Package ‘L0ggm’

May 16, 2026

Title Smooth L0 Penalty Approximations for Gaussian Graphical Models

Version 0.1.0

Date 2026-05-15

Description Provides smooth approximations to the L0 norm penalty for estimating sparse Gaussian graphical models (GGMs). Network estimation is performed using the Local Linear Approximation (LLA) framework (Fan & Li, 2001 <doi:10.1198/016214501753382273>; Zou & Li, 2008 <doi:10.1214/009053607000000802>) with five penalty functions: arctangent (Wang & Zhu, 2016 <doi:10.1155/2016/6495417>), EXP (Wang, Fan, & Zhu, 2018 <doi:10.1007/s10463-016-0588-3>), Gumbel, Log (Candes, Wakin, & Boyd, 2008 <doi:10.1007/s00041-008-9045-x>), and Weibull. Adaptive penalty parameters for EXP, Gumbel, and Weibull are estimated via maximum likelihood, and model selection uses information criteria including AIC, BIC, and EBIC (Extended BIC). Simulation functions generate multivariate normal data from GGMs with stochastic block model or small-world (Watts-Strogatz) network structures.

Depends R (>= 3.5.0)

License AGPL (>= 3.0)

Encoding UTF-8

LazyData true

NeedsCompilation yes

Imports igraph, glasso, glassoFast, Matrix, methods, psych, stats

Copyright See inst/COPYRIGHTS for details

BugReports https://github.com/AlexChristensen/L0ggm/issues

Config/roxygen2/version 8.0.0

Author Alexander Christensen [aut, cre] (ORCID: <https://orcid.org/0000-0002-9798-7037>),
Jeongwon Choi [ctb] (ORCID: <https://orcid.org/0000-0001-6087-2124>),
John Fox [cph, ctb] (Original implementation of polyserial correlations in auto_correlate.R),
Yves Rosseel [cph, ctb] (Original implementation of rmsea_ci in network_fit.R),

Alexander Robitzsch [cph, ctb] (C++ implementation of
Drezner-Wesolowsky bivariate normal CDF in polychoric_matrix.c),
David Blackman [ctb] (Original xoshiro.c implementation),
Sebastiano Vigna [ctb] (Original xoshiro.c implementation),
John Burkardt [cph, ctb] (Original ziggurat.c implementation)

Maintainer Alexander Christensen <alexpaulchristensen@gmail.com>

Repository CRAN

Date/Publication 2026-05-15 22:20:02 UTC

Contents

L0ggm-package 2
auto_correlate 3
basic_smallworld 4
categorize 5
edge_confusion 6
network_estimation 9
network_fit 13
polychoric_matrix 15
proxswap_lattice 17
simulate_sbm 20
simulate_smallworld 26
skew_tables 31
smallworldness 32
weibull_parameters 35
weibull_weights 37

Index **39**

`L0ggm-package` *L0ggm-package*

Description

Implements L0 norm regularization penalties for Gaussian graphical models

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

See Also

Useful links:

- Report bugs at https://github.com/AlexChristensen/L0ggm/issues auto_correlate *Automatic correlations*

Description

Automatically computes the proper correlations between continuous and categorical variables. NA values are not treated as categories

Usage

```
auto_correlate(
  data,
  corr = c("kendall", "pearson", "spearman"),
  ordinal_categories = 7,
  forcePD = TRUE,
  na_data = c("pairwise", "listwise"),
  empty_method = c("none", "zero", "all"),
  empty_value = c("none", "point_five", "one_over"),
  forceReturn = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`data` Matrix or data frame. Should consist only of variables to be used in the analysis

`corr` Character (length = 1). The standard correlation method to be used. Defaults to `"pearson"`. Using `"pearson"` will compute polychoric, tetrachoric, polyserial, and biserial correlations for categorical and categorical/continuous correlations by default. To obtain `"pearson"` correlations regardless, use `cor`. Other options of `"kendall"` and `"spearman"` are provided for completeness and use `cor`

`ordinal_categories` Numeric (length = 1). *Up to* the number of categories *before* a variable is considered continuous. Defaults to 7 categories before 8 is considered continuous

`forcePD` Boolean (length = 1). Whether positive definite matrix should be enforced. Defaults to `TRUE`

`na_data` Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`empty_method` Character (length = 1). Method for empty cell correction in `polychoric_matrix`. Defaults to `"none"` Available options:

- `"none"` — Adds no value (`empty_value = "none"`) to the empirical joint frequency table between two variables

- "zero" — Adds empty_value to the cells with zero in the joint frequency table between two variables
- "all" — Adds empty_value to all in the joint frequency table between two variables

empty_value
: Character (length = 1). Value to add to the joint frequency table cells in polychoric_matrix. Defaults to "none". Accepts numeric values between 0 and 1 or specific methods:

- "none" — Adds no value (0) to the empirical joint frequency table between two variables
- "point_five" — Adds 0.5 to the cells defined by empty_method
- "one_over" — Adds 1 / n where n equals the number of cells based on empty_method. For empty_method = "zero", n equals the number of zero cells

forceReturn
: Boolean (length = 1). Whether correlation matrix should be forced to return. Defaults to FALSE. Set to TRUE to receive the correlation matrix "as is"

verbose
: Boolean (length = 1). Whether messages should be printed. Defaults to FALSE

...
: Not actually used but makes it easier for general functionality in the package

Value

A symmetric numeric matrix of dimension $p \times p$, where p is the number of variables in data, with values in $[-1, 1]$ and ones on the diagonal. The correlation method used for each pair of variables depends on their types when corr = "pearson" (the default): polychoric for ordinal-ordinal pairs, polyserial for ordinal-continuous pairs, and Pearson's for continuous-continuous pairs, where *ordinal* means having at most ordinal_categories unique values. When corr is "kendall" or "spearman", cor is used for all pairs regardless of variable type. Row and column names are inherited from data. When forcePD = TRUE (default) and the resulting matrix is not positive definite, the nearest positive definite matrix is returned via nearPD.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Obtain correlations
R <- auto_correlate(basic_smallworld)
```

---

basic_smallworld *Toy Small-world Network Data Example*

Description

A toy small-world network example (n = 500)

Usage

```
data(basic_smallworld)
```

Format

A 500 $\times$ 20 continuous response matrix

Examples

```
# Generated using:
# basic_smallworld <- simulate_smallworld(
#  nodes = 20, # 20 nodes in the network
#  density = 0.30, # moderate initial lattice connectivity
#  rewire = 0.20, # 20% rewiring probability
#  sample_size = 500 # number of cases = 500
# )$data
data("basic_smallworld")
```

---

categorize — *Categorize Continuous Data*

Description

Categorizes continuous data based on Garrido, Abad and Ponsoda (2011; see references). Categorical data with 2 to 6 categories can include skew between -2 to 2 in increments of 0.05

Usage

```
categorize(data, categories, skew_value = 0)
```

Arguments

`data`	Numeric (length = n). A vector of continuous data with *n* values. For matrices, use `apply`
`categories`	Numeric (length = 1). Number of categories to create. Between 2 and 6 categories can be used with skew
`skew_value`	Numeric (length = 1). Value of skew. Ranges between -2 to 2 in increments of 0.05. Skews not in this sequence will be converted to the nearest value in this sequence. Defaults to `0` or no skew

Value

An integer vector of length `n` with values from `1` to `categories`, giving the category assignment for each observation. Category `1` corresponds to the lowest values of `data` and category `categories` to the highest. When `categories` > `6`, `cut` is used and skew is not applied.

Author(s)

Maria Dolores Nieto Canaveras <mnietoca@nebrija.es>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>, Hudson Golino <hfg9s@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Garrido, L. E., Abad, F. J., & Ponsoda, V. (2011).
Performance of Velicer's minimum average partial factor retention method with categorical variables.
Educational and Psychological Measurement, *71*(3), 551-570.

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., ... & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*(3), 292-320.

Examples

```
# Dichotomous data (no skew)
dichotomous <- categorize(
  data = rnorm(1000),
  categories = 2
)

# Dichotomous data (with positive skew)
dichotomous_skew <- categorize(
  data = rnorm(1000),
  categories = 2,
  skew_value = 1.25
)

# 5-point Likert scale (no skew)
five_likert <- categorize(
  data = rnorm(1000),
  categories = 5
)

# 5-point Likert scale (negative skew)
five_likert <- categorize(
  data = rnorm(1000),
  categories = 5,
  skew_value = -0.45
)
```

---

edge_confusion — *Confusion Matrix Metrics for Edge Comparison and Recovery*

Description

Computes many commonly used confusion matrix metrics

Usage

```
edge_confusion(
  base,
  comparison,
  metric = c("all", "sen", "spec", "ppv", "npv", "fdr", "fom", "ba", "f1", "csi", "mcc"),
  full.names = FALSE
)
```

Arguments

base
: Matrix or data frame. Network that will be treated as the "ground truth" such that a false positive represents an edge that is present in `comparison` but not in this network

comparison
: Matrix or data frame. Network that will be treated as the estimator such that a false positive represents an edge that is present in this network but not in `base`

metric
: Character vector. Defaults to `"all"` metrics. Available options:

