Package ‘sctransform’

October 14, 2022

Type Package
Title Variance Stabilizing Transformations for Single Cell UMI Data
Version 0.3.5
Date 2022-09-21

URL https://github.com/satijalab/sctransform

BugReports https://github.com/satijalab/sctransform/issues
License GPL-3 | file LICENSE
Encoding UTF-8
LazyData true
Depends R (>= 3.5.0)
LinkingTo RcppArmadillo, Rcpp (>= 0.11.0)
SystemRequirements C++11
Imports dplyr, magrittr, MASS, Matrix (>= 1.5.0), methods, future.apply, future, ggplot2, reshape2, rlang, gridExtra, matrixStats
Suggests irlba, testthat, knitr
Enhances glmGamPoi
RoxygenNote 7.2.1
NeedsCompilation yes

Author Christoph Hafemeister [aut] (<https://orcid.org/0000-0001-6365-8254>), Saket Choudhary [aut, cre] (<https://orcid.org/0000-0001-5202-7633>), Rahul Satija [ctb] (<https://orcid.org/0000-0001-9448-8833>
compare_expression

**Compare gene expression between two groups**

**Description**

Compare gene expression between two groups
**compare_expression**

*Usage*

```r
compare_expression(
  x,
  umi,
  group,
  val1,
  val2,
  method = "LRT",
  bin_size = 256,
  cell_attr = x$cell_attr,
  y = x$y,
  min_cells = 5,
  weighted = TRUE,
  randomize = FALSE,
  verbosity = 2,
  verbose = NULL,
  show_progress = NULL
)
```

*Arguments*

- **x**
  A list that provides model parameters and optionally meta data; use output of `vst` function
- **umi**
  A matrix of UMI counts with genes as rows and cells as columns
- **group**
  A vector indicating the groups
- **val1**
  A vector indicating the values of the group vector to treat as group 1
- **val2**
  A vector indicating the values of the group vector to treat as group 2
- **method**
  Either `"LRT"` for likelihood ratio test, or `"t_test"` for t-test
- **bin_size**
  Number of genes that are processed between updates of progress bar
- **cell_attr**
  Data frame of cell meta data
- **y**
  Only used if `method = "t_test"`, this is the residual matrix; default is `x$y`
- **min_cells**
  A gene has to be detected in at least this many cells in at least one of the groups being compared to be tested
- **weighted**
  Balance the groups by using the appropriate weights
- **randomize**
  Boolean indicating whether to shuffle group labels - only set to TRUE when testing methods
- **verbosity**
  An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
- **verbose**
  Deprecated; use verbosity instead
- **show_progress**
  Deprecated; use verbosity instead

*Value*

Data frame of results
Correct data by setting all latent factors to their median values and reversing the regression model

**Usage**

```r
correct(  
  x,  
  data = "y",  
  cell_attr = x$cell_attr,  
  as_is = FALSE,  
  do_round = TRUE,  
  do_pos = TRUE,  
  scale_factor = NA,  
  verbosity = 2,  
  verbose = NULL,  
  show_progress = NULL  
)
```

**Arguments**

- **x**: A list that provides model parameters and optionally meta data; use output of `vst` function
- **data**: The name of the entry in x that holds the data
- **cell_attr**: Provide cell meta data holding latent data info
- **as_is**: Use cell attributes as is and do not use the median; set to TRUE if you want to manually control the values of the latent factors; default is FALSE
- **do_round**: Round the result to integers
- **do_pos**: Set negative values in the result to zero
- **scale_factor**: Replace all values of UMI in the regression model by this value. Default is NA which uses median of total UMI as the latent factor.
- **verbosity**: An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
- **verbose**: Deprecated; use verbosity instead
- **show_progress**: Deprecated; use verbosity instead

**Value**

Corrected data as UMI counts
Examples

vst_out <- vst(pbmc, return_cell_attr = TRUE)
umi_corrected <- correct(vst_out)

**correct_counts**  
*Correct data by setting all latent factors to their median values and reversing the regression model*

**Description**

This version does not need a matrix of Pearson residuals. It takes the count matrix as input and calculates the residuals on the fly. The corrected UMI counts will be rounded to the nearest integer and negative values clipped to 0.

