Package ‘pareg’

February 3, 2024

Title  Pathway enrichment using a regularized regression approach

Version  1.6.0

Description  Compute pathway enrichment scores while accounting for term-term relations. This package uses a regularized multiple linear regression to regress differential expression p-values obtained from multi-condition experiments on a pathway membership matrix. By doing so, it is able to incorporate additional biological knowledge into the enrichment analysis and to estimate pathway enrichment scores more robustly.

URL  https://github.com/cbg-ethz/pareg

BugReports  https://github.com/cbg-ethz/pareg/issues

biocViews  Software, StatisticalMethod, GraphAndNetwork, Regression, GeneExpression, DifferentialExpression, NetworkEnrichment, Network

License  GPL-3

Encoding  UTF-8

LazyData  false

Depends  R (>= 4.2), tensorflow (>= 2.2.0), tfprobability (>= 0.10.0)

Suggests  knitr, rmarkdown, testthat (>= 2.1.0), BiocStyle, formatR, plotROC, PRROC, mgsa, topGO, msigdbr, betareg, fgsea, ComplexHeatmap, GAllay, ggSignif, circlize, enrichplot, gnewscale, tidyverse, cowplot, ggfittext, simplifyEnrichment, GSEABenchmarkR, BiocParallel, ggupset, latex2exp, org.Hs.eg.db, GO.db

VignetteBuilder  knitr

RoxygenNote  7.2.3

Imports  stats, tidyr, purrr, future, doFuture, foreach, doRNG, tibble, glue, tidygraph, igraph, proxy, dplyr, magrittr, ggplot2, ggraph, rlang, progress, Matrix, keras, nloptr, ggrepel, methods, DOSE, stringr, reticulate, logger, hms, devtools, basilisk

StagedInstall  no

git_url  https://git.bioconductor.org/packages/pareg
**Description**

Retrieve dataframe with enrichment information.

**Usage**

```r
# S3 method for class 'pareg'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```
Arguments

x An object of class pareg.
row.names Optional character vector of rownames.
optional Allow optional arguments.
... Additional arguments.

Value

Dataframe containing enrichment score and name for each pathway.

Examples

df_genes <- data.frame(
  gene = paste("g", 1:20, sep = ""),
  pvalue = c(
    rbeta(10, .1, 1),
    rbeta(10, 1, 1)
  )
)
df_terms <- rbind(
  data.frame(
    term = "foo",
    gene = paste("g", 1:10, sep = "")
  ),
  data.frame(
    term = "bar",
    gene = paste("g", 11:20, sep = "")
  )
)
fit <- pareg(df_genes, df_terms, max_iterations = 10)
as.data.frame(fit)

Description

Convert sparse similarity matrix from package data to a dense version with 1 on its diagonal. This matrix can then be used by pareg.

Usage

as_dense_sim(mat_sparse)

Arguments

mat_sparse Sparse matrix.
Value

Dense matrix

Examples

transform_y(c(0, 0.5, 1))

---

as_enrichplot_object  

Convert object of class pareg to class enrichResult.

Description

The resulting object can be passed to any method from the enrichplot package and thus allows for nice visualizations of the enrichment results. Note: term similarities are included if available.

Usage

as_enrichplot_object(x, pvalue_threshold = 0.05)

Arguments

x  
An object of class pareg.

pvalue_threshold  
Threshold to select genes for count statistics.

Value

Object of class enrichResult.

Examples

df_genes <- data.frame(  
gene = paste(“g”, 1:20, sep = “”),  
pvalue = c(  
  rbeta(10, .1, 1),  
  rbeta(10, 1, 1)  
)
)

df_terms <- rbind(  
data.frame(  
  term = “foo”,  
  gene = paste(“g”, 1:10, sep = “”)  
),  
data.frame(  
  term = “bar”,  
  gene = paste(“g”, 11:20, sep = “”)  
)
)

fit <- pareg(df_genes, df_terms, max_iterations = 10)  

as_enrichplot_object(fit)
**cluster_apply**  

Parallelize function calls on LSF cluster.

**Description**

Run function for each row of input dataframe in LSF job.

**Usage**

