Package 'psychnets'

November 18, 2026

Title Tidy Clean-Room Psychological Network Modeling

Version 0.8.2

Description Provides clean-room implementations for estimating psychometric network models, including correlation and partial-correlation networks, Gaussian graphical models with extended Bayesian information criterion (EBIC) regularization, nonparanormal and stepwise selection variants, information-filtering networks (the triangulated maximally filtered graph and the local-global inverse covariance), relative-importance networks, Ising and mixed graphical models, and the panel graphical vector-autoregressive model (Epskamp, Borsboom and Fried (2018) <doi:10.3758/s13428-017-0862-1>; Saqr, Beck and Lopez-Pernas (2024) <doi:10.1007/978-3-031-54464-4_19>; Epskamp (2020) <doi:10.1007/s11336-020-09697-3>). All methods are implemented from first principles in base R without compiled dependencies and return consistent, tidy outputs. Functions are designed to be transparent and report optimization diagnostics where applicable. For Gaussian graphical models, the graphical lasso stationarity (Karush-Kuhn-Tucker) residual quantifies the deviation of the estimated solution from the optimum of the corresponding convex optimization problem.

License GPL-3

URL https://pak.dynasite.org/psychnets, https://github.com/mohsaqr/psychnets

BugReports https://github.com/mohsaqr/psychnets/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports grDevices, graphics, parallel, stats

Suggests cocor, cograph, glasso, glmnet, igraph, IsingFit, knitr, mgm, mlVAR, mvtnorm, networktools, psych, psychonetrics, qgraph, rmarkdown, testthat (>= 3.0.0), tna

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Mohammed Saqr [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-5881-3109>),
Sonsoles López-Pernas [aut] (ORCID:
<https://orcid.org/0000-0002-9621-1392>)

Maintainer Mohammed Saqr <saqr@saqr.me>

Repository CRAN

Date/Publication 2026-11-18 11:00:02 UTC

Contents

certificate . . . 3
condition . . . 4
cor_auto . . . 6
cor_network . . . 8
dichotomize . . . 10
difference_test . . . 11
ebic_glasso . . . 13
event_frequencies . . . 16
ggm_modselect . . . 18
ggm_support_kkt . . . 20
glasso_kkt . . . 22
glm_lasso_kkt . . . 24
huge_network . . . 26
ising_fit . . . 28
ising_sampler . . . 31
lmg_certificate . . . 33
logo_network . . . 34
mgm_fit . . . 36
net_aggregate . . . 39
net_boot . . . 41
net_bridge . . . 45
net_casedrop_reliability . . . 47
net_centralities . . . 50
net_clustering . . . 52
net_compare . . . 54
net_crosswalk . . . 57
net_edge_betweenness . . . 59
net_predict . . . 60
net_smallworld . . . 61
net_split_reliability . . . 63
node_predictability . . . 65
panel_gvar . . . 66
pcor_network . . . 69
plot.psychnet . . . 72

plot.psychnet_stability 74
psychnet 77
psychnet_group 80
psychnet_multilevel 82
psychnet_panelgvar 84
psychnet_result_group 86
redundancy 87
relimp_network 89
SRL 91
srl_panel 92
tmfg_certificate 93
tmfg_network 95

Index **97**

`certificate` *Check that a fitted network solves its estimator*

Description

`certificate()` measures how far a fitted network is from the solution of its estimator. For a regularized or constrained estimator such as the graphical lasso, this is the largest violation of the optimality (Karush–Kuhn–Tucker) conditions; a value near zero means the network is the optimum, up to numerical precision.

Such an estimator defines the network as the solution of an optimization problem; the graphical lasso, for example, maximizes a penalized likelihood. Any solution must satisfy the problem's optimality conditions, so whether a returned network is that solution can be checked without a second solver; the amount by which the network violates the conditions is its certificate. The information-filtering (TMFG) and relative-importance networks are not optima of an objective; for them the certificate checks the identity that defines the graph. A network changed after fitting, by a threshold or by pruning, is no longer the solution that was checked, and its certificate is reported as missing.

`certificate()` returns the value together with the kind of certificate and whether it falls below a tolerance.

Correlation and partial-correlation networks involve no optimization and have no certificate. To check a precision matrix or regression coefficients that are not stored in a fitted network, call `glasso_kkt()`, `ggm_support_kkt()` or `glm_lasso_kkt()` directly.

Usage

```
certificate(x, tol = 1e-06)
```

Arguments

`x`	A fitted network, an object of class `psychnet` (psychnet).
`tol`	Tolerance at or below which the network is reported as certified: a single finite number >= 0. Default `1e-6`.

Details

The optimality certificates are computed by glasso_kkt() (graphical lasso networks), ggm_support_kkt() (unregularized networks restricted to a graph) and glm_lasso_kkt() (nodewise Ising and mixed graphical models). The structural certificates are tmfg_certificate(), the number of violated TMFG properties (0 to 4), and lmg_certificate(), the largest difference between the summed importance shares of a node's predictors and that node's explained variance.

Value

A data frame with one row and columns method (the estimator), certificate (the violation; smaller is better), kind ("kkt" for an optimality certificate, "structural" for TMFG and relative importance, "none" for correlation and partial-correlation networks) and certified (TRUE when certificate is at or below tol). For a network changed after fitting, or a fit that did not converge, certificate and certified are NA. For correlation and partial-correlation networks certificate is NA and certified is TRUE, because no optimization claim is made.

Conditions

psychnet_bad_input when x is not a psychnet network or tol is not a single finite number >= 0.

See Also

glasso_kkt(), ggm_support_kkt(), glm_lasso_kkt(), tmfg_certificate() and lmg_certificate() for the individual checks; psychnet() for the estimators.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
certificate(ebic_glasso(cor_matrix = S, n = 250))
certificate(tmfg_network(cor_matrix = S))
```

---

condition *Read a moderated mixed graphical model at a value of the moderator*

Description

condition() reads a moderated mixed graphical model at one value of the moderator and returns that network as an ordinary psychnet; as.data.frame() and print() read the moderated fit itself. In a moderated network model, fitted by mgm_fit() with moderators, the strength of an edge between two variables can depend on the value of a third variable, the moderator, so each pairwise association has a main effect and an interaction with the moderator.

The network at a given moderator value, for example in a treatment group compared with a control group, or at low compared with high stress, shows how the conditional associations among the other variables differ across the moderator. For each pair, condition() takes the main effect plus the interaction evaluated at value, aggregated over the two nodewise regressions and combined by the AND or OR rule. Pairs whose main effect or interaction did not survive the rule at fitting time contribute nothing. The moderator itself carries no edges, and edge weights are magnitudes

(non-negative), as in mgm::condition(). as.data.frame() on the moderated fit lists the size of the main effect and of the moderation effect of every pair with a nonzero effect, or, with value, the edges of the conditioned network.

value is read on the moderator's own scale: a level of a categorical moderator as coded in the data (1 or 2 for a moderator coded 1/2, "treat" for a factor level "treat"), or a raw value of a continuous moderator, so that value = mean(moderator) gives the network at the average moderator. The conditioned network is computed from the fit rather than estimated, so certificate() reports its certificate as missing.

condition() returns an undirected network of class psychnet (psychnet).

To fit the moderated model, use mgm_fit() with moderators.

Usage

```
condition(object, value, rule = NULL)

## S3 method for class 'psychnet_moderated'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., value = NULL)

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

Arguments

object	A psychnet_moderated fit from mgm_fit() with moderators.
value	A single moderator value: a level of a categorical moderator (number, label, factor or logical, as coded in the data), or a finite number on the raw scale of a continuous one. For as.data.frame(), NULL (default) gives the table of main and moderation effects.
rule	"AND" or "OR" combination of the two conditioned nodewise estimates of each pair. Default NULL uses the rule of the fit.
x	A psychnet_moderated fit.
row.names, optional	Ignored.
...	Unused.

Value

condition() returns an undirected network of class psychnet (psychnet); the value conditioned on is stored as mod_value.

as.data.frame() on the moderated fit returns, with value = NULL, one row per pair of nodes joined by a main effect or a moderation effect, with columns from, to, main_effect (magnitude of the pairwise main effect: the mean absolute nodewise estimate, zero when the pair does not survive the rule), interaction (magnitude of the moderator-by-pair interaction, NA for a pair that includes the moderator), moderated (interaction > 0, NA for a pair that includes the moderator) and involves_moderator. With a value, it returns the edge list (from, to, weight) of condition(x, value).

print() shows the number of nodes, the moderator and the fit settings, and returns x invisibly.

Functions

- `as.data.frame(psychnet_moderated)`: Main and moderation effects of every pair, or (with `value`) the edge list of the conditioned network.
- `print(psychnet_moderated)`: Print the nodes, the moderator and the fit settings.

Conditions

`psychnet_bad_input` when `object` is not a moderated fit, `rule` is not `"AND"` or `"OR"`, or `value` is missing, not of length one, not a level of a categorical moderator, or not a finite number for a continuous one.

See Also

`mgm_fit()` for fitting the moderated model; psychnet for the conditioned network.

Examples

```
set.seed(1)
x1 <- stats::rnorm(400); x2 <- stats::rnorm(400)
mod <- rep(1:2, each = 200)                # moderator coded 1/2
y <- x1 * (mod == 2) + stats::rnorm(400)   # x1-y edge only when mod == 2
d <- data.frame(x1 = x1, x2 = x2, y = y, mod = mod)
fit <- mgm_fit(d, types = c("g", "g", "g", "c"), moderators = "mod")
fit
as.data.frame(fit)            # main and moderation effects per edge
condition(fit, value = 1)     # no x1-y edge
condition(fit, value = 2)     # the x1-y edge appears
as.data.frame(fit, value = 2) # the same network as an edge list
```

---

cor_auto *Compute correlations suited to ordinal and continuous variables*

Description

`cor_auto()` computes a correlation matrix suited to a mix of ordinal and continuous variables: polychoric correlations between ordinal items, polyserial correlations between an ordinal item and a continuous variable, and Pearson correlations otherwise. The polychoric correlation assumes that each ordinal item is a coarse categorization of a normally distributed latent variable and estimates the correlation between the latent variables (Olsson, 1979); the polyserial correlation does the same for an ordinal item paired with a continuous variable (Olsson et al., 1982).

Questionnaire items are usually recorded on a few ordered categories, such as a five-point Likert scale, and treating these codes as continuous and computing Pearson correlations understates the association between the underlying traits. `cor_auto()` decides for each variable whether it is ordinal (integer values with at most `ordinal_max_levels` distinct values) and computes a polychoric correlation for each pair of ordinal variables, a polyserial correlation for each ordinal-continuous pair and a Pearson correlation otherwise.

It returns the resulting matrix adjusted to the nearest positive-definite correlation matrix, so that it can be used by the network estimators.

Network estimators for ordinal data commonly start from this correlation matrix. The estimators use the same correlations when given `cor_method = "auto"`, for example `ebic_glasso()`, `ggm_modselect()` and `cor_network()`. The function corresponds to `qgraph::cor_auto()`.

Usage

```
cor_auto(data, ordinal_max_levels = 7L, na_method = c("pairwise", "listwise"))
```

Arguments

`data`	A numeric data frame or matrix with one row per observation.
`ordinal_max_levels`	Largest number of distinct values for an integer-valued variable to be treated as ordinal: a whole number >= 2. Default 7.
`na_method`	Handling of missing values: `"pairwise"` (default) uses the observations available for each pair; `"listwise"` drops incomplete rows first.

Details

The polychoric correlation is the two-step estimator of Olsson (1979): the thresholds are fixed at the normal quantiles of the cumulative marginal proportions, and the likelihood of the contingency table is then maximized over the correlation. The polyserial correlation is the ad hoc estimator of Olsson, Drasgow and Dorans (1982), $r_{xy}\,\sigma_y / \sum_k \phi(\tau_k)$, rather than the two-step maximum-likelihood estimator that `qgraph::cor_auto()` obtains through `lavaan`. Both are consistent, but on finite samples they differ slightly (by up to about 0.006 in simulations), so a matrix that mixes ordinal and continuous variables does not reproduce `qgraph::cor_auto()` exactly.

An ordered factor in a data frame enters as its integer level codes and is ordinal whenever it has at most `ordinal_max_levels` levels. Unordered factors, character and logical columns are dropped with a message, and constant columns with a warning. A pair with no estimable association, such as an item observed at a single level, has correlation 0.

Value

A symmetric correlation matrix with the variable names as row and column names, one row and column per variable kept.

Conditions

`psychnet_bad_input` when `ordinal_max_levels` or `na_method` is invalid, `data` has no numeric columns or fewer than two usable variables, or a pair of variables has fewer than two joint observations; `psychnet_constant_variable` (warning) when a constant or all-missing column is dropped. A `psychnet_dropped_columns` message names non-numeric columns that were dropped.

References

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, 44(4), 443-460. doi:10.1007/BF02296207

Olsson, U., Drasgow, F., & Dorans, N. J. (1982). The polyserial correlation coefficient. *Psychometrika*, 47(3), 337-347. doi:10.1007/BF02294164

See Also

`ebic_glasso()`, `cor_network()` and the other estimators, which accept `cor_method = "auto"`.

Examples

```
set.seed(1)
z <- matrix(stats::rnorm(300 * 4), 300, 4) %*% chol(0.5^abs(outer(1:4, 1:4, "-")))
x <- apply(z, 2, function(col) as.integer(cut(col, 5)))   # 5-level Likert
cor_auto(x)
```

`cor_network` *Estimate a correlation network*

Description

`cor_network()` estimates a correlation network, in which each pair of variables is joined by their marginal (zero-order) correlation. An edge shows that two variables covary without controlling for any other variable, so it also reflects associations that run through other variables in the network.

The correlations are computed from raw data, or taken from a supplied correlation matrix. `cor_method` chooses Pearson, Spearman, Kendall or polychoric and polyserial correlations; `alpha` keeps only correlations that differ significantly from zero, with an optional multiple-comparison adjustment (`adjust`); `threshold` removes weak correlations.

`cor_network()` returns the correlation matrix as a network; the network involves no optimization and has no certificate.

The network gives a descriptive overview of the association structure, such as the clustering of items or symptoms, but it does not separate direct from indirect associations. For associations that control for the other variables use `pcor_network()` (all partial correlations) or `ebic_glasso()` (a regularized partial-correlation network); `tmfg_network()` filters the correlations to a sparse backbone. The network is the `"cor"` estimator of `bootnet`.

Usage

```
cor_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  threshold = 0,
  alpha = NULL,
```

```
  adjust = "none",
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

data	A numeric data frame or matrix with one row per observation. Not needed when cor_matrix is given. Ordered factors enter as their level codes; other non-numeric columns are dropped with a message, and constant or all-missing columns with a warning.
cor_matrix	A correlation matrix to use instead of data; a covariance matrix is rescaled to a correlation matrix. With alpha, n is then required.
n	Sample size of cor_matrix, needed only for the significance test: a single number >= 1, and a whole number >= 3 when alpha is set. Ignored when data is given.
cor_method	Correlation computed from data: "pearson" (default), "spearman", "kendall", or "auto" (polychoric and polyserial correlations for ordinal items; see cor_auto()).
threshold	Absolute edge weight below which an edge is set to zero: a single number >= 0. Default 0. A logical value (the threshold = TRUE rule of qgraph) is refused; use alpha for a significance rule.
alpha	Significance level in (0, 1): edges not significant at this level are set to zero. Default NULL, no test. A Kendall test needs raw data; partial Kendall correlations cannot be tested.
adjust	Multiple-comparison adjustment of the edge p-values, any method of stats::p.adjust(). Default "none".
na_method	Handling of missing values in data: "pairwise" (default) uses pairwise-complete correlations adjusted to the nearest positive-definite matrix; "listwise" drops incomplete rows. The two agree for complete data.
labels	Optional node labels, one unique label per node of the network.

Details

Each correlation is tested against zero with the statistic $t = r\sqrt{df/(1-r^2)}$ on $df = n - 2$ degrees of freedom for a correlation and $df = n - p$ for a full-order partial correlation of p variables. With na_method = "pairwise" and incomplete data, a correlation is tested with the number of observations available for that pair. Kendall correlations use a tie-corrected asymptotic test of the raw data. The p-values are adjusted over the distinct pairs and are computed from the correlations before threshold is applied; an edge is kept only if it passes both rules.

Value

A fitted network of class psychnet (psychnet) whose edge weights are correlations. as.data.frame() returns one row per edge with columns from, to and weight; certificate() reports that the network has no certificate. The object also stores the correlation matrix (cor_matrix), the sample size (n_eff; with pairwise data, the median number of observations per pair), the missing-data handling used (na_method; "none" with cor_matrix), the observations per pair when estimated from data

(n_pair) and, with alpha, the unadjusted p-values (p_values). Nodes keep the column order of the input.

Conditions

psychnet_degenerate_sample (error, also of class psychnet_bad_input) when the sample is too small for the test ($df \leq 0$); psychnet_bad_input when an argument is out of range, threshold is logical, cor_matrix is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, n is missing with cor_matrix and alpha, a Kendall test is requested with cor_matrix, data has fewer than two usable variables, a pair of variables is never observed together, or labels do not match the nodes; psychnet_constant_variable (warning) when a constant or all-missing column is dropped. A psychnet_dropped_columns message names non-numeric columns that were dropped.

See Also

pcor_network(); ebic_glasso(); tmfg_network(); psychnet(), which routes method = "cor" here; psychnet for the result.

Examples

```
x <- matrix(stats::rnorm(200 * 4), 200, 4)
cor_network(x)
cor_network(x, alpha = 0.05, adjust = "BH")
```

---

dichotomize *Split numeric items into binary 0/1 variables*

Description

dichotomize() splits numeric items into binary 0/1 variables, each column independently at its median (default), at its mean, or by rank. Ising network models (ising_fit(), ising_sampler()) describe binary variables, so Likert-type or continuous items must be split before such a model can be fitted; the resulting network then describes associations between scoring high and scoring low on the items rather than between their full response scales.

Missing values stay missing. The result is an integer 0/1 matrix with the dimensions of the input.

Splitting discards the information in the response scale. To keep it, estimate a Gaussian graphical model, for example ebic_glasso() with cor_method = "auto" (polychoric correlations for ordinal items), or a mixed graphical model with mgm_fit().

Usage

```
dichotomize(data, method = c("median", "mean", "rank"))
```

Arguments

data	A numeric matrix or data frame with one row per observation.
method	Split rule, applied to each column:
	"median" (default) 1 when the value is at or above the column median.
	"mean" 1 when the value is above the column mean.
	"rank" 1 when the value's average rank is above the midpoint. Tied values stay together, so the split can be unbalanced but never depends on row order.
	On an item with many responses at its floor, the median can equal the minimum, so "median" puts every observation in the upper group and the output column is constant (an Ising fit would drop it); "mean" or "rank" splits such items.

Value

An integer matrix of 0/1 values with the dimensions and dimnames of as.matrix(data); missing values stay NA.

Conditions

psychnet_bad_input when data is not numeric or has no columns, a column is constant or entirely missing, or method is not one of the rules; psychnet_constant_variable (warning) when a split puts every observation of a column in one group.

See Also

ising_fit() and ising_sampler(), which need binary data.

Examples

```
b <- dichotomize(SRL_GPT, method = "median")
table(b)                          # values are 0/1 only
```

---

difference_test *Test differences between edges or between centralities within a network*

Description

difference_test() tests whether pairs of edges, or pairs of nodes on one centrality measure, differ within a single network. It uses the bootstrapped difference test (Epskamp, Borsboom & Fried, 2018), which computes the difference of a pair in every bootstrap resample and treats the pair as different when the resulting interval excludes zero, that is, when the difference exceeds what sampling variation alone would produce. Interpreting one edge as stronger than another, or one node as more central than another, requires such a difference.

The function reads the resampled values stored by net_boot() and tests every pair of edges (type = "edge") or every pair of nodes on one bootstrapped centrality measure.

For each pair it returns the observed difference, the percentile interval of the bootstrapped difference, a two-sided bootstrap p-value and whether the interval excludes zero.

The test compares estimates within one network. To compare the same edge or the global strength between two groups, use `net_compare()`; for the interval of each edge or centrality on its own, use `net_boot()`.

Usage

```
difference_test(boot, type = "edge", ci = NULL, p_adjust = "none")

## S3 method for class 'psychnet_difference'
plot(x, style = c("box", "forest"), ...)
```

Arguments

`boot`	A `psychnet_bootstrap` object from `net_boot()`.
`type`	What to compare: `"edge"` (default) or one centrality measure bootstrapped by `net_boot()`, such as `"strength"` or `"expected_influence"`.
`ci`	Confidence level of the difference interval, a number in (0, 1). Default: the level used by `boot`.
`p_adjust`	Multiplicity adjustment for the p-values, any method of `stats::p.adjust()`. Default `"none"`.
`x`	A `psychnet_difference` data frame from `difference_test()`.
`style`	For `plot()`: `"box"` (default) or `"forest"`.
`...`	Unused.

Details

The p-value of a pair is twice the smaller of the shares of resamples in which the difference is positive or negative, with resamples in which it is exactly zero split evenly between the two, and capped at 1. `significant` is decided by the interval at level `ci` and is not changed by `p_adjust`, which adjusts only `p_value`.

Value

A data frame of class `psychnet_difference`, one row per pair, with columns `item1`, `item2`, `value1`, `value2` (the two observed values), `obs_diff` (`value1 - value2`), `lower`, `upper` (percentile interval of the bootstrapped difference), `p_value` and `significant` (the interval excludes zero). Edges are labelled `A--B`, or `A->B` for a directed estimator. `plot()` draws the table and returns `x` invisibly.

Functions

- `plot(psychnet_difference)`: The matrix of pairwise differences used by `bootnet`: items on both axes, ordered by observed value, with that value on the diagonal. An off-diagonal cell is filled when the pair differs (blue when the row item is larger, red when it is smaller), and every cell prints the sign of the difference (`+` or `-`), followed by stars for a significant

pair. `style = "forest"` instead draws one row per pair: the difference with its interval and a reference line at zero, significant pairs as filled points on solid segments and the others as open points on dashed segments.

Conditions

`psychnet_bad_input` when `boot` is not a `psychnet_bootstrap` object, `type` is not a bootstrapped quantity, `ci` or `p_adjust` is invalid, or fewer than two items can be compared.

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195-212. doi:10.3758/s13428017-08621

See Also

`net_boot()` for the resamples; `net_compare()` for differences between groups.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(150 * 5), 150, 5) %*% chol(0.4^abs(outer(1:5, 1:5, "-")))
colnames(x) <- paste0("V", 1:5)
# method = "pcor" and n_boot = 50 keep this example fast; a real analysis
# uses the "glasso" default and n_boot >= 1000.
bs <- net_boot(x, method = "pcor", n_boot = 50, cores = 1)
dt <- difference_test(bs, type = "strength")
dt
plot(dt)                  # matrix of pairwise differences
plot(dt, style = "forest") # one interval per pair
```

---

ebic_glasso *Estimate a Gaussian graphical model with the EBIC graphical lasso*

Description

`ebic_glasso()` estimates a Gaussian graphical model: a network in which variables are nodes joined by partial correlations, and a missing edge means two variables are independent given all the others. It fits the graphical lasso, which shrinks weak partial correlations to zero, and chooses the penalty by the extended Bayesian information criterion (Epskamp & Fried, 2018).

