Package ‘scde’

May 1, 2024

Type Package
Title Single Cell Differential Expression
Version 2.32.0
Description The scde package implements a set of statistical methods for analyzing single-cell RNA-seq data. scde fits individual error models for single-cell RNA-seq measurements. These models can then be used for assessment of differential expression between groups of cells, as well as other types of analysis. The scde package also contains the pagoda framework which applies pathway and gene set overdispersion analysis to identify and characterize putative cell subpopulations based on transcriptional signatures. The overall approach to the differential expression analysis is detailed in the following publication: "Bayesian approach to single-cell differential expression analysis" (Kharchenko PV, Silberstein L, Scadden DT, Nature Methods, doi: 10.1038/nmeth.2967). The overall approach to subpopulation identification and characterization is detailed in the following pre-print: "Characterizing transcriptional heterogeneity through pathway and gene set overdispersion analysis" (Fan J, Salathia N, Liu R, Kaeser G, Yung Y, Herman J, Kaper F, Fan JB, Zhang K, Chun J, and Kharchenko PV, Nature Methods, doi:10.1038/nmeth.3734).

Author Peter Kharchenko [aut, cre], Jean Fan [aut], Evan Biederstedt [aut]
Maintainer Evan Biederstedt <evan.biederstedt@gmail.com>
URL http://pklab.med.harvard.edu/scde
BugReports https://github.com/hms-dbmi/scde/issues
License GPL-2
LazyData true
Depends R (>= 3.0.0), flexmix
Imports Rcpp (>= 0.10.4), RcppArmadillo (>= 0.5.400.2.0), mgcv, Rook, rjson, MASS, Cairo, RColorBrewer, edgeR, quantreg, methods, nnet, RMTstat, extRemes, pcaMethods, BiocParallel, parallel
Suggests knitr, cba, fastcluster, WGCNA, GO.db, org.Hs.eg.db, rmarkdown
biocViews ImmunoOncology, RNASeq, StatisticalMethod, DifferentialExpression, Bayesian, Transcription, Software
LinkingTo Rcpp, RcppArmadillo
VignetteBuilder knitr
RoxygenNote 5.0.0
NeedsCompilation yes
git_url https://git.bioconductor.org/packages/scde
git_branch RELEASE_3_19
git_last_commit 2d8db8a
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-04-30

Contents

bwpca .................................................. 3
clean.counts .......................................... 4
clean.gos ............................................. 5
es.mef.small ......................................... 5
knn ...................................................... 6
knn.error.models .................................... 6
make.pagoda.app .................................... 8
o.ifm .................................................. 9
pagoda.cluster.cells ................................. 9
pagoda.effective.cells .............................. 10
pagoda.gene.clusters ................................ 11
pagoda.pathway.wPCA ................................ 12
pagoda.reduce.loading.redundancy ............... 14
pagoda.reduce.redundancy .......................... 15
pagoda.show.pathways ............................... 16
pagoda.subtract.aspect ............................. 17
pagoda.top.aspects ................................. 18
pagoda.varnorm ...................................... 19
pagoda.view.aspects .............................. 21
papply ................................................. 22
pollen .................................................. 22
scde .................................................. 23
scde.browse.diffexp ................................ 23
scde.edff ............................................ 24
scde.error.models ................................... 25
scde.expression.difference ....................... 26
scde.expression.magnitude ....................... 28
scde.expression.prior .............................. 29
scde.failure.probability ......................... 30
scde.fit.models.to.reference .................... 31
scde.posteriors .................................. 32
scde.test.gene.expression.difference .......... 33
bw pca

Determine principal components of a matrix using per-observation/per-variable weights

Description

Implements a weighted PCA

Usage

bw pca(mat, matw = NULL, npcs = 2, nstarts = 1, smooth = 0,
       em.tol = 1e-06, em.maxiter = 25, seed = 1, center = TRUE,
       n.shuffles = 0)

Arguments

mat matrix of variables (columns) and observations (rows)
matw corresponding weights
npcs number of principal components to extract
nstarts number of random starts to use
smooth smoothing span
em.tol desired EM algorithm tolerance
em.maxiter maximum number of EM iterations
seed random seed
center whether mat should be centered (weighted centering)
n.shuffles optional number of per-observation randomizations that should be performed in addition to the main calculations to determine the lambda1 (PC1 eigenvalue) magnitude under such randomizations (returned in $randvar)

Value

a list containing eigenvector matrix ($rotation), projections ($scores), variance (weighted) explained by each component ($var), total (weighted) variance of the dataset ($totalvar)
Examples

```r
set.seed(0)
mat <- matrix(c(rnorm(5*10,mean=0,sd=1), rnorm(5*10,mean=5,sd=1)), 10, 10) # random matrix
base.pca <- bwpcal(mat) # non-weighted pca, equal weights set automatically
matw <- matrix(c(rnorm(5*10,mean=0,sd=1), rnorm(5*10,mean=5,sd=1)), 10, 10) # random weight matrix
matw <- abs(matw)/max(matw)
base.pca.weighted <- bwpcal(mat, matw) # weighted pca
```

---

**clean.counts**

**Filter counts matrix**

**Description**

Filter counts matrix based on gene and cell requirements

**Usage**

```r
clean.counts(counts, min.lib.size = 1800, min.reads = 10, min.detected = 5)
```

**Arguments**

- **counts** read count matrix. The rows correspond to genes, columns correspond to individual cells
- **min.lib.size** Minimum number of genes detected in a cell. Cells with fewer genes will be removed (default: 1.8e3)
- **min.reads** Minimum number of reads per gene. Genes with fewer reads will be removed (default: 10)
- **min.detected** Minimum number of cells a gene must be seen in. Genes not seen in a sufficient number of cells will be removed (default: 5)

**Value**

a filtered read count matrix

**Examples**

```r
data(pollen)
dim(pollen)
cd <- clean.counts(pollen)
dim(cd)
```
**clean.gos**  
*Filter GOs list*

**Description**

Filter GOs list and append GO names when appropriate

**Usage**

```r
clean.gos(go.env, min.size = 5, max.size = 5000, annot = FALSE)
```

**Arguments**

- `go.env`  
  GO or gene set list
- `min.size`  
  Minimum size for number of genes in a gene set (default: 5)
- `max.size`  
  Maximum size for number of genes in a gene set (default: 5000)
- `annot`  
  Whether to append GO annotations for easier interpretation (default: FALSE)

**Value**

a filtered GO list

**Examples**

```r
# 10 sample GOs
library(org.Hs.eg.db)
go.env <- mget(ls(org.Hs.egGO2ALLEGS)[1:10], org.Hs.egGO2ALLEGS)
# Filter this list and append names for easier interpretation
go.env <- clean.gos(go.env)
```

---

**es.mef.small**  
*Sample data*

**Description**

A subset of Saiful et al. 2011 dataset containing first 20 ES and 20 MEF cells.

