Package 'M3Drop'

October 21, 2025

Version 1.35.0

Date 2024-03-25

Title Michaelis-Menten Modelling of Dropouts in single-cell RNASeq

Author Tallulah Andrews <tallulandrews@gmail.com>

Maintainer Tallulah Andrews <tallulandrews@gmail.com>

Depends R (>= 3.4), numDeriv

Imports RColorBrewer, gplots, bbmle, statmod, grDevices, graphics, stats, matrixStats, Matrix, irlba, reldist, Hmisc, methods, scater

Suggests ROCR, knitr, M3DExampleData, SingleCellExperiment, Seurat, Biobase

VignetteBuilder knitr

biocViews RNASeq, Sequencing, Transcriptomics, GeneExpression, Software, DifferentialExpression, DimensionReduction, FeatureExtraction

Collate basics.R Plotting_fxns.R Curve_fitting.R Extremes.R Normalization.R Brennecke_implementation.R Threeway_ProportionalArea_VennDiagrams.R Simulations_Functions.R Traditional_DE.R NB_UMI.R M3D_Imputation.R Other_FS_functions.R DANB_HVG.R DANB_Coexpression.R

Description This package fits a model to the pattern of dropouts in single-cell RNASeq data. This model is used as a null to identify significantly variable (i.e. differentially expressed) genes for use in downstream analysis, such as clustering cells. Also includes an method for calculating exact Pearson residuals in UMI-tagged data using a library-size aware negative binomial model.

URL https://github.com/tallulandrews/M3Drop

BugReports https://github.com/tallulandrews/M3Drop/issues

License GPL (>=2)

git_url https://git.bioconductor.org/packages/M3Drop

git_branch devel

git_last_commit 66aa05f

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-10-20

Contents

bg__calc_variables *Calculate Gene Variables*

Description

Calculates a suite of gene-specific variables including: mean, dropout rate, and their standard errors.

Usage

```
bg__calc_variables(expr_mat)
```

Arguments

expr_mat	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.

Details

Calculates mean expression and its standard error (sd/sqrt(n)), and the dropout rate, proportion of observations that are zero, and its standard error (sqrt(p*(1-p)/n)) for each gene in the dataset. Performs various checks to ensure expression matrix is of suitable format for M3Drop. Removes undetected genes if they are present.

Value

Named list of calculated values: s = vector of mean expression for each gene s_stderr = vector of mean expression standard error for each gene p = vector of dropout rate for each gene p_stderr = vector of dropout rate standard error for each gene

bg__default_mean2disp *Mean to Dispersion*

Description

Default function to calculate the negative binomial dispersion parameter given the mean expression of the gene.

Usage

```
bg__default_mean2disp(mu, coeffs=c(3.967816,-1.855054))
```

Arguments

mu	mean expression of the gene(s)
coeffs	coefficients of linear regression between log(cv2) and log10(mean expression) for observed data.

Details

Given the coefficients of a linear regression between the natural logarithm of the squared coefficient of variation and log base 10 of mean expression, calculates the dispersion parameter (1/size) of the negative binomial distribution for a given expression level.

Default coefficients were fit to the Buettner at al.[1] single cell mESC data.

Value

the dispersion parameter (1/size) for the (mu,size) paramterization of the `rnbinom` function.

References

[1] Buettner et al. (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nature Biotechnology 33 : 155-160.

Examples

```
# mu <- 100
# r <- bg__default_mean2disp(mu)
# n_cells <- 50
# sim <- rnbinom(n_cells, size=1/r, mu=mu)
```

---

bg__fit_gamma *Fit Gamma Distribution*

Description

Fits the parameters of a gamma distribution to a set of observations

Usage

```
bg__fit_gamma(x)
```

Arguments

x	vector of observations to fit.

Details

scale = var(x)/mean(x) shape = mean(x)/scale

Value

list with two entries: shape and size.

Examples

```
# dat <- rgamma(100, 0.1, 1)
# params <- bg__fit_gamma(dat)
```

`bg__fit_size_to_var` *Fit gene-specific dispersion*

Description

Fits dispersion for a specific gene for the depth-adjusted negative binomial. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
bg__fit_size_to_var(obs, mu_vec, max_size, min_size=10^-10, convergence=0.001)
```

Arguments

`obs`	observed variance corrected for library size.
`mu_vec`	expectation of depth-adjusted negative-binomial for each observation
`max_size`	maximum dispersion for genes that exhibit Poissonian expression.
`min_size`	initial dispersion estimate.
`convergence`	acceptable error on dispersion estimate.

Details

Minimized the difference between the mean of expected variances for each observation based on the depth-adjust negative binomial(DANB) and the observed variance. The observed variance is corrected for library size by subtracting the observation specific mean as per the DANB from the observed UMI counts prior to calculating the variance. The optimization is done using an iterative method similar to Newton's method, which ends when the difference between estimates is less than "convergence" or if the estimated dispersion exceeds "max_size" or after 1000 iterations.

Value

Estimated dispersion.

Examples

```
# library(M3DExampleData)
# counts <- as.matrix(Mmus_example_list$data);
# counts <- counts[rowSums(counts) > 0,];
# fit <- NBumiFitModel(counts);
# coeffs <- NBumiFitDispVsMean(fit, suppress.plot=TRUE);
```

bg__get_stats *Calculate Simulation Statistics*

Description

Given significant genes and a ground truth calculates the false discovery rate and false negative rate of the significance calls.

Usage

```
bg__get_stats(sig, TP, ngenes)
```

Arguments

sig	vector of gene IDs/names which were called as significant by some method.
TP	vector of gene IDs/names which known to be positives (ground truth).
ngenes	number of total genes under consideration.

Details

False discovery rate (FDR) is the proportion of all genes considered significant which are not in the ground truth. False negative rate (FNR) is the proportion of all ground truth genes which are not considered significant.

Value

vector of two elements: (FDR, FNR).

Examples

```
# Calls <- c(1,3,5,8,10)
# Truth <- c(2,3,5,9)
# total=10
# stats <- bg__get_stats(Calls, Truth, total)
```

bg__horizontal_residuals_MM_log10 *Calculate Horizontal Residuals*

Description

Calculates horizontal residuals from the Michaelis-Menten Function. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
bg__horizontal_residuals_MM_log10(K, p, s)
```

Arguments

K	fitted Michaelis-Menten constant.
p	Observed dropout rate.
s	Observed mean expression.

Details

Calculates the log-transformed horizontal residuals from the Michaelis-Menten function. Input values may be single values or vectors.

$$r = log_{10}(S) - \frac{(1-P)K}{P}$$

Value

Value of horizontal residual.

Examples

```
#  s <- c(10,100,100)
#  p <- c(0.9, 0.5, 0.1)
#  res <- bg__horizontal_residuals_MM_log10(10, p, s)
```

---

bg__MakeSimData *Make Simulated Data*

Description

Makes simulated data based on a negative binomial distribution inflated with zeros based on the Michaelis-Menten equation.

Usage

```
bg__MakeSimData(dispersion_fun=bg__default_mean2disp, n_cells=300, dispersion_factor=1, base_mea
bg__MakeSimDE(dispersion_fun=bg__default_mean2disp, fold_change=10, frac_change=0.1, n_cells=300
bg__MakeSimDVar(dispersion_fun=bg__default_mean2disp, fold_change=10, frac_change=0.1, n_cells=3
bg__MakeSimHVar(dispersion_fun=bg__default_mean2disp, fold_change=10, frac_change=0.1, n_cells=3
```

Arguments

dispersion_fun	a function which takes mean experssion and returns the dispersion parameter of the negative binomial distribution.
n_cells	total number of cells (columns) in the simulated dataset.
sub_pop	proportion of cells with changed expression.
frac_change	proportion of genes with changed expression.
fold_change	fold change in dispersion or mean expression.
dispersion_factor	a factor that multiplies the calculated mean-specific dispersion for all genes.
base_means	a vector of background mean expression values.
K	K of the Michaelis-Menten function

Details

Generates simulated single-cell gene expression data using a zero-inflated negative binomial distribution. A user-supplied function relates the dispersion parameter (1/size of the R parameterization of the negative binomial distribution). Zeros are added based on a Michaelis-Menten function.