**Usage**

```r
correct_counts(
  x,
  umi,
  cell_attr = x$cell_attr,
  scale_factor = NA,
  verbosity = 2,
  verbose = NULL,
  show_progress = NULL
)
```

**Arguments**

- `x`: A list that provides model parameters and optionally meta data; use output of `vst` function
- `umi`: The count matrix
- `cell_attr`: Provide cell meta data holding latent data info
- `scale_factor`: Replace all values of UMI in the regression model by this value. Default is NA which uses median of total UMI as the latent factor.
- `verbosity`: An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
- `verbose`: Deprecated; use verbosity instead
- `show_progress`: Deprecated; use verbosity instead

**Value**

Corrected data as UMI counts
diff_mean_test

Non-parametric differential expression test for sparse non-negative data

diff_mean_test(  
  y,  
  group_labels,  
  compare = "each_vs_rest",  
  R = 99,  
  log2FC_th = log2(1.2),  
  mean_th = 0.05,  
  cells_th = 5,  
  only_pos = FALSE,  
  only_top_n = NULL,  
  mean_type = "geometric",  
  verbosity = 1  
)

Arguments

A matrix of counts; must be (or inherit from) class dgCMatrix; genes are row, cells are columns

The group labels (e.g. cluster identities); will be converted to factor

Specifies which groups to compare, see details; default is 'each_vs_rest'

The number of random permutations used to derive the p-values; default is 99

Threshold to remove genes from testing; absolute log2FC must be at least this large for a gene to be tested; default is \( \log_2(1.2) \)

Threshold to remove genes from testing; gene mean must be at least this large for a gene to be tested; default is 0.05

Threshold to remove genes from testing; gene must be detected (non-zero count) in at least this many cells in the group with higher mean; default is 5

Examples

```r
vst_out <- vst(pbmc, return_cell_attr = TRUE)
umi_corrected <- correct_counts(vst_out, pbmc)
```
### `diff_mean_test`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>only_pos</code></td>
<td>Test only genes with positive fold change (mean in group 1 &gt; mean in group 2); default is FALSE</td>
</tr>
<tr>
<td><code>only_top_n</code></td>
<td>Test only the this number of genes from both ends of the log2FC spectrum after all of the above filters have been applied; useful to get only the top markers; only used if set to a numeric value; default is NULL</td>
</tr>
<tr>
<td><code>mean_type</code></td>
<td>Which type of mean to use; if 'geometric' (default) the geometric mean is used; to avoid log(0) we use log1p to add 1 to all counts and log-transform, calculate the arithmetic mean, and then back-transform and subtract 1 using exp1m; if this parameter is set to 'arithmetic' the data is used as is</td>
</tr>
<tr>
<td><code>verbosity</code></td>
<td>Integer controlling how many messages the function prints; 0 is silent, 1 (default) is not</td>
</tr>
</tbody>
</table>

#### Value

Data frame of results

#### Details

This model-free test is applied to each gene (row) individually but is optimized to make use of the efficient sparse data representation of the input. A permutation null distribution us used to assess the significance of the observed difference in mean between two groups.

The observed difference in mean is compared against a distribution obtained by random shuffling of the group labels. For each gene every random permutation yields a difference in mean and from the population of these background differences we estimate a mean and standard deviation for the null distribution. This mean and standard deviation are used to turn the observed difference in mean into a z-score and then into a p-value. Finally, all p-values (for the tested genes) are adjusted using the Benjamini & Hochberg method (fdr). The log2FC values in the output are log2(mean1 / mean2). Empirical p-values are also calculated: \( \text{emp\_pval} = \frac{b + 1}{R + 1} \) where \( b \) is the number of times the absolute difference in mean from a random permutation is at least as large as the absolute value of the observed difference in mean, \( R \) is the number of random permutations. This is an upper bound of the real empirical p-value that would be obtained by enumerating all possible group label permutations.