An edge is the association between two variables after controlling for all other variables in the network. The penalty sets weak partial correlations to exactly zero, so that only the associations supported by the data are retained, and the EBIC balances the fit of the model against the number of edges. `ebic_glasso()` fits the graphical lasso over a sequence of penalties, selects the penalty with the lowest EBIC and returns the partial-correlation network at that penalty. It takes raw data, or a correlation matrix together with its sample size. `gamma` sets how strongly the EBIC favours

sparse networks, `refit = "unregularized"` keeps the selected edges but estimates them without shrinkage, and `threshold` removes weak edges after estimation.

The returned network is the optimum of the penalized likelihood at the selected penalty, which `certificate()` checks.

This regularized partial-correlation network is the standard estimator for continuous and ordinal data in psychological network analysis. For data far from normality, `huge_network()` fits the same model after a rank-based transformation. `ggm_modselect()` selects an unregularized network by stepwise search, `pcor_network()` keeps every partial correlation, and `ising_fit()` and `mgm_fit()` estimate networks of binary and mixed variables. The model is the one fitted by `qgraph::EBICglasso()`.

Usage

```
ebic_glasso(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  gamma = 0.5,
  nlambda = 100L,
  lambda_min_ratio = 0.01,
  threshold = 0,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  na_method = c("pairwise", "listwise"),
  native = TRUE,
  lambda_path = NULL,
  penalize_diagonal = FALSE,
  refit = TRUE,
  labels = NULL
)
```

Arguments

`data`	A numeric data frame or matrix with one row per observation. Not needed when `cor_matrix` is given.
`cor_matrix`	A correlation matrix to use instead of `data`; a covariance matrix is rescaled to a correlation matrix. It must be symmetric and positive semi-definite. `n` is then required.
`n`	Sample size, required with `cor_matrix`: a single finite number= 1. Ignored when data is given.
`gamma`	EBIC hyperparameter, a number >= 0; larger values favour sparser networks and 0 gives the ordinary BIC. Default `0.5`.
`nlambda`	Number of penalties scanned, a whole number >= 2. Default `100`.
`lambda_min_ratio`	Smallest penalty as a fraction of the largest, in (0, 1). Default `0.01`.
`threshold`	Absolute partial correlation below which an edge is set to zero after estimation: a single number >= 0. Default `0`. A logical value (the `threshold = TRUE` rule of `qgraph`) is refused. A positive value leaves the certificate missing.

`cor_method` Correlation computed from `data`: `"pearson"` (default), `"spearman"`, `"kendall"`, or `"auto"` (polychoric and polyserial correlations for ordinal items; see `cor_auto()`).

`na_method` Handling of missing values in `data`: `"pairwise"` (default) uses pairwise-complete correlations adjusted to the nearest positive-definite matrix; `"listwise"` drops incomplete rows. The two agree for complete data.

`native` `TRUE` (default) uses the package's own solver. `FALSE` solves each penalty with the `glasso` package, which must be installed, to reproduce results of the `glasso` and `qgraph` packages.

`lambda_path` Optional numeric vector of positive penalties to scan, in the given order, replacing `nlambda` and `lambda_min_ratio`. Useful when resampling: the path of the full data, reused for every resample, keeps the selected penalties comparable. Default `NULL`.

`penalize_diagonal` Whether the penalty also applies to the diagonal of the precision matrix. Default `FALSE`.

`refit` What to return at the selected penalty: `TRUE` (default) the optimum at that penalty; `FALSE` the fit obtained while scanning the penalties, less precise and kept to reproduce earlier results; `"unregularized"` the unpenalized maximum-likelihood network restricted to the selected edges, without shrinkage.

`labels` Optional node labels, one unique label per node of the network (the columns left after non-numeric and constant columns are dropped).

Details

For a precision matrix Θ with E nonzero off-diagonal entries, fitted to a correlation matrix S of p variables and n observations, the criterion is

$$\mathrm{EBIC} = -2L + E \log n + 4\gamma E \log p, \quad L = \frac{n}{2}\left(\log\det\Theta - \mathrm{tr}(S\Theta)\right).$$

The penalties form a log-spaced sequence of `nlambda` values from the largest absolute off-diagonal correlation, at which the network is empty, down to `lambda_min_ratio` times that value. When two penalties give the same EBIC, the one earlier in the path is selected, which on the default path is the larger penalty and the sparser network. Unless `lambda_path` is given, a correlation matrix with no off-diagonal association returns the empty network with penalty 0.

With `na_method = "pairwise"` and incomplete data, the sample size entering the EBIC is the median number of observations available for a pair of variables.

The certificate is computed by `glasso_kkt()`. With `refit = "unregularized"` the network is no longer a penalized optimum and is checked by `ggm_support_kkt()` instead. With `refit = FALSE` or `native = FALSE` the network is solved to a looser tolerance, and its certificate can exceed the default tolerance of `certificate()`. After a positive `threshold` the certificate is missing; `print()` still shows the value for the network before thresholding.

Value

A fitted network of class `psychnet` (psychnet) whose edge weights are partial correlations. `as.data.frame()` returns one row per edge with columns `from`, `to` and `weight`; `print()` shows the selected penalty

and the certificate; `certificate()` reports the check. The object also stores the precision matrix (`precision`), the selected penalty (`lambda`), `gamma`, the correlation matrix used (`cor_matrix`), the EBIC of the selected network (`ebic`), the penalties scanned and their EBIC (`lambda_path`, `ebic_path`) and, with `refit = "unregularized"`, the selected graph (`support`). Nodes keep the column order of the input.

Conditions

`psychnet_bad_input` when an argument is out of range, `threshold` is logical, `cor_matrix` is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, `n` is missing with `cor_matrix`, `data` has fewer than two usable variables, a pair of variables is never observed together, `labels` do not match the nodes, `native = FALSE` without the `glasso` package, or every penalty fails to fit; `psychnet_constant_variable` (warning) when a constant or all-missing column is dropped. A `psychnet_dropped_columns` message names non-numeric columns that were dropped.

References

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432-441. doi:10.1093/biostatistics/kxm045

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

Epskamp, S., & Fried, E. I. (2018). A tutorial on regularized partial correlation networks. *Psychological Methods*, 23(4), 617-634. doi:10.1037/met0000167

See Also

`certificate()` and `glasso_kkt()` for the check; `huge_network()`, `ggm_modselect()` and `pcor_network()` for related Gaussian networks; `psychnet()`, which routes `method = "glasso"` here; psychnet for the result.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
fit <- ebic_glasso(cor_matrix = S, n = 250)
fit
as.data.frame(fit)
certificate(fit)
```

`event_frequencies` *Count the actions of each actor and session in an event log*

Description

`event_frequencies()` counts how often each action occurred in each occasion of an event log, where an occasion is an actor, or an actor's session. This table of action frequencies is what a network of actions, which shows which actions tend to occur together in the same occasion, is estimated from.

In an event log each row records one action by one actor, such as a student reading, taking a quiz or writing a note; behavioural and learning data often arrive in this form. Occasions are formed by `actor`, by `actor` and `session` when sessions are given, and, with `time` and `compute_sessions = TRUE`, by splitting an actor's events into sessions wherever the gap between consecutive events exceeds `time_threshold`.

The result is a data frame with one row per occasion and one column per distinct action, holding the count.

`psychnet()` performs this conversion itself when given an event log, so the function is needed only to inspect or reuse the table. Events with a missing or empty action are not counted.

Usage

```
event_frequencies(
  data,
  actor = "Actor",
  action = "Action",
  session = NULL,
  time = NULL,
  compute_sessions = TRUE,
  time_threshold = 900
)
```

Arguments

`data`	A data frame with one row per event.
`actor`	Column(s) identifying the actor. Default `"Actor"`.
`action`	Column holding the action. Default `"Action"`.
`session`	Optional column(s) identifying sessions within an actor.
`time`	Optional column of event times (numbers, dates, date-times or timestamps as text), used only to split sessions.
`compute_sessions`	Whether to split each actor's events into sessions at gaps in `time` longer than `time_threshold`. Default `TRUE`; has no effect without `time`.
`time_threshold`	Longest gap between consecutive events within one session, in the units of `time` (seconds for date-times). Default `900` (15 minutes).

Value

A data frame with one row per occasion and columns `actor`, `session` (an occasion label joining the actor, the session and, when sessions are computed from `time`, the session number, separated by `"."`) and one numeric column of counts per action.

Conditions

psychnet_bad_input when data is not a data frame, a named column is missing, action has fewer than two distinct values, actor or session has missing values, or time cannot be read as numbers or timestamps.

See Also

psychnet() with actor to estimate a network from an event log; psychnet_multilevel for within-actor and between-actor networks.

Examples

```
ev <- data.frame(
  Actor   = rep(c("a", "b"), each = 6),
  Session = rep(rep(1:2, each = 3), 2),
  Action  = c("read","quiz","read", "quiz","read","note",
              "note","note","read", "read","quiz","quiz"))
event_frequencies(ev, session = "Session")
```

---

ggm_modselect *Select an unregularized Gaussian graphical model by stepwise search*

Description

ggm_modselect() selects an unregularized Gaussian graphical model: it chooses which edges are present and then estimates them by maximum likelihood, so the partial correlations it reports are not shrunk. In a Gaussian graphical model variables are nodes joined by partial correlations, with a missing edge meaning conditional independence; a regularized estimate such as the graphical lasso instead shrinks every retained edge towards zero, and Williams et al. (2019) discuss when the unregularized approach is preferable to regularization.

ggm_modselect() generates candidate graphs from the graphical lasso over a sequence of penalties, fits the unregularized maximum-likelihood model on each candidate and keeps the one with the lowest extended Bayesian information criterion (EBIC). With stepwise = TRUE (default) it then adds or removes single edges as long as the EBIC improves.

The returned network is the maximum-likelihood network on the selected graph, which certificate() checks.

For a regularized network use ebic_glasso(); for all partial correlations without selection use pcor_network(). The procedure is the one of qgraph::ggmModSelect().

Usage

```
ggm_modselect(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  gamma = 0,
```

```
  stepwise = TRUE,
  nlambda = 100L,
  lambda_min_ratio = 0.01,
  threshold = 0,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  na_method = c("pairwise", "listwise"),
  native = TRUE,
  labels = NULL
)
```

Arguments

`data`	A numeric data frame or matrix with one row per observation. Not needed when `cor_matrix` is given.
`cor_matrix`	A correlation matrix to use instead of `data`; a covariance matrix is rescaled to a correlation matrix. `n` is then required.
`n`	Sample size, required with `cor_matrix`: a single finite number= 1. Ignored when data is given.
`gamma`	EBIC hyperparameter, a number >= 0. Default `0` (the ordinary BIC).
`stepwise`	Whether to refine the best candidate graph by adding and removing single edges. Default `TRUE`.
`nlambda`	Number of graphical lasso penalties used to generate candidate graphs, a whole number >= 2. Default `100`.
`lambda_min_ratio`	Smallest penalty as a fraction of the largest, in (0, 1). Default `0.01`.
`threshold`	Absolute partial correlation below which an edge is set to zero after estimation: a single number >= 0. Default `0`. A logical value is refused. A positive value leaves the certificate missing.
`cor_method`	Correlation computed from `data`: `"pearson"` (default), `"spearman"`, `"kendall"`, or `"auto"` (polychoric and polyserial correlations for ordinal items; see `cor_auto()`).
`na_method`	Handling of missing values in `data`: `"pairwise"` (default) or `"listwise"`. See `ebic_glasso()`.
`native`	Solver for the candidate graphs: `TRUE` (default) the package's own graphical lasso, `FALSE` the `glasso` package, which must be installed. The returned network is the unregularized fit either way.
`labels`	Optional node labels, one unique label per node of the network.

Details

The EBIC of a graph with E edges is $-2L+E\log n+4\gamma E\log p$, where L is the log-likelihood of the maximum-likelihood fit on that graph. Because the retained edges are not shrunk, the default `gamma = 0` (the ordinary BIC) is the intended setting, as in `qgraph::ggmModSelect()`; the regularized `ebic_glasso()` and `huge_network()` default to `0.5`.

The stepwise search visits every pair of variables in turn, adds or removes the edge, and keeps the change whenever the EBIC decreases; it stops when a full pass brings no improvement (at most 200

passes). A correlation matrix with no off-diagonal association returns the empty network without a search.

The certificate is computed by ggm_support_kkt(). After a positive threshold it is missing; print() still shows the value for the network before thresholding.

Value

A fitted network of class psychnet (psychnet) whose edge weights are partial correlations. as.data.frame() returns one row per edge with columns from, to and weight; certificate() reports the check. The object also stores the precision matrix (precision), the selected graph as a logical matrix (support), gamma, the correlation matrix (cor_matrix) and the EBIC of the selected graph (ebic).

Conditions

psychnet_bad_input when an argument is out of range, threshold is logical, cor_matrix is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, n is missing with cor_matrix, data has fewer than two usable variables, labels do not match the nodes, or native = FALSE without the glasso package; psychnet_no_converge (error) when no candidate graph can be fitted; psychnet_fit_failed (warning) when some candidate fits failed and were skipped; psychnet_no_converge (warning) when the selected fit's certificate exceeds 1e-6; psychnet_constant_variable (warning) when a constant or all-missing column is dropped.

References

Williams, D. R., Rhemtulla, M., Wysocki, A. C., & Rast, P. (2019). On nonregularized estimation of psychological networks. *Multivariate Behavioral Research*, 54(5), 719-750. doi:10.1080/00273171.2019.1575716

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

See Also

ebic_glasso() for the regularized network; certificate() and ggm_support_kkt() for the check; psychnet(), which routes method = "ggm" here; psychnet for the result.

Examples

```
S <- 0.5^abs(outer(1:6, 1:6, "-"))
ggm_modselect(cor_matrix = S, n = 250)
```

---

ggm_support_kkt *Check that a precision matrix is the maximum-likelihood network on a graph*

Description

ggm_support_kkt() checks whether a precision matrix is the maximum-likelihood Gaussian graphical model on a given graph, for the correlation or covariance matrix it was fitted to. Some Gaussian graphical models are estimated without a penalty: a graph is chosen first, and the precision matrix is then the maximum-likelihood estimate under the constraint that every pair of variables not joined by an edge has a partial correlation of zero.

This constrained estimate is unique and is characterized by two conditions: the covariance implied by the model reproduces the observed covariance for every pair joined by an edge and for every variance, and the precision matrix is zero for every pair that is not joined. A matrix meeting both conditions is the maximum-likelihood network on that graph.

ggm_support_kkt() returns the largest violation of these conditions; a value near zero shows that the matrix is the constrained optimum, up to numerical precision.

Networks from ggm_modselect(), logo_network() and ebic_glasso(refit = "unregularized") store this value, and certificate() reports it. Penalized graphical lasso networks are checked by glasso_kkt().

Usage

```
ggm_support_kkt(theta, cor_matrix, support, active_tol = 1e-08)
```

Arguments

theta	The precision matrix to check: a finite, square numeric matrix.
cor_matrix	The correlation or covariance matrix the model was fitted to, of the same size as theta.
support	A logical matrix of the same size, TRUE where an edge is allowed, with no missing values. It is made symmetric, so an upper- or lower-triangular matrix describes the same graph; the diagonal is ignored.
active_tol	A single finite number >= 0 (default 1e-8). It is checked but does not affect the result: every entry of theta outside the graph counts as a violation of its absolute size.

Details

For a Gaussian graphical model on a graph G, with $W = \Theta^{-1}$, the maximum-likelihood conditions are $W_{ij} = S_{ij}$ for every (i, j) in G and on the diagonal, and $\Theta_{ij} = 0$ for every (i, j) not in G. The function returns the largest absolute deviation from these conditions; a theta that is asymmetric (by more than 1e-8) or not positive definite returns Inf.

Value

A single number, the largest absolute violation of the conditions: 0 at the exact constrained optimum, Inf for an asymmetric or non-positive-definite theta.

Conditions

`psychnet_bad_input` when `theta` is not a finite square numeric matrix, `cor_matrix` is not a finite matrix of the same size, `support` is not a logical matrix of that size without missing values, or `active_tol` is not a single finite number >= 0.

See Also

`certificate()`, which reports the stored value of a fitted network; `ggm_modselect()` and `logo_network()`; `glasso_kkt()` for penalized networks.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
# An AR(1) correlation has a tridiagonal inverse: the chain graph is exact.
chain <- abs(row(S) - col(S)) <= 1
ggm_support_kkt(solve(S), S, support = chain)
# The same precision violates an empty-graph (diagonal) constraint.
ggm_support_kkt(solve(S), S, support = diag(6) > 0)
# A fitted network reports its value through certificate().
certificate(ggm_modselect(cor_matrix = S, n = 250))
```

---

glasso_kkt *Check that a precision matrix solves the graphical lasso*

Description

`glasso_kkt()` checks whether a precision matrix solves the graphical lasso for a given correlation or covariance matrix and penalty. The graphical lasso estimates a Gaussian graphical model as the precision (inverse covariance) matrix that maximizes the Gaussian likelihood minus a penalty on the absolute partial associations (Friedman et al., 2008), and a matrix is its solution if and only if it satisfies the optimality (Karush-Kuhn-Tucker) conditions of the problem.

The objective is strictly convex, so it has exactly one optimum, and checking the conditions shows whether a network is the graphical lasso solution without solving the problem a second time. In words, the covariance implied by the precision matrix must equal the observed covariance on the diagonal, differ from it by exactly the penalty for every pair joined by an edge, and by at most the penalty for every pair that is not.

`glasso_kkt()` returns the largest violation of the conditions; a value near zero shows that the matrix is the optimum, up to numerical precision.

Networks from `ebic_glasso()` and `huge_network()` store this value, and `certificate()` reports it. Unpenalized networks restricted to a graph are checked by `ggm_support_kkt()`, and nodewise Ising and mixed graphical models by `glm_lasso_kkt()`.

Usage

```
glasso_kkt(
  theta,
  cor_matrix,
  rho,
  active_tol = 1e-08,
  penalize_diagonal = FALSE
)
```

Arguments

theta	The precision matrix to check: a finite, square numeric matrix.
cor_matrix	The correlation or covariance matrix the model was fitted to, of the same size as theta.
rho	The penalty: a single finite number >= 0.
active_tol	Absolute value above which an off-diagonal entry of theta counts as an edge. Default 1e-8.
penalize_diagonal	Whether the fit being checked also penalized the diagonal. Default FALSE, the default of ebic_glasso().

Details

The objective, with the penalty on the off-diagonal entries, is

$$\min_{\Theta \succ 0} -\log\det\Theta + \operatorname{tr}(S\Theta) + \rho \sum_{i \neq j} |\Theta_{ij}|.$$

With $W = \Theta^{-1}$, the optimality conditions are $W_{ii} = S_{ii}$; $W_{ij} - S_{ij} = \rho\,\operatorname{sign}(\Theta_{ij})$ where $\Theta_{ij} \neq 0$; and $|W_{ij} - S_{ij}| \leq \rho$ otherwise. The function returns the largest absolute deviation from these conditions. Because the objective is strictly convex, a precision matrix with zero violation is its unique global optimum.

When the penalty also applies to the diagonal (penalize_diagonal = TRUE, the objective adds $\rho \sum_i |\Theta_{ii}|$), the diagonal condition becomes $W_{ii} = S_{ii} + \rho$. Checking a diagonally penalized fit with the default would report a violation of exactly ρ, so this argument must match the fit being checked.

The objective is defined only for symmetric positive-definite matrices, so a theta that is asymmetric (by more than 1e-8) or not positive definite is not a solution and returns Inf.

Value

A single number, the largest absolute violation of the optimality conditions: 0 at the exact optimum, Inf for an asymmetric or non-positive-definite theta.

Conditions

psychnet_bad_input when theta is not a finite square numeric matrix, cor_matrix is not a finite matrix of the same size, rho or active_tol is not a single finite number >= 0, or penalize_diagonal is not TRUE or FALSE.

References

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432-441. doi:10.1093/biostatistics/kxm045

See Also

`certificate()`, which reports the stored value of a fitted network; `ebic_glasso()` and `huge_network()`; `ggm_support_kkt()` and `glm_lasso_kkt()` for the other optimality checks.

Examples

```
S <- 0.5^abs(outer(1:5, 1:5, "-"))
# At rho = 0 the optimum is the inverse of S, so the violation is ~0 ...
glasso_kkt(solve(S), S, rho = 0)
# ... and a perturbed precision matrix violates the conditions.
glasso_kkt(solve(S) + diag(0.1, 5), S, rho = 0)
# A fitted network reports the same value through certificate().
certificate(ebic_glasso(cor_matrix = S, n = 200))
```

---

glm_lasso_kkt *Check that coefficients solve a lasso-penalized regression*

Description

`glm_lasso_kkt()` checks whether coefficients solve a lasso-penalized linear or logistic regression, given its predictors, response, intercept, coefficients and penalty. Such a regression, with a penalty on the absolute coefficients (the lasso), is a convex problem, and its coefficients are the optimum if and only if they satisfy the optimality (Karush-Kuhn-Tucker) conditions.

Ising models and mixed graphical models are estimated node by node: each variable is regressed on all the others with the lasso, logistic for a binary variable and linear for a continuous one, and the coefficients become the edges of the network. Checking the conditions therefore shows whether a regression was solved, without fitting it a second time. In words, the residuals must average to zero; for every coefficient that is not zero, the average product of its predictor with the residuals must equal the penalty, with the sign of the coefficient; and for every coefficient that is zero it must not exceed the penalty in absolute value. With `lambda = 0` the check applies to an unpenalized maximum-likelihood regression.

`glm_lasso_kkt()` returns the largest violation of the conditions for one regression; a value near zero shows that the coefficients are the optimum, up to numerical precision.

`ising_fit()`, `ising_sampler()` and `mgm_fit()` store the largest value over their nodewise regressions, and `certificate()` reports it. Graphical lasso networks are checked by `glasso_kkt()`.

Usage

```
glm_lasso_kkt(
  X,
  y,
```

```
  b0,
  beta,
  lambda,
  family = "gaussian",
  weights = NULL,
  active_tol = 1e-08
)
```

Arguments

X	Numeric predictor matrix, one row per observation, on the scale on which the fit was penalized.
y	Numeric response, one value per row of X; between 0 and 1 for family = "binomial".
b0	Fitted intercept, a single finite number.
beta	Fitted coefficients, one per column of X.
lambda	The penalty: a single finite number >= 0.
family	"gaussian" (default, linear regression) or "binomial" (logistic regression).
weights	Optional observation weights, finite, non-negative and not all zero, one per row of X. Default NULL, equal weights.
active_tol	Absolute value above which a coefficient counts as nonzero. Default 1e-8.

