## Package 'transformGamPoi'

March 4, 2025

```
Version 1.12.0
Description Variance-stabilizing transformations help with the analysis of
      heteroskedastic data (i.e., data where the variance is not constant, like count data).
      This package provide two types of variance stabilizing transformations: (1) methods based on the
      delta method (e.g., 'acosh', 'log(x+1)'), (2) model residual based (Pearson and randomized
      quantile residuals).
BugReports https://github.com/const-ae/transformGamPoi/issues
URL https://github.com/const-ae/transformGamPoi
License GPL-3
Encoding UTF-8
Imports glmGamPoi, DelayedArray, Matrix, MatrixGenerics,
     SummarizedExperiment, HDF5Array, methods, utils, Rcpp
Suggests testthat, TENxPBMCData, scran, knitr, rmarkdown, BiocStyle
Roxygen list(markdown = TRUE)
RoxygenNote 7.1.2
Config/testthat/edition 3
biocViews SingleCell, Normalization, Preprocessing, Regression
VignetteBuilder knitr
LinkingTo Rcpp
git_url https://git.bioconductor.org/packages/transformGamPoi
git_branch RELEASE_3_20
git_last_commit 417e94e
git_last_commit_date 2024-10-29
Repository Bioconductor 3.20
Date/Publication 2025-03-03
Author Constantin Ahlmann-Eltze [aut, cre]
     (<https://orcid.org/0000-0002-3762-068X>)
Maintainer Constantin Ahlmann-Eltze <artjom31415@googlemail.com>
```

Title Variance Stabilizing Transformation for Gamma-Poisson Models

Type Package

2 acosh\_transform

## **Contents**

.handle_data_parameter	•	•	•	•		•		•	•	•			•	•	•	•		•		•	•	•	- 2
$acosh\_transform \ . \ . \ .$																							2
$estimate\_size\_factors \ .$																							5
$residual\_transform  .  .$																							5
transformGamPoi																							8

.handle\_data\_parameter

Take any kind of data and extract the matrix

11

## Description

Adapted from glmGamPoi:::handle\_data\_parameter

## Usage

Index

```
.handle_data_parameter(data, on_disk, allow_sparse = TRUE)
```

## Value

A matrix.

 ${\tt acosh\_transform}$ 

Delta method-based variance stabilizing transformation

## Description

Delta method-based variance stabilizing transformation

## Usage

```
acosh_transform(
  data,
  overdispersion = 0.05,
  size_factors = TRUE,
  ...,
  on_disk = NULL,
  verbose = FALSE
)

shifted_log_transform(
  data,
  overdispersion = 0.05,
  pseudo_count = 1/(4 * overdispersion),
  size_factors = TRUE,
  minimum_overdispersion = 0.001,
  ...,
```

acosh\_transform 3

```
on_disk = NULL,
verbose = FALSE
)
```

#### **Arguments**

data

any matrix-like object (e.g. matrix, dgCMatrix, DelayedArray, HDF5Matrix) with one column per sample and row per gene. It can also be an object of type glmGamPoi, in which case it is directly used to calculate the variance-stabilized values.

overdispersion

the simplest count model is the Poisson model. However, the Poisson model assumes that variance = mean. For many applications this is too rigid and the Gamma-Poisson allows a more flexible mean-variance relation ( $variance = mean + mean^2 * overdispersion$ ).

overdispersion can either be

- a single boolean that indicates if an overdispersion is estimated for each gene.
- a numeric vector of length nrow(data) fixing the overdispersion to those values.
- the string "global" to indicate that one dispersion is fit across all genes.