**References**

knn.error.models

Sample error model

Description
SCDE error model generated from the Pollen et al. 2014 dataset.

References
www.ncbi.nlm.nih.gov/pubmed/25086649

knn.error.models
Build error models for heterogeneous cell populations, based on K-nearest neighbor cells.

Description
 Builds cell-specific error models assuming that there are multiple subpopulations present among the measured cells. The models for each cell are based on average expression estimates obtained from K closest cells within a given group (if groups = NULL, then within the entire set of measured cells). The method implements fitting of both the original log-fit models (when linear.fit = FALSE), or newer linear-fit models (linear.fit = TRUE, default) with locally fit overdispersion coefficient (local.theta.fit = TRUE, default).

Usage
knn.error.models(counts, groups = NULL, k = round(ncol(counts)/2),
                 min.nonfailed = 5, min.count.threshold = 1, save.model.plots = TRUE,
                 max.model.plots = 50, n.cores = parallel::detectCores(),
                 min.size.entries = 2000, min.fpm = 0, cor.method = "pearson",
                 verbose = 0, fpm.estimate.trim = 0.25, linear.fit = TRUE,
                 local.theta.fit = linear.fit, theta.fit.range = c(0.01, 100),
                 alpha.weight.power = 1/2)

Arguments

  counts         count matrix (integer matrix, rows- genes, columns- cells)
  groups         optional groups partitioning known subpopulations
  k              number of nearest neighbor cells to use during fitting. If k is set sufficiently high, all of the cells within a given group will be used.
  min.nonfailed  minimum number of non-failed measurements (within the k nearest neighbor cells) required for a gene to be taken into account during error fitting procedure
  min.count.threshold  minimum number of reads required for a measurement to be considered non-failed
knn.error.models

save.model.plots
whether model plots should be saved (file names are (group).models.pdf, or cell.models.pdf if no group was supplied)

max.model.plots
maximum number of models to save plots for (saves time when there are too many cells)

n.cores
number of cores to use through the calculations

min.size.entries
minimum number of genes to use for model fitting

min.fpm
optional parameter to restrict model fitting to genes with group-average expression magnitude above a given value

cor.method
correlation measure to be used in determining k nearest cells

verbose
level of verbosity

fpm.estimate.trim
fraction to be used in estimating group-average gene expression magnitude for model fitting (0.5 would be median, 0 would turn off trimming)

linear.fit
whether newer linear model fit with zero intercept should be used (T), or the log-fit model published originally (F)

local.theta.fit
whether local theta fitting should be used (only available for the linear fit models)

theta.fit.range
allowed range of the theta values

alpha.weight.power
1/theta weight power used in fitting theta dependency on the expression magnitude

Value

a data frame with parameters of the fit error models (rows- cells, columns- fitted parameters)

Examples

data(pollen)

kn <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
make.pagoda.app

Make the PAGODA app

Description

Create an interactive user interface to explore output of PAGODA.

Usage

make.pagoda.app(tamr, tam, varinfo, env, pwpca, clpca = NULL,
col.cols = NULL, cell.clustering = NULL, row.clustering = NULL,
title = "pathway clustering", zlim = c(-1, 1) * quantile(tamr$xv, p = 0.95))

Arguments

tamr
  Combined pathways that show similar expression patterns. Output of pagoda.reduce.redundancy

tam
  Combined pathways that are driven by the same gene sets. Output of pagoda.reduce.loading.redundancy

varinfo
  Variance information. Output of pagoda.varnorm

env
  Gene sets as an environment variable.
pwpca
  Weighted PC magnitudes for each gene set provided in the env. Output of pagoda.pathway.wPCA

clpca
  Weighted PC magnitudes for de novo gene sets identified by clustering on expression. Output of pagoda.gene.clusters

col.cols
  Matrix of column colors. Useful for visualizing cell annotations such as batch labels. Default NULL.
cell.clustering
  Dendrogram of cell clustering. Output of pagoda.cluster.cells. Default NULL.
row.clustering
  Dendrogram of combined pathways clustering. Default NULL.
title
  Title text to be used in the browser label for the app. Default, set as ‘pathway clustering’
zlim
  Range of the normalized gene expression levels, inputted as a list: c(lower_bound, upper_bound). Values outside this range will be Winsorized. Useful for increasing the contrast of the heatmap visualizations. Default, set to the 5th and 95th percentiles.

Value

PAGODA app
Sample error model

Description

SCDE error model generated from a subset of Saiful et al. 2011 dataset containing first 20 ES and 20 MEF cells.

References


determine optimal cell clustering based on the genes driving the significant aspects

Description

Determines cell clustering (hclust result) based on a weighted correlation of genes underlying the top aspects of transcriptional heterogeneity. Branch orientation is optimized if 'cba' package is installed.

Usage

pagoda.cluster.cells(tam, varinfo, method = "ward.D", include.aspects = FALSE, verbose = 0, return.details = FALSE)

Arguments

tam result of pagoda.top.aspects() call
varinfo result of pagoda.varnorm() call
method clustering method ('ward.D' by default)
include.aspects whether the aspect patterns themselves should be included alongside with the individual genes in calculating cell distance
verbose 0 or 1 depending on level of desired verbosity
return.details Boolean of whether to return just the hclust result or a list containing the hclust result plus the distance matrix and gene values

Value

hclust result
Examples

```r
data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pwpca <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
tam <- pagoda.top.aspects(pwpca, return.table = TRUE, plot=FALSE, z.score=1.96) # top aspects based on GO only
hc <- pagoda.cluster.cells(tam, varinfo)
plot(hc)
```

pagoda.effective.cells

Estimate effective number of cells based on lambda1 of random gene sets

Description

Examines the dependency between the amount of variance explained by the first principal component of a gene set and the number of genes in a gene set to determine the effective number of cells for the Tracy-Widom distribution

Usage

```r
pagoda.effective.cells(pwpca, start = NULL)
```

Arguments

- `pwpca`: result of the pagoda.pathway.wPCA() call with n.randomizations > 1
- `start`: optional starting value for the optimization (if the NLS breaks, trying high starting values usually fixed the local gradient problem)

Value

effective number of cells

Examples

```r
data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pwpca <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
pagoda.effective.cells(pwpca)
```
pagoda.gene.clusters

Determine de-novo gene clusters and associated overdispersion info

Description

Determine de-novo gene clusters, their weighted PCA lambda1 values, and random matrix expectation.

