Package 'cellGeometry'

December 11, 2025

Title Geometric Single Cell Deconvolution

Version 0.5.7

URL https://github.com/myles-lewis/cellGeometry

Description
Deconvolution of bulk RNA-Sequencing data into proportions of cells based on a reference single-cell RNA-Sequencing dataset using high-dimensional geometric methodology.

License GPL (>= 3)

Encoding UTF-8

Imports circlize, ComplexHeatmap, DelayedArray, dplyr, ensembldb, ggplot2, ggrepel, grid, gtools, matrixStats, mcprogress, parallel, pbmcapply, rlang, scales

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Suggests future.apply, ggsci, knitr, plotly, Rfast2, rmarkdown, seriation

VignetteBuilder knitr

NeedsCompilation no

Author Myles Lewis [aut, cre] (ORCID: <https://orcid.org/0000-0001-9365-5345>), Rachel Lau [ctb] (ORCID: <https://orcid.org/0000-0002-8330-8048>)

Maintainer Myles Lewis <myles.lewis@qmul.ac.uk>

Repository CRAN

Date/Publication 2025-12-11 19:00:02 UTC

Contents

add_noise 2
adjust_library_size 4
cellMarkers 4
collapse_group 7
comp_heatmap 8

cos_similarity 8
deconvolute 9
diagnose 12
fix_bulk 13
fix_group 13
gene2symbol 14
generate_samples 15
gene_angle 16
logmean 16
mergeMarkers 17
metric_set 18
plot.qqmap 19
plot_comp 19
plot_residuals 20
plot_set 21
plot_tune 22
quantile_map 23
rank_angle 24
reduceNoise 25
residuals.deconv 25
rstudent.deconv 26
scapply 27
scmean 29
signature_heatmap 31
simulate_bulk 32
slapply 33
specificity_plot 34
spillover_heatmap 36
stack_plot 37
summary.tune_deconv 39
tune_deconv 40
updateMarkers 41
violin_plot 43

Index **44**

`add_noise` *Add noise to count data*

Description

Gaussian noise can be added to the simulated count matrix in multiple ways which can be combined.

Usage

```
add_noise(counts, sd = 100)

log_noise(counts, sd = 0.1)

graded_log_noise(counts, sd = 0.1, transform = function(x) x^3)

sqrt_noise(counts, sd = 100)

shift_noise(counts, sd = 0.5, p = 0.5)
```

Arguments

counts	An integer count matrix with genes in rows and cell subclasses typically generated by simulate_bulk().
sd	Standard deviation of noise to be added.
transform	Function for controlling amount of noise by expression level in graded_log_noise().
p	Proportion of genes affected by noise.

Details

- add_noise adds simple Gaussian noise to counts. This affects low expressed genes and hardly affects highly expressed genes.
- With log_noise, counts are converted using log2+1 and Gaussian noise added, followed by conversion back to count scale. This affects all genes irrespective of expression level.
- With graded_log_noise, counts are converted to log2+1. A scaling factor is calculated for gene expression level ranging from 0 to 1, which maps to 0 to the maximum number of counts. This scaling factor is inverted from 1 to 0 (i.e. noise affects low counts more than high counts) and then passed through the function specified by transform (this controls how much the middle counts are affected). Then the Gaussian noise is multiplied by the scaling factor and added to the counts.
- With sqrt_noise, counts are square root transformed before Gaussian noise is added, and then transformed back. This still has a stronger effect on low expressed genes, but the effect is more graduated with a more gradual fall off in effect on genes with increasing expression.
- With shift_noise, whole gene rows are selected at random then each row is multiplied by a random amount varying according to 2^rnorm. This simulates shifted expression up/down due to differences in chemistry through which some genes are more or less detectable.

Value

A positive integer count matrix with genes in rows and cell subclasses in columns.

adjust_library_size *Adjust count matrix by library size*

Description

Simple tool for adjusting raw count matrix by total library size. Library size is calculated as column sums and columns are scaled to the median total library size.

Usage

```
adjust_library_size(x)
```

Arguments

x	Read count matrix with genes in rows and samples in columns.

Value

Matrix of adjusted read counts

cellMarkers *Identify cell markers*

Description

Uses geometric method based on vector dot product to identify genes which are the best markers for individual cell types.

Usage

```
cellMarkers(
  scdata,
  bulkdata = NULL,
  subclass,
  cellgroup = NULL,
  nsubclass = 25,
  ngroup = 10,
  expfilter = 0.5,
  noisefilter = 2,
  noisefraction = 0.25,
  min_cells = 10,
  remove_subclass = NULL,
  dual_mean = FALSE,
  meanFUN = "logmean",
  postFUN = NULL,
  verbose = TRUE,
```

```
  sliceMem = 16,
  cores = 1L,
  ...
)
```

Arguments

scdata	Single-cell data matrix with genes in rows and cells in columns. Can be sparse matrix or DelayedMatrix. Must have rownames representing gene IDs or gene symbols.
bulkdata	Optional data matrix containing bulk RNA-Seq data with genes in rows and samples in columns. This matrix is only used for its rownames (gene IDs), to ensure that cell markers are selected from genes in the bulk dataset.
subclass	Vector of cell subclasses matching the columns in scdata
cellgroup	Optional grouping vector of major cell types matching the columns in scdata. subclass is assumed to contain subclasses which are subsets within cellgroup overarching classes.
nsubclass	Number of genes to select for each single cell subclass. Either a single number or a vector with the number of genes for each subclass.
ngroup	Number of genes to select for each cell group. Either a single number or a vector with the number of genes for each group.
expfilter	Genes whose maximum mean expression on log2 scale per cell type are below this value are removed and not considered for the signature.
noisefilter	Sets an upper bound for noisefraction cut-off below which gene expression is set to 0. Essentially gene expression above this level must be retained in the signature. Setting this higher can allow more suppression via noisefraction and can favour more highly expressed genes.
noisefraction	Numeric value. Maximum mean log2 gene expression across cell types is calculated and values in celltypes below this fraction are set to 0. Set in conjunction with noisefilter. Note: if this is set too high (too close to 1), it can have a deleterious effect on deconvolution.
min_cells	Numeric value specifying minimum number of cells in a subclass category. Subclass categories with fewer cells will be ignored.
remove_subclass	Character vector of subclass levels to be removed from the analysis.
dual_mean	Logical whether to calculate arithmetic mean of counts as well as mean(log2(counts +1)). This is mainly useful for simulation.
meanFUN	Either a character value or function for applying mean which is passed to scmean(). Options include "logmean" (the default) or "trimmean" which is a trimmed after excluding the top/bottom 5% of values.
postFUN	Optional function applied to genemeans matrices after mean has been calculated. If meanFUN is set to "trimmean", then postFUN is set to log2s. See scmean().
verbose	Logical whether to show messages.

sliceMem	Max amount of memory in GB to allow for each subsetted count matrix object. When scdata is subsetted by each cell subclass, if the amount of memory would be above sliceMem then slicing is activated and the subsetted count matrix is divided into chunks and processed separately. This is indicated by addition of '...' in the printed timings. The limit is just under 17.2 GB (2^34 / 1e9). Above this the subsetted matrix breaches the long vector limit (>2^31 elements).
cores	Integer, number of cores to use for parallelisation using mclapply(). Parallelisation is not available on windows. Warning: parallelisation has increased memory requirements. See scmean().
...	Additional arguments passed to scmean() such as use_future.

Details

If verbose = TRUE, the function will display an estimate of the required memory. But importantly this estimate is only a guide. It is provided to help users choose the optimal number of cores during parallelisation. Real memory usage might well be more, theoretically up to double this amount, due to R's use of copy-on-modify.