- `"all"` — All available metrics (default)
- `"sen"` — Sensitivity (True Positive Rate):

$$\frac{TP}{TP + FN}$$

- `"spec"` — Specificity (True Negative Rate):

$$\frac{TN}{TN + FP}$$

- `"ppv"` — Positive Predictive Value (Precision):

$$\frac{TP}{TP + FP}$$

- `"npv"` — Negative Predictive Value:

$$\frac{TN}{TN + FN}$$

- `"fdr"` — False Discovery Rate:

$$1 - PPV = \frac{FP}{TP + FP}$$

- `"fom"` — False Omission Rate:

$$1 - NPV = \frac{FN}{TN + FN}$$

- "ba" — Balanced Accuracy:
$$\frac{Sensitivity + Specificity}{2}$$
- "f1" — F1 Score (harmonic mean of PPV and Sensitivity):
$$\frac{2TP}{2TP + FP + FN}$$
- "csi" — Critical Success Index (Jaccard / Threat Score):
$$\frac{TP}{TP + FP + FN}$$
- "mcc" — Matthews Correlation Coefficient:
$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

full.names Boolean (length = 1). Whether full or abbreviated names should be used. Defaults to FALSE. Set to TRUE for full names

Value

A named numeric vector of the requested confusion matrix metrics. Values are generally in $[0, 1]$, with the exception of "mcc" (Matthews Correlation Coefficient), which ranges from -1 to 1 where 1 indicates perfect agreement, 0 indicates chance-level performance, and -1 indicates perfect disagreement. The vector contains only the elements specified by metric (all ten metrics when metric = "all", which is the default). Names are abbreviated (e.g., "sen") when full.names = FALSE (default), or expanded (e.g., "Sensitivity") when full.names = TRUE. Any metric whose denominator is zero (e.g., sensitivity when no edges exist in base) is returned as NA.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Set split
split <- sample(
  1:nrow(basic_smallworld),
  round(nrow(basic_smallworld) / 2)
)

# Estimate networks
split1 <- network_estimation(basic_smallworld[split,])
split2 <- network_estimation(basic_smallworld[-split,])

# Estimate metrics
edge_confusion(split1, split2)
```

network_estimation *L0 Norm Regularized Network Estimation*

Description

A general function to estimate Gaussian graphical models using L0 penalty approximations. All penalties are implemented using either single-pass or full Local Linear Approximation (LLA: Fan & Li, 2001; Zou & Li, 2008)

Usage

```
network_estimation(
  data,
  n = NULL,
  corr = c("auto", "pearson", "spearman"),
  na_data = c("pairwise", "listwise"),
  penalty = c("atan", "exp", "gumbel", "log", "weibull"),
  gamma = NULL,
  adaptive = TRUE,
  nlambda = 50,
  lambda_min_ratio = 0.01,
  penalize_diagonal = TRUE,
  ic = c("AIC", "AICc", "BIC", "BIC0", "EBIC", "MBIC"),
  ebic_gamma = 0.5,
  fast = TRUE,
  LLA = FALSE,
  LLA_threshold = 0.001,
  LLA_iter = 10000,
  network_only = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data` Matrix or data frame. Should consist only of variables to be used in the analysis

`n` Numeric (length = 1). Sample size **must** be provided if `data` provided is a correlation matrix

`corr` Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options:

- `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal_categories` (see `polychoric_matrix` for more details)

- "pearson" — Pearson's correlation is computed for all variables regardless of categories
- "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into data with the sample size (n)

na_data
: Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

penalty
: Character (length = 1). Defaults to "weibull". Available options:

- "atan" — Arctangent (Wang & Zhu, 2016)

$$\lambda \cdot (\gamma + \frac{2}{\pi}) \cdot \arctan\left(\frac{|x|}{\gamma}\right)$$

- "exp" — EXP (Wang, Fan, & Zhu, 2018)

$$\lambda \cdot (1 - e^{-\frac{|x|}{\gamma}})$$

- "gumbel" — Gumbel

$$\frac{\lambda}{1 - e^{-1}} \cdot \left(e^{-e^{-\frac{|x|}{\gamma}}} - e^{-1}\right)$$

- "log" — Log (Candes, Wakin, & Boyd, 2008)

$$\frac{\lambda \cdot \log\left(1 + \frac{|x|}{\gamma}\right)}{\log\left(1 + \frac{1}{\gamma}\right)}$$

- "weibull" — Weibull

$$\lambda \cdot \left(1 - e^{-\left(\frac{|x|}{\gamma}\right)^k}\right)$$

gamma
: Numeric (length = 1). Adjusts the shape of the penalty. Defaults:

- "atan" = 0.01
- "exp" = 0.01
- "gumbel" = 0.01
- "log" = 0.10
- "weibull" = 0.01

adaptive
: Boolean (length = 1). Whether data-adaptive (gamma) parameters should be used. Defaults to TRUE. Set to FALSE to apply default gamma parameters for adaptive penalties. Available options:

- "exp"

- "gumbel"
- "weibull"

When adaptive = TRUE, gamma is set to the 10th percentile of the distribution fitted to the empirical partial correlations

nlambda
: Numeric (length = 1). Number of lambda values to test. Defaults to 50

lambda_min_ratio
: Numeric (length = 1). Ratio of lowest lambda value compared to maximal lambda. Defaults to 0.01

penalize_diagonal
: Boolean (length = 1). Should the diagonal be penalized? Defaults to TRUE

ic
: Character (length = 1). What information criterion should be used for model selection? Available options include:

- "AIC" — Akaike's information criterion: $-2L + 2E$
- "AICc" — AIC corrected: $AIC + \frac{2E^2+2E}{n-E-1}$
- "BIC" — Bayesian information criterion: $-2L + E \cdot \log{(n)}$
- "BIC0" — Bayesian information criterion not (Dicker et al., 2013): $\log{(\frac{D}{n-E})} + (\frac{\log{(n)}}{n}) \cdot E$
- "EBIC" — Extended BIC: $BIC + 4E \cdot \gamma \cdot \log{(E)}$
- "MBIC" — Modified Bayesian information criterion (Wang et al., 2018): $\log{(\frac{D}{n-E})} + (\frac{\log{(n)} \cdot E}{n}) \cdot \log{(\log{(p)})}$

Term definitions:

- n — sample size
- p — number of variables
- E — edges
- S — empirical correlation matrix
- K — estimated inverse covariance matrix (network)
- $L = \frac{n}{2} \cdot \log \text{det} K - \sum_{i=1}^{p} (SK)_{ii}$
- $D = n \cdot \sum_{i=1}^{p} (SK)_{ii} - \log \text{det} K$

Defaults to "BIC"

ebic_gamma
: Numeric (length = 1) Value to set gamma parameter in EBIC (see above). Defaults to 0.50

Only used if ic = "EBIC"

fast
: Boolean (length = 1). Whether the glassoFast version should be used to estimate the GLASSO. Defaults to TRUE.

The fast results *may* differ by less than floating point of the original GLASSO implemented by glasso and should not impact reproducibility much (set to FALSE if concerned)

LLA
: Boolean (length = 1). Should Local Linear Approximation be used to find optimal minimum? Defaults to FALSE or a single-pass approximation, which can be significantly faster (Zou & Li, 2008). Set to TRUE to find global minimum based on convergence (LLA_threshold)

LLA_threshold
: Numeric (length = 1). When performing the Local Linear Approximation, the maximum threshold until convergence is met. Defaults to 1e-03

LLA_iter — Numeric (length = 1). Maximum number of iterations to perform to reach convergence. Defaults to 10000

network_only — Boolean (length = 1). Whether the network only should be output. Defaults to TRUE. Set to FALSE to obtain all output for the network estimation method

verbose — Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... — Additional arguments to be passed on to auto.correlate

Value

When network_only = TRUE (default), returns a $p \times p$ numeric matrix of partial correlations representing the estimated Gaussian graphical model. Off-diagonal entry $[i, j]$ is the partial correlation between variables i and j controlling for all other variables, with values in $[-1, 1]$; a value of zero indicates the absence of an edge. Diagonal entries are zero. Row and column names are inherited from data.

When network_only = FALSE, returns a named list with the following elements:

- network The $p \times p$ partial correlation matrix described above
- K The $p \times p$ estimated inverse covariance (precision) matrix at the optimal lambda. Diagonal entries are the conditional precisions; off-diagonal entries are proportional to partial covariances
- R The $p \times p$ regularized covariance matrix returned by GLASSO at the optimal lambda (the w component of the GLASSO solution)
- penalty Character string naming the penalty function used (one of "atan", "exp", "gumbel", "log", "weibull")
- lambda Numeric scalar giving the regularization parameter value selected by the information criterion. Larger values correspond to sparser networks
- gamma Numeric scalar giving the shape parameter of the penalty actually used. For adaptive penalties (adaptive = TRUE), this is the data-derived value; otherwise it is the default or user-supplied value
- correlation The $p \times p$ empirical correlation matrix computed from data, used as input to the GLASSO
- criterion Character string naming the information criterion used for model selection (e.g., "bic")
- IC Numeric scalar giving the value of the information criterion at the optimal lambda

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Log penalty
Candes, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, *14*(5), 877–905.

BIC0
Dicker, L., Huang, B., & Lin, X. (2013). Variable selection and estimation with the seamless-L0 penalty. *Statistica Sinica*, *23*(2), 929–962.