Details

With fitted mean $\mu = \beta_0 + X\beta$ (gaussian) or $\mu = 1/(1 + e^{-(\beta_0 + X\beta)})$ (binomial), observation weights w and n rows, let $g_j = n^{-1} \sum_i w_i x_{ij}(y_i - \mu_i)$. The conditions are $n^{-1} \sum_i w_i(y_i - \mu_i) = 0$ for the unpenalized intercept, $g_j = \lambda \operatorname{sign}(\beta_j)$ for every active coefficient, and $|g_j| \leq \lambda$ otherwise. The function returns the largest absolute deviation from these conditions. The penalty acts on the scale of X; the package's nodewise fits standardize the predictors to mean 0 and unit variance, so a fit is checked with its standardized predictors and coefficients.

Value

A single number, the largest absolute violation of the optimality conditions; 0 at the exact optimum.

Conditions

psychnet_bad_input when X is not a finite numeric matrix matching y, beta and weights in size, y is not finite (or not between 0 and 1 for binomial), b0, lambda or active_tol is not a valid single number, family is not one of the two choices, or weights contain a negative or non-finite value or are all zero.

See Also

certificate(), which reports the stored value of a fitted network; ising_fit(), ising_sampler() and mgm_fit(); glasso_kkt() and ggm_support_kkt() for the other optimality checks.

Examples

```
set.seed(1)
x <- scale(matrix(stats::rnorm(200 * 3), 200, 3))
y <- as.numeric(x %*% c(0.5, 0, -0.3) + stats::rnorm(200))
# x has centred columns, so the least-squares intercept is mean(y) and the
# slopes are qr.solve(x, y): the unpenalized optimum, violation ~0.
glm_lasso_kkt(x, y, b0 = mean(y), beta = qr.solve(x, y), lambda = 0)
# The generating coefficients are not the sample optimum.
glm_lasso_kkt(x, y, b0 = mean(y), beta = c(0.5, 0, -0.3), lambda = 0)
```

---

huge_network *Estimate a nonparanormal Gaussian graphical model*

Description

huge_network() estimates a nonparanormal Gaussian graphical model: a regularized partial-correlation network for variables that are normal only after an unknown monotone transformation. The nonparanormal model (Liu et al., 2009) relaxes the assumption of the Gaussian graphical model that the variables are jointly normal, an assumption that skewed or heavy-tailed questionnaire and symptom data often violate, so the network of partial correlations is defined on the transformed scale and is unaffected by the shape of each variable's distribution.

The transformation is estimated from the ranks of the observations. huge_network() transforms the data, computes the correlation matrix of the transformed variables and estimates a regularized partial-correlation network on it with the EBIC graphical lasso, as in ebic_glasso(). npn chooses the transformation: normal scores of the ranks with shrinkage (default) or truncation, or the "skeptic", a correlation derived from Spearman's rank correlation.

The returned network is the optimum of the penalized likelihood at the selected penalty, which certificate() checks.

For approximately normal data, ebic_glasso() fits the same model without the transformation; its cor_method = "auto" treats ordinal items through polychoric correlations instead. The model is the one fitted by huge::huge() with the nonparanormal transformation.

Usage

```
huge_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  npn = c("shrinkage", "truncation", "skeptic"),
  gamma = 0.5,
  nlambda = 100L,
  lambda_min_ratio = 0.01,
  threshold = 0,
  na_method = c("pairwise", "listwise"),
  native = TRUE,
  labels = NULL
)
```

Arguments

data	A numeric data frame or matrix with one row per observation. Not needed when cor_matrix is given.
cor_matrix	A correlation matrix of already transformed data, used instead of data; no transformation is applied and n is required.
n	Sample size, required with cor_matrix: a single finite number= 1. Ignored when data is given.
npn	The transformation: "shrinkage" (default), "truncation" or "skeptic". Ignored with cor_matrix.
gamma	EBIC hyperparameter, a number >= 0; larger values favour sparser networks. Default 0.5.
nlambda	Number of penalties scanned, a whole number >= 2. Default 100.
lambda_min_ratio	Smallest penalty as a fraction of the largest, in (0, 1). Default 0.01.
threshold	Absolute partial correlation below which an edge is set to zero after estimation: a single number >= 0. Default 0. A logical value is refused. A positive value leaves the certificate missing.
na_method	Handling of missing values in data: "pairwise" (default) transforms each variable over its observed values and uses pairwise-complete correlations; "listwise" drops incomplete rows first.
native	TRUE (default) uses the package's own graphical lasso solver; FALSE uses the glasso package, which must be installed. See ebic_glasso().
labels	Optional node labels, one unique label per node of the network.

Details

The shrinkage transform replaces each observation by $\Phi^{-1}(r/(n+1))$, with r its rank among the n observed values of the variable. The truncation transform uses $\Phi^{-1}(r/n)$ truncated to $[\delta, 1-\delta]$, with $\delta = 1/(4n^{1/4}\sqrt{\pi \log n})$. The skeptic uses $2\sin(\pi\rho_S/6)$, with ρ_S Spearman's correlation. A correlation matrix that is not positive definite (possible for the skeptic and for pairwise data) is adjusted to the nearest positive-definite matrix before estimation.

The penalty path, the EBIC and the certificate are those of ebic_glasso() with its default settings; the certificate is computed by glasso_kkt() on the transformed correlation matrix. After a positive threshold the certificate is missing; print() still shows the value for the network before thresholding.

Value

A fitted network of class psychnet (psychnet) whose edge weights are partial correlations on the transformed scale. as.data.frame() returns one row per edge with columns from, to and weight; certificate() reports the check. The object also stores the precision matrix (precision), the selected penalty (lambda), gamma, the transformed correlation matrix (cor_matrix), the transformation used (npn, "precomputed" with cor_matrix) and the EBIC of the selected network (ebic).

Conditions

psychnet_bad_input when an argument is out of range, threshold is logical, cor_matrix is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, n is missing with cor_matrix, data has fewer than two usable variables, a pair of variables is never observed together, labels do not match the nodes, or native = FALSE without the glasso package; psychnet_constant_variable (warning) when a constant or all-missing column is dropped.

References

Liu, H., Lafferty, J., & Wasserman, L. (2009). The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10, 2295-2328.

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432-441. doi:10.1093/biostatistics/kxm045

See Also

ebic_glasso() for the untransformed model; certificate() and glasso_kkt() for the check; psychnet(), which routes method = "huge" here; psychnet for the result.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(300 * 5), 300, 5)
x <- exp(x %*% chol(0.4^abs(outer(1:5, 1:5, "-"))))   # break normality
huge_network(x)
```

---

ising_fit *Estimate a regularized Ising network for binary items*

Description

ising_fit() estimates a regularized Ising network from binary items, such as symptoms that are present or absent or items that are endorsed or not. The Ising model describes the items through a threshold for each item and an interaction for each pair, the association between two items after conditioning on all other items, and is estimated here by the nodewise lasso method of van Borkulo et al. (2014).

A positive edge means that the two items tend to be endorsed together, a negative edge that endorsing one makes the other less likely, and an absent edge that the two are conditionally independent; the network shows which items are directly associated once the rest are taken into account. The function regresses each item on all other items by L1-penalized (lasso) logistic regression and selects each item's penalty by the extended Bayesian information criterion (EBIC). Every pair is estimated twice, once in each item's regression. Under the AND rule (default) an edge is kept when both estimates are nonzero, under the OR rule when either is; its weight is the mean of the two. Rows can be filtered by their sum score (min_sum) and weighted (weights). Each regression is

checked against its optimality (Karush–Kuhn–Tucker) conditions, and `certificate()` reports the largest violation across the items.

The result is an undirected network of class `psychnet` (psychnet).

For unpenalized estimates with significance-based pruning, use `ising_sampler()`; for data that combine continuous and binary variables, `mgm_fit()`. Ordinal or continuous items can be split with `dichotomize()` first. The model is the one fitted by `IsingFit::IsingFit()`, which `native = FALSE` reproduces.

Usage

```
ising_fit(
  data,
  gamma = 0.25,
  rule = c("AND", "OR"),
  nlambda = 100L,
  lambda_min_ratio = 0.01,
  min_sum = NULL,
  weights = NULL,
  na_method = c("pairwise", "listwise"),
  native = TRUE,
  labels = NULL
)
```

Arguments

`data`	Binary data: a data frame or matrix with one row per observation. Columns may be numeric 0/1, logical, or two-level factors (the first level is coded 0). Any other column, such as a numeric column coded 1/2 or a factor with more than two levels, is an error naming it. A constant column is dropped.
`gamma`	EBIC hyperparameter, a single number >= 0; larger values give sparser networks. Default `0.25`.
`rule`	`"AND"` (default) keeps an edge when both nodewise estimates are nonzero; `"OR"` when either is.
`nlambda`	Number of penalty values on each nodewise path, a whole number >= 2. Default `100`.
`lambda_min_ratio`	Smallest penalty as a fraction of the largest, in (0, 1). Default `0.01`. With `native = FALSE`, leaving it unset uses glmnet's own default, as `IsingFit::IsingFit()` does.
`min_sum`	Minimum number of endorsed items per row; rows below it are dropped before fitting. Default `NULL` keeps every row.
`weights`	Optional non-negative observation weights, one per row after missing-data handling (and filtered with `min_sum`). Only their relative sizes matter: `weights = w` and `weights = 7 * w` give the same network. Requires `native = TRUE`. Default `NULL` (unweighted).

na_method	"pairwise" (default) fills each missing value with the item's more frequent value and keeps every row; "listwise" drops rows with a missing value. The two agree for complete data.
native	TRUE (default) uses the package's own solver. FALSE fits each regression with the glmnet package (which must be installed) and reproduces IsingFit::IsingFit(); its certificate reflects glmnet's convergence tolerance and is usually larger. weights are not accepted with native = FALSE.
labels	Optional node labels: one per node, or one per input column (labels of dropped constant columns are then discarded).

Details

Edge weights and thresholds are on the log-odds scale of the 0/1 coding. The weight of an edge is the change in the log-odds of one item when the other is endorsed, averaged over the two regressions; an item's threshold is its log-odds when every other item is 0.

Value

An undirected network of class psychnet (psychnet). as.data.frame() gives one row per edge (from, to, weight), certificate() the largest optimality violation of the nodewise regressions, and net_predict() the predictability of each node. The object also stores the node thresholds (thresholds), the combination rule (rule) and the nodewise regressions (nodewise).

Conditions

psychnet_bad_input when a column is not binary, fewer than two usable columns remain, gamma, nlambda, lambda_min_ratio, min_sum, weights, native or labels is invalid, min_sum removes every row (or every row with positive weight), weights is combined with native = FALSE, or native = FALSE is requested without the glmnet package; psychnet_constant_variable (warning) when a constant column is dropped; psychnet_no_converge (warning) when a nodewise regression does not reach its optimum; psychnet_degenerate_sample (warning) when an item takes a single value on every retained row, so its regression has no finite intercept. The last two are raised with native = TRUE.

References

van Borkulo, C. D., Borsboom, D., Epskamp, S., Blanken, T. F., Boschloo, L., Schoevers, R. A., & Waldorp, L. J. (2014). A new method for constructing networks from binary data. *Scientific Reports*, 4, 5918. doi:10.1038/srep05918

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

See Also

ising_sampler() for the unregularized model; mgm_fit() for mixed continuous and binary data; dichotomize() to prepare items; certificate() and glm_lasso_kkt() for the optimality check; net_predict(); psychnet() with method = "ising".

Examples

```
# Ising models need 0/1 data: split the five SRL construct scores at the median.
b <- dichotomize(SRL_GPT)
fit <- ising_fit(b)
fit
as.data.frame(fit)
certificate(fit)
```

`ising_sampler` *Estimate an unregularized Ising network for binary items*

Description

`ising_sampler()` estimates an unregularized Ising network from binary items, in which edges can be kept or removed by their statistical significance rather than by a penalty. The Ising model describes binary items through a threshold for each item and an interaction for each pair, the association between two items after conditioning on all others (van Borkulo et al., 2014); estimated without a penalty, the interactions are maximum-likelihood estimates, not shrunk toward zero.

The function regresses each item on all other items by ordinary (unpenalized) logistic regression. With `alpha`, every coefficient whose Wald p-value, optionally adjusted for multiplicity, exceeds `alpha` is set to zero. Every pair is estimated twice, once in each item's regression; the AND rule (default) keeps an edge when both estimates remain nonzero, the OR rule when either does, and the edge weight is the mean of the two. The certificate is the largest violation of the maximum-likelihood optimality conditions (the score equations) across the regressions, so a value near zero shows that every regression is at its maximum. Pruning sets coefficients to zero after fitting, without refitting; the pruned network is no longer the solution that was checked, and its certificate is reported as missing.

The result is an undirected network of class `psychnet` (psychnet).

Unpenalized estimates need many more observations than items. When an item is perfectly predicted by the others (separation), its maximum-likelihood estimate does not exist; the penalized `ising_fit()` is the alternative then, and in small samples generally.

Usage

```
ising_sampler(
  data,
  rule = c("AND", "OR"),
  alpha = NULL,
  adjust = "none",
  min_sum = NULL,
  weights = NULL,
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

data	Binary data: a data frame or matrix with one row per observation, with numeric 0/1, logical, or two-level factor columns, as in ising_fit(). A constant column is dropped.
rule	"AND" (default) keeps an edge when both nodewise estimates are nonzero; "OR" when either is.
alpha	Significance level for pruning, a number in (0, 1). Default NULL keeps every edge.
adjust	Multiplicity correction for the p-values used in pruning, one of stats::p.adjust.methods. Default "none". Used only with alpha.
min_sum	Minimum number of endorsed items per row; rows below it are dropped before fitting. Default NULL keeps every row.
weights	Optional non-negative observation weights, one per row after missing-data handling (and filtered with min_sum). Only their relative sizes matter. Default NULL (unweighted).
na_method	"pairwise" (default) fills each missing value with the item's more frequent value; "listwise" drops rows with a missing value.
labels	Optional node labels: one per node, or one per input column.

Details

Edge weights and thresholds are on the log-odds scale of the 0/1 coding, as in ising_fit(). The multiplicity correction (adjust) is applied over all nodewise coefficients, two per pair, before the AND or OR rule. After pruning, the thresholds are recomputed from the retained coefficients; the remaining coefficients keep their unpruned estimates.

Value

An undirected network of class psychnet (psychnet). as.data.frame() gives one row per edge (from, to, weight), certificate() the maximum-likelihood certificate, and net_predict() the predictability of each node. The certificate is missing when pruning removed a coefficient or when a regression did not converge. The object also stores the node thresholds (thresholds), the unadjusted Wald p-values of the nodewise coefficients (p_values, row = the regressed item), the certificate of the unpruned fit (fit_kkt) and what the certificate refers to (certificate_target: "returned_network", "pre_pruning_fit" or "nonconverged_fit").

Conditions

psychnet_bad_input when a column is not binary, fewer than two usable columns remain, rule, alpha, adjust, min_sum, weights or labels is invalid, or min_sum removes every row (or every row with positive weight); psychnet_constant_variable (warning) when a constant column is dropped; psychnet_separation (warning) when items are perfectly predicted by the others, so their estimates do not exist; psychnet_no_converge (warning) when a regression fails to converge for another reason; psychnet_singular_information (warning) when an item's Fisher information is singular, so its Wald p-values are unreliable.

References

van Borkulo, C. D., Borsboom, D., Epskamp, S., Blanken, T. F., Boschloo, L., Schoevers, R. A., & Waldorp, L. J. (2014). A new method for constructing networks from binary data. *Scientific Reports*, 4, 5918. doi:10.1038/srep05918

See Also

`ising_fit()` for the penalized model; `dichotomize()` to prepare items; `certificate()` and `glm_lasso_kkt()` for the optimality check; `net_predict()`; `psychnet()` with `method = "ising_sampler"`.

Examples

```
b <- dichotomize(SRL_GPT)
ising_sampler(b)                # every edge; certificate near zero
pruned <- ising_sampler(b, alpha = 0.05)
as.data.frame(pruned)
certificate(pruned)             # missing: pruning changed the fit
```

---

lmg_certificate *Check the decomposition of a relative-importance network*

Description

`lmg_certificate()` checks the decomposition stored in a relative-importance network from `relimp_network()`. Such a network divides the R^2 of each variable, regressed on all the others, among its predictors; it is defined by a decomposition rather than as the optimum of an objective, and two identities must hold for that decomposition to be correct.

The shares a node receives must sum exactly to its R^2, which is the efficiency property of the Shapley value. And that R^2 must be the one implied by the correlation matrix, which has a closed form. Together they show that the stored shares add up to the variance each node's predictors actually explain.

`lmg_certificate()` returns the larger of the two deviations: a value near zero confirms the decomposition, up to numerical precision.

`certificate()` reports the same value for a fitted network. The check does not verify the R^2 of the individual subset regressions between the empty and the full model, and the sum identity alone would hold for any set of subset values; the closed form is what ties the full model to the data.

Usage

```
lmg_certificate(x)
```

Arguments

x	A relative-importance network from `relimp_network()` (psychnet).

Details

For outcome j with shares w_{kj} and full-model R^2_j, the two deviations are $\max_j |\sum_k w_{kj} - R^2_j|$ and $\max_j |R^2_j - (1 - 1/(S^{-1})_{jj})|$. When the correlation matrix S is singular (reciprocal condition number below `1e-12`) the closed form does not exist and only the first deviation is returned. For a network with `normalized = TRUE` the unscaled shares are used.

Value

A single number, the larger of the two deviations; 0 for an exact decomposition.

Conditions

`psychnet_bad_input` when `x` is not a network from `relimp_network()`.

See Also

`relimp_network()`; `certificate()`; `tmfg_certificate()`, the other structural check.

Examples

```
S <- 0.4^abs(outer(1:5, 1:5, "-"))
lmg_certificate(relimp_network(cor_matrix = S))
```

---

logo_network — *Estimate a sparse Gaussian graphical model on a filtered graph (LoGo)*

Description

`logo_network()` estimates a sparse Gaussian graphical model on a filtered graph with the Local-Global (LoGo) method. LoGo works in two steps: it first chooses the network structure by information filtering with the triangulated maximally filtered graph (TMFG), and then estimates the partial correlations as the maximum-likelihood Gaussian graphical model on that graph (Barfuss et al., 2016).

The TMFG retains the strongest correlations under the constraint that the graph stays planar and is built from tetrahedra (four-variable cliques), which fixes the number of edges at $3(p - 2)$ for p variables. Because the graph is decomposable, the maximum-likelihood estimate has a closed form, assembled from the inverses of small local correlation matrices, so no penalty or iterative fitting is needed. The model is estimated from raw data or from a correlation matrix with its sample size.

`logo_network()` returns the partial-correlation network of this model, the maximum-likelihood network on the TMFG graph, which `certificate()` checks.

The graph alone, with correlations as edge weights, is returned by `tmfg_network()`. For a network whose sparsity is chosen by the data use `ebic_glasso()` or `ggm_modselect()`. The model is the one fitted by `NetworkToolbox::LoGo()`.

Usage

```
logo_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  threshold = 0,
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

data	A numeric data frame or matrix with one row per observation. Not needed when cor_matrix is given.
cor_matrix	A correlation matrix to use instead of data; a covariance matrix is rescaled to a correlation matrix. n is then required.
n	Sample size, required with cor_matrix and recorded on the result: a single finite number >= 1. Ignored when data is given.
cor_method	Correlation computed from data: "pearson" (default), "spearman", "kendall", or "auto" (polychoric and polyserial correlations for ordinal items; see cor_auto()).
threshold	Absolute partial correlation below which an edge is set to zero after estimation: a single number >= 0. Default 0. A logical value is refused. A positive value leaves the certificate missing.
na_method	Handling of missing values in data: "pairwise" (default) or "listwise". See ebic_glasso().
labels	Optional node labels, one unique label per node of the network.

Details

With C the four-variable cliques and T the three-variable separators of the TMFG, the precision matrix is

$$\Theta = \sum_{c \in C} [S_{cc}^{-1}]^0 - \sum_{t \in T} [S_{tt}^{-1}]^0,$$

where $[\cdot]^0$ pads a block with zeros to the full size. The certificate is computed by ggm_support_kkt() with the TMFG as the graph. After a positive threshold it is missing; print() still shows the value for the network before thresholding.

The closed form needs every clique and separator block to be invertible. A block that is singular or nearly so (reciprocal condition number below 1e-10, as with duplicate or collinear variables) is refused rather than approximated, as is a precision matrix whose certificate exceeds 1e-6.

Value

A fitted network of class psychnet (psychnet) whose edge weights are partial correlations. as.data.frame() returns one row per edge with columns from, to and weight; certificate() reports the check. The object also stores the precision matrix (precision), the TMFG graph as a logical matrix (support) and the correlation matrix (cor_matrix).

Conditions

`psychnet_singular` (error, also of class `psychnet_bad_input`) when a clique or separator block is singular or nearly so, or the precision matrix fails its certificate; `psychnet_bad_input` when there are fewer than four variables, an argument is out of range, `threshold` is logical, `cor_matrix` is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, `n` is missing with `cor_matrix`, or `labels` do not match the nodes; `psychnet_constant_variable` (warning) when a constant or all-missing column is dropped.

References

Barfuss, W., Massara, G. P., Di Matteo, T., & Aste, T. (2016). Parsimonious modeling with information filtering networks. *Physical Review E*, 94(6), 062306. doi:10.1103/PhysRevE.94.062306

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, 5(2), 161-178. doi:10.1093/comnet/cnw015

See Also

`tmfg_network()` for the filtered graph; `certificate()` and `ggm_support_kkt()` for the check; `psychnet()`, which routes `method = "logo"` here; psychnet for the result.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(300 * 6), 300, 6)
logo_network(x)
```

---

mgm_fit *Estimate a mixed graphical model for continuous and binary variables*

Description

`mgm_fit()` estimates a mixed graphical model: one network for variables of different types, such as continuous symptom ratings and binary indicators. An edge connects two variables that remain associated after conditioning on all other variables, and an absent edge indicates conditional independence (Haslbeck & Waldorp, 2020), so the network shows which variables are directly related when they cannot all be treated as Gaussian or all as binary.

The function regresses each variable on all others with the L1-penalized (lasso) regression suited to its type, linear for continuous and logistic for binary variables, and selects each variable's penalty by the extended Bayesian information criterion (EBIC). Small coefficients are then removed by a threshold (`threshold`, default `"LW"`), and the two estimates of each pair are combined by the AND rule (default) or the OR rule into an undirected edge whose weight is their mean. Variable types are detected from the data or given in `types`. Each regression is checked against its optimality (Karush–Kuhn–Tucker) conditions; `certificate()` reports the largest violation. A threshold that removes a coefficient changes the network after fitting, so its certificate is then reported as missing; `threshold = "none"` keeps it. With `moderators`, the function instead fits a moderated model in which one variable can change the strength of every edge; see `condition()`.

The result is an undirected network of class psychnet (psychnet), or a psychnet_moderated fit with moderators.

For binary data only, use ising_fit(); for continuous data only, ebic_glasso(). Categorical variables with more than two levels need native = FALSE. The model is the one fitted by mgm::mgm().

