(list) [Defaul NULL] List of vectors of batch correction models for dispersion parameter.
lsDispModelGlobal	(list) Object containing meta-data of gene-wise dispersion parameter models.
vecInterval	(integer vector length target cells) [Default NULL] Positions of cells in ordering, for which parameters are to be computed. Default all cells.

Value

matDisp (matrix number of cells x number of mixtures) Dispersion parameter estimates for given gene given the mean model.

Author(s)

David Sebastian Fischer

`decompressDropoutRateByCell` *Compute dropout rate parameter estimates from dropout rate model for a cell*

Description

Compute dropout rate parameter estimates from dropout rate model for a cell.

Usage

```
decompressDropoutRateByCell(vecDropModel, vecMu = NULL,
  matPiConstPredictors = NULL, lsDropModelGlobal)
```

Arguments

`vecDropModel`	(numerical vector number of model parameters) offset parameter, log(mu) paramter, parameters belonging to constant predictors Parameters of dropout rate model for given cell.
`vecMu`	(numerical vector number of genes) Mean parameter estimates of all genes for given cell.
`matPiConstPredictors`	(numerical matrix genes x number of constant model predictors) Other model predictors than offset and the dynamically changing mean parameter. Examples are GC- content and other gene-specific properties. This would be the global parameters as listed in the other decompression function. Here those are not a list as there is only one object.
`lsDropModelGlobal`	(list) Object containing meta-data of cell-wise dropout parameter models.

Value

vecPi (numerical vector number of cells) Dispersion parameter estimates for given gene (one per cell for given gene).

Author(s)

David Sebastian Fischer

See Also

Called by `fitZINB`.

`decompressDropoutRateByGene` *Compute dropout rate parameter estimates from dropout rate model for a gene*

Description

Compute dropout rate parameter estimates from dropout rate model for a gene.

Usage

```
decompressDropoutRateByGene(matDropModel, vecMu = NULL,
  vecPiConstPredictors = NULL, lsDropModelGlobal)
```

Arguments

`matDropModel`	(numerical matrix cell x number of model parameters) offset parameter, log(mu) parameter, parameters belonging to constant predictors Parameters of dropout rate model for all cells.
`vecMu`	(numerical vector number of genes) Mean parameter estimates of all cells for given gene.
`vecPiConstPredictors`	(numerical vector number of constant model predictors) Other model predictors than offset and the dynamically changing mean parameter. Examples are GC-content and other gene-specific properties. This would be the global parameters as listed in the other decompression function. Here those are not a list as there is only one object.
`lsDropModelGlobal`	(list) Object containing meta-data of cell-wise dropout parameter models.

Value

vecPi (numerical vector number of cells) Dispersion parameter estimates for given gene (one per cell for given gene).

Author(s)

David Sebastian Fischer

See Also

Called by `fitZINB`.

`decompressMeansByGene` *Compute mean parameter estimates from mean parameter model for a gene*

Description

Takes the model type and computes one mean parameter for each cell for one gene.

Usage

```
decompressMeansByGene(vecMuModel, lsvecBatchModel = NULL, lsMuModelGlobal,
  vecInterval = NULL)
```

Arguments

`vecMuModel`	(numerical vector number of model parameters) Parameters of mean model for given gene.
`lsvecBatchModel`	(list) List of vectors of batch correction models for mean parameter.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`vecInterval`	(integer vector length target cells) [Default NULL] Positions of cells in ordering, for which parameters are to be computed. Default all cells.

Value

vecMu (numerical vector number of cells) Mean parameter estimates for given gene given the mean model.

Author(s)

David Sebastian Fischer

See Also

Called by `fitZINB`.

`decompressMuByGeneMM` *Compute mean parameter estimate matrix from mean parameter model for a gene (strMuModel == "MM")*

Description

Compute mean parameter estimate matrix from mean parameter model for a gene (strMuModel == "MM")

Usage

```
decompressMuByGeneMM(vecMuModel, lsvecBatchModel = NULL, lsMuModelGlobal,
  vecInterval = NULL)
```

Arguments

vecMuModel	(numerical vector number of model parameters) Parameters of mean model for given gene (means by mixture).
lsvecBatchModel	(list) [Defaul NULL] List of vectors of batch correction models for mean parameter.
lsMuModelGlobal	(list) Object containing meta-data of gene-wise mean parameter models.
vecInterval	(integer vector length target cells) [Default NULL] Positions of cells in ordering, for which parameters are to be computed. Default all cells.

Value

matMu (matrix number of cells x number of mixtures) Mean parameter estimates for given gene given the mean model.

Author(s)

David Sebastian Fischer

---

evalDropoutModel *Compute value of logistic dropout function given with scale boundaries*

Description

Computes value of logistic function for one observation and uses offset to correct model. The drop-out magnitude boundary parameter scaOffset is set in here.

Usage

```
evalDropoutModel(vecPiModel, vecPiPredictors)
```

Arguments

vecPiModel	(numeric vector length linear model) Linear model for drop-out rate in logit space.
vecPiPredictors	(vector length of predictors) Predictors of the drop-out rate in the linear model. Minimum are a constant offset and log of the negative binomial mean parameter. Other gene-specific predictors can be added.

Value

scaDropoutRate (scalar) Drop-out rate estimate.

Author(s)

David Sebastian Fischer

See Also

evalDropoutModel_comp

evalDropoutModel_comp *Compiled function: evalDropoutModel*

Description

Pre-compile heavily used functions. Refer to evalDropoutModel.

Usage

```
evalDropoutModel_comp(vecPiModel, vecPiPredictors)
```

Arguments

vecPiModel	(numeric vector length linear model) Linear model for drop-out rate in logit space.
vecPiPredictors	(vector length of predictors) Predictors of the drop-out rate in the linear model. Minimum are a constant offset and log of the negative binomial mean parameter. Other gene-specific predictors can be added.

Value

scaDropoutRate (scalar) Drop-out rate estimate.

Author(s)

David Sebastian Fischer

See Also

evalDropoutModel_comp

evalImpulseModel *Compute value of impulse function given parameters.*

Description

Compute value of impulse function given parameters. Enforces lower bound on value of function to avoid numerical errors during model fitting.

Usage

```
evalImpulseModel(vecImpulseParam, vecTimepoints)
```

Arguments

vecImpulseParam	(numeric vector number of impulse model parameters 7) {beta1, beta2, h0, h1, h2, t1, t2} Vector of impulse model parameters.
vecTimepoints	(numeric vector length number of time points) Time points to be evaluated.

Value

vecImpulseValue (vec number of vecTimepoints) Model values for given time points.

Author(s)

David Sebastian Fischer

See Also

Compiled version: evalImpulseModel_comp

`evalImpulseModel_comp` *Compiled function: evalImpulseModel*

Description

Pre-compile heavily used functions. Refer to evalImpulseModel.

Usage

```
evalImpulseModel_comp(vecImpulseParam, vecTimepoints)
```

Arguments

`vecImpulseParam`	(numeric vector number of impulse model parameters) {beta1, beta2, h0, h1, h2, t1, t2} Vector of impulse model parameters.
`vecTimepoints`	(numeric vector length number of time points) Time points to be evaluated.

Value

vecImpulseValue (vec number of vecTimepoints) Model values for given time points.

Author(s)

David Sebastian Fischer

`evalLogLikGene` *Wrapper for log likelihood of (zero-inflated) negative binomial model for a vector of counts.*

Description

Chooses whether to evaluate ZINB or NB model based on input.

Usage

```
evalLogLikGene(vecCounts, vecMu, vecNormConst, vecDisp, vecPi, vecidxNotZero,
  vecidxZero)
```

Arguments

vecCounts	(count vector number of cells) Observed read counts, not observed are NA.
vecMu	(numeric vector number of cells) Negative binomial mean parameter.
vecNormConst	(numeric vector number of cells) Model scaling factors, one per cell.
vecDisp	(numeric vector number of cells) Negative binomial dispersion parameters.
vecPi	(probability vector number of cells) Drop-out rate estimates. Set to NULL to evaluate NB likelihood.
vecidxNotZero	(bool vector number of cells) Whether observation is larger than zero.
vecidxZero	(bool vector number of cells) Whether observation is zero.

Value

scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

---

evalLogLikGeneMM *Wrapper for log likelihood of zero-inflated negative binomial model for a vector of counts.*

Description

NOT YET SUPPORTED. LINEAGEPULSE CODE WILL BE EXTENDED BY MODULAR FUNCTIONALITIES AND THIS IS ONE INSTANCE OF A PLACEHOLDER USED FOR DEVELOPING. Chooses whether to evaluate ZINB or NB model based on input.

Usage

```
evalLogLikGeneMM(vecCounts, matMuParam, vecNormConst, vecDisp, vecPi,
  vecidxNotZero, vecidxZero, scaNCells)
```

Arguments

vecCounts	(count vector number of cells) Observed read counts, not observed are NA.
matMuParam	(numeric matrix number of cells x number of mixtures) Negative binomial mean parameter matrix with one mean per cell and per mixture.
vecNormConst	(numeric vector number of cells) Model scaling factors, one per cell.
vecDisp	(numeric vector number of cells) Negative binomial dispersion parameters.
vecPi	(probability vector number of cells) Drop-out rate estimates. Set to NULL to evaluate NB model.
vecidxNotZero	(bool vector number of cells) Whether observation is larger than zero.
vecidxZero	(bool vector number of cells) Whether observation is zero.
scaNCells	(scalar) Number of cells (auxillary parameter, this is length of vecCounts).

Value

NULL This will be: scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

`evalLogLikMatrix` *Wrapper for log likelihood of (zero-inflated) negative binomial model for a matrix of counts (parallelised).*

Description

This likelihood function is a wrapper computes loglikelihood of entire data set by parallelising loglikelihood computation over genes.

Usage

```
evalLogLikMatrix(matCounts, lsMuModel, lsDispModel, lsDropModel,
  matWeights = NULL, boolConstModelOnMMLL = FALSE)
```

Arguments

`matCounts`	(count matrix genes x cells) Observed read counts, not observed are NA.
`lsMuModel`	(list) Object containing description of gene-wise mean parameter models.
`lsDispModel`	(list) Object containing description of gene-wise dispersion parameter models.
`lsDropModel`	(list) Object containing description of cell-wise drop-out parameter models. Set to NULL to evaluate NB likelihood as oppose to ZINB.
`matWeights`	(numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").
`boolConstModelOnMMLL`	(bool) [Default FALSE] Whether to evaluate constant gene-wise mean model on mixture model likelihood. This is necessary for numeric stability when testing whether mixture model fits are worse than constant fits.