Local Linear Approximation
Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, *96*(456), 1348–1360.

EXP penalty
Wang, Y., Fan, Q., & Zhu, L. (2018). Variable selection and estimation using a continuous approximation to the L0 penalty. *Annals of the Institute of Statistical Mathematics*, *70*(1), 191–214.

Atan penalty
Wang, Y., & Zhu, L. (2016). Variable selection and parameter estimation with the Atan regularization method. *Journal of Probability and Statistics*, *2016*, 1–12.

Seminal simulation in network psychometrics
Williams, D. R. (2020). Beyond lasso: A survey of nonconvex regularization in Gaussian graphical models. *PsyArXiv*.

One-step Local Linear Approximation
Zou, H., & Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, *36*(4), 1509–1533.

Examples

```
# Obtain default estimator (adaptive Weibull)
weibull_network <- network_estimation(
  data = basic_smallworld, LLA = TRUE
)

# Obtain Atan network
atan_network <- network_estimation(
  data = basic_smallworld, penalty = "atan", LLA = TRUE
)

# Obtain static EXP network
exp_network <- network_estimation(
  data = basic_smallworld, penalty = "exp",
  adaptive = FALSE, LLA = TRUE
)
```

`network_fit` *Traditional Fit Metrics for Networks*

Description

Computes several traditional fit metrics for networks including

- chi-square (χ^2)
- root mean square error of approximation (RMSEA) with confidence intervals

- confirmatory fit index (CFI)
- Tucker-Lewis index (TLI)
- standardized root mean residual (SRMR)
- log-likelihood
- Akaike's information criterion (AIC)
- Bayesian information criterion (BIC)

Usage

```
network_fit(network, n, S, ci = 0.95)
```

Arguments

`network`	Matrix or data frame. A p by p square network matrix
`n`	Numeric (length = 1). Sample size
`S`	Matrix or data frame. A p by p square zero-order correlation matrix corresponding with the input `network`
`ci`	Numeric (length = 1). Confidence interval for RMSEA. Defaults to `0.95`

Value

A named numeric vector of traditional and likelihood-based fit indices. The vector always contains the following elements:

`chisq` Chi-square statistic ($\chi^2 = n \cdot F_{ML}$), where F_{ML} is the maximum likelihood discrepancy function between the model-implied and empirical correlation matrices

`df` Degrees of freedom: total number of unique off-diagonal correlations minus the number of non-zero edges in `network`

`chisq.p.value` p-value for the chi-square test of exact fit (H0: model-implied covariance equals the population covariance)

`RMSEA` Root mean square error of approximation. Values $\leq$ 0.05 indicate close fit; values $\leq$ 0.08 indicate acceptable fit

`RMSEA.XX.lower`, `RMSEA.XX.upper` Lower and upper bounds of the `ci`-level confidence interval for RMSEA, where `XX` is the integer percentage (e.g., `RMSEA.95.lower` and `RMSEA.95.upper` for a 95% CI)

`RMSEA.p.value` p-value for the one-sided test of close fit (H0: RMSEA $\leq$ 0.05)

`CFI` Comparative fit index, comparing the target model to an independence (null) baseline. Values $\geq$ 0.95 indicate acceptable fit

`TLI` Tucker-Lewis index (non-normed fit index). Values $\geq$ 0.95 indicate acceptable fit; can fall outside $[0, 1]$ for severely misspecified models

`SRMR` Standardized root mean residual: the root mean squared difference between the model-implied and observed correlation matrices. Values $\leq$ 0.08 indicate acceptable fit

`logLik` Gaussian log-likelihood of the model-implied correlation matrix, assuming zero mean structure (means are not estimated)

`AIC` Akaike's information criterion: $-2 \cdot \log L + 2 \cdot E$, where E is the number of non-zero edges in `network`

`BIC` Bayesian information criterion: $-2 \cdot \log L + E \cdot \log(n)$

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Epskamp, S., Rhemtulla, M., & Borsboom, D. (2017). Generalized network psychometrics: Combining network and latent variable models. *Psychometrika*, *82*(4), 904–927.

Examples

```
# Obtain correlation matrix
S <- auto_correlate(basic_smallworld)

# Obtain Weibull network
weibull_network <- network_estimation(data = basic_smallworld, LLA = TRUE)

# Obtain fit (expects continuous variables!)
network_fit(network = weibull_network, n = nrow(basic_smallworld), S = S)
# Scaled metrics are not yet available for
# dichotomous or polytomous data!
```

---

polychoric_matrix   *Computes Polychoric Correlations*

Description

A fast implementation of polychoric correlations in C. Uses the Beasley-Springer-Moro algorithm (Boro & Springer, 1977; Moro, 1995) to estimate the inverse univariate normal CDF, the Drezner-Wesolosky approximation (Drezner & Wesolosky, 1990) to estimate the bivariate normal CDF, and Brent's method (Brent, 2013) for optimization of rho

Usage

```
polychoric_matrix(
  data,
  na_data = c("pairwise", "listwise"),
  empty_method = c("none", "zero", "all"),
  empty_value = c("none", "point_five", "one_over"),
  ...
)
```

Arguments

`data` — Matrix or data frame. A dataset with all ordinal values (rows = cases, columns = variables). Data are required to be between `0` and `11`. Proper adjustments should be made prior to analysis (e.g., scales from -3 to 3 in increments of 1 should be shifted by added 4 to all values)

`na_data` — Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`empty_method` — Character (length = 1). Method for empty cell correction. Available options:

- `"none"` — Adds no value (`empty_value = "none"`) to the empirical joint frequency table between two variables
- `"zero"` — Adds `empty_value` to the cells with zero in the joint frequency table between two variables
- `"all"` — Adds `empty_value` to all in the joint frequency table between two variables

`empty_value` — Character (length = 1). Value to add to the joint frequency table cells. Accepts numeric values between 0 and 1 or specific methods:

- `"none"` — Adds no value (`0`) to the empirical joint frequency table between two variables
- `"point_five"` — Adds `0.5` to the cells defined by `empty_method`
- `"one_over"` — Adds `1 / n` where *n* equals the number of cells based on `empty_method`. For `empty_method = "zero"`, *n* equals the number of *zero* cells

`...` — Not used but made available for easier argument passing

Value

A symmetric numeric matrix of dimension $p \times p$, where p is the number of variables (columns) in `data`. Each off-diagonal entry $[i, j]$ is the polychoric correlation between variables i and j — the estimated Pearson correlation between the latent continuous variables assumed to underlie the observed ordinal categories — with values in $[-1, 1]$. Diagonal entries are 1. Row and column names are inherited from `data`. If any variable has zero variance, its corresponding row and column are set to `NA` and a warning is issued.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> with assistance from GPT-4

References

Beasley-Moro-Springer algorithm

Beasley, J. D., & Springer, S. G. (1977). Algorithm AS 111: The percentage points of the normal distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *26*(1), 118-121.

Moro, B. (1995). The full monte. *Risk 8 (February)*, 57-58.

Brent optimization
Brent, R. P. (2013). Algorithms for minimization without derivatives. Mineola, NY: Dover Publications, Inc.

Drezner-Wesolowsky bivariate normal approximation
Drezner, Z., & Wesolowsky, G. O. (1990). On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation*, *35*(1-2), 101-107.

Examples

```
# Categorize data
basic_categorized <- apply(
  basic_smallworld, 2, categorize, categories = 5
)

# Compute polychoric correlation matrix
correlations <- polychoric_matrix(basic_categorized)

# Randomly assign missing data
basic_categorized[sample(1:length(basic_categorized), 1000)] <- NA

# Compute polychoric correlation matrix with pairwise missing
pairwise_correlations <- polychoric_matrix(
  basic_categorized, na_data = "pairwise"
)

# Compute polychoric correlation matrix with listwise missing
pairwise_correlations <- polychoric_matrix(
  basic_categorized, na_data = "listwise"
)
```

---

proxswap_lattice — *Construct a Degree-Preserving Ring Lattice via Proximity-Swap Construction*

Description

Converts a network matrix into a connected ring lattice whose degree sequence exactly matches the original, while maximizing the average local clustering coefficient. The resulting lattice is intended as a null model for small-world network analyses. An adjacency matrix is derived automatically from `network` (non-zero entries become edges), so the function accepts any weighted or binary network directly.

Each pass randomly permutes the degree sequence onto ring positions, then greedily assigns edges in strictly increasing ring-distance order (proximity construction). Any residual degree deficit is resolved by a swap-repair phase. Passes that yield a disconnected graph or an unsatisfied degree sequence are discarded; among valid passes the one with the highest average clustering coefficient is returned. `shuffles` independent passes are attempted in total.

Usage

```
proxswap_lattice(network, weighted = FALSE, shuffles = 100)
```

Arguments

`network`	Matrix. A square, symmetric numeric matrix representing a network (e.g., partial correlations). Non-zero off-diagonal entries are treated as edges; the binary adjacency is derived internally as `network != 0`. Isolated nodes (degree zero) are supported.
`weighted`	Logical (length = 1). Whether to return a weighted ring lattice. When `TRUE`, the edge weights from `network` are reassigned to the lattice edges according to ring distance following the implementation of Muldoon, Bridgeford, & Bassett (2016): shorter-distance lattice edges receive larger weights, preserving the overall weight distribution while concentrating stronger connections locally. When `FALSE` (default), a binary adjacency matrix is returned.
`shuffles`	Numeric (length = 1). Number of independent random permutation passes to attempt. Each pass assigns the observed degree sequence to ring positions in a new random order, runs the proximity construction, and applies swap repair if needed. Only passes producing a connected graph with zero degree error are retained; the one with the highest clustering coefficient is returned. Defaults to `100`.