Usage

```
mgm_fit(
  data,
  gamma = 0.25,
  types = NULL,
  nlambda = 100L,
  lambda_min_ratio = 0.01,
  threshold = c("LW", "HW", "none"),
  rule = c("AND", "OR"),
  moderators = NULL,
  weights = NULL,
  na_method = c("pairwise", "listwise"),
  native = TRUE,
  labels = NULL
)
```

Arguments

data	A data frame or matrix with one row per observation. Columns are continuous, binary, or (with native = FALSE) categorical with more than two levels. With native = TRUE every column must be numeric. Columns with fewer than two distinct observed values are dropped. A character column with more than 10 distinct values is refused as an identifier; convert it with factor() if it is a categorical variable.
gamma	EBIC hyperparameter, a single number >= 0; larger values give sparser networks. Default 0.25.
types	Node types, one per column of data: "g" (Gaussian) or "c" (categorical). Default NULL detects them as mgm::mgm() does: factor and character columns are categorical, and so are numeric columns with at most 10 distinct integer values. Such a numeric column with more than two values is named in a warning, because it changes the model for that node; Likert items often meet this rule. A non-factor column declared "c" with more than 10 distinct values is refused.
nlambda	Number of penalty values on each nodewise path, a whole number >= 2. Default 100.
lambda_min_ratio	Smallest penalty as a fraction of the largest, in (0, 1). Default 0.01. With native = FALSE, leaving it unset uses glmnet's own default, as mgm::mgm() does.
threshold	Threshold on the selected coefficients: "LW" (default), "HW" or "none" (see Details), as in mgm::mgm().
rule	"AND" (default) keeps an edge when both nodewise estimates are nonzero; "OR" when either is.

moderators	Optional moderator: one column of data, by name or by position. The moderated model lets this variable change the strength of every pairwise edge and returns a psychnet_moderated fit, read with condition() and as.data.frame(). Needs at least three variables and does not accept weights. Default NULL.
weights	Optional non-negative observation weights, one per row after missing-data handling. Only their relative sizes matter, as in mgm::mgm(). Requires native = TRUE. Default NULL (unweighted).
na_method	"pairwise" (default) fills each missing value with the variable's mean (continuous) or most frequent level (categorical) and keeps every row; "listwise" drops rows with a missing value. The two agree for complete data.
native	TRUE (default) uses the package's own solver, for continuous and binary variables. FALSE fits each regression with the glmnet package (which must be installed), following mgm::mgm(); it also allows categorical variables with more than two levels, but not weights, and its certificate reflects glmnet's convergence tolerance. A network with a multi-level categorical node cannot be passed to net_predict(). One-hot encoding such a variable is not equivalent: it turns one node into several.
labels	Optional node labels: one per node, or one per input column (labels of dropped constant columns are then discarded).

Details

Continuous variables are standardized before fitting, binary predictors enter as 0/1 indicators, and the coefficient of a binary response is expressed on the two-class multinomial scale of mgm::mgm() (half the logistic coefficient). Edge magnitudes are therefore comparable with those of mgm::mgm(). A binary-binary edge keeps the sign of its logistic coefficient, whereas mgm::mgm() reports such an edge without a sign; compare those edges on their absolute value. Weak edges near the selection or threshold boundary can differ between the two engines.

With β_i the selected coefficients of variable i, q the number of its predictor columns ($p-1$ for p continuous or binary variables) and n the sample size, the "LW" threshold removes coefficients smaller in absolute value than $\|\beta_i\|_2\sqrt{\log(q)/n}$, and the "HW" threshold those smaller than $\sqrt{\log(q)/n}$.

Node order is the column order of data, and a categorical variable with several levels remains one node.

Value

An undirected network of class psychnet (psychnet); with moderators, a psychnet_moderated fit (see condition()). as.data.frame() gives one row per edge (from, to, weight), certificate() the largest optimality violation of the nodewise regressions (missing when the threshold removed a coefficient), and net_predict() the predictability of each node. The object also stores the node types (types), the number of levels per node (levels, 1 for continuous) and the certificate of the regressions before thresholding (fit_kkt).

Conditions

psychnet_bad_input when data is not a data frame or matrix, types is not one "g"/"c" per column, a character column looks like an identifier, a non-factor column declared "c" has more

than 10 values, a categorical variable has more than two levels or a column is not numeric with `native = TRUE`, `moderators` does not name one column or names a dropped constant column, a moderated model has fewer than three variables or is given `weights`, `weights` is combined with `native = FALSE`, `native = FALSE` is requested without the `glmnet` package, or `gamma`, `nlambda`, `lambda_min_ratio`, `weights` or `labels` is invalid; `psychnet_type_promotion` (warning) when a numeric column with more than two values is detected as categorical; `psychnet_no_converge` (warning) when a nodewise regression does not reach its optimum; `psychnet_degenerate_sample` (warning) when a binary variable takes a single value on every row with positive weight. The last two are raised with `native = TRUE`.

References

Haslbeck, J. M. B., & Waldorp, L. J. (2020). mgm: Estimating time-varying mixed graphical models in high-dimensional data. *Journal of Statistical Software*, 93(8), 1-46. doi:10.18637/jss.v093.i08

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

See Also

`condition()` for the moderated model; `ising_fit()` for binary data; `ebic_glasso()` for continuous data; `certificate()` and `glm_lasso_kkt()` for the optimality check; `net_predict()`; `psychnet()` with `method = "mgm"`.

Examples

```
set.seed(1)
f <- stats::rnorm(400)
g1 <- f + stats::rnorm(400); g2 <- f + stats::rnorm(400)
b1 <- (f + stats::rnorm(400) > 0) * 1L
d <- data.frame(g1 = g1, g2 = g2, b1 = b1, n = stats::rnorm(400))
fit <- mgm_fit(d)
fit
as.data.frame(fit)
```

---

net_aggregate *Aggregate communities of items into a network between communities*

Description

`net_aggregate()` builds a network between communities of items, such as the items of several scales, in which each community is a single node. It offers two routes: the items of each community are combined into a composite score, and a network is then estimated among the composites; or the association between two communities is summarized directly from their items.

The score methods are the mean, median or sum of the items, the first principal component, a one-factor score, or a mean in which each item is weighted by its connections to the other items of its community in the item-level network (`"loadings"`). The association methods are the mean signed edge weight between the items of two communities in the item-level network (`"average"`), Escoufier's RV coefficient (`"rv"`), or the first canonical correlation (`"canonical"`).

A score method returns a data frame with one composite column per community, ready for psychnet(); an association method returns the network between communities directly.

The centrality of single items with respect to the communities is computed by net_bridge(); pairs of nearly identical items can be found with redundancy() before aggregating.

Usage

```
net_aggregate(
  data,
  communities,
  method = "mean",
  estimator = "glasso",
  scale = TRUE,
  labels = NULL,
  ...
)
```

Arguments

data	A numeric data frame or matrix with one row per observation and one column per item.
communities	The community of each column of data, in one of three forms: an unnamed vector with one entry per column, in column order (NA for a column to leave out, such as an identifier); a vector named by item label; or a list of item-label vectors, one per community, whose names are the community names. Every retained item must belong to exactly one community. Communities are ordered by list order, factor level, numeric value, or alphabetically for character labels.
method	Aggregation method. Score methods (return a data frame): "mean" (default), "median", "sum", "pca", "factor" or "loadings". Association methods (return a network): "average", "rv" or "canonical".
estimator	Estimator of the item-level network used by "loadings" and "average", as in psychnet(). Default "glasso".
scale	If TRUE (default), standardize each item before forming composite scores.
labels	Optional item labels, one per column of data, used in place of its column names (and matched by a named communities).
...	Further arguments for the estimator.

Details

Non-numeric columns are dropped, and so is any column that is constant once incomplete rows are removed. Rows with a missing value in any numeric column are dropped (listwise deletion). communities is matched to the columns of data before anything is dropped, so a dropped column must not carry a community: give it NA, or leave it out of a named vector or list.

A community of one item is that item (standardized when scale = TRUE). Principal-component and factor scores are signed to point in the direction of the community's items. A factor score needs at least three items, so a smaller community, or one for which stats::factanal() fails, uses its first principal component instead. In "loadings", each item's weight is the sum of its absolute

edge weights to the other items of its community (equal weights when these are all zero). A pair of communities for which `stats::cancor()` fails gets weight 0 under `"canonical"`.

Value

For a score `method`, a data frame with one row per retained observation and one column per community, named by community. For an association `method`, a fitted network (psychnet) with one node per community; it is directed only for `"average"` with a directed estimator.

Conditions

`psychnet_bad_input` for an invalid argument, or when `communities` lists an item twice, names an item that is not a column, or leaves a retained item without a community; `psychnet_dropped_column` (an error, also `psychnet_bad_input`) when a dropped column carries a community; `psychnet_rows_dropped` (warning) when incomplete rows are dropped; `psychnet_constant_variable` (warning) when a constant column is dropped; `psychnet_dropped_columns` (a message) when non-numeric columns are dropped; `psychnet_fit_failed` (warning) when `stats::factanal()` or `stats::cancor()` fails for a community, naming it. With `"loadings"` or `"average"`, the estimator raises its own conditions.

See Also

`psychnet()` to estimate a network from the composite scores; `net_bridge()` for bridge centrality; `redundancy()` for redundant items.

Examples

```
net_aggregate(SRL_Claude, communities = c(1, 1, 2, 2, 2))            # reduced data
net_aggregate(SRL_Claude, communities = c(1, 1, 2, 2, 2), method = "rv")  # macro net
```

---

net_boot *Bootstrap the edges and centralities of a network*

Description

`net_boot()` bootstraps a network to show how accurately its edge weights and node centralities are estimated. The nonparametric bootstrap approximates the sampling variation of the network, whose edge weights and centralities would differ in another sample from the same population, by re-estimating it on samples drawn with replacement from the observations; the spread of the bootstrapped values shows which edges, and which differences between them, can be interpreted (Epskamp, Borsboom & Fried, 2018).

The function resamples the rows of `data`, re-estimates the network with the chosen estimator on every resample, and summarizes each edge weight and each node centrality by its bootstrap mean and percentile confidence interval. For each edge it also reports the proportion of resamples in which the edge was estimated as nonzero, and flags it as `significant` when its interval excludes zero. Options add intervals for node predictability (`predictability`) and a version of the network that keeps only the edges whose interval excludes zero (`threshold`).

It returns these intervals as tables read with `as.data.frame()`, and keeps the bootstrapped values on the object, so that `difference_test()` can test differences between edges, or between node centralities, without resampling again.

The bootstrap describes the accuracy of estimates within one sample. For the stability of the order of centralities when cases are dropped, use `net_stability()`; for the stability of the edge weights, use `net_casedrop_reliability()` and `net_split_reliability()`; to test whether two groups have different networks, use `net_compare()`.

Usage

```
net_boot(
  data,
  method = "glasso",
  n_boot = 1000L,
  ci = 0.95,
  measures = c("strength", "expected_influence"),
  centrality_fn = NULL,
  predictability = FALSE,
  threshold = FALSE,
  diff_test = FALSE,
  p_adjust = "none",
  labels = NULL,
  cores = NULL,
  engine = NULL,
  estimator_args = list(),
  ...
)

## S3 method for class 'psychnet_bootstrap'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("edges", "centrality", "predictability", "thresholded"),
  significant = FALSE
)

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

## S3 method for class 'psychnet_bootstrap'
plot(
  x,
 type = c("edges", "centrality", "edge_diff", "centrality_diff", "predictability"),
  measure = NULL,
  ...
)
```

Arguments

data	A data frame or matrix with one row per observation, or a psychnet_group estimated from a data set, in which case each group is bootstrapped with the group's own estimator settings.
method	The estimator, as in psychnet(). Default "glasso".
n_boot	Number of bootstrap resamples. Default 1000.
ci	Confidence level of the percentile intervals, a number in (0, 1). Default 0.95.
measures	Centrality measures to bootstrap. Default c("strength", "expected_influence"); any measure accepted by net_centralities() can be added.
centrality_fn	Optional function supplying measures that are not built in (see net_centralities()).
predictability	If TRUE, also bootstrap node predictability (R^2) when the estimator returns a precision matrix (the Gaussian graphical models). Default FALSE.
threshold	If TRUE, also keep the observed network with every edge whose interval includes zero set to zero. Default FALSE.
diff_test	If TRUE, also store matrices of pairwise difference p-values on the object: for the edges (up to 500 edges) and for each centrality measure. Default FALSE. The tidy version of these tests is difference_test().
p_adjust	Multiplicity adjustment for the matrices stored by diff_test, any method of stats::p.adjust(). Default "none".
labels	Optional node labels, one per node.
cores	Number of CPU cores. NULL (default) uses two thirds of the detected cores; 1 runs serially. Runs are serial on Windows.
engine	Optional solver for every refit: "base" or "glasso" for method = "glasso" or "huge", "base" or "glmnet" for "ising" or "mgm". NULL (default) keeps the estimator's default.
estimator_args	Named list of arguments for the estimator. Use it for an estimator argument that shares a name with an argument of net_boot(), e.g. estimator_args = list(threshold = TRUE). Values here override those in ....
...	Further arguments for the estimator (net_boot()); ignored by the methods.
x	A psychnet_bootstrap object.
row.names, optional	Ignored.
what	For as.data.frame(): "edges" (default), "centrality", "predictability" (needs predictability = TRUE) or "thresholded" (needs threshold = TRUE).
significant	For as.data.frame() with what = "edges": if TRUE, return only the edges whose interval excludes zero. Default FALSE.
type	For plot(): "edges" (default; edge weights with their intervals, sorted by observed weight), "centrality" (one panel per measure), "edge_diff" or "centrality_diff" (the matrix of pairwise difference tests for the edges or for one measure), or "predictability" (needs predictability = TRUE).
measure	For plot() with type = "centrality" or "centrality_diff": the measure(s) to draw. Default: all bootstrapped measures, or the first one for "centrality_diff".

Details

Each resample is passed to the estimator exactly as the full data would be, including incomplete rows and factor columns, so the estimator's own handling of missing values and variable types applies to every resample. For `method = "mgm"` the node types are decided once on the full data and kept for every resample. A resample on which a variable has no variation keeps that variable as an isolated node.

All resample indices are drawn before any network is fitted, so the result is reproducible with `set.seed()` and does not depend on `cores`.

For an estimator whose edge weights are non-negative by construction (`method = "relimp"`, whose weights are shares of explained variance), a percentile interval excludes zero almost surely: every edge is then flagged `significant` and `threshold = TRUE` removes nothing. Pairwise difference tests with `difference_test()` remain informative.

Value

`net_boot()` returns an object of class `psychnet_bootstrap`. Its tables are read with `as.data.frame()`:

- `as.data.frame(x)` (`what = "edges"`): one row per edge, with columns `from`, `to`, `observed`, `mean` (bootstrap mean), `lower`, `upper` (percentile interval), `prop_nonzero` (share of resamples in which the edge is nonzero) and `significant` (the interval excludes zero). For a directed estimator each ordered pair is a row.
- `what = "centrality"`: one row per node, with `node` and, for each measure, its observed value and the columns `<measure>_lower` and `<measure>_upper`.
- `what = "predictability"`: one row per node, with `node`, `value` (the observed R^2), `lower` and `upper`.
- `what = "thresholded"`: one row per edge, with `from`, `to`, `observed`, `weight` (the observed weight, or 0 when its interval includes zero) and `retained`.

The object also stores the observed network and the bootstrapped value of every edge and centrality in every resample, which `difference_test()` and the difference plots use. `print()` shows the estimator, the number of resamples, the interval level and the numbers of edges, significant edges and nodes, and returns `x` invisibly; `plot()` returns `x` invisibly. For a psychnet_group, `net_boot()` returns one result per group (see psychnet_result_group).

Functions

- `as.data.frame(psychnet_bootstrap)`: The edge intervals (default), the centrality or predictability intervals, or the thresholded network, chosen by `what`.
- `print(psychnet_bootstrap)`: The estimator, the number of resamples, the interval level and the edge counts.
- `plot(psychnet_bootstrap)`: Edge, centrality or predictability intervals, or a matrix of pairwise difference tests (see `type`). Edges whose interval excludes zero are filled points on solid segments, the others open points on dashed segments.

Conditions

psychnet_bad_input for an invalid argument, for a group that was not estimated from a data set, and from as.data.frame() or plot() when the requested table was not computed; psychnet_estimation_failed when the network cannot be estimated on the full data (with the estimator's message); psychnet_resample_failed (warning) when some resamples fail, naming the first error, and as an error when fewer than two succeed; psychnet_constant_variable (warning) when a constant column is dropped; psychnet_ignored_argument (warning) when predictability = TRUE but the estimator returns no precision matrix; psychnet_uninformative_test (warning) for threshold = TRUE or as.data.frame(significant = TRUE) with method = "relimp".

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195-212. doi:10.3758/s13428017-08621

See Also

difference_test() for pairwise differences; net_centralities() for the measures; net_stability(), net_casedrop_reliability() and net_split_reliability() for case-dropping and split-half stability; psychnet() for the estimators.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(150 * 5), 150, 5) %*% chol(0.4^abs(outer(1:5, 1:5, "-")))
colnames(x) <- paste0("V", 1:5)
# method = "pcor" and n_boot = 50 keep this example fast; a real analysis
# uses the "glasso" default and n_boot >= 1000.
bs <- net_boot(x, method = "pcor", n_boot = 50, cores = 1)
bs
as.data.frame(bs)
as.data.frame(bs, significant = TRUE)
as.data.frame(bs, what = "centrality")
plot(bs)                       # edge weights with their intervals
plot(bs, type = "centrality")  # centralities with their intervals
```

---

net_bridge *Compute bridge centrality between communities of nodes*

Description

net_bridge() computes the bridge centrality of each node in a network whose nodes belong to known communities, such as the symptoms of two disorders or the items of different scales. Bridge centrality (Jones, Ma & McNally, 2021) describes how strongly a node is connected to the communities other than its own; nodes with high bridge centrality are those through which the communities are linked, for example symptoms that may connect two disorders in comorbidity.

The function computes five measures for each node of an undirected network: bridge strength, the sum of the absolute weights of its edges to other communities; one-step bridge expected influence

(bridge_ei1), the sum of the signed weights of those edges; two-step bridge expected influence (bridge_ei2), which adds the signed weights of the paths of two edges that end in another community; bridge betweenness, the number of shortest paths between nodes of different communities that pass through the node; and bridge closeness, the inverse of the mean distance from the node to the nodes of other communities. Shortest paths use the positive edges, with weights inverted to distances. The communities are given by the user; normalize divides the measures by the number of nodes they can involve.

It returns one row per node, with its community and the five measures.

For centrality without regard to communities, use net_centralities(). To merge each community into a single node and estimate the network between communities, use net_aggregate().

Usage

```
net_bridge(x, communities, normalize = FALSE, labels = NULL)

## S3 method for class 'psychnet_bridge'
plot(x, measures = NULL, ...)
```

Arguments

x	An undirected fitted network (psychnet) or a symmetric weighted adjacency matrix. For plot(), the table returned by net_bridge().
communities	The community of each node, in one of three forms: an unnamed vector with one entry per node, in node order; a vector named by node label; or a list of node-label vectors, one per community, whose names are the community names, e.g. list(anxiety = c("A1", "A2"), mood = c("M1", "M2")). Every node must belong to exactly one community, and there must be at least two.
normalize	If TRUE, divide bridge strength and one-step bridge expected influence by the number q of nodes in other communities, bridge betweenness by (p - 1) q, and two-step bridge expected influence by q + (p - 1)(q - q / (p - 1)), where p is the number of nodes, as networktools::bridge(normalize = TRUE) does. Bridge closeness is unchanged. Default FALSE.
labels	Optional node labels for a bare matrix, one unique label per node.
measures	For plot(): the bridge columns to draw. Default: all five.
...	Unused.

Value

A data frame of class psychnet_bridge, one row per node, with columns node, community, bridge_strength, bridge_betweenness, bridge_closeness, bridge_ei1 and bridge_ei2. plot() draws one sorted panel per measure, each node's point coloured and shaped by its community, with a legend of the communities, and returns x invisibly.

Functions

- plot(psychnet_bridge): One sorted panel per bridge measure, with points coloured and shaped by community.

Conditions

psychnet_bad_input when x is directed, is not a finite symmetric square matrix, or labels is invalid; when communities lists a node twice, names a node that is not in the network, leaves a node without a community, has the wrong length, or defines fewer than two communities. plot() raises psychnet_bad_input for a measures entry that is not a bridge column, psychnet_plot_skipped (warning) when it skips a measure without finite values, and psychnet_nothing_to_plot (an error) when no requested measure has one.

References

Jones, P. J., Ma, R., & McNally, R. J. (2021). Bridge centrality. *Multivariate Behavioral Research*, 56(2), 353-367.

See Also

net_centralities() for centrality without communities; net_aggregate() for the network between communities.

Examples

```
S <- 0.3^abs(outer(1:6, 1:6, "-"))
fit <- ebic_glasso(cor_matrix = S, n = 400)
br <- net_bridge(fit, communities = c(1, 1, 1, 2, 2, 2))
br
plot(br)
```

---

net_casedrop_reliability

Assess the stability of edge weights when cases are dropped

Description

net_casedrop_reliability() assesses how stable the edge weights of a network are when cases are dropped. It applies the case-dropping procedure of Epskamp, Borsboom and Fried (2018) to the edge weights rather than to node centralities: the network is re-estimated on subsets from which an increasing proportion of cases has been removed, and each subset's edge weights are compared with the full-sample edge weights. The edge weights state which variables are associated, and how strongly; if the network describes the population rather than the particular sample, the edge weights estimated from a subset of the cases should resemble those of the full sample.

The function draws iter subsets without replacement at every proportion in drop_prop, re-estimates the network on each, and compares the subset and full-sample edge weights by their correlation and by the mean, median and maximum absolute difference. It summarizes each of these metrics by drop proportion and computes an edge-weight CS-coefficient: the largest proportion at which the share of subsets whose edge-weight correlation exceeds threshold itself exceeds certainty, or 0 when no proportion qualifies.

It returns one row per metric and drop proportion, with the edge-weight CS-coefficient kept alongside the table.

net_stability() applies the same procedure to node centralities, and net_split_reliability() compares networks estimated on two halves of the sample.

Usage

```
net_casedrop_reliability(
  data,
  method = "glasso",
  drop_prop = seq(0.1, 0.9, by = 0.1),
  iter = 100L,
  threshold = 0.7,
  certainty = 0.95,
  cor_method = c("pearson", "spearman", "kendall"),
  labels = NULL,
  estimator_args = list(),
  ...
)

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

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

Arguments

data	A data frame or matrix with one row per observation, or a psychnet_group estimated from a data set (assessed per group).
method	The estimator, as in psychnet(). Default "glasso".
drop_prop	Proportions of cases to drop, each in (0, 1). Default seq(0.1, 0.9, by = 0.1). Each must remove at least one case and keep at least two.
iter	Number of subsets per proportion. Default 100.
threshold	Edge-weight correlation a subset must exceed to count as stable, a number in (0, 1). Default 0.7.
certainty	Share of stable subsets that must be exceeded, a number in (0, 1). Default 0.95.
cor_method	Correlation between subset and full-sample edge weights: "pearson" (default, as in bootnet), "spearman" or "kendall".
labels	Optional node labels, one per node.
estimator_args	Named list of arguments for the estimator. Use it for an estimator argument that shares a name with an argument of this function, e.g. estimator_args = list(cor_method = "spearman"); such names given in ... do not reach the estimator.
...	Further arguments for the estimator (net_casedrop_reliability()); ignored by the methods.
x	A psychnet_casedrop object.

