Package 'seqgendiff'

May 24, 2020

Type Package

Title RNA-Seq Generation/Modification for Simulation

Version 1.2.2

Description Generates/modifies RNA-seq data for use in simulations. We provide a suite of functions that will add a known amount of signal to a real RNA-seq dataset. The advantage of using this approach over simulating under a theoretical distribution is that common/annoying aspects of the data are more preserved, giving a more realistic evaluation of your method. The main functions are select_counts(), thin_diff(), thin_lib(), thin_gene(), thin_2group(), thin_all(), and effective_cor(). See Gerard (2020) <doi:10.1186/s12859-020-3450-9> for details on the implemented methods.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/dcgerard/seqgendiff

BugReports http://github.com/dcgerard/seqgendiff/issues

RoxygenNote 7.1.0

Suggests covr, testthat (>= 2.1.0), SummarizedExperiment, DESeq2, knitr, rmarkdown, airway, limma, qvalue, edgeR, optmatch

Imports assertthat, irlba, sva, pdist, matchingR, clue, cate

VignetteBuilder knitr

NeedsCompilation no

Author David Gerard [aut, cre] (<https://orcid.org/0000-0001-9450-5023>)

Maintainer David Gerard <gerard.1787@gmail.com>

Repository CRAN

Date/Publication 2020-05-24 19:20:05 UTC

R topics documented:

`seqgendiff-package` *seqgendiff: Sequence Generation/Modification for Differential Expression Analysis and Beyond.*

Description

This package is designed to take real RNA-seq data and alter it by adding a known amount of signal. You can then use this modified dataset in simulation studies for differential expression analysis, factor analysis, confounder adjustment, or library size adjustment. The advantage of this way of simulating data is that you can see how your method behaves when the simulated data exhibit common (and annoying) features of real data. For example, in the real world data are not normally (or negative binomially) distributed and unobserved confounding is a major issue. This package will simulate data that exhibit these characteristics. The methods used in this package are described in detail in Gerard (2019).

seqgendiff Functions

`select_counts` Subsample the columns and rows of a real RNA-seq count matrix. You would then feed this sub-matrix into one of the thinning functions below.

`thin_diff` The function most users should be using for general-purpose binomial thinning. For the special applications of the two-group model or library/gene thinning, see the functions listed below.

`thin_2group` The specific application of thinning in the two-group model.

`thin_lib` The specific application of library size thinning.

`thin_gene` The specific application of total gene expression thinning.

`thin_all` The specific application of thinning all counts.

`effective_cor` Returns an estimate of the actual correlation between surrogate variables and a user-specified design matrix.

`ThinDataToSummarizedExperiment` Converts a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` Converts a ThinData object to a DESeqDataSet object.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.

`corassign` *Group assignment that is correlated with latent factors.*

Description

We extract latent factors from the log of `mat` using an SVD, then generate an underlying group-assignment variable from a conditional normal distribution (conditional on the latent factors). This underlying group-assignment variable is used to assign groups.

Usage

```
corassign(mat, nfac = NULL, corvec = NULL, return = c("group", "full"))
```

Arguments

`mat`	A matrix of count data. The rows index the individuals and the columns index the genes.
`nfac`	The number of latent factors. If `NULL`, then we will use `est.factor.num` from the cate package to choose the number of latent factors.
`corvec`	The vector of correlations. `corvec[i]` is the correlation between latent factor `i` and the underlying group-assignment variable. You can think of the correlations in `corvec` as a kind of "tetrachoric correlation." If `NULL`, then it assumes independence between factors and group assignment. Note that the correlations of the latent factors with the observed group-assignment vector (instead of the latent group-assignment vector) will be `corvec * sqrt(2 / pi)`.
`return`	What should we return? Just the group assignment (`"group"`) or a list of a bunch of things (`"full"`).

Details

If nfac is provided, then corvec must be the same length as nfac. If nfac is not provided, then it is assumed that the first nfac elements of corvec are the underlying correlations, if nfac turns out to be smaller than the length of corvec. If nfac turns out to be larger than the length of corvec, then the factors without defined correlations are assumed to have correlation 0.

Value

A list with some or all of the following elements:

x The vector of group assignments. 0L indicates membership to one group and 1L indicates membership to the other group.

nfac The number of assumed latent factors.

facmat A matrix, whose columns contain the latent factors.

groupfac The underlying group-assignment factor.

corvec The correlation vector. Note that this is the correlation between random variables observed in groupfac and facmat,

If return = "group", then the list only contains x.

Author(s)

David Gerard

References

- Jingshu Wang and Qingyuan Zhao (2015). cate: High Dimensional Factor Analysis and Confounder Adjusted Testing and Estimation. R package version 1.0.4. https://CRAN.R-project.org/package=cate

Examples

```
## Simulate data from given matrix of counts
## In practice, you would obtain Y from a real dataset, not simulate it.
set.seed(1)
nsamp <- 1000
ngene <- 10
Y <- matrix(stats::rpois(nsamp * ngene, lambda = 50), nrow = ngene)

## Set target correlation to be 0.9 and nfac to be 1
corvec <- 0.9
nfac   <- 1

## Group assignment
cout <- corassign(mat    = t(Y),
                  nfac   = nfac,
                  corvec = corvec,
                  return = "full")

## Correlation between facmat and groupfac should be about 0.9
```

```
cor(cout$facmat, cout$groupfac)

## Correlation between facmat and x should be about 0.9 * sqrt(2 / pi)
cor(cout$facmat, cout$x)
corvec * sqrt(2 / pi)
```

`effective_cor` *Estimates the effective correlation.*

Description

Will return the estimated correlation between the design matrix and the surrogate variables when you assign a target correlation. The method is described in detail in Gerard (2020).

Usage

```
effective_cor(
  design_perm,
  sv,
  target_cor,
  calc_first = c("cor", "mean"),
  method = c("optmatch", "hungarian", "marriage"),
  iternum = 1000
)
```

Arguments

`design_perm`	A numeric design matrix whose rows are to be permuted (thus controlling the amount by which they are correlated with the surrogate variables). The rows index the samples and the columns index the variables. The intercept should *not* be included (though see Section "Unestimable Components").
`sv`	A matrix of surrogate variables
`target_cor`	A numeric matrix of target correlations between the variables in `design_perm` and the surrogate variables. The rows index the observed covariates and the columns index the surrogate variables. That is, `target_cor[i,j]` specifies the target correlation between the `i`th column of `design_perm` and the `j`th surrogate variable. The surrogate variables are estimated either using factor analysis or surrogate variable analysis (see the parameter `use_sva`). The number of columns in `target_cor` specifies the number of surrogate variables. Set `target_cor` to `NULL` to indicate that `design_perm` and the surrogate variables are independent.
`calc_first`	Should we calculate the correlation of the mean `design_perm` and `sv` (`calc_first = "mean"`), or should we calculate the mean of the correlations between `design_perm` and `sv` (`calc_first = "cor"`)? This should only be changed by expert users.

method	Should we use the optimal matching technique from Hansen and Klopfer (2006) ("optmatch"), the Gale-Shapley algorithm for stable marriages ("marriage") (Gale and Shapley, 1962) as implemented in the matchingR package, or the Hungarian algorithm (Papadimitriou and Steiglitz, 1982) ("hungarian") as implemented in the clue package (Hornik, 2005)? The "optmatch" method works really well but does take a lot more computational time if you have, say, 1000 samples. If you use the "optmatch" option, you should note that the optmatch package uses a super strange license: https://cran.r-project.org/package=optmatch/LICENSE. If this license doesn't work for you (because you are not in academia, or because you don't believe in restrictive licenses), then try out the "hungarian" method.
iternum	The total number of simulated correlations to consider.