Value

vecLogLik (vector length number of genes) Loglikelihood of each gene under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

See Also

Called directly by `fitZINB` to track convergence of estimation iteration on entire data set.

`evalLogLikMuDispGeneFit` *Cost function (zero-inflated) negative binomial model for mean and dispersion model fitting*

Description

Log likelihood of (zero inflated) negative binomial model. This function is designed to allow numerical optimisation of mean and dispersion paramater model on single gene given the drop-out model.

Usage

```
evalLogLikMuDispGeneFit(vecTheta, vecCounts, lsMuModelGlobal, lsDispModelGlobal,
  vecTimepoints, vecindTimepointAssign, matDropoutLinModel,
  vecPiConstPredictors, lsDropModelGlobal, vecPiParam = NULL, vecidxNotZero,
  vecidxZero)
```

Arguments

`vecTheta`	(numeric vector dispersion (1) and impulse parameters (6)) Dispersion and mean parameter estimates.
`vecCounts`	(count vector number of cells) Observed read counts, not observed are NA.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`lsDispModelGlobal`	(list) Object containing meta-data of gene-wise dispersion parameter models.
`vecTimepoints`	(numerical vector number of unique time coordinates) Unique (pseudo)time coordinates of cells.
`vecindTimepointAssign`	(numeric vector number samples) Index of time point assigned to cell in list of sorted time points.
`matDropoutLinModel`	(matrix number of cells x number of predictors) Logistic linear model parameters of the dropout rate as a function of the mean and constant gene-wise coefficients.
`vecPiConstPredictors`	(numeric vector constant gene-wise coefficients) Constant gene-wise coeffiecients, i.e. predictors which are not the offset and not the mean parameter.
`lsDropModelGlobal`	(list) Object containing meta-data of cell-wise drop-out parameter models.
`vecPiParam`	(numeric vector number of observations) Pre-evaluated drop-out model if model is not a function on the mean parameter to be fit.
`vecidxNotZero`	(index vector vector number of cells) Observation which are larger than zero.
`vecidxZero`	(index vector number of cells) Observation which are zero.

Value

scaLogLik (scalar) Value of cost function (likelihood) for given gene.

Author(s)

David Sebastian Fischer

`evalLogLikMuDispGeneFit_comp`
Compiled function: evalLogLikMuDispGeneFit

Description

Pre-compile heavily used functions. Refer to evalLogLikMuDispGeneFit.

Usage

```
evalLogLikMuDispGeneFit_comp(vecTheta, vecCounts, lsMuModelGlobal,
  lsDispModelGlobal, vecTimepoints, vecindTimepointAssign, matDropoutLinModel,
  vecPiConstPredictors, lsDropModelGlobal, vecPiParam = NULL, vecidxNotZero,
  vecidxZero)
```

Arguments

`vecTheta`	(numeric vector dispersion (1) and impulse parameters (6)) Dispersion and mean parameter estimates.
`vecCounts`	(count vector number of cells) Observed read counts, not observed are NA.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`lsDispModelGlobal`	(list) Object containing meta-data of gene-wise dispersion parameter models.
`vecTimepoints`	(numerical vector number of unique time coordinates) Unique (pseudo)time coordinates of cells.
`vecindTimepointAssign`	(numeric vector number samples) Index of time point assigned to cell in list of sorted time points.
`matDropoutLinModel`	(matrix number of cells x number of predictors) Logistic linear model parameters of the dropout rate as a function of the mean and constant gene-wise coefficients.
`vecPiConstPredictors`	(numeric vector constant gene-wise coefficients) Constant gene-wise coeffiecients, i.e. predictors which are not the offset and not the mean parameter.
`lsDropModelGlobal`	(list) Object containing meta-data of cell-wise drop-out parameter models.
`vecPiParam`	(numeric vector number of observations) Pre-evaluated drop-out model if model is not a function on the mean parameter to be fit.
`vecidxNotZero`	(index vector vector number of cells) Observation which are larger than zero.
`vecidxZero`	(index vector number of cells) Observation which are zero.

Value

scaLogLik (scalar) Value of cost function (likelihood) for given gene.

Author(s)

David Sebastian Fischer

See Also

evalLogLikMuDispGeneFit

`evalLogLikNB` *Compute loglikelihood of negative binomial model for a vector of counts.*

Description

This liklihood function is appropriate for sequencing data without drop outs. This is the core function used for every likelihood evaluation in LineagePulse, such as maximum likelihood-based estimation. It operates on a vector of counts, such as observations of a gene. Note that for the sake of numerical stability, lower bounds on loglikelihood terms are implemented.

Usage

```
evalLogLikNB(vecCounts, vecMu, vecDisp, vecidxNotZero, vecidxZero)
```

Arguments

`vecCounts`	(count vector number of samples) Observed read counts, not observed are NA.
`vecMu`	(vector number of samples) Negative binomial mean parameter.
`vecDisp`	(scalar vector number of samples) Negative binomial dispersion parameters.
`vecidxNotZero`	(bool vector number of samples) Whether observation is larger than zero.
`vecidxZero`	(bool vector number of samples) Whether observation is zero.

Value

scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

evalLogLikNB_comp *Compiled function: evalLogLikNB*

Description

Pre-compile heavily used functions. Refer to evalLogLikNB

Usage

```
evalLogLikNB_comp(vecCounts, vecMu, vecDisp, vecidxNotZero, vecidxZero)
```

Arguments

vecCounts	(count vector number of samples) Observed read counts, not observed are NA.
vecMu	(vector number of samples) Negative binomial mean parameter.
vecDisp	(scalar vector number of samples) Negative binomial dispersion parameters.
vecidxNotZero	(bool vector number of samples) Whether observation is larger than zero.
vecidxZero	(bool vector number of samples) Whether observation is zero.

Value

scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

See Also

evalLogLikNB

evalLogLikPiZINB_ManyCells *Cost function zero-inflated negative binomial model for drop-out fitting for many cells*

Description

Log likelihood of zero inflated negative binomial model. This function is designed to allow numerical optimisation of logistic drop-out paramater model on multiple cells given the negative binomial mean and dispersion parameters. This function is optimised for memory usage vs evalLogLikPiZINB_SingleCell at the cost of computation time: The parameter models are not kept as a gene x cell matrix but as the raw model objects which are evaluated within the cost function.

Usage

```
evalLogLikPiZINB_ManyCells(vecTheta, matCounts, lsMuModel, lsDispModel,
  lsDropModel)
```

Arguments

vecTheta	(numeric vector length linear model) Parameter estimates for logit linear model for drop-out rate.
matCounts	(count matrix genes x cells) Observed read counts, not observed are NA.
lsMuModel	(list) Object containing description of gene-wise mean parameter models.
lsDispModel	(list) Object containing description of gene-wise dispersion parameter models.
lsDropModel	(list) Object containing description of cell-wise drop-out parameter models.

Value

scaLogLik (scalar) Value of cost function: zero-inflated negative binomial likelihood.

Author(s)

David Sebastian Fischer

See Also

Called by fitting wrapper: `fitPi_ManyCells`. Calls `evalLogLikMatrix`. Compiled version: evalLogLikPiZINB_ManyCells_comp.

---

evalLogLikPiZINB_ManyCells_comp

Cost function zero-inflated negative binomial model for drop-out fitting

---

Description

Refer to evalLogLikPiZINB_ManyCells.

Usage

```
evalLogLikPiZINB_ManyCells_comp(vecTheta, matCounts, lsMuModel, lsDispModel,
  lsDropModel)
```

Arguments

vecTheta	(numeric vector length linear model) Parameter estimates for logit linear model for drop-out rate.
matCounts	(count matrix genes x cells) Observed read counts, not observed are NA.
lsMuModel	(list) Object containing description of gene-wise mean parameter models.
lsDispModel	(list) Object containing description of gene-wise dispersion parameter models.
lsDropModel	(list) Object containing description of cell-wise drop-out parameter models.

Value

scaLogLik (scalar) Value of cost function: zero-inflated negative binomial likelihood.

Author(s)

David Sebastian Fischer

See Also

Compiled version of evalLogLikPiZINB_ManyCells

`evalLogLikPiZINB_SingleCell` *Cost function zero-inflated negative binomial model for drop-out fitting*

Description

Log likelihood of zero inflated negative binomial model. This function is designed to allow numerical optimisation of logistic drop-out paramater model on single gene given the negative binomial mean and dispersion parameters.

Usage

```
evalLogLikPiZINB_SingleCell(vecTheta, vecCounts, vecMu, scaNormConst, vecDisp,
  matPiAllPredictors, strDropModel, vecidxNotZero, vecidxZero)
```

Arguments

`vecTheta`	(numeric vector length linear model) Parameter estimates for logit linear model for drop-out rate.
`vecCounts`	(count vector number of genes) Observed read counts, not observed are NA.
`vecMu`	(vector number of cells) Negative binomial mean parameter estimate.
`scaNormConst`	(scalar) Model scaling factors, one per cell.
`vecDisp`	(vector number of cells) Negative binomial dispersion parameter estimate.
`matPiAllPredictors`	(matrix genes x predictors) Predictors of the drop-out rate in the linear model. Minimum are a constant offset and log of the negative binomial mean parameter. Other gene-specific predictors can be added.
`strDropModel`	(str) "logistic_ofMu", "logistic" [Default "logistic_ofMu"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors.
`vecidxNotZero`	(bool vector number of cells) Whether observation is larger than zero.
`vecidxZero`	(bool vector number of cells) Whether observation is zero.

Value

scaLogLik (scalar) Value of cost function: zero-inflated negative binomial likelihood.

Author(s)

David Sebastian Fischer

See Also

Called by fitting wrapper: `fitPi_SingleCell`. Calls `evalLogLikZINB`. Compiled version: evalLogLikPiZINB_SingleCell_comp.

`evalLogLikPiZINB_SingleCell_comp` *Cost function zero-inflated negative binomial model for drop-out fitting*

Description

Refer to evalLogLikPiZINB_SingleCell().

Usage

```
evalLogLikPiZINB_SingleCell_comp(vecTheta, vecCounts, vecMu, scaNormConst,
  vecDisp, matPiAllPredictors, strDropModel, vecidxNotZero, vecidxZero)
```

Arguments

`vecTheta`	(numeric vector length linear model) Parameter estimates for logit linear model for drop-out rate.
`vecCounts`	(count vector number of genes) Observed read counts, not observed are NA.
`vecMu`	(vector number of cells) Negative binomial mean parameter estimate.
`scaNormConst`	(scalar) Model scaling factors, one per cell.
`vecDisp`	(vector number of cells) Negative binomial dispersion parameter estimate.
`matPiAllPredictors`	(matrix genes x predictors) Predictors of the drop-out rate in the linear model. Minimum are a constant offset and log of the negative binomial mean parameter. Other gene-specific predictors can be added.
`strDropModel`	(str) "logistic_ofMu", "logistic" [Default "logistic_ofMu"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors.
`vecidxNotZero`	(bool vector number of cells) Whether observation is larger than zero.
`vecidxZero`	(bool vector number of cells) Whether observation is zero.

