Package ‘scone’

February 1, 2018

Version 1.3.0

Title Single Cell Overview of Normalized Expression data

Description SCONE is an R package for comparing and ranking the performance of different normalization schemes for single-cell RNA-seq and other high-throughput analyses.

Author Michael Cole [aut, cre, cph], Davide Risso [aut, cph]

Maintainer Michael Cole <mbeloc@gmail.com>

License Artistic-2.0

Depends R (>= 3.4), methods, SummarizedExperiment

Imports graphics, stats, utils, aroma.light, BiocParallel, class, cluster, compositions, diptest, edgeR, fpc, gplots, grDevices, hexbin, limma, matrixStats, mixtools, RColorBrewer, boot, rhdf5, RUVSeq, rARPACK

Suggests BiocStyle, DT, ggplot2, knitr, miniUI, NMF, plotly, reshape2, rmarkdown, scran, scRNAseq, shiny, testthat, visNetwork

VignetteBuilder knitr

LazyLoad yes

biocViews Normalization, Preprocessing, QualityControl, GeneExpression, RNASeq, Software, Transcriptomics, Sequencing, SingleCell, Coverage

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

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

`biplot_color` *Function for biplotting with no point labels and with points color-coded according to a quantitative variable. For example: the rank of normalization performance.*

Description

This function implements biplot for `prcomp` objects.

Usage

```
biplot_color(x, y, rank = TRUE, ties_method = c("max", "min", "first",
  "last", "random"), choices = 1:2, expand = 1, ...)
```

Arguments

`x`	`prcomp` object.
`y`	numeric. Quantitative values used to color the points. If rank is FALSE, all values must be positive integers and less than or equal to the length of y.
`rank`	logical. If TRUE (default) y will be transformed by the rank() function
`ties_method`	character. ties.method used by the rank() function

`choices`	numeric. 2 principal components to plot. Default to first two PCs.
`expand`	numeric. value used to adjust the spread of the arrows relative to the points.
`...`	arguments passed to plot.

Value

Invisibly returns scaled point coordinates used in plot.

Examples

```
mat <- matrix(rnorm(1000), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")

pc <- prcomp(mat)

biplot_color(pc, rank(pc$x[,1]))
```

`biplot_interactive` *Interactive biplot*

Description

This is a wrapper around `biplot_color`, creating a shiny gadget to allow the user to select specific points in the graph.

Usage

```
biplot_interactive(x, ...)
```

Arguments

`x`	a `SconeExperiment` object.
`...`	passed to `biplot_color`.

Details

Since this is based on the shiny gadget feature, it will not work in static documents, such as vignettes or markdown / knitr documents. See `biplot_color` for more details on the internals.

Value

A `SconeExperiment` object representing selected methods.

Examples

```
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity,
   uq=UQ_FN, deseq=DESEQ_FN,  fq=FQT_FN),
evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
   bpparam = BiocParallel::SerialParam())
## Not run:
biplot_interactive(res)

## End(Not run)
```

CLR_FN *Centered log-ratio (CLR) normalization wrapper function*

Description

Centered log-ratio (CLR) normalization wrapper function

Usage

```
CLR_FN(ei)
```

Arguments

ei	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling wrapper for clr).

Value

CLR normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- CLR_FN(ei)
```

control_genes *Data: Positive and Negative Control Genes*

Description

Sets of "positive" and "negative" control genes, useful arguments for scone.

Details

These gene sets can be used as negative or positive controls, either for RUV factor normalization or for evaluation and ranking of the normalization workflows.

Gene set datasets are in the form of `data.frame`, with the first column containing the gene symbols and an (optional) second column containing additional information (such as cortical layer or cell cycle phase).

Note that the gene symbols follow the mouse conventions (i.e. capitalized) or the human conventions (i.e, all upper-case), based on the original publication. One can use the toupper, tolower, and toTitleCase functions to alter symbol conventions.

Mouse gene symbols in `cortical_markers` are transcribed from Figure 3 of Molyneaux et al. (2007): "laminar-specific expression of 66 genes within the neocortex."

Human gene symbols in `housekeeping` are derived from the list of "housekeeping" genes from the cDNA microarray analysis of Eisenberg and Levanon (2003): "[HK genes] belong to the class of genes that are EXPRESSED in all tissues." "... from 47 different human tissues and cell lines."

Human gene symbols in `housekeeping_revised` from Eisenberg and Levanon (2013): "This list provided ... is based on analysis of next-generation sequencing (RNA-seq) data. At least one variant of these genes is expressed in all tissues uniformly... The RefSeq transcript according to which we deemed the gene 'housekeeping' is given." Housekeeping exons satisfy "(i) expression observed in all tissues; (ii) low variance over tissues: standard-deviation [log2(RPKM)]<1; and (iii) no exceptional expression in any single tissue; that is, no log-expression value differed from the averaged log2(RPKM) by two (fourfold) or more." "We define a housekeeping gene as a gene for which at least one RefSeq transcript has more than half of its exons meeting the previous criteria (thus being housekeeping exons)."

Human gene symbols in `cellcycle_genes` from Macosko et al. (2015) and represent a set of genes marking G1/S, S, G2/M, M, and M/G1 phases.

References

Molyneaux, B.J., Arlotta, P., Menezes, J.R. and Macklis, J.D.. Neuronal subtype specification in the cerebral cortex. Nature Reviews Neuroscience, 2007, 8(6):427-437.

Eisenberg E, Levanon EY. Human housekeeping genes are compact. Trends in Genetics, 2003, 19(7):362-5.

Eisenberg E, Levanon EY. Human housekeeping genes, revisited. Trends in Genetics, 2013, 29(10):569-74.

Macosko, E. Z., et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell, 2015, 161.5:1202-1214.

Examples

```
data(housekeeping)
data(housekeeping_revised)
data(cellcycle_genes)
data(cortical_markers)
```

DESEQ_FN *Relative log-expression (RLE; DESeq) scaling normalization wrapper function*

Description

Relative log-expression (RLE; DESeq) scaling normalization wrapper function

Usage

```
DESEQ_FN(ei)
```

Arguments

`ei`	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling wrapper for `calcNormFactors`).

Value

RLE normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- DESEQ_FN(ei)
```

estimate_ziber *Parameter estimation of zero-inflated bernoulli model*

Description

This function implements an expectation-maximization algorithm for a zero-inflated bernoulli model of transcript detection, modeling gene expression state (off of on) as a bernoulli draw on a gene-specific expression rate (Z in 0,1). Detection conditioned on expression is a logistic function of gene-level features. The bernoulli model is modeled numerically by a logistic model with an intercept.

Usage

```
estimate_ziber(x, fp_tresh = 0, gfeatM = NULL, bulk_model = FALSE,
  pos_controls = NULL, em_tol = 0.01, maxiter = 100, verbose = FALSE)
```

Arguments

`x` matrix. An expression data matrix (genes in rows, cells in columns)

`fp_tresh` numeric. Threshold for calling a positive detection (D = 1). Default 0.

`gfeatM` matrix. Numeric gene level determinants of drop-out (genes in rows, features in columns)

`bulk_model` logical. Use median log-expression of gene in detected fraction as sole gene-level feature. Default FALSE. Ignored if gfeatM is specified.

`pos_controls` logical. TRUE for all genes that are known to be expressed in all cells.