Default values of base_means, K, and dispersion_fun were fit to the Buettner et al. 2015 data [1].

`bg__MakeSimData` generates simulated single-cell data for a single homogeneous population.

`bg__MakeSimDE` generates simulated single-cell data for two different populations where a proportion of genes have a `fold_change` difference in the mean for population "2".

`bg__MakeSimDVar` generates simulated single-cell data for two different populations where a proportion of genes have a `fold_change` difference in the dispersion for population "2".

`bg__MakeSimHVar` generates simulated single-cell data for a single homogeneous population where a proportion of genes have a `fold_change` increase in dispersion over the expectation given the mean expression of the gene.

Value

`bg__MakeSimData` : a gene expression matrix where rows are genes, columns are cells. `bg__MakeSimDE`, `bg__MakeSimDVar`, `bg__MakeSimHVar` : a list of three named items: data : the gene expression matrix where rows are genes, columns are cells cell_labels : a vector of 1 or 2 indicating which cells are the unchanged ("1") or changed ("2") population. TP : a vector of row IDs of those genes that change (true positives).

References

[1] Buettner et al. (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nature Biotechnology 33 : 155-160.

Examples

```
#  means = c(1,2,5,10,20,50,100,200,500,1000,2000,5000)
#  population1 <- bg__MakeSimData(n_cells=10, base_means=means)
#  population2 <- bg__MakeSimData(n_cells=10, base_means=means*2, dispersion_factor=0.5)
#  sim_DE <- bg__MakeSimDE(n_cells=100, base_means=means)
#  sim_DVar <- bg__MakeSimDVar(n_cells=100, sub_pop=0.25, base_means=means)
#  sim_HVar <- bg__MakeSimHVar(base_means=means, fold_change=3)
```

---

bg__nbumiGroupDE *Perform Traditional Differential Expression*

Description

Performs traditional (i.e. compare defined groups) differential expression using a negative binomial model with MM zero-inflation. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Arguments

`counts`	a numeric matrix of raw UMI counts, columns = samples, rows = genes.
`fit`	fit NB UMI model from NBumiFitModel
`groups`	a vector of biological group IDs for each cell(columns).

Details

THIS FUNCTION SHOULD NOT BE USED.

unfinished__nbumiGroupDE Uses a log-likelihood ratio test to perform model selection between a model of constant mean expression vs a model of different mean expression across the biological groups. Probabilities of observing the data given the model are calculated using a negative binomial distribution with means adjusted for the total molecules detected per cell and dispersion fit to observed variance and adjusted to the mean of each group based on a globally fit power-law relationship. Significance is evaluated using the chi-square distribution.

Value

A table of fold mean expression differences for each biological group relative to the global mean expression level with raw p-values and FDR corrected p-values for each gene.

Examples

```
#library(M3DExampleData)
#counts <- as.matrix(Mmus_example_list$data);
#counts <- counts[rowSums(counts) > 0,];
#counts <- counts[1:1000,]
#fit <- NBumiFitModel(counts);
#DE_output <- bg__nbumiGroupDE(counts, fit, Mmus_example_list$labels)
```

---

bg__shift_size *Shift Size Parameter*

Description

Shifts a fitted size paramter according to the a power-law relationship.

Usage

```
bg__shift_size(mu_all, size_all, mu_group, coeffs)
```

Arguments

mu_all	vector of original mus
size_all	vector of original sizes
mu_group	vector of new mus
coeffs	coefficients of the power-law fit

Details

Wrapper of short function which converts values to their logs then shifts them using the parameters of a linear fit before un-logging them.

Value

vector of new size parameters

Examples

```
#require("M3DExampleData")
#counts <- NBumiConvertData(Mmus_example_list$data, is.counts=TRUE);
#fit <- NBumiFitModel(counts);
#ceoffs <- NBumiFitDispVsMean(fit, suppress.plot=TRUE)
#new_size <- bg__shift_size(rowMeans(counts), fit$size, rowMeans(counts)*2, coeffs)
```

`bg__var_vs_drop` *Variance vs Dropout Rate*

Description

Fixes the mean expression level and simulates a single gene with different fold changes across two equally sized subpopulations.

Usage

```
bg__var_vs_drop(pop_size, fixed_mean, K=10.3, dispersion_from_mean=bg__default_mean2disp, suppres
```

Arguments

`pop_size`	size of each subpopulation, i.e. 1/2 the total number of cells.
`fixed_mean`	expected mean expression across all cells.
`K`	K of the Michaelis-Menten function.
`dispersion_from_mean`	function which calculates the dispersion parameter of the negative binomial distribution given the mean expression of the gene.
`suppress.plot`	Whether to plot variance vs fold change and dropout rate vs fold change plots.

Details

Simulates a single gene that is differentially expressed across two equally sized cell populations (`pop_size`) with a constant expected mean expression across all cells. Simulations are performed for fold changes of every integer value from 1 (no DE) to 100. Simulations use a negative binomial distribution inflated with zeros according to a Michaelis-Menten function.

Default mean2disp relationship and K were fit to the Buettner et al. [1] mESC data.

Calculates pearson correlations between observed sample variance and dropout rate and fold change of the differential expression. Sample variance is decomposed into between subpopulation and within subpopulation variance ANOVA. Optionally plots the relationship between observed sample variance and dropout rate and the fold change.

Value

Name list of values: var_r = pearson correlation between observed sample variance and fold change. drop_r = pearson correlation between observed dropout rate and fold change. vars = vector of observed sample variances drops = vector of observed dropout rates fc = vector of expected fold changes Vbtw = vector of between subpopulation variances Vwithin = vector of within subpopulation variances

References

[1] Buettner et al. (2015) Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. Nature Biotechnology 33 : 155-160.

Examples

```
# mu100_N200 <- bg__var_vs_drop(100,100, suppress.plot=FALSE)
# mu100_N2000 <- bg__var_vs_drop(1000,100)
# mu1000_N200 <- bg__var_vs_drop(1000,100)
# c(mu100_N200$var_r,mu100_N2000$var_r,mu1000_N200$var_r)
```

BrenneckeGetVariableGenes

Identify Highly Variable Genes

Description

Implements the method of Brennecke et al. (2013) to identify highly variable genes.

Usage

```
BrenneckeGetVariableGenes(expr_mat, spikes=NA, suppress.plot=FALSE, fdr=0.1, minBiolDisp=0.5, fitN
```

Arguments

`expr_mat`	a numeric matrix of normalized or raw (not log-transformed) expression values, columns = samples, rows = genes.
`spikes`	a vector of gene names of row numbers of spike-in genes which are subject to only technical variance.
`suppress.plot`	Whether to make the plot or just calculate the requisite values.
`fdr`	Use FDR to identify significantly highly variable genes.
`minBiolDisp`	Minimum percentage of variance due to biological factors.
`fitMeanQuantile`	Threshold for genes to be used in fitting. May need to be decreased in low-depth/umi-tagged datasets to achieve good fit.

Details

Identifies significantly highly variable genes as detailed in Brennecked et al [1]. If spike-ins are provided they are used fit a function to the relationship between gene expression and variance due to technical factors. If spike-ins are not provided then all genes are used in the fitting.

Value

Vector of names of highly variable genes.

References

Brennecke et al. (2013) Accounting for technical noise in single-cell RNA-seq experiments. Nature Methods 10, 1093-1095. doi:10.1038/nmeth.2645

Examples

```
library(M3DExampleData)
HVG <- BrenneckeGetVariableGenes(Mmus_example_list$data)
HVG_spike <- BrenneckeGetVariableGenes(Mmus_example_list$data, spikes=5550:5600)
```

Consensus_FS *Consensus Feature Selection*

Description

Performs seven different feature selection methods then calculates the consensus ranking of features from that.

