Package: ghypernet (via r-universe)

October 8, 2024

Type Package

Title Fit and Simulate Generalised Hypergeometric Ensembles of Graphs

Version 1.1.1.1

Date 2022-12-05

URL https://ghyper.net

Description Provides functions for model fitting and selection of generalised hypergeometric ensembles of random graphs (gHypEG). To learn how to use it, check the vignettes for a quick tutorial. Please reference its use as Casiraghi, G., Nanumyan, V. (2019) <doi:10.5281/zenodo.2555300> together with those relevant references from the one listed below. The package is based on the research developed at the Chair of Systems Design, ETH Zurich. Casiraghi, G., Nanumyan, V., Scholtes, I., Schweitzer, F. (2016) <arXiv:1607.02441>. Casiraghi, G., Nanumyan, V., Scholtes, I., Schweitzer, F. (2017) <doi:10.1007/978-3-319-67256-4_11>. Casiraghi, G., (2017) <arXiv:1702.02048> Brandenberger, L., Casiraghi, G., Nanumyan, V., Schweitzer, F. (2019) <doi:10.1145/3341161.3342926> Casiraghi, G. (2019) <doi:10.1007/s41109-019-0241-1>. Casiraghi, G., Nanumyan, V. (2021) <doi:10.1038/s41598-021-92519-y>. Casiraghi, G. (2021) <doi:10.1088/2632-072X/ac0493>.

Depends R (>= 3.0)

License AGPL-3

Imports pbmcapply, plyr, numbers, purrr, extraDistr, dplyr, rlang, reshape2, rootSolve, methods, texreg

Suggests BiasedUrn, igraph, knitr, rmarkdown, ggplot2, ggraph, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Language en-GB

LazyData true

Config/testthat/edition 3

Repository https://gi0na.r-universe.dev

RemoteUrl https://github.com/gi0na/r-ghypernet

RemoteRef HEAD

RemoteSha 42935e73a1235d74dd7f60d21b6c9eed973f3912

Contents

adj2el . . . 3
adj_karate . . . 4
as.ghype . . . 4
bccm . . . 5
BootstrapProperty . . . 6
checkGraphtype . . . 8
coef.nrm . . . 8
compute_xi . . . 9
conf.test . . . 10
contacts.adj . . . 11
cospons_mat . . . 11
coxsnellR2 . . . 12
CreateIgGraphs . . . 12
create_predictors . . . 13
create_predictors.list . . . 14
dt . . . 14
dtcommittee . . . 15
el2adj . . . 15
extract.nrm.cluster . . . 16
FitOmega . . . 16
get_zero_dummy . . . 17
ghype . . . 18
gof.test . . . 20
highschool.multiplex . . . 21
highschool.predictors . . . 21
homophily_stat . . . 22
isNetwork . . . 23
JnBlock . . . 24
linkSignificance . . . 25
logl . . . 26
logLik.ghype . . . 28
loglratio . . . 28
lr.test . . . 29
mat2vec.ix . . . 30
mcfaddenR2 . . . 31
nr.ci . . . 32
nr.significance . . . 32

nrm . . . 33
nrmChoose . . . 35
nrmSelection . . . 36
onlinesim_mat . . . 39
predict.nrm . . . 39
reciprocity_stat . . . 40
regularm . . . 41
residuals.nrm . . . 42
rghype . . . 42
RMSE . . . 43
RMSLE . . . 44
scm . . . 44
sharedPartner_stat . . . 45
summary.nrm . . . 46
summary.nrm_selection . . . 47
vec2mat . . . 47
vertexlabels . . . 48

Index **49**

`adj2el` *Maps adjacency matrix to edgelist*

Description

Maps adjacency matrix to edgelist

Usage

```
adj2el(adj, directed = TRUE)
```

Arguments

`adj`	matrix, the adjacency matrix
`directed`	boolean, is the graph directed?

Value

a dataframe containing the edgelist

Examples

```
data(contacts.adj)
el <- adj2el(contacts.adj)
```

adj_karate *Zachary's Karate Club graph*

Description

Weighted adjacency matrix reporting interactions among 34 nodes.

Usage

```
adj_karate
```

Format

a 34x34 matrix

Source

package 'igraphdata'

as.ghype *Map list to ghype object*

Description

Manually map a list to a ghype object

Usage

```
as.ghype(object, ...)

## S3 method for class 'list'
as.ghype(object, ...)

## S3 method for class 'nrm'
as.ghype(object, ...)
```

Arguments

object	list object to map to ghype.
...	additional arguments to be passed to logl function.

Value

an object of class "ghype"

Methods (by class)

- as.ghype(list): Map list to ghype
- as.ghype(nrm): Map list to ghype

Examples

```
ll <- list(call = NULL, 'adj' = NULL, 'xi'= matrix(36,4,4), 'omega' = matrix(1,4,4),
     'n' = 4, 'm' = 12, 'directed' = TRUE, 'selfloops' = TRUE,
     'regular' = TRUE, 'unbiased' = TRUE, 'df' = 1)
model <- as.ghype(ll)
```

---

bccm *Fitting bccm models*

Description

bccm is used to fit a block-constrained configuration model.

Usage

```
bccm(
  adj,
  labels,
  directed = NULL,
  selfloops = NULL,
  directedBlocks = FALSE,
  homophily = FALSE,
  inBlockOnly = FALSE,
  xi = NULL,
  regular = FALSE,
  ...
)

## S3 method for class 'bccm'
print(x, suppressCall = FALSE, ...)
```

Arguments

adj	the adjacency matrix of the graph.
labels	vector or list. contains the vertex labels to generate the blocks in the bccm. In the case of bipartite graphs should be a list of two vectors, the first one with row labels and the second one with column labels.
directed	a boolean argument specifying whether the graph is directed or not.
selfloops	boolean argument specifying whether the model should incorporate selfloops.

directedBlocks	boolean argument specifying whether the model should incorporate directed blocks. Default to FALSE.
homophily	boolean argument specifying whether the model should fit only homophily blocks. Default to FALSE.
inBlockOnly	boolean argument specifying whether the model should fit only blocks over the diagonal. Default to FALSE.
xi	an optional matrix defining the combinatorial matrix of the model.
regular	optional boolean, fit regular gnp model? if not specified chosen through lr.test.
...	optional arguments to print or plot methods.
x	object of class 'bccm'
suppressCall	logical, indicating whether to print the call that generated x

Value

bccm returns an object of class 'bccm' and 'ghype'. 'bccm' objects expand 'ghype' objects incorporating the parameter estimates.