Value

A list object with S3 class 'cellMarkers' containing:

call	the matched call
best_angle	named list containing a matrix for each cell type with genes in rows. Rows are ranked by lowest specificity angle for that cell type and highest maximum expression. Columns are: angle the specificity angle in radians, angle.deg the same angle in degrees, max the maximum mean expression across all cell types, rank the rank of the mean gene expression for that cell type compared to the other cell types
group_angle	named list of matrices similar to best_angle, for each cell subclass
geneset	character vector of selected gene markers for cell types
group_geneset	character vector of selected gene markers for cell subclasses
genemeans	matrix of mean log2+1 gene expression with genes in rows and cell types in columns
genemeans_filtered	matrix of gene expression for cell types following noise reduction
groupmeans	matrix of mean log2+1 gene expression with genes in rows and cell subclasses in columns
groupmeans_filtered	matrix of gene expression for cell subclasses following noise reduction
cell_table	factor encoded vector containing the groupings of the cell types within cell subclasses, determined by which subclass contains the maximum number of cells for each cell type
spillover	matrix of spillover values between cell types
subclass_table	contingency table of the number of cells in each subclass

opt — list storing options, namely arguments nsubclass, ngroup, expfilter, noisefilter, noisefraction

genemeans_ar — if dual_mean is TRUE, optional matrix of arithmetic mean, i.e. log2(mean(counts)+1)

genemeans_filtered_ar — optional matrix of arithmetic mean following noise reduction

The 'cellMarkers' object is designed to be passed to deconvolute() to deconvolute bulk RNA-Seq data. It can be updated rapidly with different settings using updateMarkers(). Ensembl gene ids can be substituted for recognisable gene symbols by applying gene2symbol().

Author(s)

Myles Lewis

See Also

deconvolute() updateMarkers() gene2symbol()

---

collapse_group *Collapse groups in cellMarkers object*

Description

Experimental function for collapsing groups in a cellMarkers objects.

Usage

```
collapse_group(mk, groups, weights = NULL)
```

Arguments

mk — A 'cellMarkers' class object.

groups — Character vector of groups to be collapsed. The collapsed group retains the name of the 1st element.

weights — Optional vector of weights for calculating the mean gene expression across groups. If left as NULL weights are determined by the total cell count in each group.

Value

An updated cellMarkers class object.

comp_heatmap *Compensation heatmap*

Description

Plots a heatmap of the compensation matrix for cell subclasses using ComplexHeatmap.

Usage

```
comp_heatmap(
  x,
  cell_table = NULL,
  text = NULL,
  cutoff = 0.2,
  fontsize = 8,
  subset = NULL,
  ...
)
```

Arguments

x	object of class 'deconv' or a matrix of compensation values.
cell_table	optional grouping vector to separate the heatmap rows and columns into groups.
text	Logical whether to show values whose absolute value > cutoff. By default only shown for smaller matrices.
cutoff	Absolute threshold for showing values.
fontsize	Numeric value for font size for cell values when text = TRUE.
subset	Character vector of groups to be subsetted.
...	optional arguments passed to ComplexHeatmap::Heatmap()

Value

No return value. Draws a ComplexHeatmap.

cos_similarity *Gene signature cosine similarity matrix*

Description

Computes the cosine similarity matrix from the gene signature matrix of a cellMarkers object or any matrix. Note that this function computes cosine similarity between matrix columns, unlike dist() which computes the distance metric between matrix rows.

Usage

```
cos_similarity(x, use_filter = NULL)
```

Arguments

x	Either a matrix or a 'cellMarkers' class or 'deconv' class object.
use_filter	Logical whether to use filtered gene signature.

Value

A symmetric similarity matrix.

---

deconvolute *Deconvolute bulk RNA-Seq using single-cell RNA-Seq signature*

Description

Deconvolution of bulk RNA-Seq using vector projection method with adjustable compensation for spillover.

Usage

```
deconvolute(
  mk,
  test,
  logged_bulk = FALSE,
  count_space = TRUE,
  comp_amount = 1,
  group_comp_amount = 0,
  weights = NULL,
  weight_method = "equal",
  adjust_comp = TRUE,
  use_filter = TRUE,
  arith_mean = FALSE,
  convert_bulk = FALSE,
  check_comp = FALSE,
  npass = 1,
  outlier_method = c("var.e", "cooks", "rstudent"),
  outlier_cutoff = switch(outlier_method, var.e = 4, cooks = 1, rstudent = 10),
  outlier_quantile = 0.9,
  verbose = TRUE,
  cores = 1L
)
```

Arguments

mk	object of class 'cellMarkers'. See cellMarkers().
test	matrix of bulk RNA-Seq to be deconvoluted with genes in rows and samples in columns. We recommend raw counts as input, but normalised data can be provided, in which case set logged_bulk = TRUE.
logged_bulk	Logical, whether log2 transformed bulk RNA-Seq data is used as input in test.
count_space	Logical, whether deconvolution is performed in count space (as opposed to log2 space). Signature and test revert to count scale by 2^ exponentiation during deconvolution.
comp_amount	either a single value from 0-1 for the amount of compensation or a numeric vector with the same length as the number of cell subclasses to deconvolute.
group_comp_amount	either a single value from 0-1 for the amount of compensation for cell group analysis or a numeric vector with the same length as the number of cell groups to deconvolute.
weights	Optional vector of weights which affects how much each gene in the gene signature matrix affects the deconvolution.
weight_method	Optional. Choices include "none" or "equal" in which gene weights are calculated so that each gene has equal weighting in the vector projection; "equal" overrules any vector supplied by weights.
adjust_comp	logical, whether to optimise comp_amount to prevent negative cell proportion projections.
use_filter	logical, whether to use denoised signature matrix.
arith_mean	logical, whether to use arithmetic means (if available) for signature matrix. Mainly useful with pseudo-bulk simulation.
convert_bulk	either "ref" to convert bulk RNA-Seq to scRNA-Seq scaling using reference data or "qqmap" using quantile mapping of the bulk to scRNA-Seq datasets, or "none" (or FALSE) for no conversion.
check_comp	logical, whether to analyse compensation values across subclasses. See plot_comp().
npass	Number of passes. If npass set to 2 or more this activates removal of genes with excess variance of the residuals.
outlier_method	Method for identifying outlying genes. Options are to use the variance of the residuals for each genes, Cook's distance or absolute Studentized residuals (see details).
outlier_cutoff	Cutoff for removing genes which are outliers based on method selected by outlier_method.
outlier_quantile	Controls quantile for the cutoff for identifying outliers for outlier_method = "cook" or "rstudent".
verbose	logical, whether to show messages.
cores	Number of cores for parallelisation via parallel::mclapply().

Details

Equal weighting of genes by setting `weight_method = "equal"` can help devolution of subclusters whose signature genes have low expression. It is enabled by default.

If a normalised (i.e. logged) bulk matrix is provided instead of raw counts, then it is important that zero expression is true zero. For this reason we do not recommend use of VST (variance stabilised transformed counts) which has a variable offset.

Multipass deconvolution can be activated by setting `npass` to 2 or higher. This is designed to remove genes which behave inconsistently due to noise in either the sc or bulk datasets, which is increasingly likely if you have larger signature geneset, i.e. if `nsubclass` is large. Or you may receive a warning message "Detected genes with extreme residuals". Three methods are available for identifying outlier genes (i.e. whose residuals are too noisy) controlled by `outlier_method`:

- `var.e`, this calculates the variance of the residuals across samples for each gene. Genes whose variance of residuals are outliers based on Z-score standardisation are removed during successive passes.
- `cooks`, this considers the deconvolution as if it were a regression and applies Cook's distance to the residuals and the hat matrix. This seems to be the most stringent method (removes fewest genes).
- `rstudent`, externally Studentized residuals are used.

The cutoff specified by `outlier_cutoff` which is used to determine which genes are outliers is very sensitive to the outlier method. With `var.e` the variances are Z-score scaled. With Cook's distance it is typical to consider a value of >1 as fairly strong indication of an outlier, while 0.5 is considered a possible outlier. With Studentized residuals, these are expected to be on a t distribution scale. However, since gene expression itself does not derive from a normal distribution, the errors and residuals are not normally distributed either, which probably explains the need for a very high cut-off. In practice the choice of settings seems to be dataset dependent.