Usage

pagoda.gene.clusters(varinfo, trim = 3.1/ncol(varinfo$mat),
    n.clusters = 150, n.samples = 60, cor.method = "p",
    n.internal.shuffles = 0, n.starts = 10, n.cores = detectCores(),
    verbose = 0, plot = FALSE, show.random = FALSE, n.components = 1,
    method = "ward.D", secondary.correlation = FALSE,
    n.cells = ncol(varinfo$mat), old.results = NULL)

Arguments

varinfo  varinfo adjusted variance info from pagoda.varinfo() (or pagoda.subtract.aspect())
trim  additional Winsorization trim value to be used in determining clusters (to remove clusters that group outliers occurring in a given cell). Use higher values (5-15) if the resulting clusters group outlier patterns
n.clusters  number of clusters to be determined (recommended range is 100-200)
n.samples  number of randomly generated matrix samples to test the background distribution of lambda1 on
cor.method  correlation method ("pearson", "spearman") to be used as a distance measure for clustering
n.internal.shuffles  number of internal shuffles to perform (only if interested in set coherence, which is quite high for clusters by definition, disabled by default; set to 10-30 shuffles to estimate)
n.starts  number of wPCA EM algorithm starts at each iteration
n.cores  number of cores to use
verbose  verbosity level
plot  whether a plot showing distribution of random lambda1 values should be shown (along with the extreme value distribution fit)
show.random  whether the empirical random gene set values should be shown in addition to the Tracy-Widom analytical approximation
n.components  number of PC to calculate (can be increased if the number of clusters is small and some contain strong secondary patterns - rarely the case)
method  clustering method to be used in determining gene clusters
secondary.correlation
whether clustering should be performed on the correlation of the correlation matrix instead

n.cells
number of cells to use for the randomly generated cluster lambda1 model

old.results
optionally, pass old results just to plot the model without recalculating the stats

Value
a list containing the following fields:

• clusters a list of genes in each cluster values
• xf extreme value distribution fit for the standardized lambda1 of a randomly generated pattern
• tci index of a top cluster in each random iteration
• cl.goc weighted PCA info for each real gene cluster
• varm standardized lambda1 values for each randomly generated matrix cluster
• clvlm a linear model describing dependency of the cluster lambda1 on a Tracy-Widom lambda1 expectation

Examples

data(pollen)
cd <- clean.counts(pollen)
knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
clpca <- pagoda.gene.clusters(varinfo, trim=7.1/ncol(varinfo$mat), n.clusters=150, n.cores=10, plot=FALSE)
Arguments

varinfo: adjusted variance info from pagoda.varinfo() or pagoda.subtract.aspect()

setenv: environment listing gene sets (contains variables with names corresponding to gene set name, and values being vectors of gene names within each gene set)

n.components: number of principal components to determine for each gene set

n.cores: number of cores to use

min.pathway.size: minimum number of observed genes that should be contained in a valid gene set

max.pathway.size: maximum number of observed genes in a valid gene set

n.randomizations: number of random gene sets (of the same size) to be evaluated in parallel with each gene set (can be kept at 5 or 10, but should be increased to 50-100 if the significance of pathway overdispersion will be determined relative to random gene set models)

n.internal.shuffles: number of internal (independent row shuffles) randomizations of expression data that should be evaluated for each gene set (needed only if one is interested in gene set coherence P values, disabled by default; set to 10-30 to estimate)

n.starts: number of random starts for the EM method in each evaluation

center: whether the expression matrix should be recentered

batch.center: whether batch-specific centering should be used

proper.gene.names: alternative vector of gene names (replacing rownames(varinfo$mat)) to be used in cases when the provided setenv uses different gene names

verbose: verbosity level

Value

a list of weighted PCA info for each valid gene set

Examples

data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)

varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)

# create go environment
library(org.Hs.eg.db)

# translate gene names to ids
ids <- unlist(lapply(mget(rownames(cd), org.Hs.egALIAS2EG, ifnotfound = NA), function(x) x[1]))

rids <- names(ids); names(rids) <- ids

go.env <- lapply(mget(ls(org.Hs.egGO2ALLEGS), org.Hs.egGO2ALLEGS), function(x) as.character(na.omit(rids[x])))

# clean GOs

go.env <- clean.gos(go.env)

# convert to an environment
go.env <- list2env(go.env)
pwpca <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)

pagoda.reduce.loading.redundancy

*Collapse aspects driven by the same combinations of genes*

**Description**

Examines PC loading vectors underlying the identified aspects and clusters aspects based on a product of loading and score correlation (raised to corr.power). Clusters of aspects driven by the same genes are determined based on the distance.threshold and collapsed.

**Usage**

```r
pagoda.reduce.loading.redundancy(tam, pwpca, clpca = NULL, plot = FALSE,
    cluster.method = "complete", distance.threshold = 0.01, corr.power = 4,
    n.cores = detectCores(), abs = TRUE, ...)
```

**Arguments**

- `tam` output of pagoda.top.aspects()
- `pwpca` output of pagoda.pathway.wPCA()
- `clpca` output of pagoda.gene.clusters() (optional)
- `plot` whether to plot the resulting clustering
- `cluster.method` one of the standard clustering methods to be used (fastcluster::hclust is used if available or stats::hclust)
- `distance.threshold` similarity threshold for grouping interdependent aspects
- `corr.power` power to which the product of loading and score correlation is raised
- `n.cores` number of cores to use during processing
- `abs` Boolean of whether to use absolute correlation
- `...` additional arguments are passed to the pagoda.view.aspects() method during plotting

**Value**

A list structure analogous to that returned by pagoda.top.aspects(), but with addition of a $cnam element containing a list of aspects summarized by each row of the new (reduced) $xv and $xvw
Examples

```r
data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pw pca <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
tam <- pagoda.top.aspect s(pw pca, return.table = TRUE, plot=FALSE, z.score=1.96) # top aspects based on GO only
tamr <- pagoda.reduce.loading.redundancy(tam, pw pca)
```

pagoda.reduce.redundancy

*Collapse aspects driven by similar patterns (i.e. separate the same sets of cells)*

Description

Examines PC loading vectors underlying the identified aspects and clusters aspects based on score correlation. Clusters of aspects driven by the same patterns are determined based on the `distance.threshold`.