Methods (by generic)

- print(bccm): Print method for elements of class 'bccm'.

See Also

bccm

Examples

```
data("vertexlabels","adj_karate")
blockmodel <- bccm(adj = adj_karate, labels = vertexlabels, directed = FALSE, selfloops = FALSE)

data('adj_karate')
data('vertexlabels')
bcc.model <- bccm(adj_karate, labels=vertexlabels, directed=FALSE, selfloops=FALSE)
print(bcc.model)
```

BootstrapProperty *BootstrapProperty computes igraph analytics function on ensemble*

Description

BootstrapProperty computes igraph analytics function on ensemble

Usage

```
BootstrapProperty(
  graph,
  property,
  directed,
  selfloops,
  nsamples = 1000,
  xi = NULL,
  omega = NULL,
  model = NULL,
  m = NULL,
  seed = NULL,
  ...
)
```

Arguments

graph	igraph graph
property	igraph function that can be applied to a graph
directed	boolean
selfloops	boolean
nsamples	number of samples from ensemble. defaults to 1000
xi	matrix, default null
omega	matrix, default null
model	ghype model from which to extract xi and omega, default to null
m	int, number of edges to sample from model
seed	seed
...	other parameters to pass to ‘property‘

Value

vector of length nsamples

Examples

```
library(igraph)
data('adj_karate')
result <- BootstrapProperty(adj_karate, page_rank, FALSE, FALSE, nsamples=10)
```

checkGraphtype *Check graph input type (for whether it's a graph or a edgelist).*

Description

Returns TRUE if the supplied object `graph` is an adjacency matrix. Returns FALSE if the provided object is an edgelist. The function checks whether the edgelist conforms to our standards (sender, target, edgecount).

Usage

```
checkGraphtype(graph)
```

Arguments

`graph`	A graph adjacency matrix or an edgelist.

Value

TRUE or FALSE. Returns TRUE if the provided object `graph` is an adjacency matrix.

coef.nrm *Extraction method for coefficients of models of class 'nrm'.*

Description

Extraction method for coefficients of models of class `'nrm'`.

Usage

```
## S3 method for class 'nrm'
coef(object, ...)
```

Arguments

`object`	object of class `'nrm'`.
`...`	optional arguments to print methods.

Value

coefficients of nrm model.

Author(s)

Giona Casiraghi

See Also

nrm

---

compute_xi *Auxiliary function. Computes combinatorial matrix.*

Description

Combinatorial matrix computed according to soft configuration model or 'regular' gnp model.

Usage

```
compute_xi(adj, directed, selfloops, regular = FALSE)

ComputeXi(adj, directed, selfloops, regular = FALSE)
```

Arguments

adj	adjacency matrix
directed	boolean, whether the model is for a directed network
selfloops	boolean, whether the model contains selfloops
regular	boolean. Is the combinatorial matrix computed for configuration model or for regular gnp model? default FALSE.

Value

combinatorial matrix

Examples

```
data('adj_karate')
xi = compute_xi(adj_karate, directed = FALSE, selfloops = FALSE)
```

`conf.test` *Test regular (gnp) vs configuration model*

Description

Likelihood ratio test for gnp vs configuration model.

Usage

```
conf.test(
  graph,
  directed,
  selfloops,
  nempirical = NULL,
  parallel = NULL,
  seed = NULL
)
```

Arguments

`graph`	adjacency matrix or igraph graph
`directed`	a boolean argument specifying whether object is directed or not.
`selfloops`	a boolean argument specifying whether the model should incorporate selfloops.
`nempirical`	optional, number of graphs to sample from null distribution for empirical distribution.
`parallel`	optional, number of cores to use or boolean for parallel computation. If passed TRUE uses all cores-1, else uses the number of cores passed. If none passed performed not in parallel.
`seed`	optional integer

Value

p-value of test.

Examples

```
data("adj_karate")
conf.test(graph = adj_karate, directed = FALSE, selfloops = FALSE, seed=123)
```

contacts.adj *Highschool contact network adjacency matrix*

Description

contacts.adj: contains the adjacency matrix of 327 x 327 highschool students.

Usage

```
data(highschool.predictors)
```

Format

327x327 adjacency matrix

Source

http://www.sociopatterns.org

References

Casiraghi, G. Multiplex Network Regression: How do relations drive interactions? 15 (2017).

Mastrandrea, R., Fournet, J. & Barrat, A. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. PLoS One 10, 1–26 (2015).

cospons_mat *Swiss MPs network adjacency matrix*

Description

cospons_mat: contains the adjacency matrix of 163 x 163 MPs.

Usage

```
data(cospons_mat)
```

Format

163x163 adjacency matrix

`coxsnellR2` *Computes Cox and Snell pseudo R-squared for nrm models.*

Description

Computes Cox and Snell pseudo R-squared for nrm models.

Usage

```
coxsnellR2(mod0, mod1, m)
```

Arguments

`mod0`	nrm null model
`mod1`	nrm alternative model
`m`	number of edges

Value

Cox and Snell pseudo R-squared

Author(s)

GC

`CreateIgGraphs` *Convert a list of adjacency matrices to a list of igraph graphs.*

Description

Convert a list of adjacency matrices to a list of igraph graphs.

Usage

```
CreateIgGraphs(adjlist, directed, selfloops, weighted = NULL)
```

Arguments

`adjlist`	a list of adjacency matrices
`directed`	a boolean argument specifying whether object is directed or not.
`selfloops`	a boolean argument specifying whether the model should incorporate selfloops.
`weighted`	boolean, generate weighted graphs?

Value

list of igraph graphs.

Examples

```
data('adj_karate')
adj_list <- list(adj_karate)
glist <- CreateIgGraphs(adj_list, FALSE, FALSE)
```

`create_predictors` *Create a nrmpredictor object from passed argument*

Description

Create a nrmpredictor object from passed argument

Usage

```
create_predictors(predictors, ...)

createPredictors(predictors, ...)
```

Arguments

`predictors`	the dataframe or list of predictors for to apply nrm model selection
`...`	additional parameters passed to the different methods (currently disabled)

Value

nested list of nrmpredictor class

Examples

```
data('highschool.predictors')
predictors <- create_predictors(highschool.predictors)
```

create_predictors.list *Create a nrmpredictor object from list*

Description

Create a nrmpredictor object from list

Usage

```
## S3 method for class 'list'
create_predictors(predictors, ...)
```

Arguments

predictors	the dataframe or list of predictors for to apply nrm model selection
...	additional parameters used to creating the predictor object (currently disabled)

Value

nested list of nrmpredictor class

Examples

```
data('highschool.predictors')
predictors <- create_predictors(highschool.predictors)
```

dt *Swiss MPs attribute data frame.*

Description

dt: contains different attributes of the 163 MPs, such as their names, their party affiliation (variable: *party*), their parliamentary group affiliation (variable: *parlGroup*), the Canton (or state) they represent (variable: *canton*), their gender (variable: *gender*) and date of birth (variable: *birthdate*).