Value

A list object of S3 class 'deconv' containing:

`call` the matched call

`mk` the original 'cellMarkers' class object

`subclass` list object containing:

- `output`, the amount of each subclass based purely on project gene expression
- `percent`, the proportion of each subclass scaled as a percentage so that the total amount across all subclasses adds to 100%
- `spillover`, the spillover matrix
- `compensation`, the mixed final compensation matrix which incorporates `comp_amount`
- `rawcomp`, the original unadjusted compensation matrix
- `comp_amount`, the final values for the amount of compensation across each cell subclass after adjustment to prevent negative values
- `residuals`, residuals, that is gene expression minus fitted values

- `var.e`, variance of weighted residuals for each gene
- `weights`, vector of weights
- `resvar`, s^2 the estimate of the gene expression variance for each sample
- `se`, standard errors of cell counts
- `hat`, diagonal elements of the hat matrix
- `removed`, vector of outlying genes removed during successive passes

`group`	similar list object to `subclass`, but with results for the cell group analysis.
`nest_output`	alternative matrix of cell output results for each subclass adjusted so that the cell outputs across subclasses are nested as a proportion of cell group outputs.
`nest_percent`	alternative matrix of cell proportion results for each subclass adjusted so that the percentages across subclasses are nested within cell group percentages. The total percentage still adds to 100%.
`comp_amount`	original argument `comp_amount`
`comp_check`	optional list element returned when `check_comp = TRUE`

Author(s)

Myles Lewis

See Also

`cellMarkers()` `updateMarkers()` `rstudent.deconv()` `cooks.distance.deconv()`

`diagnose` *Diagnostics for cellMarker signatures*

Description

Diagnostic tool which prints information for identifying cell subclasses or groups with weak signatures.

Usage

```
diagnose(object, group = NULL, angle_cutoff = 30, weak = 2)
```

Arguments

`object`	A 'cellMarkers' or 'deconv' class object.
`group`	Character vector to focus on cell subclasses within a particular group or groups.
`angle_cutoff`	Angle in degrees below which cell cluster vectors are considered to overlap too much. Range 0-90. See `cos_similarity()`.
`weak`	Number of 1st ranked genes for each cell cluster at which/below its gene set is considered weak.

Value

No return value. Prints information about the cellMarkers signature showing cells subclasses with weak signatures and diagnostic information including which cell subclasses each problematic signature spills into.

---

fix_bulk *Fix in missing genes in bulk RNA-Seq matrix*

Description

Fills in missing genes in a bulk RNA-Seq matrix based on the gene signature of a 'cellMarkers' objects. Signature is taken from both the subclass gene set and group gene set.

Usage

```
fix_bulk(bulk, mk)
```

Arguments

bulk	matrix of bulk RNA-Seq
mk	object of class 'cellMarkers'. See cellMarkers().

Details

This is a convenience function if you have an existing cellMarkers signature object and you do not want to remove genes from the existing signatures by running updateMarkers() with the desired bulk data, and are prepared to accept the assumption that genes which are missing in the bulk RNA-Seq dataset have zero expression. We recommend you check which signature genes are missing from the bulk data first.

Value

Expanded bulk matrix with extra rows for missing genes, filled with zeros.

---

fix_group *Fix cellMarkers signature with no cell groups*

Description

This function is designed to fix cellMarkers objects which were not created with a cellgroup vector and therefore have no cell grouping categories. This can cause issues during merging of cellMarkers objects.

Usage

```
fix_group(mk, lab)
```

Arguments

mk	A 'cellMarkers' class object.
lab	Character value to label the overarching group.

Value

A 'cellMarkers' class list object in which the elements cell_table and groupmeans have been updated.

See Also

mergeMarkers()

---

gene2symbol *Converts ensembl gene ids to symbols*

Description

Uses a loaded ensembl database to convert ensembl gene ids to symbol. If a vector is provided, a vector of symbols is returned. If a cellMarkers object is provided, the rownames in the genemeans, genemeans_filtered, groupmeans and groupmeans_filtered elements are changed to symbol and the cellMarkers object is returned.

Usage

```
gene2symbol(x, ensdb, dups = c("omit", "pass"))
```

Arguments

x	Either a vector of ensembl gene ids to convert or a 'cellMarkers' class object.
ensdb	An ensembl database object loaded via the AnnotationHub bioconductor package.
dups	Character vector specifying action for duplicated gene symbols. "omit" means that duplicated gene symbols are not replaced, but left as ensembl gene ids. "pass" means that all gene ids are replaced where possible even if that leads to duplicates. Duplicates can cause problems with rownames and updateMarkers() in particular.

Value

If x is a vector, a vector of symbols is returned. If no symbol is available for particular ensembl id, the id is left untouched. If x is a 'cellMarkers' class object, a 'cellMarkers' object is returned with rownames in the results elements and genesets converted to gene symbols, and an extra element symbol containing a named vector of converted genes.

See Also

cellMarkers()

`generate_samples` *Generate random cell number samples*

Description

Used for simulating pseudo-bulk RNA-Seq from a 'cellMarkers' object. Cell counts are randomly sampled from the uniform distribution, using the original subclass contingency table as a limit on the maximum number of cells in each subclass.

Usage

```
generate_samples(
  object,
  n,
  equal_sample = TRUE,
  method = c("unif", "dirichlet"),
  alpha = 1.5
)
```

Arguments

`object`	A 'cellMarkers' class object
`n`	Integer value for the number of samples to generate
`equal_sample`	Logical whether to sample subclasses equally or generate samples with proportions of cells in keeping with the original subtotal of cells in the main scRNA-Seq data.
`method`	Either "unif" or "dirichlet" to specify whether cell numbers are drawn from uniform distribution or dirichlet distribution.
`alpha`	Shape parameter for `gtools::rdirichlet()`. Automatically expanded to be a vector whose length is the number of subclasses.

Details

Leaving `equal_sample = TRUE` is better for tuning deconvolution parameters.

Value

An integer matrix with `n` rows, with columns for each cell subclasses in `object`, representing cell counts for each cell subclass. Designed to be passed to `simulate_bulk()`.

See Also

`simulate_bulk()`

gene_angle *Vector based best marker selection*

Description

Core function which takes a matrix of mean gene expression (assumed to be log2 transformed to be more Gaussian). Mean gene expression per gene is scaled to a unit hypersphere assuming each gene represents a vector in space with dimensions representing each cell subclass/group.

Usage

```
gene_angle(genemeans)
```

Arguments

genemeans	matrix of mean gene expression with genes in rows and celltypes, tissues or subclasses in columns.

Value

a list whose length is the number of columns in genemeans, with each element containing a dataframe with genes in rows, sorted by best marker status as determined by minimum vector angle and highest maximum gene expression per celltype/tissue.

logmean *Mean Objects*

Description

Functions designed for use with `scmean()` to calculate mean gene expression in each cell cluster across matrix rows.

Usage

```
logmean(x)

trimmean(x)

log2s(x)
```

Arguments

x	A count matrix

Value

Numeric vector of mean values.

`logmean` applies `log2(x+1)` then calculates `rowMeans`.

`trimmean` applies a trimmed mean to each row of gene counts, excluding the top and bottom 5% of values which helps to exclude outliers. Note, this needs the `Rfast2` package to be installed. When `trimmean` is used with `scmean()`, `postFUN` is typically set to `log2s`. This simply applies log2(x+1) after the trimmed mean of counts has been calculated.

`mergeMarkers` *Merge cellMarker signatures*

Description

Takes 2 cellMarkers signatures, merges them and recalculates optimal gene signatures.

Usage

```
mergeMarkers(
  mk1,
  mk2,
  remove_subclass = NULL,
  remove_group = NULL,
  transform = c("qq", "linear.qq", "scale", "none"),
  scale = 1,
  ...
)
```

Arguments

`mk1`	The reference 'cellMarkers' class object.
`mk2`	A 'cellMarkers' class object containing cell signatures to merge into `mk1`.
`remove_subclass`	Optional character vector of subclasses to remove when merging.
`remove_group`	Optional character vector of cell groups to remove when merging.
`transform`	Either "qq" which applies `quantile_map()` to `mk2` to quantile transform it onto the same distribution as `mk1`, "linear.qq", which determines the quantile transformation and then applies a linear approximation of this, "scale" which simply scales the gene expression by the value `scale`, or "none" for no transformation.
`scale`	Numeric value determining the scaling factor for `mk2` if `transform` is set to "scale".
`...`	Optional arguments and settings passed to `updateMarkers()`.