Details

Both comparisons in the CS rule are strict, as in `bootnet::corStability()`. The edge-weight correlation of a subset is 0 when either edge-weight vector is constant (for example an empty subset network), and undefined when fewer than two edges are available. An undefined correlation, or a subset on which the network could not be estimated, counts as not exceeding `threshold`, so such subsets never raise the CS-coefficient (`bootnet` instead leaves undefined correlations out). Each subset is passed to the estimator as the full data would be (see `net_boot()`).

Value

A data frame of class `psychnet_casedrop`, one row per metric and drop proportion, with columns `metric` (`"correlation"`, `"mean_abs_dev"`, `"median_abs_dev"` or `"max_abs_dev"`), `drop_prop` (nominal), `actual_drop` (the proportion dropped after rounding to whole cases), `mean`, `sd` and `n_valid` (subsets in which the metric was defined; `mean` and `sd` are `NA` when too few were). The edge-weight CS-coefficient and the correlation of every subset are kept with the table; `print()` shows the CS-coefficient above the table. `plot()` draws the four metrics against the proportion dropped, each with a band of +/- 1 SD, and marks `threshold` and the CS-coefficient on the correlation panel. `print()` and `plot()` return `x` invisibly. For a psychnet_group, the function returns one result per group.

Functions

- `print(psychnet_casedrop)`: The edge-weight CS-coefficient, followed by the table of metrics.
- `plot(psychnet_casedrop)`: The four metrics against the proportion of cases dropped.

Conditions

`psychnet_bad_input` for an invalid argument, including drop proportions that remove no case or keep fewer than two, and for a group that was not estimated from a data set; `psychnet_estimation_failed` when the network cannot be estimated on the full data; `psychnet_resample_failed` (warning) when some subsets cannot be estimated, naming the first error, and as an error when none can; `psychnet_undefined_statistic` (warning) when the edge-weight correlation is undefined in some subsets; `psychnet_constant_centrality` (warning) when the full-sample edge weights are all equal (for example an empty network), which makes the CS-coefficient 0; `psychnet_constant_variable` (warning) when a constant column is dropped.

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy. *Behavior Research Methods*, 50(1), 195-212.

See Also

`net_stability()` for centralities; `net_split_reliability()` for split-half agreement; `net_boot()` for bootstrap intervals of each edge.

Examples

```
# method, iter and drop_prop are chosen so the example runs quickly; a
# real analysis uses the defaults (method = "glasso", iter = 100,
# drop_prop = seq(0.1, 0.9, 0.1)).
rel <- net_casedrop_reliability(SRL_Claude, method = "pcor", iter = 5,
                                drop_prop = c(0.25, 0.5))
rel
plot(rel)
```

`net_centralities` *Compute the centrality of each node in a network*

Description

`net_centralities()` computes the centrality of each node in a network, by default its strength and expected influence. Centrality indices summarize the position of each node in a network; for psychometric networks, whose edges are signed and weighted, the most informative are strength, the sum of the absolute weights of a node's edges, and expected influence, the sum of their signed weights, in which negative edges offset positive ones (Robinaugh, Millner & McNally, 2016). A node with high strength is strongly connected to the other variables; expected influence also shows whether these connections are, on balance, positive or negative.

On request the function adds two-step expected influence, which also counts the expected influence a node passes on through its neighbours, and betweenness and closeness, which are computed on shortest paths through the absolute edge weights, inverted to distances (Opsahl, Agneessens & Skvoretz, 2010). Because betweenness and closeness ignore the sign of an edge, they are less suited to signed networks. Further measures can be supplied as a function (`centrality_fn`). For a directed network, such as one from `relimp_network()`, strength and expected influence sum a node's outgoing edges.

It returns one row per node, with one column per requested measure.

Whether differences in centrality can be interpreted depends on their accuracy and stability; see `net_boot()`, `difference_test()` and `net_stability()`. Centrality with respect to communities is computed by `net_bridge()`, and the betweenness of edges by `net_edge_betweenness()`.

Usage

```
net_centralities(
  x,
  measures = c("strength", "expected_influence"),
  centrality_fn = NULL,
  ...
)

## S3 method for class 'psychnet_centrality'
plot(
  x,
  type = c("bar", "line"),
```

```
  scale = c("raw", "z", "relative"),
  measures = NULL,
  ...
)
```

Arguments

x	A fitted network (psychnet), a weighted adjacency matrix, or a psychnet_group (one table per group). For `plot()`, the table returned by `net_centralities()`.
measures	Measures to compute: any of `"strength"`, `"expected_influence"` (the defaults), `"expected_influence_2step"`, `"betweenness"`, `"closeness"`, and names supplied by `centrality_fn`. For `plot()`: the measure columns to draw (default: all).
centrality_fn	Optional function that takes the weighted adjacency matrix (with node labels as dimnames) and returns a named list of centrality vectors, one numeric value per node, for measures that are not built in. A named vector is matched to the nodes by label; an unnamed one is taken in node order.
...	Unused.
type	For `plot()`: `"bar"` (default; one sorted panel per measure) or `"line"` (one panel per measure, with the nodes in a shared order).
scale	For `plot(type = "line")`: `"raw"` (default), `"z"` (z-scores per measure) or `"relative"` (rescaled to 0-1 per measure). `type = "bar"` always shows raw values.

Details

Two-step expected influence is the node's expected influence plus the sum, over its neighbours, of each edge weight times that neighbour's expected influence.

Betweenness and closeness are normalized, as in igraph's `betweenness(normalized = TRUE)` and `closeness(normalized = TRUE)`: betweenness is divided by the number of pairs of other nodes, `(p - 1)(p - 2) / 2`, and closeness is the number of reachable nodes divided by the sum of the distances to them (the inverse mean distance). They therefore differ from `qgraph::centrality_auto()`, which reports betweenness as a count of paths and closeness as `1 / sum(distances)`.

Value

A data frame of class `psychnet_centrality`, one row per node, with a `node` column and one column per requested measure, named as in `measures`. `plot()` draws it and returns it invisibly. For a psychnet_group, one table per group (see psychnet_result_group).

Functions

- `plot(psychnet_centrality)`: One panel per measure, nodes sorted by value (`type = "bar"`) or in a shared order (`type = "line"`).

Conditions

`psychnet_bad_input` when `x` is not a fitted network or a square numeric matrix, a measure is not built in and `centrality_fn` does not supply it, `centrality_fn` returns something other than a named list of one numeric value per node (or names that are not the node labels), or a group holds within- and between-person networks. `plot()` raises `psychnet_bad_input` for a `measures` entry that is not a measure column, `psychnet_plot_skipped` (warning) when it skips a measure without finite values, and `psychnet_nothing_to_plot` (an error) when no requested measure has one.

References

Robinaugh, D. J., Millner, A. J., & McNally, R. J. (2016). Identifying highly influential nodes in the complicated grief network. *Journal of Abnormal Psychology*, 125(6), 747-757. doi:10.1037/abn0000181

Opsahl, T., Agneessens, F., & Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3), 245-251. doi:10.1016/j.socnet.2010.03.006

See Also

`net_boot()` and `net_stability()` for the accuracy and stability of centralities; `net_bridge()`; `net_edge_betweenness()`; `net_predict()` for node predictability.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
cent <- net_centralities(ebic_glasso(cor_matrix = S, n = 250))
cent
plot(cent)
plot(cent, type = "line")
```

---

net_clustering *Compute weighted clustering coefficients of the nodes*

Description

`net_clustering()` computes weighted clustering coefficients for each node of a network. The clustering coefficient of a node measures how strongly its neighbours are connected to each other: in a psychometric network, a node with high clustering lies within a closely related group of variables, whereas a node with low clustering links variables that are otherwise unrelated.

Several generalizations of the clustering coefficient to weighted networks exist, and the function returns four of them: that of Watts and Strogatz (1998), which uses only the presence of edges; of Zhang and Horvath (2005), which uses the edge weights; of Onnela et al. (2005), which uses the geometric mean of the weights in each triangle; and of Barrat et al. (2004), which weights triangles by node strength. For the first three it also returns the signed versions of Costantini and Perugini (2014), in which a triangle with one or three negative edges lowers the coefficient.

It returns one row per node, with one column per coefficient.

For a summary of clustering at the level of the whole network, compared with random graphs, use `net_smallworld()`; for other node measures, use `net_centralities()`.

Usage

```
net_clustering(x, labels = NULL)
```

Arguments

x	An undirected fitted network (psychnet) or a symmetric weighted adjacency matrix.
labels	Optional node labels for a bare matrix, one unique label per node.

Details

Edge weights outside [-1, 1] are first divided by the largest absolute weight. A node with fewer than two neighbours closes no triangle and has coefficient 0. The Barrat coefficient uses absolute weights.

Value

A data frame of class psychnet_clustering, one row per node, with columns node, clustWS, signed_clustWS, clustZhang, signed_clustZhang, clustOnnela, signed_clustOnnela and clustBarrat.

Conditions

psychnet_bad_input when x is directed, is not a finite symmetric square matrix, or labels is invalid.

References

Costantini, G., & Perugini, M. (2014). Generalization of clustering coefficients to signed correlation networks. *PLoS ONE*, 9(2), e88669. doi:10.1371/journal.pone.0088669

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393, 440-442. doi:10.1038/30918

Zhang, B., & Horvath, S. (2005). A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1), Article 17. doi:10.2202/15446115.1128

Onnela, J.-P., Saramaki, J., Kertesz, J., & Kaski, K. (2005). Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, 71(6), 065103. doi:10.1103/PhysRevE.71.065103

Barrat, A., Barthelemy, M., Pastor-Satorras, R., & Vespignani, A. (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11), 3747-3752. doi:10.1073/pnas.0400087101

See Also

net_smallworld() for the small-world index; net_centralities() for other node measures.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
net_clustering(ebic_glasso(cor_matrix = S, n = 400))
```

`net_compare` *Compare the networks of two groups with a permutation test*

Description

`net_compare()` tests whether the networks of two groups differ, using the Network Comparison Test (van Borkulo et al., 2022). It estimates an EBIC graphical lasso network in each group and compares them on three statistics: M, the largest difference in any edge weight; S, the difference in global strength; and E, the difference in each edge.

The groups can be, for example, patients and controls, or the same people at two time points. Networks estimated in two groups differ in their sample estimates even when the population networks are identical, so the test compares the observed differences with those produced by assigning the observations to the two groups at random. M tests network structure invariance and is the largest absolute difference in any edge weight; S tests global strength invariance and is the absolute difference in the sum of the absolute edge weights; E tests the invariance of each edge and is the absolute difference in that edge's weight.

The null distributions of M, S and E are obtained by re-estimating both networks after each of `iter` random reassignments of the rows to the two groups. With `paired = TRUE` the rows are pairs of measurements of the same unit, and a permutation swaps the two members of each pair with probability one half. A p-value is the share of permutations, with the observed data counted as one of them, whose statistic is at least as large as the observed one. Options compare signed rather than absolute weights in S (`abs = FALSE`, a test of global expected influence), compare the presence of edges only (`weighted = FALSE`), and adjust the per-edge p-values for multiple testing (`p_adjust`).

It returns the two global tests, read with `summary()`, and the per-edge tests, read with `as.data.frame()`.

The test compares partial-correlation networks, so both networks are always EBIC graphical lassos (`ebic_glasso()`). For differences between edges or between centralities within one network, use `difference_test()`; for the accuracy of each network on its own, use `net_boot()`.

Usage

```
net_compare(
  data1,
  data2 = NULL,
  iter = 1000L,
  gamma = 0.5,
  paired = FALSE,
  abs = TRUE,
  weighted = TRUE,
  p_adjust = "none",
  na_method = c("pairwise", "listwise")
)
```

```
## S3 method for class 'psychnet_nct'
summary(object, alpha = 0.05, ...)

## S3 method for class 'psychnet_nct'
as.data.frame(x, row.names = NULL, optional = FALSE, alpha = 0.05, ...)

## S3 method for class 'psychnet_nct'
print(x, n = Inf, alpha = 0.05, ...)

## S3 method for class 'psychnet_nct'
plot(x, type = c("strength", "structure", "edges"), alpha = 0.05, ...)
```

Arguments

data1, data2	Data frames or matrices with the same numeric variables in the same order, one row per observation, holding finite values or NA. Alternatively, data1 can be a psychnet_group estimated from a data set, with data2 naming the two levels to compare as they appear in the grouping column (e.g. c(1, 4); default: the two levels of a two-level group); or data1 and data2 can be two member networks of one psychnet_group, which keep their rows and are compared as that group's levels. A network estimated on its own keeps no data and is refused.
iter	Number of permutations. Default 1000.
gamma	EBIC hyperparameter of the graphical lasso. Default 0.5. For a group, the group's own value is used unless gamma is given.
paired	If TRUE, row i of data1 and row i of data2 are measurements of the same unit. Default FALSE, except for a group built with psychnet(within =), whose levels are compared paired by person unless paired = FALSE; persons observed in only one level are left out.
abs	If TRUE (default), S compares the sums of absolute edge weights (global strength); if FALSE, the sums of signed edge weights (global expected influence). M and E always use absolute differences.
weighted	If FALSE, both networks are reduced to the presence or absence of each edge (1 or 0) before comparison. Default TRUE.
p_adjust	Multiplicity adjustment for the per-edge p-values, any method of stats::p.adjust(). Default "none". M and S are not adjusted.
na_method	"pairwise" (default) or "listwise" handling of missing values; see Details. For a group, the group's own setting is used unless na_method is given.
alpha	For the methods: significance level for the significant column, the * marks and the colouring of edges. Default 0.05.
...	Unused.
x, object	A psychnet_nct object.
row.names, optional	Ignored.
n	For print(): number of edges to list, largest difference first. Default Inf (all).

`type` For `plot()`: `"strength"` (default) or `"structure"` draws the permutation distribution of S or M with the observed value and p-value; `"edges"` draws the observed difference of every edge.

Details

The two groups must hold the same variables in the same order; non-numeric columns, such as an identifier or a group label, are dropped first.

Missing values follow `na_method`. `"pairwise"` (default) keeps every row, permutes rows as usual, and estimates each network from the pairwise-complete correlation matrix, projected to the nearest positive definite correlation matrix, with the median number of jointly observed rows as the sample size, as `ebic_glasso()` does. `"listwise"` first drops incomplete rows; in a paired design a pair is dropped when either member is incomplete. Complete data give the same result under both.

A variable without variation in a group, or in a permuted split (a rare binary item can be constant in one), enters that network as an isolated node, as with every estimator in the package. The statistics are then defined in every permutation, so the test remains exact.

The statistics and the permutation scheme are those of `NetworkComparisonTest::NCT()`, and the letters M and S follow van Borkulo et al. (2022).

Value

An object of class `psychnet_nct`. `summary()` returns the two global tests as a data frame, one row per test, with columns `test` (`"network structure"`, and `"global strength"`, or `"global expected influence"` when `abs = FALSE`), `statistic` (`"M"`, `"S"`), `observed`, `p_value` and `significant` (`p_value < alpha`). `as.data.frame()` returns the edge tests, one row per node pair, with columns `from`, `to`, `weight_1`, `weight_2` (the edge in each group's network; 0 or 1 when `weighted = FALSE`), `difference` (statistic E), `p_value` (adjusted by `p_adjust`), `p_raw` (before adjustment) and `significant` (adjusted `p_value < alpha`).

The object also stores the two estimated networks, the permutation distributions of M, S and E, the global strength of each group, the number of rows used in each group, the group names (the argument names, or the group levels) and the settings. `print()` shows the two global tests, the global strength of each group and every edge, largest difference first, and returns `x` invisibly; `plot()` returns `x` invisibly.

Functions

- `summary(psychnet_nct)`: The two global tests, M and S, one row each.
- `as.data.frame(psychnet_nct)`: The per-edge tests, one row per node pair.
- `print(psychnet_nct)`: The global tests, the global strength of each group, and every edge, largest difference first.
- `plot(psychnet_nct)`: The permutation distribution of S or M with the observed value marked, or the observed difference of every edge (see `type`); significant edges are coloured and marked `*`.

Conditions

psychnet_bad_input for an invalid argument; when the groups' columns differ in name, order or number, or hold infinite values; when a group has fewer than three rows or paired groups differ in size; and, for groups, when the levels are not found or the two networks are not members of one psychnet_group. psychnet_degenerate_sample when, with na_method = "pairwise", two variables share fewer than two observed rows in a group or in a permuted split. psychnet_constant_variable (warning) when a variable is constant in an observed group; psychnet_estimator_mismatch (warning) when a group was estimated with another estimator or settings, or is compared with a gamma or na_method other than its own; psychnet_rows_dropped (warning) when a paired comparison of a group leaves out persons observed in only one level. psychnet_renamed on access to the fields M and S, which the object does not have; summary() reports both tests.

References

van Borkulo, C. D., van Bork, R., Boschloo, L., Kossakowski, J. J., Tio, P., Schoevers, R. A., Borsboom, D., & Waldorp, L. J. (2022). Comparing network structures on three aspects: A permutation test. *Psychological Methods*. doi:10.1037/met0000476

See Also

difference_test() for differences within one network; net_boot() for the accuracy of a network; ebic_glasso() for the estimator; psychnet_group for networks estimated per group.

Examples

```
set.seed(1)
# group a has a stronger chain than group b
a <- matrix(stats::rnorm(100 * 4), 100, 4) %*% chol(0.5^abs(outer(1:4, 1:4, "-")))
b <- matrix(stats::rnorm(100 * 4), 100, 4) %*% chol(0.2^abs(outer(1:4, 1:4, "-")))
colnames(a) <- colnames(b) <- paste0("V", 1:4)
# iter = 10 keeps the example fast (smallest p = 1/11); use iter >= 1000.
fit <- net_compare(a, b, iter = 10)
fit
print(fit, n = 3)          # the three largest edge differences
summary(fit)
as.data.frame(fit)
plot(fit)                  # permutation distribution of global strength
plot(fit, type = "edges")  # per-edge differences
```

---

net_crosswalk *Map the arguments of established network packages to psychnets*

Description

net_crosswalk() maps the arguments of qgraph, IsingFit, mgm and psychonetrics functions to their psychnets counterparts, as a table with one row per argument. It shows which function fits the same model, which argument corresponds to which, and where the defaults or the meaning of an argument differ, for analyses moved to this package from those packages.

The table covers `qgraph::EBICglasso()` (`ebic_glasso()`), `qgraph::cor_auto()` (`cor_auto()`), `qgraph::ggmModSelect()` (`ggm_modselect()`), `IsingFit::IsingFit()` (`ising_fit()`), `mgm::mgm()` (`mgm_fit()`) and `psychonetrics::panelgvar()` (`panel_gvar()`); `reference` selects one of them. Each row records whether the argument is identical, renamed, has a different default or a different meaning, or exists in only one of the two functions, with a short note.

The result is a data frame with one row per argument; nothing is fitted.

Intensive time-series models (`graphicalVAR`, `mlVAR`) are not covered by this package.

Usage

```
net_crosswalk(
 reference = c("all", "EBICglasso", "cor_auto", "ggmModSelect", "IsingFit", "mgm",
    "panelgvar")
)
```

Arguments

`reference`	The reference function to show: `"all"` (default), `"EBICglasso"`, `"cor_auto"`, `"ggmModSelect"`, `"IsingFit"`, `"mgm"` or `"panelgvar"`.

Value

A data frame with one row per argument and columns `reference` (the reference function, as `pkg::fn`), `psychnet` (the corresponding function in this package), `ref_arg` and `psychnet_arg` (the argument names, `"-"` when there is no counterpart), `status` (`"identical"`, `"renamed"`, `"default differs"`, `"semantics differ"`, `"reference only"` or `"psychnet only"`) and `note`.

Conditions

`psychnet_bad_input` when `reference` is not one of the listed values.

See Also

`psychnet()`, which also accepts the `qgraph` and `bootnet` method names.

Examples

```
net_crosswalk("EBICglasso")
net_crosswalk("IsingFit")
net_crosswalk("panelgvar")
```

net_edge_betweenness *Compute the edge betweenness of a network*

Description

net_edge_betweenness() computes the betweenness of each edge in a network. Edge betweenness describes the role of an edge in connecting the network: it counts how many of the shortest paths between pairs of nodes run through the edge. An edge with high betweenness links parts of the network that would otherwise be far apart, for example two groups of closely related symptoms.

The function finds shortest paths on the absolute edge weights, inverted to distances so that a strong association is a short step (with invert = FALSE the weights are used as distances). When two nodes are joined by several equally short paths, each path counts as its share, 1 / (number of such paths).

It returns one row per edge. The value is a count, as returned by igraph::edge_betweenness(), not a proportion: on the path a-b-c-d-e the four edges score 4, 6, 6 and 4.

For the betweenness of nodes, use net_centralities(); for the connections of nodes to predefined communities, use net_bridge().

Usage

```
net_edge_betweenness(x, invert = TRUE, labels = NULL)
```

Arguments

x	A fitted network (psychnet) or a square weighted adjacency matrix. A bare matrix that is not symmetric is treated as directed, the entry in row i and column j being the edge from node i to node j.
invert	If TRUE (default), edge weights are inverted to distances (strong association = short path); if FALSE, weights are distances.
labels	Optional node labels for a bare matrix, one unique label per node.

Value

A data frame with one row per edge and columns from, to and edge_betweenness (the number of shortest paths through the edge). An undirected network has one row per edge, counting each pair of nodes once; a directed network has one row per arc, counting ordered pairs.

Conditions

psychnet_bad_input when x is not a fitted network or a square numeric matrix, contains non-finite weights, or labels does not give one unique label per node.