Details

This function permutes the rows of design_perm many times, each time calculating the Pearson correlation between the columns of design_perm and the columns of sv. It then returns the averages of these Pearson correlations. The permutation is done using permute_design.

Value

A matrix of correlations. The rows index the observed covariates and the columns index the surrogate variables. Element (i, j) is the estimated correlation between the ith variable in design_perm and the jth variable in sv.

Author(s)

David Gerard

References

- Gale, David, and Lloyd S. Shapley. "College admissions and the stability of marriage." The American Mathematical Monthly 69, no. 1 (1962): 9-15.
- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.
- Hansen, Ben B., and Stephanie Olsen Klopfer. "Optimal full matching and related designs via network flows." Journal of computational and Graphical Statistics 15, no. 3 (2006): 609-627.
- Hornik K (2005). "A CLUE for CLUster Ensembles." Journal of Statistical Software, 14(12). doi: 10.18637/jss.v014.i12
- C. Papadimitriou and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs: Prentice Hall.

Examples

```
## Generate the design matrices and set target correlation -----------------
n <- 10
design_perm <- cbind(rep(c(0, 1), each = n / 2),
                     rep(c(0, 1), length.out = n))
```

```
sv <- matrix(rnorm(n))
target_cor <- matrix(c(0.9, 0.1), ncol = 1)

## Get estimated true correlation ------------------------------------------
## You should use a much larger iternum in practice
effective_cor(design_perm = design_perm,
              sv = sv,
              target_cor = target_cor,
              iternum = 10)
```

`est_sv` *Estimate the surrogate variables.*

Description

This will use either sva or an SVD on the residuals of a regression of `mat` on `design_obs` to estimate the surrogate variables.

Usage

```
est_sv(mat, n_sv, design_obs, use_sva = FALSE)
```

Arguments

`mat`	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
`n_sv`	The number of surrogate variables.
`design_obs`	A numeric matrix of observed covariates that are NOT to be a part of the signal generating process. Only used in estimating the surrogate variables (if `target_cor` is not `NULL`). The intercept should *not* be included (it will sometimes produce an error if it is included).
`use_sva`	A logical. Should we use surrogate variable analysis (Leek and Storey, 2008) using `design_obs` to estimate the hidden covariates (`TRUE`) or should we just do an SVD on `log2(mat + 0.5)` after regressing out `design_obs` (`FALSE`)? Setting this to `TRUE` allows the surrogate variables to be correlated with the observed covariates, while setting this to `FALSE` assumes that the surrogate variables are orthogonal to the observed covariates. This option only matters if `design_obs` is not `NULL`. Defaults to `FALSE`.

Value

A matrix of estimated surrogate variables. The columns index the surrogate variables and the rows index the individuals. The surrogate variables are centered and scaled to have mean 0 and variance 1.

Author(s)

David Gerard

fix_cor *Fixes an invalid target correlation.*

Description

Shrinks the target correlation using a uniform scaling factor so that the overall correlation matrix is positive semi-definite. The method is described in detail in Gerard (2020).

Usage

```
fix_cor(design_perm, target_cor, num_steps = 51)
```

Arguments

design_perm	A numeric design matrix whose rows are to be permuted (thus controlling the amount by which they are correlated with the surrogate variables). The rows index the samples and the columns index the variables. The intercept should *not* be included (though see Section "Unestimable Components").
target_cor	A numeric matrix of target correlations between the variables in design_perm and the surrogate variables. The rows index the observed covariates and the columns index the surrogate variables. That is, target_cor[i,j] specifies the target correlation between the ith column of design_perm and the jth surrogate variable. The surrogate variables are estimated either using factor analysis or surrogate variable analysis (see the parameter use_sva). The number of columns in target_cor specifies the number of surrogate variables. Set target_cor to NULL to indicate that design_perm and the surrogate variables are independent.
num_steps	The number of steps between 0 and 1 to take in the grid search for the shrinkage factor. The step-size would be 1 / (num_steps -1).

Details

Let W = cor(design_perm). Let R = target_cor. Then the overall correlation matrix is:

$$\begin{pmatrix} W & R \\ R' & I_K \end{pmatrix}.$$

This function applies a multiplicative scaling factor to R until the above matrix is positive semi-definite. That is, it finds a between 0 and 1 such that

$$\begin{pmatrix} W & aR \\ aR' & I_K \end{pmatrix}$$

is positive semi-definite.

Value

A matrix of correlations the same dimension as target_cor. Actually, the returned matrix is a * target_cor, where a was determined to make the overall correlation matrix positive semi-definite.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.

Examples

```
n <- 10
design_perm <- matrix(rep(c(0, 1), length.out = n))
target_cor <- matrix(seq(1, 0, length.out = 10), nrow = 1)
new_cor <- seqgendiff:::fix_cor(design_perm = design_perm, target_cor = target_cor)
new_cor / target_cor

## In the case of one observed covariate, the requirement is just that
## the sum of squared correlations is less than or equal to one.
sum(target_cor ^ 2)
sum(new_cor ^ 2)
```

`permute_design` *Permute the design matrix so that it is approximately correlated with the surrogate variables.*

Description

Permute the design matrix so that it is approximately correlated with the surrogate variables.

Usage

```
permute_design(
  design_perm,
  sv,
  target_cor,
  method = c("optmatch", "hungarian", "marriage")
)
```

Arguments

`design_perm`	A numeric design matrix whose rows are to be permuted (thus controlling the amount by which they are correlated with the surrogate variables). The rows index the samples and the columns index the variables. The intercept should *not* be included (though see Section "Unestimable Components").
`sv`	A matrix of surrogate variables

`target_cor` A numeric matrix of target correlations between the variables in `design_perm` and the surrogate variables. The rows index the observed covariates and the columns index the surrogate variables. That is, `target_cor[i,j]` specifies the target correlation between the `i`th column of `design_perm` and the `j`th surrogate variable. The surrogate variables are estimated either using factor analysis or surrogate variable analysis (see the parameter `use_sva`). The number of columns in `target_cor` specifies the number of surrogate variables. Set `target_cor` to `NULL` to indicate that `design_perm` and the surrogate variables are independent.