Usage

```
Consensus_FS(counts, norm=NA, is.spike=rep(FALSE, times=nrow(counts)), pcs=c(2,3), include_cors=T
```

Arguments

counts	raw count matrix, rows=genes, cols=cells
norm	normalized but not log-transformed gene expression matrix, rows=genes, cols=cells
is.spike	logical, vector of whether each gene is/isn't a spike-in
pcs	which principle components to use to score genes
include_cors	logical, whether to perform gene-gene correlation feature selection which is much slower than all other methods.

Details

Performs: NBumiFeatureSelectionCombinedDrop (aka: DANB_drop) NBumiFeatureSelectionHighVar (aka: DANB_var) BrenneckeGetVariableGenes (aka: HVG) M3DropFeatureSelection (aka: M3Drop) giniFS irlbaPcaFS (with provided PCs) corFS ("both" direction)

Genes are ranked by each method and the consensus (Cons) is calculated as the average of those ranks.

Automatically removes invariant genes. If only raw counts are provided then will apply counts per million normalization (scaled to the median library size) for those methods which require normalized data.

Value

Table of ranking of each gene by each method including the consensus (Cons). Columns are feature selection methods named using the shorter aliases (see: Details).

Examples

```
library(M3DExampleData)
norm <- as.matrix(Mmus_example_list$data[1:500,]);
norm <- norm[rowSums(norm) > 0,];
counts <- NBumiConvertToInteger(norm);
spikes <- sample(1:nrow(counts), 50);
spikes <- rownames(norm)[spikes];
spikes <- rownames(norm)
Features_consensus <- Consensus_FS(counts, norm, is.spike=spikes);
```

`Fitting_Dropout_Models` *Fit functions to the dropouts vs expression distribution.*

Description

Fits the modified Michaelis-Menten equation (MM), a logistic regession (logistic), or a double exponential (ZIFA) function to the relationship between mean expression and dropout-rate (proportion of zero values).

Usage

```
bg__fit_MM(p, s)
bg__fit_logistic(p, s)
bg__fit_ZIFA(p, s)
```

Arguments

`p`	a vector of dropout rates for each gene.
`s`	a vector of mean expression values for each gene. Must be the same order & length as p.

Details

Fits one of different models to the relationship between dropout rate and mean expression. The three models are: `bg__fit_MM` : the Michaelis-Menten function

$$P = 1 - \frac{S}{S + K}$$

(see: [1]). Fit using `mle2` using normally distributed error. `bg__fit_logistic` : a logistic regression between P and log base 10 of S (used by [2]). Fit using `glm` (excludes genes where S == 0). `bg__fit_ZIFA` : a double exponential

$$P = e^{\lambda S^2}$$

(used by [3]). Fit using `lm` after log-transformation (genes were P == 0 are assigned a value of one tenth of the smallest P which is not 0).

Value

Named list including: K,fitted_err/B0,B1/lambda,fitted_err : the fitted parameters predictions : predicted values of p for each gene SSr/SAr : sum of squared/absolute residuals model : vector of string descriptors of the fit

References

[1] Keener, J.; Sneyd, J. (2008). Mathematical Physiology: I: Cellular Physiology (2 ed.). Springer. ISBN 978-0-387-75846-6 [2] Kharchenko, PV; Silberstein, L; Scadden, DT. (2014) Bayesian approach to single-cell differential expression analysis. Nature Methods. 11:740-742 [3] Pierson, E; Yau, C. (2015) ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome Biology. 16:241 doi:10.1186/s13059-015-0805-z

Examples

```
#  library(M3DExampleData)
#  gene_info = bg__calc_variables(Mmus_example_list$data)
#  MM_fit = bg__fit_MM(gene_info$p, gene_info$s)
#  logistic_fit = bg__fit_logistic(gene_info$p, gene_info$s)
#  ZIFA_fit = bg__fit_ZIFA(gene_info$p, gene_info$s)
```

`Imputation` *Normalized Data using the DANB model*

Description

Normalizes data to a common library size, imputing zeros as needed.

Usage

```
NBumiImputeNorm(counts, fit, total_counts_per_cell=median(fit$vals$tis))
```

Arguments

`counts`	raw count matrix, rows=genes, cols=cells
`fit`	Output from `NBumiFitModel`
`total_counts_per_cell`	reference library size to normalize all cells to.

Details

Converts raw counts into positions in the CDF for the depth-adjusted negative binomial model fit to each observation. Adjusts the DANB parameters (mean and size) for the new library size. Then calculate the normalized counts for the equivalent position in the CDF for the NB using the new parameters.

Value

Normalized count matrix.

Examples

```
library(M3DExampleData)
counts <- as.matrix(Mmus_example_list$data);
counts <- counts[rowSums(counts) > 0,];
fit <- NBumiFitModel(counts);
normed_counts <- NBumiImputeNorm(counts, fit, 1000000)
```

M3DropCleanData *Filter Expression Data*

Description

Filters and normalizes a given expression matrix. Removes low quality cells and undetected genes, and normalizes counts to counts per million. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
M3DropCleanData(expr_mat, labels = NA, is.counts = TRUE, suppress.plot = FALSE, pseudo_genes = NA,
bg__filter_cells(expr_mat, labels = NA, suppress.plot = FALSE, min_detected_genes = NA)
```

Arguments

expr_mat	a numeric matrix of raw or normalized (not log-transformed) expression values, columns = samples/cells, rows = genes.
labels	a vector of length equal to the number of columns of expr_mat with names or group IDs for each cell.
is.counts	logical, whether the provided data is unnormalized read/fragment counts.
suppress.plot	logical, whether to plot the distribution of number of detected genes per cell.
pseudo_genes	a vector of gene names of known pseudogenes which will be removed from the cleaned data.
min_detected_genes	minimum number of genes/cell for a cell to be included in the cleaned data.

Details

Retains genes detected (expression>0) in more than 3 cells and with mean normalized expression >= 10^-5. If min_detected_genes is defined all cells not reaching the threshold are removed. Otherwise, fits a normal distribution to the distribution of detected genes/cell and removes those cells with significantly few detected genes (FDR 5%). This fit is plotted for visual inspection. If is.counts==TRUE then each column is converted to counts per million (ignoring ERCC spike-ins if present).

Value

A list with elements: data, the normalized filtered expression matrix; and labels, labels of the remaining cells.

Examples

```
library(M3DExampleData)
# Remove all cells with < 2000 detected genes and convert to cpm
cpm <- M3DropCleanData(Mmus_example_list$data, Mmus_example_list$labels,
 is.counts=TRUE, min_detected_genes=2000)
# Removes cells with significantly few detected genes (FDR=5%)
filtered_only <- M3DropCleanData(Mmus_example_list$data, Mmus_example_list$labels,
 is.counts=FALSE)
# QCed <- bg__filter_cells(Mmus_example_list$data[,1:10], Mmus_example_list$labels[1:10], suppress.plot=TRUE
```

`M3DropConvertData` *Convert Data to be suitable for M3Drop*

Description

Recognizes a variety of R objects/classes and extracts expression matrices from them then converts that to a normalized but not-log transformed matrix appropriate for input into M3Drop functions.

Usage