Usage

```r
pagoda.reduce.redundancy(tamr, distance.threshold = 0.2,
cluster.method = "complete", distance = NULL,
weighted.correlation = TRUE, plot = FALSE, top = Inf, trim = 0,
abs = FALSE, ...)
```

Arguments

- **tamr**: output of `pagoda.reduce.loading.redundancy()`
- **distance.threshold**: similarity threshold for grouping interdependent aspects
- **cluster.method**: one of the standard clustering methods to be used (`fastcluster::hclust` is used if available or `stats::hclust`)
- **distance**: distance matrix
- **weighted.correlation**: Boolean of whether to use a weighted correlation in determining the similarity of patterns
- **plot**: Boolean of whether to show plot
- **top**: Restrict output to the top n aspects of heterogeneity
- **trim**: Winsorization trim to use prior to determining the top aspects
- **abs**: Boolean of whether to use absolute correlation
- **...**: additional arguments are passed to the `pagoda.view.aspects()` method during plotting
pagoda.show.pathways

Value

A list structure analogous to that returned by pagoda.top.aspects(), but with addition of a $cnam
element containing a list of aspects summarized by each row of the new (reduced) $xv and $xvw.

Examples

data(pollen)
cd <- clean.counts(pollen)
knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pwpc <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
tam <- pagoda.top.aspects(pwpc, return.table = TRUE, plot=FALSE, z.score=1.96) # top aspects based on GO only
tamr <- pagoda.reduce.loading.redundancy(tam, pwpc)
tamr2 <- pagoda.reduce.redundancy(tamr, distance.threshold = 0.9, plot = TRUE, labRow = NA, labCol = NA, box = TRUE, ...)
pagoda.subtract.aspect

showRowLabels controls whether row labels are shown in the plot
cexCol column label size (cex)
cexRow row label size (cex)
nstarts number of random starts for the wPCA
cell.clustering cell clustering
show.cell.dendrogram whether cell dendrogram should be shown
plot whether the plot should be shown
box whether to draw a box around the plotted matrix
trim optional Winsorization trim that should be applied
return.details whether the function should return the matrix as well as full PCA info instead of just PC1 vector
... additional arguments are passed to the c.view.pathways

Value
cell scores along the first principal component of shown genes (returned as invisible)

pagoda.subtract.aspect

Control for a particular aspect of expression heterogeneity in a given population

Description

Similar to subtracting n-th principal component, the current procedure determines (weighted) projection of the expression matrix onto a specified aspect (some pattern across cells, for instance sequencing depth, or PC corresponding to an undesired process such as ribosomal pathway variation) and subtracts it from the data so that it is controlled for in the subsequent weighted PCA analysis.

Usage

pagoda.subtract.aspect(varinfo, aspect, center = TRUE)

Arguments

varinfo normalized variance info (from pagoda.varnorm())
aspect a vector giving a cell-to-cell variation pattern that should be controlled for (length should be corresponding to ncol(varinfo$mat))
center whether the matrix should be re-centered following pattern subtraction
pagoda.top.aspects

**Value**

a modified varinfo object with adjusted expression matrix (varinfo$mat)

**Examples**

```r
data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)

# create go environment
library(org.Hs.eg.db)
# translate gene names to ids
ids <- unlist(lapply(mget(rownames(cd), org.Hs.egALIAS2EG, ifnotfound = NA), function(x) x[1]))
rids <- names(ids); names(rids) <- ids
go.env <- lapply(mget(ls(org.Hs.egGO2ALLEGS), org.Hs.egGO2ALLEGS), function(x) as.character(na.omit(rids[x])))
# clean GOs
go.env <- clean.gos(go.env)
# convert to an environment
go.env <- list2env(go.env)
# subtract the pattern
cc.pattern <- pagoda.show.pathways(ls(go.env)[1:2], varinfo, go.env, show.cell.dendrogram = TRUE, showRowLabels = TRUE) # Look at pattern from 2 GO annotations
varinfo.cc <- pagoda.subtract.aspect(varinfo, cc.pattern)
```

---

**pagoda.top.aspects**

Score statistical significance of gene set and cluster overdispersion

**Description**

Evaluates statistical significance of the gene set and cluster lambda1 values, returning either a text table of Z scores, etc, a structure containing normalized values of significant aspects, or a set of genes underlying the significant aspects.

**Usage**

```r
pagoda.top.aspects(pwpca, clpca = NULL, n.cells = NULL,
                   z.score = qnorm(0.05/2, lower.tail = FALSE), return.table = FALSE,
                   return.genes = FALSE, plot = FALSE, adjust.scores = TRUE,
                   score.alpha = 0.05, use.oe.scale = FALSE, effective.cells.start = NULL)
```

**Arguments**

- `pwpca` output of pagoda.pathway.wPCA()
- `clpca` output of pagoda.gene.clusters() (optional)
- `n.cells` effective number of cells (if not provided, will be determined using pagoda.effective.cells())
pagoda.varnorm

**z.score**
Z score to be used as a cutoff for statistically significant patterns (defaults to 0.05 P-value)

**return.table**
whether a text table showing

**return.genes**
whether a set of genes driving significant aspects should be returned

**plot**
whether to plot the cv/n vs. dataset size scatter showing significance models

**adjust.scores**
whether the normalization of the aspect patterns should be based on the adjusted Z scores - qnorm(0.05/2, lower.tail = FALSE)

**score.alpha**
significance level of the confidence interval for determining upper/lower bounds

**use.oe.scale**
whether the variance of the returned aspect patterns should be normalized using observed/expected value instead of the default chi-squared derived variance corresponding to overdispersion Z score

**effective.cells.start**
starting value for the pagoda.effective.cells() call

**Value**
if return.table = FALSE and return.genes = FALSE (default) returns a list structure containing the following items:

- xv a matrix of normalized aspect patterns (rows- significant aspects, columns- cells)
- xvw corresponding weight matrix
- gw set of genes driving the significant aspects
- df text table with the significance testing results

**Examples**

```r
data(pollen)
cd <- clean.counts(pollen)
knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pw pca <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
tam <- pagoda.top.aspects(pw pca, return.table = TRUE, plot=FALSE, z.score=1.96) # top aspects based on GO only
```

---

**pagoda.varnorm**

*Normalize gene expression variance relative to transcriptome-wide expectations*

**Description**

Normalizes gene expression magnitudes to ensure that the variance follows chi-squared statistics with respect to its ratio to the transcriptome-wide expectation as determined by local regression on expression magnitude (and optionally gene length). Corrects for batch effects.
pagoda.varnorm(models, counts, batch = NULL, trim = 0, prior = NULL, fit.genes = NULL, plot = TRUE, minimize.underdispersion = FALSE, n.cores = detectCores(), n.randomizations = 100, weight.k = 0.9, verbose = 0, weight.df.power = 1, smooth.df = -1, max.adj.var = 10, theta.range = c(0.01, 100), gene.length = NULL)

