Package ‘sctransform’

December 17, 2019

Type Package

Title Variance Stabilizing Transformations for Single Cell UMI Data

Version 0.2.1

Description A normalization method for single-cell UMI count data using a variance stabilizing transformation. The transformation is based on a negative binomial regression model with regularized parameters. As part of the same regression framework, this package also provides functions for batch correction, and data correction. See Hafemeister and Satija 2019 <doi:10.1101/576827> for more details.

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

BugReports https://github.com/ChristophH/sctransform/issues

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.0.2)

Imports MASS, Matrix, methods, future.apply, ggplot2, reshape2, gridExtra

LinkingTo Rcpp (>= 0.11.0), RcppEigen

Suggests irlba, testthat, knitr

RoxygenNote 6.1.1

NeedsCompilation yes

Author Christoph Hafemeister [aut, cre] (<https://orcid.org/0000-0001-6365-8254>)

Maintainer Christoph Hafemeister <chafemeister@nygenome.org>

Repository CRAN

Date/Publication 2019-12-17 15:00:02 UTC
**compare_expression**

**R topics documented:**

- compare_expression .......................................................... 2
- correct ................................................................. 3
- correct_counts ............................................................ 4
- generate ................................................................. 5
- get_model_var ............................................................. 5
- get_residuals .............................................................. 6
- get_residual_var .......................................................... 7
- is_outlier ................................................................. 8
- pbmc ....................................................................... 8
- plot_model ............................................................... 9
- plot_model_pars .......................................................... 10
- robust_scale ............................................................. 11
- robust_scale_binned ..................................................... 11
- row_gmean ............................................................. 12
- row_var ................................................................. 12
- smooth_via_pca .......................................................... 13
- vst ............................................................... 14

**Index**

```
compare_expression    Compare gene expression between two groups
```

**Description**

Compare gene expression between two groups

**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, show_progress = TRUE)
```

**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
**correct**  

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

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

### Usage
```
correct(x, data = "y", cell_attr = x$cell_attr, do_round = TRUE,  
do_pos = TRUE, show_progress = TRUE)
```

### 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
- **do_round**: Round the result to integers
- **do_pos**: Set negative values in the result to zero
- **show_progress**: Whether to print progress bar

### Examples
```r
## Not run:
vst_out <- vst(pbmc, return_cell_attr = TRUE)  
# create fake clusters  
clustering <- 1:ncol(pbmc) %>% 100  
res <- compare_expression(x = vst_out, umi = pbmc, group = clustering, val1 = 0, val2 = 3)

## End(Not run)
```
Value
Corrected data as UMI counts

Examples

```r
corrected_counts <- 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, show_progress = TRUE)
```

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

- `show_progress`  
  Whether to print progress bar

Value
Corrected data as UMI counts

Examples

```r
corrected_counts <- correct_counts(vst_out)
```
**generate**

*Generate data from regularized models.*

**Description**

Generate data from regularized models. This generates data from the background, i.e. no residuals are added to the simulated data. The cell attributes for the generated cells are sampled from the input with replacement.

**Usage**

```r
generate(vst_out, genes = rownames(vst_out$model_pars_fit),
          cell_attr = vst_out$cell_attr, n_cells = nrow(cell_attr))
```

**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**

```r
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 \( \text{var} = \mu + \mu^2 / \theta \)

**Usage**

```r
get_model_var(vst_out, cell_attr = vst_out$cell_attr,
              use_nonreg = FALSE, bin_size = 256, show_progress = TRUE)
```
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)
show_progress  Whether to print progress bar

Value

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

Examples

```
## Not run:
vst_out <- vst(pbmc)
res_var <- get_model_var(vst_out)
## End(Not run)
```

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,
             show_progress = TRUE)
```

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
get_residual_var

    cell_attr  Data frame of cell meta data
    bin_size   Number of genes to put in each bin (to show progress)
    show_progress  Whether to print progress bar

Value

A matrix of residuals

Examples

## Not run:
vst_out <- vst(pbmc)
pearson_res <- get_residuals(vst_out, pbmc)
deviance_res <- get_residuals(vst_out, pbmc, residual_type = \"deviance\")
## End(Not run)

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, bin_size = 256,
    show_progress = TRUE)

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)
show_progress   Whether to print progress bar
Value

A vector of residual variances (after clipping)

Examples