`method` Should we use the optimal matching technique from Hansen and Klopfer (2006) (`"optmatch"`), the Gale-Shapley algorithm for stable marriages (`"marriage"`) (Gale and Shapley, 1962) as implemented in the matchingR package, or the Hungarian algorithm (Papadimitriou and Steiglitz, 1982) (`"hungarian"`) as implemented in the clue package (Hornik, 2005)? The `"optmatch"` method works really well but does take a lot more computational time if you have, say, 1000 samples. If you use the `"optmatch"` option, you should note that the optmatch package uses a super strange license: https://cran.r-project.org/package=optmatch/LICENSE. If this license doesn't work for you (because you are not in academia, or because you don't believe in restrictive licenses), then try out the `"hungarian"` method.

Value

A list with two elements:

`design_perm` A row-permuted version of the user-provided `design_perm`.

`latent_var` A matrix of the latent variables on which `design_perm` was matched.

Author(s)

David Gerard

References

- Hansen, Ben B., and Stephanie Olsen Klopfer. "Optimal full matching and related designs via network flows." Journal of computational and Graphical Statistics 15, no. 3 (2006): 609-627.
- Gale, David, and Lloyd S. Shapley. "College admissions and the stability of marriage." The American Mathematical Monthly 69, no. 1 (1962): 9-15.
- C. Papadimitriou and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs: Prentice Hall.
- Hornik K (2005). "A CLUE for CLUster Ensembles." Journal of Statistical Software, 14(12). doi: 10.18637/jss.v014.i12

poisthin *Apply Poisson thinning to a matrix of count data.*

Description

This is now defunct. Please try out `select_counts` and `thin_2group`.

Usage

```
poisthin(
  mat,
  nsamp = nrow(mat),
  ngene = ncol(mat),
  gselect = c("max", "random", "rand_max", "custom", "mean_max"),
  gvec = NULL,
  skip_gene = 0L,
  signal_fun = stats::rnorm,
  signal_params = list(mean = 0, sd = 1),
  prop_null = 1,
  alpha = 0,
  group_assign = c("frac", "random", "cor"),
  group_prop = 0.5,
  corvec = NULL
)
```

Arguments

`mat`	A matrix of count data. The rows index the individuals and the columns index the genes.
`nsamp`	The number of samples to select from `mat`.
`ngene`	The number of genes to select from `mat`.
`gselect`	How should we select the subset of genes? Should we choose the `ngene` most median expressed genes (`"max"`), a random sample of the genes (`"random"`), a random sample of the most expressed genes (`"rand_max"`), a user-provided list (`"custom"`), or by maximum mean expression level (`"mean_max"`)? If `"custom"`, then `gvec` should be specified. Expression levels of a gene are measured by median expression across individuals with ties broken by mean expression.
`gvec`	A logical of length `ncol(mat)`. A `TRUE` in position i indicates inclusion into the smaller dataset. Hence, `sum(gvec)` should equal `ngene`.
`skip_gene`	The number of maximally expressed genes to skip. Not used if `gselect = "custom"`.
`signal_fun`	A function that returns the signal. This should take as input `n` for the number of samples to return and then return only a vector of samples.
`signal_params`	A list of additional arguments to pass to `signal_fun`.
`prop_null`	The proportion of genes that are null.

`alpha` If b is an effect and s is an empirical standard deviation, then we model b/s^{α} as being exchangeable.

`group_assign` How should we assign groups? Exactly specifying the proportion of individuals in each group (`"frac"`), with a Bernoulli distribution (`"random"`), or correlated with latent factors (`"cor"`)? If `group_assign = "cor"`, then you have to specify `corvec`. If `group_assign = "frac"` or `group_assign = "random"`, then the proportion of samples in each group is specified with the `group_prop` argument.

`group_prop` The proportion of individuals that are in group 1. This proportion is deterministic if `group_assign = "frac"`, and is the expected proportion if `group_assign = "random"`. This argument is not used if `group_assign = "cor"`.

`corvec` A vector of correlations. `corvec[i]` is the correlation of the latent group assignment vector with the ith latent confounder. Only used if `group_assign = "cor"`. This vector is constrained so that `crossprod(corvec) < 1`. The number of latent factors is taken to be the length of corvec. Note that the correlations of the latent factors with the observed group-assignment vector (instead of the latent group-assignment vector) will be `corvec * sqrt(2 / pi)`.

Details

Given a matrix of RNA-seq counts, this function will randomly select two groups of samples and add signal to a known proportion of the genes. This signal is the log (base 2) effect size of the group indicator in a linear model. The user may specify the distribution of the effects.

The Poisson thinning approach first randomly assigns samples to be in one of two groups. Then, given this assignment, will Binomially sample counts with a sample size of the gene expression counts and a probability that is a function of the effect size. For details, see Gerard and Stephens (2017).

Value

A list with the following elements:

- `Y`: A matrix of altered counts with `nsamp` rows and `ngene` columns.
- `X`: A design matrix. The first column contains a vector ones (for an intercept term) and the second column contains an indicator for group membership.
- `beta`: The approximately true effect sizes of $log(Y)\ X\beta$.
- `corassign`: The output from the call to `corassign`. Only returned if `group_assign = "cor"`.

Author(s)

David Gerard

References

Gerard, D., & Stephens, M. (2017). Unifying and generalizing methods for removing unwanted variation based on negative controls. arXiv preprint arXiv:1705.08393.

Examples

```
## Simulate data from given matrix of counts
## In practice, you would obtain Y from a real dataset, not simulate it.
set.seed(1)
nsamp <- 10
ngene <- 1000
Y <- matrix(stats::rpois(nsamp * ngene, lambda = 50), nrow = ngene)


## Apply thinning
poisout <- poisthin(mat          = t(Y),
                    nsamp        = 9,
                    ngene        = 999,
                    signal_fun   = stats::rnorm,
                    signal_params = list(mean = 0, sd = 1),
                    prop_null    = 0.9)

## Dimension of count matrix is smaller.
dim(poisout$Y)

## Can verify signal was added by estimating it with lm().
betahat <- coef(lm(log2(poisout$Y + 1) ~ poisout$X[, 2]))[2, ]
plot(poisout$beta, betahat, xlab = "Coefficients", ylab = "Estimates")
abline(0, 1, col = 2, lty = 2)
```

---

select_counts *Subsample the rows and columns of a count matrix.*

Description

It is a good idea to subsample (each iteration) the genes and samples from a real RNA-seq dataset prior to applying thin_diff (and related functions) so that your conclusions are not dependent on the specific structure of your dataset. This function is designed to efficiently do this for you.

Usage

```
select_counts(
  mat,
  nsamp = ncol(mat),
  ngene = nrow(mat),
  gselect = c("random", "max", "mean_max", "custom"),
  gvec = NULL,
  filter_first = FALSE,
  nskip = 0L
)
```

Arguments

mat	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
nsamp	The number of samples (columns) to select from mat.
ngene	The number of genes (rows) to select from mat.
gselect	How should we select the subset of genes? Options include: random Randomly select the genes, with each gene having an equal probability of being included in the subsampled matrix. max Choose the ngene most median-expressed genes. Ties are broken by mean-expression. mean_max Choose the ngene most mean-expressed genes. custom A user-specified list of genes. If gselect = "custom" then gvec needs to be non-NULL.
gvec	A logical vector of length nrow(mat). A TRUE in position i indicates inclusion into the smaller dataset. Hence, sum(gvec) should equal ngene.
filter_first	Should we first filter genes by the method of Chen et al. (2016) (TRUE) or not (FALSE)? If TRUE then the edgeR package should be installed.
nskip	The number of median-maximally expressed genes to skip. Not used if gselect = "custom".