`em_tol` numeric. Convergence treshold on log-likelihood.

`maxiter` numeric. The maximum number of iterations. Default 100.

`verbose` logical. Whether or not to print the value of the likelihood at each iteration.

Value

a list with the following elements:

- W coefficients of sample-specific logistic drop-out model
- Alpha intercept and gene-level parameter matrix
- X intercept
- Beta coefficient of gene-specific logistic expression model
- fnr_character the probability, per gene, of P(D=0|E=1)
- p_nodrop 1 - the probability P(drop|Y), useful as weights in weighted PCA
- expected_state the expected value E[Z] (1 = "on")
- loglik the log-likelihood
- convergence 0 if the algorithm converged and 1 if maxiter was reached

Examples

```
mat <- matrix(rpois(1000, lambda = 3), ncol=10)
mat = mat * matrix(1-rbinom(1000, size = 1, prob = .01), ncol=10)
ziber_out = suppressWarnings(estimate_ziber(mat,
   bulk_model = TRUE,
   pos_controls = 1:10))
```

`factor_sample_filter` *Factor-based Sample Filtering: Function to filter single-cell RNA-Seq libraries.*

Description

This function returns a sample-filtering report for each cell in the input expression matrix, describing whether it passed filtering by factor-based filtering, using PCA of quality metrics.

Usage

```
factor_sample_filter(expr, qual, gene_filter = NULL, max_exp_pcs = 5,
  qual_select_q_thresh = 0.01, force_metrics = NULL, good_metrics = NULL,
  min_qual_variance = 0.7, zcut = 1, mixture = TRUE, dip_thresh = 0.01,
  plot = FALSE, hist_breaks = 20)
```

Arguments

`expr` matrix The data matrix (genes in rows, cells in columns).

`qual` matrix Quality metric data matrix (cells in rows, metrics in columns).

`gene_filter` Logical vector indexing genes that will be used for PCA. If NULL, all genes are used.

`max_exp_pcs` numeric number of expression PCs used in quality metric selection. Default 5.

`qual_select_q_thresh` numeric. q-value threshold for quality/expression correlation significance tests. Default 0.01

`force_metrics` logical. If not NULL, indexes quality metric to be forcefully included in quality PCA.

`good_metrics` logical. If not NULL, indexes quality metric that indicate better quality when of higher value.

`min_qual_variance` numeric. Minimum proportion of selected quality variance addressed in filtering. Default 0.70

`zcut` A numeric value determining threshold Z-score for sd, mad, and mixture subcriteria. Default 1.

`mixture` A logical value determining whether mixture modeling sub-criterion will be applied per primary criterion (quality score). If true, a dip test will be applied to each quality score. If a metric is multimodal, it is fit to a two-component normal mixture model. Samples deviating zcut sd's from optimal mean (in the inferior direction), have failed this sub-criterion.

`dip_thresh` A numeric value determining dip test p-value threshold. Default 0.05.

`plot` logical. Should a plot be produced?

`hist_breaks` hist() breaks argument. Ignored if ‘plot=FALSE‘.

Details

None

Value

A logical, representing samples passing factor-based filter.

Examples

```
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")
mfilt = factor_sample_filter(expr = mat,
qc, plot = TRUE,qual_select_q_thresh = 1)
```

`fast_estimate_ziber` *Fast parameter estimation of zero-inflated bernoulli model*

Description

This function implements Newton's method for solving zero of Expectation-Maximization equation at the limit of parameter convergence: a zero-inflated bernoulli model of transcript detection, modeling gene expression state (off of on) as a bernoulli draw on a gene-specific expression rate (Z in 0,1). Detection conditioned on expression is a logistic function of gene-level features. The bernoulli model is modeled numerically by a logistic model with an intercept.

Usage

```
fast_estimate_ziber(x, fp_tresh = 0, gfeatM = NULL, bulk_model = FALSE,
  pos_controls = NULL, rate_tol = 0.01, maxiter = 100, verbose = FALSE)
```

Arguments

`x`	matrix. An expression data matrix (genes in rows, cells in columns)
`fp_tresh`	numeric. Threshold for calling a positive detection (D = 1). Default 0.
`gfeatM`	matrix. Numeric gene level determinants of drop-out (genes in rows, features in columns)
`bulk_model`	logical. Use median log-expression of gene in detected fraction as sole gene-level feature. Default FALSE. Ignored if gfeatM is specified.
`pos_controls`	logical. TRUE for all genes that are known to be expressed in all cells.
`rate_tol`	numeric. Convergence treshold on expression rates (0-1).
`maxiter`	numeric. The maximum number of steps per gene. Default 100.
`verbose`	logical. Whether or not to print the value of the likelihood at each iteration.

Value

a list with the following elements:

- W coefficients of sample-specific logistic drop-out model
- Alpha intercept and gene-level parameter matrix
- X intercept
- Beta coefficient of gene-specific logistic expression model
- fnr_character the probability, per gene, of P(D=0|E=1)
- p_nodrop 1 - the probability P(drop|Y), useful as weights in weighted PCA
- expected_state the expected value E[Z] (1 = "on")
- loglik the log-likelihood
- convergencefor all genes, 0 if the algorithm converged and 1 if maxiter was reached

Examples

```
mat <- matrix(rpois(1000, lambda = 3), ncol=10)
mat = mat * matrix(1-rbinom(1000, size = 1, prob = .01), ncol=10)
ziber_out = suppressWarnings(fast_estimate_ziber(mat,
   bulk_model = TRUE,
   pos_controls = 1:10))
```

FQ_FN *Full-quantile normalization wrapper function*

Description

Full-quantile normalization wrapper function

Usage

```
FQ_FN(ei)

FQT_FN(ei)
```

Arguments

ei	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE "scaling" wrapper for `normalizeQuantileRank.matrix`).

Unlike FQ_FN, FQT_FN handles ties carefully (see `normalizeQuantiles` for details).

Value

Full-quantile normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- FQ_FN(ei)

ei <- matrix(0:20,nrow = 7)
eo <- FQT_FN(ei)
```

get_bio *Get Factor of Biological Conditions and Batch*

Description

Get Factor of Biological Conditions and Batch

Usage

```
get_bio(x)

get_batch(x)

## S4 method for signature 'SconeExperiment'
get_bio(x)

## S4 method for signature 'SconeExperiment'
get_batch(x)
```

Arguments

x	an object of class SconeExperiment.

Value

NULL or a factor containing bio or batch covariate.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat, bio = factor(rep(c(1,2),each = 5)),
          batch = factor(rep(c(1,2),times = 5)))
bio = get_bio(obj)
batch = get_batch(obj)
```

get_design *Retrieve Design Matrix*

Description

Given a SconeExperiment object created by a call to scone, it will return the design matrix of the selected method.

Usage

```
get_design(x, method)

## S4 method for signature 'SconeExperiment,character'
get_design(x, method)

## S4 method for signature 'SconeExperiment,numeric'
get_design(x, method)
```

Arguments

x	a SconeExperiment object containing the results of scone.
method	character or numeric. Either a string identifying the normalization scheme to be retrieved, or a numeric index with the rank of the normalization method to retrieve (according to scone ranking of normalizations).