Usage

```
data(dt)
```

Format

163x8 data.frame

dtcommittee *Swiss MPs committee affiliation data frame.*

Description

dtcommittee: a list of committees each MP was part of during their stay in parliament

Usage

```
data(dtcommittee)
```

Format

163x2 data.frame

el2adj *Maps edgelist to adjacency matrix*

Description

Maps edgelist to adjacency matrix

Usage

```
el2adj(el, nodes = NULL)
```

Arguments

el	dataframe containing a (weighted) edgelist. Column 1 is the sender, column 2 is the receiver, column 3 the number of edges.
nodes	optional vector containing all node names in case disconnected nodes should be included.

Value

the (weighted) adjacency matrix corresponding the edgelist passed

`extract.nrm.cluster` *Extract details from statistical models for table construction. The function has methods for a range of statistical models.*

Description

Extract details from statistical models for table construction. The function has methods for a range of statistical models.

Usage

```
extract.nrm.cluster(model, ...)
```

Arguments

`model`	A statistical model object.
`...`	Custom parameters, which are handed over to subroutines. The arguments are usually passed to the summary function, but in some cases to other functions.

Value

The function returns a texreg object.

Author(s)

L. Brandenberger, G. Casiraghi

`FitOmega` *Fit propensity matrix for full model*

Description

(auxiliary function)

Usage

```
FitOmega(adj, xi, directed, selfloops)
```

Arguments

`adj`	adjacency matrix
`xi`	combinatorial matrix
`directed`	boolean
`selfloops`	boolean

Value

propensity matrix

Examples

```
data(adj_karate)
xi <- compute_xi(adj_karate, FALSE, FALSE)
FitOmega(adj_karate, xi, FALSE, FALSE)
```

`get_zero_dummy` *Create a dummy variable to encode zero values of another variable.*

Description

Use this to substitute zero-values in your nrm values. Zero values in the predictors are recognized in the gHypEG regression as structural zeroes. To ensure this does not happen, please recode your zero-values in all your predictors, ideally using a dummy variable fitting an optimal value for the zeroes. This function takes a predictor that needs to be recoded and returns a list containing two matrices. The first one contains the original predictor recoded such that all zero values are 1 (and thus do not impact the model). The second one consist of a matrix with 1 where the original predictor was different from 0, and 'zero_values' where the original predictor was 0. If 'zero_values' is not specified, it is fixed to e to simplify the interpretation of the results.

Usage

```
get_zero_dummy(dat, name = NULL, zero_values = NULL)
```

Arguments

`dat`	matrix, the predictor for which the zero values should be recoded.
`name`	optional character, the name of the predictor to create a named list
`zero_values`	optional numeric, the value to assign to the zero values of 'dat' in the dummy variable. It defaults to e to simplify the interpretation of the results.

Value

a possibly named list of two matrices. The first one is the recoded version of 'dat' where all zeroes are changed to 1. The second is the dummy variable such that dummy[dat==0] <- zero_values and 1 otherwise.

See Also

`reciprocity_stat` or `sharedPartner_stat`

ghype *Fitting gHypEG models*

Description

ghype is used to fit gHypEG models when the propensity matrix is known. It can be used to estimate a null model (soft configuration model), or the benchmark 'full-model', where the propensity matrix is fitted such that the expected graph from the fitted model is the one passed to the function.

Usage

```
ghype(
  graph,
  directed,
  selfloops,
  xi = NULL,
  omega = NULL,
  unbiased = FALSE,
  regular = FALSE,
  ...
)

## S3 method for class 'matrix'
ghype(
  graph,
  directed,
  selfloops,
  xi = NULL,
  omega = NULL,
  unbiased = FALSE,
  regular = FALSE,
  ...
)

## Default S3 method:
ghype(
  graph,
  directed,
  selfloops,
  xi = NULL,
  omega = NULL,
  unbiased = FALSE,
  regular = FALSE,
  ...
)

## S3 method for class 'igraph'
```

```
ghype(
  graph,
  directed,
  selfloops,
  xi = NULL,
  omega = NULL,
  unbiased = FALSE,
  regular = FALSE,
  ...
)

## S3 method for class 'ghype'
print(x, suppressCall = FALSE, ...)
```

Arguments

graph	either an adjacency matrix or an igraph graph.
directed	a boolean argument specifying whether graph is directed or not.
selfloops	a boolean argument specifying whether the model should incorporate selfloops.
xi	an optional matrix defining the combinatorial matrix of the model.
omega	an optional matrix defining the propensity matrix of the model.
unbiased	a boolean argument specifying whether to model the hypergeometric ensemble (no propensity), defaults to FALSE.
regular	a boolean argument specifying whether to model the 'gnp' ensemble (no xi), defaults to FALSE.
...	further arguments passed to or from other methods.
x	ghype model
suppressCall	boolean, suppress print of the call

Value

ghype return an object of class "ghype".

Methods (by class)

- `ghype(matrix)`: Fitting ghype models from an adjacency matrix
- `ghype(default)`: Generating a ghype model from given xi and omega
- `ghype(igraph)`: Fitting ghype models from an igraph graph

Methods (by generic)

- `print(ghype)`: Print method for ghype object.

Examples

```
data("adj_karate")
fullmodel <- ghype(graph = adj_karate, directed = FALSE, selfloops = FALSE, unbiased = FALSE)

data('adj_karate')
model <- scm(adj_karate, FALSE, FALSE)
print(model)
```

---

gof.test *Perform a goodness-of-fit test*

Description

Perform a goodness-of-fit test

Usage

```
gof.test(
  model,
  Beta = TRUE,
  nempirical = NULL,
  parallel = NULL,
  returnBeta = FALSE,
  seed = NULL
)
```

Arguments

model	ghype model to test
Beta	boolean, whether to use empirical Beta distribution approximation. Default TRUE
nempirical	optional scalar, number of replicates for empirical beta distribution.
parallel	optional, number of cores to use or boolean for parallel computation. If passed TRUE uses all cores-1, else uses the number of cores passed. If none passed performed not in parallel.
returnBeta	boolean, return estimated parameters of Beta distribution? Default FALSE.
seed	scalar, seed for the empirical distribution.