Value

A list object of S3 class 'cellMarkers'. See `cellMarkers()` for details. If `transform = "qq"` then an additional element `qqmerge` is returned containing the quantile mapping function between the 2 datasets.

See Also

`cellMarkers()` `updateMarkers()` `quantile_map()`

`metric_set` *Calculate R-squared and metrics on deconvoluted cell subclasses*

Description

Calculates Pearson r-squared, R-squared and RMSE comparing subclasses in each column of `obs` with matching columns in deconvoluted `pred`. Samples are in rows. For use if ground truth is available, e.g. simulated pseudo-bulk RNA-Seq data.

Usage

```
metric_set(obs, pred)
```

Arguments

`obs`	Observed matrix of cell amounts with subclasses in columns and samples in rows.
`pred`	Predicted (deconvoluted) matrix of cell amounts with rows and columns matching `obs`.

Details

Pearson r-squared ranges from 0 to 1. R-squared, calculated as 1 - rss/tss, ranges from -Inf to 1.

Value

Matrix containing Pearson r-squared, R-squared and RMSE values.

plot.qqmap *Quantile-quantile plot*

Description

Produces a QQ plot showing the conversion function from the first dataset to the second.

Usage

```
## S3 method for class 'qqmap'
plot(x, points = TRUE, ...)
```

Arguments

x	A 'qqmap' class object created by quantile_map().
points	Logical whether to show quantile points.
...	Optional plotting parameters passed to plot().

Value

No return value. Produces a QQ plot using base graphics with a red line showing the conversion function.

plot_comp *Plot compensation analysis*

Description

Plots the effect of varying compensation from 0 to 1 for each cell subclass, examining the minimum subclass output result following a call to deconvolute(). For this function to work, the argument plot_comp must be set to TRUE during the call to deconvolute().

Usage

```
plot_comp(x, overlay = TRUE, mfrow = NULL, ...)
```

Arguments

x	An object of class 'deconv' generated by deconvolute().
overlay	Logical whether to overlay compensation curves onto a single plot.
mfrow	Optional vector of length 2 for organising plot layout. See par(). Only used when overlay = FALSE.
...	Optional graphical arguments passed to plot().

Value

No return value, plots the effect of varying compensation on minimum subclass output for each cell subclass.

`plot_residuals` *Residuals plot*

Description

Plots residuals from a deconvolution result object against bulk gene expression (on semi-log axis). Normal residuals, weighted residuals or Studentized residuals can be visualised to check for heteroscedasticity and genes with extreme errors.

Usage

```
plot_residuals(
  fit,
  test,
  type = c("reg", "student", "weight"),
  show_outliers = TRUE,
  show_plot = TRUE,
  ...
)

ggplot_residuals(
  fit,
  test,
  type = c("reg", "student", "weight"),
  show_outliers = TRUE
)
```

Arguments

`fit`	'deconv' class deconvolution object
`test`	bulk gene expression matrix assumed to be in raw counts
`type`	Specifies type of residuals to be plotted
`show_outliers`	Logical whether to show any remaining outlying extreme genes in red
`show_plot`	Logical whether to show plot using base graphics (used to allow return of dataframe of points without plotting)
`...`	Optional arguments passed to `plot()`

Value

Produces a scatter plot in base graphics. Returns invisibly a dataframe of the coordinates of the points. The ggplot version returns a ggplot2 plotting object.

plot_set *Scatter plots to compare deconvoluted subclasses*

Description

Produces scatter plots using base graphics to compare actual cell counts against deconvoluted cell counts from bulk (or pseudo-bulk) RNA-Seq. Mainly for use if ground truth is available, e.g. for simulated pseudo-bulk RNA-Seq data.

Usage

```
plot_set(
  obs,
  pred,
  mfrow = NULL,
  show_zero = FALSE,
  show_identity = FALSE,
  cols = NULL,
  colour = "blue",
  title = "",
  cex.title = 1,
  ...
)
```

Arguments

obs	Observed matrix of cell amounts with subclasses in columns and samples in rows.
pred	Predicted (deconvoluted) matrix of cell amounts with rows and columns matching obs.
mfrow	Optional vector of length 2 for organising plot layout. See par().
show_zero	Logical whether to force plot to include the origin.
show_identity	Logical whether to show the identity line.
cols	Optional vector of column indices to plot to show either a subset of columns or change the order in which columns are plotted. NA skips a plot space to introduce a gap between plots.
colour	Colour for the regression lines.
title	Title for page of plots.
cex.title	Font size for title.
...	Optional arguments passed to plot().

Value

No return value. Produces scatter plots using base graphics.

plot_tune *Plot tuning curves*

Description

Produces a ggplot2 plot of R-squared/RMSE values generated by `tune_deconv()`.

Usage

```
plot_tune(
  result,
  group = "subclass",
  xvar = colnames(result)[1],
  fix = NULL,
  metric = attr(result, "metric"),
  title = NULL
)
```

Arguments

`result`	Dataframe of tuning results generated by `tune_deconv()`.
`group`	Character value specifying column in `result` to be grouped by colour; or `NULL` to average R-squared/RMSE values across the grid and show the generalised mean effect of varying the parameter specified by `xvar`.
`xvar`	Character value specifying column in `result` to vary along the x axis.
`fix`	Optional list specifying parameters to be fixed at specific values.
`metric`	Specifies tuning metric: either "RMSE", "Rsq" or "pearson".
`title`	Character value for the plot title.

Details

If `group` is set to `"subclass"`, then the tuning parameter specified by `xvar` is varied on the x axis. Any other tuning parameters (i.e. if 2 or more have been tuned) are fixed to their best tuned values.

If `group` is set to a different column than `"subclass"`, then the mean R-squared/RMSE values in `result` are averaged over subclasses. This makes it easier to compare the overall effect (mean R-squared/RMSE) of 2 tuned parameters which are specified by `xvar` and `group`. Any remaining parameters not shown are fixed to their best tuned values.

If `group` is `NULL`, the tuning parameter specified by `xvar` is varied on the x axis and R-squared/RMSE values are averaged over the whole grid to give the generalised mean effect of varying the `xvar` parameter.

Value

ggplot2 scatter plot.

quantile_map *Quantile mapping function between two scRNA-Seq datasets*

Description

Quantile mapping to combine two scRNA-Seq datasets based on mapping either the distribution of mean log2+1 gene expression in cell clusters to the distribution of the 2nd dataset, or mapping the quantiles of one matrix of gene expression (with genes in rows) to another.

Usage

```
quantile_map(
  x,
  y,
  n = 10000,
  remove_noncoding = TRUE,
  remove_zeros = FALSE,
  smooth = "loess",
  span = 0.15,
  knots = c(0.25, 0.75, 0.85, 0.95, 0.97, 0.99, 0.999),
  respace = FALSE,
  silent = FALSE
)
```

Arguments

x	scRNA-Seq data whose distribution is to be mapped onto y: either a matrix of gene expression on log2+1 scale, or a 'cellMarkers' class object, in which case the $genemeans list element is extracted.
y	Reference scRNA-Seq data: either a matrix of gene expression on log2+1 scale, or a 'cellMarkers' class object, in which case the $genemeans list element is extracted.
n	Number of quantiles to split x and y.
remove_noncoding	Logical, whether to remove noncoding. This is a basic filter which looks at the gene names (rownames) in both matrices and removes genes containing "-" which are usually antisense or mitochondrial genes, or "." which are either pseudogenes or ribosomal genes.
remove_zeros	Logical, whether to remove zeros from both datasets. This shifts the quantile relationships.
smooth	Either "loess" or "lowess" which apply loess() or lowess() to smooth the QQ fitted line, or "ns" which uses natural splines via ns(). With any other value no smoothing is applied. With no smoothing or "loess/lowess", interpolation is limited to the original range of x, i.e. it will clip for values > max(x).
span	controls the degree of smoothing in loess() and lowess().

knots	Vector of quantile points for knots for fitting natural splines.
respace	Logical whether to respace quantile points so their x axis density is more even. Can help spline fitting.
silent	Logical whether to suppress messages.