There are multiple ways the group comparisons can be specified based on the compare parameter. The default, 'each_vs_rest', does multiple comparisons, one per group vs all remaining cells. 'all_vs_all' also does multiple comparisons, covering all groups pairs. If compare is set to a length two character vector, e.g. c('T-cells', 'B-cells'), one comparison between those two groups is done. To put multiple groups on either side of a single comparison, use a list of length two. E.g. \( \text{compare} = \text{list}(\text{c('cluster1', 'cluster5')}, \text{c('cluster3')}) \).

#### Examples

```r
clustering <- 1:ncol(pbmc) %% 2
tst_out <- vst(pbmc, return_corrected_umi = TRUE)
de_res <- diff_mean_test(y = vst_out$umi_corrected, group_labels = clustering)
```
diff_mean_test_conserved

Find differentially expressed genes that are conserved across samples

Description

Find differentially expressed genes that are conserved across samples

Usage

```r
diff_mean_test_conserved(
  y,
  group_labels,
  sample_labels,
  balanced = TRUE,
  compare = "each_vs_rest",
  pval_th = 1e-04,
  ...
)
```

Arguments

- `y`: A matrix of counts; must be (or inherit from) class dgCMatrix; genes are rows, cells are columns
- `group_labels`: The group labels (i.e. clusters or time points); will be converted to factor
- `sample_labels`: The sample labels; will be converted to factor
- `balanced`: Boolean, see details for explanation; default is TRUE
- `compare`: Specifies which groups to compare, see details; currently only 'each_vs_rest' (the default) is supported
- `pval_th`: P-value threshold used to call a gene differentially expressed when summarizing the tests per gene
- `...`: Parameters passed to `diff_mean_test`

Value

Data frame of results

Details

This function calls `diff_mean_test` repeatedly and aggregates the results per group and gene.

If balanced is TRUE (the default), it is assumed that each sample spans multiple groups, as would be the case when merging or integrating samples from the same tissue followed by clustering. Here the group labels would be the clusters and cluster markers would have support in each sample.

If balanced is FALSE, an unbalanced design is assumed where each sample contributes to one group. An example is a time series experiment where some samples are taken from time point 1
while other samples are taken from time point 2. The time point would be the group label and the
goal would be to identify differentially expressed genes between time points that are supported by
many between-sample comparisons.

Output columns:

group1   Group label of the first group of cells

group2   Group label of the second group of cells; currently fixed to 'rest'
gene     Gene name (from rownames of input matrix)
n_tests  The number of tests this gene participated in for this group
log2FC_min,median,max  Summary statistics for log2FC across the tests
mean1,2_median  Median of group mean across the tests
pval_max  Maximum of p-values across tests
de_tests  Number of tests that showed this gene having a log2FC going in the same direction as
          log2FC_median and having a p-value <= pval_th

The output is ordered by group1, -de_tests, -abs(log2FC_median), pval_max

Examples

clustering <- 1:ncol(pbmc) %% 2
sample_id <- 1:ncol(pbmc) %% 3
vst_out <- vst(pbmc, return_corrected_umi = TRUE)
def_res <- diff_mean_test_conserved(y = vst_out$umi_corrected,
group_labels = clustering, sample_labels = sample_id)
get_model_var

Arguments

vst_out A list that provides model parameters and optionally meta data; use output of vst function
genes The gene names for which to generate data; default is rownames(vst_out$model_pars_fit)
cell_attr Provide cell meta data holding latent data info; default is vst_out$cell_attr
n_cells Number of cells to generate; default is nrow(cell_attr)

Value

Generated data as dgCMatrix

Examples

vst_out <- vst(pbmc, return_cell_attr = TRUE)
generated_data <- generate(vst_out)

get_model_var

Return average variance under negative binomial model

Description

This is based on the formula var = mu + mu^2 / theta

Usage

get_model_var(
  vst_out,  
cell_attr = vst_out$cell_attr,  
use_nonreg = FALSE,  
bin_size = 256,  
verbosity = 2,  
verbose = NULL,  
show_progress = NULL
)

Arguments

vst_out The output of a vst run
cell_attr Data frame of cell meta data
use_nonreg Use the non-regularized parameter estimates; boolean; default is FALSE
bin_size Number of genes to put in each bin (to show progress)
verbosity An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
get_nz_median

verbose
Depreciated; use verbosity instead
show_progress
Depreciated; use verbosity instead

Value
A named vector of variances (the average across all cells), one entry per gene.