Value

p-value of test. If returnBeta=TRUE returns the p-value together with the parameters of the beta distribution.

Examples

```
data("adj_karate")
confmodel <- scm(graph = adj_karate, directed = FALSE, selfloops = FALSE)
gof.test(model = confmodel, seed = 123)
```

---

highschool.multiplex *Highschool contact network multiplex representation*

Description

highschool.multiplex: list containing the adjacency matrix of 327 x 327 highschool students, and the adjacency matrices corresponding to the 5 predictors used in Casiraghi2017.

Usage

```
data(highschool.multiplex)
```

Format

6x327x327 list of adjacency matrices

Source

http://www.sociopatterns.org

References

Casiraghi, G. Multiplex Network Regression: How do relations drive interactions? 15 (2017).

Mastrandrea, R., Fournet, J. & Barrat, A. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. PLoS One 10, 1–26 (2015).

---

highschool.predictors *Highschool contact network predictors*

Description

highschool.predictors: list containing the adjacency matrices corresponding to the 5 predictors used in Casiraghi2017.

Usage

```
data(highschool.predictors)
```

Format

5x327x327 list of adjacency matrices

Source

http://www.sociopatterns.org

References

Casiraghi, G. Multiplex Network Regression: How do relations drive interactions? 15 (2017).

Mastrandrea, R., Fournet, J. & Barrat, A. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. PLoS One 10, 1–26 (2015).

---

homophily_stat *Calculate homophily in multi-edge graphs.*

Description

The function calculates homophily matrices. If you supply a categorical variable (factor, character), the function returns attribute matches for dyads from the same group. If you supply a continuous variable (numeric, integers), the function returns absolute difference effects for each dyad in the graph.

Usage

```
homophily_stat(
  variable = variable,
  type = "categorical",
  nodes = nodes,
  these.categories.only = NULL,
  zero_values = NULL
)
```

Arguments

variable	A attribute variable. Can be categorical (attribute matches) or continuous (absolute difference effects).
type	set to categorical. Can be set to absdiff instead. If set to categorical, the homophily statistic calculates matches between dyads from the same group (analogous to dummy variables measuring attribute match between two nodes (=10) and attribute mismatch (=1)). If set to absdiff it calculates the difference in values from variable for each dyad in the graph.
nodes	optional character/factor vector. If an edgelist is provided, you have to provide a list of unique identifiers of your nodes in the graph. This is because in the edgelist, isolates are usually not recorded. If you do not specify isolates in your nodes object, they are excluded from the analysis (falsifies data).

these.categories.only
: optional vector specifying the categories to be used, if only a subset of factor(variable) is needed.

zero_values
: optional numeric value. Use this to substitute zero-values in your homophily change statistic matrix. Zero values in the predictors are recognized in the gHypEG regression as structural zeroes. To ensure this does not happen, please recode your zero-values in all your predictors, ideally using a dummy variable fitting an optimal value for the zeroes. Only useful with absdiff type.

Value

Homophily change statistic matrix.

Author(s)

LB, GC

See Also

reciprocity_stat or sharedPartner_stat

Examples

```
homop_stat <- homophily_stat(variable = vertexlabels, nodes = rownames(adj_karate))
nrm(w=list('homophily'= homop_stat), adj_karate, directed = FALSE, selfloops = FALSE)
```

---

isNetwork *Test null model vs full ghype.*

Description

isNetwork tests a graph for the SCM vs the full ghype model.

Usage

```
isNetwork(
  graph,
  directed,
  selfloops,
  Beta = TRUE,
  nempirical = NULL,
  parallel = FALSE,
  returnBeta = FALSE,
  seed = NULL
)
```

Arguments

graph	adjacency matrix or igraph graph
directed	a boolean argument specifying whether object is directed or not.
selfloops	a boolean argument specifying whether the model should incorporate selfloops.
Beta	boolean, use Beta test? default TRUE
nempirical	optional, number of graphs to sample from null distribution for empirical distribution.
parallel	optional, number of cores to use or boolean for parallel computation. If passed TRUE uses all cores-1, else uses the number of cores passed. If none passed performed not in parallel.
returnBeta	boolean, return estimated parameters of Beta distribution? Default FALSE.
seed	optional integer, seed for empirical lr.test

Value

p-value of test.

Examples

```
data("adj_karate")
isNetwork(graph = adj_karate, directed = FALSE, selfloops = FALSE, seed=123)
```

JnBlock *Fisher Information matrix for estimators in block models.*

Description

Fisher Information matrix for estimators in block models.

Usage

```
JnBlock(omegaBlocks, xiBlocks, mBlocks, m)
```

Arguments

omegaBlocks	the block parameters (vector)
xiBlocks	the xi-block (vector)
mBlocks	the adj-block (vector)
m	the number of edges (scalar)

Value

Fisher Information matrix

`linkSignificance` *Estimate statistical deviations from ghype model*

Description

linkSignificance allows to estimate the statistical deviations of an observed graph from a ghype model.

Usage

```
linkSignificance(
  graph,
  model,
  under = FALSE,
  log.p = FALSE,
  binomial.approximation = FALSE,
  give_pvals = FALSE
)

link_significance(
  graph,
  model,
  under = FALSE,
  log.p = FALSE,
  binomial.approximation = FALSE,
  give_pvals = TRUE
)
```

Arguments

`graph`	an adjacency matrix or a igraph object.
`model`	a ghype model
`under`	boolean, estimate under-represented deviations? Default FALSE: i.e. returns over representation
`log.p`	boolean, return log values of probabilities
`binomial.approximation`	boolean, force binomial? default FALSE
`give_pvals`	boolean, return p-values for both under and over significance? when FALSE, it returns probabilty of observing stricly more (or less) edges than in graph. When TRUE returns probability of observing exactly as many edges or more (less) than in graph, like a standard pvalue.

Value

matrix of probabilities with same size as adjacency matrix.

Examples

```
data("adj_karate")
fullmodel <- ghype(graph = adj_karate, directed = FALSE, selfloops = FALSE)
link_significance(graph = adj_karate, model = fullmodel, under=FALSE)
```

`logl` *General method to compute log-likelihood for ghype models.*

Description

General method to compute log-likelihood for ghype models.