Details

Algorithm

Pair precomputation. A ring distance matrix is computed as $d_{ij} = \min(|i-j|, n-|i-j|)$, giving true circular distances bounded by $\lfloor n/2 \rfloor$. All unique unordered pairs at each distance $r = 1, \ldots, \lfloor n/2 \rfloor$ are extracted once (retaining only entries where $i < j$) and cached for reuse across all passes.

Proximity construction. The observed degree sequence is randomly permuted onto ring positions. Starting from an empty graph, edges are added in distance order. At each distance band the eligible pairs (both endpoints have remaining degree budget and are not yet connected) are sorted by descending $\min(\text{budget}_i, \text{budget}_j)$ so that high-need pairs receive short connections first. Pairs are then assigned sequentially with per-pair budget re-checks. The pass exits early once all budgets reach zero. This phase is implemented in compiled C via `proximity_pass_c`.

Swap repair. If any degree deficit remains after the proximity pass, the highest-deficit node i is connected to its nearest available ring neighbour, scanning clockwise and counter-clockwise positions in interleaved distance order. When no direct partner with remaining budget exists, an edge swap is performed: a nearby unconnected node j is found, one of j's existing edges to k is removed, and a new edge (i, j) is added. Node k recovers its budget for resolution in a subsequent iteration. This repeats until all deficits are resolved or the iteration cap ($2n^2$) is reached. This phase is implemented in compiled C via `swapping_pass_c`.

Weight assignment. When `weighted = TRUE`, edge weights are reassigned after the binary topology is finalised. The observed weights are sorted by *absolute value* in descending order (largest magnitude first) and mapped onto lattice edges ranked by ascending ring distance (i.e., $d_{ij} = \min(|i-j|, n-|i-j|)$), so that shorter (more local) connections receive the largest-magnitude weights. Original signs are preserved: the sorted weight vector — not its absolute values — is assigned to the lattice edges. This directly implements the distance-weight principle of Muldoon, Bridgeford, & Bassett (2016), using the ring's structural distances rather than any network-derived proxy. Ties in ring distance are broken at random.

Pass selection. A pass is valid only if the resulting graph is connected and has zero residual degree error. Among valid passes, the one with the highest average clustering coefficient is returned.

Empirical fallback. If no valid pass is found, or if the best lattice clustering coefficient is lower than that of the original network, the empirical adjacency (or weighted matrix, if `weighted = TRUE`) is returned with a warning.

Value

A square symmetric matrix of the same dimension as `network`, with row and column names preserved, representing the resulting ring lattice. When `weighted = FALSE` (default), entries are binary integers (`0L`/`1L`); when `weighted = TRUE`, entries contain the reassigned edge weights from `network` with original signs preserved. The average clustering coefficient of the returned graph is attached as the attribute `"CC"` and can be retrieved with `attr(result, "CC")`. When the empirical fallback is triggered (see Details), `"CC"` reflects the empirical network's clustering coefficient rather than a lattice's.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Logic for weight assignments

Muldoon, S. F., Bridgeford, E. W., & Bassett, D. S. (2016). Small-world propensity and weighted brain connectivity. *Scientific Reports*, *6*, 22057.

Examples

```
# Get network
network <- network_estimation(basic_smallworld)

# Construct binary ring lattice
L <- proxswap_lattice(network)

# Retrieve the attached clustering coefficient
attr(L, "CC")

# Degree sequences should match exactly
cbind(target = colSums(network != 0), achieved = colSums(L))

# Construct weighted ring lattice
L_weighted <- proxswap_lattice(network, weighted = TRUE)

# Retrieve the attached clustering coefficient
attr(L_weighted, "CC")
```

simulate_sbm *Simulates Stochastic Block Model Data*

Description

Simulates data from a Gaussian Graphical Model (GGM) with a stochastic block model (SBM) structure. Nodes are partitioned into communities, with edge density controlled separately within and between communities via a `blocks` x `blocks` density matrix. Absolute edge weights are drawn from a Weibull distribution whose parameters are predicted from the network size, sample size, and signal-to-noise ratio using a Seemingly Unrelated Regression (SUR) model fitted to 194 empirical psychometric networks (Huth et al., 2025). The resulting population partial correlation matrix is used to generate multivariate normal (or skewed) data.

A `diffusion` parameter controls the minimum proportion of the strongest edges that are reserved for within-community positions. Because the remaining edges are shuffled randomly, the effective within-community advantage will generally exceed `1 - diffusion`; see Details.

Parameters do not have default values (except `negative_proportion`, `diffusion`, `target_condition`, `max_correlation`, and `max_iterations`) and must each be set. See Details and Examples to get started.

Usage

```
simulate_sbm(
  nodes,
  blocks,
  density_matrix,
  snr = 1,
  diffusion = 0.3,
  diffusion_range = NULL,
  negative_proportion,
  sample_size,
  skew = 0,
  skew_range = NULL,
  target_condition = 30,
  max_correlation = 0.8,
  max_iterations = 100
)
```

Arguments

`nodes` Numeric (length = 1 or `blocks`). Number of nodes per community block. Can be a single value applied to all blocks, or a vector of length `blocks` specifying each block's size individually. Minimum of three nodes per block. The total number of nodes (`sum(nodes)`) should be between 8 and 54 to remain within the range of the empirical networks used to fit the Weibull parameter model; values outside this range are accepted but will trigger extrapolation and may produce a warning from `weibull_parameters`.

blocks Numeric (length = 1). Number of community blocks.

density_matrix Matrix (dim = blocks x blocks). A symmetric numeric matrix specifying edge probabilities within and between community blocks. Diagonal entry $[i, i]$ gives the within-block edge probability for community i; off-diagonal entry $[i, j]$ ($i \neq j$) gives the between-block edge probability for the pair of communities i and j. All entries must be in $[0, 1]$ and the matrix must be symmetric (i.e., density_matrix[i,j] == density_matrix[j,i]). See Details for construction guidance.

snr Numeric (length = 1). Signal-to-noise ratio of the absolute partial correlation weights, defined as $\bar{|w|}/\text{SD}(|w|)$. This ratio governs the shape and scale of the Weibull distribution used to generate edge weights: values below 1 produce wider, more heterogeneous weight distributions; values above 1 produce narrower, more homogeneous distributions. Note that the same SNR can arise from different combinations of mean and standard deviation (e.g., mean = 0.05, SD = 0.05 and mean = 0.15, SD = 0.15 both give SNR = 1), so SNR alone does not determine the absolute magnitude of edge weights, which is additionally governed by nodes and sample_size via weibull_parameters. Empirically observed SNR values ranged from 0.648 to 1.712; values outside this range are accepted but will trigger a warning. Defaults to 1.

diffusion Numeric (length = 1). Controls the minimum proportion of the top-ranked edges (by absolute weight) that are guaranteed to be placed within communities rather than distributed freely across the network. Specifically, 1 - diffusion of the within-community edges are reserved for the highest-weight draws; the remaining edges (both within- and between-community) are filled from a randomly shuffled pool. Because shuffled edges can land within communities by chance, the actual proportion of top edges that end up within communities will on average exceed 1 - diffusion. Lower values of diffusion (e.g., 0.10) therefore produce stronger community contrast than higher values (e.g., 0.90), but diffusion should be interpreted as a floor on within-community weight concentration, not an exact control. Defaults to 0.30. Must be between 0 and 1.

diffusion_range

Numeric (length = 2). If provided, overrides diffusion by drawing the diffusion proportion uniformly from this interval on each call. Useful for introducing replication-to-replication variability in community contrast. For example, diffusion_range = c(0.05, 0.20) samples a value between 5% and 20% on each draw. The same floor interpretation applies as for diffusion. Both values must be between 0 and 1.

negative_proportion

Numeric (length = 1). Proportion of between-community edges assigned a negative sign (inhibitory connections). Each between-community edge is independently signed negative with this probability (i.e., a Bernoulli draw per edge), so the realized proportion will vary around the specified value. Must be between 0 and 1. Applies only to between-community edges; all within-community edges are positive. If not provided, a value is sampled from a truncated normal distribution reflecting the empirical distribution of true negative partial correlations across 194 psychometric networks: mean = 0.34, SD = 0.086, bounded to $[0.083, 0.55]$.

sample_size Numeric (length = 1). Number of observations to generate from the population multivariate distribution. Also influences the predicted Weibull scale parameter via weibull_parameters: larger samples are associated with smaller, more precisely estimated edge weights.

skew
Numeric (length = 1 or sum(nodes)). Skew applied to each variable after generation from the multivariate normal. Can be a single value applied to all variables, or one value per variable. Values are rounded to the nearest 0.05 increment and must be in $[-2, 2]$. Defaults to 0 (no skew).

skew_range
Numeric (length = 2). If provided, overrides skew by drawing a skew value independently and uniformly from this interval for each variable. Both values must be in $[-2, 2]$.

target_condition
Numeric (length = 1). Target condition number (computed via kappa with exact = TRUE) used when ridge regularization is required to recover a positive definite precision matrix. The smallest ridge penalty λ that brings the condition number to this target is found via root-finding (uniroot), subject to a maximum shrinkage of approximately 23% following Peeters et al. (2020). After conditioning, the Weibull bounds are re-verified on the updated edge weights; draws that fall outside the empirical bounds after conditioning are rejected. Lower values produce better-conditioned (more stable) matrices. Defaults to 30. Values up to 100 are accepted but not recommended.

max_correlation
Numeric (length = 1). Maximum allowed absolute pairwise correlation in the population correlation matrix R. Any draw where max(abs(R[lower.tri(R)])) > max_correlation is rejected and a new attempt is made. Must be between 0 and 1. Defaults to 0.80.

max_iterations
Numeric (length = 1). Maximum number of attempts to find a connected network with valid edge weights before stopping with an error. The error message reports a frequency table of rejection reasons to assist with diagnosing convergence failures. Defaults to 100.