Examples

vst_out <- vst(pbmc, return_cell_attr = TRUE)
res_var <- get_model_var(vst_out)

get_nz_median(umi, genes = NULL)

Description
Get median of non zero UMIs from a count matrix using a subset of genes (slow)

Usage
get_nz_median(umi, genes = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>umi</td>
<td>Count matrix</td>
</tr>
<tr>
<td>genes</td>
<td>List of genes to calculate statistics. Default is NULL which returns the non-zero median using all genes</td>
</tr>
</tbody>
</table>

Value
A numeric value representing the median of non-zero entries from the UMI matrix
get_nz_median2

Get median of non zero UMIs from a count matrix

Description
Get median of non zero UMIs from a count matrix

Usage
get_nz_median2(umi)

Arguments
umi Count matrix

Value
A numeric value representing the median of non-zero entries from the UMI matrix

get_residuals

Return Pearson or deviance residuals of regularized models

Description
Return Pearson or deviance residuals of regularized models

Usage
get_residuals(
  vst_out,
  umi,
  residual_type = "pearson",
  res_clip_range = c(-sqrt(ncol(umi)), sqrt(ncol(umi))),
  min_variance = vst_out$arguments$min_variance,
  cell_attr = vst_out$cell_attr,
  bin_size = 256,
  verbosity = vst_out$arguments$verbosity,
  verbose = NULL,
  show_progress = NULL
)
get_residual_var

Arguments

vst_out The output of a vst run
umi The UMI count matrix that will be used
residual_type What type of residuals to return; can be 'pearson' or 'deviance'; default is 'pearson'
res_clip_range Numeric of length two specifying the min and max values the results will be clipped to; default is c(-sqrt(ncol(umi)), sqrt(ncol(umi)))
min_variance Lower bound for the estimated variance for any gene in any cell when calculating pearson residual; default is vst_out$arguments$min_variance
cell_attr Data frame of cell meta data
bin_size Number of genes to put in each bin (to show progress)
verbosity An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
verbose Deprecated; use verbosity instead
show_progress Deprecated; use verbosity instead

Value

A matrix of residuals

Examples

vst_out <- vst(pbmcm, return_cell_attr = TRUE)
pearson_res <- get_residuals(vst_out, pbmcm)
deviance_res <- get_residuals(vst_out, pbmcm, residual_type = 'deviance')

get_residual_var

Return variance of residuals of regularized models

Description

This never creates the full residual matrix and can be used to determine highly variable genes.

Usage

get_residual_var(
  vst_out,
  umi,
  residual_type = "pearson",
  res_clip_range = c(-sqrt(ncol(umi)), sqrt(ncol(umi))),
  min_variance = vst_out$arguments$min_variance,
  cell_attr = vst_out$cell_attr,
is_outlier

bin_size = 256,
verbosity = vst_out$arguments$verbosity,
verbose = NULL,
show_progress = NULL
)

Arguments

vst_out The output of a vst run
umi The UMI count matrix that will be used
residual_type What type of residuals to return; can be 'pearson' or 'deviance'; default is 'pearson'
res_clip_range Numeric of length two specifying the min and max values the residuals will be clipped to; default is c(-sqrt(ncol(umi)), sqrt(ncol(umi)))
min_variance Lower bound for the estimated variance for any gene in any cell when calculating pearson residual; default is vst_out$arguments$min_variance
cell_attr Data frame of cell meta data
bin_size Number of genes to put in each bin (to show progress)
verbosity An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
verbose Deprecated; use verbosity instead
show_progress Deprecated; use verbosity instead

Value

A vector of residual variances (after clipping)

Examples

vst_out <- vst(pbmc, return_cell_attr = TRUE)
res_var <- get_residual_var(vst_out, pbmc)

is_outlier Identify outliers

Description

Identify outliers

Usage

is_outlier(y, x, th = 10)
**make.sparse**

**Arguments**

- `y` Dependent variable
- `x` Independent variable
- `th` Outlier score threshold