Value

scaLogLik (scalar) Value of cost function: zero-inflated negative binomial likelihood.

Author(s)

David Sebastian Fischer

See Also

Compiled version of `evalLogLikPiZINB_SingleCell`.

`evalLogLikZINB` *Compute loglikelihood of zero-inflated negative binomial model for a vector of counts.*

Description

This liklihood function is appropriate for sequencing data with high drop out rate, commonly observed in single cell data (e.g. scRNA-seq). This is the core function used for every likelihood evaluation in LineagePulse, such as maximum likelihood-based estimation. It operates on a vector of counts, such as observations of a gene. Note that for the sake of numerical stability, lower bounds on loglikelihood terms are implemented.

Usage

```
evalLogLikZINB(vecCounts, vecMu, vecDisp, vecPi, vecidxNotZero, vecidxZero)
```

Arguments

`vecCounts`	(count vector number of samples) Observed read counts, not observed are NA.
`vecMu`	(vector number of samples) Negative binomial mean parameter.
`vecDisp`	(scalar vector number of samples) Negative binomial dispersion parameters.
`vecPi`	(probability vector number of samples) Drop-out rate estimates.
`vecidxNotZero`	(bool vector number of samples) Whether observation is larger than zero.
`vecidxZero`	(bool vector number of samples) Whether observation is zero.

Value

scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

`evalLogLikZINB_comp` *Compiled function: evalLogLikZINB*

Description

Pre-compile heavily used functions. Refer to evalLogLikZINB

Usage

```
evalLogLikZINB_comp(vecCounts, vecMu, vecDisp, vecPi, vecidxNotZero, vecidxZero)
```

Arguments

vecCounts	(count vector number of samples) Observed read counts, not observed are NA.
vecMu	(vector number of samples) Negative binomial mean parameter.
vecDisp	(scalar vector number of samples) Negative binomial dispersion parameters.
vecPi	(probability vector number of samples) Drop-out rate estimates.
vecidxNotZero	(bool vector number of samples) Whether observation is larger than zero.
vecidxZero	(bool vector number of samples) Whether observation is zero.

Value

scaLogLik (scalar) Likelihood under zero-inflated negative binomial model.

Author(s)

David Sebastian Fischer

See Also

evalLogLikZINB

---

fitLPModels *Fit all models necessary for LineagePulse*

Description

Fit alternative H1 and null H0 on both ZINB and NB noise model to a data set using cycles of coordinate ascent. The algorithm first fits the either H1 or H0 together with the dropout model by iterating over cell-wise (dropout models) and gene-wise (negative binomial models) parameters. Subsequently, the remaining model (H0 or H1) is estimated by iterating over zero-inflated negative binomial mean and dispersion parameter estimation condition on the previously estimated logistic drop-out model. The NB noise model based models are estimated in parallel across genes.

Usage

```
fitLPModels(objLP, matPiConstPredictors, strMuModel = "constant",
  strDispModelFull = "constant", strDispModelRed = "constant",
  strDropModel = "logistic_ofMu", strDropFitGroup = "PerCell",
  boolEstimateNoiseBasedOnH0 = TRUE, scaMaxEstimationCycles = 20,
  boolVerbose = FALSE, boolSuperVerbose = FALSE)
```

Arguments

objLP	(LineagePulseObject) LineagePulseObject to which null and alternative model are to be fitted.
matPiConstPredictors	(numeric matrix genes x number of constant gene-wise drop-out predictors) Predictors for logistic drop-out fit other than offset and mean parameter (i.e. parameters which are constant for all observations in a gene and externally supplied.) Is null if no constant predictors are supplied.

strMuModel (str) "constant", "groups", "MM", "splines","impulse" [Default "impulse"] Model according to which the mean parameter is fit to each gene in the alternative model (H1).

strDispModelFull
(str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene in the alternative model (H1).

strDispModelRed
(str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene in the null model (H0).

strDropModel (str) "logistic_ofMu", "logistic" [Default "logistic_ofMu"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors.

strDropFitGroup
(str) "PerCell", "AllCells" [Defaul "PerCell"] Definition of groups on cells on which separate drop-out model parameterisations are fit. "PerCell" - one parametersiation (fit) per cell "ForAllCells" - one parametersiation (fit) for all cells

boolEstimateNoiseBasedOnH0
(bool) [Default FALSE] Whether to co-estimate logistic drop-out model with the constant null model or with the alternative model. The co-estimation with the noise model typically extends the run-time of this model-estimation step strongly. While the drop-out model is more accurate if estimated based on a more realistic model expression model (the alternative model), a trade-off for speed over accuracy can be taken and the dropout model can be chosen to be estimated based on the constant null expression model (set to TRUE).

scaMaxEstimationCycles
(integer) [Default 20] Maximum number of estimation cycles performed in fitZINB(). One cycle contain one estimation of of each parameter of the zero-inflated negative binomial model as coordinate ascent.

boolVerbose (bool) Whether to follow convergence of the iterative parameter estimation with one report per cycle.

boolSuperVerbose
(bool) Whether to follow convergence of the iterative parameter estimation in high detail with local convergence flags and step-by-step loglikelihood computation.

Value

objLP (LineagePulseObject) LineagePulseObject with models with and fitting reporters added.

Author(s)

David Sebastian Fischer

See Also

Called by `runLineagePulse`. Calls model estimation wrappers: `fitContinuousModels`.

`fitModel` *Fit (zero-inflated) negative binomial model to data*

Description

This function fits a ZINB or NB model with variable input. It is the wrapper for the individual fits of the full and alternative models in LineagePulse.

Usage

```
fitModel(matCounts, dfAnnotation, vecConfoundersMu = NULL,
  vecConfoundersDisp = NULL, vecNormConst, scaDFSplinesMu = NULL,
  scaDFSplinesDisp = NULL, matWeights = NULL, matPiConstPredictors = NULL,
  lsDropModel = NULL, matMuModelInit = NULL, lsmatBatchModelInitMu = NULL,
  matDispModelInit = NULL, lsmatBatchModelInitDisp = NULL, strMuModel,
  strDispModel, strDropModel = "logistic_ofMu", strDropFitGroup = "PerCell",
  scaMaxEstimationCycles = 20, boolVerbose = TRUE,
  boolSuperVerbose = TRUE)
```

Arguments

`matCounts`	(matrix genes x cells) Count data of all cells, unobserved entries are NA.
`dfAnnotation`	(data frame cells x meta characteristics) Annotation table which contains meta data on cells.
`vecConfoundersMu`	(vector of strings number of confounders on mean) [Default NULL] Confounders to correct for in mu batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
`vecConfoundersDisp`	(vector of strings number of confounders on dispersion) [Default NULL] Confounders to correct for in dispersion batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
`vecNormConst`	(numeric vector number of cells) Model scaling factors, one per cell. These factors linearly scale the mean model for evaluation of the loglikelihood.
`scaDFSplinesMu`	(sca) [Default NULL] If strMuModel=="splines", the degrees of freedom of the natural cubic spline to be used as a mean parameter model.
`scaDFSplinesDisp`	(sca) [Default NULL] If strDispModelFull=="splines" or strDispModelRed=="splines", the degrees of freedom of the natural cubic spline to be used as a dispersion parameter model.
`matWeights`	(numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").
`matPiConstPredictors`	(numeric matrix genes x number of constant gene-wise drop-out predictors) [Default NULL] Predictors for logistic drop-out fit other than offset and mean parameter (i.e. parameters which are constant for all observations in a gene and externally supplied.) Is null if no constant predictors are supplied.
`lsDropModel`	(list) [Default NULL] Object containing description of cell-wise drop-out parameter models.

matMuModelInit (numeric matrix genes x mu model parameters) [Default NULL] Contains initialisation of mean model parameters according to the used model.

lsmatBatchModelInitMu
(list) [Default NULL] Initialisation of batch correction models for mean parameter.

matDispModelInit
(numeric matrix genes x disp model parameters) [Default NULL] Contains initialisation of dispersion model parameters according to the used model.

lsmatBatchModelInitDisp
(list) [Default NULL] Initialisation of batch correction models for dispersion parameter.

strMuModel (str) "constant", "groups", "MM", "splines","impulse" [Default "impulse"] Model according to which the mean parameter is fit to each gene as a function of population structure in the alternative model (H1).

strDispModel (str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the given model.

strDropModel (str) "logistic_ofMu", "logistic", "none" [Default "logistic_ofMu"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors. "none" - negative binomial noise model without zero-inflation.

strDropFitGroup
(str) "PerCell", "AllCells" [Defaul "PerCell"] Definition of groups on cells on which separate drop-out model parameterisations are fit. "PerCell" - one parametersiation (fit) per cell "ForAllCells" - one parametersiation (fit) for all cells

scaMaxEstimationCycles
(integer) [Default 20] Maximum number of estimation cycles performed in fitZINB(). One cycle contain one estimation of of each parameter of the zero-inflated negative binomial model as coordinate ascent.

boolVerbose (bool) [Default TRUE] Whether to follow convergence of the iterative parameter estimation with one report per cycle.

boolSuperVerbose
(bool) [Default TRUE] Whether to follow convergence of the iterative parameter estimation in high detail with local convergence flags and step-by-step loglikelihood computation.

Details

For ZINB models with drop-out model estimation: The estimation is iterative coordinate ascent over gene-wise and cell-wise model if the drop-out model is not set a priori. If the drop-out model is given, the estimation is a single M-like step of the iterative coordinate ascent.

Convergence of iterative coordinate ascent is tracked with the the loglikelihood of the entire data matrix. Every step is a maximum likelihood estimation of the target parameters conditioned on the remaining parameter estimates. Therefore, convergence to a local optimum is guaranteed if the algorithm is run until convergence. Parallelisation of each estimation step is implemented where conditional independences of parameter estimations allow so.

Convergence can be followed with verbose=TRUE (at each iteration) or at each step (boolSuperVerbose=TRUE).

To save memory, not the entire parameter matrix (genes x cells) but the parmater models are stored in the objects lsMuModel, lsDispModel and lsDropModel. In short, these object contain the gene/cell-wise parameters of the model used to constrain the parameter in question and the predictors necessary to evaluate the parameter model to receive the observation-wise paramter values.