Details

The numeric method will always return the design matrix corresponding to row method of the scone_params slot. This means that if scone was run with eval=TRUE, get_design(x, 1) will return the top ranked method. If scone was run with eval=FALSE, get_design(x, 1) will return the first normalization in the order saved by scone.

Value

The design matrix.

Methods (by class)

- x = SconeExperiment,method = character: If method is a character, it will return the design matrix corresponding to the normalization scheme specified by the character string. The string must be one of the row.names of the slot scone_params.
- x = SconeExperiment,method = numeric: If method is a numeric, it will return the design matrix according to the scone ranking.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat, bio = factor(rep(c(1,2),each = 5)),
           batch = factor(rep(c(1,2),times = 5)))
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0,
           adjust_batch = "yes", adjust_bio = "yes",
           eval_kclust=2, bpparam = BiocParallel::SerialParam())
design_top = get_design(res,1)
```

get_negconruv *Get Negative and Positive Controls*

Description

Get Negative and Positive Controls

Usage

```
get_negconruv(x)

get_negconeval(x)

get_poscon(x)

## S4 method for signature 'SconeExperiment'
get_negconruv(x)

## S4 method for signature 'SconeExperiment'
get_negconeval(x)

## S4 method for signature 'SconeExperiment'
get_poscon(x)
```

Arguments

x	an object of class SconeExperiment.

Value

NULL or a logical vector.

For get_negconruv the returned vector indicates which genes are negative controls to be used for RUV.

For get_negconeval the returned vector indicates which genes are negative controls to be used for evaluation.

For get_poscon the returned vector indicates which genes are positive controls to be used for evaluation.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat,negcon_ruv = 1:50 %in% 1:10,
           negcon_eval = 1:50 %in% 11:20,
           poscon = 1:50 %in% 21:30)
negcon_ruv = get_negconruv(obj)
negcon_eval = get_negconeval(obj)
poscon = get_poscon(obj)
```

get_normalized *Retrieve Normalized Matrix*

Description

Given a SconeExperiment object created by a call to scone, it will return a matrix of normalized counts (in log scale if log=TRUE).

Usage

```
get_normalized(x, method, ...)

## S4 method for signature 'SconeExperiment,character'
get_normalized(x, method, log = FALSE)

## S4 method for signature 'SconeExperiment,numeric'
get_normalized(x, method, log = FALSE)
```

Arguments

x	a SconeExperiment object containing the results of scone.
method	character or numeric. Either a string identifying the normalization scheme to be retrieved, or a numeric index with the rank of the normalization method to retrieve (according to scone ranking of normalizations).
...	additional arguments for specific methods.
log	logical. Should the data be returned in log-scale

Details

If scone was run with return_norm="in_memory", this function simply retrieves the normalized data from the assays slote of object.

If scone was run with return_norm="hdf5", this function will read the normalized matrix from the specified hdf5 file.

If scone was run with return_norm="no", this function will compute the normalized matrix on the fly.

The numeric method will always return the normalization corresponding to row method of the scone_params slot. This means that if scone was run with eval=TRUE, get_normalized(x, 1) will return the top ranked method. If scone was run with eval=FALSE, get_normalized(x,1) will return the first normalization in the order saved by scone.

Value

A matrix of normalized counts in log-scale.

Methods (by class)

- x = SconeExperiment,method = character: If method is a character, it will return the normalized matrix corresponding to the normalization scheme specified by the character string.The string must be one of the row.names of the slot scone_params.
- x = SconeExperiment,method = numeric: If method is a numeric, it will return the normalized matrix according to the scone ranking.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0,
           eval_kclust=2, bpparam = BiocParallel::SerialParam())
top_norm = get_normalized(res,1)
```

get_params *Extract scone parameters*

Description

Extract scone parameters

Usage

```
get_params(x)

## S4 method for signature 'SconeExperiment'
get_params(x)
```

Arguments

x	an object of class SconeExperiment.

Value

A data.frame containing workflow parameters for each scone workflow.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN),
           run = FALSE, k_ruv=0, k_qc=0, eval_kclust=2)
params = get_params(res)
```

get_qc *Get Quality Control Matrix*

Description

Get Quality Control Matrix

Usage

```
get_qc(x)

## S4 method for signature 'SconeExperiment'
get_qc(x)
```

Arguments

x	an object of class SconeExperiment.

Value

NULL or the quality control (QC) metric matrix.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat,
         qc = cbind(colSums(mat),colSums(mat > 0)))
qc = get_qc(obj)
```

get_scores *Extract scone scores*

Description

Extract scone scores

Usage

```
get_scores(x)

get_score_ranks(x)

## S4 method for signature 'SconeExperiment'
get_scores(x)

## S4 method for signature 'SconeExperiment'
get_score_ranks(x)
```

Arguments

`x`	an object of class `SconeExperiment`.

Value

`get_scores` returns a matrix with all (non-missing) scone scores, ordered by average score rank.

`get_score_ranks` returns a vector of average score ranks.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0,
           eval_kclust=2, bpparam = BiocParallel::SerialParam())
scores = get_scores(res)
score_ranks = get_score_ranks(res)
```

`impute_expectation` *Imputation of zero abundance based on general zero-inflated model*

Description

This function is used to impute the data, weighted by probability of data coming from the zero-inflation part of the distribution.

Usage

```
impute_expectation(expression, impute_args)
```

Arguments

`expression`	the data matrix (genes in rows, cells in columns)
`impute_args`	arguments for imputation (see details)

Details

The imputation is carried out with the following formula: y_ij* = y_ij * Pr(No Drop | y_ij) + mu_i * Pr(Drop | y_ij).

impute_args must contain 2 elements: 1) p_nodrop = posterior probability of data not having resulted from drop-out (genes in rows, cells in columns) 2) mu = expected expression of dropped data (genes in rows, cells in columns)

Value

the imputed expression matrix.

Examples

```
mat <- matrix(rpois(1000, lambda = 3), ncol=10)
mat = mat * matrix(1-rbinom(1000, size = 1, prob = .01), ncol=10)

mu = matrix(rep(3/ppois(0,lambda = 3,lower.tail = FALSE),1000),ncol = 10)

p_false = 1 / ( 1 + ppois(0, lambda = 3, lower.tail = TRUE ) /
    (0.01 * ppois(0, lambda = 3, lower.tail = FALSE) ) )

p_nodrop = matrix(rep(1-p_false,1000),ncol = 10)
p_nodrop[mat > 0] = 1

impute_args = list()
impute_args = list(mu = mu, p_nodrop = p_nodrop)

imat = impute_expectation(mat,impute_args = impute_args)
```

impute_null *Null or no-op imputation*

Description

Null or no-op imputation

Usage

```
impute_null(expression, impute_args)
```

Arguments

expression	the data matrix (genes in rows, cells in columns)
impute_args	arguments for imputation (not used)

Value

the imputed expression matrix.

Examples

```
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
imat = impute_null(mat)
```

lm_adjust *Linear Adjustment Normalization*

Description

Given a matrix with log expression values and a design matrix, this function fits a linear model and removes the effects of the batch factor as well as of the linear variables encoded in W.

Usage

```
lm_adjust(log_expr, design_mat, batch = NULL, weights = NULL)
```

Arguments

log_expr	matrix. The log gene expression (genes in row, samples in columns).
design_mat	matrix. The design matrix (usually the result of make_design).
batch	factor. A factor with the batch information, identifying batch effect to be removed.
weights	matrix. A matrix of weights.