**Value**

Boolean vector

---

**Description**

Convert a given matrix to dgCMatrix

**Usage**

```r
make.sparse(mat)
```

**Arguments**

- `mat` Input matrix

**Value**

A dgCMatrix

---

**pbmc**

**Peripheral Blood Mononuclear Cells (PBMCs)**

**Description**

UMI counts for a subset of cells freely available from 10X Genomics

**Usage**

```r
pbmc
```

**Format**

A sparse matrix (dgCMatrix, see Matrix package) of molecule counts. There are 914 rows (genes) and 283 columns (cells). This is a downsamples version of a 3K PBMC dataset available from 10x Genomics.

**Source**

https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k
plot_model

Plot observed UMI counts and model

Description

Plot observed UMI counts and model

Usage

```r
plot_model(
  x,
  umi,
  goi,
  x_var = x$arguments$latent_var[1],
  cell_attr = x$cell_attr,
  do_log = TRUE,
  show_fit = TRUE,
  show_nr = FALSE,
  plot_residual = FALSE,
  batches = NULL,
  as_poisson = FALSE,
  arrange_vertical = TRUE,
  show_density = FALSE,
  gg_cmds = NULL
)
```

Arguments

- **x**: The output of a vst run
- **umi**: UMI count matrix
- **goi**: Vector of genes to plot
- **x_var**: Cell attribute to use on x axis; will be taken from x$arguments$latent_var[1] by default
- **cell_attr**: Cell attributes data frame; will be taken from x$cell_attr by default
- **do_log**: Log10 transform the UMI counts in plot
- **show_fit**: Show the model fit
- **show_nr**: Show the non-regularized model (if available)
- **plot_residual**: Add panels for the Pearson residuals
- **batches**: Manually specify a batch variable to break up the model plot in segments
- **as_poisson**: Fix model parameter theta to Inf, effectively showing a Poisson model
- **arrange_vertical**: Stack individual ggplot objects or place side by side
- **show_density**: Draw 2D density lines over points
- **gg_cmds**: Additional ggplot layer commands
plot_model_pars

Value

A ggplot object

Examples

vst_out <- vst(pbmc, return_cell_attr = TRUE)
plot_model(vst_out, pbmc, 'EMC4')

plot_model_pars

Plot estimated and fitted model parameters

Description

Plot estimated and fitted model parameters

Usage

plot_model_pars(
  vst_out,
  xaxis = "gmean",
  show_theta = FALSE,
  show_var = FALSE,
  verbosity = 2,
  verbose = NULL,
  show_progress = NULL
)

Arguments

vst_out The output of a vst run
xaxis Variable to plot on X axis; default is "gmean"
show_theta Whether to show the theta parameter; default is FALSE (only the overdispersion factor is shown)
show_var Whether to show the average model variance; default is FALSE
 verbosity An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2
 verbose Deprecated; use verbosity instead
show_progress Deprecated; use verbosity instead

Value

A ggplot object
Examples

```r
vst_out <- vst(pbmc, return_gene_attr = TRUE)
plot_model_pars(vst_out)
```

---

**robust_scale**

*Robust scale using median and mad*

**Description**

Robust scale using median and mad

**Usage**

```r
robust_scale(x)
```

**Arguments**

- `x` Numeric

**Value**

Numeric

---

**robust_scale_binned**

*Robust scale using median and mad per bin*

**Description**

Robust scale using median and mad per bin

**Usage**

```r
robust_scale_binned(y, x, breaks)
```

**Arguments**

- `y` Numeric vector
- `x` Numeric vector
- `breaks` Numeric vector of breaks

**Value**

Numeric vector of scaled score
**row_gmean**  
*Geometric mean per row*

**Description**  
Geometric mean per row

**Usage**  
`row_gmean(x, eps = 1)`

**Arguments**  

- `x`: matrix of class `matrix` or `dgCMatrix`
- `eps`: small value to add to `x` to avoid `log(0)`; default is 1

**Value**  
geometric means

---

**row_var**  
*Variance per row*

**Description**  
Variance per row

**Usage**  
`row_var(x)`

**Arguments**  

- `x`: matrix of class `matrix` or `dgCMatrix`

**Value**  
variances
smooth_via_pca  Smooth data by PCA

Description

Perform PCA, identify significant dimensions, and reverse the rotation using only significant dimensions.