```r
## Not run:
vst_out <- vst(pbmc)
res_var <- get_residual_var(vst_out, pbmc)

## End(Not run)
```

---

**is_outlier**

*Identify outliers*

**Description**

Identify outliers

**Usage**

```r
is_outlier(y, x, th = 10)
```

**Arguments**

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

**Value**

Boolean vector

---

**pbmc**

*Peripheral Blood Mononuclear Cells (PBMCs)*

**Description**

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

**Usage**

```r
pbmc
```
plot_model

Format

A sparse matrix (dgCMatrix, see Matrix package) of molecule counts. There are 914 rows (genes) and 283 columns (cells). This is a downsampled 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

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 = TRUE,
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

```r
## Not run:
vst_out <- vst(pbmc, return_cell_attr = TRUE)
plot_model(vst_out, pbmc, 'PPBP')

## End(Not run)
```

---

plot_model_pars  

Plot estimated and fitted model parameters

Description

Plot estimated and fitted model parameters

Usage

```r
plot_model_pars(vst_out, show_var = FALSE)
```

Arguments

- `vst_out`: The output of a vst run
- `show_var`: Whether to show the average model variance; boolean; default is FALSE

Value

A ggplot object

Examples

```r
## Not run:
vst_out <- vst(pbmc, return_gene_attr = TRUE)
plot_model_pars(vst_out)

## End(Not run)
```
robust_scale

Robust scale using median and mad

Description
Robust scale using median and mad

Usage
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
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**

```r
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**

```r
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 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 centering and prior to PCA

Value

Smoothed data

Examples

vst_out <- vst(pbc)
y_smooth <- smooth_via_pca(vst_out$y, do_plot = TRUE)
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.

**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,
    res_clip_range = c(-sqrt(ncol(umi)), sqrt(ncol(umi))),
    bin_size = 256, 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_given = NULL, show_progress = TRUE)
```

**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', 'nb_fast', 'nb', 'nb_theta_given'
- **do_regularize**
  Boolean that, if set to FALSE, will bypass parameter regularization and use all genes in first step (ignoring n_genes).
- **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)))
- **bin_size**
  Number of genes to put in each bin (to show progress)
**min_cells**  
Only use genes that have been detected in at least this many cells; default is 5

**residual_type**  
What type of residuals to return; can be 'pearson', 'deviance', or 'none'; default is 'pearson'

**return_cell_attr**  
Make cell attributes part of the output; default is FALSE

**return_gene_attr**  
Calculate gene attributes and make part of output; default is TRUE

**return_corrected_umi**  
If set to TRUE output will contain corrected UMI matrix; see `correct` function

**min_variance**  
Lower bound for the estimated variance for any gene in any cell when calculating pearson residual; default is -Inf

**bw_adjust**  
Kernel bandwidth adjustment factor used during regularization; factor will be applied to output of bw.SJ; default is 3

**gmean_eps**  
Small value added when calculating geometric mean of a gene to avoid log(0); default is 1

**theta_given**  
Named numeric vector of fixed theta values for the genes; will only be used if method is set to nb_theta_given; default is NULL

**show_progress**  
Whether to print messages and show progress bar

**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)
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', glm will be called with `family = poisson` and the negative binomial theta parameter will be estimated using the response residuals in \texttt{MASS::theta.ml}. If `method` is set to 'nb_fast', glm 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 \texttt{MASS::glm.nb}.

Examples

\begin{verbatim}
  vst_out <- vst(pbm)
\end{verbatim}
Index

*Topic datasets
  pbmc, 8

compare_expression, 2
correct, 3
correct_counts, 4

generate, 5
get_model_var, 5
get_residual_var, 7
get_residuals, 6

is_outlier, 8

pbmc, 8
plot_model, 9
plot_model_pars, 10

robust_scale, 11
robust_scale_binned, 11
row_gmean, 12
row_var, 12

smooth_via_pca, 13
vst, 14