Value

list

- lsMuModel (list) Object containing description of gene-wise mean parameter models.
- lsDispModel (list) Object containing description of gene-wise dispersion parameter models.
- lsDropModel (list) Object containing description of cell-wise drop-out parameter models.
- matWeights (numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").
- boolConvergenceModel: (bool) Convergence status of model estimation.
- vecEMLogLikModel: (numeric vector number of genes) Likelihood of model fits by iterative coordinate ascent iteration.
- strReport: (str) Log of model estimation to be added to overall log.

Author(s)

David Sebastian Fischer

See Also

Called by `fitContinuousModels`. Calls parameter estimation wrappers: `fitPiZINB`, `fitZINBMuDisp`. Calls `evalLogLikMatrix` to follow convergence.

`fitMuDisp` *Coordinate mean and dispersion parameter co-estimation step*

Description

Auxillary function that calls the estimation functions for the different mean and dispersion models according to their needs. Note that one function has to be coded for each combination of mean and dispersion models.

Usage

```
fitMuDisp(matCountsProc, vecNormConst, lsMuModel, lsDispModel, lsDropModel,
  matWeights)
```

Arguments

`matCountsProc`	(matrix genes x cells) Observed read counts, not observed are NA.
`vecNormConst`	(numeric vector number of cells) Model scaling factors, one per cell.
`lsMuModel`	(list) Object containing description of gene-wise mean parameter models.
`lsDispModel`	(list) Object containing description of gene-wise dispersion parameter models.
`lsDropModel`	(list) Object containing description of cell-wise drop-out parameter models.
`matWeights`	(numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").

Value

list

- matMuModel (numeric matrix genes x mu model parameters) Contains the mean model parameters according to the used model.
- lsmatBatchModelMu (list) Fit of batch correction models for mean parameter.
- matDispModel (numeric matrix genes x disp model parameters) Contains the dispersion model parameters according to the used model.
- lsmatBatchModelDisp (list) Fit of batch correction models for dispersion parameter.
- vecConvergence (numeric vector number of genes) Convergence status of optim for each gene.
- vecLL (numeric vector number of genes) Likelihood of model fit.

Author(s)

David Sebastian Fischer

See Also

Called by `fitModel`. Calls fitting wrappers: `fitImpulseZINB`, `fitContinuousZINB`.

`fitMuDispGene` *Optim wrapper for gene-wise models other than mixture model.*

Description

Given a parameter initialisation, this function performs numerical optimisation using BFGS of the likelihood function given the supplied mean and dispersion model. This is the wrapper that calls optim.

Usage

```
fitMuDispGene(vecCounts, vecMuModelGuess, lsvecBatchParamGuessMu,
  lsMuModelGlobal, vecDispGuess, lsvecBatchParamGuessDisp, lsDispModelGlobal,
  matDropoutLinModel, vecPiConstPredictors, lsDropModelGlobal, vecPiParam)
```

Arguments

`vecCounts`	(count vector number of cells) Observed read counts, not observed are NA.
`vecMuModelGuess`	(numeric vector number of mean model parameters) Initialisation for impulse model.
`lsvecBatchParamGuessMu`	(list) Object containing initialisation for mean parameter batch correction model.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`vecDispGuess`	(numeric vector number of dispersion model parameters) Initialisation for dispersion model.

`lsvecBatchParamGuessDisp`
: (list) Object containing initialisation for dispersion parameter batch correction model.

`lsDispModelGlobal`
: (list) Object containing meta-data of gene-wise dispersion parameter models.

`matDropoutLinModel`
: (matrix number of cells x number of predictors) Logistic linear model parameters of the dropout rate as a function of the mean and constant gene-wise coefficients.

`vecPiConstPredictors`
: (numeric vector constant gene-wise coefficients) Constant gene-wise coeffiecients, i.e. predictors which are not the offset and not the mean parameter.

`lsDropModelGlobal`
: (list) Object containing meta-data of cell-wise drop-out parameter models.

`vecPiParam`
: (numeric vector number of observations) Pre-evaluated drop-out model if model is not a function on the mean parameter to be fit.

Details

This function performs error handling of the numerical fitting procedure. This function corrects for the likelihood sensitivity bounds used in the cost function.

Value

list

- vecMuModel (numeric vector number of mu model parameters) Contains the mean model parameters according to the used model.
- lsvecBatchModelMu (list) Fit of batch correction models for mean parameter to given gene.
- vecDispModel (numeric vector number of dispersion model parameters) Contains the dispersion model parameters according to the used model.
- lsvecBatchModelDisp (list) Fit of batch correction models for dispersion parameter to given gene.
- scaConvergence (numeric vector number of genes) Convergence status of optim for given gene.
- scaLL (numeric vector number of genes) Likelihood of model fit for given gene.

Author(s)

David Sebastian Fischer

See Also

Called once for each gene by `fitZINBMuDisp` or within wrapper `fitZINBImpulse` once for each initalisation of each gene. Calls fitting likelihood functions: `evalLogLikContinuousZINB_comp`.

`fitMuDispGeneImpulse` *Multiple initilalisation wrapper for impulse mean model*

Description

Multiple initialisation are tried for the impulse model. Therefore, this wrapper sits ontop of fitContinuousZINB() in the fitting hierarchy and wraps multiple initialisations at the level of one gene.

Usage

```
fitMuDispGeneImpulse(vecCounts, vecImpulseParamGuess, lsvecBatchParamGuessMu,
  lsMuModelGlobal, vecDispGuess, lsvecBatchParamGuessDisp, lsDispModelGlobal,
  matDropoutLinModel, vecPiConstPredictors, lsDropModelGlobal, vecPiParam)
```

Arguments

`vecCounts`	(count vector number of cells) Observed read counts, not observed are NA.
`vecImpulseParamGuess`	(numeric vector number of impulse model parameters) Initialisation for impulse model.
`lsvecBatchParamGuessMu`	(list) Object containing initialisation for mean parameter batch correction model.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`vecDispGuess`	(numeric vector number of dispersion model parameters) Initialisation for dispersion model.
`lsvecBatchParamGuessDisp`	(list) Object containing initialisation for dispersion parameter batch correction model.
`lsDispModelGlobal`	(list) Object containing meta-data of gene-wise dispersion parameter models.
`matDropoutLinModel`	(matrix number of cells x number of predictors) Logistic linear model parameters of the dropout rate as a function of the mean and constant gene-wise coefficients.
`vecPiConstPredictors`	(numeric vector constant gene-wise coefficients) Constant gene-wise coeffiecients, i.e. predictors which are not the offset and not the mean parameter.
`lsDropModelGlobal`	(list) Object containing meta-data of cell-wise drop-out parameter models.
`vecPiParam`	(numeric vector number of observations) Pre-evaluated drop-out model if model is not a function on the mean parameter to be fit.

Value

list

- vecMuModel (numeric vector number of mu model parameters) Contains the mean model parameters according to the used model.

- lsvecBatchModelMu (list) Fit of batch correction models for mean parameter to given gene.
- vecDispModel (numeric vector number of dispersion model parameters) Contains the dispersion model parameters according to the used model.
- lsvecBatchModelDisp (list) Fit of batch correction models for dispersion parameter to given gene.
- scaConvergence (numeric vector number of genes) Convergence status of optim for given gene.
- scaLL (numeric vector number of genes) Likelihood of model fit for given gene.

Author(s)

David Sebastian Fischer

See Also

Called by mean-dispersion co-estimation wrapper `fitZINBMuDisp`. Calls optimisation wrapper `fitContinuousZINB` for each initialisation.

`fitMuDispGeneMM` *Optim wrapper for gene-wise models other than mixture model.*

Description

NOT YET SUPPORTED. LINEAGEPULSE CODE WILL BE EXTENDED BY MODULAR FUNCTIONALITIES AND THIS IS ONE INSTANCE OF A PLACEHOLDER USED FOR DEVELOPING. Given a parameter initialisation, this function performs numerical optimisation using BFGS of the likelihood function given the supplied mean and dispersion model. This is the wrapper that calls optim.

Usage

```
fitMuDispGeneMM(vecCounts, vecMuGuess, lsvecBatchParamGuessMu, lsMuModelGlobal,
  vecDispGuess, lsvecBatchParamGuessDisp, lsDispModelGlobal, matDropoutLinModel,
  vecPiConstPredictors, lsDropModelGlobal, vecPiParam, matWeights, MAXIT,
  RELTOL)
```

Arguments

`vecCounts`	(count vector number of cells) Observed read counts, not observed are NA.
`vecMuGuess`	(numeric vector number of mean model parameters) Initialisation for impulse model.
`lsvecBatchParamGuessMu`	(list) Object containing initialisation for mean parameter batch correction model.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.
`vecDispGuess`	(numeric vector number of dispersion model parameters) Initialisation for dispersion model.
`lsvecBatchParamGuessDisp`	(list) Object containing initialisation for dispersion parameter batch correction model.

lsDispModelGlobal
: (list) Object containing meta-data of gene-wise dispersion parameter models.

matDropoutLinModel
: (matrix number of cells x number of predictors) Logistic linear model parameters of the dropout rate as a function of the mean and constant gene-wise coefficients.

vecPiConstPredictors
: (numeric vector constant gene-wise coefficients) Constant gene-wise coeffiecients, i.e. predictors which are not the offset and not the mean parameter.

lsDropModelGlobal
: (list) Object containing meta-data of cell-wise drop-out parameter models.

vecPiParam
: (numeric vector number of observations) Pre-evaluated drop-out model if model is not a function on the mean parameter to be fit.

matWeights
: (numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").

MAXIT
: (scalar) Maximum number of BFGS iterations handed to optim().

RELTOL
: (scalar) Cost function convergence criterium handed to optim().

Details

This function performs error handling of the numerical fitting procedure. This function corrects for the likelihood sensitivity bounds used in the cost function.

Value

list

- vecMuModel (numeric vector number of mu model parameters) Contains the mean model parameters according to the used model.
- lsvecBatchModelMu (list) Fit of batch correction models for mean parameter to given gene.
- vecDispModel (numeric vector number of dispersion model parameters) Contains the dispersion model parameters according to the used model.
- lsvecBatchModelDisp (list) Fit of batch correction models for dispersion parameter to given gene.
- scaConvergence (numeric vector number of genes) Convergence status of optim for given gene.
- scaLL (numeric vector number of genes) Likelihood of model fit for given gene.

Author(s)

David Sebastian Fischer

fitPi — *Global wrapper for fitting of all drop-out models*

Description

This function fits all drop-out models required on this data set.

Usage

```
fitPi(matCounts, lsMuModel, lsDispModel, lsDropModel)
```

Arguments

matCounts	(count matrix genes x cells) Observed read counts, not observed are NA.
lsMuModel	(list) Object containing description of gene-wise mean parameter models.
lsDispModel	(list) Object containing description of gene-wise dispersion parameter models.
lsDropModel	(list) Object containing description of cell-wise drop-out parameter models.