Note that overdispersion = 0 and overdispersion = FALSE are equivalent and both reduce the Gamma-Poisson to the classical Poisson model. Default: 0.05 which is roughly the overdispersion observed on ostensibly homogeneous cell lines.

size\_factors

in large scale experiments, each sample is typically of different size (for example different sequencing depths). A size factor is an internal mechanism of GLMs to correct for this effect.

size\_factors is either a numeric vector with positive entries that has the same lengths as columns in the data that specifies the size factors that are used. Or it can be a string that species the method that is used to estimate the size factors (one of c("normed\_sum", "deconvolution", "poscounts")). Note that "normed\_sum" and "poscounts" are fairly simple methods and can lead to suboptimal results. For the best performance, I recommend to use size\_factors = "deconvolution" which calls scran::calculateSumFactors(). However, you need to separately install the scran package from Bioconductor for this method to work. Also note that size\_factors = 1 and size\_factors = FALSE are equivalent. If only a single gene is given, no size factor is estimated (ie. size\_factors = 1). Default: "normed\_sum".

• • •

additional parameters for  $glmGamPoi::glm_gp()$  which is called in case overdispersion = TRUE.

on\_disk

a boolean that indicates if the dataset is loaded into memory or if it is kept on disk to reduce the memory usage. Processing in memory can be significantly faster than on disk. Default: NULL which means that the data is only processed in memory if data is an in-memory data structure.

verbose

boolean that decides if information about the individual steps are printed. Default: FALSE

pseudo\_count

instead of specifying the overdispersion, the shifted\_log\_transform is commonly parameterized with a pseudo-count (pseudo-count=1/(4\*overdispersion)). If both the pseudo-count and overdispersion is specified, the overdispersion is ignored. Default: 1/(4 \* overdispersion)

4 acosh\_transform

minimum\_overdispersion

the acosh\_transform converges against 2\*sqrt(x) for overdispersion == 0. However, the shifted\_log\_transform would just become 0, thus here we apply the minimum\_overdispersion to avoid this behavior.

#### Value

a matrix (or a vector if the input is a vector) with the transformed values.

#### **Functions**

- acosh\_transform: 1/sqrt(alpha) acosh(2 \* alpha \* x + 1)
- shifted\_log\_transform: 1/sqrt(alpha)log(4\*alpha\*x+1)

#### References

Ahlmann-Eltze, Constantin, and Wolfgang Huber. "Transformation and Preprocessing of Single-Cell RNA-Seq Data." bioRxiv (2021).

Ahlmann-Eltze, Constantin, and Wolfgang Huber. "glmGamPoi: Fitting Gamma-Poisson Generalized Linear Models on Single Cell Count Data." Bioinformatics (2020)

Dunn, Peter K., and Gordon K. Smyth. "Randomized quantile residuals." Journal of Computational and Graphical Statistics 5.3 (1996): 236-244.

Hafemeister, Christoph, and Rahul Satija. "Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression." Genome biology 20.1 (2019): 1-15.

Hafemeister, Christoph, and Rahul Satija. "Analyzing scRNA-seq data with the sctransform and offset models." (2020)

Lause, Jan, Philipp Berens, and Dmitry Kobak. "Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data." Genome Biology (2021).

#### See Also

acosh\_transform, shifted\_log\_transform, and residual\_transform

## **Examples**

estimate\_size\_factors 5

```
estimate_size_factors Estimate the Size Factors
```

## Description

Estimate the Size Factors

#### Usage

```
estimate_size_factors(Y, method, verbose = FALSE)
```

## **Arguments**

```
Y any matrix-like object (base::matrix(), DelayedArray, HDF5Matrix, Matrix::Matrix(), etc.)

method one of c("normed_sum", "deconvolution", "poscounts")
```

## Value

a vector with one size factor per column of Y

 $residual\_transform$ 

Residual-based Variance Stabilizing Transformation

## **Description**

Fit an intercept Gamma-Poisson model that corrects for sequencing depth and return the residuals as variance stabilized results for further downstream application, for which no proper count-based method exist or is performant enough (e.g., clustering, dimensionality reduction).

## Usage

```
residual_transform(
  data,
  residual_type = c("randomized_quantile", "pearson", "analytic_pearson"),
  clipping = FALSE,
  overdispersion = 0.05,
  size_factors = TRUE,
  offset_model = TRUE,
  overdispersion_shrinkage = TRUE,
  ridge_penalty = 2,
  on_disk = NULL,
  return_fit = FALSE,
  verbose = FALSE,
  ...
)
```

6 residual\_transform

#### **Arguments**

data

any matrix-like object (e.g. matrix, dgCMatrix, DelayedArray, HDF5Matrix) with one column per sample and row per gene. It can also be an object of type glmGamPoi, in which case it is directly used to calculate the variance-stabilized values.

residual\_type

a string that specifies what kind of residual is returned as variance stabilized-value.