Arguments

models: model matrix (select a subset of rows to normalize variance within a subset of cells)
counts: read count matrix
batch: measurement batch (optional)
trim: trim value for Winsorization (optional, can be set to 1-3 to reduce the impact of outliers, can be as large as 5 or 10 for datasets with several thousand cells)
prior: expression magnitude prior
fit.genes: a vector of gene names which should be used to establish the variance fit (default is NULL: use all genes). This can be used to specify, for instance, a set spike-in control transcripts such as ERCC.
plot: whether to plot the results
minimize.underdispersion: whether underdispersion should be minimized (can increase sensitivity in datasets with high complexity of population, however cannot be effectively used in datasets where multiple batches are present)
n.cores: number of cores to use
n.randomizations: number of bootstrap sampling rounds to use in estimating average expression magnitude for each gene within the given set of cells
weight.k: k value to use in the final weight matrix
verbose: verbosity level
weight.df.power: power factor to use in determining effective number of degrees of freedom (can be increased for datasets exhibiting particularly high levels of noise at low expression magnitudes)
smooth.df: degrees of freedom to be used in calculating smoothed local regression between coefficient of variation and expression magnitude (and gene length, if provided). Leave at -1 for automated guess.
max.adj.var: maximum value allowed for the estimated adjusted variance (capping of adjusted variance is recommended when scoring pathway overdispersion relative to randomly sampled gene sets)
theta.range: valid theta range (should be the same as was set in knn.error.models() call
gene.length: optional vector of gene lengths (corresponding to the rows of counts matrix)
Value

a list containing the following fields:

• mat  adjusted expression magnitude values
• matw  weight matrix corresponding to the expression matrix
• arv  a vector giving adjusted variance values for each gene
• avmodes  a vector estimated average expression magnitudes for each gene
• modes  a list of batch-specific average expression magnitudes for each gene
• prior  estimated (or supplied) expression magnitude prior
• edf  estimated effective degrees of freedom
• fit.genes  fit.genes parameter

Examples

data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)

pagoda.view.aspects  View PAGODA output

Description

Create static image of PAGODA output visualizing cell hierarchy and top aspects of transcriptional heterogeneity

Usage

pagoda.view.aspects(tamr, row.clustering = hclust(dist(tamr$xv)), top = Inf, ...

Arguments

tamr  Combined pathways that show similar expression patterns. Output of pagoda.reduce.redundancy
row.clustering  Dendrogram of combined pathways clustering
top  Restrict output to the top n aspects of heterogeneity
...  additional arguments are passed to the view.aspects method during plotting

Value

PAGODA heatmap
Examples

```r
data(pollen)
cd <- clean.counts(pollen)

knn <- knn.error.models(cd, k=ncol(cd)/4, n.cores=10, min.count.threshold=2, min.nonfailed=5, max.model.plots=10)
varinfo <- pagoda.varnorm(knn, counts = cd, trim = 3/ncol(cd), max.adj.var = 5, n.cores = 1, plot = FALSE)
pwpc <- pagoda.pathway.wPCA(varinfo, go.env, n.components=1, n.cores=10, n.internal.shuffles=50)
tam <- pagoda.top.aspects(pwpc, return.table = TRUE, plot=FALSE, z.score=1.96) # top aspects based on GO only
pagoda.view.aspects(tam)
```

Description

Abstracts out mclapply implementation, and defaults to lapply when only one core is requested (helps with debugging)

Usage

```r
papply(..., n.cores = n)
```

Arguments

- `...`: parameters to pass to lapply, mclapply, bplapply, etc.
- `n.cores`: number of cores. If 1 core is requested, will default to lapply

pollen

Sample data

Description

Single cell data from Pollen et al. 2014 dataset.

References

scde

single-cell Differential Expression (with Pathway And Gene set Overdispersion Analysis)

Description

The scde package implements a set of statistical methods for analyzing single-cell RNA-seq data. scde fits individual error models for single-cell RNA-seq measurements. These models can then be used for assessment of differential expression between groups of cells, as well as other types of analysis. The scde package also contains the pagoda framework which applies pathway and gene set overdispersion analysis to identify and characterize putative cell subpopulations based on transcriptional signatures. See vignette("diffexp") for a brief tutorial on differential expression analysis. See vignette("pagoda") for a brief tutorial on pathway and gene set overdispersion analysis to identify and characterize cell subpopulations. More extensive tutorials are available at http://pklab.med.harvard.edu/scde/index.html. (test)

Author(s)

Peter Kharchenko <Peter_Kharchenko@hms.harvard.edu>
Jean Fan <jeanfan@fas.harvard.edu>

scde.browse.diffexp View differential expression results in a browser

Description

Launches a browser app that shows the differential expression results, allowing to sort, filter, etc. The arguments generally correspond to the scde.expression.difference() call, except that the results of that call are also passed here. Requires Rook and rjson packages to be installed.

Usage

scde.browse.diffexp(results, models, counts, prior, groups = NULL, batch = NULL, geneLookupURL = NULL, server = NULL, name = "scde", port = NULL)

Arguments

results result object returned by scde.expression.difference(). Note to browse group posterior levels, use return.posteriors = TRUE in the scde.expression.difference() call.
models model matrix
counts count matrix
prior prior
 groups | group information 
 batch | batch information 

geneLookupURL | The URL that will be used to construct links to view more information on gene names. By default (if can't guess the organism) the links will forward to ENSEMBL site search, using geneLookupURL = "http://useast.ensembl.org/Multi/Search/Results?q = {0}". The "0" in the end will be substituted with the gene name. For instance, to link to GeneCards, use "http://www.genecards.org/cgi-bin/carddisp.pl?gene = {0}".

server | optional previously returned instance of the server, if want to reuse it.
name | app name (needs to be altered only if adding more than one app to the server using server parameter)
port | Interactive browser port 

**Value**

server instance, on which `stop()` function can be called to kill the process.