Value

vecLinModel (numeric vector length linear model) Linear model for drop-out rate in logit space for given cell.

Author(s)

David Sebastian Fischer

See Also

Called by ZINB fitting wrapper `fitModel`. Calls optim wrapper `fitPi_ManyCells` or `fitPi_SingleCell`.

fitPi_ManyCells — *Optim wrapper for drop-out model fitting on many cells*

Description

This function fits a logistic drop-out model to a set of cells based on given gene-specific predictors (which enter the linear model). Parameter estimation of the linear model is performed by maximum likelihood based on the overall likelihood.

Usage

```
fitPi_ManyCells(vecParamGuess, matCounts, lsMuModel, lsDispModel, lsDropModel)
```

Arguments

vecParamGuess	(numeric vector length linear model) Initial parameter estimates for logit linear model for drop-out rate.
matCounts	(count matrix genes x cells) Observed read counts, not observed are NA.
lsMuModel	(list) Object containing description of gene-wise mean parameter models.
lsDispModel	(list) Object containing description of gene-wise dispersion parameter models.
lsDropModel	(list) Object containing description of cell-wise drop-out parameter models.

Value

vecLinModel (numeric vector length linear model) Linear model for drop-out rate in logit space for given cell.

Author(s)

David Sebastian Fischer

See Also

Called by drop-out estimation wrapper code in `fitPi`. Calls fitting cost function: `evalLogLikPiZINB_ManyCells_comp`.

---

fitPi_SingleCell *Optim wrapper for drop-out model fitting on single cell*

Description

This function fits a logistic drop-out model to a cell based on given gene-specific predictors (which enter the linear model). Parameter estimation of the linear model is performed by maximum likelihood based on the overall likelihood.

Usage

```
fitPi_SingleCell(vecParamGuess, vecCounts, vecMuParam, scaNormConst,
  vecDispParam, matPiAllPredictors, lsDropModelGlobal)
```

Arguments

vecParamGuess	(numeric vector length linear model) Parameter estimates for logit linear model for drop-out rate.
vecCounts	(count vector number of genes) Observed read counts, not observed are NA.
vecMuParam	(vector number of cells) Negative binomial mean parameter estimate.
scaNormConst	(scalar) Model scaling factors, one per cell.
vecDispParam	(vector number of cells) Negative binomial dispersion parameter estimate.
matPiAllPredictors	(matrix genes x predictors) Predictors of the drop-out rate in the linear model. Minimum are a constant offset and log of the negative binomial mean parameter. Other gene-specific predictors can be added.
lsDropModelGlobal	(list) Object containing meta-data of cell-wise drop-out parameter models.

Value

vecLinModel (numeric vector length linear model) Linear model for drop-out rate in logit space for given cell.

Author(s)

David Sebastian Fischer

See Also

Called by drop-out estimation wrapper code in `fitPi`. Calls fitting cost function: `evalLogLikPiZINB_SingleCell_comp`

`getFitsDispersion` *Get dispersion model fits*

Description

Return dispersion model fits per gene and cell as matrix for chosen model.

Usage

```
getFitsDispersion(lsDispModel, vecGeneIDs = NULL)
```

Arguments

`lsDispModel`	(list) Object containing description of gene-wise dispersion parameter models.
`vecGeneIDs`	(vector of strings) Gene IDs for which dispersion model fits are to be extracted.

Value

(numeric matrix genes x cells) Dispersion parameter fits.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 20,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# Get dispersion parameter fits on alternative model:
```

```
# Use H1 model fits.
vecDispersionFits <- getFitsDispersion(
     lsDispModel = lsDispModelH1(objLP),
     vecGeneIDs = rownames(lsSimulatedData$counts)[1])
```

`getFitsDropout` *Get drop-out model fits*

Description

Return drop-out model fits per gene and cell as matrix for chosen models.

Usage

```
getFitsDropout(lsMuModel, lsDropModel, vecGeneIDs = NULL)
```

Arguments

`lsMuModel`	(list) Object containing description of gene-wise mean parameter models.
`lsDropModel`	(list) Object containing description of cell-wise drop-out parameter models.
`vecGeneIDs`	(vector of strings) [Default NULL] Gene IDs for which posteriors of drop-out are to be computed.

Value

(numeric matrix genes x cells) Drop-out rate fits.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 20,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# Get drop-out rate fits on alternative model:
# Use H1 model fits.
vecDropoutFits <- getFitsDropout(
     lsMuModel = lsMuModelH1(objLP),
     lsDropModel = lsDropModel(objLP),
     vecGeneIDs = rownames(lsSimulatedData$counts)[1])
```

getFitsMean *Get mean model fits*

Description

Return mean model fits per gene and cell as matrix for chosen model.

Usage

```
getFitsMean(lsMuModel, vecGeneIDs = NULL)
```

Arguments

lsMuModel	(list) Object containing description of gene-wise mean parameter models.
vecGeneIDs	(vector of strings) Gene IDs for which mean model fits are to be extracted.

Value

(numeric matrix genes x cells) Mean parameter fits.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 20,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# Get mean parameter fits on alternative model:
# Use H1 model fits.
vecMeanFits <- getFitsMean(
     lsMuModel = lsMuModelH1(objLP),
     vecGeneIDs = rownames(lsSimulatedData$counts)[1])
#plot(lsSimulatedData$annot$continuous, vecMeanFits)
```

`getNormData` *Return depth and batch corrected data*

Description

The data normalisation is based on the model normalisation used by and inferred by LineagePulse, e.g. for data visualisation.

Usage

```
getNormData(matCounts, lsMuModel, vecGeneIDs, boolDepth = TRUE,
  boolBatch = TRUE)
```

Arguments

`matCounts`	(numeric matrix genes x cells) Count data.
`lsMuModel`	(list) Mean parameter model parameters.
`vecGeneIDs`	(vector of strings) Gene IDs for which mean model fits are to be extracted.
`boolDepth`	(bool) [Default TRUE] Whether to normalize for sequencing depth.
`boolBatch`	(bool) [Default TRUE] Whether to normalize for batch.

Value

(numeric matrix genes x cells) Input data normalized by library size factors (optional) and by inferred batch factors (optional).

Author(s)

David Sebastian Fischer

See Also

Called by `fitZINB`. Can be called by user.

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 20,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# Get batch correction on alternative model:
# Use H1 model fits.
```

```
matNormData <- getNormData(
     matCounts = lsSimulatedData$counts,
     lsMuModel = lsMuModelH1(objLP),
     vecGeneIDs = rownames(lsSimulatedData$counts)[1],
     boolDepth = TRUE, boolBatch = TRUE)
```

`getPostDrop` *Get posteriors of drop-out*

Description

Return posteriors of drop-out per gene and cell as matrix for chosen models.

Usage

```
getPostDrop(matCounts, lsMuModel, lsDispModel, lsDropModel, vecGeneIDs)
```

Arguments

`matCounts`	(count matrix genes x cells) Observed read counts, not observed are NA.
`lsMuModel`	(list) Object containing description of gene-wise mean parameter models.
`lsDispModel`	(list) Object containing description of gene-wise dispersion parameter models.
`lsDropModel`	(list) Object containing description of cell-wise drop-out parameter models.
`vecGeneIDs`	(vector of strings) [Default NULL] Gene IDs for which posteriors of drop-out are to be computed.

Value

(numeric matrix genes x cells) Posterior probability of observation not being generated by drop-out.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 20,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
# Get posterior of drop-out on alternative model:
```

```
# Use H1 model fits.
vecPosteriorDropoutFits <- getPostDrop(
     matCounts = lsSimulatedData$counts,
     lsMuModel = lsMuModelH1(objLP),
     lsDispModel = lsDispModelH1(objLP),
     lsDropModel = lsDropModel(objLP),
     vecGeneIDs = rownames(lsSimulatedData$counts)[1])
```

`initDispModel` *Initialise dispersion model container object*

Description

Either use supplied fits from previous fitting or initialise from count data.

Usage

```
initDispModel(matCounts, dfAnnotation, vecConfoundersDisp, scaDFSplinesDisp,
  matWeights, matDispModelInit, lsmatBatchModelInitDisp, strDispModel,
  strMuModel, MAXIT_BFGS_MuDisp, RELTOL_BFGS_MuDisp)
```

Arguments

`matCounts`	(matrix genes x cells) Count data of all cells, unobserved entries are NA.
`dfAnnotation`	(data frame cells x meta characteristics) Annotation table which contains meta data on cells.
`vecConfoundersDisp`	(vector of strings number of confounders on dispersion) [Default NULL] Confounders to correct for in dispersion batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
`scaDFSplinesDisp`	(sca) [Default NULL] If strDispModelFull=="splines" or strDispModelRed=="splines", the degrees of freedom of the natural cubic spline to be used as a dispersion parameter model.
`matWeights`	(numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").
`matDispModelInit`	(numeric matrix genes x disp model parameters) [Default NULL] Contains initialisation of dispersion model parameters according to the used model.
`lsmatBatchModelInitDisp`	(list) [Default NULL] Initialisation of batch correction models for dispersion parameter.
`strDispModel`	(str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the given model.
`strMuModel`	(str) "constant", "groups", "MM", "splines","impulse" [Default "impulse"] Model according to which the mean parameter is fit to each gene as a function of population structure in the alternative model (H1).

MAXIT_BFGS_MuDisp
: (sca) Maximum number of iterations in BFGS estimation of Mu/Disp models. This is a control parameter to optim().

RELTOL_BFGS_MuDisp
: (sca) Relative tolerance of BFGS estimation of Mu/Disp models. This is a control parameter to optim().

Value

lsDispModel (list) Initialisation of dispersion model object.

Author(s)

David Sebastian Fischer

See Also

Called by `fitModel`.

`initDropModel` *Initialise drop-out model container object*

Description

Either use supplied fits from previous fitting or initialise from count data.

Usage