"randomized\_quantile" The discrete nature of count distribution stops simple transformations from obtaining a truly standard normal residuals. The trick of of quantile randomized residuals is to match the cumulative density function of the Gamma-Poisson and the Normal distribution. Due to the discrete nature of Gamma-Poisson distribution, a count does not correspond to a single quantile of the Normal distribution, but to a range of possible value. This is resolved by randomly choosing one of the mapping values from the Normal distribution as the residual. This ensures perfectly normal distributed residuals, for the cost of introducing randomness. More details are available in the documentation of statmod::qresiduals() and the corresponding publication by Dunn and Smyth (1996).

"pearson" The Pearson residuals are defined as  $res = (y-m)/sqrt(m+m^2*theta)$ .

"analytic\_pearson" Similar to the method above, however, instead of estimating m using a GLM model fit, m is approximated by  $m_i j = (\sum_j y_{ij})(\sum_i y_{ij})/(\sum_{i,j} y_{ij})$ . For all details, see Lause et al. (2021). Note that overdispersion\_shrinkage and ridge\_penalty are ignored when fitting analytic Pearson residuals.

The two above options are the most common choices, however you can use any residual\_type supported by glmGamPoi::residuals.glmGamPoi(). Default: "randomized\_quantile"

clipping

a single boolean or numeric value specifying that all residuals are in the range [-clipping, +clipping]. If clipping = TRUE, we use the default of clipping = sqrt(ncol(data)) which is the default behavior for sctransform. Default: FALSE, which means no clipping is applied.

overdispersion

the simplest count model is the Poisson model. However, the Poisson model assumes that variance = mean. For many applications this is too rigid and the Gamma-Poisson allows a more flexible mean-variance relation ( $variance = mean + mean^2 * overdispersion$ ).

overdispersion can either be

- a single boolean that indicates if an overdispersion is estimated for each gene.
- a numeric vector of length nrow(data) fixing the overdispersion to those values.
- the string "global" to indicate that one dispersion is fit across all genes.

Note that overdispersion = 0 and overdispersion = FALSE are equivalent and both reduce the Gamma-Poisson to the classical Poisson model. Default: 0.05 which is roughly the overdispersion observed on ostensibly homogeneous cell lines.

offset\_model

boolean to decide if  $\beta_1$  in  $y = \beta_0 + \beta_1 log(sf)$ , is set to 1 (i.e., treating the log of the size factors as an offset ) or is estimated per gene. From a theoretical point, it should be fine to treat  $\beta_1$  as an offset, because a cell that is twice as big, should have twice as many counts per gene (without any gene-specific effects).

residual\_transform 7

However, sctransform suggested that it would be advantageous to nonetheless estimate  $\beta_0$  as it may counter data artifacts. On the other side, Lause et al. (2020) demonstrated that the estimating  $\beta_0$  and  $\beta_1$  together can be difficult. If you still want to fit sctransform's model, you can set the ridge\_penalty argument to a non-zero value, which shrinks  $\beta_1$  towards 1 and resolves the degeneracy.

Default: TRUE.

overdispersion\_shrinkage, size\_factors

arguments that are passed to the underlying call to glmGamPoi::glm\_gp(). De-

fault for each: TRUE.

ridge\_penalty another argument that is passed to glmGamPoi::glm\_gp(). It is ignored if

offset\_model = TRUE. Default: 2.

on\_disk a boolean that indicates if the dataset is loaded into memory or if it is kept on

disk to reduce the memory usage. Processing in memory can be significantly faster than on disk. Default: NULL which means that the data is only processed

in memory if data is an in-memory data structure.

return\_fit boolean to decide if the matrix of residuals is returned directly (return\_fit

= FALSE) or if in addition the glmGamPoi-fit is returned (return\_fit = TRUE).