Details

Constructing density_matrix

The density_matrix is a blocks x blocks symmetric matrix where entry $[i, i]$ gives the within-block edge probability for community i, and entry $[i, j]$ (for $i \neq j$) gives the between-block edge probability for communities i and j. The simplest construction uses a uniform off-diagonal density with block-specific diagonals:

```
# Uniform within (0.90) and between (0.20) density for 3 blocks
dm <- matrix(0.20, nrow = 3, ncol = 3)
diag(dm) <- 0.90
```

For asymmetric community structure, each diagonal entry can differ:

```
# Varying within-block density per community
dm <- matrix(0.20, nrow = 3, ncol = 3)
diag(dm) <- c(0.85, 0.90, 0.95)
```

For full pairwise control over between-block densities, specify the complete symmetric matrix directly:

```
dm <- matrix(c(
  0.90, 0.20, 0.10,
  0.20, 0.85, 0.30,
  0.10, 0.30, 0.95
), nrow = 3, ncol = 3)
```

Diffusion and within-community weight concentration

The `diffusion` parameter does not exactly fix the proportion of top-ranked edges placed within communities. Instead, `1 - diffusion` of the within-community edge slots are filled deterministically from the highest-weight draws. The remaining edge slots (within- and between-community alike) are filled from a randomly shuffled pool of lower-ranked weights, meaning some additional high-weight edges will land within communities by chance. The realized within-community weight advantage will therefore always be at least as large as implied by `1 - diffusion`, and typically larger. The `Q` field in the returned `parameters` list (Newman-Girvan modularity) provides a post-hoc summary of the actual community contrast achieved.

Value

A named list with four elements:

`data` Numeric matrix of dimension `sample_size` x `sum(nodes)` containing the simulated observations drawn from the population GGM. Rows are cases; columns are variables named `V01`, `V02`, etc. Values are continuous (or skewed continuous when `skew != 0`). To produce ordinal data, pass the columns through `categorize`.

`parameters` A list of input, derived, and estimated parameters:

- `nodes` — Integer vector of length `blocks` giving the number of nodes per block (scalar input is expanded to this length)
- `blocks` — Number of community blocks
- `sample_size` — Number of simulated observations
- `skew` — Named numeric vector of per-variable skew values actually applied (after rounding and possible resampling)
- `density_matrix` — The `blocks` x `blocks` density matrix as supplied
- `negative_proportion` — The proportion of between-community edges assigned a negative sign, either as supplied or as sampled from the empirical distribution
- `weibull` — Named numeric vector of length 2 giving the Weibull `shape` and `scale` parameters of the absolute edge weight distribution actually used. If ridge conditioning was applied, these are re-estimated from the conditioned network via MLE.
- `diffusion` — Numeric vector of length 2 giving the within-block reservation range as a proportion of within-community edges, on the internal scale used by the sampler (i.e., `range(1 - diffusion)` or `range(1 - diffusion_range)`).

Both values are equal when diffusion is scalar. Note this is the complement of the user-supplied diffusion value and represents the fraction of within-community slots filled from the top-ranked draws.

- omega — Smallworldness omega statistic of the generated network (Telesford et al., 2011). Values near zero indicate small-world structure; negative values indicate lattice-like structure; positive values indicate random-like structure (see smallworldness)
- Q — Newman-Girvan modularity of the population network (Omega) with respect to the block membership, computed via igraph::modularity on absolute edge weights. Provides a summary of the community contrast actually achieved after weight assignment and any ridge conditioning.

population Population-level network parameters:

- R — Population correlation matrix derived from the GGM via pcor2cor
- Omega — Population partial correlation matrix (the GGM edge weight matrix), with zeros for absent edges
- membership — Named integer vector of length sum(nodes) giving the community block assignment (1 to blocks) for each node

convergence Iteration and conditioning diagnostics:

- iterations — Number of sampling attempts needed to find a valid network (including graph structure and edge weight draws)
- rejections — Character vector of length max_iterations + 1 recording the rejection reason for each failed attempt; entries for successful or unused iterations are empty strings. Common reasons include disconnected graph structure, condition number exceeding target_condition, maximum correlation exceeding max_correlation, and Weibull parameters falling outside empirical bounds after ridge conditioning.
- lambda — Ridge regularization parameter λ added to the diagonal of the precision matrix to ensure positive definiteness; NA if no conditioning was required
- condition — Condition number of the final population correlation matrix R, computed via kappa with exact = TRUE

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Seminal introduction to Stochastic Block Models
Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, *5*(2), 109–137.

Empirical network data used to fit the Weibull SUR model
Huth, K. B. S., Haslbeck, J. M. B., Keetelaar, S., Van Holst, R. J., & Marsman, M. (2025). Statistical evidence in psychological networks. *Nature Human Behaviour*.

Maximum ridge shrinkage bound
Peeters, C. F., van de Wiel, M. A., & van Wieringen, W. N. (2020). The spectral condition number plot for regularization parameter evaluation. *Computational Statistics*, *35*(2), 629–646.

Examples

```
# Construct density matrix for 3 blocks with uniform densities
dm <- matrix(0.20, nrow = 3, ncol = 3)
diag(dm) <- 0.90

# Basic 3-block simulation with equal-sized communities
result <- simulate_sbm(
  nodes = 6, # 6 nodes per block = 18 total
  blocks = 3,
  sample_size = 500,
  density_matrix = dm
)

# Unequal block sizes
result <- simulate_sbm(
  nodes = c(4, 6, 8), # 18 total nodes
  blocks = 3,
  sample_size = 500,
  density_matrix = dm
)

# Varying within-block density per community
dm_varying <- matrix(0.20, nrow = 3, ncol = 3)
diag(dm_varying) <- c(0.85, 0.90, 0.95)

result <- simulate_sbm(
  nodes = 6,
  blocks = 3,
  sample_size = 500,
  density_matrix = dm_varying
)

# Full pairwise between-block density control
dm_pairwise <- matrix(c(
  0.90, 0.20, 0.10,
  0.20, 0.85, 0.30,
  0.10, 0.30, 0.95
), nrow = 3, ncol = 3)

result <- simulate_sbm(
  nodes = 6,
  blocks = 3,
  sample_size = 500,
  density_matrix = dm_pairwise
)

# Fix the proportion of negative between-community edges
result <- simulate_sbm(
  nodes = 6,
  blocks = 3,
  sample_size = 500,
  density_matrix = dm,
```

```
  negative_proportion = 0.20
)

# Introduce variability in diffusion across replications
result <- simulate_sbm(
  nodes = 6,
  blocks = 3,
  sample_size = 500,
  density_matrix = dm,
  diffusion_range = c(0.30, 0.70)
)
```

simulate_smallworld *Simulates Small-World GGM Data*

Description

Simulates data from a Gaussian Graphical Model (GGM) with a small-world network structure. The generative process proceeds in three stages. First, a ring lattice is constructed with `neighbors + 1` nearest-neighbor connections per node, where `neighbors` is derived from `nodes` and `density`. Second, the lattice is randomly pruned to the target `density`, introducing degree heterogeneity from the outset. Third, edges are rewired with probability `rewire`, where rewired edges are placed preferentially on node pairs with higher combined degree (degree- weighted rewiring). This approach produces more realistic degree distributions than standard Watts-Strogatz rewiring while preserving the local clustering structure of the lattice. Edge weights are assigned by structural priority: edges retained from the pruned lattice receive larger partial correlation weights based on their original neighbor distance, grounded in the empirical observation that shorter-distance connections tend to carry stronger weights in psychometric networks. The resulting network is used to generate multivariate normal (or skewed) data. Parameters do not have default values (except `negative_proportion`, `snr`, `target_condition`, `max_correlation`, and `max_iterations`) and must each be set. See Details and Examples to get started.

Usage

```
simulate_smallworld(
  nodes,
  density,
  rewire,
  snr = 1,
  negative_proportion,
  sample_size,
  skew = 0,
  skew_range = NULL,
  target_condition = 30,
  max_correlation = 0.8,
  max_iterations = 100
)
```

Arguments

`nodes` Numeric (length = 1). Number of nodes in the network. Minimum of three nodes. The total number of nodes should be between 8 and 54 to remain within the range of the empirical networks used to fit the Weibull parameter model; values outside this range are accepted but will trigger extrapolation and may produce a warning from `weibull_parameters`.