**Examples**

```r
data(es.mef.small)

cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)

sg <- factor(gsub("(MEF|ESC).*", "\1", colnames(cd)), levels = c("ESC", "MEF"))

names(sg) <- colnames(cd)

o.ifm <- scde.error.models(counts = cd, groups = sg, n.cores = 10, threshold.segmentation = TRUE)

o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)

# make sure groups corresponds to the models (o.ifm)

groups <- factor(gsub("(MEF|ESC).*", "\1", rownames(o.ifm)), levels = c("ESC", "MEF"))

names(groups) <- row.names(o.ifm)

ediff <- scde.expression.difference(o.ifm, cd, o.prior, groups = groups, n.randomizations = 100, n.cores = 10, verbose = 1)

scde.browse.diffexp(ediff, o.ifm, cd, o.prior, groups = groups, geneLookupURL="http://www.informatics.jax.org/searchtool/Search.do?query={0}")
```

---

**scde.edff**

*Internal model data*

**Description**

Numerically-derived correction for NB->chi squared approximation stored as an local regression model
scde.error.models

Fit single-cell error/regression models

Description

Fit error models given a set of single-cell data (counts) and an optional grouping factor (groups). The cells (within each group) are first cross-compared to determine a subset of genes showing consistent expression. The set of genes is then used to fit a mixture model (Poisson-NB mixture, with expression-dependent concomitant).

Usage

scde.error.models(counts, groups = NULL, min.nonfailed = 3,
threshold.segmentation = TRUE, min.count.threshold = 4,
zero.count.threshold = min.count.threshold, zero.lambda = 0.1,
save.crossfit.plots = FALSE, save.model.plots = TRUE, n.cores = 12,
min.size.entries = 2000, max.pairs = 5000, min.pairs.per.cell = 10,
verbose = 0, linear.fit = TRUE, local.theta.fit = linear.fit,
theta.fit.range = c(0.01, 100))

Arguments

counts read count matrix. The rows correspond to genes (should be named), columns correspond to individual cells. The matrix should contain integer counts

groups an optional factor describing grouping of different cells. If provided, the cross-fits and the expected expression magnitudes will be determined separately within each group. The factor should have the same length as ncol(counts).

min.nonfailed minimal number of non-failed observations required for a gene to be used in the final model fitting

threshold.segmentation use a fast threshold-based segmentation during cross-fit (default: TRUE)

min.count.threshold the number of reads to use to guess which genes may have "failed" to be detected in a given measurement during cross-cell comparison (default: 4)

zero.count.threshold threshold to guess the initial value (failed/non-failed) during error model fitting procedure (defaults to the min.count.threshold value)

zero.lambda the rate of the Poisson (failure) component (default: 0.1)

save.crossfit.plots whether png files showing cross-fit segmentations should be written out (default: FALSE)

save.model.plots whether pdf files showing model fits should be written out (default = TRUE)

n.cores number of cores to use
min.size.entries
minimum number of genes to use when determining expected expression magnitude during model fitting

max.pairs
maximum number of cross-fit comparisons that should be performed per group (default: 5000)

min.pairs.per.cell
minimum number of pairs that each cell should be cross-compared with

verbose
1 for increased output

linear.fit
Boolean of whether to use a linear fit in the regression (default: TRUE).

local.theta.fit
Boolean of whether to fit the overdispersion parameter theta, ie. the negative binomial size parameter, based on local regression (default: set to be equal to the linear.fit parameter)

theta.fit.range
Range of valid values for the overdispersion parameter theta, ie. the negative binomial size parameter (default: c(1e-2, 1e2))

details
Note: the default implementation has been changed to use linear-scale fit with expression-dependent NB size (overdispersion) fit. This represents an interative improvement on the originally published model. Use linear.fit=F to revert back to the original fitting procedure.

value
a model matrix, with rows corresponding to different cells, and columns representing different parameters of the determined models

examples

```r
data(es.mef.small)
cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
sg <- factor(gsub("(MEF|ESC).*", \1", \"\", colnames(cd)), levels = c("ESC", "MEF"))
names(sg) <- colnames(cd)

O.ifm <- scde.error.models(counts = cd, groups = sg, n.cores = 10, threshold.segmentation = TRUE)
```

scde.expression.difference

Test for expression differences between two sets of cells

description
Use the individual cell error models to test for differential expression between two groups of cells.
Usage

scde.expression.difference(models, counts, prior, groups = NULL,
batch = NULL, n.randomizations = 150, n.cores = 10,
batch.models = models, return.posteriors = FALSE, verbose = 0)

Arguments

models models determined by scde.error.models
counts read count matrix
prior gene expression prior as determined by scde.expression.prior
groups a factor determining the two groups of cells being compared. The factor entries should correspond to the rows of the model matrix. The factor should have two levels. NAs are allowed (cells will be omitted from comparison).
batch a factor (corresponding to rows of the model matrix) specifying batch assignment of each cell, to perform batch correction
n.randomizations number of bootstrap randomizations to be performed
n.cores number of cores to utilize
batch.models (optional) separate models for the batch data (if generated using batch-specific group argument). Normally the same models are used.
return.posteriors whether joint posterior matrices should be returned
verbose integer verbose level (1 for verbose)

Value

default: a data frame with the following fields:

- lb, mle, ub lower bound, maximum likelihood estimate, and upper bound of the 95% conservative estimate of expression-fold change (equals to the min(abs(c(lb, ub))), or 0 if the CI crosses the 0 Z uncorrected Z-score of expression difference cZ expression difference Z-score corrected for multiple hypothesis testing using Holm procedure
If batch correction has been performed (batch has been supplied), analogous data frames are returned in slots $batch.adjusted for batch-corrected results, and $batch.effect for the differences explained by batch effects alone.

return.posteriors = TRUE: A list is returned, with the default results data frame given in the $results slot. difference.posterior returns a matrix of estimated expression difference posteriors (rows - genes, columns correspond to different magnitudes of fold-change - log2 values are given in the column names) joint.posteriors a list of two joint posterior matrices (rows - genes, columns correspond to the expression levels, given by prior$x grid)

Examples

data(es.mef.small)
cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
sg <- factor(gsub("(MEF|ESC).*", \"\1\", colnames(cd)), levels = c("ESC", "MEF"))
names(sg) <- colnames(cd)

o.ifm <- scde.error.models(counts = cd, groups = sg, n.cores = 10, threshold.segmentation = TRUE)
o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)
# make sure groups corresponds to the models (o.ifm)
groups <- factor(gsub("(MEF|ESC).*", \1", "\", rownames(o.ifm)), levels = c("ESC", "MEF"))
names(groups) <- row.names(o.ifm)
ediff <- scde.expression.difference(o.ifm, cd, o.prior, groups = groups, n.randomizations = 100, n.cores = n.cores,

---

scde.expression.magnitude

*Return scaled expression magnitude estimates*

---

**Description**

Return point estimates of expression magnitudes of each gene across a set of cells, based on the regression slopes determined during the model fitting procedure.