Usage

smooth_via_pca(
  x,
  elbow_th = 0.025,
  dims_use = NULL,
  max_pc = 100,
  do_plot = FALSE,
  scale. = FALSE
)

Arguments

  x               A data matrix with genes as rows and cells as columns
  elbow_th        The fraction of PC sdev drop that is considered significant; low values will lead
t                  to more PCs being used
  dims_use        Directly specify PCs to use, e.g. 1:10
  max_pc          Maximum number of PCs computed
  do_plot         Plot PC sdev and sdev drop
  scale.          Boolean indicating whether genes should be divided by standard deviation after
t                  centering and prior to PCA

Value

Smoothed data

Examples

vst_out <- vst(pbm)
y_smooth <- smooth_via_pca(vst_out$y, do_plot = TRUE)
umify

Quantile normalization of cell-level data to match typical UMI count data

Description

Quantile normalization of cell-level data to match typical UMI count data

Usage

umify(counts)

Arguments

counts A matrix of class dgCMatrix with genes as rows and columns as cells

Value

A UMI-fied count matrix

Details

sctransform::vst operates under the assumption that gene counts approximately follow a Negative Binomial distribution. For UMI-based data that seems to be the case, however, non-UMI data does not behave in the same way. In some cases it might be better to apply a transformation to such data to make it look like UMI data. This function applies such a transformation function.

Cells in the input matrix are processed independently. For each cell the non-zero data is transformed to quantile values. Based on the number of genes detected a smooth function is used to predict the UMI-like counts.

The functions have be trained on various public data sets and come as part of the package (see umify_data data set in this package).

Examples

silly_example <- umify(pbmc)
umify_data

Transformation functions for umify

Description

The functions have been trained on various public data sets and relate quantile values to log-counts. Here the expected values at various points are given.

Usage

```r
umify_data
```

Format

A list of length two. The first element is a data frame with group, quantile and log-counts values. The second element is a vector of breaks to be used with cut to group observations.

vst

Variance stabilizing transformation for UMI count data

Description

Apply variance stabilizing transformation to UMI count data using a regularized Negative Binomial regression model. This will remove unwanted effects from UMI data and return Pearson residuals. Uses future_lapply; you can set the number of cores it will use to n with plan(strategy = "multicore", workers = n). If n_genes is set, only a (somewhat-random) subset of genes is used for estimating the initial model parameters. For details see doi:10.1186/s1305901918741.