Usage

```
logl(
  object,
  xi = NULL,
  omega = NULL,
  directed = NULL,
  selfloops = NULL,
  adj = NULL,
  multinomial = NULL,
  ...
)

## S3 method for class 'ghype'
logl(
  object,
  xi = NULL,
  omega = NULL,
  directed = NULL,
  selfloops = NULL,
  adj = NULL,
  multinomial = NULL,
  ...
)

## S3 method for class 'matrix'
logl(
  object,
  xi = NULL,
  omega = NULL,
  directed = NULL,
  selfloops = NULL,
  adj = NULL,
  multinomial = NULL,
```

```
  ...
)
```

Arguments

object	either an adjacency matrix or ghype model If a ghype model is passed, then ‘xi‘, ‘omega‘, ‘directed‘, ‘selfloops‘ are ignored If an adjacency matrix is passed, then ‘adj‘ is ignored
xi	matrix, combinatorial matrix to build ghype model, considered only if object is an adjacency matrix
omega	matrix, propensity matrix to build ghype model, considered only if object is an adjacency matrix
directed	boolean, is ghype model directed? considered only if object is an adjacency matrix
selfloops	boolean, has ghype model selfloops? considered only if object is an adjacency matrix
adj	optional matrix, adjacency matrix of which to compute log-likelihood, considered only if object is ghype model If adj is not passed, and object is a ghype model, the log-likelihood is computed for the original adjacency matrix stored in object.
multinomial	optional boolean. Force multinomial approximation? If not chosen, multinomial chosen for large graphs.
...	additional parameters passed to and from internal methods

Value

loglikelihood value

Methods (by class)

- `logl(ghype)`: Computes log-likelihood for ghype models from model object
- `logl(matrix)`: Computes log-likelihood for ghype models from adjacency.

Examples

```
data('adj_karate')
model <- scm(adj_karate, FALSE, FALSE)
logl(object = model)
new_adj <- adj_karate
new_adj[3,4] <- 10
logl(object=model, adj=new_adj)
```

logLik.ghype *Extract Log-Likelihood*

Description

Extract Log-Likelihood

Usage

```
## S3 method for class 'ghype'
logLik(object, ...)
```

Arguments

object	ghype model.
...	additional arguments passed to and from internal methods.

Value

Returns an object of class logLik. This is a number with at least one attribute, "df" (degrees of freedom), giving the number of (estimated) parameters in the model.

loglratio *Compute log-likelihood ratio for ghype models.*

Description

Compute log-likelihood ratio for ghype models.

Usage

```
loglratio(mod0, mod1)
```

Arguments

mod0	ghype, null model
mod1	ghype, alternative model

Value

scalar, log-likelihood ratio

Examples

```
data('adj_karate')
sc.model <- scm(adj_karate, FALSE, FALSE)
full.model <- ghype(adj_karate, FALSE, FALSE)
loglratio(sc.model,full.model)
```

`lr.test` *Perform likelihood ratio test between two ghype models.*

Description

lr.test allows to test between two nested ghype models whether there is enough evidence for the alternative (more complex) model compared to the null model.

Usage

```
lr.test(
  nullmodel,
  altmodel,
  df = NULL,
  Beta = TRUE,
  seed = NULL,
  nempirical = NULL,
  parallel = FALSE,
  returnBeta = FALSE,
  method = NULL
)
```

Arguments

`nullmodel`	ghype object. The null model
`altmodel`	ghype object. The alternative model
`df`	optional scalar. the number of degrees of freedom.
`Beta`	boolean, whether to use empirical Beta distribution approximation. Default TRUE
`seed`	scalar, seed for the empirical distribution.
`nempirical`	optional scalar, number of replicates for empirical beta distribution.
`parallel`	optional, number of cores to use or boolean for parallel computation. If passed TRUE uses all cores-1, else uses the number of cores passed. If none passed performed not in parallel.
`returnBeta`	boolean, return estimated parameters of Beta distribution? Default FALSE.
`method`	string, for internal use

Value

p-value of test. If returnBeta=TRUE returns the p-value together with the parameters of the beta distribution.

Examples

```
data("adj_karate")
regularmodel <- regularm(graph = adj_karate, directed = FALSE, selfloops = FALSE)
confmodel <- scm(graph = adj_karate, directed = FALSE, selfloops = FALSE)
lr.test(nullmodel = regularmodel, altmodel = confmodel, seed = 123)
```

---

mat2vec.ix *Auxiliary function, gives mask for matrix for directed, undirected etc.*

Description

Auxiliary function, gives mask for matrix for directed, undirected etc.

Usage

```
mat2vec.ix(mat, directed, selfloops)
```

Arguments

mat	matrix
directed	a boolean argument specifying whether object is directed or not.
selfloops	a boolean argument specifying whether the model should incorporate selfloops.

Value

a boolean matrix that can be used to mask adjacency matrices.

Examples

```
data('adj_karate')
mat2vec.ix(adj_karate, FALSE, FALSE)
```

`mcfaddenR2` *Computes Mc Fadden pseudo R-squared.*

Description

Pass either the models or the model parameters as arguments

Usage

```
mcfaddenR2(
  adj = NULL,
  xi = NULL,
  omega0 = NULL,
  omega1 = NULL,
  directed,
  selfloops,
  mod0 = NULL,
  mod1 = NULL,
  nparam
)
```

Arguments

`adj`	optimal adjacency matrix
`xi`	optional xi matrix
`omega0`	optional propensity matrix of null model
`omega1`	optional propensity matrix of alternative model
`directed`	boolean, is the model directed?
`selfloops`	boolean, are there selfloops?
`mod0`	nrm null model
`mod1`	nrm alternative model
`nparam`	integer, number of parameters

Value

Mc Fadden pseudo R-squared.

nr.ci *Confidence intervals for nrm models.*

Description

Internal function to compute confidence intervals for estimated parameters of nrm model

Usage

```
nr.ci(nr.m, w, adj, pval)
```

Arguments

nr.m	nrm model from which getting coefficients
w	list of predictors
adj	adjacency matrix
pval	numeric. confidence level

Value

matrix reporting values of predictors and confidence bounds

nr.significance *Computes the significance of more complex model against a simpler model by means of a likelihood ratio test.*

Description

Computes the significance of more complex model against a simpler model by means of a likelihood ratio test.

Usage

```
nr.significance(mod0 = NULL, mod1, adj = NULL)
```

Arguments

mod0	null nrm model (optional). defaults to the scm model.
mod1	alternative nrm model, the model to test
adj	adjacency matrix for which performing the test. (optional) defaults to the matrix used for mod1.

Value

p-value of the lr test mod0 vs mod1

nrm *Fitting gHypEG regression models for multi-edge networks.*

Description

nrm is used to fit multi-edge network regression models.