```r
cluster_apply(
  df_iter,  
  func,  
  .bsub_params = c("-n", "2", "-W", "24:00", "-R", "rusage[mem=10000]")
  , .tempdir = ",",  
  .packages = c(),  
  ...  
)
```

**Arguments**

- **df_iter** Dataframe over whose rows to iterate.
- **func** Function to apply to each dataframe row. Its arguments must be all dataframe columns.
- **.bsub_params** Parameters to pass to `bsub` during job submission.
- **.tempdir** Location to store auxiliary files in.
- **.packages** Packages to import in each job.
- **...** Extra arguments for function.

**Value**

Dataframe created by concatenating results of each function call.

**Examples**

```r
## Not run:
foo <- 42
cluster_apply(
  data.frame(i = seq_len(3), group = c("A", "B", "C")),
  function(i, group) {
    log_debug("hello")
    data.frame(group = group, i = i, foo = foo, result = foo + 2 * i)
  },
  .packages = c(logger)
)
## End(Not run)
```
compute_term_similarities

Term similarity computation.

Description
Generate similarity matrix for input terms.

Usage
compute_term_similarities(
  df_terms,
  similarity_function = jaccard,
  max_similarity = 1
)

Arguments
  df_terms Dataframe storing pathway database.
  similarity_function Function to compute similarity between two sets.
  max_similarity Value to fill diagonal with.

Value
Symmetric matrix of similarity scores.

Examples
  df_terms <- data.frame(
    gene = c("a", "b", "a", "b", "c", "a", "c", "d")
  )
  compute_term_similarities(df_terms)

create_model_df Create design matrix.

Description
Store term membership for each gene.

Usage
create_model_df(df_genes, df_terms, pvalue_threshold = 0.05)
**Arguments**

- `df_genes`: Dataframe storing gene names and DE p-values.
- `df_terms`: Dataframe storing pathway database.
- `pvalue_threshold`: P-value threshold to create binary columns `pvalue_sig` and `pvalue_notsig`.

**Value**

Dataframe.

**Examples**

```r
df_genes <- data.frame(
  gene = c("g1", "g2"),
  pvalue = c(0.1, 0.2)
)
df_terms <- data.frame(
  term = c("A", "B", "B", "C"),
  gene = c("g1", "g2", "g1", "g2")
)
create_model_df(df_genes, df_terms)
```

---

**cv_edgenet**

*Find the optimal shrinkage parameters for edgenet*

**Description**

Finds the optimal regularization parameters using cross-validation for edgenet. We use the BOBYQA algorithm to find the optimal regularization parameters in a cross-validation framework.

**Usage**

```r
cv_edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = NA_real_,
  psigx = NA_real_,
  psigy = NA_real_,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian,
  optim.thresh = 0.01,
  optim.maxit = 100,
  lambda_range = seq(0, 2, length.out = 10),
)```

cv_edgenet

psigx_range = seq(0, 500, length.out = 10),
psigy_range = seq(0, 500, length.out = 10),
nfolds = 2,
cv_method = c("grid_search", "grid_search_lsf", "optim"),
tempdir = "."
)

## S4 method for signature 'matrix,numeric'
cv_edgenet(
X,
Y,
G.X = NULL,
G.Y = NULL,
lambda = NA_real_,
psigx = NA_real_,
psigy = NA_real_,
thresh = 1e-05,
maxit = 1e+05,
learning.rate = 0.01,
family = gaussian,
optim.thresh = 0.01,
optim.maxit = 100,
lambda_range = seq(0, 2, length.out = 10),
psigx_range = seq(0, 500, length.out = 10),
psigy_range = seq(0, 500, length.out = 10),
nfolds = 2,
cv_method = c("grid_search", "grid_search_lsf", "optim"),
tempdir = "."
)

## S4 method for signature 'matrix,matrix'
cv_edgenet(
X,
Y,
G.X = NULL,
G.Y = NULL,
lambda = NA_real_,
psigx = NA_real_,
psigy = NA_real_,
thresh = 1e-05,
maxit = 1e+05,
learning.rate = 0.01,
family = gaussian,
optim.thresh = 0.01,
optim.maxit = 100,
lambda_range = seq(0, 2, length.out = 10),
psigx_range = seq(0, 500, length.out = 10),
psigy_range = seq(0, 500, length.out = 10),
cv_edgenet

```
nfolds = 2,
cv_method = c("grid_search", "grid_search_lsf", "optim"),
tempdir = ")."
```

**Arguments**

- **X**
  - input matrix, of dimension \((n \times p)\) where \(n\) is the number of observations and \(p\) is the number of covariables. Each row is an observation vector.