See Also

net_centralities() for node betweenness and other measures; net_bridge() for bridge centrality.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
net_edge_betweenness(ebic_glasso(cor_matrix = S, n = 400))
```

`net_predict` *Compute how well each node is predicted by the network*

Description

`net_predict()` computes the predictability of each node in a network. The predictability of a node is the proportion of its variance explained by the nodes it is connected to (Haslbeck & Waldorp, 2018): where centrality describes how a node is connected, predictability describes how much of the node's variation its connections account for. It places the edges on an absolute scale: a node with low predictability depends largely on variables outside the network, which limits how far it can be changed through its neighbours.

For Gaussian graphical models, predictability is the explained variance (R^2), which follows from the estimated precision matrix without the data. For a relative-importance network (`relimp_network()`) it is the R^2 stored with the fit. For the nodewise models (`ising_fit()`, `ising_sampler()` and `mgm_fit()`) it is computed from the data by predicting each node from the others: R^2 for continuous nodes and, for binary nodes, the proportion of correct classifications (CC) and the normalized accuracy (nCC), the gain over always predicting the more frequent category as a share of the gain that was possible.

It returns one row per node, with the type of the node, the metric used and its value, and the raw accuracy of binary nodes.

`node_predictability()` returns the same values as a vector for plotting; `net_boot()` with `predictability = TRUE` gives bootstrap intervals for Gaussian graphical models.

Usage

```
net_predict(x, data = NULL, ...)
```

Arguments

`x`	A fitted network (psychnet), or a psychnet_group (one table per group).
`data`	The data the network was estimated from; required for the nodewise models (`ising`, `ising_sampler`, `mgm`), ignored otherwise.
`...`	Unused.

Details

For a Gaussian graphical model with precision matrix Θ and correlation matrix S, the predictability of node j is $R_j^2 = 1 - 1/(\Theta_{jj} S_{jj})$.

For the nodewise models, the columns of `data` are matched to the nodes by name (the original column names, then the node labels) when the names allow it, and otherwise by position; binary categories are coded as when the network was fitted, and a category not seen in fitting is refused. Incomplete rows are left out.

Value

A data frame with one row per node and columns `node`, `type` (`"gaussian"` or `"binary"`), `metric` (`"R2"` or `"nCC"`), `predictability` (the value of `metric`) and `accuracy` (CC for binary nodes, `NA` for Gaussian ones). For a psychnet_group, one table per group (see psychnet_result_group).

Conditions

`psychnet_bad_input` for a network without predictability (for example from `cor_network()` or `tmfg_network()`); for a nodewise model when `data` is missing, lacks columns for some nodes, contains a category not seen in fitting, or has no complete row; for a mixed graphical model with a categorical node of more than two levels; and for a group of within- and between-person networks.

References

Haslbeck, J. M. B., & Waldorp, L. J. (2018). How well do network models predict observations? On the importance of predictability in network models. *Behavior Research Methods*, 50(2), 853-861. doi:10.3758/s134280170910x

See Also

`node_predictability()`; `net_centralities()` for centrality; `net_boot()` for intervals.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
net_predict(ebic_glasso(cor_matrix = S, n = 250))
# Nodewise models need the data they were fitted to.
b <- dichotomize(SRL_GPT)
net_predict(ising_fit(b), data = b)
```

---

net_smallworld *Compute the small-world index of a network*

Description

`net_smallworld()` computes the small-world index sigma of a network. A small-world network combines the high clustering of a regular lattice with the short paths between nodes of a random graph, and Humphries and Gurney (2008) quantify this with sigma: the transitivity of the network relative to that of comparable random graphs, divided by its average shortest-path length relative to theirs, `sigma = (C / C_rand) / (L / L_rand)`. A value of sigma above 1 indicates small-world structure.

The function computes the index on the unweighted network, in which an edge is present when its weight is nonzero, and compares it with `n_rand` random graphs that keep the degree of every node, obtained by rewiring pairs of edges. Because the reference graphs are random, the result varies between calls unless `seed` is given.

It returns one row with sigma, the transitivity and average shortest-path length of the network, and their means over the random graphs.

For the clustering of each node, use `net_clustering()`.

Usage

```
net_smallworld(x, n_rand = 100L, seed = NULL)
```

Arguments

x	An undirected fitted network (psychnet) or a symmetric weighted adjacency matrix.
n_rand	Number of random reference graphs. Default 100.
seed	Optional seed for the random graphs. The caller's random number stream is restored afterwards. Default NULL (the current stream).

Details

Transitivity is the global clustering coefficient: three times the number of triangles divided by the number of connected triples. The average shortest-path length is taken over the pairs of nodes joined by a path. The index is undefined when the random graphs have no triangles or no paths.

Value

A one-row data frame of class psychnet_smallworld, with columns smallworldness (sigma; NA when undefined), transitivity, aspl (average shortest-path length), and transitivity_rand and aspl_rand (their means over the random graphs).

Conditions

psychnet_bad_input when x is directed, is not a finite symmetric square matrix of at least two nodes, or n_rand is not a positive whole number; psychnet_undefined_statistic (warning) when the random graphs have no triangles or no paths, so that sigma is NA.

References

Humphries, M. D., & Gurney, K. (2008). Network 'small-world-ness': A quantitative method for determining canonical network equivalence. *PLoS ONE*, 3(4), e0002051. doi:10.1371/journal.pone.0002051

See Also

net_clustering() for clustering per node.

Examples

```
S <- 0.4^abs(outer(1:8, 1:8, "-"))
net_smallworld(ebic_glasso(cor_matrix = S, n = 400), n_rand = 50, seed = 1)
```

net_split_reliability *Assess the split-half reliability of a network's edge weights*

Description

net_split_reliability() assesses the split-half reliability of a network's edge weights. Split-half reliability is the agreement between networks estimated on two random halves of the sample, which stand in for two independent samples from the same population; it shows how reproducible the estimated structure is at that sample size. Repeating the split many times gives the distribution of this agreement rather than a value that depends on one arbitrary split.

The function draws iter random splits, with a share split of the rows in the first half, estimates a network on each half, and compares the two edge-weight vectors by their correlation and by the mean, median and maximum absolute difference.

For each of these metrics it returns the mean, the standard deviation and the 2.5th and 97.5th percentiles across splits.

For the stability of the edge weights as cases are removed, use net_casedrop_reliability(); for the stability of centralities, use net_stability(); for bootstrap intervals of each edge, use net_boot().

Usage

```
net_split_reliability(
  data,
  method = "glasso",
  iter = 100L,
  split = 0.5,
  cor_method = c("pearson", "spearman", "kendall"),
  labels = NULL,
  estimator_args = list(),
  ...
)

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

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

Arguments

data	A data frame or matrix with one row per observation, or a psychnet_group estimated from a data set (assessed per group).
method	The estimator, as in psychnet(). Default "glasso".
iter	Number of random splits. Default 100.

split	Share of the rows in the first half, a number in (0, 1). Default 0.5. Each half must keep at least two rows.
cor_method	Correlation between the edge weights of the two halves: "pearson" (default), "spearman" or "kendall".
labels	Optional node labels, one per node.
estimator_args	Named list of arguments for the estimator. Use it for an estimator argument that shares a name with an argument of this function, e.g. estimator_args = list(cor_method = "spearman"); such names given in ... do not reach the estimator.
...	Further arguments for the estimator (net_split_reliability()); ignored by the methods.
x	A psychnet_reliability object.

Details

The edge-weight correlation of a split is 0 when exactly one half has a constant edge-weight vector (for example an empty network against a non-empty one), and undefined when both halves are constant (they agree, but no correlation exists) or fewer than two edges are available. Splits in which a metric is undefined are left out of its summary and counted in n_valid. Each half is passed to the estimator as the full data would be (see net_boot()); the splits are drawn before any network is fitted, so the result is reproducible with set.seed().

Value

A data frame of class psychnet_reliability, one row per metric ("mean_abs_dev", "median_abs_dev", "correlation", "max_abs_dev"), with columns metric, mean, sd, lower and upper (2.5th and 97.5th percentiles across splits) and n_valid (splits in which the metric was defined; the summaries are NA when it was defined in none). The values of every split are kept with the table for plot(), which draws a histogram of each metric across splits with its mean marked. print() and plot() return x invisibly. For a psychnet_group, the function returns one result per group.

Functions

- print(psychnet_reliability): The number of splits and the table of metrics.
- plot(psychnet_reliability): A histogram of each metric across splits.

Conditions

psychnet_bad_input for an invalid argument, including a split that leaves fewer than two rows in a half, and for a group that was not estimated from a data set; psychnet_estimation_failed when, without labels, the network cannot be estimated on the full data; psychnet_resample_failed (warning) when some halves cannot be estimated, naming the first error, and as an error when fewer than two splits succeed; psychnet_undefined_statistic (warning) when a metric is undefined in some splits; psychnet_constant_variable (warning) when a constant column is dropped.

See Also

net_casedrop_reliability() and net_stability() for case-dropping stability; net_boot() for bootstrap intervals.

Examples

```
# method and iter are chosen so the example runs quickly; a real analysis
# uses the defaults (method = "glasso", iter = 100).
sh <- net_split_reliability(SRL_Claude, method = "pcor", iter = 10)
sh
plot(sh)
```

---

node_predictability *Return node predictability as a vector for plotting*

Description

`node_predictability()` returns node predictability as a numeric vector in node order, limited to the range 0 to 1, as `cograph::splot()` expects for `pie_values`. Node predictability is the proportion of a node's variance explained by its neighbours in the network (see `net_predict()`), and is often drawn as a ring around each node of a network plot.

For binary nodes the value is the normalized accuracy (nCC). Use `net_predict()` for the full table, including the type of each node and the raw accuracy.

Usage

```
node_predictability(x, data = NULL)
```

Arguments

`x`	A fitted network (psychnet).
`data`	The data the network was estimated from; required for the nodewise models (`ising`, `ising_sampler`, `mgm`), ignored otherwise.

Value

A numeric vector with one value per node, in node order, named by node label, each value between 0 and 1.

Conditions

The conditions of `net_predict()`.

See Also

`net_predict()` for the table.

Examples

```
S <- 0.4^abs(outer(1:6, 1:6, "-"))
node_predictability(ebic_glasso(cor_matrix = S, n = 250))
```

panel_gvar *Estimate temporal, contemporaneous and between-person networks from panel data*

Description

`panel_gvar()` estimates temporal, contemporaneous and between-person networks from panel data, which follow many persons over a few measurement waves. It fits the panel graphical vector-autoregressive (panel-GVAR) model, which separates three kinds of association among the measured variables (Epskamp, 2020).

The **temporal** network is directed: an edge from A to B is the within-person effect of A at one wave on B at the next, given all other variables, and a self-loop is an autoregressive effect. The **contemporaneous** network holds the partial correlations among a person's deviations at the same wave that the temporal effects leave unexplained. The **between** network holds the partial correlations among the persons' stable means. Together they distinguish variables that predict one another over time within persons, that vary together within persons at the same wave, and that differ together between persons.

The model is fitted by maximum likelihood to long data (one row per person and wave) or wide data (one row per person). By default every edge is estimated and reported with its standard error, p-value, 95% Wald interval and a `significant` flag at `alpha`. `select = "prune"` removes the edges that are not significant and refits the model, then adds back removed edges that improve the fit (`stepup`). `select = "ebic"` estimates the three networks with a lasso penalty chosen by the extended Bayesian information criterion (EBIC), the panel counterpart of `ebic_glasso()`. The fitted model is checked against its optimality conditions (see Details), and `as.data.frame(x, what = "diagnostics")` reports the result.

The result is a `psychnet_panelgvar` object holding the three networks (psychnet_panelgvar).

The model needs at least three waves. For a network from cross-sectional data, use `psychnet()`; for networks of within-person deviations without a temporal model, `psychnet()` with `within`. The model is the one fitted by `psychonetrics::panelgvar()`.

Usage

```
panel_gvar(
  data,
  vars,
  id = NULL,
  wave = NULL,
  select = c("none", "prune", "ebic"),
  alpha = 0.01,
  adjust = "none",
  recursive = FALSE,
  stepup = TRUE,
  criterion = c("bic", "none"),
  gamma = 0.5,
  nlambda = 50L,
```

```
  lambda_min_ratio = 1e-04,
  refit = FALSE,
  standardize = c("none", "z", "z_per_wave"),
  labels = NULL,
  na_method = "listwise"
)
```

Arguments

data	A data frame or matrix: **long** (one row per person per wave; give `vars`, `id` and `wave`) or **wide** (one row per person; give `vars` as a design matrix).
vars	Long data: the names of the variable columns. Wide data: a character matrix of column names with one row per variable and one column per wave (the `psychonetrics` design); its row names become the labels.
id, wave	Long data only: the names of the person and wave columns. Neither may contain missing values or be one of `vars`, and each person has at most one row per wave. Waves are ordered by factor level order, by value for numbers, dates and times, numerically for character values that are all numbers, and alphabetically (with a warning) for any other character values; pass a factor to set the order.
select	Edge selection: `"none"` (default) estimates every edge and reports its significance; `"prune"` removes non-significant edges and refits (see Details); `"ebic"` estimates penalized networks with the penalty chosen by EBIC.
alpha	Significance level, in (0, 1), for the `significant` flag, the solid and dashed edges of `plot()`, and pruning. Default `0.01`, as in `psychonetrics::prune()`.
adjust	Multiplicity correction for the p-values tested in pruning, one of stats::p.adjust.methods. Default `"none"`. Used only with `select = "prune"`; the reported p-values and `significant` flags are always unadjusted.
recursive	With `select = "prune"`: repeat pruning until no edge has p > `alpha`. Default `FALSE`.
stepup	With `select = "prune"`: after pruning, add back removed edges that improve the fit (see Details). Default `TRUE`, because one-pass pruning can remove parameters that are each non-significant but jointly significant, such as an autoregression and correlated cross-lagged effects. `FALSE` gives pruning alone.
criterion	With `stepup`: `"bic"` (default) also requires BIC to decrease when an edge is added; `"none"` uses the significance tests only.
gamma, nlambda, lambda_min_ratio	With `select = "ebic"`: the EBIC hyperparameter (default `0.5`), the number of penalty values (default `50`, a whole number >= 2), and the smallest penalty as a fraction of the largest (default `1e-4`).
refit	With `select = "ebic"`: refit the selected networks by unpenalized maximum likelihood, so that standard errors and p-values are available. Default `FALSE` returns the penalized estimates, for which standard errors, p-values and `significant` are `NA`.
standardize	Rescaling before fitting: `"none"` (default), `"z"` (one mean and standard deviation per variable, pooled over waves), or `"z_per_wave"` (each variable z-scored

within each wave). "z_per_wave" removes changes in means and variances across waves, which the model would otherwise assume absent, and makes the edges standardized. Standardization uses the available values before listwise deletion.

labels — Optional node labels, unique and non-empty (default: vars for long data, the design row names for wide data).

na_method — Only "listwise" (default): a person with any missing value is dropped. "pairwise" is refused, because a pairwise covariance is not a likelihood for this model.

Details

Write y_{it} for the scores of person i at wave t. The model is $y_{it} = \mu + b_i + w_{it}$, with means μ shared by all waves, stable person deviations b_i with covariance Σ_b, and a stationary within-person process $w_{it} = Bw_{i,t-1} + \zeta_{it}$ with innovation covariance Σ_ζ. The temporal network is B (the edge from j to i is B_{ij}); the contemporaneous and between networks are the partial correlations implied by Σ_ζ and Σ_b. When the means drift across waves, standardize = "z_per_wave" removes the drift.

Pruning fixes to zero, in one pass, every temporal, contemporaneous and between edge whose Wald p-value (after adjust) exceeds alpha, and refits; recursive = TRUE repeats until every remaining edge is significant. The step-up search then frees the removed edge with the largest modification index while that index is significant at alpha, and keeps it when the likelihood-ratio test of the refit is significant at alpha and, with criterion = "bic", BIC decreases; the first rejected edge ends the search. This is psychonetrics' prune() followed by stepup().

With select = "ebic", the lasso penalty applies to the temporal (autoregressive effects included), contemporaneous and between parameters on the data scale, over nlambda log-spaced values from the smallest penalty that empties every network down to lambda_min_ratio times that value. The selected penalty minimizes $\mathrm{EBIC} = -2\log L + k\log n + 4\gamma k\log V$, with k the number of nonzero free parameters, n the number of persons and V the number of wave-by-variable columns. Penalized estimates below 1e-5 in absolute value on the data scale are set to zero before the certificate is computed.

The certificate is the largest absolute gradient of the likelihood over the free parameters or, for select = "ebic" without refit, the largest violation of the lasso optimality (Karush–Kuhn–Tucker) conditions. Pruned and penalized models are refitted or re-solved after selection, so the certificate refers to the networks returned. It is computed on standardized variables and does not depend on the variables' units. The likelihood is not convex: a value near zero shows a stationary point, and the fit is started from several points whose agreement is reported.

Value

An object of class c("psychnet_panelgvar", "netobject_group") (psychnet_panelgvar): a named list of three networks of class psychnet, temporal (directed), contemporaneous and between (undirected). as.data.frame() gives one row per edge with its standard error, p-value and Wald interval; as.data.frame(x, what = ...) gives the "parameters", "fit" indices, "variance" decomposition, "covariance" matrices, EBIC "path" or "diagnostics", whose score column is the certificate. summary() gives one row per network and plot() draws the three networks.

Conditions

psychnet_bad_input when data, vars, id or wave cannot be resolved (columns missing or duplicated, id and wave the same or among vars, missing persons or waves, a person with two rows in one wave), a variable is not numeric, there are fewer than two variables or fewer than three waves, there are no more complete persons than wave-by-variable columns, the stacked covariance is singular, a variable does not vary as standardize requires, labels is invalid, an argument is outside its allowed values, or na_method is not "listwise"; psychnet_no_converge (error) when no start reaches a feasible optimum, the fitted model is worse than its nested baseline, or no penalized model is feasible. Warnings: psychnet_no_converge when the score exceeds 1e-6 at an interior solution; psychnet_boundary when the contemporaneous or between covariance is near singular (partial correlations near +/-1, common with three waves), so estimates involving it are unreliable; psychnet_singular_information when some parameters are not identified at the estimate (their standard errors are NA); psychnet_approximate_se when a fit being pruned had a singular information and its unidentified parameters were pruned; psychnet_listwise_drop when persons are removed for missing values; psychnet_wave_order when character waves are ordered alphabetically; psychnet_ignored_argument when adjust is set without pruning.

References

Epskamp, S. (2020). Psychometric network models from time-series and panel data. *Psychometrika*, 85(1), 206-231. doi:10.1007/s11336020096973

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in Neural Information Processing Systems*, 23, 604-612.

See Also

psychnet_panelgvar for reading the fit; ebic_glasso() for cross-sectional data; psychnet() with within; certificate() for the certificates of the other estimators.

Examples

```
# srl_panel: 100 students x 4 weeks, simulated from a panel-GVAR in which
# effort in one week drives efficacy in the next.
fit <- panel_gvar(srl_panel,
                  vars = c("efficacy", "planning", "effort", "enjoyment",
                           "anxiety"),
                  id = "student", wave = "week")
fit
as.data.frame(fit)
as.data.frame(fit, what = "fit")
as.data.frame(fit, what = "variance")
```

---

pcor_network *Estimate a partial correlation network*

Description

pcor_network() estimates a partial correlation network, in which each pair of variables is joined by their correlation after controlling for all other variables in the network. An edge is therefore an association that the remaining variables do not account for, and for normally distributed data a partial correlation of zero means that the two variables are conditionally independent.

This is the unregularized Gaussian graphical model, in which every partial correlation is estimated and none is set to zero by the model itself. The partial correlations are computed from the inverse of the correlation matrix of raw data or of a supplied correlation matrix. alpha keeps only partial correlations that differ significantly from zero, with an optional multiple-comparison adjustment (adjust), and threshold removes weak ones.

pcor_network() returns the partial correlations as a network; the network involves no optimization and has no certificate.

With many variables relative to the sample size, partial correlations are estimated with large error, and with fewer observations than variables they are not identified. ebic_glasso() and ggm_modselect() then estimate a sparse network instead; cor_network() gives the marginal correlations. The network is the "pcor" estimator of bootnet.

Usage

```
pcor_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  threshold = 0,
  alpha = NULL,
  adjust = "none",
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

data	A numeric data frame or matrix with one row per observation. Not needed when cor_matrix is given. Ordered factors enter as their level codes; other non-numeric columns are dropped with a message, and constant or all-missing columns with a warning.
cor_matrix	A correlation matrix to use instead of data; a covariance matrix is rescaled to a correlation matrix. With alpha, n is then required.
n	Sample size of cor_matrix, needed only for the significance test: a single number >= 1, and a whole number >= 3 when alpha is set. Ignored when data is given.
cor_method	Correlation computed from data: "pearson" (default), "spearman", "kendall", or "auto" (polychoric and polyserial correlations for ordinal items; see cor_auto()).
threshold	Absolute edge weight below which an edge is set to zero: a single number >= 0. Default 0. A logical value (the threshold = TRUE rule of qgraph) is refused; use alpha for a significance rule.

alpha	Significance level in (0, 1): edges not significant at this level are set to zero. Default NULL, no test. A Kendall test needs raw data; partial Kendall correlations cannot be tested.
adjust	Multiple-comparison adjustment of the edge p-values, any method of stats::p.adjust(). Default "none".
na_method	Handling of missing values in data: "pairwise" (default) uses pairwise-complete correlations adjusted to the nearest positive-definite matrix; "listwise" drops incomplete rows. The two agree for complete data.
labels	Optional node labels, one unique label per node of the network.

Details

With Θ the inverse correlation matrix, the partial correlation of variables i and j is $-\Theta_{ij}/\sqrt{\Theta_{ii}\Theta_{jj}}$. The significance test is described in cor_network(); for p variables it has $n - p$ degrees of freedom. It needs a single sample size, so with pairwise data whose pairs have different numbers of observations alpha is refused.

When the correlation matrix is singular, as with fewer observations than variables or with collinear or duplicate columns, the nearest positive-definite matrix is inverted so that a network is still returned, and a warning says that its edges are not interpretable.

Value

A fitted network of class psychnet (psychnet) whose edge weights are partial correlations. as.data.frame() returns one row per edge with columns from, to and weight; certificate() reports that the network has no certificate. The object also stores the precision matrix (precision), the correlation matrix (cor_matrix), the sample size (n_eff), the missing-data handling used (na_method), the observations per pair when estimated from data (n_pair) and, with alpha, the unadjusted p-values (p_values). Nodes keep the column order of the input.

Conditions

psychnet_singular (warning) when the correlation matrix is singular or nearly so; psychnet_degenerate_sample (error, also of class psychnet_bad_input) when the sample is too small for the test ($n \leq p$); psychnet_bad_input when an argument is out of range, threshold is logical, alpha is combined with Kendall correlations or with unequal pairwise sample sizes, cor_matrix is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, n is missing with cor_matrix and alpha, data has fewer than two usable variables, or labels do not match the nodes; psychnet_constant_variable (warning) when a constant or all-missing column is dropped.

See Also

cor_network(); ebic_glasso() and ggm_modselect() for sparse networks; psychnet(), which routes method = "pcor" here; psychnet for the result.

Examples

```
x <- matrix(stats::rnorm(200 * 4), 200, 4)
pcor_network(x)
pcor_network(x, alpha = 0.05, adjust = "holm")
```

`plot.psychnet` *Fitted psychometric network*

Description

A `psychnet` object holds a fitted psychometric network; every estimator in the package returns its network in this form. The network consists of nodes, which are the variables, and weighted edges, which are the estimated associations: partial correlations for the Gaussian graphical models, correlations for `cor_network()`, symmetrized regression weights for the Ising and mixed graphical models, and directed importance shares for `relimp_network()`.

`print()` reports the estimator, the numbers of nodes and edges, the selected penalty and the certificate; `summary()` adds the range of the edge weights; `as.data.frame()` returns the edges as a table; and `plot()` draws the network with the `cograph` package. The certificate is read with `certificate()`.