```
M3DropConvertData(input, is.log=FALSE, is.counts=FALSE, pseudocount=1)
```

Arguments

`input`	a matrix, data.frame or object
`is.log`	has the data been log-transformed? (assumes log-base 2 with pseudocount of 1)
`is.counts`	is the data raw unnormalized counts? (raw counts will be CPM normalized)
`pseudocount`	pseudocount added before log-transformation

Details

You must have loaded the respective packages (in parentheses) into your namespace before running this function on the respective objects. Note that to maintain scalability sparse matrices will remain as such.

Supported classes/objects:

SCESet (scater <= 1.4.0) uses "exprs" or if unavailable then "counts"

SingleCellExperiment (scater >= 1.6.0) uses "normcounts" if available, then "logcounts", which is assumed to be log-normalized, then "counts"

ExpressionSet (Biobase) uses "exprs", specify log/counts using arguments

seurat (Seurat) uses "raw.data" as counts.

Matrix/Dataframe classes :

dgCMatrix (Matrix) specify log/counts using arguments

data.table (data.table) specify log/counts using arguments

DataTable (S4Vectors) specify log/counts using arguments

DataFrame (S4Vectors) specify log/counts using arguments

AnnotatedDataFrame (Biobase) specify log/counts using arguments

matrix (base-r) specify log/counts using arguments

data.frame (base-r) specify log/counts using arguments

Value

A normalized but not log-transformed matrix appropriate for input into M3Drop functions.

Examples

```
# Simulated raw count matrix:
set.seed(42)
counts <- matrix(rpois(200, lambda=3), ncol=10)
expr_mat <- M3DropConvertData(counts, is.counts=TRUE)

# log normalized data frame
lognorm <-log2( t(t(counts)/colSums(counts)*100)+1 )
lognorm <- as.data.frame(lognorm)
expr_mat <- M3DropConvertData(lognorm)

# Sparse matrix
require("Matrix")
counts <- Matrix(counts, sparse=TRUE)
expr_mat <- M3DropConvertData(counts, is.counts=TRUE)

# SingleCellExperiment Object
require("SingleCellExperiment")
SCE <- SingleCellExperiment(assays=list(counts=counts))
expr_mat <- M3DropConvertData(SCE)
```

`M3DropDropoutModels` *Fit functions to the dropouts vs expression distribution.*

Description

Fits the modified Michaelis-Menten equation (MM), a logistic regession (logistic), or a double exponential (ZIFA) function to the relationship between mean expression and dropout-rate (proportion of zero values).

Usage

```
M3DropDropoutModels(expr_mat, xlim=NA, suppress.plot=FALSE)
```

Arguments

`expr_mat`	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
`xlim`	limits for x-axis of plot.
`suppress.plot`	logical, whether to plot fit curves or not.

Details

Plots the dropout-rate (P) vs average gene expression (S) for all genes. Fits three different models and adds the fitted curves to the plot. The three models are: MMfit : the Michaelis-Menten function

$$P = 1 - \frac{S}{S + K}$$

(see: [1]). LogiFit : a logistic regression between P and log base 10 of S (used by [2]). ExpoFit : a double exponential

$$P = e^{\lambda S^2}$$

(used by [3]).

Value

Invisibly, a list of output from each fit (MMfit, LogiFit, ExpoFit).

References

[1] Keener, J.; Sneyd, J. (2008). Mathematical Physiology: I: Cellular Physiology (2 ed.). Springer. ISBN 978-0-387-75846-6 [2] Kharchenko, PV; Silberstein, L; Scadden, DT. (2014) Bayesian approach to single-cell differential expression analysis. Nature Methods. 11:740-742 [3] Pierson, E; Yau, C. (2015) ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome Biology. 16:241 doi:10.1186/s13059-015-0805-z

Examples

```
library(M3DExampleData)
norm <- M3DropConvertData(Mmus_example_list$data, is.counts=TRUE)
M3DropDropoutModels(norm)
```

M3DropExpressionHeatmap

Plot Heatmap of Gene Expression

Description

Plots a customized heatmap of scaled log expression values. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
M3DropExpressionHeatmap(genes, expr_mat, cell_labels=NA, interesting_genes=NA, key_genes=genes,
bg__expression_heatmap(genes, expr_mat, cell_labels=NA, gene_labels=NA, key_genes=genes, key_cel
```

Arguments

genes	a character vector of gene names to be plot.
expr_mat	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
cell_labels	factor of labels for each cell in the expression matrix that will be used to colour the bar on the top of the heatmap.
interesting_genes	list of vectors of gene names that will be used to colour the bar to the left of the heatmap.
key_genes	a character vector of gene names to be labelled on the heatmap.
key_cells	a character vector of cells to be labelled on the heatmap. Unlabelled cells will be assigned a numerical index
gene_labels	factor of labels for each gene that will be used to colour the bar on the left of the heatmap.

Details

Modifies the gplots function heatmap.2 to replace the row dendrogram with a legend of the colours used in the columns colour bar (cell_labels) and use a custom colour scalling. Expression is displayed as Z-scores of log transformed expression (adding a pseudocount of 1) coloured blue-white-red centered at 0 and binned in the range [-2,2].

`M3DropExpressionHeatmap` is a wrapper for `bg__expression_heatmap` that checks and reformats provided arguments.

Value

Invisibly, output from heatmap.2 call.

Examples

```
 library(M3DExampleData)
M3DropExpressionHeatmap(head(rownames(Mmus_example_list$data),20),Mmus_example_list$data, cell_labels = Mm
# bg__expression_heatmap(head(rownames(Mmus_example_list$data),20),Mmus_example_list$data, cell_labels = Mm
```

M3DropFeatureSelection

Differentially Expressed Genes.

Description

Use Michaelis-Menten curve to find differentially expressed (DE) genes. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
M3DropFeatureSelection(expr_mat, mt_method="bon", mt_threshold=0.05, suppress.plot=FALSE, xlim=N
 bg__test_DE_K_equiv(gene_info, fit=NA)
```

Arguments

`expr_mat`	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
`mt_method`	the multiple testing method used in p.adjust
`mt_threshold`	the threshold for identifying significantly DE genes.
`suppress.plot`	logical, whether to plot the fitted curve and highlight selected features.
`xlim`	specify the limits of the x-axis of the plot.
`fit`	Output from fitting the Michaelis-Menten equation (see: `bg__fit_MM`
`gene_info`	List of calculated gene-specific values output by `bg__calc_variables`

Details

Fits a Michaelis-Menten function to the dropout-rate (if not provided) of the provided expression matrix. Identifies genes where the gene-specific K calculated as (S = mean expression, P = dropout rate):

$$K = \frac{S * P}{1 - P}$$

is significantly larger than the K fitted to the entire dataset. Combines standard errors of the fitted K, the gene-specific dropout rate and the gene-specific average expression using error propagation rules. Determines the signficance of the gene-specific K using a Z-test of the log-transformed Ks with the propagated error then applies the specified multiple testing correction to identify DE genes. Plots the dropout rate vs gene expression with the fitted MM curve and highlights in purple the significantly DE genes.

Value

`M3DropFeatureSelection` : a data.frame of significantly differentially expressed genes with columns: Gene, p.value, q.value `bg__test_DE_K_equiv` : a named list of containing: pval : the significance of differential expression for each gene fold_change : ratio of gene-specific K to globally fit K

Examples

```
library(M3DExampleData)
norm <- M3DropConvertData(Mmus_example_list$data, is.counts=TRUE)
DE_genes <- M3DropFeatureSelection(norm,
 mt_method="fdr", mt_threshold=0.01)
#  gene_info <- bg__calc_variables(Normalized_data$data[1:1000,])
#  DE_output <- bg__test_DE_K_equiv(gene_info)
```

`M3DropGetExtremes` *Get outliers from MM curve.*

Description

Identifies outliers left and right of a fitted Michaelis-Menten curve. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
M3DropGetExtremes(expr_mat, fdr_threshold=0.1, percent=NA, v_threshold=c(0.05,0.95), suppress.pl
bg__get_extreme_residuals(expr_mat, fit=NA, fdr_threshold=0.1, percent=NA, v_threshold=c(0.05, 0
```

Arguments

`expr_mat`	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
`fit`	output from fitting the Michaelis-Menten equation, see `bg__fit_MM`
`fdr_threshold`	the threshold for identifying significant outliers after multiple testing correction.
`percent`	identify this percentage of data that is most extreme in each direction.
`v_threshold`	restrict to this range of dropout rates to avoid poorly fit regions of the data.
`direction`	"left" or "right", which tail of outliers to examine.
`suppress.plot`	logical, whether to plot the fitted Michaelis-Menten curve and highlight to identified most extreme outliers.