Details

The function assumes that the columns of the design matrix corresponding to the variable for which expression needs to be adjusted, start with either the word "batch" or the letter "W" (case sensitive). Any other covariate (including the intercept) is kept.

Value

The corrected log gene expression.

Examples

```
set.seed(141)
bio = as.factor(rep(c(1,2),each = 2))
batch = as.factor(rep(c(1,2),2))
design_mat = make_design(bio,batch, W = NULL)

log_expr = matrix(rnorm(20),ncol = 4)
adjusted_log_expr = lm_adjust(log_expr = log_expr,
  design_mat = design_mat,
  batch = batch)
```

make_design *Make a Design Matrix*

Description

This function builds a design matrix for the Adjustment Normalization Step, in which covariates are two (possibly nested) categorical factors and one or more continuous variables.

Usage

```
make_design(bio, batch, W, nested = FALSE)
```

Arguments

`bio`	factor. The biological covariate.
`batch`	factor. The batch covariate.
`W`	numeric. Either a vector or matrix containing one or more continuous covariates (e.g. RUVg factors).
`nested`	logical. Whether or not to consider a nested design (see details).

Details

If nested=TRUE a nested design is used, i.e. the batch variable is assumed to be nested within the bio variable. Here, nested means that each batch is composed of samples from only *one* level of bio, while each level of bio may contain multiple batches.

Value

The design matrix.

Examples

```
bio = as.factor(rep(c(1,2),each = 2))
batch = as.factor(rep(c(1,2),2))
design_mat = make_design(bio,batch, W = NULL)
```

metric_sample_filter *Metric-based Sample Filtering: Function to filter single-cell RNA-Seq libraries.*

Description

This function returns a sample-filtering report for each cell in the input expression matrix, describing which filtering criteria are satisfied.

Usage

```
metric_sample_filter(expr, nreads = colSums(expr), ralign = NULL,
  gene_filter = NULL, pos_controls = NULL, scale. = FALSE, glen = NULL,
  AUC_range = c(0, 15), zcut = 1, mixture = TRUE, dip_thresh = 0.05,
  hard_nreads = 25000, hard_ralign = 15, hard_breadth = 0.2,
  hard_auc = 10, suff_nreads = NULL, suff_ralign = NULL,
  suff_breadth = NULL, suff_auc = NULL, plot = FALSE, hist_breaks = 10,
  ...)
```

Arguments

`expr` matrix The data matrix (genes in rows, cells in columns).

`nreads` A numeric vector representing number of reads in each library. Default to ‘colSums‘ of ‘expr‘.

`ralign` A numeric vector representing the proportion of reads aligned to the reference genome in each library. If NULL, filtered_ralign will be returned NA.

`gene_filter` A logical vector indexing genes that will be used to compute library transcriptome breadth. If NULL, filtered_breadth will be returned NA.

`pos_controls` A logical, numeric, or character vector indicating positive control genes that will be used to compute false-negative rate characteristics. If NULL, filtered_fnr will be returned NA.

`scale.` logical. Will expression be scaled by total expression for FNR computation? Default = FALSE

`glen` Gene lengths for gene-length normalization (normalized data used in FNR computation).

`AUC_range` An array of two values, representing range over which FNR AUC will be computed (log(expr_units)). Default c(0,15)

`zcut` A numeric value determining threshold Z-score for sd, mad, and mixture subcriteria. Default 1. If NULL, only hard threshold sub-criteria will be applied.

`mixture` A logical value determining whether mixture modeling sub-criterion will be applied per primary criterion (metric). If true, a dip test will be applied to each metric. If a metric is multimodal, it is fit to a two-component normal mixture model. Samples deviating zcut sd’s from optimal mean (in the inferior direction), have failed this sub-criterion.

`dip_thresh` A numeric value determining dip test p-value threshold. Default 0.05.

`hard_nreads` numeric. Hard (lower bound on) nreads threshold. Default 25000.

`hard_ralign` numeric. Hard (lower bound on) ralign threshold. Default 15.

`hard_breadth` numeric. Hard (lower bound on) breadth threshold. Default 0.2.

`hard_auc` numeric. Hard (upper bound on) fnr auc threshold. Default 10.

`suff_nreads` numeric. If not null, serves as an overriding upper bound on nreads threshold.

`suff_ralign` numeric. If not null, serves as an overriding upper bound on ralign threshold.

`suff_breadth` numeric. If not null, serves as an overriding upper bound on breadth threshold.

`suff_auc` numeric. If not null, serves as an overriding lower bound on fnr auc threshold.

`plot` logical. Should a plot be produced?

`hist_breaks` hist() breaks argument. Ignored if ‘plot=FALSE‘.

`...` Arguments to be passed to methods.

Details

For each primary criterion (metric), a sample is evaluated based on 4 sub-criteria: 1) Hard (encoded) threshold 2) Adaptive thresholding via sd's from the mean 3) Adaptive thresholding via mad's from the median 4) Adaptive thresholding via sd's from the mean (after mixture modeling) A sample must pass all sub-criteria to pass the primary criterion.

Value

A list with the following elements:

- filtered_nreads Logical. Sample has too few reads.
- filtered_ralign Logical. Sample has too few reads aligned.
- filtered_breadth Logical. Samples has too few genes detected (low breadth).
- filtered_fnr Logical. Sample has a high FNR AUC.

Examples

```
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")
mfilt = metric_sample_filter(expr = mat,nreads = qc[,"NCOUNTS"],
   plot = TRUE, hard_nreads = 0)
```

---

scone *Normalize Expression Data and Evaluate Normalization Performance*

Description

This function applies and evaluates a variety of normalization schemes with respect to a specified SconeExperiment containing scRNA-Seq data. Each normalization consists of three main steps:

- Impute: Replace observations of zeroes with expected expression values.
- Scale: Match sample-specific expression scales or quantiles.
- Adjust: Adjust for sample-level batch factors / unwanted variation.

Following completion of each step, the normalized expression matrix is scored based on SCONE's data-driven evaluation criteria.

Usage

```
scone(x, ...)

## S4 method for signature 'SconeExperiment'
scone(x, imputation = list(none = impute_null),
  impute_args = NULL, zero = c("none", "preadjust", "postadjust", "strong"),
  scaling, k_ruv = 5, k_qc = 5, adjust_bio = c("no", "yes", "force"),
  adjust_batch = c("no", "yes", "force"), run = TRUE, evaluate = TRUE,
  eval_pcs = 3, eval_proj = NULL, eval_proj_args = NULL,
  eval_kclust = 2:10, verbose = FALSE, stratified_pam = FALSE,
  stratified_cor = FALSE, stratified_rle = FALSE, return_norm = c("no",
  "in_memory", "hdf5"), hdf5file, bpparam = BiocParallel::bpparam())
```

Arguments

`x` a `SconeExperiment` object.

`...` see specific S4 methods for additional arguments.

`imputation` list or function. (A list of) function(s) to be used for imputation. By default only scone::impute_null is included.

`impute_args` arguments passed to all imputation functions.

`zero` character. Zero-handling option, see Details.

`scaling` list or function. (A list of) function(s) to be used for scaling normalization step.

`k_ruv` numeric. The maximum number of factors of unwanted variation. Adjustment step models will include a range of 1 to k_ruv factors of unwanted variation. If 0, RUV adjustment will not be performed.