Default: FALSE.

verbose boolean that decides if information about the individual steps are printed. De-

fault: FALSE

... additional parameters passed to glmGamPoi::glm\_gp().

#### **Details**

Internally, this method uses the glmGamPoi package. The function goes through the following steps

- 1. fit model using glmGamPoi::glm\_gp()
- 2. plug in the trended overdispersion estimates
- 3. call glmGamPoi::residuals.glmGamPoi() to calculate the residuals.

## Value

a matrix (or a vector if the input is a vector) with the transformed values. If return\_fit = TRUE, a list is returned with two elements: fit and Residuals.

#### References

Ahlmann-Eltze, Constantin, and Wolfgang Huber. "glmGamPoi: Fitting Gamma-Poisson Generalized Linear Models on Single Cell Count Data." Bioinformatics (2020)

Dunn, Peter K., and Gordon K. Smyth. "Randomized quantile residuals." Journal of Computational and Graphical Statistics 5.3 (1996): 236-244.

Hafemeister, Christoph, and Rahul Satija. "Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression." Genome biology 20.1 (2019): 1-15.

Hafemeister, Christoph, and Rahul Satija. "Analyzing scRNA-seq data with the sctransform and offset models." (2020)

Lause, Jan, Philipp Berens, and Dmitry Kobak. "Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data." Genome Biology (2021).

#### See Also

```
glmGamPoi::glm_gp(), glmGamPoi::residuals.glmGamPoi(), sctransform::vst(), statmod::qresiduals()
```

8 transformGamPoi

#### **Examples**

```
# Load a single cell dataset
sce <- TENxPBMCData::TENxPBMCData("pbmc4k")</pre>
# Reduce size for this example
set.seed(1)
sce_red <- sce[sample(which(rowSums2(counts(sce)) > 0), 1000),
                sample(ncol(sce), 200)]
counts(sce_red) <- as.matrix(counts(sce_red))</pre>
# Residual Based Variance Stabilizing Transformation
rq <- residual_transform(sce_red, residual_type = "randomized_quantile",</pre>
                          verbose = TRUE)
pearson <- residual_transform(sce_red, residual_type = "pearson", verbose = TRUE)</pre>
# Plot first two principal components
pearson_pca \leftarrow prcomp(t(pearson), rank. = 2)
rq_pca \leftarrow prcomp(t(rq), rank. = 2)
plot(rq_pca$x, asp = 1)
points(pearson_pca$x, col = "red")
```

transformGamPoi

Variance Stabilizing Transformation for Gamma Poisson Data

#### **Description**

Variance Stabilizing Transformation for Gamma Poisson Data

## Usage

```
transformGamPoi(
  data,
  transformation = c("acosh", "shifted_log", "randomized_quantile_residuals",
    "pearson_residuals", "analytic_pearson_residuals"),
  overdispersion = 0.05,
  size_factors = TRUE,
    ...,
  on_disk = NULL,
  verbose = FALSE
)
```

## **Arguments**

data

any matrix-like object (e.g. matrix, dgCMatrix, DelayedArray, HDF5Matrix) with one column per sample and row per gene. It can also be an object of type glmGamPoi, in which case it is directly used to calculate the variance-stabilized values.

transformation one of c("acosh", "shifted\_log", "randomized\_quantile\_residuals", "pearson\_residuals"). See acosh\_transform, shifted\_log\_transform, or residual\_transform for more information.

transformGamPoi 9

overdispersion the simplest count model is the Poisson model. However, the Poisson model assumes that variance = mean. For many applications this is too rigid and the Gamma-Poisson allows a more flexible mean-variance relation (variance =  $mean + mean^2 * overdispersion$ ).

overdispersion can either be

- a single boolean that indicates if an overdispersion is estimated for each gene.
- a numeric vector of length nrow(data) fixing the overdispersion to those values.
- the string "global" to indicate that one dispersion is fit across all genes.