```
initDropModel(matCounts, matPiConstPredictors, lsDropModel, strDropModel,
  strDropFitGroup, MAXIT_BFGS_Pi, RELTOL_BFGS_Pi)
```

Arguments

matCounts
: (matrix genes x cells) Count data of all cells, unobserved entries are NA.

matPiConstPredictors
: (numeric matrix genes x number of constant gene-wise drop-out predictors) [Default NULL] Predictors for logistic drop-out fit other than offset and mean parameter (i.e. parameters which are constant for all observations in a gene and externally supplied.) Is null if no constant predictors are supplied.

lsDropModel
: (list) [Default NULL]

strDropModel
: (str) "logistic_ofMu", "logistic", "none" [Default "logistic_ofMu"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors. "none" - negative binomial noise model without zero-inflation.

strDropFitGroup
: (str) "PerCell", "AllCells" [Defaul "PerCell"] Definition of groups on cells on which separate drop-out model parameterisations are fit. "PerCell" - one parametersiation (fit) per cell "ForAllCells" - one parametersiation (fit) for all cells

MAXIT_BFGS_Pi
: (sca) Maximum number of iterations in BFGS estimation of dropout models. This is a control parameter to optim().

RELTOL_BFGS_Pi
: (sca) Relative tolerance of BFGS estimation of dropout models. This is a control parameter to optim().

Value

lsDropModel (list) Initialisation of drop-out model object.

Author(s)

David Sebastian Fischer

See Also

Called by `fitModel`.

`initialiseImpulseParameters`
Estimate impulse model parameter initialisations

Description

The initialisations reflect intuitive parameter choices corresponding to a peak and to a valley model and are based natural cubic spline fits in log space with a gaussian noise model.

Usage

```
initialiseImpulseParameters(vecCounts, lsMuModelGlobal)
```

Arguments

`vecCounts`	(count vector number of samples) Count data.
`lsMuModelGlobal`	(list) Object containing meta-data of gene-wise mean parameter models.

Value

list (length 2)

- peak: (numeric vector number of impulse model parameters) Impulse model parameter initialisation corresponding to a peak.
- valley: (numeric vector number of impulse model parameters) Impulse model parameter initialisation corresponding to a valley.

Author(s)

David Sebastian Fischer

See Also

Called by impulse model fitting wrappers: `fitImpulseZINB`.

`initMuModel` *Initialise mean model container object*

Description

Either use supplied fits from previous fitting or initialise from count data.

Usage

```
initMuModel(matCounts, dfAnnotation, vecConfoundersMu, vecNormConst,
  scaDFSplinesMu, matWeights, matMuModelInit, lsmatBatchModelInitMu, strMuModel,
  MAXIT_BFGS_MuDisp, RELTOL_BFGS_MuDisp)
```

Arguments

`matCounts`	(matrix genes x cells) Count data of all cells, unobserved entries are NA.
`dfAnnotation`	(data frame cells x meta characteristics) Annotation table which contains meta data on cells.
`vecConfoundersMu`	(vector of strings number of confounders on mean) [Default NULL] Confounders to correct for in mu batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
`vecNormConst`	(numeric vector number of cells) Model scaling factors, one per cell. These factors linearly scale the mean model for evaluation of the loglikelihood.
`scaDFSplinesMu`	(sca) [Default NULL] If strMuModel=="splines", the degrees of freedom of the natural cubic spline to be used as a mean parameter model.
`matWeights`	(numeric matrix cells x mixtures) [Default NULL] Assignments of cells to mixtures (for strMuModel="MM").
`matMuModelInit`	(numeric matrix genes x mu model parameters) [Default NULL] Contains initialisation of mean model parameters according to the used model.
`lsmatBatchModelInitMu`	(list) [Default NULL] Initialisation of batch correction models for mean parameter.
`strMuModel`	(str) "constant", "groups", "MM", "splines","impulse" [Default "impulse"] Model according to which the mean parameter is fit to each gene as a function of population structure in the alternative model (H1).
`MAXIT_BFGS_MuDisp`	(sca) Maximum number of iterations in BFGS estimation of Mu/Disp models. This is a control parameter to optim().
`RELTOL_BFGS_MuDisp`	(sca) Relative tolerance of BFGS estimation of Mu/Disp models. This is a control parameter to optim().

Value

lsMuModel (list) Initialisation of mean model object.

Author(s)

David Sebastian Fischer

See Also

Called by `fitModel`.

`LineagePulseObject-class`

Container class for LineagePulse output

Description

LineagePulse output and intermediate results such as model fits.

Slots

`dfAnnotationProc` (data frame cells x meta characteristics) Annotation table which contains meta data on cells.

`dfResults` (data frame samples x reported characteristics) Summary of fitting procedure and differential expression results for each gene.

`lsDispModelH1` (list) Object containing description of gene-wise dispersion parameter models of H1.

`lsDispModelH0` (list) Object containing description of gene-wise dispersion parameter models of H0.

`lsDispModelConst` (list) Object containing description of gene-wise dispersion parameter models of constant model (necessary if H0 is not constant (mixture models)).

`lsDispModelH1_NB` (list) Object containing description of gene-wise dispersion parameter models of H1 with NB noise model.

`lsDispModelH0_NB` (list) Object containing description of gene-wise dispersion parameter models of H0 with NB noise model.

`lsDropModel` (list)

`lsMuModelH0` (list) Object containing description of gene-wise mean parameter models of H0.

`lsMuModelH1` (list) Object containing description of gene-wise mean parameter models of H1.

`lsMuModelConst` (list) Object containing description of gene-wise means parameter models of constant model (necessary if H0 is not constant (mixture models)).

`lsMuModelH0_NB` (list) Object containing description of gene-wise mean parameter models of H0 with NB noise model.

`lsMuModelH1_NB` (list) Object containing description of gene-wise mean parameter models of H1. with NB noise model.

`lsFitConvergence` (list) Fitting reporter summary.

`matCountsProc` (count matrix genes x cells) Sparse matrix of counts, non-observed values are NA.

`matWeights` (numeric matrix cells x mixtures) Assignments of cells to mixtures.

`scaDFSplinesDisp` (scalar) If strDispModelFull=="splines" or strDispModelRed=="splines", the degrees of freedom of the natural cubic spline to be used as a dispersion parameter model.

`scaDFSplinesMu` (scalar) If strMuModel=="splines", the degrees of freedom of the natural cubic spline to be used as a mean parameter model.

`strReport` (str) LineagePulse stdout report (log).

`vecAllGenes` (vector of strings) All genes originally supplied, including all zero genes.

`vecConfoundersDisp` (vector of strings number of confounders on dispersion) Confounders to correct for in dispersion batch correction model, are subset of column names of dfAnnotation which describe condounding variables.

`vecConfoundersMu` (vector of strings number of confounders on mean) Confounders to correct for in mu batch correction model, are subset of column names of dfAnnotation which describe condounding variables.

`scaOmega` (scalar) Regularisation parameter on entropy penalty on mixture weights. Needed for mixture model mode only.

`boolFixedPopulations` Whether a population of cells is to be fixed to be assigned to a subset of centroids (RSA scenario). Needed for mixture model mode only.

`vecNCentroidsPerPop` Number of centroids per population (RSA scenario). Needed for mixture model mode only.

`vecH0Pop` (integer vector) Needed for mixture model mode only.

`vecNormConst` (numeric vector length number of cells) Normalisation constants (library size factors) for each cell.

`strVersion` (str) Version of LineagePulse that was run.

Author(s)

David Sebastian Fischer

`LPsetters` *LineagePulseObject setters*

Description

Setters for LineagePulse-Object. Set the entry defined by the function name to in objLP to value.

Usage

```
dfAnnotationProc(objLP) <- value

dfResults(objLP) <- value

lsMuModelH0(objLP) <- value

lsMuModelH1(objLP) <- value

lsMuModelConst(objLP) <- value

lsMuModelH0_NB(objLP) <- value

lsMuModelH1_NB(objLP) <- value

lsDispModelH0(objLP) <- value

lsDispModelH1(objLP) <- value
```

```
lsDispModelConst(objLP) <- value

lsDispModelH0_NB(objLP) <- value

lsDispModelH1_NB(objLP) <- value

lsDropModel(objLP) <- value

lsFitConvergence(objLP) <- value

matCountsProc(objLP) <- value

matWeights(objLP) <- value

scaDFSplinesDisp(objLP) <- value

scaDFSplinesMu(objLP) <- value

strReport(objLP) <- value

vecAllGenes(objLP) <- value

vecConfoundersDisp(objLP) <- value

vecConfoundersMu(objLP) <- value

scaOmega(objLP) <- value

boolFixedPopulation(objLP) <- value

vecH0Pop(objLP) <- value

vecNormConst(objLP) <- value

strVersion(objLP) <- value
```

Arguments

objLP	(LineagePulse-Object) A LineagePulse output object to write into.
value	(str) The value to be set.

Value

The LineagePulse output object with updated entry.

Author(s)

David Sebastian Fischer

`names,LineagePulseObject-method`
List-like accessor methods for LineagePulseObject: names()

Description

names() function for LineagePulseObject. Allow usage of LineagePulse ouput object like a list with respect to the most relevant output: dfLineagePulseResults and vecDEGenes. List of all available list-object like accessors: names,LineagePulseObject-method, [[,LineagePulseObject,character,missing-method, $,LineagePulseObject-method.

Usage

```
## S4 method for signature 'LineagePulseObject'
names(x)
```

Arguments

x	(LineagePulseObject) LineagePulse output object.

Value

Names of elements in x available via list-like accessors.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 10,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
names(objLP)
```

`plotCellDensity` *Plot density of cells in continuous covariate*

Description

Uses kernel density estimate.

Usage

```
plotCellDensity(objLP)
```

Arguments

`objLP`	(LineagePulseObject) LineagePulseObject to base plot on.