Details

Fits a Michaelis-Menten function to the dropout-rate of the provided data, then identifies the most extreme left and/or right outliers from the curve. Horizontal residuals are calculated as :

$$\log_{10} S - \log_{10} \frac{K * (1 - P)}{P}$$

. Extreme left[right] outliers are identified either as the `percent` smallest[largest] horizontal residuals. If `percent` is undefined (default) a normal distribution is fitted to the horizontal residuals and a Z-test is used to identify significant outliers after FDR multiple testing correction.

Only genes with dropout rates within v_threshold will be considered to avoid the skewing of residuals due to the exponential parts of the MM curve near P = 0 & P = 1.

`M3DropGetExtremes` identifies both left and right residuals using the provided thresholds in each direction. Eg. will return the `percent` smallest and `percent` largest residuals. It also plots the fitted MM curve and highlights the left and right extreme outliers unless suppress.plot=TRUE . `bg__get_extreme_residuals` identifies one-sided extreme outliers.

Value

`M3DropGetExtremes` List containing elements left and right, vectors of the names of the extreme genes to the left and right of the curve respectively. `bg__get_extreme_residuals` A vector of names of the extreme genes in the specified direction.

Examples

```
  library(M3DExampleData)
  norm <- M3DropConvertData(Mmus_example_list$data, is.counts=TRUE)
  extreme_gene_lists <- M3DropGetExtremes(norm, fdr_threshold=0.1)
  extreme_gene_lists <- M3DropGetExtremes(norm, percent=0.01)
#  Lextremes <- bg__get_extreme_residuals(norm, fdr_threshold=0.1, direction="left")
```

M3DropGetHeatmapClusters

Extracts clusters/ordered names from heatmap output

Description

Extracts the groupings correponding to the given number of clusters from heatmap output.

Usage

```
M3DropGetHeatmapClusters(heatout, k, type="cell")
 M3DropGetHeatmapNames(heatout, type="cell")
```

Arguments

`heatout`	Output from a gene-expression heatmap.
`k`	Number of clusters.
`type`	One of "cell" or "gene" indicating whether to get clusters or names of cells(columns) or genes(rows).

Details

M3DropGetHeatmapClusters : Traverses down the row or column dendrogram and cuts at the first point where there are at least k clusters. M3DropGetHeatmapNames: gets the names of the cells/genes in the order that they appear on the dendrogram.

Value

A vector of cluster labels for each cell.

Examples

```
library(M3DExampleData)
genes <- rownames(Mmus_example_list$data)[1:20]
heatmap_out <- M3DropExpressionHeatmap(genes, Mmus_example_list$data)
clusters <- M3DropGetHeatmapClusters(heatmap_out, k=5)
heatmap_gene_labels <- M3DropGetHeatmapNames(heatmap_out, type="gene")
```

M3DropGetMarkers *Identify marker genes*

Description

Calculates area under the ROC curve for each gene to predict the best group of cells from all other cells.

Usage

```
M3DropGetMarkers(expr_mat, labels)
```

Arguments

expr_mat	a numeric matrix of normalized expression values, columns = samples, rows = genes.
labels	a vector of group ids for each cell/sample.

Details

Uses the ROCR package to calculate the AUC for each gene for the group with the highest average rank. Significant is calculated using a Wilcox rank-sum test.

Value

A dataframe with a row for each gene and columns: AUC, Group (which label this gene had the highest average rank for), and pval (uncorrected p-value of prediction).

Examples

```
library(M3DExampleData)
norm <- M3DropConvertData(Mmus_example_list$data, is.counts=TRUE)
marker_gene_table <- M3DropGetMarkers(norm, Mmus_example_list$labels)
```

M3DropPlottingFunctions *Make M3Drop Plots*

Description

Background functions used for making M3Drop plots. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
bg__dropout_plot_base(expr_mat, xlim=NA, suppress.plot=FALSE)
bg__add_model_to_plot(fitted_model, base_plot, lty=1, lwd=1, col="dodgerblue", legend_loc="toprig
bg__highlight_genes(base_plot, expr_mat, genes, col="darkorange", pch=16)
```

Arguments

`expr_mat`	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
`xlim`	limits of x-axis of plot, for comparing distributions across many datasets.
`suppress.plot`	Whether to make the plot or just return values.
`fitted_model`	output from fitting a model to dropout rate vs mean expression, see `bg__fit_MM`,`bg__fit_ZIFA`,`bg__f`
`base_plot`	output from `bg__dropout_plot_base`
`legend_loc`	coordinates of top-right corner of the legend.
`genes`	list of genes to be highlighted.
`col`, `lty`, `lwd`, `pch`	Graphical parameters passed to plotted points & lines respectively.

Details

`bg__dropout_plot_base` plots grey-scale distribution of dropout-rate vs log10(mean expression) for each each. Colour indicates density of points light-grey = low, black = high.

`bg__add_model_to_plot` adds a line based on the predicted dropout rate for each gene from a particular model.

`bg__highlight_genes` highlights specified genes in the given colour.

Value

`bg__dropout_plot_base` A named list of output: p = dropout rate for each gene s = mean expression of each gene xes = log10 transformed mean expression of each gene data = original expression matrix order = ordering of xes from smallest to largest `bg__add_model_to_plot` Invisibly location of the plotted legend. `bg__highlight_genes` Invisibly the coordinates of the highlighted genes.

M3DropTestShift *Test for horizontal shift.*

Description

Tests whether a given set of genes are significantly shifted to the left or right of the Michaelis-Menten curve.

Usage

```
M3DropTestShift(expr_mat, genes_to_test, name="", background=rownames(expr_mat), suppress.plot=F
```

Arguments

expr_mat	a numeric matrix of normalized (not log-transformed) expression values, columns = samples, rows = genes.
genes_to_test	vector of gene names to test.
name	string used to title the plot.
background	vector of gene names to test against. (default = all genes)
suppress.plot	logical, whether to the fitted Michaelis-Menten curve and highlight the given set of genes to test.

Details

Fits a Michaelis-Menten function to the dropout-rate of the provided data, then tests whether a given set of genes (eg. pseudogenes) is significantly shifted left or right of the curve. Horizontal residuals are calculated as :

$$\log_{10} S - \log_{10} \frac{K * (1 - P)}{P}$$

. Uses a Wilcox rank-sum test/Mann-Whitney U test to compare the residuals for the given genes to the residuals for all genes.

Value

A one row dataframe with columns: sample (median horizontal residual of genes in the test set), pop (median horizontal residual of genes in the background set), p.value

Examples

```
library(M3DExampleData)
gene_set <- c("Dppa2","Tdgf1","Rnf130","Tet1","Uhrf1","Pttg1","Zfp600","Stat1")
shift_output <- M3DropTestShift(Mmus_example_list$data, gene_set)
```

M3DropThreeSetVenn *Three-way Venn Diagram*

Description

Plot an area-proportional three-set Venn Diagram with labels.

Usage

```
M3DropThreeSetVenn(set1, set2, set3, names)
```

Arguments

set1	a vector of items in the first set.
set2	a vector of items in the second set.
set3	a vector of items in the third set.
names	a vector of names of each set

Details

Approximates area-proportional three-set Venn Diagram with code by David J. States (available at: http://tolstoy.newcastle.edu.au/R/help/03a/1115.html). Then places labels within each circle and overlap-region using code by Tallulah Andrews.

Value

None

Examples

```
SetA <- c(1:20)
SetB <- c(15:30)
SetC <- c(5,10,15,20,25,30,35,40,45,50,55,60)
M3DropThreeSetVenn(SetA, SetB, SetC, names=c("A","B","C"))
```

M3DropTraditionalDE *Perform Traditional Differential Expression*

Description

Performs traditional (i.e. compare defined groups) differential expression using a negative binomial model with MM zero-inflation. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Usage