Details

The samples (columns) are chosen randomly, with each sample having an equal probability of being in the sub-matrix. The genes are selected according to one of four schemes (see the description of the gselect argument).

If you have edgeR installed, then some functionality is provided for filtering out the lowest expressed genes prior to applying subsampling (see the filter_first argument). This filtering scheme is described in Chen et al. (2016). If you want more control over this filtering, you should use the filterByExpr function from edgeR directly. You can install edgeR by following instructions here: https://doi.org/doi:10.18129/B9.bioc.edgeR.

Value

A numeric matrix, which is a ngene by nsamp sub-matrix of mat. If rownames(mat) is NULL, then the row names of the returned matrix are the indices in mat of the selected genes. If colnames(mat) is NULL, then the column names of the returned matrix are the indices in mat of the selected samples.

Author(s)

David Gerard

References

- Chen, Yunshun, Aaron TL Lun, and Gordon K. Smyth. "From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline." F1000Research 5 (2016).

Examples

```
## Simulate data from given matrix of counts
## In practice, you would obtain mat from a real dataset, not simulate it.
set.seed(1)
n   <- 100
p   <- 1000
mat <- matrix(stats::rpois(n * p, lambda = 50), nrow = p)

## Subsample the matrix, then feed it into a thinning function
submat <- select_counts(mat = mat, nsamp = 10, ngene = 100)
thout  <- thin_2group(mat = submat, prop_null = 0.5)

## The rownames and colnames (if NULL in mat) tell you which genes/samples
## were selected.
rownames(submat)
colnames(submat)
```

`summary.ThinData` *Provide summary output of a ThinData S3 object.*

Description

Provide summary output of a ThinData S3 object.

Usage

```
## S3 method for class 'ThinData'
summary(object, ...)
```

Arguments

`object`	A ThinData S3 object. This is generally output by either `thin_diff`, `thin_2group`, or `thin_lib`.
`...`	Not used.

Value

Returns nothing. Prints out some summary information on `object`.

Author(s)

David Gerard

ThinDataToDESeqDataSet *Converts a ThinData S3 object into a DESeqDataSet S4 object.*

Description

The design formula in the resulting DESeqDataSet is just the sum of all variables in `designmat` from the ThinData object (except the intercept term). You should change this design formula if you want to study other models.

Usage

```
ThinDataToDESeqDataSet(obj)
```

Arguments

`obj`	A ThinData S3 object. This is generally output by either `thin_diff`, `thin_2group`, `thin_lib`, `thin_gene`, or `thin_all`.

Value

A `DESeqDataSet` S4 object. This will allow you to insert the simulated data directly into DESeq2.

Author(s)

David Gerard

Examples

```
## Generate simulated data and modify using thin_diff().
## In practice, you would use real data, not simulated.
set.seed(1)
n <- 10
p <- 1000
Z <- matrix(abs(rnorm(n, sd = 4)))
alpha <- matrix(abs(rnorm(p, sd = 1)))
mat <- round(2^(alpha %*% t(Z) + abs(matrix(rnorm(n * p, sd = 5),
                                            nrow = p,
                                            ncol = n))))
design_perm <- cbind(rep(c(0, 1), length.out = n), runif(n))
coef_perm   <- matrix(rnorm(p * ncol(design_perm), sd = 6), nrow = p)
design_obs  <- matrix(rnorm(n), ncol = 1)
target_cor <- matrix(c(0.9, 0))
thout <- thin_diff(mat           = mat,
                   design_perm   = design_perm,
                   coef_perm     = coef_perm,
                   target_cor    = target_cor,
                   design_obs    = design_obs,
```

```
                    permute_method = "hungarian")

## Convert ThinData object to DESeqDataSet object.
seobj <- ThinDataToDESeqDataSet(thout)
class(seobj)

## The "O1" variable in the colData corresponds to design_obs.
## The "P1" and "P2" variables in colData correspond to design_perm.
seobj
```

`ThinDataToSummarizedExperiment` *Converts a ThinData S3 object into a SummarizedExperiment S4 object.*

Description

This only keeps the `mat`, `design_obs`, `designmat`, and `coefmat` elements of the ThinData object.

Usage

```
ThinDataToSummarizedExperiment(obj)
```

Arguments

`obj`	A ThinData S3 object. This is generally output by either `thin_diff`, `thin_2group`, `thin_lib`, `thin_gene`, or `thin_all`.

Value

A `SummarizedExperiment` S4 object. This is often used in Bioconductor when performing differential expression analysis.

Author(s)

David Gerard

Examples

```
## Generate simulated data and modify using thin_diff().
## In practice, you would use real data, not simulated.
set.seed(1)
n <- 10
p <- 1000
Z <- matrix(abs(rnorm(n, sd = 4)))
alpha <- matrix(abs(rnorm(p, sd = 1)))
```

```
mat <- round(2^(alpha %*% t(Z) + abs(matrix(rnorm(n * p, sd = 5),
                                            nrow = p,
                                            ncol = n))))
design_perm <- cbind(rep(c(0, 1), length.out = n), runif(n))
coef_perm   <- matrix(rnorm(p * ncol(design_perm), sd = 6), nrow = p)
design_obs  <- matrix(rnorm(n), ncol = 1)
target_cor <- matrix(c(0.9, 0))
thout <- thin_diff(mat            = mat,
                   design_perm    = design_perm,
                   coef_perm      = coef_perm,
                   target_cor     = target_cor,
                   design_obs     = design_obs,
                   permute_method = "hungarian")

## Convert ThinData object to SummarizedExperiment object.
seobj <- ThinDataToSummarizedExperiment(thout)
class(seobj)

## The "O1" variable in the colData corresponds to design_obs.
## The "P1" and "P2" variables in colData correspond to design_perm.
seobj
```

---

thin_2group *Binomial thinning in the two-group model.*

Description

Given a matrix of real RNA-seq counts, this function will randomly assign samples to one of two groups, draw the log2-fold change in expression between two groups for each gene, and add this signal to the RNA-seq counts matrix. The user may specify the proportion of samples in each group, the proportion of null genes (where the log2-fold change is 0), and the signal function. This is a specific application of the general binomial thinning approach implemented in thin_diff.