**Usage**

`scde.expression.magnitude(models, counts)`

**Arguments**

- `models`: models determined by `scde.error.models`
- `counts`: count matrix

**Value**

A matrix of expression magnitudes on a log scale (rows - genes, columns - cells)

**Examples**

data(es.mef.small)

cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
data(o.ifm) # Load precomputed model. Use ?scde.error.models to see how o.ifm was generated
# get expression magnitude estimates
lfpm <- scde.expression.magnitude(o.ifm, cd)
scde.expression.prior  Estimate prior distribution for gene expression magnitudes

Description

Use existing count data to determine a prior distribution of genes in the dataset

Usage

scde.expression.prior(models, counts, length.out = 400, show.plot = FALSE, pseudo.count = 1, bw = 0.1, max.quantile = 1 - 0.001, max.value = NULL)

Arguments

models  models determined by scde.error.models
counts  count matrix
length.out  number of points (resolution) of the expression magnitude grid (default: 400). Note: larger numbers will linearly increase memory/CPU demands.
show.plot  show the estimate posterior
pseudo.count  pseudo-count value to use (default 1)
bw  smoothing bandwidth to use in estimating the prior (default: 0.1)
max.quantile  determine the maximum expression magnitude based on a quantile (default: 0.999)
max.value  alternatively, specify the exact maximum expression magnitude value

Value

a structure describing expression magnitude grid ($x$, on log10 scale) and prior ($y$)

Examples

data(es.mef.small)
cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
data(o.ifm)  # Load precomputed model. Use ?scde.error.models to see how o.ifm was generated
o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)
scde.failure.probability

Calculate drop-out probabilities given a set of counts or expression magnitudes

Description

Returns estimated drop-out probability for each cell (row of models matrix), given either an expression magnitude

Usage

scde.failure.probability(models, magnitudes = NULL, counts = NULL)

Arguments

models models determined by scde.error.models
magnitudes a vector (length(counts) == nrow(models)) or a matrix (columns correspond to cells) of expression magnitudes, given on a log scale
counts a vector (length(counts) == nrow(models)) or a matrix (columns correspond to cells) of read counts from which the expression magnitude should be estimated

Value

a vector or a matrix of drop-out probabilities

Examples

data(es.mef.small)
cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
data(o.ifm) # Load precomputed model. Use ?scde.error.models to see how o.ifm was generated
o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)
# calculate probability of observing a drop out at a given set of magnitudes in different cells
mags <- c(1.0, 1.5, 2.0)
p <- scde.failure.probability(o.ifm, magnitudes = mags)
# calculate probability of observing the dropout at a magnitude corresponding to the
# number of reads actually observed in each cell
self.p <- scde.failure.probability(o.ifm, counts = cd)
scde.fit.models.to.reference

Fit scde models relative to provided set of expression magnitudes

Description

If group-average expression magnitudes are available (e.g. from bulk measurement), this method can be used to fit individual cell error models relative to that reference.

Usage

scde.fit.models.to.reference(counts, reference, n.cores = 10, zero.count.threshold = 1, nrep = 1, save.plots = FALSE, plot.filename = "reference.model.fits.pdf", verbose = 0, min.fpm = 1)

Arguments

counts    count matrix
reference  a vector of expression magnitudes (read counts) corresponding to the rows of the count matrix
n.cores   number of cores to use
zero.count.threshold    read count to use as an initial guess for the zero threshold
nrep      number independent of mixture fit iterations to try (default = 1)
save.plots  whether to write out a pdf file showing the model fits
plot.filename model fit pdf filename
verbose   verbose level
min.fpm    minimum reference fpm of genes that will be used to fit the models (defaults to 1). Note: fpm is calculated from the reference count vector as reference/sum(reference)*1e6

Value

matrix of scde models

Examples

data(es.mef.small)
cd  <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
o.ifm  <- scde.error.models(counts = cd, groups = sg, n.cores = 10, threshold.segmentation = TRUE)
o.prior  <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)
# calculate joint posteriors across all cells
jp <- scde.posteriors(models = o.ifm, cd, o.prior, n.cores = 10, return.individual.posterior.modes = TRUE, n.randomizations = 100)
# use expected expression magnitude for each gene
av.mag <- as.numeric(jp$jp) %*% as.numeric(colnames(jp$jp))
# translate into counts
av.mag.counts <- as.integer(round(av.mag))
# now, fit alternative models using av.mag as a reference (normally this would correspond to bulk RNA expression magnitude)
ref.models <- scde.fit.models.to.reference(cd, av.mag.counts, n.cores = 1)

---

**scde.posteriors**

*Calculate joint expression magnitude posteriors across a set of cells*

**Description**

Calculates expression magnitude posteriors for the individual cells, and then uses bootstrap resampling to calculate a joint expression posterior for all the specified cells. Alternatively during batch-effect correction procedure, the joint posterior can be calculated for a random composition of cells of different groups (see batch and composition parameters).

**Usage**

```r
scde.posteriors(models, counts, prior, n.randomizations = 100, batch = NULL,
                 composition = NULL, return.individual.posteriors = FALSE,
                 return.individual.posterior.modes = FALSE, ensemble.posterior = FALSE,
                 n.cores = 20)
```

**Arguments**

- `models`: models models determined by `scde.error.models`
- `counts`: read count matrix
- `prior`: gene expression prior as determined by `scde.expression.prior`
- `n.randomizations`: number of bootstrap iterations to perform
- `batch`: a factor describing which batch group each cell (i.e. each row of models matrix) belongs to
- `composition`: a vector describing the batch composition of a group to be sampled
- `return.individual.posteriors`: whether expression posteriors of each cell should be returned
- `return.individual.posterior.modes`: whether modes of expression posteriors of each cell should be returned
- `ensemble.posterior`: Boolean of whether to calculate the ensemble posterior (sum of individual posteriors) instead of a joint (product) posterior. (default: FALSE)
- `n.cores`: number of cores to utilize
Value

**default:** a posterior probability matrix, with rows corresponding to genes, and columns to expression levels (as defined by `prior$x`)

**return.individual.posterior.modes:** a list is returned, with the `$jp` slot giving the joint posterior matrix, as described above. The `$modes` slot gives a matrix of individual expression posterior mode values on log scale (rows - genes, columns - cells)

**return.individual.posteriors:** a list is returned, with the `$post` slot giving a list of individual posterior matrices, in a form analogous to the joint posterior matrix, but reported on log scale

Examples

data(es.mef.small)
cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
data(o.ifm) # Load precomputed model. Use ?scde.error.models to see how o.ifm was generated
o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)
# calculate joint posteriors
jp <- scde.posteriors(o.ifm, cd, o.prior, n.cores = 1)

---

**scde.test.gene.expression.difference**

*Test differential expression and plot posteriors for a particular gene*

Description

The function performs differential expression test and optionally plots posteriors for a specified gene.