`k_qc` numeric. The maximum number of quality metric PCs. Adjustment step models will include a range of 1 to k_qc quality metric PCs. If 0, QC factor adjustment will not be performed.

`adjust_bio` character. If 'no', bio will not be included in Adjustment step models; if 'yes', both models with and without 'bio' will be run; if 'force', only models with 'bio' will be run.

`adjust_batch` character. If 'no', batch will not be included in Adjustment step models; if 'yes', both models with and without 'batch' will be run; if 'force', only models with 'batch' will be run.

`run` logical. If FALSE the normalization and evaluation are not run, but normalization parameters are returned in the output object for inspection by the user.

`evaluate` logical. If FALSE the normalization methods will not be evaluated.

`eval_pcs` numeric. The number of principal components to use for evaluation. Ignored if evaluate=FALSE.

`eval_proj` function. Projection function for evaluation (see `score_matrix` for details). If NULL, PCA is used for projection.

`eval_proj_args` list. List of arguments passed to projection function as eval_proj_args.

`eval_kclust` numeric. The number of clusters (> 1) to be used for pam tightness evaluation. If an array of integers, largest average silhouette width (tightness) will be reported. If NULL, tightness will be returned NA.

`verbose` logical. If TRUE some messagges are printed.

`stratified_pam` logical. If TRUE then maximum ASW for PAM_SIL is separately computed for each biological-cross-batch stratum (accepting NAs), and a weighted average is returned as PAM_SIL.

`stratified_cor` logical. If TRUE then cor metrics are separately computed for each biological-cross-batch stratum (accepts NAs), and weighted averages are returned for EXP_QC_COR, EXP_UV_COR, & EXP_WV_COR. Default FALSE.

`stratified_rle` logical. If TRUE then rle metrics are separately computed for each biological-cross-batch stratum (accepts NAs), and weighted averages are returned for RLE_MED & RLE_IQR. Default FALSE.

`return_norm` character. If "no" the normalized values will not be returned with the output object. This will create a much smaller object and may be useful for large datasets and/or when many combinations are compared. If "in_memory" the normalized values will be returned as part of the output. If "hdf5" they will be written on file using the `rhdf5` package.

hdf5file character. If return_norm="hdf5", the name of the file onto which to save the normalized matrices.

bpparam object of class bpparamClass that specifies the back-end to be used for computations. See bpparam for details.

Details

If run=FALSE only the scone_params slot of the output object is populated with a data.frame, each row corresponding to a set of normalization parameters.

If x has a non-empty scone_params slot, only the subset of normalizations specified in scone_params are performed and evaluated.

The zero arguments supports 3 zero-handling options:

- none: Default. No special zero-handling.
- preadjust: Restore prior zero observations to zero following Impute and Scale steps.
- postadjust: Set prior zero observations and all negative expression values to zero following the Adjust Step.
- strong: Apply both preadjust and postadjust options.

Evaluation metrics are defined in score_matrix. Each metric is assigned a +/- signature for conversion to scores: Positive- signature metrics increase with improving performance, including BIO_SIL, PAM_SIL, and EXP_WV_COR. Negative-signature metrics decrease with improving performance, including BATCH_SIL, EXP_QC_COR, EXP_UV_COR, RLE_MED, and RLE_IQR. Scores are computed so that higer-performing methods are assigned higher scores.

Note that if one wants to include the unnormalized data in the final comparison of normalized matrices, the identity function must be included in the scaling list argument. Analogously, if one wants to include non-imputed data in the comparison, the scone::impute_null function must be included.

If return_norm="hdf5", the normalized matrices will be written to the hdf5file file. This must be a string specifying (a path to) a new file. If the file already exists, it will return error. In this case, the SconeExperiment object will not contain the normalized counts.

If return_norm="no" the normalized matrices are computed to copmute the scores and then discarded.

In all cases, the normalized matrices can be retrieved via the get_normalized function.

Value

A SconeExperiment object with the log-scaled normalized data matrix as elements of the assays slot, if return_norm is "in_memory", and with the performance metrics and scores.

See Also

get_normalized, get_design

Examples

```
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
no_results <- scone(obj, scaling=list(none=identity,
           uq=UQ_FN, deseq=DESEQ_FN),
           run=FALSE, k_ruv=0, k_qc=0, eval_kclust=2)

results <- scone(obj, scaling=list(none=identity,
           uq=UQ_FN, deseq=DESEQ_FN),
           run=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())

results_in_memory <- scone(obj, scaling=list(none=identity,
           uq=UQ_FN, deseq=DESEQ_FN),
           k_ruv=0, k_qc=0, eval_kclust=2,
           return_norm = "in_memory",
           bpparam = BiocParallel::SerialParam())
```

SconeExperiment-class *Class SconeExperiment*

Description

Objects of this class store, at minimum, a gene expression matrix and a set of covariates (sample metadata) useful for running scone. These include, the quality control (QC) metrics, batch information, and biological classes of interest (if available).

The typical way of creating SconeExperiment objects is via a call to the SconeExperiment function or to the scone function. If the object is a result to a scone call, it will contain the results, e.g., the performance metrics, scores, and normalization workflow comparisons. (See Slots for a full list).

This object extends the SummarizedExperiment class.

The constructor SconeExperiment creates an object of the class SconeExperiment.

Usage

```
SconeExperiment(object, ...)

## S4 method for signature 'SummarizedExperiment'
SconeExperiment(object, which_qc = integer(),
  which_bio = integer(), which_batch = integer(),
  which_negconruv = integer(), which_negconeval = integer(),
  which_poscon = integer(), is_log = FALSE)

## S4 method for signature 'matrix'
SconeExperiment(object, qc, bio, batch, negcon_ruv = NULL,
  negcon_eval = negcon_ruv, poscon = NULL, is_log = FALSE)
```

Arguments

object	Either a matrix or a SummarizedExperiment containing the raw gene expression.
...	see specific S4 methods for additional arguments.
which_qc	index that specifies which columns of ‘colData‘ correspond to QC measures.

`which_bio` index that specifies which column of ‘colData‘ corresponds to ‘bio‘.

`which_batch` index that specifies which column of ‘colData‘ corresponds to ‘batch‘.

`which_negconruv` index that specifies which column of ‘rowData‘ has information on negative controls for RUV.

`which_negconeval` index that specifies which column of ‘rowData‘ has information on negative controls for evaluation.

`which_poscon` index that specifies which column of ‘rowData‘ has information on positive controls.

`is_log` are the expression data in log scale?

`qc` numeric matrix with the QC measures.

`bio` factor with the biological class of interest.

`batch` factor with the batch information.

`negcon_ruv` a logical vector indicating which genes to use as negative controls for RUV.

`negcon_eval` a logical vector indicating which genes to use as negative controls for evaluation.

`poscon` a logical vector indicating which genes to use as positive controls.

Details

The QC matrix, biological class, and batch information are stored as elements of the ‘colData‘ of the object.

The positive and negative control genes are stored as elements of the ‘rowData‘ of the object.

Value

A `SconeExperiment` object.

Slots

`which_qc` integer. Index of columns of ‘colData‘ that contain the QC metrics.

`which_bio` integer. Index of the column of ‘colData‘ that contains the biological classes information (it must be a factor).