Details

The conversion uses the function approxfun() which uses interpolation. It is not designed to perform stepwise (exact) quantile transformation of every individual datapoint.

Value

A list object of class 'qqmap' containing:

quantiles	Dataframe containing matching quantiles of x and y
map	A function of form FUN(x) where x can be supplied as a numeric vector or matrix and the same type is returned. The function converts given data points to the distribution of y.

See Also

approxfun()

---

rank_angle *Rank distance angles from a cosine similarity matrix*

Description

Converts a cosine similarity matrix to angular distance. Then orders the elements in increasing angle. Elements below angle_cutoff are returned in a dataframe.

Usage

```
rank_angle(x, angle_cutoff = 45)
```

Arguments

x	a cosine similarity matrix generated by cos_similarity().
angle_cutoff	Cutoff angle in degrees below which to subset the dataframe.

Value

a dataframe of rows and columns as factors and the angle between that row and column extracted from the cosine similarity matrix. Row and column location are stored as factors so that they can be converted back to coordinates in the similarity matrix easily using as.integer().

reduceNoise *Reduce noise in single-cell data*

Description

Simple filter for removing noise in single-cell data.

Usage

```
reduceNoise(cellmat, noisefilter = 2, noisefraction = 0.25)
```

Arguments

cellmat	Matrix of log2 mean gene expression in rows with cell types in columns.
noisefilter	Sets an upper bound for noisefraction cut-off below which gene expression is set to 0. Essentially gene expression above this level must be retained in the signature. Setting this higher can allow more suppression via noisefraction and can favour more highly expressed genes.
noisefraction	Numeric value. Maximum mean log2 gene expression across cell types is calculated and values in celltypes below this fraction are set to 0. Set in conjunction with noisefilter. Note: if this is set too high (too close to 1), it can have a deleterious effect on deconvolution.

Value

Filtered mean gene expression matrix with genes in rows and cell types in columns.

residuals.deconv *Extract Deconvolution Residuals*

Description

Extracts residuals from a deconvolution model. As the model uses a reduced signature gene set for deconvolution, in order to extract residuals for all genes, these need to recalculated by supplying the bulk count matrix test.

Usage

```
## S3 method for class 'deconv'
residuals(object, test = NULL, arith_mean = FALSE, use_filter = FALSE, ...)
```

Arguments

object	a 'deconv' class object
test	bulk gene expression matrix assumed to be in raw counts
arith_mean	logical, whether to use arithmetic mean as gene signature
use_filter	logical, whether to use denoised signature matrix
...	retained for class compatibility

Value

Matrix of residuals.

---

rstudent.deconv *Regression Deletion Diagnostics*

Description

Functions for computing regression diagnostics including standardised or Studentized residuals as well as Cook's distance.

Usage

```
## S3 method for class 'deconv'
rstudent(model, ...)

## S3 method for class 'deconv'
rstandard(model, ...)

## S3 method for class 'deconv'
cooks.distance(model, ...)
```

Arguments

model	'deconv' class object
...	retained for class compatibility

Details

Residuals are first adjusted for gene weights (if used). `rstandard` and `rstudent` give standardized and Studentized residuals respectively. Standardised residuals are calculated based on the hat matrix:

$$H = X(X^T X)^{-1} X^T$$

Leverage $h_{ii} = diag(H)$ is used to standardise the residuals:

$$t_i = \frac{\hat{\varepsilon}_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}$$

Studentized residuals are calculated based on excluding the i th case. Note this corresponds to refitting the regression, but without recomputing the non-negative compensation matrix. Cook's distance is calculated as:

$$D_i = \frac{e_i^2}{ps^2}\left[\frac{h_{ii}}{(1-h_{ii})^2}\right]$$

where p is the number of predictors (cell subclasses) and s^2 is the mean squared error. In this model the intercept is not included.

Value

Matrix of adjusted residuals or Cook's distance.

See Also

`stats::influence.measures()`

`scapply` *Single-cell apply a function to a matrix split by a factor*

Description

Workhorse function designed to handle large scRNA-Seq gene expression matrices such as embedded Seurat matrices, and apply a function to columns of the matrix split as a ragged array by an index factor, similar to `tapply()`, `by()` or `aggregate()`. Note that here the index is applied to columns as these represent cells in the single-cell format, rather than rows as in `aggregate()`. Very large matrices are handled by slicing rows into blocks to avoid excess memory requirements.

Usage

```
scapply(
  x,
  INDEX,
  FUN,
  combine = NULL,
  combine2 = "c",
  progress = TRUE,
  sliceMem = 16,
  cores = 1L,
  ...
)
```

Arguments

`x`	matrix, sparse matrix or DelayedMatrix of raw counts with genes in rows and cells in columns.
`INDEX`	a factor whose length matches the number of columns in `x`. It is coerced to a factor. `NA` are tolerated and the matching columns in `x` are skipped.

FUN	Function to be applied to each subblock of the matrix.
combine	A function or a name of a function to apply to the list output to bind the final results together, e.g. 'cbind' or 'rbind' to return a matrix, or 'unlist' to return a vector.
combine2	A function or a name of a function to combine results after slicing. As the function is usually applied to blocks of 30000 genes or so, the result is usually a vector with an element per gene. Hence 'c' is the default function for combining vectors into a single longer vector. However if each gene returns a number of results (e.g. a vector or dataframe), then combine2 could be set to 'rbind'.
progress	Logical, whether to show progress.
sliceMem	Max amount of memory in GB to allow for each subsetted count matrix object. When x is subsetted by each cell subclass, if the amount of memory would be above sliceMem then slicing is activated and the subsetted count matrix is divided into chunks and processed separately. The limit is just under 17.2 GB (2^34 / 1e9). At this level the subsetted matrix breaches the long vector limit (>2^31 elements).
cores	Integer, number of cores to use for parallelisation using mclapply(). Parallelisation is not available on windows. Warning: parallelisation increases the memory requirement by multiples of sliceMem.
...	Optional arguments passed to FUN.

Details

The limit on sliceMem is that the number of elements manipulated in each block must be kept below the long vector limit of 2^31 (around 2e9). Increasing cores requires substantial amounts of spare RAM. combine works in a similar way to .combine in foreach(); it works across the levels in INDEX. combine2 is nested and works across slices of genes (an inner loop), so it is only invoked if slicing occurs which is when a matrix has a larger memory footprint than sliceMem.

Value

By default returns a list, unless combine is invoked in which case the returned data type will depend on the functions specified by FUN and combine.

Author(s)

Myles Lewis

See Also

scmean() which applies a fixed function logmean() in a similar manner, and slapply() which applies a function to a big matrix with slicing but without splitting by an index factor.

Examples

```
# equivalent
m <- matrix(sample(0:100, 1000, replace = TRUE), nrow = 10)
cell_index <- sample(letters[1:5], 100, replace = TRUE)
```

```
o <- scmean(m, cell_index)
o2 <- scapply(m, cell_index, function(x) rowMeans(log2(x +1)),
              combine = "cbind")
identical(o, o2)
```

---

scmean *Single-cell mean log gene expression across cell types*

Description

Workhorse function which takes as input a scRNA-Seq gene expression matrix such as embedded in a Seurat object, calculates log2(counts +1) and averages gene expression over a vector specifying cell subclasses or cell types. Very large matrices are handled by slicing rows into blocks to avoid excess memory requirements.