`density` Numeric (length = 1). Target edge density of the network after pruning the initial ring lattice. Controls the number of nearest neighbors `k` each node is connected to via $k = \text{round}((\text{nodes} \times \frac{\text{nodes}-1}{2} \times \text{density})/\text{nodes})$, subject to a minimum of 2 and a maximum of $\lfloor(\text{nodes} - 1)/2\rfloor$. A ring lattice with `neighbors + 1` connections per node is first generated and then pruned to this density, introducing degree heterogeneity before rewiring. Must be between 0 and 1. A minimum density sufficient to maintain a connected graph is enforced; values below this threshold will produce an informative error.

`rewire` Numeric (length = 1). Probability of rewiring each edge. Unlike the standard Watts-Strogatz model, rewired edges are placed using degree-weighted random selection: node pairs with higher combined degree have a greater probability of receiving a rewired edge (see Details). Values near 0 preserve the pruned lattice structure; values near 1 produce approximately random networks. The small-world regime typically occurs at intermediate values (roughly 0.01 to 0.30). Must be between 0 and 1.

`snr` Numeric (length = 1). Signal-to-noise ratio of the absolute partial correlation weights, defined as $|\bar{w}|/\text{SD}(|w|)$. Values less than 1 produce wider, more heterogeneous weight distributions; values greater than 1 produce narrower, more homogeneous distributions. Empirically observed SNR values ranged from 0.648 to 1.712; values outside this range are accepted but will trigger a warning. Defaults to `1`.

`negative_proportion`
Numeric (length = 1). Proportion of edges that are negative (inhibitory). Must be between 0 and 0.50. The upper bound of 0.50 is a mathematical constraint of the sign-flipping procedure used to assign negative edges (see Details). If not provided, a value is sampled from a truncated normal distribution reflecting the empirical distribution of true negative partial correlations across 194 psychometric networks: mean = 0.34, SD = 0.086, bounded to $[0.083, 0.50]$.

`sample_size` Numeric (length = 1). Number of observations to generate from the population multivariate normal distribution. Also influences the predicted Weibull scale parameter via `weibull_parameters`: larger samples are associated with smaller, more precisely estimated edge weights.

`skew` Numeric (length = 1 or `nodes`). Skew applied to each variable after generation from the multivariate normal. Can be a single value (applied to all variables) or one value per variable. Values are rounded to the nearest increment of 0.05 in the range $[-2, 2]$. Defaults to `0` (no skew).

`skew_range` Numeric (length = 2). If provided, a skew value is drawn uniformly from this range for each variable, overriding `skew`. Both values must be between -2 and 2.

`target_condition`
Numeric (length = 1). Target condition number (using `kappa` with `exact = TRUE`) applied when ridge regularization is needed to recover a positive defi-

nite precision matrix. The smallest ridge penalty λ that brings the condition number to this target is found via root-finding (uniroot), subject to a maximum shrinkage of approximately 23% following Peeters et al. (2020). Lower values produce better-conditioned matrices. Defaults to `30`. Values up to `100` are accepted but not recommended.

`max_correlation`
: Numeric (length = 1). Maximum allowed absolute pairwise correlation in the population correlation matrix `R`. Any draw where `max(abs(R[lower.tri(R)]))` > `max_correlation` is rejected and a new attempt is made. Must be between 0 and 1. Defaults to `0.80`.

`max_iterations`
: Numeric (length = 1). Maximum number of attempts to find (1) a connected network structure, (2) a network satisfying the small-world screening criterion, and (3) a valid set of edge weights. The error message reports a frequency table of rejection reasons to assist with diagnosing convergence failures. Defaults to `100`.

Details

Lattice generation and pruning

The generative process begins by constructing a ring lattice with `neighbors + 1` nearest-neighbor connections per node via sample_smallworld with `p = 0`. The `+1` overshoot ensures the lattice always has more edges than the target density, guaranteeing that pruning can proceed. The lattice is then randomly pruned to the target `density` by removing edges uniformly at random, subject to the constraint that the pruned graph remains connected. Because removal is random, different nodes lose different numbers of edges, producing a heterogeneous degree distribution before any rewiring occurs. This heterogeneity provides a non-uniform prior for the subsequent degree-weighted rewiring step.

Degree-weighted rewiring

Each edge in the pruned lattice is independently selected for rewiring with probability `rewire`. For each selected edge (i, j), node i is kept fixed and the j endpoint is redirected to a new target node. Valid targets are restricted to node pairs involving node i that (1) are not currently connected, and (2) have never been occupied during the current rewiring pass (i.e., were absent in the original pruned lattice). Among valid targets, the new endpoint is selected with probability proportional to $\sqrt{d_i + d_k}$, where d_i and d_k are the current degrees of the two nodes in the candidate pair. The square-root transformation moderates the rich-get-richer tendency of linear preferential attachment, producing degree heterogeneity consistent with the empirical range of psychometric networks without generating extreme hubs. Node degrees are updated incrementally after each rewire so that subsequent rewiring steps reflect the current graph state.

Edge weight assignment

Edge weights are assigned by ranking edges according to their distance in the pruned lattice before rewiring. Edges retained from the pruned lattice receive their original neighbor distance (1 = nearest neighbor, 2 = second nearest, etc.); rewired edges are assigned a distance of `neighbors + 1`, placing them at the bottom of the priority order. Absolute edge weights are drawn from a Weibull distribution whose parameters are predicted from the network size, sample size, and signal- to-noise ratio using a Seemingly Unrelated Regression (SUR) model fitted to 194 empirical psychometric networks (Huth et al., 2025). The sorted (descending) Weibull weights are then mapped onto the distance ranking so that the largest weights go to the shortest-distance edges. This assignment is grounded in the empirical observation that shorter-distance local connections tend to carry larger partial correlation weights in psychometric networks.

Negative edge assignment

Negative edges are introduced by flipping the signs of a subset of nodes — multiplying all edges incident to those nodes by -1. The number of nodes to flip is derived from `negative_proportion` via the inversion formula $(1 - \sqrt{1 - 2p})/2$, where p is the target proportion of negative edges. This formula assumes that an edge is negative if and only if exactly one of its endpoints is flipped, giving an expected negative proportion of $2 \cdot (k/n) \cdot (1 - k/n)$ for k flipped nodes out of n total. The formula requires $p \leq 0.50$, which is why `negative_proportion` is bounded at 0.50.

Value

A named list with four elements:

`data` Numeric matrix of dimension `sample_size` x `nodes` containing the simulated observations drawn from the population GGM. Rows are cases; columns are variables named `V01`, `V02`, etc. Values are continuous (or skewed continuous when `skew != 0`). To produce ordinal data, pass the columns through `categorize`.

`parameters` A list of input, derived, and estimated parameters:

- `nodes` — Number of nodes (as supplied)
- `density` — Target edge density (as supplied)
- `neighbors` — Number of nearest neighbors `k` derived from `nodes` and `density`; the initial ring lattice uses `neighbors + 1` connections before pruning
- `rewire` — Edge rewiring probability (as supplied)
- `negative_proportion` — Proportion of negative edges, either as supplied or as sampled from the empirical distribution
- `sample_size` — Number of simulated observations
- `skew` — Named numeric vector of per-variable skew values actually applied (after rounding and possible resampling)
- `weibull` — Named numeric vector of length 2 giving the Weibull `shape` and `scale` parameters of the absolute edge weight distribution actually used. If ridge conditioning was applied, these are re-estimated from the conditioned network via MLE.
- `omega` — Smallworldness omega statistic of the generated network (Telesford et al., 2011). Values near zero indicate small-world structure; negative values indicate lattice-like structure; positive values indicate random-like structure (see `smallworldness`)
- `Q` — Newman-Girvan modularity of the population network (`Omega`) computed via `igraph::modularity` on absolute edge weights, using the community partition returned by `igraph::cluster_leiden` with `objective_function = "modularity"`. Because `cluster_leiden` is a heuristic algorithm, `Q` reflects a high-quality but not necessarily optimal partition. Values near zero indicate absence of community structure, consistent with the network theory of psychopathology.

`population` Population-level network parameters:

- `R` — Population correlation matrix derived from the GGM via `pcor2cor`
- `Omega` — Population partial correlation matrix (the GGM edge weight matrix), with zeros for absent edges

`convergence` Iteration and conditioning diagnostics:

- `iterations` — Number of sampling attempts needed to find a valid network
- `rejections` — Character vector recording the rejection reason for each failed attempt. Common reasons include disconnected graph structure, omega exceeding the small-world threshold, condition number exceeding `target_condition`, and maximum correlation exceeding `max_correlation`
- `lambda` — Ridge regularization parameter λ added to the diagonal of the precision matrix to ensure positive definiteness; `NA` if no conditioning was required
- `condition` — Condition number of the final population correlation matrix `R`

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Seminal introduction to the Watts-Strogatz small-world model
Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440–442.

Logic for weight assignments
Muldoon, S. F., Bridgeford, E. W., & Bassett, D. S. (2016). Small-world propensity and weighted brain networks. *Scientific Reports*, *6*(1), 22057.

Analytical approximation of random-graph average path length
Newman, M. E. J., Strogatz, S. H., & Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, *64*(2), 026118.