`which_batch` integer. Index of the column of ‘colData‘ that contains the batch information (it must be a factor).

`which_negconruv` integer. Index of the column of ‘rowData‘ that contains a logical vector indicating which genes to use as negative controls to infer the factors of unwanted variation in RUV.

`which_negconeval` integer. Index of the column of ‘rowData‘ that contains a logical vector indicating which genes to use as negative controls to evaluate the performance of the normalizations.

`which_poscon` integer. Index of the column of ‘rowData‘ that contains a logical vector indicating which genes to use as positive controls to evaluate the performance of the normalizations.

`hdf5_pointer` character. A string specifying to which file to write / read the normalized data.

`imputation_fn` list of functions used by scone for the imputation step.

`scaling_fn` list of functions used by scone for the scaling step.

scone_metrics matrix. Matrix containing the "raw" performance metrics. See scone for a description of each metric.

scone_scores matrix. Matrix containing the performance scores (transformed metrics). See scone for a discussion on the difference between scores and metrics.

scone_params data.frame. A data frame containing the normalization schemes applied to the data and compared.

scone_run character. Whether scone was run and in which mode ("no", "in_memory", "hdf5").

is_log logical. Are the expression data in log scale?

nested logical. Is batch nested within bio? (Automatically set by scone).

rezero logical. TRUE if scone was run with zero="preadjust" or zero="strong".

fixzero logical. TRUE if scone was run with zero="postadjust" or zero="strong".

impute_args list. Arguments passed to all imputation functions.

See Also

get_normalized, get_params, get_batch, get_bio, get_design, get_negconeval, get_negconruv, get_poscon, get_qc, get_scores, and get_score_ranks to access internal fields, select_methods for subsetting by method, and scone for running scone workflows.

Examples

```
set.seed(42)
nrows <- 200
ncols <- 6
counts <- matrix(rpois(nrows * ncols, lambda=10), nrows)
rowdata <- data.frame(poscon=c(rep(TRUE, 10), rep(FALSE, nrows-10)))
coldata <- data.frame(bio=gl(2, 3))
se <- SummarizedExperiment(assays=SimpleList(counts=counts),
                           rowData=rowdata, colData=coldata)

scone1 <- SconeExperiment(assay(se), bio=coldata$bio, poscon=rowdata$poscon)

scone2 <- SconeExperiment(se, which_bio=1L, which_poscon=1L)
```

sconeReport *SCONE Report Browser: Browse Evaluation of Normalization Performance*

Description

This function opens a shiny application session for visualizing performance of a variety of normalization schemes.

Usage

```
sconeReport(x, methods, qc = NULL, bio = NULL, batch = NULL,
  poscon = character(), negcon = character(), eval_proj = NULL,
  eval_proj_args = NULL, subsample_genes = 100, subsample_cells = 100,
  sub_scone = TRUE)
```

Arguments

`x` a `SconeExperiment` object

`methods` character specifying the normalizations to report.

`qc` matrix. QC metrics to be used for QC evaluation report. Required.

`bio` factor. A biological condition (variation to be preserved). Default NULL.

`batch` factor. A known batch variable (variation to be removed). Default NULL.

`poscon` character. Genes to be used as positive controls for evaluation. These genes should be expected to change according to the biological phenomenon of interest. Default empty character.

`negcon` character. Genes to be used as negative controls for evaluation. These genes should be expected not to change according to the biological phenomenon of interest. Default empty character.

`eval_proj` function. Projection function for evaluation (see `score_matrix` for details). If NULL, PCA is used for projection.

`eval_proj_args` list. List of args passed to projection function as eval_proj_args.

Value

An object that represents the SCONE report app.

Examples

```
set.seed(101)
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")
## Not run:
sconeReport(res,rownames(get_params(res)), qc = qc)

## End(Not run)
```

`scone_easybake` *Wrapper for Running Essential SCONE Modules*

Description

Wrapper for Running Essential SCONE Modules

Usage

```
scone_easybake(expr, qc, bio = NULL, batch = NULL, negcon = NULL,
  verbose = c("0", "1", "2"), out_dir = getwd(), seed = 112233,
  filt_cells = TRUE, filt_genes = TRUE, always_keep_genes = NULL,
  fnr_maxiter = 1000, norm_impute = c("yes", "no", "force"),
  norm_scaling = c("none", "sum", "deseq", "tmm", "uq", "fq", "detect"),
  norm_rezero = FALSE, norm_k_max = NULL, norm_qc_expl = 0.5,
  norm_adjust_bio = c("yes", "no", "force"), norm_adjust_batch = c("yes",
  "no", "force"), eval_dim = NULL, eval_expr_expl = 0.1,
  eval_poscon = NULL, eval_negcon = negcon, eval_max_kclust = 10,
  eval_stratified_pam = TRUE, report_num = 13, out_rda = FALSE, ...)
```

Arguments

`expr` matrix. The expression data matrix (genes in rows, cells in columns).

`qc` data frame. The quality control (QC) matrix (cells in rows, metrics in columns) to be used for filtering, normalization, and evaluation.

`bio` factor. The biological condition to be modeled in the Adjustment Step as variation to be preserved. If adjust_bio="no", it will not be used for normalization, but only for evaluation.

`batch` factor. The known batch variable to be included in the adjustment model as variation to be removed. If adjust_batch="no", it will not be used for normalization, but only for evaluation.

`negcon` character. The genes to be used as negative controls for filtering, normalization, and evaluation. These genes should be expressed uniformily across the biological phenomenon of interest. Default NULL.

`verbose` character. Verbosity level: higher level is more verbose. Default "0".

`out_dir` character. Output directory. Default getwd().

`seed` numeric. Random seed. Default 112233.

`filt_cells` logical. Should cells be filtered? Set to FALSE if low quality cells have already been excluded. If cells are not filtered, then initial gene filtering (the one that is done prior to cell filtering) is disabled as it becomes redundant with the gene filtering that is done after cell filtering. Default TRUE.

`filt_genes` logical. Should genes be filtered post-sample filtering? Default TRUE.

`always_keep_genes`
logical. A character vector of gene names that should never be excluded (e.g., marker genes). Default NULL.

`fnr_maxiter` numeric. Maximum number of iterations in EM estimation of expression posteriors. If 0, then FNR estimation is skipped entirely, and as a consequence no imputation will be performed, disregarding the value of the "norm_impute" argument. Default 1000.

`norm_impute` character. Should imputation be included in the comparison? If 'force', only imputed normalizations will be run. Default "yes."

`norm_scaling` character. Scaling options to be included in the Scaling Step. Default c("none", "sum", "deseq", "tmm", "uq", "fq", "detect"). See details.

`norm_rezero` logical. Restore prior zeroes and negative values to zero following normalization. Default FALSE.

`norm_k_max` numeric. Max number (norm_k_max) of factors of unwanted variation modeled in the Adjustment Step. Default NULL.

`norm_qc_expl` numeric. In automatic selection of norm_k_max, what fraction of variation must be explained by the first norm_k_max PCs of qc? Default 0.5. Ignored if norm_k_max is not NULL.

`norm_adjust_bio`
character. If 'no' it will not be included in the model; if 'yes', both models with and without 'bio' will be run; if 'force', only models with 'bio' will be run. Default "yes."

`norm_adjust_batch`
character. If 'no' it will not be modeled in the Adjustment Step; if 'yes', both models with and without 'batch' will be run; if 'force', only models with 'batch' will be run. Default "yes."