Usage

```r
scde.test.gene.expression.difference(gene, models, counts, prior,
  groups = NULL, batch = NULL, batch.models = models,
  n.randomizations = 1000, show.plots = TRUE, return.details = FALSE,
  verbose = FALSE, ratio.range = NULL, show.individual.posteriors = TRUE,
  n.cores = 1)
```

Arguments

- **gene**: name of the gene to be tested
- **models**: models
- **counts**: read count matrix (must contain the row corresponding to the specified gene)
- **prior**: expression magnitude prior
- **groups**: a two-level factor specifying between which cells (rows of the models matrix) the comparison should be made
show.app

batch optional multi-level factor assigning the cells (rows of the model matrix) to different batches that should be controlled for (e.g. two or more biological replicates). The expression difference estimate will then take into account the likely difference between the two groups that is explained solely by their difference in batch composition. Not all batch configuration may be corrected this way.

batch.models optional set of models for batch comparison (typically the same as models, but can be more extensive, or recalculated within each batch)

n.randomizations number of bootstrap/sampling iterations that should be performed

show.plots whether the plots should be shown

return.details whether the posterior should be returned

verbose set to T for some status output

ratio.range optionally specifies the range of the log2 expression ratio plot

show.individual.posteriors whether the individual cell expression posteriors should be plotted

n.cores number of cores to use (default = 1)

Value

by default returns MLE of log2 expression difference, 95

Examples

data(es.mef.small)

cd <- clean.counts(es.mef.small, min.lib.size=1000, min.reads = 1, min.detected = 1)
data(o.ifm) # Load precomputed model. Use ?scde.error.models to see how o.ifm was generated

o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)

scde.test.gene.expression.difference("Tdh", models = o.ifm, counts = cd, prior = o.prior)

show.app View PAGODA application

Description

Installs a given pagoda app (or any other rook app) into a server, optionally making a call to show it in the browser.

Usage

show.app(app, name, browse = TRUE, port = NULL, ip = "127.0.0.1", server = NULL)
view.aspects

Arguments

- **app**: pagoda app (output of make.pagoda.app()) or another rook app
- **name**: URL path name for this app
- **browse**: whether a call should be made for browser to show the app
- **port**: optional port on which the server should be initiated
- **ip**: IP on which the server should listen (typically localhost)
- **server**: an (optional) Rook server instance (defaults to ___scde.server)

Value

Rook server instance

Examples

```r
app <- make.pagoda.app(tamr2, tam, varinfo, go.env, pwpca, clpca, col.cols=col.cols, cell.clustering=hc, title="NPCs")
# show app in the browser (port 1468)
show.app(app, "pollen", browse = TRUE, port=1468)
```

Description

Internal function to visualize aspects of transcriptional heterogeneity as a heatmap. Used by `pagoda.view.aspects`.

Usage

```r
view.aspects(mat, row.clustering = NA, cell.clustering = NA, zlim = c(-1, 1) * quantile(mat, p = 0.95), row.cols = NULL, col.cols = NULL, cols = colorRampPalette(c("darkgreen", "white", "darkorange"), space = "Lab")(1024), show.row.var.colors = TRUE, top = Inf, ...)
```

Arguments

- **mat**: Numeric matrix
- **row.clustering**: Row dendrogram
- **cell.clustering**: Column dendrogram
- **zlim**: Range of the normalized gene expression levels, inputted as a list: `c(lower_bound, upper_bound)`. Values outside this range will be Winsorized. Useful for increasing the contrast of the heatmap visualizations. Default, set to the 5th and 95th percentiles.
- **row.cols**: Matrix of row colors.
col.cols  Matrix of column colors. Useful for visualizing cell annotations such as batch labels.

cols  Heatmap colors

show.row.var.colors  Boolean of whether to show row variance as a color track

top  Restrict output to the top n aspects of heterogeneity

...  additional arguments for heatmap plotting

Value

A heatmap

---

ViewPagodaApp-class  A Reference Class to represent the PAGODA application

Description

This ROOK application class enables communication with the client-side ExtJS framework and Inchlib HTML5 canvas libraries to create the graphical user interface for PAGODA Refer to the code in make.pagoda.app for usage example

Fields

results  Output of the pathway clustering and redundancy reduction
genes  List of genes to display in the Detailed clustering panel
mat  Matrix of posterior mode count estimates
matw  Matrix of weights associated with each estimate in mat
goenv  Gene set list as an environment
renv  Global environment
name  Name of the application page; for display as the page title
trim  Trim quantity used for Winsorization for visualization
batch  Any batch or other known confounders to be included in the visualization as a column color track
winsorize.matrix

Description
Sets the ncol(mat)*trim top outliers in each row to the next lowest value same for the lowest outliers

Usage
winsorize.matrix(mat, trim)

Arguments

mat matrix
trim fraction of outliers (on each side) that should be Winsorized, or (if the value is >= 1) the number of outliers to be trimmed on each side

Value
Winsorized matrix

Examples

set.seed(0)
mat <- matrix(c(rnorm(5*10, mean=0, sd=1), rnorm(5*10, mean=5, sd=1)), 10, 10) # random matrix
mat[1,1] <- 1000 # make outlier
range(mat) # look at range of values
win.mat <- winsorize.matrix(mat, 0.1)
range(win.mat) # note outliers removed
Index

bwpca, 3

show.app, 34

clean.counts, 4

view.aspects, 27, 35

clean.gos, 5

ViewPagodaApp (ViewPagodaApp-class), 36

es.mef.small, 5

ViewPagodaApp-class, 36

knn, 6

winsorize.matrix, 37

knn.error.models, 6


make.pagoda.app, 8, 36

o.ifm, 9


pagoda.cluster.cells, 8, 9

papply, 22

pagoda.effective.cells, 10

pollen, 22

pagoda.gene.clusters, 8, 11

cande, 23

pagoda.pathway.wPCA, 8, 12

cande-package (cande), 23

apgoda.reduce.loading.redundancy, 8, 14

cande.browse.diffexp, 23

apgoda.reduce.redundancy, 8, 15, 21

cande.edff, 24

apgoda.show.pathways, 16

cande.error.models, 25, 27–30, 32

cande.expression.difference, 26

cande.expression.magnitude, 28

cande.expression.prior, 27, 29, 32

cande.failure.probability, 30

cande.fit.models.to.reference, 31

cande.posteriors, 32

cande.test.gene.expression.difference, 33

cande.varnorm, 8, 19

cande.view.aspects, 21, 35

cande.show.aspects, 34

cande.view.aspects, 21, 35

cande.varnorm, 8, 19

cande.show.aspects, 34

cande.view.aspects, 21, 35