Usage

```
nrm(
  w,
  adj,
  xi = NULL,
  pval = 0.01,
  directed = TRUE,
  selfloops = TRUE,
  regular = FALSE,
  ...
)

## Default S3 method:
nrm(
  w,
  adj,
  xi = NULL,
  pval = 0.01,
  directed = FALSE,
  selfloops = FALSE,
  regular = FALSE,
  ci = TRUE,
  significance = FALSE,
  null = FALSE,
  init = NULL,
  ...
)

## S3 method for class 'nrm'
print(x, suppressCall = FALSE, ...)
```

Arguments

w	an object of class 'list' containing the predictors layers (explanatory variables/covariates) of the multiplex, passed as adjacency matrices. The entries of the list can be named.
adj	matrix. The adjacency matrix of the response network (dependent variable).
xi	optional matrix. Passes a non-standard Ξ matrix.

pval	the significance level used to compute confidence intervals of the parameters. Per default, set to 0.01.
directed	logical. If TRUE the response variable is considered the adjacency matrix of directed graph. If FALSE only the upper triangular of adj is considered. Default set to FALSE.
selfloops	logical. Whether selfloops are allowed. Default set to FALSE.
regular	logical. Whether the gHypEG regression should be performed with correction of combinatorial effects (TRUE) or without (FALSE).
...	optional arguments to print or plot methods.
ci	logical. Whether to compute confidences for the parameters. Defaults to TRUE.
significance	logical. Whether to test the model significance against the null by means of lr-test.
null	logical. Is this a null model? Used for internal routines.
init	numeric. Vector of initial values used for numerical MLE. If only a single value is passed, this is repeated to match the number of predictors in w.
x	object of class 'nrm'
suppressCall	logical, indicating whether to print the call that generated x

Value

nrm returns an object of class 'nrm'.

The function summary is used to obtain and print a summary and analysis of the results. The generic accessory functions coefficients, etc, extract various useful features of the value returned by nrm.

An object of class 'nrm' is a list containing at least the following components:

coef	a named vector of coefficients.
confint	a named matrix with confidence intervals and standard deviation for each coefficient.
omega	the estimated propensity matrix.
xi	the matrix of possibilities.
loglikelihood	log-likelihood of the estimated model.
AIC	AIC of the estimated model.
R2	Mc Fadden pseudo R-squared
csR2	Cox and Snells pseudo R-squared
significance	the p-value of the likelihood-ratio test for the estimated model against the null.

Methods (by class)

- nrm(default): Default method for nrm

Methods (by generic)

- print(nrm): Print method for elements of class 'nrm'.

Author(s)

Giona Casiraghi

References

Casiraghi, Giona. 'Multiplex Network Regression: How do relations drive interactions?.' arXiv preprint arXiv:1702.02048 (2017).

See Also

nrm

Examples

```
## For a complete example see the vignette

data('highschool.predictors')

highschool.m <- nrm(w=highschool.predictors[1], adj=contacts.adj, directed=FALSE,
  selfloops=FALSE)

highschool.m


data('highschool.predictors')

highschool.m <- nrm(w=highschool.predictors, adj=contacts.adj, directed=FALSE,
  selfloops=FALSE)

highschool.m
```

---

nrmChoose *Selects the best set of predictors among the given sets by means of AIC.*

Description

Computes all the models defined by a list of groups of predictors Returns the best model according to AIC and id of the corresponding predictors in the list The different models are computed in parallel

Usage

```
nrmChoose(
  adj,
  w.list,
  xi = NULL,
```

```
  directed,
  selfloops,
  pval = 0.05,
  init = NULL,
  ncores = NULL
)

nrm_choose(
  adj,
  w.list,
  xi = NULL,
  directed,
  selfloops,
  pval = 0.05,
  init = NULL,
  ncores = NULL
)
```

Arguments

adj	adjacency matrix
w.list	nrmPredictor object. Nested list of predictors to be selected.
xi	Xi matrix (optional). defaults to scm Xi matrix.
directed	logical. Is the network directed?
selfloops	logical. Does the network contain selfloops?
pval	numeric. the significance at which computing confidence intervals. defaults to 0.05
init	initial values for the MLE numerical maximisation. (See nrm.)
ncores	Number of cores for parallelisation of selection process. (optional) Defaults to number of available cores - 1.

Value

list containing the best model according to AIC and id of the corresponding predictors in the list

---

nrmSelection *Perform AIC forward selection for nrm.*

Description

Perform AIC forward selection for nrm.

Usage

```
nrmSelection(
  adj,
  predictors,
  directed,
  selfloops,
  pval = 0.05,
  xi = NULL,
  init = NULL,
  ncores = NULL,
  ...
)

nrm_selection(
  adj,
  predictors,
  directed,
  selfloops,
  pval = 0.05,
  xi = NULL,
  init = NULL,
  ncores = NULL,
  ...
)

## Default S3 method:
nrm_selection(
  adj,
  predictors,
  directed,
  selfloops,
  pval = 0.05,
  xi = NULL,
  init = NULL,
  ncores = NULL,
  ...
)

## S3 method for class 'nrmpredictor'
nrm_selection(
  adj,
  predictors,
  directed,
  selfloops,
  pval = 0.05,
  xi = NULL,
  init = NULL,
  ncores = NULL,
```

```
  ...
)

## S3 method for class 'nrm_selection'
print(x, ...)
```

Arguments

adj	the adjacency matrix of the response network
predictors	list containing the set of predictors as sublists.
directed	logical, is the response network directed?
selfloops	logical, do the response network allows selfloops?
pval	the significance at which computing confidence intervals.
xi	optional, the possibility matrix Ξ.
init	optional, initial values passed to the solver to estimate the MLE.
ncores	optional, number of cores over which parallelise the task.
...	optional arguments to print or plot methods.
x	object of class 'nrm_selection'.

Value

A nrm object

Methods (by class)

- nrm_selection(default): Default method for the nrm stepwise selection.
- nrm_selection(nrmpredictor): Method for the nrm stepwise selection when nrmpredictors are passed.

Methods (by generic)

- print(nrm_selection): Print method for elements of class 'nrm_selection'.

Author(s)

Giona Casiraghi

See Also

nrm

nrm_selection

Examples

```
data('highschool.predictors')
models <- nrm_selection(adj=contacts.adj,predictors=create_predictors(highschool.predictors),
  ncores=1,directed=FALSE,selfloops=FALSE)
texreg::screenreg(models$models, digits=3)
```

---

onlinesim_mat *Swiss MPs committee similarity matrix.*

Description

onlinesim_mat: a similarity matrix of how similar two MPs are in their online social media presence (shared supportees).