`eval_dim` numeric. The number of principal components to use for evaluation. Default NULL.

`eval_expr_expl` numeric. In automatic selection of eval_dim, what fraction of variation must be explained by the first eval_dim PCs of expr? Default 0.1. Ignored if eval_dim is not NULL.

`eval_poscon` character. The genes to be used as positive controls for evaluation. These genes should be expected to change according to the biological phenomenon of interest.

`eval_negcon` character. Alternative negative control gene list for evaluation only.

`eval_max_kclust`
numeric. The max number of clusters (> 1) to be used for pam tightness evaluation. If NULL, tightness will be returned NA.

`eval_stratified_pam`
logical. If TRUE then maximum ASW for PAM_SIL is separately computed for each biological-cross-batch condition (accepting NAs), and a weighted average is returned as PAM_SIL. Default TRUE.

`report_num` numeric. Number of top methods to report. Default 13.

`out_rda` logical. If TRUE, sconeResults.Rda file with the object that the scone function returns is saved in the out_dir (may be very large for large datasets, but useful for post-processing) Default FALSE.

`...` extra params passed to the metric_sample_filter and scone when they're called by easybake

Details

"ADD DESCRIPTION"

Value

Directory structure "ADD DESCRIPTION"

Examples

```
set.seed(101)
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
```

```
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NREADS","RALIGN")
## Not run:
scone_easybake(mat, qc = as.data.frame(qc), verbose = "2",
   norm_adjust_bio= "no",
   norm_adjust_batch= "no", norm_k_max = 0,
   fnr_maxiter = 0, filt_cells=FALSE, filt_genes=FALSE,
   eval_stratified_pam = FALSE,
   out_dir="~/scone_out")

## End(Not run)
```

score_matrix *SCONE Evaluation: Evaluate an Expression Matrix*

Description

This function evaluates a (normalized) expression matrix using SCONE criteria, producing 8 metrics based on i) Clustering, ii) Correlations and iii) Relative Expression.

Usage

```
score_matrix(expr, eval_pcs = 3, eval_proj = NULL, eval_proj_args = NULL,
  eval_kclust = NULL, bio = NULL, batch = NULL, qc_factors = NULL,
  uv_factors = NULL, wv_factors = NULL, is_log = FALSE,
  stratified_pam = FALSE, stratified_cor = FALSE, stratified_rle = FALSE)
```

Arguments

`expr` matrix. The expression data matrix (genes in rows, cells in columns).

`eval_pcs` numeric. The number of principal components to use for evaluation (Default 3). Ignored if !is.null(eval_proj).

`eval_proj` function. Projection function for evaluation (see Details). If NULL, PCA is used for projection

`eval_proj_args` list. List of arguments passed to projection function as eval_proj_args (see Details).

`eval_kclust` numeric. The number of clusters (> 1) to be used for pam tightness (PAM_SIL) evaluation. If an array of integers, largest average silhouette width (tightness) will be reported in PAM_SIL. If NULL, PAM_SIL will be returned NA.

`bio` factor. A known biological condition (variation to be preserved), NA is allowed. If NULL, condition ASW, BIO_SIL, will be returned NA.

`batch` factor. A known batch variable (variation to be removed), NA is allowed. If NULL, batch ASW, BATCH_SIL, will be returned NA.

`qc_factors` Factors of unwanted variation derived from quality metrics. If NULL, qc correlations, EXP_QC_COR, will be returned NA.

`uv_factors`	Factors of unwanted variation derived from negative control genes (evaluation set). If NULL, uv correlations, EXP_UV_COR, will be returned NA.
`wv_factors`	Factors of wanted variation derived from positive control genes (evaluation set). If NULL, wv correlations, EXP_WV_COR, will be returned NA.
`is_log`	logical. If TRUE the expr matrix is already logged and log transformation will not be carried out prior to projection. Default FALSE.
`stratified_pam`	logical. If TRUE then maximum ASW is separately computed for each biological-cross-batch stratum (accepts NAs), and a weighted average silhouette width is returned as PAM_SIL. Default FALSE.
`stratified_cor`	logical. If TRUE then cor metrics are separately computed for each biological-cross-batch stratum (accepts NAs), and weighted averages are returned for EXP_QC_COR, EXP_UV_COR, & EXP_WV_COR. Default FALSE.
`stratified_rle`	logical. If TRUE then rle metrics are separately computed for each biological-cross-batch stratum (accepts NAs), and weighted averages are returned for RLE_MED & RLE_IQR. Default FALSE.

Details

Users may specify their own eval_proj function that will be used to compute Clustering and Correlation metrics. This eval_proj() function must have 2 input arguments:

- e matrix. log-transformed (+ pseudocount) expression data (genes in rows, cells in columns).
- eval_proj_args list. additional function arguments, e.g. prior data weights.

and it must output a matrix representation of the original data (cells in rows, factors in columns). The value of eval_proj_args is passed to the user-defined function from the eval_proj_args argument of the main score_matrix() function call.

Value

A list with the following metrics:

- BIO_SIL Average silhouette width by biological condition.
- BATCH_SIL Average silhouette width by batch condition.
- PAM_SIL Maximum average silhouette width from PAM clustering (see stratified_pam argument).
- EXP_QC_COR Coefficient of determination between expression pcs and quality factors (see stratified_cor argument).
- EXP_UV_COR Coefficient of determination between expression pcs and negative control gene factors (see stratified_cor argument).
- EXP_WV_COR Coefficient of determination between expression pcs and positive control gene factors (see stratified_cor argument).
- RLE_MED The mean squared median Relative Log Expression (RLE) (see stratified_rle argument).
- RLE_IQR The variance of the inter-quartile range (IQR) of the RLE (see stratified_rle argument).

Examples

```
set.seed(141)
bio = as.factor(rep(c(1,2),each = 2))
batch = as.factor(rep(c(1,2),2))
log_expr = matrix(rnorm(20),ncol = 4)

scone_metrics = score_matrix(log_expr,
   bio = bio, batch = batch,
   eval_kclust = 2, is_log = TRUE)
```

SCRAN_FN *Simple deconvolution normalization wrapper*

Description

Simple deconvolution normalization wrapper

Usage

```
SCRAN_FN(ei)
```

Arguments

ei	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling wrapper for `computeSumFactors`).

Value

scran normalized matrix.

Examples

```
ei <- matrix(0:76,nrow = 7)
eo <- SCRAN_FN(ei)
```

select_methods *Get a subset of normalizations from a SconeExperiment object*

Description

This method let a user extract a subset of normalizations. This is useful when the original dataset is large and/or many normalization schemes have been applied.

In such cases, the user may want to run scone in mode `return_norm = "no"`, explore the results, and then select the top performing methods for additional exploration.

Usage

```
select_methods(x, methods)

## S4 method for signature 'SconeExperiment,character'
select_methods(x, methods)

## S4 method for signature 'SconeExperiment,numeric'
select_methods(x, methods)
```

Arguments

`x`	a `SconeExperiment` object.
`methods`	either character or numeric specifying the normalizations to select.

Details

The numeric method will always return the normalization corresponding to the `methods` rows of the `scone_params` slot. This means that if `scone` was run with `eval=TRUE`, `select_methods(x, 1:3)` will return the top three ranked method. If `scone` was run with `eval=FALSE`, it will return the first three normalization in the order saved by scone.