- **Y**
  - output matrix, of dimension \((n \times q)\) where \(n\) is the number of observations and \(q\) is the number of response variables. Each row is an observation vector.

- **G.X**
  - non-negativ affinity matrix for \(X\), of dimensions \((p \times p)\) where \(p\) is the number of covariables. Providing a graph \(G.X\) will optimize the regularization parameter \(\psi.gx\). If this is not desired just set \(G.X\) to NULL.

- **G.Y**
  - non-negativ affinity matrix for \(Y\), of dimensions \((q \times q)\) where \(q\) is the number of responses \(Y\). Providing a graph \(G.Y\) will optimize the regularization parameter \(\psi.gy\). If this is not desired just set \(G.Y\) to NULL.

- **lambda**
  - numerical shrinkage parameter for LASSO. Per default this parameter is set to \(NA._real_\) which means that \(lambda\) is going to be estimated using cross-validation. If any numerical value for \(lambda\) is set, estimation of the optimal parameter will not be conducted.

- **psigx**
  - numerical shrinkage parameter for graph-regularization of \(G.X\). Per default this parameter is set to \(NA._real_\) which means that \(psigx\) is going to be estimated in the cross-validation. If any numerical value for \(psigx\) is set, estimation of the optimal parameter will not be conducted.

- **psigy**
  - numerical shrinkage parameter for graph-regularization of \(G.Y\). Per default this parameter is set to \(NA._real_\) which means that \(psigy\) is going to be estimated in the cross-validation. If any numerical value for \(psigy\) is set, estimation of the optimal parameter will not be conducted.

- **thresh**
  - numerical threshold for the optimizer

- **maxit**
  - maximum number of iterations for the optimizer \((\text{integer})\)

- **learning.rate**
  - step size for Adam optimizer \((\text{numerical})\)

- **family**
  - family of response, e.g. \(\text{gaussian}\) or \(\text{binomial}\)

- **optim.thresh**
  - numerical threshold criterion for the optimization to stop. Usually \(1e^{-3}\) is a good choice.

- **optim.maxit**
  - the maximum number of iterations for the optimization \((\text{integer})\). Usually \(1e^{4}\) is a good choice.

- **lambda_range**
  - range of lambda to use in CV grid.

- **psigx_range**
  - range of psigx to use in CV grid.

- **psigy_range**
  - range of psigy to use in CV grid.

- **nfolds**
  - the number of folds to be used - default is 10.

- **cv_method**
  - which cross-validation method to use.

- **tempdir**
  - where to store auxiliary files.
Value

An object of class `cv_edgenet`

parameters  the estimated, optimal regularization parameters
lambda  optimal estimated value for regularization parameter lambda (or, if provided as argument, the value of the parameter)
psigx  optimal estimated value for regularization parameter psigx (or, if provided as argument, the value of the parameter)
psigy  optimal estimated value for regularization parameter psigy (or, if provided as argument, the value of the parameter)
estimated.parameters  names of parameters that were estimated
family  family used for estimated
fit  an edgenet object fitted with the optimal, estimated parameters
call  the call that produced the object

Examples

```r
X <- matrix(rnorm(100 * 10), 100, 10)
b <- matrix(rnorm(100), 10)
G.X <- abs(rWishart(1, 10, diag(10))[, , 1])
G.Y <- abs(rWishart(1, 10, diag(10))[, , 1])
diag(G.X) <- diag(G.Y) <- 0

# estimate the parameters of a Gaussian model
Y <- X %*% b + matrix(rnorm(100 * 10), 100)

## dont use affinity matrices and estimate lambda
fit <- cv_edgenet(
  X = X,
  Y = Y,
  family = gaussian,
  maxit = 1,
  lambda_range = c(0, 1)
)

## only provide one matrix and estimate lambda
fit <- cv_edgenet(
  X = X,
  Y = Y,
  G.X = G.X,
  psigx = 1,
  family = gaussian,
  maxit = 1,
  lambda_range = c(0, 1)
)

## estimate only lambda with two matrices
fit <- cv_edgenet(
  X = X,
  Y = Y,
  G.Y = G.Y,
  psigy = 1,
  family = gaussian,
  maxit = 1,
  lambda_range = c(0, 1)
)
```