Analyses of a fitted network, such as `net_centralities()`, `net_boot()` and `net_stability()`, return their own result classes, documented on their pages. Networks estimated per group are collected in a psychnet_group.

Usage

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

## S3 method for class 'psychnet'
x$name

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

## S3 method for class 'psychnet'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., include_zero = FALSE)

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

Arguments

`x`, `object`	A `psychnet` object.
`...`	For `plot()`, passed to `cograph::splot()`; ignored by the other methods.
`name`	For `$`: a field name, or one of the earlier names listed in Details.
`row.names`, `optional`	Ignored.
`include_zero`	For `as.data.frame()`: keep absent (zero-weight) edges. Default `FALSE`.

Details

The object is a list that follows the network object of the cograph and Nestimate packages, so it can be passed to cograph::splot() directly. It stores the weighted adjacency matrix with a zero diagonal (weights), the node table (nodes, with id, label and name), the edge table in the cograph format with integer node indices (edges), whether the network is directed (directed), the estimator (method) and the sample size (n), together with fields specific to each estimator, such as the precision matrix (precision), the selected penalty (lambda), the EBIC hyperparameter (gamma), the correlation matrix (cor_matrix), the certificate (kkt), the Ising thresholds (thresholds) and the node types of a mixed model (types). Field access with $ also recognises the names used by earlier versions: graph (the weights), labels, n_nodes, n_edges and n_obs.

Value

as.data.frame() returns a data frame with one row per edge (each node pair once for an undirected network, each ordered pair for a directed one) and columns from, to (node labels) and weight. print() returns x invisibly. summary() prints the network with its edge-weight range and returns the edge table invisibly. plot() returns the value of cograph::splot() invisibly. $ returns the requested field, or NULL when the name is neither a field nor an earlier name.

Functions

- plot(psychnet): Draws the network with cograph::splot(), passing ... on; needs the suggested package cograph.
- $: Field access, including the legacy aliases.
- print(psychnet): The estimator, the node and edge counts, the selected penalty and the certificate (see certificate()).
- as.data.frame(psychnet): The tidy edge list, one row per edge.
- summary(psychnet): The printout plus the edge-weight range.

Conditions

psychnet_bad_input from plot() when the cograph package is not installed.

See Also

psychnet() and the estimators it routes to; certificate(); psychnet_group for networks per group.

Examples

```
fit <- pcor_network(SRL_GPT)
fit
summary(fit)
as.data.frame(fit)
as.data.frame(fit, include_zero = TRUE)
if (requireNamespace("cograph", quietly = TRUE)) plot(fit)
```

`plot.psychnet_stability` *Assess the stability of centralities when cases are dropped*

Description

`net_stability()` assesses how stable the centralities of a network are when cases are dropped, using the case-dropping bootstrap (Epskamp, Borsboom & Fried, 2018). The network is re-estimated on subsets of the data from which an increasing proportion of cases has been removed, the centralities of each subset are correlated with those of the full sample, and the correlation-stability coefficient (CS-coefficient) summarizes the result: the largest proportion of cases that can be dropped while the subset centralities still correlate highly with the full-sample centralities in nearly all subsets.

Centrality indices are often used to rank the nodes of a psychometric network, for example to identify the symptoms most strongly connected to the others, and such a ranking is informative only if a similar ranking would be found in another sample. Epskamp, Borsboom and Fried (2018) suggest that the CS-coefficient should not be below 0.25, and preferably above 0.5, before differences in centrality are interpreted.

The function draws `iter` subsets without replacement at every proportion in `drop_prop`, re-estimates the network on each, and correlates each subset's centralities with the full-sample centralities, for every measure in `measures`. The CS-coefficient of a measure is the largest proportion at which the share of subsets whose correlation exceeds `threshold` itself exceeds `certainty`, or 0 when no proportion qualifies.

It returns the summary by drop proportion, the CS-coefficient of each measure and the correlation of every subset, read with `as.data.frame()`.

For the stability of the edge weights rather than the centralities, use `net_casedrop_reliability()`; for intervals around each edge and centrality, use `net_boot()`.

Usage

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

net_stability(
  data,
  method = "glasso",
  measures = c("strength", "expected_influence"),
  centrality_fn = NULL,
  drop_prop = seq(0.1, 0.9, by = 0.1),
  iter = 100L,
  threshold = 0.7,
  certainty = 0.95,
  cor_method = c("pearson", "spearman"),
  labels = NULL,
  estimator_args = list(),
```

```
  ...
)

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

## S3 method for class 'psychnet_stability'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  what = c("table", "cs", "correlations")
)
```

Arguments

x	A psychnet_stability object.
...	Further arguments for the estimator (net_stability()); ignored by the methods.
data	A data frame or matrix with one row per observation, or a psychnet_group estimated from a data set (assessed per group).
method	The estimator, as in psychnet(). Default "glasso".
measures	Centrality measures to assess. Default c("strength", "expected_influence"); any measure accepted by net_centralities() can be added.
centrality_fn	Optional function supplying measures that are not built in (see net_centralities()).
drop_prop	Proportions of cases to drop, each in (0, 1). Default seq(0.1, 0.9, by = 0.1). Each must remove at least one case and keep at least two.
iter	Number of subsets per proportion. Default 100.
threshold	Correlation a subset must exceed to count as stable, a number in (0, 1). Default 0.7.
certainty	Share of stable subsets that must be exceeded, a number in (0, 1). Default 0.95.
cor_method	Correlation between subset and full-sample centralities: "pearson" (default, as in bootnet) or "spearman" (stability of the ranking).
labels	Optional node labels, one per node.
estimator_args	Named list of arguments for the estimator. Use it for an estimator argument that shares a name with an argument of net_stability(), e.g. estimator_args = list(threshold = TRUE).
row.names, optional	Ignored.
what	For as.data.frame(): "table" (default), "cs" or "correlations".

Details

Both comparisons in the CS rule are strict, as in `bootnet::corStability()`. The correlation of a subset is 0 when either centrality vector is constant across nodes, and undefined when fewer than two values are available. An undefined correlation, or a subset on which the network could not be estimated, counts as not exceeding `threshold`, so such subsets never raise the CS-coefficient (`bootnet` instead leaves undefined correlations out). A measure whose full-sample centrality is constant has a CS-coefficient of 0.

Each subset is passed to the estimator as the full data would be (see `net_boot()`). The subsets are drawn before any network is fitted, so the result is reproducible with `set.seed()`.

Value

`net_stability()` returns an object of class `psychnet_stability`. Its tables are read with `as.data.frame()`:

- `as.data.frame(x)` (`what = "table"`): one row per measure and drop proportion, with columns `measure`, `drop_prop` (nominal), `actual_drop` (the proportion dropped after rounding to whole cases), `mean_cor` and `sd_cor` (over the defined correlations), `prop_above` (share of subsets with a correlation above `threshold`) and `n_valid` (subsets with a defined correlation).
- `what = "cs"`: one row per measure, with columns `measure` and `cs`.
- `what = "correlations"`: one row per measure and subset, with columns `measure`, `drop_prop`, `subset` and `correlation` (`NA` when undefined or when the subset could not be estimated), from which the CS-coefficient can be recomputed.

`print()` shows the estimator, the number of subsets per proportion and the CS-coefficient of each measure; `print()` and `plot()` return `x` invisibly. For a psychnet_group, `net_stability()` returns one result per group (see psychnet_result_group).

Functions

- `plot(psychnet_stability)`: The mean correlation with the full-sample centralities against the proportion of cases dropped, one line per measure (each with its own colour, line type and point shape) with a band of +/- 1 SD, the `threshold` as a dashed line, and each measure's CS-coefficient in the legend.
- `print(psychnet_stability)`: The estimator, the subsets per proportion and the CS-coefficient of each measure.
- `as.data.frame(psychnet_stability)`: The summary by drop proportion (`what = "table"`), the CS-coefficient of each measure (`what = "cs"`), or the correlation of every subset (`what = "correlations"`).

Conditions

`psychnet_bad_input` for an invalid argument, including drop proportions that remove no case or keep fewer than two, and for a group that was not estimated from a data set; `psychnet_estimation_failed` when the network cannot be estimated on the full data; `psychnet_resample_failed` (warning) when some subsets cannot be estimated, naming the first error, and as an error when none can; `psychnet_constant_centrality` (warning) when a full-sample centrality is constant across nodes; `psychnet_constant_variable` (warning) when a constant column is dropped.

References

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50(1), 195-212. doi:10.3758/s13428017-08621

See Also

`net_casedrop_reliability()` for the edge weights; `net_boot()` for bootstrap intervals; `net_centralities()` for the measures.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(200 * 5), 200, 5) %*% chol(0.4^abs(outer(1:5, 1:5, "-")))
colnames(x) <- paste0("V", 1:5)
# method = "pcor" and a small iter keep this example fast; a real analysis
# uses the "glasso" default and iter = 100.
cs <- net_stability(x, method = "pcor", drop_prop = c(0.3, 0.6), iter = 10)
cs                              # the CS-coefficient of each measure
as.data.frame(cs, what = "cs")  # the same, as a data frame
as.data.frame(cs)               # correlations by drop proportion
plot(cs)                        # mean correlation by drop proportion
```

`psychnet` *Estimate a psychometric network with any of the package's estimators*

Description

`psychnet()` estimates a psychometric network with any of the cross-sectional estimators, selected by `method`. A psychometric network represents variables, such as questionnaire items, symptoms or behaviours, as nodes and their conditional or marginal associations as edges.

The estimators differ in what an edge means: partial-correlation networks describe the association between two variables after controlling for all others, regularized networks retain only the associations the data support, and Ising and mixed graphical models apply the same idea to binary and mixed data. Besides `method`, the arguments select the variables (`vars`), estimate one network per group (`group`), scale each variable within persons first (`within`), or treat `data` as an event log (`actor`, `action`, `session`, `time`), which is converted to action frequencies per occasion before a network is estimated.

`psychnet()` returns a fitted network; with `group`, one network per group, and for some event data a pair of within-actor and between-actor networks.

Each method can also be called directly, which exposes its full set of options. Panel data measured in a few waves are modelled by `panel_gvar()`, which `psychnet()` does not route. The method-specific pages describe each model: `ebic_glasso()`, `ggm_modselect()`, `huge_network()`, `tmfg_network()`, `logo_network()`, `relimp_network()`, `cor_network()`, `pcor_network()`, `ising_fit()`, `ising_sampler()` and `mgm_fit()`.

Usage

```
psychnet(
  data,
 method = c("glasso", "cor", "pcor", "ising", "mgm", "huge", "ggm", "tmfg", "logo",
    "relimp", "ising_sampler"),
  threshold = 0,
  gamma = NULL,
  labels = NULL,
  vars = NULL,
  group = NULL,
  within = NULL,
  source = c("auto", "data", "eventdata"),
  actor = NULL,
  action = "Action",
  session = NULL,
  time = NULL,
  compute_sessions = TRUE,
  time_threshold = 900,
  id = NULL,
  standardize = TRUE,
  ...
)
```

Arguments

data	A data frame or matrix with one row per observation and one column per variable, or a long event log (one row per event).
method	The estimator, a single name: "glasso" (default), "ggm", "huge", "tmfg", "logo", "relimp", "cor", "pcor", "ising", "ising_sampler" or "mgm", or a qgraph/bootnet alias. Event data accept "glasso", "pcor", "ggm", "huge", "tmfg" and "logo".
threshold	Absolute edge weight below which edges are set to zero, for "cor", "pcor", "glasso", "huge", "ggm" and "logo": a single number >= 0 (default 0). For "mgm", give its rule ("LW", "HW" or "none"); the default 0 leaves the mgm_fit() default in place.
gamma	EBIC hyperparameter for "glasso", "huge", "ggm", "ising" and "mgm". NULL (default) keeps each method's own default.
labels	Optional node labels, one per node.
vars	The variables to use as nodes (default: all). A name range (motivation:regulation), an index range (3:9), names (c(joy, fear) or c("joy", "fear")) or a single name, as in the select argument of subset(). For an event log the actions are the nodes.
group	Optional grouping column(s). One network is estimated per level and a psychnet_group is returned.
within	Optional person-identifier column. Each variable is centred and scaled within persons over all rows before estimation (each person's mean subtracted and the

result divided by the person's standard deviation, as `mlVAR::mlVAR(scaleWithin = TRUE)`), and the person column is not a node. With `group`, rows are matched by person so that `net_compare()` can pair two levels (`paired = TRUE`). Without `group`, a person's rows are pooled and are not independent. Numeric tables only.

`source` Kind of input: `"auto"` (default; an event log when `actor` or `id` is given, otherwise a numeric table), `"data"` or `"eventdata"`.

`actor`, `action`, `session`, `time`
Event-log columns: the actor, the action (default `"Action"`), an optional session within actor, and an optional time used only to split actors into sessions. Giving `actor` selects event data.

`compute_sessions`, `time_threshold`
Whether to split each actor into sessions where the gap in `time` exceeds `time_threshold` (defaults `TRUE` and 900, i.e. 15 minutes for times in seconds). Has no effect without `time`.

`id` Actor column of an event log already reduced to a numeric table with one row per occasion.

`standardize` For event data with several occasions per actor: `TRUE` (default) centres each actor's occasions and fits one network; `FALSE` returns separate within-actor and between-actor networks.

`...` Further arguments for the estimator, such as `cor_method` for the correlation-based methods, `npn` for `"huge"`, `rule` for the Ising methods and `alpha` for `"cor"`, `"pcor"` and `"ising_sampler"`. For an event table, `na_method = "listwise"` removes incomplete rows before centring; `"pairwise"` (default) uses the available values.

Details

`method` accepts the package's short names and the names used by `qgraph` and `bootnet` (`"EBICglasso"`, `"ggmModSelect"`, `"TMFG"`, `"LoGo"`, `"IsingFit"`, `"IsingSampler"`), ignoring case and separators; all resolve to the same estimators, and the fitted network records the short name. `net_crosswalk()` maps the arguments of the reference packages to those of this package.

Value

A fitted network of class `psychnet` (psychnet); with `group`, a psychnet_group with one network per level; for event data with several occasions per actor and `standardize = FALSE`, a psychnet_multilevel pair of within-actor and between-actor networks. Edges are read with `as.data.frame()`.

Conditions

`psychnet_bad_input` when `method` has several values or is unknown, `vars` names a missing column, `within` does not name one column of a numeric table, or event data are combined with a method other than the Gaussian ones; `psychnet_ignored_argument` (warning) when `threshold` or `gamma` is set for a method that does not use it. With `group`, the conditions listed in psychnet_group. The chosen estimator raises its own conditions.

See Also

psychnet for the fitted object; certificate() to check a regularized fit; net_centralities(), net_boot() and net_compare() for analyses of fitted networks; event_frequencies() for the event-log conversion.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(200 * 5), 200, 5)
colnames(x) <- c("joy", "fear", "calm", "anger", "trust")
psychnet(x, method = "glasso")
psychnet(x, method = "pcor", vars = joy:anger)     # name range
psychnet(x, method = "glasso", vars = 1:3)         # index range
psychnet(x, method = "EBICglasso")                 # qgraph alias, same result

ev <- data.frame(
  Actor  = rep(paste0("s", 1:30), each = 20),
  Action = sample(c("read", "quiz", "note", "watch"), 600, replace = TRUE))
psychnet(ev, actor = "Actor", action = "Action")   # event data, auto-detected

d <- data.frame(g = rep(c("A", "B"), each = 100),
                matrix(stats::rnorm(200 * 4), 200, 4,
                       dimnames = list(NULL, paste0("V", 1:4))))
psychnet(d, method = "glasso", group = "g")         # one network per level

# Within-person networks, one per week, scaled within each student over all
# four weeks; compare two weeks (paired by student automatically).
nets <- psychnet(srl_panel, vars = efficacy:anxiety, group = "week",
                 within = "student")
nets
set.seed(1)
net_compare(nets, c(1, 4), iter = 3)  # iter >= 1000 in practice
```

---

psychnet_group *Networks estimated separately for each group*

Description

A psychnet_group holds networks estimated separately for each group: one fitted network per level of the grouping variable, estimated with the same method and settings. When the same variables are measured in several groups, such as clinical and control samples, cohorts or occasions, these networks show how the association structure differs between them.

It is returned by psychnet() with group. print() lists the groups with their numbers of nodes and edges and their sample sizes, summary() returns these counts as a table, and as.data.frame() stacks the edges of all groups; cograph::splot() draws the networks in a grid. The analysis functions net_centralities(), net_predict(), net_boot(), net_stability(), net_casedrop_reliability() and net_split_reliability() accept the object and return one result per group (a psychnet_result_group); net_compare() tests the difference between two of its levels.

Resampling and comparison assume independent rows. Groups formed from several occasions per actor of an event log are refused by these functions, because they would need resampling at the actor level.

Usage

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

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

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

Arguments

x	A psychnet_group object.
...	Ignored.
row.names, optional	Ignored.
object	A psychnet_group object.

Details

Groups are ordered by the levels of the grouping variable when it is a factor, so its first level is group 1 in net_compare(), and otherwise by first appearance. Each network keeps the rows it was estimated from, so two members can be passed to net_compare() directly.

Value

print() returns x invisibly, after printing one row per group with its numbers of nodes and edges and its sample size.

as.data.frame() returns a data frame with one row per edge per group and columns group, from, to and weight.

summary() returns a data frame with one row per group (or per group and level when a group holds within-actor and between-actor networks) and columns group, nodes, edges and mean_abs_weight.

Conditions

psychnet_rows_dropped (warning) when rows with a missing group value are dropped; psychnet_bad_input when a group column is missing, a level is the empty string, there are fewer than 2 levels, or a level of a numeric table has fewer than 3 rows; psychnet_small_sample (warning) when a level has no more rows than variables.

See Also

psychnet() to estimate the networks; psychnet for a single network; net_compare() to compare two groups.

Examples

```
d <- rbind(data.frame(SRL_GPT, source = "GPT"),
           data.frame(SRL_Claude, source = "Claude"))
nets <- psychnet(d, method = "glasso", group = "source")
nets
summary(nets)
as.data.frame(nets)
```

---

psychnet_multilevel *Within-actor and between-actor networks from event data*

Description

A psychnet_multilevel object holds a pair of Gaussian graphical networks estimated from event data with several occasions per actor: a within-actor network and a between-actor network. psychnet() returns it for event data (an event log, or a table of action frequencies with one row per occasion and an id column) with several occasions per actor and standardize = FALSE, estimating both networks with the chosen method.

When the same actors, such as students in a learning environment, contribute several occasions (sessions), the association between the frequencies of two actions has two sources. Within actors, it shows whether an actor who performs one action more often than usual on an occasion also performs the other more often. Between actors, it shows whether actors who perform one action more often on average also perform the other more often. The two can differ in size and even in sign. The within network is estimated from each occasion's deviations from its actor's mean action frequencies, with the pooled within-actor degrees of freedom (occasions minus actors) as sample size; the between network from the actors' mean frequencies, with the number of actors as sample size.

print() reports the numbers of actors, occasions, nodes and edges, summary() and as.data.frame() return data frames, and plot() draws the two networks side by side.

With standardize = TRUE (the default), psychnet() returns only the within-actor network, as a single psychnet. For persons measured at fixed waves, with effects from one wave to the next, use panel_gvar().

Usage

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

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

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

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

Arguments

x, object	A psychnet_multilevel object.
...	Passed to cograph::splot() by plot(); ignored by the other methods.
row.names, optional	Ignored.

Value

print() returns x invisibly, after printing the numbers of actors and occasions and each network's numbers of nodes and edges.

as.data.frame() returns one row per edge of the two networks, with columns level ("within" or "between"), from, to and weight.

summary() returns a data frame with one row per level ("within", "between") and columns level, nodes, edges, mean_abs_weight (0 for a network without edges) and n (the sample size used at that level).

plot() draws the two networks side by side with cograph::splot() and returns x invisibly.

Conditions

psychnet_bad_input from plot() when the cograph package is not installed.

See Also

psychnet() for fitting; event_frequencies() for the conversion of an event log; psychnet for the two networks.

Examples

```
set.seed(1)
ev <- data.frame(
  Actor   = sample(paste0("s", 1:40), 1200, replace = TRUE),
  Session = sample(1:3, 1200, replace = TRUE),
  Action  = sample(c("read", "quiz", "note", "watch"), 1200, replace = TRUE))
# Actions drawn at random, so neither network is expected to have edges.
ml <- psychnet(ev, actor = "Actor", session = "Session", standardize = FALSE)
ml
summary(ml)
as.data.frame(ml)
if (requireNamespace("cograph", quietly = TRUE)) plot(ml)
```

psychnet_panelgvar *Read the networks, parameters and fit of a panel graphical VAR*

Description

A psychnet_panelgvar object holds the three networks of a panel graphical vector-autoregressive model fitted by panel_gvar(): the directed temporal network of lag-1 within-person effects, the contemporaneous network of within-person partial correlations at the same wave, and the between network of partial correlations among the persons' stable means. Its methods read the fitted model as tidy tables, summarize it, and draw the three networks side by side.

as.data.frame() gives the edges by default and, through what, the full parameter table, the fit indices, the split of each variable's variance into between- and within-person parts, the innovation and between-person covariances, the EBIC penalty path, or the diagnostics that include the certificate. summary() counts the edges of each network, and plot() draws the networks with significance shown by line type.

as.data.frame() and summary() return data frames; print() and plot() return the object invisibly.

Usage

```
## S3 method for class 'psychnet_panelgvar'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
 what = c("edges", "parameters", "fit", "diagnostics", "variance", "covariance", "path"),
  include_zero = FALSE
)