Usage

```
thin_2group(
  mat,
  prop_null = 1,
  signal_fun = stats::rnorm,
  signal_params = list(mean = 0, sd = 1),
  group_prop = 0.5,
  corvec = NULL,
  alpha = 0,
  permute_method = c("optmatch", "hungarian", "marriage"),
  type = c("thin", "mult")
)
```

Arguments

mat	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
prop_null	The proportion of genes that are null.
signal_fun	A function that returns the signal. This should take as input n for the number of samples to return and then return only a vector of samples. Additional parameters may be passed through signal_params.
signal_params	A list of additional arguments to pass to signal_fun.
group_prop	The proportion of individuals that are in group 1.
corvec	A vector of target correlations. corvec[i] is the target correlation of the latent group assignment vector with the ith surrogate variable. The default is to set this to NULL, in which case group assignment is made independently of any unobserved confounding.
alpha	The scaling factor for the signal distribution from Stephens (2016). If $x_1, x_2, ..., x_n$ are drawn from signal_fun, then the signal is set to $x_1 s_1^\alpha, x_2 s_2^\alpha, ..., x_n s_n^\alpha$, where s_g is the empirical standard deviation of gene g. Setting this to 0 means that the effects are exchangeable, setting this to 1 corresponds to the p-value prior of Wakefield (2009). You would rarely set this to anything but 0 or 1.
permute_method	Should we use the optimal matching technique from Hansen and Klopfer (2006) ("optmatch"), the Gale-Shapley algorithm for stable marriages ("marriage") (Gale and Shapley, 1962) as implemented in the matchingR package, or the Hungarian algorithm (Papadimitriou and Steiglitz, 1982) ("hungarian") as implemented in the clue package (Hornik, 2005)? The "optmatch" method works really well but does take a lot more computational time if you have, say, 1000 samples. If you use the "optmatch" option, you should note that the optmatch package uses a super strange license: https://cran.r-project.org/package=optmatch/LICENSE. If this license doesn't work for you (because you are not in academia, or because you don't believe in restrictive licenses), then try out the "hungarian" method.
type	Should we apply binomial thinning (type = "thin") or just naive multiplication of the counts (type = "mult"). You should always have this set to "thin".

Details

The specific application of binomial thinning to the two-group model was used in Gerard and Stephens (2017) and Gerard and Stephens (2018). This is a specific case of the general method described in Gerard (2020).

Value

A list-like S3 object of class ThinData. Components include some or all of the following:

mat The modified matrix of counts.

designmat The design matrix of variables used to simulate signal. This is made by column-binding design_fixed and the permuted version of design_perm.

coefmat A matrix of coefficients corresponding to designmat.

`design_obs` Additional variables that should be included in your design matrix in downstream fittings. This is made by column-binding the vector of 1's with `design_obs`.

`sv` A matrix of estimated surrogate variables. In simulation studies you would probably leave this out and estimate your own surrogate variables.

`cormat` A matrix of target correlations between the surrogate variables and the permuted variables in the design matrix. This might be different from the `target_cor` you input because we pass it through `fix_cor` to ensure positive semi-definiteness of the resulting covariance matrix.

`matching_var` A matrix of simulated variables used to permute `design_perm` if the `target_cor` is not `NULL`.

Author(s)

David Gerard

References

- Gale, David, and Lloyd S. Shapley. "College admissions and the stability of marriage." The American Mathematical Monthly 69, no. 1 (1962): 9-15.
- Gerard, David and Matthew Stephens (2017). "Unifying and generalizing methods for removing unwanted variation based on negative controls." *arXiv* preprint arXiv:1705.08393.
- David Gerard and Matthew Stephens (2018). "Empirical Bayes shrinkage and false discovery rate estimation, allowing for unwanted variation." *Biostatistics*, doi: 10.1093/biostatistics/kxy029.
- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.
- Hansen, Ben B., and Stephanie Olsen Klopfer. "Optimal full matching and related designs via network flows." Journal of computational and Graphical Statistics 15, no. 3 (2006): 609-627.
- Hornik K (2005). "A CLUE for CLUster Ensembles." Journal of Statistical Software, 14(12). doi: 10.18637/jss.v014.i12
- C. Papadimitriou and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs: Prentice Hall.
- Stephens, Matthew. "False discovery rates: a new deal." Biostatistics 18, no. 2 (2016): 275-294.
- Wakefield, Jon. "Bayes factors for genome-wide association studies: comparison with P-values." Genetic epidemiology 33, no. 1 (2009): 79-86.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_diff` For the more general thinning approach.

`ThinDataToSummarizedExperiment` For converting a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` For converting a ThinData object to a DESeqDataSet object.

Examples

```
## Simulate data from given matrix of counts
## In practice, you would obtain Y from a real dataset, not simulate it.
set.seed(1)
nsamp <- 10
ngene <- 1000
Y <- matrix(stats::rpois(nsamp * ngene, lambda = 50), nrow = ngene)
thinout <- thin_2group(mat           = Y,
                       prop_null     = 0.9,
                       signal_fun    = stats::rexp,
                       signal_params = list(rate = 0.5))

## 90 percent of genes are null
mean(abs(thinout$coef) < 10^-6)

## Check the estimates of the log2-fold change
Ynew <- log2(t(thinout$mat + 0.5))
X    <- thinout$designmat
Bhat <- coef(lm(Ynew ~ X))["X", ]
plot(thinout$coefmat,
     Bhat,
     xlab = "log2-fold change",
     ylab = "Estimated log2-fold change")
abline(0, 1, col = 2, lwd = 2)
```

`thin_all` *Binomial thinning for altering read-depth.*

Description

Given a matrix of real RNA-seq counts, this function will apply a thinning factor uniformly to every count in this matrix. This uniformly lowers the read-depth for the entire dataset. The thinning factor should be provided on the log2-scale. This is a specific application of the binomial thinning approach in `thin_diff`. Though this particular form of thinning was used by Robinson and Storey (2014) in the context of deriving read-depth suggestions. It is also described in detail in Gerard (2020).

Usage

```
thin_all(mat, thinlog2, type = c("thin", "mult"))
```

Arguments

`mat`	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
`thinlog2`	A numeric scalar. This is the amount to shrink each count in `mat` (on the log2-scale). For example, a value of 0 means that we do not thin, a value of 1 means that we thin by a factor of 2, a value of 2 means we thin by a factor of 4, etc.

`type` Should we apply binomial thinning (`type = "thin"`) or just naive multiplication of the counts (`type = "mult"`). You should always have this set to `"thin"`.

Value

A list-like S3 object of class `ThinData`. Components include some or all of the following:

`mat` The modified matrix of counts.

`designmat` The design matrix of variables used to simulate signal. This is made by column-binding `design_fixed` and the permuted version of `design_perm`.

`coefmat` A matrix of coefficients corresponding to `designmat`.

`design_obs` Additional variables that should be included in your design matrix in downstream fittings. This is made by column-binding the vector of 1's with `design_obs`.

`sv` A matrix of estimated surrogate variables. In simulation studies you would probably leave this out and estimate your own surrogate variables.

`cormat` A matrix of target correlations between the surrogate variables and the permuted variables in the design matrix. This might be different from the `target_cor` you input because we pass it through `fix_cor` to ensure positive semi-definiteness of the resulting covariance matrix.