edgenet

G.X = G.X,
G.Y,
psigx = 1,
psigy = 1,
family = gaussian,
maxit = 1,
lambda_range = c(0, 1)
)
## estimate only psigx
fit <- cv_edgenet(
  X = X,
  Y = Y,
  G.X = G.X,
  G.Y,
  lambda = 1,
  psigy = 1,
  family = gaussian,
  maxit = 1,
  psigx_range = c(0, 1)
)
## estimate all parameters
fit <- cv_edgenet(
  X = X,
  Y = Y,
  G.X = G.X,
  G.Y,
  family = gaussian,
  maxit = 1,
  lambda_range = c(0, 1),
  psigy_range = c(0, 1),
  psigx_range = c(0, 1)
)
## if Y is vectorial, we cannot use an affinity matrix for Y
fit <- cv_edgenet(
  X = X,
  Y = Y[, 1],
  G.X = G.X,
  family = gaussian,
  maxit = 1,
  lambda_range = c(0, 1),
  psigx_range = c(0, 1),
  psigy_range = c(0, 1)
)

edgenet

Fit a graph-regularized linear regression model using edge-based regularization. Adapted from https://github.com/dirmeier/netReg.

Description

Fit a graph-regularized linear regression model using edge-penalization. The coefficients are computed using graph-prior knowledge in the form of one/two affinity matrices. Graph-regularization
is an extension to previously introduced regularization techniques, such as the LASSO. See the vignette for details on the objective function of the model: vignette("edgenet", package="netReg")

Usage

edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 0,
  psigx = 0,
  psigy = 0,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,numeric'
edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 0,
  psigx = 0,
  psigy = 0,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)

## S4 method for signature 'matrix,matrix'
edgenet(
  X,
  Y,
  G.X = NULL,
  G.Y = NULL,
  lambda = 0,
  psigx = 0,
  psigy = 0,
  thresh = 1e-05,
  maxit = 1e+05,
  learning.rate = 0.01,
  family = gaussian
)
Arguments

\( X \)
input matrix, of dimension \((n \times p)\) where \(n\) is the number of observations and \(p\) is the number of covariables. Each row is an observation vector.

\( Y \)
output matrix, of dimension \((n \times q)\) where \(n\) is the number of observations and \(q\) is the number of response variables. Each row is an observation vector.

\( G.X \)
non-negativ affinity matrix for \(X\), of dimensions \((p \times p)\) where \(p\) is the number of covariables

\( G.Y \)
non-negativ affinity matrix for \(Y\), of dimensions \((q \times q)\) where \(q\) is the number of responses

\( \lambda \)
numerical shrinkage parameter for LASSO.

\( psigx \)
numerical shrinkage parameter for graph-regularization of \(G.X\)

\( psigy \)
numerical shrinkage parameter for graph-regularization of \(G.Y\)

\( \text{thresh} \)
numerical threshold for optimizer

\( \text{maxit} \)
maximum number of iterations for optimizer (integer)

\( \text{learning.rate} \)
step size for Adam optimizer (numerical)

\( \text{family} \)
family of response, e.g. \text{gaussian} or \text{binomial}

Value

An object of class edgenet

\( \text{beta} \)
the estimated \((p \times q)\)-dimensional coefficient matrix \( \hat{B} \)

\( \text{alpha} \)
the estimated \((q \times 1)\)-dimensional vector of intercepts

\( \text{parameters} \)
regularization parameters

\( \text{lambda} \)
regularization parameter \( \lambda \)

\( \text{psigx} \)
regularization parameter \( psigx \)

\( \text{psigy} \)
regularization parameter \( psigy \)

\( \text{family} \)
a description of the error distribution and link function to be used. Can be a \text{pareg::family} function or a character string naming a family function, e.g. \text{gaussian} or "gaussian".

\( \text{call} \)
the call that produced the object

References

Cheng, Wei and Zhang, Xiang and Guo, Zhishan and Shi, Yu and Wang, Wei (2014), Graph-regularized dual Lasso for robust eQTL mapping.