Usage

```
data(onlinesim_mat)
```

Format

163x163 similarity matrix

---

predict.nrm *Method to predict the expected values of a nrm model*

Description

Method to predict the expected values of a nrm model

Usage

```
## S3 method for class 'nrm'
predict(object, m = NULL, adj = NULL, null = FALSE, multinomial = NULL, ...)
```

Arguments

object	nrm object from which to predict
m	integer, the number of edges to be used
adj	optional matrix, the adjacency matrix from which to get the number of edges
null	optional boolean, is it a null model? default FALSE
multinomial	logical. Optional argument. Whether to use multinomial approximation. If left blank it is selected automatically based on network size.
...	other arguments

Value

numeric, predicted values from nrm model. (If model is undirected, only upper.tri of adjacency matrix is returned.)

Examples

```
data('highschool.predictors')
highschool.m <- nrm(w=highschool.predictors[1], adj=contacts.adj, directed=FALSE, selfloops=FALSE)
predict(highschool.m, contacts.adj)

data('highschool.predictors')
highschool.m <- nrm(w=highschool.predictors, adj=contacts.adj, directed=FALSE, selfloops=FALSE)
predict(highschool.m, contacts.adj)
```

---

reciprocity_stat *Calculate weighted reciprocity change statistics for multi-edge graphs.*

Description

The function takes either an edgelist or an adjacency matrix and returns an adjacency matrix with the reciprocity change statistic. This reciprocity matrix can then be used as a predictor in the gHypEG regression.

Usage

```
reciprocity_stat(graph, nodes = NULL, zero_values = NULL)
```

Arguments

graph	A graph adjacency matrix or an edgelist. The edgelist needs to have 3 columns: a sender vector, a target vector and an edgecount vector.
nodes	optional character/factor vector. If an edgelist is provided, you have to provide a list of unique identifiers of your nodes in the graph. This is because in the edgelist, isolates are usually not recorded. If you do not specify isolates in your nodes object, they are excluded from the analysis (falsifies data).
zero_values	optional numeric value. Use this to substitute zero-values in your reciprocity change statistic matrix. Zero values in the predictors are recognized in the gHypEG regression as structural zeros. To ensure this does not happen, please recode your zero-values in all your predictors, ideally using a dummy variable fitting an optimal value for the zeroes.

Value

Reciprocity change statistic matrix.

Author(s)

LB, GC

See Also

sharedPartner_stat or homophily_stat

Examples

```
recip_stat <- reciprocity_stat(adj_karate)
recip_stat_dummy <- get_zero_dummy(recip_stat, name = 'reciprocity')
nrm(w=recip_stat_dummy, adj_karate, directed = FALSE, selfloops = FALSE)
```

---

regularm *Fit the gnm model*

Description

regularm is wrapper for ghype that allows to specify a gnm regular model. i.e. where all entries of the combinatorial matrix Xi are the same.

Usage

```
regularm(graph, directed = NULL, selfloops = NULL, ...)
```

Arguments

graph	either an adjacency matrix or an igraph graph
directed	optional boolean, if not specified detected from graph
selfloops	optional boolean, if not specified detected from graph
...	additional parameters passed to the ghype function

Value

ghype object

Examples

```
data("adj_karate")
regularmodel <- regularm(graph = adj_karate, directed = FALSE, selfloops = FALSE)
```

residuals.nrm *Method to compute residuals of nrm models*

Description

Method to compute residuals of nrm models

Usage

```
## S3 method for class 'nrm'
residuals(object, adj, RMSLE = FALSE, null = FALSE, ...)
```

Arguments

object	nrm object
adj	adjacency against which to compute residuals
RMSLE	logical, return log residuals? default FALSE
null	logical. use null model?
...	additional parameters to be passed to and from internal functions.

Value

numeric vector, residuals of nrm model fit against the original data

Examples

```
data('highschool.predictors')
highschool.m <- nrm(w=highschool.predictors[1], adj=contacts.adj, directed=FALSE, selfloops=FALSE)
residuals(highschool.m, contacts.adj)
```

rghype *Generate random realisations from ghype model.*

Description

Generate random realisations from ghype model.

Usage

```
rghype(nsamples, model, m = NULL, multinomial = NULL, seed = NULL)
```

Arguments

nsamples	scalar number of realisations
model	ghype model
m	optional scalar, number of edges to draw
multinomial	optional boolean, draw from multinomial?
seed	optional scalar, seed for random sampling.

Value

list of adjacency matrices.

Examples

```
data('adj_karate')
model <- scm(adj_karate, FALSE, FALSE)
rghype(1, model)
```

RMSE *Computes the Root Mean Squared Error*

Description

Computes the Root Mean Squared Error

Usage

```
RMSE(model, adj, null = FALSE)
```

Arguments

model	nrm model estimate
adj	original adjacency matrix
null	logical, whether to compute using null model

Value

numeric, root mean squared error of residuals of nrm model fit

Examples

```
data('highschool.predictors')
highschool.m <- nrm(w=highschool.predictors[1], adj=contacts.adj, directed=FALSE, selfloops=FALSE)
RMSE(highschool.m, contacts.adj)
```

RMSLE *Computes the Root Mean Squared Logged Error*

Description

Computes the Root Mean Squared Logged Error

Usage

```
RMSLE(model, adj, null = FALSE)
```

Arguments

model	nrm model estimate
adj	original adjacency matrix
null	logical, whether to compute using null model

Value

numeric, root mean squared logged error of residuals of nrm model fit

Examples

```
data('highschool.predictors')
highschool.m <- nrm(w=highschool.predictors[1], adj=contacts.adj, directed=FALSE, selfloops=FALSE)
RMSLE(highschool.m, contacts.adj)
```

scm *Fit the Soft-Configuration Model*

Description

scm is wrapper for ghype that allows to specify a soft-configuration model.

Usage

```
scm(graph, directed = NULL, selfloops = NULL, ...)
```

Arguments

graph	either an adjacency matrix or an igraph graph
directed	optional boolean, if not specified detected from graph
selfloops	optional boolean, if not specified detected from graph
...	additional parameters passed to the ghype function

Value

ghype object

Examples

```
data("adj_karate")
confmodel <- scm(graph = adj_karate, directed = FALSE, selfloops = FALSE)
```

`sharedPartner_stat` *Calculate (un-)weighted shared partner change statistics for multi-edge graphs.*

Description

The function calculates the change statistic for shared partners for each dyad in the graph. Shared partner statistics count for each dyad involving nodes i and j in the graph, how many nodes k these two nodes have in common (or share). The shared partner k counts are weighted by their interactions with the focal nodes i and j. This is necessary in dense multi-edge graphs to ensure that meaningful triadic closure is detected. The statistic can be calculated in 3 different forms: undirected, incoming shared partners (where shared partner k: k->i and k->j) and outgoing shared partners (where shared partner k: k<-i and k<-j).