Usage

```
scmean(
  x,
  celltype,
  FUN = "logmean",
  postFUN = NULL,
  verbose = TRUE,
  sliceMem = 16,
  cores = 1L,
  load_balance = FALSE,
  use_future = FALSE
)
```

Arguments

x	matrix, sparse matrix or DelayedMatrix of raw counts with genes in rows and cells in columns.
celltype	a vector of cell subclasses or types whose length matches the number of columns in x. It is coerced to a factor. NA are tolerated and the matching columns in x are skipped.
FUN	Character value or function for applying mean. When applied to a matrix of count values, this must return a vector. Recommended options are "logmean" (the default) or "trimmean".
postFUN	Optional function to be applied to whole matrix after mean has been calculated, e.g. log2s.
verbose	Logical, whether to print messages.

sliceMem	Max amount of memory in GB to allow for each subsetted count matrix object. When x is subsetted by each cell subclass, if the amount of memory would be above sliceMem then slicing is activated and the subsetted count matrix is divided into chunks and processed separately. This is indicated by addition of '...' in the timings. The limit is just under 17.2 GB (2^34 / 1e9). At this level the subsetted matrix breaches the long vector limit (>2^31 elements).
cores	Integer, number of cores to use for parallelisation using mclapply(). Parallelisation is not available on windows. Warning: parallelisation increases the memory requirement by multiples of sliceMem. cores is ignored if use_future = TRUE.
load_balance	Logical, whether to load balance memory requirements across cores (experimental).
use_future	Logical, whether to use the future backend for parallelisation via future_lapply() instead of the default which is mclapply(). Note, the future.apply package needs to be installed to enable this.

Details

Mean functions which can be applied by setting FUN include logmean (the default) which applies row means to log2(counts+1), or trimmean which calculates the trimmed mean of the counts after top/bottom 5% of values have been excluded. Alternatively FUN = rowMeans calculates the arithmetic mean of counts.

If FUN = trimmean or rowMeans, postFUN needs to be set to log2s which is a simple function which applies log2(x+1).

sliceMem can be set lower on machines with less RAM, but this will slow the analysis down. cores increases the theoretical amount of memory required to around cores * sliceMem in GB. For example on a 64 GB machine, we find a significant speed increase with cores = 3L. Above this level, there is a risk that memory swap will slow down processing.

Value

a matrix of mean log2 gene expression across cell types with genes in rows and cell types in columns.

Author(s)

Myles Lewis

See Also

scapply() which is a more general version which can apply any function to the matrix. logmean, trimmean are options for controlling the type of mean applied.

signature_heatmap *Gene signature heatmap*

Description

Produces a heatmap of genes signatures for each cell subclass using ComplexHeatmap.

Usage

```
signature_heatmap(
  x,
  type = c("subclass", "group", "groupsplit"),
  top = Inf,
  use_filter = NULL,
  arith_mean = FALSE,
  rank = c("max", "angle"),
  scale = c("none", "max", "sphere"),
  col = rev(hcl.colors(10, "Greens3")),
  text = TRUE,
  fontsize = 6.5,
  outlines = FALSE,
  outline_col = "black",
  subset = NULL,
  add_genes = NULL,
  ...
)
```

Arguments

x	Either a gene signature matrix with genes in rows and cell subclasses in columns, an object of S3 class 'cellMarkers' generated by cellMarkers(), or an object of class 'deconv' generated by deconvolute().
type	Either "subclass" or "group" specifying whether to show the cell subclass or cell group signature from a 'cellMarkers' or 'deconv' object. "groupsplit" shows the distribution of mean gene expression for the group signature across subclasses.
top	Specifies the number of genes per subclass/group to be displayed.
use_filter	Logical whether to show denoised gene signature.
arith_mean	Logical whether to show log2(arithmetic mean), if calculated, instead of usual mean(log2(counts +1)).
rank	Either "max" or "angle" controlling whether genes (rows) are ordered in the heatmap by max expression (the default) or lowest angle (a measure of specificity of the gene as a cell marker).
scale	Character value controlling scaling of genes: "none" for no scaling, "max" to equalise the maximum mean expression between genes, "sphere" to scale genes to the unit hypersphere where cell subclasses or groups are dimensions.

col	Vector of colours passed to ComplexHeatmap::Heatmap().
text	Logical whether to show values of the maximum cell in each row.
fontsize	Numeric value for font size for cell values when text = TRUE.
outlines	Logical whether to outline boxes with maximum values in each row. This supercedes text.
outline_col	Colour for the outline boxes when outlines = TRUE.
subset	Character vector of groups to be subsetted.
add_genes	Character vector of gene names to be added to the heatmap.
...	Optional arguments passed to ComplexHeatmap::Heatmap().

Value

A 'Heatmap' class object.

---

simulate_bulk *Simulate pseudo-bulk RNA-Seq*

Description

Simulates pseudo-bulk RNA-Seq dataset using two modes. The first mode uses a 'cellMarkers' class object and a matrix of counts for the numbers of cells of each cell subclass. This method converts the log2 gene means back for each cell subclass back to count scale and then calculates pseudo-bulk count values based on the cell amounts specified in samples. In the 2nd mode, a single-cell RNA-Seq dataset is required, such as a matrix used as input to cellMarkers(). Cells from the relevant subclass are sampled from the single-cell matrix in the appropriate amounts based on samples, except that sampling is scaled up by the factor times.

Usage

```
simulate_bulk(
  object,
  samples,
  subclass,
  times = 1,
  method = c("dirichlet", "unif"),
  alpha = 1
)
```

Arguments

object	Either a 'cellMarkers' class object, or a single cell count matrix with genes in rows and cells in columns, with rownames representing gene IDs/symbols. The matrix can be a sparse matrix or DelayedMatrix.
samples	An integer matrix of cell counts with samples in rows and columns for each cell subclass in object. This can be generated using generate_samples().

subclass	Vector of cell subclasses matching the columns in object. Only used if object is a single cell count matrix.
times	Scaling factor to increase sampling of cells. Cell counts in samples are scaled up by being multiplied by this number. Only used if object is a single cell count matrix.
method	Either "dirichlet" or "unif" to specify whether cells are sampled based on the Dirichlet distribution with K = number of cells in each subclass, or sampled uniformly. When cells are oversampled uniformly, in the limit the summed gene expression tends to the arithmetic mean of the subclass x sample frequency. Dirichlet sampling provides proper randomness with sampling.
alpha	Shape parameter for Dirichlet sampling.

Details

The first method can give perfect deconvolution if the following settings are used with deconvolute(): count_space = TRUE, convert_bulk = FALSE, use_filter = FALSE and comp_amount = 1.

Value

An integer count matrix with genes in rows and cell subclasses in columns. This can be used as test with the deconvolute() function.

See Also

generate_samples() deconvolute() add_noise()

---

slapply *Apply a function to a big matrix by slicing*

Description

Workhorse function ('slice apply') designed to handle large scRNA-Seq gene expression matrices such as embedded Seurat matrices, and apply a function to the whole matrix. Very large matrices are handled by slicing rows into blocks to avoid excess memory requirements.

Usage

```
slapply(x, FUN, combine = "c", progress = TRUE, sliceMem = 16, cores = 1L, ...)
```

Arguments

x	matrix, sparse matrix or DelayedMatrix of raw counts with genes in rows and cells in columns.
FUN	Function to be applied to each subblock of the matrix.

combine	A function or a name of a function to combine results after slicing. As the function is usually applied to blocks of 30000 genes or so, the result is usually a vector with an element per gene. Hence 'c' is the default function for combining vectors into a single longer vector. However if each gene row returns a number of results (e.g. a vector or dataframe), then combine could be set to 'rbind'.
progress	Logical, whether to show progress.
sliceMem	Max amount of memory in GB to allow for each subsetted count matrix object. When x is subsetted by each cell subclass, if the amount of memory would be above sliceMem then slicing is activated and the subsetted count matrix is divided into chunks and processed separately. The limit is just under 17.2 GB (2^34 / 1e9). At this level the subsetted matrix breaches the long vector limit (>2^31 elements).
cores	Integer, number of cores to use for parallelisation using mclapply(). Parallelisation is not available on windows. Warning: parallelisation has increased memory requirements.
...	Optional arguments passed to FUN.

Details

The limit on sliceMem is that the number of elements manipulated in each block must be kept below the long vector limit of 2^31 (around 2e9). Increasing cores requires substantial amounts of spare RAM. combine works in a similar way to .combine in foreach() across slices of genes; it is only invoked if slicing occurs.