`matching_var` A matrix of simulated variables used to permute `design_perm` if the `target_cor` is not `NULL`.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.
- Robinson, David G., and John D. Storey. "subSeq: determining appropriate sequencing depth through efficient read subsampling." Bioinformatics 30, no. 23 (2014): 3424-3426.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_diff` For the more general thinning approach.

`thin_lib` For thinning sample-wise.

`thin_gene` For thinning gene-wise.

`ThinDataToSummarizedExperiment` For converting a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` For converting a ThinData object to a DESeqDataSet object.

Examples

```
## Generate count data and set thinning factor
## In practice, you would obtain mat from a real dataset, not simulate it.
set.seed(1)
n <- 10
p <- 1000
lambda <- 1000
mat <- matrix(lambda, ncol = n, nrow = p)
thinlog2 <- 1

## Thin read-depths
thout <- thin_all(mat = mat, thinlog2 = thinlog2)

## Compare empirical and theoretical proportions
mean(thout$mat) / lambda
2 ^ -thinlog2
```

`thin_base` *Base binomial thinning function.*

Description

Given a matrix of counts (Y) where $log_2(E[Y]) = Q$, a design matrix (X), and a matrix of coefficients (B), `thin_diff` will generate a new matrix of counts such that $log_2(E[Y]) = BX' + u1' + Q$, where u is some vector of intercept coefficients. This function is used by all other thinning functions. The method is described in detail in Gerard (2020).

Usage

```
thin_base(mat, designmat, coefmat, relative = TRUE, type = c("thin", "mult"))
```

Arguments

`mat`	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
`designmat`	A design matrix. The rows index the samples and the columns index the variables. The intercept should *not* be included.
`coefmat`	A matrix of coefficients. The rows index the genes and the columns index the samples.
`relative`	A logical. Should we apply relative thinning (`TRUE`) or absolute thinning (`FALSE`). Only experts should change the default.
`type`	Should we apply binomial thinning (`type = "thin"`) or just naive multiplication of the counts (`type = "mult"`). You should always have this set to `"thin"`.

Value

A matrix of new RNA-seq read-counts. This matrix has the signal added from `designmat` and `coefmat`.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_diff` For the function most users should be using for general-purpose binomial thinning.

`thin_2group` For the specific application of thinning in the two-group model.

`thin_lib` For the specific application of library size thinning.

`thin_gene` For the specific application of total gene expression thinning.

`thin_all` For the specific application of thinning all counts uniformly.

Examples

```
## Simulate data from given matrix of counts
## In practice, you would obtain Y from a real dataset, not simulate it.
set.seed(1)
nsamp <- 10
ngene <- 1000
Y <- matrix(stats::rpois(nsamp * ngene, lambda = 100), nrow = ngene)
X <- matrix(rep(c(0, 1), length.out = nsamp))
B <- matrix(seq(3, 0, length.out = ngene))
Ynew <- thin_base(mat = Y, designmat = X, coefmat = B)

## Demonstrate how the log2 effect size is B
Bhat <- coefficients(lm(t(log2(Ynew)) ~ X))["X", ]
plot(B, Bhat, xlab = "Coefficients", ylab = "Coefficient Estimates")
abline(0, 1, col = 2, lwd = 2)
```

thin_diff *Binomial thinning for differential expression analysis.*

Description

Given a matrix of real RNA-seq counts, this function will add a known amount of signal to the count matrix. This signal is given in the form of a Poisson / negative binomial / mixture of negative binomials generalized linear model with a log (base 2) link. The user may specify any arbitrary design matrix and coefficient matrix. The user may also control for the amount of correlation between the observed covariates and any unobserved surrogate variables. The method is described in detail in Gerard (2020).

Usage

```
thin_diff(
  mat,
  design_fixed = NULL,
  coef_fixed = NULL,
  design_perm = NULL,
  coef_perm = NULL,
  target_cor = NULL,
  use_sva = FALSE,
  design_obs = NULL,
  relative = TRUE,
  change_colnames = TRUE,
  permute_method = c("optmatch", "hungarian", "marriage"),
  type = c("thin", "mult")
)
```

Arguments

mat	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
design_fixed	A numeric design matrix whose rows are fixed and not to be permuted. The rows index the samples and the columns index the variables. The intercept should *not* be included (though see Section "Unestimable Components").
coef_fixed	A numeric matrix. The coefficients corresponding to design_fixed. The rows index the genes and the columns index the variables.
design_perm	A numeric design matrix whose rows are to be permuted (thus controlling the amount by which they are correlated with the surrogate variables). The rows index the samples and the columns index the variables. The intercept should *not* be included (though see Section "Unestimable Components").
coef_perm	A numeric matrix. The coefficients corresponding to design_perm. The rows index the genes and the columns index the variables.

target_cor A numeric matrix of target correlations between the variables in design_perm and the surrogate variables. The rows index the observed covariates and the columns index the surrogate variables. That is, target_cor[i,j] specifies the target correlation between the ith column of design_perm and the jth surrogate variable. The surrogate variables are estimated either using factor analysis or surrogate variable analysis (see the parameter use_sva). The number of columns in target_cor specifies the number of surrogate variables. Set target_cor to NULL to indicate that design_perm and the surrogate variables are independent.

use_sva A logical. Should we use surrogate variable analysis (Leek and Storey, 2008) using design_obs to estimate the hidden covariates (TRUE) or should we just do an SVD on log2(mat + 0.5) after regressing out design_obs (FALSE)? Setting this to TRUE allows the surrogate variables to be correlated with the observed covariates, while setting this to FALSE assumes that the surrogate variables are orthogonal to the observed covariates. This option only matters if design_obs is not NULL. Defaults to FALSE.

design_obs A numeric matrix of observed covariates that are NOT to be a part of the signal generating process. Only used in estimating the surrogate variables (if target_cor is not NULL). The intercept should *not* be included (it will sometimes produce an error if it is included).

relative A logical. Should we apply relative thinning (TRUE) or absolute thinning (FALSE). Only experts should change the default.

change_colnames A logical. Should we change the column-names of the design matrices (TRUE) or not (FALSE)? Each new column name begins with either "O" (observed), "P" (permuted), or "F" (fixed), followed by a number. The letters correspond to whether the variables come from design_obs, design_perm, or design_fixed. Setting this to TRUE also changes the column-names of the corresponding coefficient matrices. Defaults to TRUE.

permute_method Should we use the optimal matching technique from Hansen and Klopfer (2006) ("optmatch"), the Gale-Shapley algorithm for stable marriages ("marriage") (Gale and Shapley, 1962) as implemented in the matchingR package, or the Hungarian algorithm (Papadimitriou and Steiglitz, 1982) ("hungarian") as implemented in the clue package (Hornik, 2005)? The "optmatch" method works really well but does take a lot more computational time if you have, say, 1000 samples. If you use the "optmatch" option, you should note that the optmatch package uses a super strange license: https://cran.r-project.org/package=optmatch/LICENSE. If this license doesn't work for you (because you are not in academia, or because you don't believe in restrictive licenses), then try out the "hungarian" method.

type Should we apply binomial thinning (type = "thin") or just naive multiplication of the counts (type = "mult"). You should always have this set to "thin".