Empirical network data used to fit the Weibull SUR model
Huth, K. B. S., Haslbeck, J. M. B., Keetelaar, S., Van Holst, R. J., & Marsman, M. (2025). Statistical evidence in psychological networks. *Nature Human Behaviour*.

Maximum ridge shrinkage bound
Peeters, C. F., van de Wiel, M. A., & van Wieringen, W. N. (2020). The spectral condition number plot for regularization parameter evaluation. *Computational Statistics*, *35*(2), 629–646.

Examples

```
# Basic small-world network (moderate density, moderate rewiring)
result <- simulate_smallworld(
  nodes = 20,
  density = 0.30,
  rewire = 0.20,
  sample_size = 500
)
```

```
# Lattice-like structure (low rewiring preserves local connectivity)
result <- simulate_smallworld(
  nodes = 20,
  density = 0.30,
  rewire = 0.01,
  sample_size = 500
)

# Random-like structure (high rewiring destroys lattice regularity)
result <- simulate_smallworld(
  nodes = 20,
  density = 0.30,
  rewire = 0.80,
  sample_size = 500
)

# Fix the proportion of negative edges
result <- simulate_smallworld(
  nodes = 20,
  density = 0.30,
  rewire = 0.20,
  sample_size = 500,
  negative_proportion = 0.20
)
```

---

skew_tables *Skew Tables*

Description

Tables for skew based on the number of categories (2, 3, 4, 5, or 6) in the data

Usage

```
data(skew_tables)
```

Format

A list (length = 5)

Examples

```
data("skew_tables")
```

`smallworldness` *Computes Various Small-Worldness Metrics*

Description

Computes the small-worldness of a network using one of five methods. All simulation-based methods generate degree-preserving random graphs as the null baseline. The lattice reference network (where required) is constructed using `proxswap_lattice`, which produces a degree-preserving ring lattice that maximizes the clustering coefficient

Usage

```
smallworldness(
  network,
  lattice,
  method = c("analytical", "omega", "S", "SWI", "SWP"),
  weighted = FALSE,
  iter = 100
)
```

Arguments

`network` Matrix or data frame. A square, symmetric numeric matrix representing a network (e.g., partial correlations). Absolute values are taken internally, so signed weights are handled automatically. Non-zero off-diagonal entries are treated as edges.

`lattice` Matrix (optional). A pre-computed lattice adjacency matrix to use as the regular-network reference for the `"omega"` and `"SWP"` methods. If not provided, a lattice is generated automatically via `proxswap_lattice`. Ignored by `"analytical"` and `"S"`

`method` Character (length = 1). The method used to compute small-worldness. Defaults to `"SWP"`. Available options:

- `"analytical"` — Computes the Humphries & Gurney (2008) S metric using closed-form approximations for the Erdős–Rényi random graph baseline:
$$S = \frac{C/C_{\text{rand}}}{L/L_{\text{rand}}}$$
where $C_{\text{rand}} \approx \langle k \rangle / n$ and $L_{\text{rand}} \approx \ln(n)/\ln(\langle k \rangle)$. No random graphs are generated. The `weighted` argument is ignored. Values greater than 1 indicate small-world structure
- `"S"` — Computes the Humphries & Gurney (2008) S metric by simulation. Random graphs are generated under the Erdős–Rényi model (`sample_gnm`) with the same number of nodes and edges as `network`. The empirical and random clustering coefficients both use global transitivity (C^{Δ}). When `weighted = TRUE`, edge weights are reassigned to each random graph topology via `assign_weights`. Values greater than 1 indicate small-world structure

- "omega" — Computes the ω metric of Telesford et al. (2011):

$$\omega = \frac{L_{\text{rand}}}{L} - \frac{C}{C_{\text{latt}}}$$

Random graphs preserve the degree sequence via edge-switching (sample_degseq, method = "edge.switching.simple"). The clustering coefficient uses average local transitivity ($\bar{C}$). Note: Telesford et al. (2011) defined and validated ω exclusively on binary, unweighted networks. When weighted = TRUE, edge weights are reassigned to each random graph topology via assign_weights and the lattice is constructed with weighted = TRUE, following the approach of Muldoon et al. (2016); this is an extension beyond the original formulation. Values near zero indicate small-world structure; negative values indicate lattice-like structure; positive values indicate random-like structure. Bounded in $[-1, 1]$

- "SWI" — Computes the Small-World Index of Neal (2015):

$$\text{SWI} = \frac{L - L_{\text{latt}}}{L_{\text{rand}} - L_{\text{latt}}} \times \frac{C - C_{\text{rand}}}{C_{\text{latt}} - C_{\text{rand}}}$$

Each term captures where the observed network's path length and clustering coefficient fall within the lattice-to-random range, so that SWI = 1 only when $L = L_{\text{rand}}$ and $C = C_{\text{latt}}$ simultaneously. Because this ideal is mathematically unachievable in a finite network, SWI = 1 is a conceptual upper bound rather than a realisable value. Random graphs preserve the degree sequence via edge-switching (sample_degseq, method = "edge.switching.simple"). The clustering coefficient uses average local transitivity ($\bar{C}$). When weighted = TRUE, edge weights are reassigned to each random graph topology and the lattice is constructed with weighted = TRUE. Note: SWI and SWP were developed independently and concurrently; both apply the same double-normalisation framework but differ in how the two deviation terms are combined (product vs. Euclidean distance). Values close to 1 indicate strong small-world structure; values near 0 indicate lattice-like or random-like structure. Bounded in $[0, 1]$

- "SWP" — Computes the Small-World Propensity of Muldoon et al. (2016):

$$\phi = 1 - \sqrt{\frac{\Delta_C^2 + \Delta_L^2}{2}}$$

where $\Delta_C = (C_{\text{latt}} - C)/(C_{\text{latt}} - C_{\text{rand}})$ and $\Delta_L = (L - L_{\text{rand}})/(L_{\text{latt}} - L_{\text{rand}})$. Both Δ_C and Δ_L are bounded to $[0, 1]$, guaranteeing $\phi \in [0, 1]$. Random graphs preserve the degree sequence via edge-switching. The clustering coefficient uses average local transitivity ($\bar{C}$). When weighted = TRUE, edge weights are reassigned to each random graph topology via assign_weights and the lattice is constructed with weighted = TRUE. Values close to 1 indicate strong small-world structure; a pragmatic threshold of $\phi_T = 0.60$ has been suggested

weighted — Logical (length = 1). Whether to compute small-worldness on the weighted network. When TRUE, edge weights from network are preserved for the empirical graph and reassigned to random and lattice graph topologies via assign_weights,

	following Muldoon et al. (2016). When `FALSE` (default), all graphs are treated as binary. Ignored when `method = "analytical"`
`iter`	Numeric (length = 1). Number of random graphs to generate when estimating the null baseline. Defaults to `100`. Ignored when `method = "analytical"`. Higher values produce more stable estimates at the cost of computation time

Value

Numeric (length = 1). The small-worldness value computed using the chosen `method`:

- `"analytical"` and `"S"` — $S > 1$ indicates small-world structure
- `"omega"` — Values near $|\omega| < 0.50$ indicate small-world structure (ranges between -1 and 1)
- `"SWI"` — Values close to 1 indicate strong small-world structure; values near 0 indicate lattice-like or random-like structure (ranges between 0 and 1)
- `"SWP"` — Values $\phi > 0.60$ indicate strong small-world structure (ranges between 0 and 1)

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

omega

Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H., & Laurienti, P. J. (2011). The ubiquity of small-world networks. *Brain Connectivity*, *1*(5), 367–375.

S and analytical

Humphries, M. D., & Gurney, K. (2008). Network 'small-world-ness': A quantitative method for determining canonical network equivalence. *PLoS ONE*, *3*(4), e0002051.

SWI

Neal, Z. P. (2015). Making big communities small: Using network science to understand the ecological and behavioral requirements for community social capital. *American Journal of Community Psychology*, *55*(3), 369–380.

SWP

Muldoon, S. F., Bridgeford, E. W., & Bassett, D. S. (2016). Small-world propensity and weighted brain networks. *Scientific Reports*, *6*(1), 22057.

Examples

```
# Get network
network <- network_estimation(basic_smallworld)

# Compute SWP (default)
swp <- smallworldness(network)

# Compute omega
omega <- smallworldness(network, method = "omega")

# Compute analytical S
```

```
S <- smallworldness(network, method = "analytical")

# Compute simulated S
S_sim <- smallworldness(network, method = "S")

# Compute SWI
swi <- smallworldness(network, method = "SWI")

# Compute weighted SWP
swp_w <- smallworldness(network, weighted = TRUE)

# Compute weighted omega
omega_w <- smallworldness(network, method = "omega", weighted = TRUE)
```

---

weibull_parameters *Predict Weibull Parameters for Edge Weight Distributions*

Description

Predicts the shape and scale parameters of a Weibull distribution that characterizes the absolute partial correlation edge weights of a psychometric network, given its number of nodes and sample size. Parameter estimates are derived from a Seemingly Unrelated Regression (SUR) model fitted to empirical network data from Huth et al. (2025), where absolute partial correlations were found to follow a Weibull distribution more consistently than Beta, Gamma, or log-normal alternatives.