Value

The returned data type will depend on the functions specified by FUN and combine.

Author(s)

Myles Lewis

See Also

scapply()

---

specificity_plot *Specificity plot*

Description

Scatter plot showing specificity of genes as markers for a particular cell subclass. Optimal gene markers for that cell subclass are those genes which are closest to or lie on the y axis, while also being of highest mean expression.

Usage

```
specificity_plot(
  mk,
  subclass = NULL,
  group = NULL,
  type = 1,
  use_filter = FALSE,
  nrank = 8,
  nsubclass = NULL,
  expfilter = NULL,
  scheme = NULL,
  add_labels = NULL,
  label_pos = "right",
  axis_extend = 0.4,
  nudge_x = NULL,
  nudge_y = NULL,
  ...
)

specificity_plotly(
  mk,
  subclass = NULL,
  group = NULL,
  type = 1,
  use_filter = FALSE,
  nrank = 8,
  nsubclass = NULL,
  expfilter = NULL,
  scheme = NULL,
  ...
)
```

Arguments

mk	a 'cellMarkers' class object.
subclass	character value specifying the subclass to be plotted.
group	character value specifying cell group to be plotted. One of subclass or group must be specified.
type	Numeric value, either 1 (the default) for a plot of angle on x axis and mean expression on y axis; or 2 for a plot projecting the vector angle into the same plain. See Details below.
use_filter	logical, whether to use gene mean expression to which noise reduction filtering has been applied.
nrank	number of ranks of subclasses to display.
nsubclass	numeric value, number of top markers to label. By default this is obtained from mk for that subclass.

expfilter	numeric value for the expression filter level below which genes are excluded from being markers. Defaults to the level used when cellMarkers() or updateMarkers() was called.
scheme	Vector of colours for points.
add_labels	character vector of additional genes to label
label_pos	character value, either "left" or "right" specifying which side to add labels. Only for type = 1 plots.
axis_extend	numeric value, specifying how far to extend the x axis to the left as a proportion. Only invoked when label_pos = "left".
nudge_x, nudge_y	Label adjustments passed to geom_label_repel() or geom_text_repel().
...	Optional arguments passed to geom_label_repel() or geom_text_repel() for specificity_plot() or plot_ly() for specificity_plotly().

Details

For type = 1, coordinates are drawn as x = angle of vector in degrees, y = mean gene expression of each gene in the subclass of interest. This version is easier to use to identify additional gene markers. The plotly version allows users to hover over points and identify which gene they belong to.

If type = 2, the coordinates are drawn as x = vector length * sin(angle) and y = vector length * cos(angle), where vector length is the Euclidean length of that gene in space where each cell subclass is a dimension. Angle is the angle between the projected vector in space against perfection for that cell subclass, i.e. the vector lying perfectly along the subclass dimension with no deviation along other subclass dimensions, i.e. a gene which is expressed solely in that subclass and has 0 expression in all other subclasses. y is equal to the mean expression of each gene in the subclass of interest. x represents the Euclidean distance of mean expression in all other subclasses, i.e. overall non-specific gene expression in other subclasses. Thus, the plot represents a rotation of all genes as vectors around the axis of the subclass of interest onto the same plane so that the angle with the subclass of interest is visualised between genes.

Colour is used to overlay the ranking of each gene across the subclasses, showing for each gene where the subclass of interest is ranked compared to the other subclasses. Best markers have the subclass of interest ranked 1st.

Value

ggplot2 or plotly scatter plot object.

---

spillover_heatmap *Spillover heatmap*

Description

Produces a heatmap from a 'cellMarkers' or 'deconv' class object showing estimated amount of spillover between cell subclasses. The amount that each cell subclass's overall vector spillovers (projects) into other cell subclasses' vectors is shown in each row. Thus the column gives an estimate of how much the most influential (specific) genes for a cell subclass are expressed in other cells.

Usage

```
spillover_heatmap(
  x,
  text = NULL,
  cutoff = 0.5,
  fontsize = 8,
  subset = NULL,
  ...
)
```

Arguments

x	Either a 'cellMarkers' or 'deconv' class object or a spillover matrix.
text	Logical whether to show values of cells where spillover > cutoff. By default only shown for smaller matrices.
cutoff	Threshold for showing values.
fontsize	Numeric value for font size for cell values when text = TRUE.
subset	Character vector of groups to be subsetted.
...	Optional arguments passed to ComplexHeatmap::Heatmap().

Value

No return value. Draws a heatmap using ComplexHeatmap.

---

stack_plot *Stacked bar plot*

Description

Produces stacked bar plots using base graphics or ggplot2 showing amounts of cell subclasses in deconvoluted bulk samples.

Usage

```
stack_plot(
  x,
  percent = FALSE,
  order_col = 1,
  scheme = NULL,
  order_cells = c("none", "increase", "decrease"),
  seriate = NULL,
  cex.names = 0.7,
  show_xticks = TRUE,
  ...
)

stack_ggplot(
  x,
  percent = FALSE,
  order_col = 1,
  scheme = NULL,
  order_cells = c("none", "increase", "decrease"),
  seriate = NULL,
  legend_ncol = NULL,
  legend_position = "bottom",
  show_xticks = FALSE
)
```

Arguments

x	matrix of deconvolution results with samples in rows and cell subclasses or groups in columns. If a 'deconv' class object is supplied the deconvolution values for the cell subclasses are extracted and plotted.
percent	Logical whether to scale the matrix rows as percentage.
order_col	Numeric value for which column (cell subclass) to use to sort the bars - this only applies if `percent = TRUE`. If a vector of column indices is supplied, these columns are averaged first using `rowMeans()`. If `percent = FALSE`, then the default is to sort bars from low to high based on the row sums (i.e. total subclass cell amounts in each sample). Setting `order_col = 0` disables sorting of bars; in this case bars are shown in the original order of the rows of `x`.
scheme	Vector of colours. If not supplied, the default scheme uses `scales::hue_pal()`.
order_cells	Character value specifying with cell types are ordered by abundance.
seriate	Character value which enables ordering of samples using the `seriation` package. Any matrix based seriation methods can be used to order the samples. Recommended options include "CA", "BEA" or "BEA_TSP".
cex.names	Character expansion controlling bar names font size.
show_xticks	Logical whether to show rownames as x axis labels.
...	Optional arguments passed to `graphics::barplot()`.

legend_ncol — Number of columns for ggplot2 legend. If set to NULL ggplot2 sets the column number automatically.

legend_position — Position of ggplot2 legend

Value

The base graphics function has no return value. It plots a stacked barchart using base graphics. The ggplot2 version returns a ggplot2 object.

---

summary.tune_deconv *Summarising deconvolution tuning*

Description

summary method for class 'tune_deconv'.

Usage

```
## S3 method for class 'tune_deconv'
summary(
  object,
  metric = attr(object, "metric"),
  method = attr(object, "method"),
  ...
)
```

Arguments

object — dataframe of class 'tune_deconv'.

metric — Specifies tuning metric to choose optimal tune: either "RMSE", "Rsq" or "pearson".

method — Either "top" or "overall". Determines how best parameter values are chosen. With "top" the single top configuration is chosen. With "overall", the average effect of varying each parameter is calculated using the mean R-squared across all variations of other parameters. This can give a more stable choice of final tuning.

... — further arguments passed to other methods.

Value

If method = "top" prints the row representing the best tuning of parameters (maximum mean R squared, averaged across subclasses). For method = "overall", the average effect of varying each parameter is calculated by mean R-squared across the rest of the grid and the best value for each parameter is printed. Invisibly returns a dataframe of mean metric values (Pearson r^2, R^2, RMSE) averaged over subclasses.

tune_deconv *Tune deconvolution parameters*

Description

Performs an exhaustive grid search over a tuning grid of cell marker and deconvolution parameters for either updateMarkers() (e.g. expfilter or nsubclass) or deconvolute() (e.g. comp_amount).