```
bg__get_mean2disp(expr_mat)
bg__fitdispersion(expr_mat)
```

Arguments

expr_mat	a numeric matrix of library-size normalized expression values, columns = samples, rows = genes.

Details

THESE FUNCTIONS SHOULD NOT BE USED.

unfinished__m3dropTraditionalDE : Uses a log-likelihood ratio test to perform model selection between a model of constant mean expression vs a model of different mean expression across the biological groups. Probabilities of observing the data given the model are calculated using a zero-inflated negative binomial distribution. Global relationships between mean and dispersion (power-law) as well as mean and dropouts (Michaelis-Menten) for genes are fit independently for each batch. Dispersions are fixed for each batch and calculated using the fitted power-law using the global mean expression of each gene. Significance is evaluated using the chi-square distribution.

unfinished__m3dropTraditionalDEShiftDisp: Uses a log-likelihood ratio test to perform model selection between a model of constant mean expression vs a model of different mean expression across the biological groups. Probabilities of observing the data given the model are calculated using a zero-inflated negative binomial distribution. Global relationships between mean and dispersion (power-law) as well as mean and dropouts (Michaelis-Menten) for genes are fit independently for each batch. Dispersions are shifted from the global variance according to the mean expression for each biological group, using batch-specific power-laws. Significance is evaluated using the chi-square distribution.

broken__m3dropCTraditionalDE : Uses a log-likelihood ratio test to perform model selection between a model of constant mean expression vs a model of different mean expression across the biological groups. Probabilities of observing the data given the model are calculated using a zero-inflated negative binomial distribution. Global relationships between mean and dispersion (power-law) as well as mean and dropouts (Michaelis-Menten) for genes are fit to the full dataset. Significance is evaluated using the chi-square distribution.

bg__get_mean2disp fits a power-law relationship between the squared coefficient of variation and mean expression of each gene, which is used to calculate the expected dispersion parameter for the negative binomial distribution from a given mean expression value.

bg__fitdispersion estimates gene-specific dispersions from the mean and variance of gene expression values

$$r = \frac{\mu^2}{\sigma^2 - \mu}$$

. Then fits a power-law relationship between the estimated dispersion and mean exprssion.

Value

bg__m3dropTraditionalDE : a table of observed mean expression levels for each biological group and each batch with raw p-values and FDR corrected p-values for each gene. bg__m3dropTraditionalDEShiftDisp : a table of observed mean expression levels for each biological group and each batch with raw p-values and FDR corrected p-values for each gene. broken__m3dropCTraditionalDE : a table of observed mean expression levels for each biological group with raw p-values and FDR corrected p-values for each gene. bg__get_mean2disp : a function which calculates the expected dispersion given the mean expression. bg__fitdispersion : exponent of the power-law relationship between dispersion and mean expression.

Examples

```
library(M3DExampleData)
#Normalized_data <- M3DropCleanData(Mmus_example_list$data,
#                  labels = Mmus_example_list$labels,
#                  is.counts=TRUE, min_detected_genes=2000)
#DE_output <- bg__m3dropTraditionalDE(Normalized_data$data[1:100,], Normalized_data$labels)
#DE_shifted_output <- bg__m3dropTraditionalDEShiftDisp(Normalized_data$data[1:100,], Normalized_data$labels)
#DE_output_batches <- bg__m3dropTraditionalDE(Normalized_data$data[1:100,], Normalized_data$labels, batches=
```

NBumiCheckFit *Check Fit Quality*

Description

Checks the quality of the fit of the depth-adjusted negative binomial model.

Usage

```
NBumiCheckFit(counts, fit, suppress.plot=FALSE)
NBumiCheckFitFS(counts, fit, suppress.plot=FALSE)
```

Arguments

counts	a numeric matrix of raw UMI counts, columns = samples, rows = genes.
fit	output from NBumiFitModel or NBumiFitBasicModel.
suppress.plot	Whether to plot the observed vs expected frequency of dropouts for genes & cells.

Details

Calculates expected dropouts for genes and cells and compares to observed values. Optionally plots observed vs expected dropouts for both genes and cells. NBumiCheckFit uses depth-adjusted negative binomial with gene-specific dispersions. NBumiCheckFitFS uses depth-adjusted negative binomial with dispersions calculated from the power-law between gene-specific dispersion and mean expression (as is used for feature selection).

Value

Invisibly, named list of output: gene_error = sum of squared error between observed and expected gene-specific total dropouts cell_error = sum of squared error between observed and expected cell-specific total dropouts exp_ps = gene by sample matrix of probability of a dropout for the negative binomial model fitted to each observation.

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
fit_quality <- NBumiCheckFitFS(counts, fit);
```

NBumiCoexpression *Variance-based Feature Selection*

Description

Ranks genes by residual dispersion from mean-dispersion power-law relationship.

Usage

```
NBumiCoexpression(counts, fit, gene_list=NULL, method=c("both", "on", "off"))
```

Arguments

counts	raw count matrix (e.g. from NBumiConvertData).
fit	output from NBumiFitModel or NBumiFitBasicModel.
gene_list	set of gene names to test coexpression of.
method	type of coexpression to test (see: Details).

Details

Tests for co-expression using the normal approximation of a binomial test. Co-expression is defined according to the method argument as follows:

on two genes are both >0 in more cells than expected.

off two genes are both 0 in more cells than expected.

both two genes are either both >0 or both 0 in more cells than expected.

In all cases the null expectation is calculated using the fit library-size adjusted negative binomial model. This remove spurious co-expression due to cells with differing numbers of detected genes.

Value

a matrix of Z-scores for each pair of genes in the provided gene list.

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
genes <- c("Sox2", "Eomes", "Zscan4d", "Obox1", "Obox3")
co <- NBumiCoexpression(counts, fit, genes, method="both");
on <- NBumiCoexpression(counts, fit, genes, method="on");
off <- NBumiCoexpression(counts, fit, genes, method="off");
```

NBumiCompareModels *Compare negative binomial models*

Description

Compares the fit of the depth-adjusted negative binomial model and basic negative binomial model.

Usage

```
NBumiCompareModels(counts, size_factor=(Matrix::colSums(counts)/median(Matrix::colSums(counts))))
```

Arguments

counts	a numeric matrix of raw UMI counts, columns = samples, rows = genes.
size_factor	a calculated size factor for library size normalization.

Details

Compares the fit of the depth-adjusted negative binomial model and basic negative binomial model. Depth -adjusted negative binomial is fit to raw molecule counts. Basic negative binomial is fit to library-size normalized counts. The absolute error between observed gene-specific dropouts and expectations given each model is calculated. And a plot of fitted and observed mean-expression vs dropouts is created.

Value

A named list containing: errors : Vector of errors for each model. basic_fit : object for the basic negative binomial. adjusted_fit : object for the depth-adjusted negative binomial.

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
out <- NBumiCompareModels(counts);
out$errors
```

NBumiConvertData *Convert Data to be suitable for NBumi*

Description

Recognizes a variety of R objects/classes and extracts expression matrices from them then converts that to a count matrix for input into NBumi functions.

Usage

```
NBumiConvertData(input, is.log=FALSE, is.counts=FALSE, pseudocount=1)
```

Arguments

input	a matrix, data.frame or object
is.log	has the data been log-transformed? (assumes log-base 2 with pseudocount of 1)
is.counts	is the data raw unnormalized counts? (raw counts will be CPM normalized)
pseudocount	pseudocount added before log-transformation

Details

You must have loaded the respective packages (in parentheses) into your namespace before running this function on the respective objects. Note that to maintain scalability sparse matrices will remain as such.

Supported classes/objects:

SCESet (scater <= 1.4.0) uses "counts" or if unavailable then "exprs"

SingleCellExperiment (scater >= 1.6.0) uses "counts", if unavailable then "logcounts", which is assumed to be log-normalized.