Note that overdispersion = 0 and overdispersion = FALSE are equivalent and both reduce the Gamma-Poisson to the classical Poisson model. Default: 0.05 which is roughly the overdispersion observed on ostensibly homogeneous cell lines.

size\_factors

in large scale experiments, each sample is typically of different size (for example different sequencing depths). A size factor is an internal mechanism of GLMs to correct for this effect.

size\_factors is either a numeric vector with positive entries that has the same lengths as columns in the data that specifies the size factors that are used. Or it can be a string that species the method that is used to estimate the size factors (one of c("normed\_sum", "deconvolution", "poscounts")). Note that "normed\_sum" and "poscounts" are fairly simple methods and can lead to suboptimal results. For the best performance, I recommend to use size\_factors = "deconvolution" which calls scran::calculateSumFactors(). However, you need to separately install the scran package from Bioconductor for this method to work. Also note that size\_factors = 1 and size\_factors = FALSE are equivalent. If only a single gene is given, no size factor is estimated (ie. size\_factors = 1). Default: "normed\_sum".

additional parameters passed to acosh\_transform, shifted\_log\_transform, or residual transform

on\_disk

a boolean that indicates if the dataset is loaded into memory or if it is kept on disk to reduce the memory usage. Processing in memory can be significantly faster than on disk. Default: NULL which means that the data is only processed in memory if data is an in-memory data structure.

verbose

boolean that decides if information about the individual steps are printed. Default: FALSE

#### Value

a matrix (or a vector if the input is a vector) with the transformed values.

### References

Ahlmann-Eltze, Constantin, and Wolfgang Huber. "Transformation and Preprocessing of Single-Cell RNA-Seq Data." bioRxiv (2021).

Ahlmann-Eltze, Constantin, and Wolfgang Huber. "glmGamPoi: Fitting Gamma-Poisson Generalized Linear Models on Single Cell Count Data." Bioinformatics (2020)

Dunn, Peter K., and Gordon K. Smyth. "Randomized quantile residuals." Journal of Computational and Graphical Statistics 5.3 (1996): 236-244.

10 transformGamPoi

Hafemeister, Christoph, and Rahul Satija. "Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression." Genome biology 20.1 (2019): 1-15.

Hafemeister, Christoph, and Rahul Satija. "Analyzing scRNA-seq data with the sctransform and offset models." (2020)

Lause, Jan, Philipp Berens, and Dmitry Kobak. "Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data." Genome Biology (2021).

#### See Also

```
acosh_transform, shifted_log_transform, and residual_transform
```

## **Examples**

```
# Load a single cell dataset
sce <- TENxPBMCData::TENxPBMCData("pbmc4k")</pre>
# Reduce size for this example
set.seed(1)
sce_red <- sce[sample(which(rowSums2(counts(sce)) > 0), 1000),
                sample(ncol(sce), 200)]
assay(sce_red, "acosh") <- transformGamPoi(sce_red, "acosh")</pre>
assay(sce_red, "shifted_log") <- transformGamPoi(sce_red, "shifted_log")</pre>
# Residual Based Variance Stabilizing Transformation
rq <- transformGamPoi(sce_red, transformation = "randomized_quantile", on_disk = FALSE,</pre>
                        verbose = TRUE)
pearson <- transformGamPoi(sce_red, transformation = "pearson", on_disk = FALSE, verbose = TRUE)</pre>
plot(rowMeans2(counts(sce_red)), rowVars(assay(sce_red, "acosh")), log = "x")
points(rowMeans2(counts(sce_red)), rowVars(assay(sce_red, "shifted_log")), col = "red")
points(rowMeans2(counts(sce_red)), rowVars(rq), col = "blue")
# Plot first two principal components
acosh_pca <- prcomp(t(assay(sce_red, "acosh")), rank. = 2)</pre>
rq_pca \leftarrow prcomp(t(rq), rank. = 2)
pearson_pca <- prcomp(t(pearson), rank. = 2)</pre>
plot(acosh_pca$x, asp = 1)
points(rq_pca$x, col = "blue")
points(pearson_pca$x, col = "green")
```

# **Index**