Usage

```
sharedPartner_stat(
  graph,
  directed,
  weighted = TRUE,
  triad.type = "undirected",
  nodes = NULL,
  zero_values = NULL
)
```

Arguments

`graph`	A graph adjacency matrix or an edgelist. The edgelist needs to have 3 columns: a sender vector, a target vector and an edgecount vector.
`directed`	boolean. Is the graph directed?
`weighted`	set to TRUE.
`triad.type`	set to `undirected`. Can be set to `incoming` or `outgoing` instead. This then corresponds to directed triadic closure in the multi-edge graph.
`nodes`	optional character/factor vector. If an edgelist is provided, you have to provide a list of unique identifiers of your nodes in the graph. This is because in the edgelist, isolates are usually not recorded. If you do not specify isolates in your nodes object, they are excluded from the analysis (falsifies data).

zero_values	optional numeric value. Use this to substitute zero-values in your shared partner change statistic matrix. Zero values in the predictors are recognized in the gHypEG regression as structural zeros. To ensure this does not happen, please recode your zero-values in all your predictors, ideally using a dummy variable fitting an optimal value for the zeroes.

Value

Shared partner change statistic matrix.

Author(s)

LB, GC, GV

See Also

reciprocity_stat or homophily_stat

Examples

```
tri_stat <- sharedPartner_stat(adj_karate, directed = FALSE)
tri_stat_dummy <- get_zero_dummy(tri_stat, name = 'shared_partners')
nrm(w=tri_stat_dummy, adj_karate, directed = FALSE, selfloops = FALSE)
```

---

summary.nrm *Summary method for elements of class 'nrm'.*

Description

Currently it provides the same output as 'print.nrm'

Usage

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

## S3 method for class 'summary.nrm'
print(x, ...)
```

Arguments

object	an object of class 'nrm', usually, a result of a call to nrm.
...	further arguments passed to or from other methods.
x	object of class 'summary.nrm' returned by [summary.nrm()].

Value

The function summary.nrm computes and returns a list of summary statistics of the fitted nrm model given in object.

summary.nrm_selection *Summary method for elements of class* 'nrm_selection'.

Description

Summary method for elements of class 'nrm_selection'.

Usage

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

## S3 method for class 'summary.nrm_selection'
print(x, ...)
```

Arguments

object	an object of class 'nrm_selection', usually, a result of a call to nrm_selection.
...	further arguments passed to or from other methods.
x	object of class 'summary.nrm_selection' returned by [summary.nrm._selection()].

Value

The function summary.nrm_selection computes and returns a list of summary statistics of the fitted nrm_selection model given in object.

vec2mat *Auxiliary function, produces matrix from vector*

Description

The number of elements of vec are the number of non-zero elements in the adjacency matrix. It performs the opposite operation of 'mat2vec.ix'.

Usage

```
vec2mat(vec, directed, selfloops, n)
```

Arguments

vec	vector to be put in matrix form
directed	a boolean argument specifying whether object is directed or not.
selfloops	a boolean argument specifying whether the model should incorporate selfloops.
n	vector. if length(n)==1, n is the number of vertices. If length(n)==3 first element is number of vertices, second and third elements are number of vertices for row and column of bipartite matrix.

Value

matrix nxn generated from vector.

Examples

```
data('adj_karate')
ix <- mat2vec.ix(adj_karate, FALSE, FALSE)
vec <- adj_karate[ix]
vec2mat(vec, FALSE, FALSE, nrow(adj_karate))
```

---

vertexlabels *Zachary's Karate Club vertex faction assignment*

Description

Vector reporting the assignment of nodes to communities.

Usage

```
vertexlabels
```

Format

a 34-vector with the assignment of nodes to faction 1 or 2

Source

package 'igraphdata'

Index

∗ **datasets**
adj_karate, 4
contacts.adj, 11
cospons_mat, 11
dt, 14
dtcommittee, 15
highschool.multiplex, 21
highschool.predictors, 21
onlinesim_mat, 39
vertexlabels, 48
∗ **models**
nrm, 33
∗ **multivariate**
nrm, 33
∗ **nonlinear**
nrm, 33
∗ **regression**
nrm, 33
∗ **sna**
nrm, 33

adj2el, 3
adj_karate, 4
as.ghype, 4

bccm, 5, *6*
BootstrapProperty, 6

checkGraphtype, 8
coef.nrm, 8
compute_xi, 9
ComputeXi *(*compute_xi*)*, 9
conf.test, 10
contacts.adj, 11
cospons_mat, 11
coxsnellR2, 12
create_predictors, 13
create_predictors.list, 14
CreateIgGraphs, 12
createPredictors *(*create_predictors*)*, 13

dt, 14
dtcommittee, 15

el2adj, 15
extract.nrm.cluster, 16

FitOmega, 16

get_zero_dummy, 17
ghype, 18, *41*, *44*
gof.test, 20

highschool.multiplex, 21
highschool.predictors, 21
homophily_stat, 22, *41*, *46*

isNetwork, 23

JnBlock, 24

link_significance *(*linkSignificance*)*, 25
linkSignificance, 25
logl, 26
logLik.ghype, 28
loglratio, 28
lr.test, 29

mat2vec.ix, 30
mcfaddenR2, 31

nr.ci, 32
nr.significance, 32
nrm, *9*, 33, *35*, *38*, *46*
nrm_choose *(*nrmChoose*)*, 35
nrm_selection, *47*
nrm_selection *(*nrmSelection*)*, 36
nrmChoose, 35
nrmSelection, 36

onlinesim_mat, 39

predict.nrm, 39

print.bccm *(*bccm*)*, 5
print.ghype *(*ghype*)*, 18
print.nrm *(*nrm*)*, 33
print.nrm_selection *(*nrmSelection*)*, 36
print.summary.nrm *(*summary.nrm*)*, 46
print.summary.nrm_selection
*(*summary.nrm_selection*)*, 47

reciprocity_stat, *17*, *23*, 40, *46*
regularm, 41
residuals.nrm, 42
rghype, 42
RMSE, 43
RMSLE, 44

scm, 44
sharedPartner_stat, *17*, *23*, *41*, 45
summary.nrm, *46*, 46
summary.nrm_selection, *47*, 47

vec2mat, 47
vertexlabels, 48