Value

A list-like S3 object of class ThinData. Components include some or all of the following:

mat The modified matrix of counts.

`designmat` The design matrix of variables used to simulate signal. This is made by column-binding `design_fixed` and the permuted version of `design_perm`.

`coefmat` A matrix of coefficients corresponding to `designmat`.

`design_obs` Additional variables that should be included in your design matrix in downstream fittings. This is made by column-binding the vector of 1's with `design_obs`.

`sv` A matrix of estimated surrogate variables. In simulation studies you would probably leave this out and estimate your own surrogate variables.

`cormat` A matrix of target correlations between the surrogate variables and the permuted variables in the design matrix. This might be different from the `target_cor` you input because we pass it through `fix_cor` to ensure positive semi-definiteness of the resulting covariance matrix.

`matching_var` A matrix of simulated variables used to permute `design_perm` if the `target_cor` is not `NULL`.

Mathematical Formulation

Let

N Be the number of samples.

G Be the number of genes.

Y Be an G by N matrix of real RNA-seq counts. This is `mat`.

X_1 Be an N by P_1 user-provided design matrix. This is `design_fixed`.

X_2 Be an N by P_2 user-provided design matrix. This is `design_perm`.

X_3 Be an N by P_3 matrix of known covariates. This is `design_obs`.

Z Be an N by K matrix of unobserved surrogate variables. This is estimated when `target_cor` is not `NULL`.

M Be a G by N of additional (unknown) unwanted variation.

We assume that Y is Poisson distributed given X_3 and Z such that

$$\log_2(EY) = \mu 1_N' + B_3 X_3' + AZ' + M.$$

`thin_diff()` will take as input X_1, X_2, B_1, B_2, and will output a $\tilde{Y}$ and W such that $\tilde{Y}$ is Poisson distributed given X_1, X_2, X_3, W, Z, and M such that

$$\log_2(E\tilde{Y}) \approx \tilde{\mu} 1_N' + B_1 X_1' + B_2 X_2' W' + B_3 X_3' + AZ' + M,$$

where W is an N by N permutation matrix. W is randomly drawn so that WX_2 and Z are correlated approximately according to the target correlation matrix.

The Poisson assumption may be generalized to a mixture of negative binomials.

Unestimable Components

It is possible to include an intercept term or a column from `design_obs` into either `design_fixed` or `design_perm`. This will not produce an error and the specified thinning will be applied. However, If any column of `design_fixed` or `design_perm` is a vector of ones or contains a column from `design_obs`, then the corresponding columns in `coef_fixed` or `coef_perm` cannot be estimated

by *any* method. This is represented in the output by having duplicate columns in `designmat` and `design_obs`.

Including duplicate columns in `design_fixed` and `design_perm` is also allowed but, again, will produce unestimable coefficients.

Including an intercept term in `design_obs` will produce an error if you are specifying correlated surrogate variables.

Author(s)

David Gerard

References

- Gale, David, and Lloyd S. Shapley. "College admissions and the stability of marriage." The American Mathematical Monthly 69, no. 1 (1962): 9-15.
- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.
- Hansen, Ben B., and Stephanie Olsen Klopfer. "Optimal full matching and related designs via network flows." Journal of computational and Graphical Statistics 15, no. 3 (2006): 609-627.
- Hornik K (2005). "A CLUE for CLUster Ensembles." Journal of Statistical Software, 14(12). doi: 10.18637/jss.v014.i12
- Leek, Jeffrey T., and John D. Storey. "A general framework for multiple testing dependence." Proceedings of the National Academy of Sciences 105, no. 48 (2008): 18718-18723.
- C. Papadimitriou and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity. Englewood Cliffs: Prentice Hall.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_2group` For the specific application of `thin_diff` to the two-group model.

`thin_lib` For the specific application of `thin_diff` to library size thinning.

`thin_gene` For the specific application of `thin_diff` to total gene expression thinning.

`thin_all` For the specific application of `thin_diff` to thinning all counts uniformly.

`thin_base` For the underlying thinning function used in `thin_diff`.

`sva` For the implementation of surrogate variable analysis.

`ThinDataToSummarizedExperiment` For converting a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` For converting a ThinData object to a DESeqDataSet object.

Examples

```
## Generate simulated data with surrogate variables
## In practice, you would obtain mat from a real dataset, not simulate it.
set.seed(1)
n <- 10
p <- 1000
Z <- matrix(abs(rnorm(n, sd = 4)))
alpha <- matrix(abs(rnorm(p, sd = 1)))
mat <- round(2^(alpha %*% t(Z) + abs(matrix(rnorm(n * p, sd = 5),
                                            nrow = p,
                                            ncol = n))))

## Choose simulation parameters
design_perm <- cbind(rep(c(0, 1), length.out = n), runif(n))
coef_perm <- matrix(rnorm(p * ncol(design_perm), sd = 6), nrow = p)

## Specify one surrogate variable (number of columns in taget_cor),
## highly correlated with first observed covariate and uncorrelated
## with second observed covariate
target_cor <- matrix(c(0.9, 0))

## Thin
thout <- thin_diff(mat = mat,
                   design_perm = design_perm,
                   coef_perm = coef_perm,
                   target_cor = target_cor)

## target_cor approximates correlation between estimated surrogate variable
## and matching variable.
cor(thout$matching_var, thout$sv)

## Estimated surrogate variable is associated with true surrogate variable
## (because the signal is strong in this case)
plot(Z, thout$sv, xlab = "True SV", ylab = "Estimated SV")

## So target_cor approximates correlation between surrogate variable and
## matching variables
cor(thout$matching_var, Z)

## Correlation between permuted covariates and surrogate variables are less
## close to target_cor
cor(thout$designmat, Z)

## Estimated signal is correlated to true single. First variable is slightly
## biased because the surrogate variable is not included.
Ynew <- log2(t(thout$mat) + 0.5)
X <- thout$designmat
coef_est <- t(coef(lm(Ynew ~ X))[2:3, ])

plot(thout$coefmat[, 1], coef_est[, 1],
     main = "First Variable",
     xlab = "Coefficient",
```

```
     ylab = "Estimated Coefficient")
abline(0, 1, col = 2, lwd = 2)

plot(thout$coefmat[, 2], coef_est[, 2],
     main = "Second Variable",
     xlab = "Coefficient",
     ylab = "Estimated Coefficient")
abline(0, 1, col = 2, lwd = 2)

## But estimated coefficient of the first variable is slightly closer when
## the surrogate variable is included.
Ynew <- log2(t(thout$mat) + 0.5)
X <- cbind(thout$designmat, thout$sv)
coef_est <- t(coef(lm(Ynew ~ X))[2:3, ])

plot(thout$coefmat[, 1], coef_est[, 1],
     main = "First Variable",
     xlab = "Coefficient",
     ylab = "Estimated Coefficient")
abline(0, 1, col = 2, lwd = 2)

plot(thout$coefmat[, 2], coef_est[, 2],
     main = "Second Variable",
     xlab = "Coefficient",
     ylab = "Estimated Coefficient")
abline(0, 1, col = 2, lwd = 2)
```

`thin_gene` *Binomial thinning for altering total gene expression levels*

Description

Given a matrix of real RNA-seq counts, this function will apply a separate, user-provided thinning factor to each gene. This uniformly lowers the counts for all samples in a gene. The thinning factor should be provided on the log2-scale. This is a specific application of the binomial thinning approach in `thin_diff`. The method is described in detail in Gerard (2020).