Usage

```r
vst(
  umi,
  cell_attr = NULL,
  latent_var = c("log_umi"),
  batch_var = NULL,
  latent_var_nonreg = NULL,
  n_genes = 2000,
  n_cells = NULL,
  method = "poisson",
  do_regularize = TRUE,
  theta_regularization = "od_factor",
  res_clip_range = c(-sqrt(ncol(umi)), sqrt(ncol(umi))),
  bin_size = 500,
  min_cells = 5,
  residual_type = "pearson",
)```
return_cell_attr = FALSE,
return_gene_attr = TRUE,
return_corrected_umi = FALSE,
min_variance = -Inf,
bw_adjust = 3,
gmean_eps = 1,
theta_estimation_fun = "theta.ml",
theta_given = NULL,
exclude_poisson = FALSE,
use_geometric_mean = TRUE,
use_geometric_mean_offset = FALSE,
fix_intercept = FALSE,
fix_slope = FALSE,
scale_factor = NA,
vst.flavor = FALSE,
verbosity = 2,
verbose = NULL,
show_progress = NULL
)

Arguments

umi A matrix of UMI counts with genes as rows and cells as columns
cell_attr A data frame containing the dependent variables; if omitted a data frame with umi and gene will be generated
latent_var The independent variables to regress out as a character vector; must match column names in cell_attr; default is c("log_umi")
batch_var The dependent variables indicating which batch a cell belongs to; no batch interaction terms used if omitted
latent_var_nonreg The non-regularized dependent variables to regress out as a character vector; must match column names in cell_attr; default is NULL
n_genes Number of genes to use when estimating parameters (default uses 2000 genes, set to NULL to use all genes)
n_cells Number of cells to use when estimating parameters (default uses all cells)
method Method to use for initial parameter estimation; one of 'poisson', 'qpoisson', 'nb_fast', 'nb', 'nb_theta_given', 'glmGamPoi', 'offset', 'offset_shared_theta_estimate', 'glmGamPoi_offset'; default is 'poisson'
do_regularize Boolean that, if set to FALSE, will bypass parameter regularization and use all genes in first step (ignoring n_genes); default is FALSE
theta_regularization Method to use to regularize theta; use 'log_theta' for the behavior prior to version 0.3; default is 'od_factor'
res_clip_range Numeric of length two specifying the min and max values the results will be clipped to; default is c(-sqrt(ncol(umi)), sqrt(ncol(umi)))
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin_size</td>
<td>Number of genes to process simultaneously; this will determine how often the progress bars are updated and how much memory is being used; default is 500</td>
</tr>
<tr>
<td>min_cells</td>
<td>Only use genes that have been detected in at least this many cells; default is 5</td>
</tr>
<tr>
<td>residual_type</td>
<td>What type of residuals to return; can be 'pearson', 'deviance', or 'none'; default is 'pearson'</td>
</tr>
<tr>
<td>return_cell_attr</td>
<td>Make cell attributes part of the output; default is FALSE</td>
</tr>
<tr>
<td>return_gene_attr</td>
<td>Calculate gene attributes and make part of output; default is TRUE</td>
</tr>
<tr>
<td>return_corrected_umi</td>
<td>If set to TRUE output will contain corrected UMI matrix; see correct function</td>
</tr>
<tr>
<td>min_variance</td>
<td>Lower bound for the estimated variance for any gene in any cell when calculating pearson residual; one of 'umi_median', 'model_median', 'model_mean' or a numeric. default is -Inf. When set to 'umi_median' uses (median of non-zero UMIs / 5)^2 as the minimum variance so that a median UMI (often 1) results in a maximum pearson residual of 5. When set to 'model_median' or 'model_mean' uses the mean/median of the model estimated mu per gene as the minimum_variance.</td>
</tr>
<tr>
<td>bw_adjust</td>
<td>Kernel bandwidth adjustment factor used during regularization; factor will be applied to output of bw.SJ; default is 3</td>
</tr>
<tr>
<td>gmean_eps</td>
<td>Small value added when calculating geometric mean of a gene to avoid log(0); default is 1</td>
</tr>
<tr>
<td>theta_estimation_fun</td>
<td>Character string indicating which method to use to estimate theta (when method = poisson); default is 'theta.ml', but 'theta.mm' seems to be a good and fast alternative</td>
</tr>
<tr>
<td>theta_given</td>
<td>If method is set to nb_theta_given, this should be a named numeric vector of fixed theta values for the genes; if method is offset, this should be a single value; default is NULL</td>
</tr>
<tr>
<td>exclude_poisson</td>
<td>Exclude poisson genes (i.e. mu &lt; 0.001 or mu &gt; variance) from regularization; default is FALSE</td>
</tr>
<tr>
<td>use_geometric_mean</td>
<td>Use geometric mean instead of arithmetic mean for all calculations ; default is TRUE</td>
</tr>
<tr>
<td>use_geometric_mean_offset</td>
<td>Use geometric mean instead of arithmetic mean in the offset model; default is FALSE</td>
