Package ‘scde’

December 22, 2016

Type Package
Title Single Cell Differential Expression
Version 2.2.0
Date 2016-01-20
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, Kaeber 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]
Maintainer Jean Fan <jeanfan@fas.harvard.edu>
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 RNASeq, StatisticalMethod, DifferentialExpression, Bayesian, Transcription, Software
LinkingTo Rcpp, RcppArmadillo
R topics documented:

VignetteBuilder knitr
RoxygenNote 5.0.0
NeedsCompilation yes

R topics documented:

bw pca .......................... 3
clean.counts .......................... 4
clean.gos .......................... 4
es.mef.small .......................... 5
knn ................................ 5
knn.error.models ......................... 6
make.pagoda.models ................. 7
o.ifm ................................ 8
pagoda.cluster.cells .............. 8
pagoda.effective.cells ............ 9
pagoda.gene.clusters .............. 10
pagoda.pathway.wPCA ............. 11
pagoda.reduce.loading.redundancy 13
pagoda.reduce.redundancy ........ 14
pagoda.show.pathways ............ 15
pagoda.subtract.aspect .......... 16
pagoda.top.aspects ............... 17
pagoda.varnorm .................... 18
pagoda.view.aspects ............. 20
papply ................................ 21
pollen ................................ 21
scde .............................. 21
scde.browse.diffexp ............. 22
scde.diff ................................ 23
scde.error.models ............... 23
scde.expression.difference ...... 25
scde.expression.magnitude ...... 26
scde.expression.prior .......... 27
scde.failure.probability ....... 27
scde.fit.models.to.reference ... 28
scde.posteriors ................... 29
scde.test.gene.expression.difference 30
show.app ............................ 32
view.aspects ...................... 32
ViewPagodaApp-class ............. 33
winsorize.matrix ............... 34

Index 35
**bw pca**

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

**Description**

Implements a weighted PCA

**Usage**

```r
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 <- bw pca(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 <- bw pca(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
  - Minimum size for number of genes in a gene set (default: 5)
  - 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 Sample error model**

**Description**

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

**References**

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
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
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)
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)

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
pagoda.cluster.cells

clipca  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

o.ifm  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

pagoda.cluster.cells  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)
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

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)
pagoda.gene.clusters

Determine de-novo gene clusters and associated overdispersion info

Value

effective number of cells

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)
  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 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

trim number of clusters to be determined (recommended range is 100-200)

n.clusters number of randomly generated matrix samples to test the background distribution of lambda1 on

n.samples correlation method ("pearson", "spearman") to be used as a distance measure for clustering

cor.method 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.internal.shuffles number of wPCA EM algorithm starts at each iteration

n.starts number of cores to use

n.cores
pagoda.pathway.wPCA

Run weighted PCA analysis on pre-annotated gene sets

Description

For each valid gene set (having appropriate number of genes) in the provided environment (setenv), the method will run weighted PCA analysis, along with analogous analyses of random gene sets of the same size, or shuffled expression magnitudes for the same gene set.
pagoda.pathway.wPCA

Usage

pagoda.pathway.wPCA(varinfo, setenv, n.components = 2, n.cores = detectCores(), min.pathway.size = 10, max.pathway.size = 1000, n.randomizations = 10, n.internal.shuffles = 0, n.starts = 10, center = TRUE, batch.center = TRUE, proper.gene.names = NULL, verbose = 0)

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])))
### 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`
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

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 = 50)
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
	tamr <- pagoda.reduce.loading.redundancy(tam, pwpca)
tamr2 <- pagoda.reduce.redundancy(tamr, distance.threshold = 0.9, plot = TRUE, labRow = NA, labCol = NA, box = TRUE)

pagoda.show.pathways View pathway or gene weighted PCA

Description

Takes in a list of pathways (or a list of genes), runs weighted PCA, optionally showing the result.

Usage

pagoda.show.pathways(pathways, varinfo, goenv = NULL, n.genes = 20,
	two.sided = FALSE, n.pc = rep(1, length(pathways)), colcols = NULL,
	zlim = NULL, showRowLabels = FALSE, cexCol = 1, cexRow = 1,
	nstarts = 10, cell.clustering = NULL, show.cell.dendrogram = TRUE,
	plot = TRUE, box = TRUE, trim = 0, return.details = FALSE, ...)

Arguments

pathways character vector of pathway or gene names
varinfo output of pagoda.varnorm()
goenv environment mapping pathways to genes
n.genes number of genes to show
two.sided whether the set of shown genes should be split among highest and lowest loading (T) or if genes with highest absolute loading (F) should be shown
n.pc optional integer vector giving the number of principal component to show for each listed pathway
colcols optional column color matrix
zlim optional z color limit
showRowLabels controls whether row labels are shown in the plot
cexCol column label size (cex)
cexRow row label size (cex)
Control for a particular aspect of expression heterogeneity in a given population

pagoda.subtract.aspect

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

Value

a modified varinfo object with adjusted expression matrix (varinfo$mat)
pagoda.top.aspects

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)

# 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

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())
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)
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.

Usage

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)
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

pagoda.view.aspects(tam)

pagoda.view.aspects View PAGODA output

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

Usage

epagoda.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

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

pagoda.view.aspects(tam)
**papply**

**wrapper around different mclapply mechanisms**

**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](http://pklab.med.harvard.edu/scde/index.html) (test)

**Author(s)**

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

```r
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)
scde.browse.diffexp(ediff, o.ifm, cd, o.prior, groups = groups, geneLookupURL="http://www.informatics.jax.org")

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**

```r
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, i.e. 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, i.e. 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 interactive 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
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**

```r
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. 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(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

```r
cnde.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

```r
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) == nrows(models)) or a matrix (columns correspond to cells) of expression magnitudes, given on a log scale
- counts: a vector (length(counts) == nrows(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
scde.posteriors

zero.count.threshold  
read count to use as an initial guess for the zero threshold

nenrep  
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)
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

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

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

```r
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)
```
Usage

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
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)

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

view.aspects  View heatmap

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

Usage
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, ...)

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

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

## Usage

```r
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

```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
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

bw pca, 3

clean.counts, 4
clean.gos, 4
es.mef.small, 5

knn, 5
knn.error.models, 6

make.pagoda.app, 7, 33

o.ifm, 8

pagoda.cluster.cells, 8, 8
pagoda.effective.cells, 9
pagoda.gene.clusters, 8, 10
pagoda.pathway.wPCA, 7, 11
pagoda.reduce.loading.redundancy, 7, 13
pagoda.reduce.redundancy, 7, 14, 20
pagoda.show.pathways, 15
pagoda.subtract.aspect, 16
pagoda.top.aspects, 17
pagoda.varnorm, 7, 18
pagoda.view.aspects, 20, 32
papply, 21
pollen, 21

scde, 21
scde-package (scde), 21
scde.browse.diffexp, 22
scde.edff, 23
scde.error.models, 23, 25–28, 30
scde.expression.difference, 25
scde.expression.magnitude, 26
scde.expression.prior, 25, 27, 30
scde.failure.probability, 27
scde.fit.models.to.reference, 28
scde.posteriors, 29
scde.test.gene.expression.difference, 30
show.app, 32

view.aspects, 20, 32
ViewPagodaApp (ViewPagodaApp-class), 33

ViewPagodaApp-class, 33

winsorize.matrix, 34