

# 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](mailto:chafemeister@nygenome.org)>

**Repository** CRAN

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

## 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

<b>Index</b>	<b>17</b>
--------------	-----------

---

compare_expression	<i>Compare gene expression between two groups</i>
--------------------	---------------------------------------------------

---

### Description

Compare gene expression between two groups

### Usage

```
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

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
show_progress	Show progress bar

**Value**

Data frame of results

**Examples**

```
## 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)
```

---

correct	<i>Correct data by setting all latent factors to their median values and reversing the regression model</i>
---------	-------------------------------------------------------------------------------------------------------------

---

**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

**Value**

Corrected data as UMI counts

**Examples**

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

---

correct_counts	<i>Correct data by setting all latent factors to their median values and reversing the regression model</i>
----------------	-------------------------------------------------------------------------------------------------------------

---

**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**

```
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**

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

---

generate	<i>Generate data from regularized models.</i>
----------	-----------------------------------------------

---

### 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

```
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

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

---

get_model_var	<i>Return average variance under negative binomial model</i>
---------------	--------------------------------------------------------------

---

### Description

This is based on the formula  $\text{var} = \mu + \mu^2 / \theta$

### Usage

```
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	<i>Return Pearson or deviance residuals of regularized models</i>
---------------	-------------------------------------------------------------------

---

**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

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	<i>Return variance of residuals of regularized models</i>
------------------	-----------------------------------------------------------

---

**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**

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

## End(Not run)
```

---

is_outlier	<i>Identify outliers</i>
------------	--------------------------

---

**Description**

Identify outliers

**Usage**

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

**Arguments**

y	Dependent variable
x	Independent variable
th	Outlier score threshold

**Value**

Boolean vector

---

pbmc	<i>Peripheral Blood Mononuclear Cells (PBMCs)</i>
------	---------------------------------------------------

---

**Description**

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

**Usage**

```
pbmc
```



**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

**Value**

A ggplot object

**Examples**

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

---

plot_model_pars	<i>Plot estimated and fitted model parameters</i>
-----------------	---------------------------------------------------

---

**Description**

Plot estimated and fitted model parameters

**Usage**

```
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**

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

---

robust_scale	<i>Robust scale using median and mad</i>
--------------	------------------------------------------

---

**Description**

Robust scale using median and mad

**Usage**

```
robust_scale(x)
```

**Arguments**

x	Numeric
---	---------

**Value**

Numeric

---

robust_scale_binned	<i>Robust scale using median and mad per bin</i>
---------------------	--------------------------------------------------

---

**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	<i>Geometric mean per row</i>
-----------	-------------------------------

---

**Description**

Geometric mean per row

**Usage**

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

**Arguments**

x	matrix of class <code>matrix</code> or <code>dgCMatrix</code>
eps	small value to add to x to avoid <code>log(0)</code> ; default is 1

**Value**

geometric means

---

row_var	<i>Variance per row</i>
---------	-------------------------

---

**Description**

Variance per row

**Usage**

```
row_var(x)
```

**Arguments**

x	matrix of class <code>matrix</code> or <code>dgCMatrix</code>
---	---------------------------------------------------------------

**Value**

variances

---

smooth_via_pca	<i>Smooth data by PCA</i>
----------------	---------------------------

---

### 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(pbmc)  
y_smooth <- smooth_via_pca(vst_out$y, do_plot = TRUE)
```

**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**

```
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**

<code>umi</code>	A matrix of UMI counts with genes as rows and cells as columns
<code>cell_attr</code>	A data frame containing the dependent variables; if omitted a data frame with <code>umi</code> and <code>gene</code> will be generated
<code>latent_var</code>	The independent variables to regress out as a character vector; must match column names in <code>cell_attr</code> ; default is <code>c("log_umi")</code>
<code>batch_var</code>	The dependent variables indicating which batch a cell belongs to; no batch interaction terms used if omitted
<code>latent_var_nonreg</code>	The non-regularized dependent variables to regress out as a character vector; must match column names in <code>cell_attr</code> ; default is <code>NULL</code>
<code>n_genes</code>	Number of genes to use when estimating parameters (default uses 2000 genes, set to <code>NULL</code> to use all genes)
<code>n_cells</code>	Number of cells to use when estimating parameters (default uses all cells)
<code>method</code>	Method to use for initial parameter estimation; one of <code>'poisson'</code> , <code>'nb_fast'</code> , <code>'nb'</code> , <code>'nb_theta_given'</code>
<code>do_regularize</code>	Boolean that, if set to <code>FALSE</code> , will bypass parameter regularization and use all genes in first step (ignoring <code>n_genes</code> ).
<code>res_clip_range</code>	Numeric of length two specifying the min and max values the results will be clipped to; default is <code>c(-sqrt(ncol(umi)), sqrt(ncol(umi)))</code>
<code>bin_size</code>	Number of genes to put in each bin (to show progress)

<code>min_cells</code>	Only use genes that have been detected in at least this many cells; default is 5
<code>residual_type</code>	What type of residuals to return; can be 'pearson', 'deviance', or 'none'; default is 'pearson'
<code>return_cell_attr</code>	Make cell attributes part of the output; default is FALSE
<code>return_gene_attr</code>	Calculate gene attributes and make part of output; default is TRUE
<code>return_corrected_umi</code>	If set to TRUE output will contain corrected UMI matrix; see <code>correct</code> function
<code>min_variance</code>	Lower bound for the estimated variance for any gene in any cell when calculating pearson residual; default is -Inf
<code>bw_adjust</code>	Kernel bandwidth adjustment factor used during regularization; factor will be applied to output of <code>bw.SJ</code> ; default is 3
<code>gmean_eps</code>	Small value added when calculating geometric mean of a gene to avoid $\log(0)$ ; default is 1
<code>theta_given</code>	Named numeric vector of fixed theta values for the genes; will only be used if method is set to <code>nb_theta_given</code> ; default is NULL
<code>show_progress</code>	Whether to print messages and show progress bar

## Value

A list with components

<code>y</code>	Matrix of transformed data, i.e. Pearson residuals, or deviance residuals; empty if <code>residual_type = 'none'</code>
<code>umi_corrected</code>	Matrix of corrected UMI counts (optional)
<code>model_str</code>	Character representation of the model formula
<code>model_pars</code>	Matrix of estimated model parameters per gene (theta and regression coefficients)
<code>model_pars_outliers</code>	Vector indicating whether a gene was considered to be an outlier
<code>model_pars_fit</code>	Matrix of fitted / regularized model parameters
<code>model_str_nonreg</code>	Character representation of model for non-regularized variables
<code>model_pars_nonreg</code>	Model parameters for non-regularized variables
<code>genes_log_gmean_step1</code>	log-geometric mean of genes used in initial step of parameter estimation
<code>cells_step1</code>	Cells used in initial step of parameter estimation
<code>arguments</code>	List of function call arguments
<code>cell_attr</code>	Data frame of cell meta data (optional)
<code>gene_attr</code>	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 `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 `MASS::glm.nb`.

## Examples

```
vst_out <- vst(pbmc)
```



# 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