Usage

```
weibull_parameters(nodes, sample_size, snr = 1, bootstrap = FALSE)
```

Arguments

nodes	Integer. The number of nodes (variables) in the network. Must be between 8 and 54, reflecting the range of the empirical networks used to fit the underlying SUR model.
sample_size	Integer. The sample size of the dataset from which the network is estimated.
snr	Numeric (length = 1). Signal-to-noise ratio of partial correlations ($|\bar{w}|/\mathrm{SD}(|w|)$). Values less than 1 indicate wider range of partial correlations (w) whereas values greater than 1 indicate narrower range. Defaults to `1` where the mean of the partial correlations ($|\bar{w}|$) is equal to the standard deviation ($\mathrm{SD}(|w|)$)
bootstrap	Logical. If `TRUE`, a randomly sampled residual from the SUR model fit is added to each predicted parameter, introducing empirically grounded variability suitable for use in simulation or bootstrapping contexts. Defaults to `FALSE`.

Details

The shape and scale parameters are predicted from derived network descriptors that differ between the two SUR equations:

- `rlp` The reciprocal log of the number of nodes, $1/\log(p)$, capturing the diminishing marginal effect of network size on edge weight distributions. Used in both the shape and scale equations.
- `scaling` The standard error of partial correlations defined as $\sqrt{1/(n-p)}$, where n is the sample size and p is the number of nodes. Larger values indicate greater sampling uncertainty in the partial correlation estimates. Used in the scale equation only.

The two SUR equations have an asymmetric structure reflecting different theoretical roles for sampling precision. Shape — which governs the concentration of the edge weight distribution — is determined solely by signal characteristics of the network via `snr` and `rlp`. Scale — which governs the typical magnitude of edge weights — additionally depends on `scaling`, as the expected size of partial correlations is directly affected by estimation precision. This asymmetry is both empirically supported (dropping `scaling` from the shape equation costs $\Delta R^2 < 0.006$) and theoretically coherent.

These predictors enter two SUR equations whose coefficients are stored in the internal `weibull_weights` dataset. SUR was used to account for the correlated residuals between the shape and scale equations across networks (residual correlation = 0.261). Model fit was strong: the shape equation achieved $R^2 = 0.887$ (RMSE = 0.048) and the scale equation achieved $R^2 = 0.885$ (RMSE = 0.011). Shape residuals were normally distributed (Shapiro-Wilk W = 0.990, p = 0.173). Scale residuals showed a modest departure from normality (Shapiro-Wilk W = 0.973, p < 0.001), consistent with slight right skew in the scale outcome and test sensitivity at n = 194 rather than a substantive violation. Heteroskedasticity was detected in both the shape equation (Breusch-Pagan p < 0.001) and the scale equation (Breusch-Pagan p < 0.001); robust standard errors (HC3) were used for inference. Multicollinearity among predictors in the scale equation was negligible (VIF $\leq$ 1.60); the shape equation contains only two predictors with no multicollinearity concern.

`nodes` influences predicted shape and scale via `rlp`. `sample_size` influences predicted scale via `scaling`, and both parameters via their joint contribution to `snr` when it is estimated from data rather than supplied directly.

Empirically, shape values ranged from approximately 0.72 to 1.63 (M = 1.07, SD = 0.14) and scale values from approximately 0.03 to 0.19 (M = 0.10, SD = 0.03) across the 194 networks used to fit the model. Shape values near 1 indicate approximately exponential edge weight distributions; values above 1 indicate a rising hazard (mode-bearing distribution).

When `bootstrap = TRUE`, residuals are drawn via `shuffle()` – a random sampling without replacement – from the empirical SUR residuals, preserving the observed marginal residual distribution.

Value

A named numeric vector of length 2:

- `shape` The predicted Weibull shape parameter ($k > 0$).
- `scale` The predicted Weibull scale parameter ($\lambda > 0$).

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Huth, K. B. S., Haslbeck, J. M. B., Keetelaar, S., Van Holst, R. J., & Marsman, M. (2025). Statistical evidence in psychological networks. *Nature Human Behaviour*.

Examples

```
# Predict parameters for a 10-node network with n = 500
weibull_parameters(nodes = 10, sample_size = 500)

# With bootstrapped residuals for use in simulation
weibull_parameters(nodes = 10, sample_size = 500, bootstrap = TRUE)
```

---

weibull_weights *SUR Model Coefficients and Residuals for Weibull Parameter Prediction*

Description

A named list encoding the Seemingly Unrelated Regression (SUR) model fitted to Weibull shape and scale parameters derived from 194 empirical networks. The object is consumed internally by weibull_parameters to generate data-driven Weibull parameter estimates given a network's number of nodes and sample size.

Usage

```
data(weibull_weights)
```

Format

A named list with two elements, `shape` and `scale`, each containing:

`coefficients` A named numeric vector of regression coefficients from the SUR model. The equations are asymmetric: the shape equation includes an intercept and two predictors (`snr`, `rlp`); the scale equation includes an intercept and three predictors (`snr`, `rlp`, `scaling`). Predictors are defined as follows:

- `snr` Signal-to-noise ratio of the absolute partial correlations, computed as the mean divided by the standard deviation of the absolute edge weights. Used in both equations.
- `rlp` Reciprocal log of node count, $1/\log(p)$, where p is the number of nodes. Used in both equations.
- `scaling` Standard error of partial correlations, $\sqrt{1/(n-p)}$, where n is the sample size and p is the number of nodes. Used in the scale equation only.

`residuals` A numeric vector of residuals from the fitted SUR equation, used by weibull_parameters to introduce empirically grounded variability when `bootstrap = TRUE`.

Details

Absolute partial correlations from 222 deduplicated empirical networks (Huth et al., 2025) were fitted to Beta, Gamma, log-normal, and Weibull distributions via maximum likelihood. Weibull provided the best fit most consistently: it outperformed each alternative by more than 2 log-likelihood units far more often than the reverse (vs. Beta: 56–0; vs. Gamma: 15–2; vs. log-normal: 155–13; vs. Exponential: 54–0).

The resulting Weibull shape and scale parameters were then jointly modelled as a function of network descriptors using Seemingly Unrelated Regression (`systemfit`), which accounts for correlated residuals between the shape and scale equations across networks (residual correlation = 0.261). Prior to fitting, networks with fewer than eight nodes ($p < 8$), more than 300,000 observations, or fewer than one observation per edge (ope ≤ 1) were excluded, as Huth et al. (2025) demonstrated that networks in this regime show the weakest statistical evidence for edge presence or absence, yielding unstable parameter estimates (n = 28 excluded). This left n = 194 networks for analysis. Shape and scale parameters were modelled on their original scales.

The two equations have an asymmetric structure. Shape — which governs the concentration of the edge weight distribution — is predicted from `snr` and `rlp` only, reflecting that the shape of the distribution is a property of the network's signal structure independent of sampling precision. Scale — which governs typical edge weight magnitude — additionally includes `scaling`, as the expected size of partial correlations is directly affected by estimation precision. Dropping `scaling` from the shape equation costs $\Delta R^2 < 0.006$ and yields cleaner inference; all shape predictors are significant under HC3-robust standard errors (both $p < 0.001$).

Variance inflation factors for the scale equation (≤ 1.60) confirmed the absence of problematic multicollinearity; the shape equation contains only two predictors with no multicollinearity concern. Breusch-Pagan tests indicated statistically significant heteroskedasticity in both equations; however, all predictors remained significant under HC3-robust standard errors, indicating no material effect on inference. Shape residuals were normally distributed (Shapiro-Wilk W = 0.990, p = 0.173). Scale residuals showed a modest departure from normality (Shapiro-Wilk W = 0.973, p < 0.001), consistent with slight right skew in the scale outcome and test sensitivity at n = 194 rather than a substantive violation, as confirmed by visual inspection of the residual histogram. Model fit was strong: shape $R^2 = 0.887$ (RMSE = 0.048); scale $R^2 = 0.885$ (RMSE = 0.011).

References

Huth, K. B. S., Haslbeck, J. M. B., Keetelaar, S., Van Holst, R. J., & Marsman, M. (2025). Statistical evidence in psychological networks. *Nature Human Behaviour*.

Examples

```
data("weibull_weights")

# Inspect SUR coefficients for each equation
weibull_weights$shape$coefficients
weibull_weights$scale$coefficients

# Predict Weibull parameters for a new network
weibull_parameters(nodes = 12, sample_size = 500)
```

Index

∗ **datasets**
basic_smallworld, 4
skew_tables, 31
weibull_weights, 37

auto_correlate, 3

basic_smallworld, 4

categorize, 5, *23*, *29*
cor, *3, 4*
cut, *5*

edge_confusion, 6

glasso, *11*
glassoFast, *11*

kappa, *22*, *27*

L0ggm *(*L0ggm-package*)*, 2
L0ggm-package, 2

nearPD, *4*
network_estimation, 9
network_fit, 13

polychoric_matrix, *3, 4*, *9*, 15
proxswap_lattice, 17, *32*

sample_degseq, *33*
sample_gnm, *32*
sample_smallworld, *28*
simulate_sbm, 20
simulate_smallworld, 26
skew_tables, 31
smallworldness, *24*, *29*, 32

uniroot, *22*, *28*

weibull_parameters, *20–22*, *27*, 35, *37*
weibull_weights, 37