Usage

```
thin_gene(mat, thinlog2, relative = FALSE, type = c("thin", "mult"))
```

Arguments

`mat`	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
`thinlog2`	A vector of numerics. Element i is the amount to thin (on the log2 scale) for gene i. For example, a value of 0 means that we do not thin, a value of 1 means that we thin by a factor of 2, a value of 2 means we thin by a factor of 4, etc.

`relative` A logical. Should we apply relative thinning (`TRUE`) or absolute thinning (`FALSE`). Only experts should change the default.

`type` Should we apply binomial thinning (`type = "thin"`) or just naive multiplication of the counts (`type = "mult"`). You should always have this set to `"thin"`.

Value

A list-like S3 object of class `ThinData`. Components include some or all of the following:

`mat` The modified matrix of counts.

`designmat` The design matrix of variables used to simulate signal. This is made by column-binding `design_fixed` and the permuted version of `design_perm`.

`coefmat` A matrix of coefficients corresponding to `designmat`.

`design_obs` Additional variables that should be included in your design matrix in downstream fittings. This is made by column-binding the vector of 1's with `design_obs`.

`sv` A matrix of estimated surrogate variables. In simulation studies you would probably leave this out and estimate your own surrogate variables.

`cormat` A matrix of target correlations between the surrogate variables and the permuted variables in the design matrix. This might be different from the `target_cor` you input because we pass it through `fix_cor` to ensure positive semi-definiteness of the resulting covariance matrix.

`matching_var` A matrix of simulated variables used to permute `design_perm` if the `target_cor` is not `NULL`.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_diff` For the more general thinning approach.

`thin_lib` For thinning sample-wise instead of gene-wise.

`thin_all` For thinning all counts uniformly.

`ThinDataToSummarizedExperiment` For converting a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` For converting a ThinData object to a DESeqDataSet object.

Examples

```
## Generate count data and thinning factors
## In practice, you would obtain mat from a real dataset, not simulate it.
set.seed(1)
n <- 10
p <- 1000
lambda <- 1000
mat <- matrix(lambda, ncol = n, nrow = p)
thinlog2 <- rexp(n = p, rate = 1)

## Thin total gene expressions
thout <- thin_gene(mat = mat, thinlog2 = thinlog2)

## Compare empirical thinning proportions to specified thinning proportions
empirical_propvec <- rowMeans(thout$mat) / lambda
specified_propvec <- 2 ^ (-thinlog2)
plot(empirical_propvec, specified_propvec,
     xlab = "Empirical Thinning Proportion",
     ylab = "Specified Thinning Proportion")
abline(0, 1, col = 2, lwd = 2)
```

---

thin_lib *Binomial thinning for altering library size.*

Description

Given a matrix of real RNA-seq counts, this function will apply a separate, user-provided thinning factor to each sample. This uniformly lowers the counts for all genes in a sample. The thinning factor should be provided on the log2-scale. This is a specific application of the binomial thinning approach in `thin_diff`. The method is described in detail in Gerard (2020).

Usage

```
thin_lib(mat, thinlog2, relative = FALSE, type = c("thin", "mult"))
```

Arguments

`mat`	A numeric matrix of RNA-seq counts. The rows index the genes and the columns index the samples.
`thinlog2`	A vector of numerics. Element i is the amount to thin (on the log2-scale) for sample i. For example, a value of 0 means that we do not thin, a value of 1 means that we thin by a factor of 2, a value of 2 means we thin by a factor of 4, etc.
`relative`	A logical. Should we apply relative thinning (`TRUE`) or absolute thinning (`FALSE`). Only experts should change the default.
`type`	Should we apply binomial thinning (`type = "thin"`) or just naive multiplication of the counts (`type = "mult"`). You should always have this set to `"thin"`.

Value

A list-like S3 object of class `ThinData`. Components include some or all of the following:

`mat` The modified matrix of counts.

`designmat` The design matrix of variables used to simulate signal. This is made by column-binding `design_fixed` and the permuted version of `design_perm`.

`coefmat` A matrix of coefficients corresponding to `designmat`.

`design_obs` Additional variables that should be included in your design matrix in downstream fittings. This is made by column-binding the vector of 1's with `design_obs`.

`sv` A matrix of estimated surrogate variables. In simulation studies you would probably leave this out and estimate your own surrogate variables.

`cormat` A matrix of target correlations between the surrogate variables and the permuted variables in the design matrix. This might be different from the `target_cor` you input because we pass it through `fix_cor` to ensure positive semi-definiteness of the resulting covariance matrix.

`matching_var` A matrix of simulated variables used to permute `design_perm` if the `target_cor` is not `NULL`.

Author(s)

David Gerard

References

- Gerard, D (2020). "Data-based RNA-seq simulations by binomial thinning." *BMC Bioinformatics*. 21(1), 206. doi: 10.1186/s12859-020-3450-9.

See Also

`select_counts` For subsampling the rows and columns of your real RNA-seq count matrix prior to applying binomial thinning.

`thin_diff` For the more general thinning approach.

`thin_gene` For thinning gene-wise instead of sample-wise.

`thin_all` For thinning all counts uniformly.

`ThinDataToSummarizedExperiment` For converting a ThinData object to a SummarizedExperiment object.

`ThinDataToDESeqDataSet` For converting a ThinData object to a DESeqDataSet object.

Examples

```
## Generate count data and thinning factors
## In practice, you would obtain mat from a real dataset, not simulate it.
set.seed(1)
n <- 10
p <- 1000
lambda <- 1000
mat <- matrix(lambda, ncol = n, nrow = p)
```

```
thinlog2 <- rexp(n = n, rate = 1)

## Thin library sizes
thout <- thin_lib(mat = mat, thinlog2 = thinlog2)

## Compare empirical thinning proportions to specified thinning proportions
empirical_propvec <- colMeans(thout$mat) / lambda
specified_propvec <- 2 ^ (-thinlog2)
empirical_propvec
specified_propvec
```

`uncorassign` *Group assignment independent of anything.*

Description

Group assignment independent of anything.

Usage

```
uncorassign(n, return = c("group", "full"))
```

Arguments

`n`	The sample size.
`return`	Should we just return a list with just the vector of assignment (`"group"`) or a list with the vector of assignments and the vector of latent variables (`"full"`)?

Value

A list with some or all of the following elements.

`x` The group assignment. `1L` for one group and `0L` for the other group.

`w` The latent assignment vector (only returned if `return = "full"`). Negative corresponds to one group and positive corresponds to the other group.

Author(s)

David Gerard

Index