Value

gplotKDE (ggplot object) ggplot2 kernel density estimator plot.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
gplotCellDensity <- plotCellDensity(objLP = objLP)
#print(gplotCellDensity)
```

plotGene *Plot counts and model for one gene*

Description

Plot counts and model for one gene as a scatter plot with model trajectories: Both as counts or log10 counts vs continuous covariate . Dropout rates can be given and are visualised as the colour of the observation points. A reference model trajectory can be added. Contour plots: A contour plot aids the interpretation of the model trajectory in context of the observed data if local overplotting occurs because of a non-uniform distribution of cells in the continuous covariate (a very common scenario in pseudotime for example). In such a case, one might mistake cells with high counts from an interval with high cellular intensity as a indication of increased expression in this interval. Instead, one can explain such local maxima based on the simple phenomenom that more samples were taken from the underyling distribution in that continuous covariate interval distribution. To migitate this visually, we added the option to add "contours": A contour is an alternative to a confidence interval which includes information on the local sampling density. The contour is specific to a number of cells n. It is an expression trajectory in continuous covariate which lies at the line above which n cells are expected within a local continuous covariate bin given its sampling density and the H1 non-constant expression model. Contours therefore quantify the increase in maximum observed counts in continuous covariate intervals due to sampling density, which is not due to the expression model. Note that this effect is corrected for in the model fitting and that the contour plots just aids visual interpretation of the fit to the data!

Usage

```
plotGene(objLP, strGeneID, vecReferenceMuParam = NULL,
  strTitleSuffix = NULL, boolLogPlot = TRUE, boolColourByDropout = TRUE,
  boolH1NormCounts = FALSE, boolLineageContour = FALSE, boolTime = FALSE,
  bwDensity = NULL, scaGgplot2Size = 0.5, scaGgplot2Alpha = 0.5)
```

Arguments

objLP	(LineagePulseObject) LineagePulseObject base plot on.
strGeneID	(str) Name of gene, used for title of plot.
vecReferenceMuParam	(numeric vector length number of cells) [Default NULL] Reference mean trajectory which can be plotted
strTitleSuffix	(str) String to be added to title.
boolLogPlot	(bool) [Default TRUE] Whether to log transform y-axis.
boolColourByDropout	(bool) [Default TRUE] Whether to colour scatter plot by posterior of drop-out.
boolH1NormCounts	(bool) [Default FALSE] Whether to show normalised counts (size factors and H1 batch factor estimates) as oppose to raw counts.
boolLineageContour	(bool) [Default FALSE] Whether to the "lineage contour" lines to the scatter plot.
boolTime	(bool) [Default TRUE] Whether continuous covariate is time, this simplifies the scatter plot strongly. Show mean expression per time point as orange point and 25% and 75% quantile of observations per time point as error bars.

bwDensity (bandwith numeric or string) [Default NULL] Bandwith to be used to kernel density smooting of cell density in continuous covariate (used if boolLineageContour=TRUE). If not set, defaults to stats:density() default.

scaGgplot2Size (scalar) size in ggplot2 scatter

scaGgplot2Alpha (scalar) alpha in ggplot2 scatter

Value

gplotGene (ggplot object) Model rajectories and scatter plot for given gene.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
matDropoutPredictors <- as.matrix(data.frame(
    log_means = log(rowMeans(lsSimulatedData$counts)+1) ))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "splines", scaDFSplinesMu = 6,
    strDropModel="logistic",
    matPiConstPredictors = matDropoutPredictors)
gplotExprProfile <- plotGene(
    objLP = objLP,
    strGeneID = rownames(lsSimulatedData$counts)[1],
    boolLineageContour = FALSE)
#print(gplotExprProfile)
```

---

processSCData *Prepare single cell data for analysis*

Description

Check that input is correctly supplied and formatted. Then process input data for analysis. This function catches downstream errors arising from incorrect input parameter combinations and explains the input error to the user. Input passing this function should not lead to errors arising from incorrect input. Helper functions: checkNull() Check whether object was supplied (is not NULL). checkNumeric() Checks whether elements are numeric. checkCounts() Checks whether elements are count data.

Usage

```
processSCData(counts, dfAnnotation, vecConfoundersDisp, vecConfoundersMu,
  matPiConstPredictors, vecNormConstExternal, strDispModelFull, strDispModelRed,
  strMuModel, scaDFSplinesDisp, scaDFSplinesMu, scaMaxEstimationCycles,
  boolVerbose, boolSuperVerbose)
```

Arguments

counts	(matrix genes x samples) (matrix genes x cells (sparseMatrix or standard) or file) Matrix: Count data of all cells, unobserved entries are NA. file: .mtx file from which count matrix is to be read.
dfAnnotation	(data frame cells x meta characteristics) Annotation table which contains meta data on cells.
vecConfoundersDisp	(vector of strings number of confounders on dispersion) [Default NULL] Confounders to correct for in dispersion batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
vecConfoundersMu	(vector of strings number of confounders on mean) [Default NULL] Confounders to correct for in mu batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.
matPiConstPredictors	(numeric matrix genes x number of constant gene-wise drop-out predictors) Predictors for logistic drop-out fit other than offset and mean parameter (i.e. parameters which are constant for all observations in a gene and externally supplied.) Is null if no constant predictors are supplied.
vecNormConstExternal	(numeric vector number of cells) Model scaling factors, one per cell. These factors will linearly scale the mean model for evaluation of the loglikelihood. Must be named according to the column names of matCounts. Supplied by user.
strDispModelFull	(str) "constant" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the alternative model (H1).
strDispModelRed	(str) "constant" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the null model (H0).
strMuModel	(str) "constant" [Default "impulse"] Model according to which the mean parameter is fit to each gene as a function of population structure in the alternative model (H1).
scaDFSplinesDisp	(sca) [Default 3] If strDispModelFull=="splines" or strDispModelRed=="splines", the degrees of freedom of the natural cubic spline to be used as a dispersion parameter model.
scaDFSplinesMu	(sca) [Default 3] If strMuModel=="splines", the degrees of freedom of the natural cubic spline to be used as a mean parameter model.
scaMaxEstimationCycles	(integer) [Default 20] Maximum number of estimation cycles performed in fitZINB(). One cycle contain one estimation of of each parameter of the zero-inflated negative binomial model as coordinate ascent.

`boolVerbose` (bool) Whether to follow convergence of the iterative parameter estimation with one report per cycle.

`boolSuperVerbose` (bool) Whether to follow convergence of the iterative parameter estimation in high detail with local convergence flags and step-by-step loglikelihood computation.

Value

list (length 3)

- objLP (LineagePulseObject) Initialisation of LineagePulseObject.
- matCountsProcFull (matrix genes x samples) Processed count data of all cells, unobserved entries are NA.
- matPiConstPredictorsProc (numeric matrix genes x number of constant gene-wise drop-out predictors) Processed version of matPiConstPredictors.

Author(s)

David Sebastian Fischer

See Also

Called by `runLineagePulse`.

---

runDEAnalysis *Differential expression analysis*

Description

Performs differential expression analysis based on previously estimated null and alternative models. (I) Compute loglikelihood of data under null H0 and alternative H1 model. (II) Differential expression analysis as loglikelihood ratio test.

Usage

```
runDEAnalysis(objLP)
```

Arguments

`objLP` (LineagePulseObject) LineagePulseObject with fitted null and alternative models.

Value

objLP (LineagePulseObject) LineagePulseObject with analysis summary table (dfResults) added.

- Gene: Gene ID.
- p: P-value for differential expression with ZINB noise.
- mean: Inferred mean parameter of constant model of first batch.

- padj: Benjamini-Hochberg false-discovery rate corrected p-value for differential expression analysis with NB noise.
- p_nb: P-value for differential expression with ZINB noise.
- padj_nb: Benjamini-Hochberg false-discovery rate corrected p-value for differential expression analysis with NB noise.
- loglik_full_zinb: Loglikelihood of full model with ZINB noise.
- loglik_red_zinb: Loglikelihood of reduced model with ZINB noise.
- loglik_full_nb: Loglikelihood of full model with NB noise.
- loglik_red_nb: Loglikelihood of reduced model with NB noise.
- df_full: Degrees of freedom of full model.
- df_red: Degrees of freedom of reduced model
- allZero (bool) Whether there were no observed non-zero observations of this gene. If TRUE, fitting and DE analsysis were skipped and entry is NA.

Author(s)

David Sebastian Fischer

See Also

Called by `runLineagePulse`.

`runLineagePulse` *LineagePulse wrapper: Differential expression analysis on scRNA-seq*

Description

This function performs all steps of longitudinal or discrete differential expression analysis in a continuous covariate (such as pseudotime) or according to a grouping (such as clusters or dconditions).

Usage

```
runLineagePulse(counts, dfAnnotation = NULL, vecConfoundersMu = NULL,
  vecConfoundersDisp = NULL, strMuModel = "splines",
  strDispModelFull = "constant", strDispModelRed = "constant",
  strDropModel = "logistic", strDropFitGroup = "PerCell",
  scaDFSplinesMu = 6, scaDFSplinesDisp = 3, matPiConstPredictors = NULL,
  vecNormConstExternal = NULL, boolEstimateNoiseBasedOnH0 = TRUE,
  scaMaxEstimationCycles = 20, scaNProc = 1, boolVerbose = TRUE,
  boolSuperVerbose = FALSE)
```

Arguments

`counts`	(matrix genes x cells (sparseMatrix or standard), SummarizedExperiment or file) Matrix: Count data of all cells, unobserved entries are NA. SummarizedExperiment or SingleCellExperiment: Count data of all cells in assay(counts) and annotation data can be supplied as colData(counts) or separately via dfAnnotation. file: .mtx file from which count matrix is to be read.

dfAnnotation (data frame cells x meta characteristics) [Default NULL] Annotation table which contains meta data on cells. This data frame may be supplied as colData(counts) if counts is a SummerizedExperiment or SingleCellExperiment object. May contain the following columns cell: Cell IDs. continuous: Pseudotemporal coordinates of cells. Confounder1: Batch labels of cells with respect to first confounder. Name is arbitrary: Could for example be "patient" with batch labels patientA, patientB, patientC. Confounder2: As Confounder1 for another confounding variable. ... ConfounderX. population: Fixed population assignments (for strMuModel="MM"). Cells not assigned have to be NA. groups: Discrete grouping of cells (e.g. clusters or experimental conditions which are to be used as popuation structure if strMuModel or strDispModel are "groups"). rownames: Must be IDs from column cell. Remaining entries in table are ignored.

vecConfoundersMu (vector of strings number of confounders on mean) [Default NULL] Confounders to correct for in mu batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.

vecConfoundersDisp (vector of strings number of confounders on dispersion) [Default NULL] Confounders to correct for in dispersion batch correction model, must be subset of column names of dfAnnotation which describe condounding variables.

strMuModel (str) "constant", "groups", "MM", "splines","impulse" [Default "splines"] Model according to which the mean parameter is fit to each gene as a function of population structure in the alternative model (H1).

strDispModelFull (str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the alternative model (H1).

strDispModelRed (str) "constant", "groups", "splines" [Default "constant"] Model according to which dispersion parameter is fit to each gene as a function of population structure in the null model (H0).

strDropModel (str) "logistic_ofMu", "logistic" [Default "logistic"] Definition of drop-out model. "logistic_ofMu" - include the fitted mean in the linear model of the drop-out rate and use offset and matPiConstPredictors. "logistic" - only use offset and matPiConstPredictors.

strDropFitGroup (str) "PerCell", "AllCells" [Defaul "PerCell"] Definition of groups on cells on which separate drop-out model parameterisations are fit. "PerCell" - one parametersiation (fit) per cell "ForAllCells" - one parametersiation (fit) for all cells

scaDFSplinesMu (sca) [Default 6] If strMuModel=="splines", the degrees of freedom of the natural cubic spline to be used as a mean parameter model.

scaDFSplinesDisp (sca) [Default 3] If strDispModelFull=="splines" or strDispModelRed=="splines", the degrees of freedom of the natural cubic spline to be used as a dispersion parameter model.

matPiConstPredictors (numeric matrix genes x number of constant gene-wise drop-out predictors) Predictors for logistic drop-out fit other than offset and mean parameter (i.e. parameters which are constant for all observations in a gene and externally supplied.) Is null if no constant predictors are supplied

`vecNormConstExternal`
: (numeric vector number of cells) Model scaling factors, one per cell. These factors will linearly scale the mean model for evaluation of the loglikelihood. Must be named according to the column names of matCounts.