\textit{Bioinformatics}
family

Family objects for models

Description

Family objects provide a convenient way to specify the details of the models used by pareg. See also stats::family for more details.

Usage

family(object, ...)

 gaussian(link = c("identity"))

 bernoulli(link = c("logit", "probit", "log"))

 beta(link = c("logit", "probit", "log"))

 beta_phi_lm(link = c("logit", "probit", "log"))

 beta_phi_var(link = c("logit", "probit", "log"))
generate_similarity_matrix

Similarity matrix generation.

**Arguments**

- **object**: a object for which the family should be returned (e.g. edgenet)
- **...**: further arguments passed to methods
- **link**: name of a link function

**Value**

An object of class `pareg.family`

- **family**: name of the family
- **link**: name of the link function
- **linkinv**: inverse link function
- **loss**: loss function

**Examples**

```
  gaussian()
  bernoulli("probit")$link
  beta()$loss
```

```r
generate_similarity_matrix(c(1, 2, 3))
```

**Description**

Generate block-structured similarity matrices corresponding to cluster structures.

**Usage**

```
generate_similarity_matrix(cluster_sizes)
```

**Arguments**

- **cluster_sizes**: List of cluster sizes.

**Value**

Similarity matrix with samples as row-/colnames.

**Examples**

```
generate_similarity_matrix(c(1, 2, 3))
```
**jaccard**  
*Jaccard similarity.*

**Description**  
Compute Jaccard similarity between two sets.

**Usage**  
`jaccard(x, y)`

**Arguments**
- `x`  
  First set.
- `y`  
  Second set.

**Value**  
Jaccard similarity between set `x` and `y`.

**See Also**  
Other pathway similarity methods: `overlap_coefficient()`

**Examples**
```
jaccard(c(1, 2, 3), c(2, 3, 4))
```

**overlap_coefficient**  
*Overlap coefficient.*

**Description**  
Compute overlap coefficient between two sets.

**Usage**  
`overlap_coefficient(x, y)`

**Arguments**
- `x`  
  First set.
- `y`  
  Second set.
Value

Overlap coefficient between set x and y.

See Also

Other pathway similarity methods: jaccard()

Examples

overlap_coefficient(c(1, 2, 3), c(2, 3, 4))

Description

Run model to compute pathway enrichments. Can model inter-pathway relations, cross-validation and much more.

Usage

pareg(
  df_genes,
  df_terms,
  lasso_param = NA_real_,
  network_param = NA_real_,
  term_network = NULL,
  cv = FALSE,
  cv_cores = NULL,
  family = beta,
  response_column_name = "pvalue",
  max_iterations = 1e+05,
  lasso_param_range = seq(0, 2, length.out = 10),
  network_param_range = seq(0, 500, length.out = 10),
  log_level = NULL,
  ...
)

Arguments

df_genes      Dataframe storing gene names and DE p-values.
df_terms      Dataframe storing pathway database.
lasso_param   Lasso regularization parameter.
network_param Network regularization parameter.
term_network  Term similarity network as adjacency matrix.
cv            Estimate best regularization parameters using cross-validation.
cv_cores  How many cores to use for CV parallelization.
family  Distribution family of response.
response_column_name  Which column of model dataframe to use as response.
max_iterations  How many iterations to maximally run optimizer for.
lasso_param_range  LASSO regularization parameter search space in grid search of CV.
network_param_range  Network regularization parameter search space in grid search of CV.
log_level  Control verbosity (logger::INFO, logger::DEBUG, ...).
...  Further arguments to pass to `(cv.)edgenet`.

Value
An object of class `pareg`.

Examples
```r
df_genes <- data.frame(
  gene = paste("g", 1:20, sep = ""),
  pvalue = c(
    rbeta(10, .1, 1),
    rbeta(10, 1, 1)
  )
)
df_terms <- rbind(
  data.frame(
    term = "foo",
    gene = paste("g", 1:10, sep = "")
  ),
  data.frame(
    term = "bar",
    gene = paste("g", 11:20, sep = "")
  )
)
pareg(df_genes, df_terms, max_iterations = 10)
```

Description
Declare Python packages needed to run this R package.