ExpressionSet (Biobase) uses "exprs", specify log/counts using arguments

seurat (Seurat) uses "raw.data" as counts.

Matrix/Dataframe classes :

dgCMatrix (Matrix) specify log/counts using arguments

data.table (data.table) specify log/counts using arguments

DataTable (S4Vectors) specify log/counts using arguments

DataFrame (S4Vectors) specify log/counts using arguments

AnnotatedDataFrame (Biobase) specify log/counts using arguments

matrix (base-r) specify log/counts using arguments

data.frame (base-r) specify log/counts using arguments

Counts are rounded up to integers if necessary, if counts are unavailable then this will attempt to convert log2 normalized expression to counts by de-logging, subtracting the pseudocount, and then un-normalizing by rescaling cells based on their relative number of detected genes, finally expression is rounded up to integers for use as counts.

Value

A count matrix appropriate for input into NBumi functions.

Examples

```
# Simulated raw count matrix:
set.seed(42)
counts <- matrix(rpois(200, lambda=3), ncol=10)
input_counts <- NBumiConvertData(counts, is.counts=TRUE)

# log normalized data frame
lognorm <-log2( t(t(counts)/colSums(counts)*100)+1 )
lognorm <- as.data.frame(lognorm)
input_counts <- NBumiConvertData(lognorm)

# Sparse matrix
```

```
require("Matrix")
counts <- Matrix(counts, sparse=TRUE)
input_counts <- NBumiConvertData(counts, is.counts=TRUE)

# SingleCellExperiment Object
require("SingleCellExperiment")
SCE <- SingleCellExperiment(assays=list(counts=counts))
input_counts <- NBumiConvertData(SCE)
```

`NBumiConvertToInteger` *Turn a matrix of expression values into integer counts*

Description

Reformats a provided expression matrix to be compatible with NBumi modelling.

Usage

```
NBumiConvertToInteger(mat)
```

Arguments

`mat`	a numeric matrix of expression values (ideally raw UMI counts), columns = samples, rows = genes.

Details

Coerces the provided data to a matrix then rounds all values up (ceiling) to integers and removes all rows where all values are zero.

Value

Rounded, integer matrix of the original data.

Examples

```
mat <- matrix(rgamma(1000, shape=10, scale=20), ncol=10)
mat_int <- NBumiConvertToInteger(mat)
is.integer(mat_int)
```

`NBumiFeatureSelectionCombinedDrop`
Dropout-based Feature Selection

Description

Ranks genes by significance of increase in dropouts compared to expectation.

Usage

```
NBumiFeatureSelectionCombinedDrop(fit, ntop=NULL, method="fdr", qval.thresh=2, suppress.plot=TRUE
```

Arguments

fit	output from NBumiFitModel or NBumiFitBasicModel.
ntop	number of top ranked genes to return
method	correction method for multiple comparisons (check ?p.adjust.methods for more details)
qval.thresh	significant threshold
suppress.plot	logical, whether to plot the fitted curve and highlight selected features

Details

Calculates dropout probability for each observation using depth-adjusted negative binomial means and dispersions calculated from a fitted power-law relationship between mean and dispersion. Total dropouts per gene are modelled using the normal approximation of the sum of bernoulli variables. And significance is evaluated using a Z-test.

If provided, ntop will overrule the significance threshold.

Value

dataframe with columns: Gene effect_size (difference between observed and expected dropout rate) p.value q.value (corrected by adjustment method specifed by the method argument)

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
Drop_features <- names(NBumiFeatureSelectionCombinedDrop(fit, qval.thresh=0.05));
```

NBumiFeatureSelectionHighVar

Variance-based Feature Selection

Description

Ranks genes by residual dispersion from mean-dispersion power-law relationship.

Usage

```
NBumiFeatureSelectionHighVar(fit)
```

Arguments

fit	output from NBumiFitModel or NBumiFitBasicModel.

Details

Uses linear regression on log-transformed mean expression and fitted dispersions. Ranks genes by the residuals of this fit, negative = high variance, positive = low variance.

Value

Sorted vector of residuals.

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
HVGs <- names(NBumiFeatureSelectionHighVar(fit)[1:2000]);
```

NBumiFeatureSelectionOther

Other Feature Selection Methods

Description

Ineffective alternative feature selection methods based on the depth-adjusted negative binomial model. Functions tagged with "bg__" are not meant for direct usage and are not available in the Bioconductor release.

Arguments

`fit`	output from NBumiFitModel or NBumiFitBasicModel.
`window_size`	window for calculating the moving median.

Details

Calculates dropout probability for each observation using depth-adjusted negative binomial means and dispersions calculated from a fitted power-law relationship between mean and dispersion. Total dropouts per gene are modelled using the normal approximation of the sum of bernoulli variables. And significance is evaluated using a Z-test.

`obsolete__nbumiFeatureSelectionDropouts` Ranks genes by significance of increase in dropouts compared to expectation allowing for gene-specific dispersions. `obsolete__nbumiFeatureSelectionHighVarDist2Me` Ranks genes by the distance to median of log-transformed estimated dispersions.

Value

Sorted vector of p-values/distances.

Examples

```
library(M3DExampleData)
#counts <- as.matrix(Mmus_example_list$data);
#counts <- counts[rowSums(counts) > 0,];
#fit <- NBumiFitModel(counts);
#Dropout_features <- names(obsolete__nbumiFeatureSelectionDropouts(fit)[1:2000]);
#dist2med_features <- names(obsolete__nbumiFeatureSelectionHighVarDist2Med(fit)[1:2000]);
```

NBumiFitDispVsMean *Fit function between mean and dispersion*

Description

Fits a power-law relationship between mean expression and dispersion.

Usage

```
NBumiFitDispVsMean(fit, suppress.plot=TRUE)
```

Arguments

fit	output from NBumiFitModel or NBumiFitBasicModel.
suppress.plot	Whether to plot the calculated fit.

Details

Fits a power-law relationship between mean expression and fitted gene-specific dispersions, using linear regression on ln transformed values. Lowly expressed genes, mean expression < 2^4, are excluded as long as at least 2000 genes remain for fitting.

Value

Coefficients of linear regression.

Examples

```
library(M3DExampleData)
counts <- as.matrix(Mmus_example_list$data);
counts <- counts[rowSums(counts) > 0,];
fit <- NBumiFitModel(counts);
coeffs <- NBumiFitDispVsMean(fit, suppress.plot=TRUE);
```

NBumiFitModel *Fit Depth-Adjusted Negative Binomial Model*

Description

Fits means and dispersions for depth-adjusted or basic negative binomial models to a read/UMI count matrix.

Usage

```
NBumiFitModel(counts)
NBumiFitBasicModel(counts)
```

Arguments

counts	a numeric matrix of raw UMI counts, columns = samples, rows = genes.

Details

NBumiFitModel Fits a depth-adjusted negative binomial model for each expression observation. Each expression is modelled using a negative binomial distribution with mean equal to t_i*t_j/T, where t_i is the total counts for sample i, t_j is the total counts for gene j and T is the total counts. Dispersions (R size parameter) are fit such that: var_j(counts_ij-mu_ij) = sum(mu_ij+mu_ij^2/size_j). Cases where genes exhibit poissonian behavior (size->infinity) are assigned a size of 10^10. NBumiFitBasicModel Fits a basic negative binomial model for each expression observation. Each expression is modelled using a negative binomial distribution with mean equal to t_j/n, where t_j is the total counts for gene j and n is the number of cells. Dispersions (R size parameter) are fit such that: var_j(counts_ij) = sum(mu_j+mu_j^2/size_j). Cases where genes exhibit poissonian behavior (size->infinity) are assigned a size of 10^10.