</tr>
<tr>
<td>fix_intercept</td>
<td>Fix intercept as defined in the offset model; default is FALSE</td>
</tr>
<tr>
<td>fix_slope</td>
<td>Fix slope to log(10) (equivalent to using library size as an offset); default is FALSE</td>
</tr>
<tr>
<td>scale_factor</td>
<td>Replace all values of UMI in the regression model by this value instead of the median UMI; default is NA</td>
</tr>
</tbody>
</table>
vst.flavor

When set to 'v2' sets method = glmGamPoi_offset, n_cells=2000, and exclude_poisson = TRUE which causes the model to learn theta and intercept only besides excluding poisson genes from learning and regularization; default is NULL which uses the original scTransform model

verbosity

An integer specifying whether to show only messages (1), messages and progress bars (2) or nothing (0) while the function is running; default is 2

verbose

Deprecated; use verbosity instead

show_progress

Deprecated; use verbosity instead

Value

A list with components

- **y**
  - Matrix of transformed data, i.e. Pearson residuals, or deviance residuals; empty if residual_type = 'none'

- **umi_corrected**
  - Matrix of corrected UMI counts (optional)

- **model_str**
  - Character representation of the model formula

- **model_pars**
  - Matrix of estimated model parameters per gene (theta and regression coefficients)

- **model_pars_outliers**
  - Vector indicating whether a gene was considered to be an outlier

- **model_pars_fit**
  - Matrix of fitted / regularized model parameters

- **model_str_nonreg**
  - Character representation of model for non-regularized variables

- **model_pars_nonreg**
  - Model parameters for non-regularized variables

- **genes_log_gmean_step1**
  - log-geometric mean of genes used in initial step of parameter estimation

- **cells_step1**
  - Cells used in initial step of parameter estimation

- **arguments**
  - List of function call arguments

- **cell_attr**
  - Data frame of cell meta data (optional)

- **gene_attr**
  - Data frame with gene attributes such as mean, detection rate, etc. (optional)

- **times**
  - Time stamps at various points in the function

Details

In the first step of the algorithm, per-gene glm model parameters are learned. This step can be done on a subset of genes and/or cells to speed things up. If method is set to 'poisson', a poisson regression is done and the negative binomial theta parameter is estimated using the response residuals in theta_estimation_fun. If method is set to 'qpoisson', coefficients and overdispersion (phi) are estimated by quasi poisson regression and theta is estimated based on phi and the mean fitted value - this is currently the fastest method with results very similar to 'glmGamPoi' If method is set to 'nb_fast', coefficients and theta are estimated as in the 'poisson' method, but coefficients are then re-estimated using a proper negative binomial model in a second call to glm with family = MASS::negative.binomial(theta = theta). If method is set to 'nb', coefficients and theta are
estimated by a single call to MASS::glm.nb. If method is set to ’glmGamPoi’, coefficients and theta are estimated by a single call to glmGamPoi::glm_gp.

A special case is method = 'offset'. Here no regression parameters are learned, but instead an offset model is assumed. The latent variable is set to log_umi and a fixed slope of log(10) is used (offset). The intercept is given by log(gene_mean) - log(avg_cell_umi). See Lause et al. doi:10.1186/s13059021024517 for details. Theta is set to 100 by default, but can be changed using the theta_given parameter (single numeric value). If the offset method is used, the following parameters are overwritten: cell_attr <- NULL, latent_var <- c('log_umi'), batch_var <- NULL, latent_var_nonreg <- NULL, n_genes <- NULL, n_cells <- NULL, do_regularize <- FALSE. Further, method = 'offset_shared_theta_estimate' exists where the 250 most highly expressed genes with detection rate of at least 0.5 are used to estimate a theta that is then shared across all genes. Thetas are estimated per individual gene using 5000 randomly selected cells. The final theta used for all genes is then the average.

Examples

vst_out <- vst(pbm)
Index

* datasets
  pbmc, 15
  umify_data, 22

compare_expression, 2
correct, 4
correct_counts, 5
diff_mean_test, 6
diff_mean_test_conserved, 8
generate, 9
get_model_var, 10
get_nz_median, 11
get_nz_median2, 12
get_residual_var, 13
get_residuals, 12

is_outlier, 14

make.sparse, 15

pbmc, 15
plot_model, 16
plot_model_pars, 17

robust_scale, 18
robust_scale_binned, 18
row_gmean, 19
row_var, 19

smooth_via_pca, 20

umify, 21
umify_data, 22

vst, 22