`boolEstimateNoiseBasedOnH0`
: (bool) [Default TRUE] Whether to co-estimate logistic drop-out model with the constant null model or with the alternative model. The co-estimation with the noise model typically extends the run-time of this model-estimation step strongly. While the drop-out model is more accurate if estimated based on a more realistic model expression model (the alternative model), a trade-off for speed over accuracy can be taken and the dropout model can be chosen to be estimated based on the constant null expression model (set to TRUE).

`scaMaxEstimationCycles`
: (integer) [Default 20] Maximum number of estimation cycles performed in fitZINB(). One cycle contain one estimation of of each parameter of the zero-inflated negative binomial model as coordinate ascent.

`scaNProc`
: (scalar) [Default 1] Number of processes for parallelisation.

`boolVerbose`
: (bool) Whether to follow convergence of the iterative parameter estimation with one report per cycle.

`boolSuperVerbose`
: (bool) Whether to follow convergence of the iterative parameter estimation in high detail with local convergence flags and step-by-step loglikelihood computation.

Details

This function is the wrapper function for the LineagePulse algorithm which performs differential expression analysis in pseudotime. Note that LineagePulse has many input parameters but only few will be relevant for you and you will be able to leave the remaining ones as their defaults. Read up on specific input parameters in the input parameter annotation of this function in the vignette.

Value

dfDEAnalysis (data frame genes x reported variables) Summary of differential expression analysis:

- Gene: Gene ID.
- p: P-value for differential expression with ZINB noise.
- mean: Inferred mean parameter of constant model of first batch.
- padj: Benjamini-Hochberg false-discovery rate corrected p-value for differential expression analysis with NB noise.
- p_nb: P-value for differential expression with ZINB noise.
- padj_nb: Benjamini-Hochberg false-discovery rate corrected p-value for differential expression analysis with NB noise.
- loglik_full_zinb: Loglikelihood of full model with ZINB noise.
- loglik_red_zinb: Loglikelihood of reduced model with ZINB noise.
- loglik_full_nb: Loglikelihood of full model with NB noise.
- loglik_red_nb: Loglikelihood of reduced model with NB noise.
- df_full: Degrees of freedom of full model.
- df_red: Degrees of freedom of reduced model
- allZero (bool) Whether there were no observed non-zero observations of this gene. If TRUE, fitting and DE analsysis were skipped and entry is NA.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
matDropoutPredictors <- as.matrix(data.frame(
    log_means = log(rowMeans(lsSimulatedData$counts)+1) ))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "splines", scaDFSplinesMu = 6,
    strDropModel = "logistic",
    matPiConstPredictors = matDropoutPredictors)
tail(objLP$dfResults)
```

---

simulateContinuousDataSet

Simulate a data set for LinagePulse Simulates a data set with genes with constant and impulse expression traces. Expression strength and variation in impulse like traces are parameterised and random. All temporary files are saved into dirOutSimulation and only the objects necessary for running LineagePulse (the count matrix and the continuous covariate vector are returned). The remaining objects representing hidden parameters can be used to evaluate parameter estimates. Cells are distributed uniformly in the continuous covariate.

Description

Set either vecGeneWiseDropoutRates or matDropoutModelExternal.

Usage

```
simulateContinuousDataSet(scaNCells, scaNConst, scaNLin, scaNImp,
  scaMumax = 100, scaSDMuAmplitude = 1, vecNormConstExternal = NULL,
  vecDispExternal = NULL, vecGeneWiseDropoutRates = NULL,
  matDropoutModelExternal = NULL)
```

Arguments

scaNCells	(scalar) Number of cells in data set.
scaNConst	(scalar) Number of constant expression profiles (genes) in data set.

scaNLin (scalar) Number of linear expression profiles (genes) in data set.

scaNImp (scalar) Number of impulse model expression profiles (genes) in data set.

scaMumax (scalar) [Default 1000] Maximum expression mean parameter to be used.

scaSDMuAmplitude
(scalar) [Default 1] Standard deviation of normal distribution form which the amplitude change within an impulse trace is drawn.

vecNormConstExternal
(numeric vector number of cells) [Default NULL] Size factors for data set. Size factors are set to 1 if this is not specified (NULL).

vecDispExternal
(numeric vector number of genes) [Default NULL] Dispersion parameters per gene supplied by user.s

vecGeneWiseDropoutRates
(numeric vector number of cells) [Default NULL] One drop-out rate per gene.

matDropoutModelExternal
(numeric matrix cells x 2) [Default NULL] External drop-out model, has to have one row for each simulated cell.

Value

list (length 2)

- continuous (numerical vector length number of cells) Continuous covariates (1D) of cells. One scalar per cell. This covariate could be pseudotime for example.'
- counts (matrix genes x cells) Sampled count data of all cells after drop-out.

Author(s)

David Sebastian Fischer

See Also

Called by separately by user.

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
plot(lsSimulatedData$annot$continuous, lsSimulatedData$counts[1,])
```

`sortGeneTrajectories` *Cluster expression mean trajectories*

Description

Sorts inferred gene trajectories by peak time in continuous covariate. Optional: Can create a heatmap of the gene trajectories sorted according to peak time. The heatmap is based on z-scores.

Usage

```
sortGeneTrajectories(vecIDs, lsMuModel, dirHeatmap = NULL)
```

Arguments

`vecIDs`	(vector of strings) Names of genes to cluster.
`lsMuModel`	(list) Object containing description of gene-wise mean parameter models.
`dirHeatmap`	(str directory) [Default NULL] Directory to which heatmap is saved to. Return heatmap object if NULL.

Value

list (length 3) If dirHeatmap is not NULL, only vecSortedGenes is returned and the two heatmaps are printed to pdfs in the directory dirHeatmap. vecSortedGenes: (string vector number of IDs) hmGeneSorted: genes sorted by peak time in continuous covariate hmGeneClusters: genes sorted by clustering

Author(s)

David Sebastian Fischer

See Also

Called by user.

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
matDropoutPredictors <- as.matrix(data.frame(
    log_means = log(rowMeans(lsSimulatedData$counts)+1) ))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "splines", scaDFSplinesMu = 6,
    strDropModel="logistic",
```

```
    matPiConstPredictors = matDropoutPredictors)
lsHeatmaps <- sortGeneTrajectories(
    vecIDs = objLP$dfResults[which(objLP$dfResults$padj < 0.01),]$gene,
    lsMuModel = lsMuModelH1(objLP),
    dirHeatmap = NULL)
#print(lsHeatmaps$hmGeneSorted)
```

`testDropout` *Test for existance of drop-out with log-likelihood ratio test*

Description

Performs one test for entire data set.

Usage

```
testDropout(objLP)
```

Arguments

`objLP`	(LineagePulseObject) LineagePulseObject with fitted null and alternative models.

Value

(data frame) Summary of hypothesis test

- Gene: Gene ID.
- p: P-value for existance of drop-out in data set. If high, the data can be explained with models based on NB noise and zero-inflation is not necessary: The null hypothesis of no zero-inflation cannot be rejected.
- loglik_zinb: Loglikelihood of full model with ZINB noise (all genes).
- loglik_nb: Loglikelihood of reduced model with NB noise (all genes).
- df_full: Degrees of freedom of full model with ZINB noise (all genes).
- df_red: Degrees of freedom of reduced model with NB noise (all genes).

Author(s)

David Sebastian Fischer

See Also

Called by user.

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 100,
    scaNConst = 10,
    scaNLin = 10,
    scaNImp = 10,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 30),
    vecGeneWiseDropoutRates = rep(0.1, 30))
matDropoutPredictors <- as.matrix(data.frame(
    log_means = log(rowMeans(lsSimulatedData$counts)+1) ))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "splines", scaDFSplinesMu = 6,
    strDropModel="logistic",
    matPiConstPredictors = matDropoutPredictors)
testDropout(objLP)$p
```

---

writeReport *Print LineagePulse report*

Description

Print LineagePulse report string to .txt file or stdout.

Usage

```
writeReport(object, file = "")
```

Arguments

object	(LineagePulseObject) Output object of LineagePulse.
file	(file) [DEFAULT ""] File to print report to. Default is stdout.

Value

nothing to return

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 10,
    scaNConst = 2,
    scaNLin = 2,
```

```
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
writeReport(objLP, file="")
```

[[,LineagePulseObject,character,missing-method

List-like accessor methods for LineagePulseObject: names()

Description

names() function for LineagePulseObject. Allow usage of LineagePulse ouput object like a list with respect to the most relevant output: dfLineagePulseResults and vecDEGenes. List of all available list-object like accessors: names,LineagePulseObject-method, [[,LineagePulseObject,character,missing-method, $,LineagePulseObject-method.

Usage

```
## S4 method for signature 'LineagePulseObject,character,missing'
x[[i, j, ...]]
```

Arguments

x	(LineagePulseObject) LineagePulse output object.
i	(str) Element from x list to be retrieved.
j	() Ignored argument to generic.
...	() Ignored argument to generic.

Value

Target element from x.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 10,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
```

```
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
head(objLP[["dfResults"]])
```

`$,LineagePulseObject-method`

List-like accessor methods for LineagePulseObject: $

Description

$ accessor function for LineagePulseObject, relies on [[ ]]. Allow usage of LineagePulse ouput object like a list with respect to the most relevant output: dfLineagePulseResults and vecDEGenes. List of all available list-object like accessors: names,LineagePulseObject-method, [[,LineagePulseObject,character,missi method, $,LineagePulseObject-method.

Usage

```
## S4 method for signature 'LineagePulseObject'
x$name
```

Arguments

x	(LineagePulseObject) LineagePulse output object.
name	(str) Element from x list to be retrieved.

Value

Target element from x.

Author(s)

David Sebastian Fischer

Examples

```
lsSimulatedData <- simulateContinuousDataSet(
    scaNCells = 10,
    scaNConst = 2,
    scaNLin = 2,
    scaNImp = 2,
    scaMumax = 100,
    scaSDMuAmplitude = 3,
    vecNormConstExternal=NULL,
    vecDispExternal=rep(20, 6),
```

```
    vecGeneWiseDropoutRates = rep(0.1, 6))
objLP <- runLineagePulse(
    counts = lsSimulatedData$counts,
    dfAnnotation = lsSimulatedData$annot,
    strMuModel = "impulse")
head(objLP$dfResults)
```

Index