Value

A named list of: mus = a genes by samples matrix of means for the observation-specific negative binomial sizes = a vector of dispersions for each gene. vals = named list of summary statistics of the original count matrix: tis = total molecules per cell, tjs = total molecules per gene, dis = total dropouts per cell, djs = total dropouts per genes, total = total molecules in dataset, nc = number of cells, ng = number of genes

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
```

NBumiHVG *Variance-based Feature Selection Accounting for Library Size and Sample Variance*

Description

Tests for significantly high variability in droplet-based datasets.

Usage

```
NBumiHVG(counts, fit, fdr_thresh=0.05, suppress.plot=FALSE, method=c("DANB", "basic"))
```

Arguments

counts	raw count matrix (e.g. from NBumiConvertData).
fit	output from NBumiFitModel or NBumiFitBasicModel.
fdr_thresh	multiple testing correction threshold to apply to filter output.
suppress.plot	whether to plot mean vs variance & selected features.
method	whether to use DANB dispersions or raw sample variances.

Details

Assumes a constant dispersion parameter due to technical noise (see: [1]), which is estimated using linear regression. Gene-specific observed sample variances are compared to their respective expected variances. Significance is evaluated using a Z-test with the expected variance of the sample variance for a Negative Binomial (see: [2]).

The method argument controls whether the expected and observed variances are adjusted to account for uneven library sizes between cells. The default "DANB" option does the adjustment.

Value

a data.frame with columns: Gene, effect.size, p.value, q.value

References

[1] Svensson, V. (2019) Droplet scRNA-seq is not zero-inflated. bioRxiv. doi: https://doi.org/10.1101/582064
[2] Rose, C. and Smith, M. D. Mathematical Statistics with Mathematica. New York: Springer-Verlag, 2002. p264

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
HVGs <- NBumiHVG(counts, fit, suppress.plot=TRUE);
HVGs_uncorrected <- NBumiHVG(counts, fit, suppress.plot=TRUE, method="basic");
```

NBumiPearsonResiduals *Calculate Pearson Residuals*

Description

Uses the NBumi depth-adjusted negative binomial model to calculate Pearson Residuals as an approach to normalizing 10X/UMI tagged data.

Usage

```
NBumiPearsonResiduals(counts, fits=NULL)
NBumiPearsonResidualsApprox(counts, fits=NULL)
```

Arguments

counts	a numeric matrix of raw UMI counts, columns = samples/cells, rows = genes.
fits	the output from NBumiFitModel.

Details

Calculates a unique expectation for each gene expression observation based on the depth-adjusted negative binomial model (see: NBumiFitModel for details). This expection (mu) is equal to t_i*t_j/T, where t_i is the total counts for sample i, t_j is the total counts for gene j and T is the total counts.

`NBumiPearsonResidualApprox` Pearson residuals are approximated as: (counts - expectation) / sqrt(expectation). This assumes a Poisson-distribution of counts.

`NBumiPearsonResiduals` Pearson residuals are approximated as: (counts - mu) / sqrt(mu + mu^2/size). This uses a negative-binomial distribution of counts.

Value

a matrix of pearson residuals of equal dimension as your original matrix.

See Also

NBumiFitModel

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
fit <- NBumiFitModel(counts);
pearson1 <- NBumiPearsonResiduals(counts,fit)
pearson2 <- NBumiPearsonResidualsApprox(counts,fit)
```

`Other Feature Selection Methods`

Other Feature Selection Methods

Description

Other feature selection methods which only rank genes.

Usage

```
irlbaPcaFS(expr_mat, pcs=c(2,3))
giniFS(expr_mat, suppress.plot=TRUE)
corFS(expr_mat, dir=c("both", "pos", "neg"), fdr=NULL)
```

Arguments

`expr_mat`	normalized but not log-transformed gene expression matrix, rows=genes, cols=cells.
`pcs`	which principle components to use to score genes.
`suppress.plot`	whether to plot the gene expression vs Gini score.
`dir`	direction of correlation to consider.
`fdr`	apply an fdr threshold for significant features.

Details

`irlbaPcaFS` Features are ranked by the sum of the magnitude of the loadings for the specified principle components. PCA is performed using the irlba package using sparse matricies for speed. `giniFS` Fits a loess curve between the maximum expression value and gini-index of each gene. Genes are ranked by p-value from a normal distribution fit to the residuals of the curve. As proposed by GiniClust [1]. `corFS` Calculates all gene-gene correlations then ranks genes by the magnitude of the most positive or negative correlation. "both" will rank by the average of the magnitudes of the most positive & negative correlation.

Value

Sorted vector of scores for each gene from best features to worst features.

Examples

```
library(M3DExampleData)
norm <- M3DropConvertData(Mmus_example_list$data[1:500,]);
Features_gini <- giniFS(norm, suppress.plot=FALSE);
Features_cor <- corFS(norm);
Features_pca <- irlbaPcaFS(norm);
```

PoissonUMIFeatureSelectionDropouts

Dropout-based Feature Selection

Description

Ranks genes by significance of increase in dropouts compared to expectation.

Usage

```
PoissonUMIFeatureSelectionDropouts(fit)
```

Arguments

`fit`	output from NBumiFitModel or NBumiFitBasicModel.

Details

Calculates dropout probability for each observation using depth-adjusted negative binomial means and dispersions equal to the mean (Poisson). Total dropouts per gene are modelled using the normal approximation of the sum of bernoulli variables. And significance is evaluated using a Z-test.

Value

Sorted vector of p-values

Examples

```
library(M3DExampleData)
counts <- as.matrix(Mmus_example_list$data);
counts <- counts[rowSums(counts) > 0,];
fit <- NBumiFitModel(counts);
Dropout_features <- names(PoissonUMIFeatureSelectionDropouts(fit)[1:2000]);
```

`Simulation Trifecta` *Make Simulated Data from a provided scRNASeq dataset.*

Description

Fits either a zero-inflated negative binomial model (M3Drop) or the depth-adjusted negative binomial model (NBumi) to a provided dataset then simulates data from that model including differentially expressed (DE), differentially variable (DV), or globally unusually variable (HV) genes.

Usage

```
M3DropSimulationTrifecta(original_data, n_genes=25000, n_cells=250, sub_pop_prop=0.5)
NBumiSimulationTrifecta(original_data, n_genes=25000, n_cells=250, sub_pop_prop=0.5)
```

Arguments

`original_data`	the expression matrix of a scRNASeq dataset to base the simulations on. Should be normalized (not log-transformed) values for M3Drop or raw counts for NBumi.
`n_genes`	number of genes to simulated for each expression matrix.
`n_cells`	number of cells to simulate for each condition (total columns of final matrices = 2*n_cells).
`sub_pop_prop`	proportion of cells in one of the sub-populations.

Details

Generates simulated single-cell gene expression data based on an existing dataset. Three expression matrices are produced each with the same cell and gene-specific parameters but where the log2-fold change is applied to either the mean (DE), or variance (DV) in one half of the cells or is applied to the variance across all the cells (HV).

Mean expression for each simulated gene is drawn from a log-normal distribution fit to the original dataset. These means are then bottom thresholded to ensure all genes have a mean expression >= 10^-5, and top thresholded to ensure no gene has a mean expression greater than the largest mean expression in the original dataset.

`M3DropSimulationTrifecta`

`NBumiSimulationTrifecta` : Cell-specific library sizes are drawn from a gamma distribution fit to the original data.

Value

a named list of output including: truth - the true log (base 2) fold changes in expression level or variability for each gene. groups - a vector specifying the group ID for each cell for the DE and DV genes (1 = control, 2 = different). de - the count matrix containing genes differentially expressed across the two groups. dv - the count matrix containing genes with differential variability across the two groups. hv - the count matrix containing genes with globally unusual variability.

Examples

```
library(M3DExampleData)
counts <- NBumiConvertData(Mmus_example_list$data)
norm <- M3DropConvertData(Mmus_example_list$data, is.log=FALSE, is.counts=TRUE)
ZINB_sim <- M3DropSimulationTrifecta(norm)
DANB_sim <- NBumiSimulationTrifecta(counts)
```

Index