Value

A `SconeExperiment` object with selected method data.

Methods (by class)

- `x = SconeExperiment,methods = character`: If `methods` is a character, it will return the subset of methods named in `methods` (only perfect match). The string must be a subset of the `row.names` of the slot `scone_params`.
- `x = SconeExperiment,methods = numeric`: If `methods` is a numeric, it will return the subset of methods according to the scone ranking.

Examples

```
set.seed(42)
mat <- matrix(rpois(500, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0,
           eval_kclust=2, bpparam = BiocParallel::SerialParam())
select_res = select_methods(res,1:2)
```

simple_FNR_params *Fit Simple False-Negative Model*

Description

Fits a logistic regression model of false negative observations as a function of expression level, using a set of positive control (ubiquitously expressed) genes

Usage

```
simple_FNR_params(expr, pos_controls, fn_tresh = 0.01)
```

Arguments

expr	matrix A matrix of transcript-proportional units (genes in rows, cells in columns).
pos_controls	A logical, numeric, or character vector indicating control genes that will be used to compute false-negative rate characteristics. User must provide at least 2 control genes.
fn_tresh	Inclusive threshold for negative detection. Default 0.01. fn_tresh must be non-negative.

Details

logit(Probability of False Negative) ~ a + b*(median log-expr)

Value

A matrix of logistic regression coefficients corresponding to glm fits in each sample (a and b in columns 1 and 2 respectively). If the a & b fit does not converge, b is set to zero and only a is estimated.

Examples

```
mat <- matrix(rpois(1000, lambda = 3), ncol=10)
mat = mat * matrix(1-rbinom(1000, size = 1, prob = .01), ncol=10)
fnr_out = simple_FNR_params(mat,pos_controls = 1:10)
```

subsample_cells *Function to subsample Scone object by subsampling cells*

Description

This function subsamples a scone object in a number of different ways

Usage

```
subsample_cells(scone_object, percent = 100, at_bio = FALSE, seed = 100,
  verbose = FALSE)
```

Arguments

scone_object	a SconeExperiment object
percent	percent of cells you should take if subsampling purely by percent
at_bio	option to subsample to the number of cells of within the smallest bio group,
seed	the random seed
verbose	option to display more information about subsampling reducing every cell count in a bio group to the same number

Value

The subsampled Scone Object

Examples

```
# Create a scone object
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")

# Subsample the scone object
subsampled_res <- subsample_cells(res,percent=50, at_bio = FALSE, seed = 100, verbose = TRUE)
```

subsample_cells_with_min_bio *Internal function to subsample Scone object by subsampling cells by size of smallest bio group*

Description

This function subsamples a scone object to the size of smallest bio group

Usage

```
subsample_cells_with_min_bio(scone_object, seed = 100, verbose = FALSE)
```

Arguments

scone_object	a SconeExperiment object
seed	the random seed
verbose	option to display more information about subsampling

Value

The subsampled Scone Object

Examples

```
# Create a scone object
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")
res$bio <- c('a', 'a', 'a', 'a', 'a', 'a', 'a', 'b', 'b', 'b')

# Subsample the scone object
subsampled_res <- subsample_cells_with_min_bio(res, seed = 100, verbose = TRUE)
```

subsample_genes *Function to subsample Scone object by subsampling genes*

Description

This function subsamples a scone object in a number of different ways by genes

Usage

```
subsample_genes(scone_object, percent = 100, keep_all_control = TRUE,
  seed = 100, verbose = FALSE)
```

Arguments

`scone_object` a `SconeExperiment` object

`percent` percent of genes you should take if subsampling purely by percent

`keep_all_control` option to keep all control genes and only subsample remaining genes to fill to specificed percent

`seed` the random seed

`verbose` option to display more information about subsampling

Value

The subsampled Scone Object

Examples

```
# Create a scone object
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste("X", 1:ncol(mat), sep="")
obj <- SconeExperiment(mat)
res <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c("NCOUNTS","NGENES")

# Subsample the scone object
subsampled_res <- subsample_genes(res, percent=50, keep_all_control = TRUE, seed = 100, verbose = TRUE)
```

subsample_scone *Wrapper Function to subsample Scone object*

Description

This function subsamples a scone object in a number of different ways of a variety of normalization schemes.

Usage

```
subsample_scone(my_scone, subsample_gene_level = 100,
  subsample_cell_level = 100, cells_first = TRUE, at_bio = TRUE,
  keep_all_control = TRUE, seed = 100, verbose = FALSE)
```

Arguments

`my_scone` a `SconeExperiment` object

`subsample_gene_level` percent of genes you should take if subsampling genes by percent

`subsample_cell_level` percent of cells you should take if subsampling cells by percent

`cells_first` option to control wether to subsample cells and then genes, or the other way around

`at_bio` option to subsample to the number of cells of within the smallest bio group, reducing every cell count in a bio group to the same number

`keep_all_control` option to keep all control genes and only subsample remaining genes to fill to specificed percent,

`seed` the random seed

`verbose` option to display more information about subsampling

Value

The subsampled Scone Object

Examples

```
# Create a scone object
mat <- matrix(rpois(1000, lambda = 5), ncol=10)
colnames(mat) <- paste(”X”, 1:ncol(mat), sep=””)
obj <- SconeExperiment(mat)
scone_object <- scone(obj, scaling=list(none=identity, uq=UQ_FN, deseq=DESEQ_FN),
           evaluate=TRUE, k_ruv=0, k_qc=0, eval_kclust=2,
           bpparam = BiocParallel::SerialParam())
qc = as.matrix(cbind(colSums(mat),colSums(mat > 0)))
rownames(qc) = colnames(mat)
colnames(qc) = c(”NCOUNTS”,”NGENES”)

scone_object$bio <- c('a', 'a', 'a', 'a', 'a', 'a', 'a', 'b', 'b', 'b')

# Subsample the scone object
subsampled_res <- subsample_scone(scone_object, subsample_gene_level = 50, subsample_cell_level = 100,
                                  cells_first=TRUE, at_bio = TRUE,
                                  keep_all_control=TRUE, seed = 100, verbose = TRUE)
```

`SUM_FN` *Sum scaling normalization function*

Description

Sum scaling normalization function

Usage

```
SUM_FN(ei)
```

Arguments

`ei`	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling by library size or summed expression.

Value

Sum-scaled normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- SUM_FN(ei)
```

`TMM_FN` *Weighted trimmed mean of M-values (TMM) scaling normalization wrapper function*

Description

Weighted trimmed mean of M-values (TMM) scaling normalization wrapper function

Usage

```
TMM_FN(ei)
```

Arguments

`ei`	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling wrapper for `calcNormFactors`).

Value

TMM normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- TMM_FN(ei)
```

UQ_FN *Upper-quartile (UQ) scaling normalization wrapper function*

Description

Upper-quartile (UQ) scaling normalization wrapper function

Usage

```
UQ_FN(ei)
```

Arguments

ei	Numerical matrix. (rows = genes, cols = samples).

Details

SCONE scaling wrapper for `calcNormFactors`).

Value

UQ normalized matrix.

Examples

```
ei <- matrix(0:20,nrow = 7)
eo <- UQ_FN(ei)
```

Index