Usage

```
tune_deconv(
  mk,
  test,
  samples,
  grid,
  output = "output",
  metric = "RMSE",
  method = "top",
  verbose = TRUE,
  cores = 1,
  ...
)
```

Arguments

mk	cellMarkers class object
test	matrix of bulk RNA-Seq to be deconvoluted. Passed to deconvolute().
samples	matrix of cell amounts with subclasses in columns and samples in rows. Note that if this has been generated by simulate_bulk(), using a value of times other than 1, then it is important that this is adjusted for here.
grid	Named list of vectors for the tuning grid similar to expand.grid(). Names represent the parameter to be tuned which must be an argument in either updateMarkers() or deconvolute(). The elements of each vector are the values to be tuned for each parameter.
output	Character value, either "output" or "percent" specifying which output from the subclass results element resulting from a call to deconvolute(). This deconvolution result is compared against the actual sample cell numbers in samples, using metric_set().
metric	Specifies tuning metric to choose optimal tune: either "RMSE", "Rsq" or "pearson".
method	Either "top" or "overall". Determines how best parameter values are chosen. With "top" the single top configuration is chosen. With "overall", the average effect of varying each parameter is calculated using the mean R-squared across all variations of other parameters. This can give a more stable choice of final tuning.

verbose	Logical whether to show progress.
cores	Number of cores for parallelisation via parallel::mclapply(). Parallelisation is not available on windows.
...	Optional arguments passed to deconvolute() to control fixed settings.

Details

Tuning plots on the resulting object can be visualised using plot_tune(). If best_tune is set to "overall", this corresponds to setting subclass = NULL in plot_tune().

Once the results output has been generated, arguments such as metric or method can be changed to see different best tunes using summary() (see summary.tune_deconv()).

test and samples matrices can be generated by simulate_bulk() and generate_samples() based on the original scRNA-Seq count dataset.

Value

Dataframe with class 'tune_deconv' whose columns include: the parameters being tuned via grid, cell subclass and R squared.

See Also

plot_tune() summary.tune_deconv()

---

updateMarkers *Update cellMarkers object*

Description

Updates a 'cellMarkers' gene signature object with new settings without having to rerun calculation of gene means, which can be slow.

Usage

```
updateMarkers(
  object = NULL,
  genemeans = NULL,
  groupmeans = NULL,
  add_gene = NULL,
  add_groupgene = NULL,
  remove_gene = NULL,
  remove_groupgene = NULL,
  remove_subclass = NULL,
  remove_group = NULL,
  bulkdata = NULL,
  nsubclass = object$opt$nsubclass,
  ngroup = object$opt$ngroup,
```

```
  expfilter = object$opt$expfilter,
  noisefilter = object$opt$noisefilter,
  noisefraction = object$opt$noisefraction,
  verbose = TRUE
)
```

Arguments

object	A 'cellMarkers' class object. Either object or genemeans must be specified.
genemeans	A matrix of mean gene expression with genes in rows and cell subclasses in columns.
groupmeans	Optional matrix of mean gene expression for overarching main cell groups (genes in rows, cell groups in columns).
add_gene	Character vector of gene markers to add manually to the cell subclass gene signature.
add_groupgene	Character vector of gene markers to add manually to the cell group gene signature.
remove_gene	Character vector of gene markers to manually remove from the cell subclass gene signature.
remove_groupgene	Character vector of gene markers to manually remove to the cell group gene signature.
remove_subclass	Character vector of cell subclasses to remove.
remove_group	Optional character vector of cell groups to remove.
bulkdata	Optional data matrix containing bulk RNA-Seq data with genes in rows. This matrix is only used for its rownames, to ensure that cell markers are selected from genes in the bulk dataset.
nsubclass	Number of genes to select for each single cell subclass. Either a single number or a vector with the number of genes for each subclass.
ngroup	Number of genes to select for each cell group.
expfilter	Genes whose maximum mean expression on log2 scale per cell type are below this value are removed and not considered for the signature.
noisefilter	Sets an upper bound for noisefraction cut-off below which gene expression is set to 0. Essentially gene expression above this level must be retained in the signature. Setting this higher can allow more suppression via noisefraction and can favour more highly expressed genes.
noisefraction	Numeric value. Maximum mean log2 gene expression across cell types is calculated and values in celltypes below this fraction are set to 0. Set in conjunction with noisefilter. Note: if this is set too high (too close to 1), it can have a deleterious effect on deconvolution.
verbose	Logical whether to show messages.

Value

A list object of S3 class 'cellMarkers'. See `cellMarkers()` for details. If `gene2symbol()` has been called, an extra list element `symbol` will be present. The list element `update` stores the call to `updateMarkers()`.

Author(s)

Myles Lewis

See Also

`cellMarkers()` `gene2symbol()`

---

violin_plot *Cell subclass violin plot*

Description

Produces violin plots using ggplot2 showing amounts of cell subclasses in deconvoluted bulk samples.

Usage

```
violin_plot(x, percent = FALSE, order_cols = c("none", "increase", "decrease"))
```

Arguments

`x`	matrix of deconvolution results with samples in rows and cell subclasses or groups in columns. If a 'deconv' class object is supplied the deconvolution values for the cell subclasses are extracted and plotted.
`percent`	Logical whether to scale the matrix rows as percentage.
`order_cols`	Character value specifying with cell types are ordered by mean abundance.

Value

A ggplot2 plotting object.

Index

add_noise, 2
add_noise(), *33*
adjust_library_size, 4
aggregate(), *27*
approxfun(), *24*

by(), *27*

cellMarkers, 4
cellMarkers(), *10*, *12–14*, *18*, *31*, *32*, *43*
collapse_group, 7
comp_heatmap, 8
ComplexHeatmap::Heatmap(), *8*, *32*, *37*
cooks.distance.deconv
        *(*rstudent.deconv*)*, 26
cooks.distance.deconv(), *12*
cos_similarity, 8
cos_similarity(), *12*, *24*

deconvolute, 9
deconvolute(), *7*, *19*, *31*, *33*, *40*, *41*
diagnose, 12
dist(), *8*

expand.grid(), *40*

fix_bulk, 13
fix_group, 13

gene2symbol, 14
gene2symbol(), *7*, *43*
gene_angle, 16
generate_samples, 15
generate_samples(), *32*, *33*, *41*
ggplot_residuals *(*plot_residuals*)*, 20
graded_log_noise *(*add_noise*)*, 2
graded_log_noise(), *3*
graphics::barplot(), *38*

loess(), *23*
log2s *(*logmean*)*, 16
log_noise *(*add_noise*)*, 2
logmean, 16, *30*
lowess(), *23*

mergeMarkers, 17
mergeMarkers(), *14*
metric_set, 18
metric_set(), *40*

ns(), *23*

parallel::mclapply(), *41*
plot(), *19*, *20*
plot.qqmap, 19
plot_comp, 19
plot_comp(), *10*
plot_residuals, 20
plot_set, 21
plot_tune, 22
plot_tune(), *41*

quantile_map, 23
quantile_map(), *17–19*

rank_angle, 24
reduceNoise, 25
residuals.deconv, 25
rstandard.deconv *(*rstudent.deconv*)*, 26
rstudent.deconv, 26
rstudent.deconv(), *12*

scapply, 27
scapply(), *30*, *34*
scmean, 29
scmean(), *5*, *6*, *16*, *17*, *28*
shift_noise *(*add_noise*)*, 2
signature_heatmap, 31
simulate_bulk, 32
simulate_bulk(), *3*, *15*, *40*, *41*
slapply, 33
slapply(), *28*

specificity_plot, 34
specificity_plotly *(*specificity_plot*)*, 34
spillover_heatmap, 36
sqrt_noise *(*add_noise*)*, 2
stack_ggplot *(*stack_plot*)*, 37
stack_plot, 37
stats::influence.measures(), *27*
summary.tune_deconv, 39
summary.tune_deconv(), *41*

tapply(), *27*
trimmean, *30*
trimmean *(*logmean*)*, 16
tune_deconv, 40
tune_deconv(), *22*

updateMarkers, 41
updateMarkers(), *7*, *12–14*, *17, 18*, *40*

violin_plot, 43