Usage

```r
pareg_env
```
**pathway_similarities**

**Format**

An object of class BasiliskEnvironment of length 1.

---

pathway_similarities  
*Collection of pathway similarity matrices.*

**Description**

Contains matrices for various pathway databases and similarity measures. Note that the matrices are sparse, upper triangular and subsampled to a maximum size of $1000x1000$ if necessary. They can be transformed to a dense representation using pareg::as_dense_sim.

**Usage**

pathway_similarities

**Format**

A list of lists of matrices. * Pathway database 1 * Similarity measure 1 * Similarity measure 2 * ...  
* Pathway database 2 * ...

---

plot.pareg  
*Plot pareg object.*

**Description**

Check pareg::plot_pareg_with_args for details. Needed because of WARNING in 'checking S3 generic/method consistency''

**Usage**

```r
## S3 method for class 'pareg'
plot(x, ...)
```

**Arguments**

- `x`  
  An object of class pareg.

- `...`  
  Parameters passed to pareg::plot_pareg_with_args

**Value**

ggplot object.
plot_pareg_with_args  

Plot result of enrichment computation.

Description

Visualize pathway enrichments as network.

Usage

plot_pareg_with_args(
  x,
  show_term_names = TRUE,
  min_similarity = 0,
  term_subset = NULL
)

Arguments

x  An object of class pareg.
show_term_names  Whether to plot node labels.
min_similarity  Don’t plot edges for similarities below this value.
term_subset  Subset of terms to show.

Value

ggplot object.

Examples

df_genes <- data.frame(
  gene = paste("g", 1:20, sep = ""),
  pvalue = c(
    rbeta(10, .1, 1),
    rbeta(10, 1, 1)
  )
)
df_terms <- rbind(
  data.frame(
    term = "foo",
    gene = paste("g", 1:10, sep = "")
  ),
  data.frame(
    term = "bar",
    gene = paste("g", 11:20, sep = "")
  )
)
fit <- pareg(df_genes, df_terms, max_iterations = 10)
plot(fit)
**similarity_sample**  
*Sample elements based on similarity structure.*

**Description**
Choose similar object more often, depending on `similarity_factor`.

**Usage**
```
similarity_sample(sim_mat, size, similarity_factor = 1)
```

**Arguments**
- **sim_mat**: Similarity matrix with samples as row/col names.
- **size**: How many samples to draw.
- **similarity_factor**: Uniform sampling for 0. Weights mixture of uniform and similarity vector for each draw.

**Value**
Vector of samples.

**Examples**
```
similarity_sample(matrix(runif(100), nrow = 10, ncol = 10), 3)
```

---

**transform_y**  
*Transform vector from [0, 1] to (0, 1).*

**Description**
Make (response) vector conform to Beta assumptions as described in section 2 of the betareg vignette [https://cran.r-project.org/web/packages/betareg/vignettes/betareg.pdf](https://cran.r-project.org/web/packages/betareg/vignettes/betareg.pdf).

**Usage**
```
transform_y(y)
```

**Arguments**
- **y**: Numeric vector in [0, 1]^N

**Value**
Numeric vector in (0, 1)^N
Examples

transform_y(c(0, 0.5, 1))
Index

* datasets
  pareg_env, 18
  pathway_similarities, 19
* pathway similarity methods
  jaccard, 16
  overlap_coefficient, 16

as.data.frame.pareg, 2
as_dense_sim, 3
as_enrichplot_object, 4

beroulli (family), 14
beta (family), 14
beta_phi_lm (family), 14
beta_phi_var (family), 14

cluster_apply, 5
compute_term_similarities, 6
create_model_df, 6
cv.edgenet, 7
cv.edgenet, matrix, matrix-method
  (cv.edgenet), 7
cv.edgenet, matrix, numeric-method
  (cv.edgenet), 7

edgenet, 11
edgenet, matrix, matrix-method (edgenet),
  11
edgenet, matrix, numeric-method
  (edgenet), 11

family, 14

gaussian (family), 14
generate_similarity_matrix, 15

jaccard, 16, 17
overlap_coefficient, 16, 16

pareg, 17

pareg::family, 13
pareg_env, 18
pathway_similarities, 19
plot.pareg, 19
plot_pareg_with_args, 20
similarity_sample, 21
stats::family, 14

transform_y, 21