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

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

## S3 method for class 'psychnet_panelgvar'
plot(x, edge_labels = TRUE, ...)
```

Arguments

x, object	A psychnet_panelgvar object from panel_gvar().
row.names, optional	Ignored.

`...` For `plot()`, passed to `cograph::splot()`; named arguments override the plotting defaults, and unique abbreviations are accepted. Ignored by the other methods.

`what` For `as.data.frame()`, the table to return:

- `"edges"` (default) one row per estimated edge of the three networks, with standard error, p-value and 95% Wald interval; the autoregressive effects are the temporal rows with `from == to`.
- `"parameters"` every model parameter, including the means, the scale parameters and the edges fixed to zero.
- `"fit"` one row of fit indices.
- `"diagnostics"` one row: the number of persons and waves, the certificate (`score`), whether the solution is on a boundary, whether standard errors are available, and how many optimizer starts reached the best optimum. The starts are those of the full model, of the penalized solution at the selected penalty with `select = "ebic"`, or of its refit with `refit = TRUE`.
- `"variance"` one row per variable: the model-implied between-person variance, the stationary within-person variance, and the between-person share of the total (an intraclass correlation).
- `"covariance"` one row per entry of the lower triangle, diagonal included, of the innovation (contemporaneous) and between-person covariance matrices, on the data scale.
- `"path"` only for `select = "ebic"`: one row per penalty value on the EBIC path.

`include_zero` For `what = "edges"`: also return edges fixed to zero by pruning or the penalty. Default `FALSE`.

`edge_labels` For `plot()`: print each edge's weight on it. Default `TRUE`.

Details

`plot()` draws the three networks side by side on one circle layout, so each node has the same position in every panel, with the autoregressive effects as self-loops. Solid edges are significant at the fit's `alpha` and dashed edges are not (an edge without a p-value counts as not significant); for penalized estimates every selected edge is solid. Colour shows sign, blue for positive and vermillion for negative, and the signed weight is printed on each edge, so sign does not rest on colour alone. A network without edges is marked "no edges". Overriding `edge_style` removes the significance key from the legend. Plotting needs the `cograph` package.

Value

`as.data.frame()` returns a data frame:

edges columns `network`, `from`, `to`, `weight`, `se`, `p`, `ci_lower`, `ci_upper` and `significant` ($p <$ the fit's `alpha`; `NA` for penalized estimates). A temporal edge from `from` to `to` is the effect of `from` at one wave on `to` at the next.

parameters columns `matrix` (`mu`, `beta`, `omega_zeta_within`, `delta_zeta_within`, `omega_zeta_between`, `delta_zeta_between`), `row`, `col`, `estimate`, `se`, `p`, `ci_lower`, `ci_upper` and `fixed`.

fit columns `logl`, `npar`, `df`, `chisq`, `pvalue`, `baseline_chisq`, `baseline_df`, `cfi`, `tli`, `rmsea`, `aic` and `bic`.

diagnostics columns n, waves, score, boundary, boundary_part, se_available, starts_agree and starts.

variance columns variable, between, within and share_between.

covariance columns matrix ("innovation" or "between"), row, col and value.

path columns lambda (the penalty on the data scale), ebic, edges (nonzero penalized parameters), kkt (the lasso certificate), final (re-solved to full precision) and selected (the penalty returned).

summary() returns a data frame with one row per network and columns network, directed, nodes, edges, significant (NA for penalized estimates) and mean_abs_weight, autoregressive effects excluded. print() and plot() return x invisibly.

Conditions

psychnet_bad_input when what is not one of the listed tables, what = "path" is requested for a fit without select = "ebic", plot() is called without the cograph package, or a plotting argument is ambiguous or given twice.

See Also

panel_gvar() for fitting the model.

Examples

```
# srl_panel: 100 students x 4 weeks, simulated from a panel-GVAR in which
# effort in one week drives efficacy in the next.
fit <- panel_gvar(srl_panel,
                  vars = c("efficacy", "planning", "effort", "enjoyment",
                           "anxiety"),
                  id = "student", wave = "week")
summary(fit)
as.data.frame(fit, what = "parameters")
as.data.frame(fit, what = "diagnostics")
as.data.frame(fit, what = "covariance")
if (requireNamespace("cograph", quietly = TRUE)) plot(fit)
```

---

psychnet_result_group *Results of an analysis applied to networks per group*

Description

A psychnet_result_group holds the results of an analysis applied to networks estimated per group, such as their centralities or bootstrapped edges: one result per group, keyed by group level, so that they can be printed together and combined into one table.

It is returned when a psychnet_group is passed to net_centralities(), net_predict(), net_boot(), net_stability(), net_casedrop_reliability() or net_split_reliability(). print() shows each group's result under a heading, and as.data.frame() stacks the groups' tables with a leading group column.

Usage

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

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

Arguments

x	A psychnet_result_group object.
...	Ignored by print(); passed by as.data.frame() to the as.data.frame() method of each group's result (for example what).
row.names, optional	Ignored.

Value

print() returns x invisibly, after printing each group's result under a group heading.

as.data.frame() returns the groups' tables, each converted by its own as.data.frame() method, stacked with a leading group column. An error in converting any group's result is raised, not skipped.

See Also

psychnet_group for the networks; the analysis functions listed above for the columns of each table.

Examples

```
d <- rbind(data.frame(SRL_GPT, source = "GPT"),
           data.frame(SRL_Claude, source = "Claude"))
nets <- psychnet(d, method = "glasso", group = "source")
cent <- net_centralities(nets)
cent
as.data.frame(cent)
```

---

redundancy *Detect redundant pairs of items*

Description

redundancy() detects pairs of items that measure nearly the same thing, which appear in a network as two nodes with nearly the same connections. Such topological overlap complicates the interpretation of the network, and of centrality in particular (Hallquist, Wright & Molenaar, 2021), so redundant pairs are commonly identified before a network is estimated, and then combined or reduced to one item.

The function compares, for every pair of items, their correlations with each of the other items, using a significance test for two dependent correlations that share a variable. A pair is flagged as

redundant when the proportion of significantly different correlations is below `threshold` and the two items correlate above `cor_min` with each other. This is the procedure of `goldbricker()` in the `networktools` package.

It returns the flagged pairs, most redundant first.

To combine flagged items into one variable before estimating the network, `net_aggregate()` forms a composite score per group of items.

Usage

```
redundancy(
  data,
  p = 0.05,
  threshold = 0.25,
  cor_min = 0.5,
  cor_method = c("auto", "pearson", "spearman", "kendall")
)

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

Arguments

`data`	A numeric data frame or matrix with one row per observation and one column per item.
`p`	Significance level of each test of two correlations. Default `0.05`.
`threshold`	A pair is flagged when its proportion of significantly different correlations is below this value. Default `0.25`.
`cor_min`	Minimum correlation between the two items: a pair is flagged only when its correlation exceeds this value. Default `0.5`.
`cor_method`	Correlation type: `"auto"` (default; polychoric or polyserial correlations where items are ordinal, see `cor_auto()`), `"pearson"`, `"spearman"` or `"kendall"`.
`x`	A `psychnet_redundancy` object.
`...`	Unused.

Details

Non-numeric columns are dropped, and so is any column that is constant once incomplete rows are removed. Rows with a missing value in any numeric column are dropped (listwise deletion, as in `goldbricker()`), and the number of remaining rows is the sample size of every test.

Value

A data frame of class `psychnet_redundancy`, one row per flagged pair, most redundant first, with columns `item1`, `item2`, `proportion` (share of significantly different correlations) and `correlation` (the correlation of the two items); it has no rows when no pair is flagged. The proportions for all pairs are kept with the result as the attribute `proportion_matrix`. `print()` shows the flagged pairs, or a note that none were flagged, and returns `x` invisibly.

Functions

- `print(psychnet_redundancy)`: The thresholds used and the flagged pairs.

Conditions

`psychnet_bad_input` for an invalid argument, for data that are not a data frame or matrix, or for fewer than three usable items; `psychnet_degenerate_sample` when fewer than four complete rows remain; `psychnet_rows_dropped` (warning) when incomplete rows are dropped; `psychnet_constant_variable` (warning) when a constant column is dropped; `psychnet_dropped_columns` (a message) when non-numeric columns are dropped.

References

Hallquist, M. N., Wright, A. G. C., & Molenaar, P. C. M. (2021). Problems with centrality measures in psychopathology networks. *Multivariate Behavioral Research*, 56(2), 199-223.

See Also

`net_aggregate()` to combine items; `cor_auto()` for the correlations.

Examples

```
redundancy(SRL_Claude)
```

`relimp_network` *Estimate a directed relative-importance network*

Description

`relimp_network()` estimates a directed relative-importance network: for each variable, how much of its variance the other variables explain together and how that explained variance is shared among them. Each variable is regressed on all the others, and its R^2 is divided among the predictors by the LMG method (Lindeman et al., 1980; Groemping, 2006): a predictor's share is its increase in R^2 when it enters the regression, averaged over all orders in which the predictors can enter, which is the Shapley value of the predictor.

The shares are non-negative and sum to the R^2. The network is directed: the edge from a predictor to an outcome carries the predictor's share of the outcome's explained variance, so the two edges between a pair of variables can differ. `relimp_network()` computes these shares for every variable from raw data or a correlation matrix. With `normalized = TRUE` each outcome's shares are rescaled to sum to 1, giving proportions of explained variance rather than proportions of total variance. The computation considers every subset of predictors, so its cost doubles with each added variable; `max_nodes` limits the network size.

It returns the network of shares; the identity that defines the decomposition is checked by `certificate()`.

For undirected conditional associations use `ebic_glasso()` or `pcor_network()`. The shares are those of `relaimpo::calc.relimp(type = "lmg")` applied to each variable in turn.

Usage

```
relimp_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  max_nodes = 21L,
  normalized = FALSE,
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

data	A numeric data frame or matrix with one row per observation. Not needed when cor_matrix is given.
cor_matrix	A correlation matrix to use instead of data; it must be symmetric and positive semi-definite, and a covariance matrix is rescaled to a correlation matrix.
n	Accepted and ignored: the decomposition needs only the correlation matrix. It exists so that a call written for the other estimators, (cor_matrix = , n =), also works here.
cor_method	Correlation computed from data: "pearson" (default), "spearman", "kendall", or "auto" (polychoric and polyserial correlations for ordinal items; see cor_auto()).
max_nodes	Largest number of variables accepted, a whole number >= 2. Default 21; larger networks are refused.
normalized	Whether to rescale each outcome's incoming shares to sum to 1, as bootnet and relaimpo report them. Default FALSE, which keeps shares that sum to the outcome's R^2.
na_method	Handling of missing values in data: "pairwise" (default) or "listwise". See ebic_glasso().
labels	Optional node labels, one unique label per node of the network.

Details

The R^2 of each subset regression is computed from the correlation matrix. The certificate, computed by lmg_certificate(), checks that each outcome's shares sum to its full-model R^2 and that this R^2 agrees with its closed form from the inverse correlation matrix.

Value

A fitted network of class psychnet (psychnet), directed, whose edge weights are importance shares. as.data.frame() returns one row per edge with columns from (the predictor), to (the outcome) and weight; certificate() reports the structural check. The object also stores each node's full-model R^2 (r2), the correlation matrix (cor_matrix), whether the shares were normalized (normalized) and, with normalized = TRUE, the unscaled shares (raw_importance), from which the certificate is computed. The sample size is recorded when the network is built from data and is NA with cor_matrix.

Conditions

`psychnet_bad_input` when there are more variables than `max_nodes`, `max_nodes` or `normalized` is invalid, `cor_matrix` is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, or `labels` do not match the nodes; `psychnet_constant_variable` (warning) when a constant or all-missing column is dropped.

References

Lindeman, R. H., Merenda, P. F., & Gold, R. Z. (1980). *Introduction to bivariate and multivariate analysis*. Scott, Foresman.

Groemping, U. (2006). Relative importance for linear regression in R: The package relaimpo. *Journal of Statistical Software*, 17(1), 1-27. doi:10.18637/jss.v017.i01

See Also

`lmg_certificate()` and `certificate()` for the check; `ebic_glasso()` and `pcor_network()` for undirected networks; `psychnet()`, which routes `method = "relimp"` here; psychnet for the result.

Examples

```
S <- 0.4^abs(outer(1:5, 1:5, "-"))
relimp_network(cor_matrix = S)
```

SRL *Self-regulated learning scores generated by large language models*

Description

`SRL` is a set of five datasets of Motivated Strategies for Learning Questionnaire (MSLQ) construct scores, one per large language model: `SRL_GPT`, `SRL_Gemini`, `SRL_Claude`, `SRL_Mistral`, and `SRL_LLaMa`. Each is a random sample of 300 cases generated by that model in the study *"Delving into the psychology of Machines: Exploring the structure of self-regulated learning via LLM-generated survey responses"* (Computers in Human Behavior, 2025).

Only the five models whose responses carry estimable structure are included. Two other generators from the original study (ChatGPT and LeChat) produced near-independent items (mean absolute inter-item correlation ≈ 0.03), so their network is the empty graph; they are omitted. The five retained models span the spectrum the paper describes, from realistically structured (`SRL_GPT`) to strongly "over-coherent" (`SRL_Gemini`, `SRL_Claude`). Because the five data sets share their variables, they suit examples of group comparison (`net_compare()`) as well as single networks.

Usage

`SRL_GPT`

`SRL_Gemini`

`SRL_Claude`

`SRL_Mistral`

`SRL_LLaMa`

Format

Each is a data frame with 300 rows and 5 variables – the MSLQ construct scores (each the mean of that construct's Likert items, range 1–7):

`CSU` cognitive strategy use

`IV` intrinsic value

`SE` self-efficacy

`SR` self-regulation

`TA` test anxiety

Source

Saqr, M. (2025). Delving into the psychology of Machines: Exploring the structure of self-regulated learning via LLM-generated survey responses. *Computers in Human Behavior*, 173, 108769. doi:10.1016/j.chb.2025.108769

See Also

srl_panel for panel data; `ebic_glasso()` and `psychnet()`.

Examples

```
# partial-correlation network of the five constructs for one model
net <- ebic_glasso(SRL_GPT)
net
net_centralities(net)
```

`srl_panel` *Simulated weekly panel of self-regulated learning*

Description

`srl_panel` is a simulated panel of 100 students measured in four consecutive weeks on five self-regulated-learning indicators – a small, fully shareable dataset for the panel graphical VAR (`panel_gvar()`).

The data were simulated, once, from a stationary random-intercept VAR(1) (the panel graphical VAR model) whose structure was taken from the weekly means of real experience-sampling data (the anonymized `esm_srl` data of the idiographic package): the contemporaneous partial correlations, the temporal pattern (autoregressive effects and an effort -> efficacy lagged effect, scaled to 0.6 of the estimated values so trait and state differences are separable in four waves), and the correlations among students' average levels, with 60% of each variable's variance between students. Values are on a 0-100 scale (one of the 2,000 values was bounded to the scale) and rounded to one decimal. No real student is represented.

Usage

```
srl_panel
```

Format

A data frame with 400 rows (100 students x 4 weeks) and 7 columns:

`student` Student identifier, `"S001"` to `"S100"`.

`week` Week of measurement, 1 to 4.

`efficacy` Self-efficacy.

`planning` Planning.

`effort` Effort regulation.

`enjoyment` Enjoyment.

`anxiety` Anxiety.

Source

Simulated; structure derived from the weekly means of the `esm_srl` data in the idiographic package.

See Also

`panel_gvar()` for the model; SRL for cross-sectional data.

Examples

```
head(srl_panel)
fit <- panel_gvar(srl_panel,
                  vars = c("efficacy", "planning", "effort", "enjoyment",
                           "anxiety"),
                  id = "student", wave = "week")
summary(fit)
```

---

tmfg_certificate *Check that a network has the structure of a TMFG*

Description

`tmfg_certificate()` checks that a network from `tmfg_network()` has the structure of a triangulated maximally filtered graph (TMFG). A TMFG is defined by its structure rather than as the optimum of an objective: for p variables it is a connected planar graph with exactly $3(p-2)$ edges that is chordal (every cycle of four or more nodes has a chord) and is built from four-variable cliques (Massara et al., 2016).

These properties are what make the graph a TMFG and what the LoGo model (`logo_network()`) relies on, so checking them shows that a returned network is a valid TMFG, for instance that no edge was lost or added. `tmfg_certificate()` counts how many of the defining properties the network violates.

It returns that count: 0 confirms a valid TMFG.

`certificate()` reports the same value for a fitted TMFG network. The check does not assess which correlations the construction chose to retain.

Usage

```
tmfg_certificate(x)
```

Arguments

x	A TMFG network from `tmfg_network()` (psychnet).

Details

Four properties are checked on the stored graph: the number of edges is $3(p-2)$; the graph is chordal, tested by maximum-cardinality search; it is connected; and it is planar. For a connected chordal graph with $3(p-2)$ edges, planarity is checked as the structure of a stacked triangulation: no triangle has more than two common neighbours, and no triangle has two common neighbours that are adjacent to each other.

Value

A single integer from 0 to 4, the number of violated properties; 0 confirms a valid TMFG.

Conditions

`psychnet_bad_input` when `x` is not a network from `tmfg_network()`.

References

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, 5(2), 161-178. doi:10.1093/comnet/cnw015

See Also

`tmfg_network()`; `certificate()`; `lmg_certificate()`, the other structural check.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(200 * 6), 200, 6)
tmfg_certificate(tmfg_network(x))
```

`tmfg_network` *Filter a correlation network with the triangulated maximally filtered graph*

Description

`tmfg_network()` filters a correlation matrix to a sparse backbone with the triangulated maximally filtered graph (TMFG). The TMFG retains, for p variables, exactly $3(p-2)$ edges, chosen greedily to carry as much of the absolute correlation as possible while the graph stays planar and built from tetrahedra (four-variable cliques) (Massara et al., 2016).

Like other information-filtering networks, the TMFG keeps the strongest associations while imposing a fixed, interpretable structure. It shows which associations form the core of the correlation structure, without a sample-size-dependent selection rule. `tmfg_network()` builds the TMFG from raw data or a correlation matrix.

It returns the graph with the signed correlation of each retained pair as the edge weight; the edges are marginal correlations, not partial correlations. The structure that defines the graph is checked by `certificate()`.

For a Gaussian graphical model with partial correlations on the TMFG graph use `logo_network()`; for an unfiltered correlation network use `cor_network()`. The graph is the one built by `NetworkToolbox::TMFG()`.

Usage

```
tmfg_network(
  data = NULL,
  cor_matrix = NULL,
  n = NULL,
  cor_method = c("pearson", "spearman", "kendall", "auto"),
  na_method = c("pairwise", "listwise"),
  labels = NULL
)
```

Arguments

`data`	A numeric data frame or matrix with one row per observation. Not needed when `cor_matrix` is given.
`cor_matrix`	A correlation matrix to use instead of `data`; a covariance matrix is rescaled to a correlation matrix.
`n`	Accepted and ignored: the TMFG needs no sample size. It exists so that a call written for the other estimators, `(cor_matrix = , n = )`, also works here.
`cor_method`	Correlation computed from `data`: `"pearson"` (default), `"spearman"`, `"kendall"`, or `"auto"` (polychoric and polyserial correlations for ordinal items; see `cor_auto()`).
`na_method`	Handling of missing values in `data`: `"pairwise"` (default) or `"listwise"`. See `ebic_glasso()`.
`labels`	Optional node labels, one unique label per node of the network.

Details

The construction starts from the four variables with the largest sums of above-average absolute correlations, joined as a tetrahedron. It then adds the remaining variables one at a time, each time choosing the variable and the triangular face of the current graph for which the variable's summed absolute correlation with the face's three variables is largest, and joining the variable to those three. The result is a chordal planar graph with $3(p-2)$ edges, which `tmfg_certificate()` verifies.

The graph always has $3(p-2)$ edges, but a retained pair whose correlation is exactly zero has edge weight 0 and is therefore absent from `as.data.frame()` and from the printed edge count. With continuous data this happens only for a correlation matrix with exact zeros, such as a constructed or block-diagonal `cor_matrix`.

Value

A fitted network of class `psychnet` (psychnet) whose edge weights are the signed correlations on the retained edges. `as.data.frame()` returns one row per edge with columns `from`, `to` and `weight`; `certificate()` reports the structural check. The object also stores the TMFG graph as a logical matrix, including any retained edge of weight 0 (`adjacency`), its four-variable cliques and three-variable separators (`cliques`, `separators`; used by `logo_network()`) and the correlation matrix (`cor_matrix`). The sample size is recorded when the network is built from `data` and is `NA` with `cor_matrix`.

Conditions

`psychnet_bad_input` when there are fewer than four variables, `cor_matrix` is not a finite, symmetric, positive semi-definite matrix with a positive diagonal, an argument is invalid, or `labels` do not match the nodes; `psychnet_constant_variable` (warning) when a constant or all-missing column is dropped.

References

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, 5(2), 161-178. doi:10.1093/comnet/cnw015

See Also

`tmfg_certificate()` and `certificate()` for the check; `logo_network()` for the Gaussian graphical model on this graph; `psychnet()`, which routes `method = "tmfg"` here; psychnet for the result.

Examples

```
set.seed(1)
x <- matrix(stats::rnorm(200 * 6), 200, 6)
tmfg_network(x)
```

Index

∗ **datasets**
SRL, 91
srl_panel, 92
$.psychnet *(*plot.psychnet*)*, 72

as.data.frame.psychnet *(*plot.psychnet*)*, 72
as.data.frame.psychnet_bootstrap *(*net_boot*)*, 41
as.data.frame.psychnet_group *(*psychnet_group*)*, 80
as.data.frame.psychnet_moderated *(*condition*)*, 4
as.data.frame.psychnet_multilevel *(*psychnet_multilevel*)*, 82
as.data.frame.psychnet_nct *(*net_compare*)*, 54
as.data.frame.psychnet_panelgvar *(*psychnet_panelgvar*)*, 84
as.data.frame.psychnet_result_group *(*psychnet_result_group*)*, 86
as.data.frame.psychnet_stability *(*plot.psychnet_stability*)*, 74

certificate, 3
certificate(), *5*, *9*, *14–16*, *18*, *20–22*, *24–30*, *32–36*, *38, 39*, *69*, *71–73*, *80*, *89–91*, *94–96*
condition, 4
condition(), *36*, *38, 39*
cor_auto, 6
cor_auto(), *9*, *15*, *19*, *35*, *58*, *70*, *88–90*, *95*
cor_network, 8
cor_network(), *7, 8*, *61*, *70–72*, *77*, *95*

dichotomize, 10
dichotomize(), *29, 30*, *33*
difference_test, 11
difference_test(), *42–45*, *50*, *54*, *57*

ebic_glasso, 13
ebic_glasso(), *7, 8*, *10*, *18–20*, *22–24*, *26–28*, *34, 35*, *37*, *39*, *54*, *56–58*, *66*, *69–71*, *77*, *89–92*, *95*
event_frequencies, 16
event_frequencies(), *80*, *83*

ggm_modselect, 18
ggm_modselect(), *7*, *14*, *16*, *21, 22*, *34*, *58*, *70, 71*, *77*
ggm_support_kkt, 20
ggm_support_kkt(), *3, 4*, *15*, *20*, *22*, *24, 25*, *35, 36*
glasso_kkt, 22
glasso_kkt(), *3, 4*, *15, 16*, *21, 22*, *24, 25*, *27, 28*
glm_lasso_kkt, 24
glm_lasso_kkt(), *3, 4*, *22*, *24*, *30*, *33*, *39*

huge_network, 26
huge_network(), *14*, *16*, *19*, *22*, *24*, *77*

ising_fit, 28
ising_fit(), *10, 11*, *14*, *24, 25*, *31–33*, *37*, *39*, *58*, *60*, *77*
ising_sampler, 31
ising_sampler(), *10, 11*, *24, 25*, *29, 30*, *60*, *77*

lmg_certificate, 33
lmg_certificate(), *4*, *90, 91*, *94*
logo_network, 34
logo_network(), *21, 22*, *77*, *93*, *95, 96*

mgm_fit, 36
mgm_fit(), *4–6*, *10*, *14*, *24, 25*, *29, 30*, *58*, *60*, *77, 78*

net_aggregate, 39
net_aggregate(), *46, 47*, *88, 89*
net_boot, 41

net_boot(), *11–13*, *49, 50*, *52*, *54*, *57*, *60, 61*, *63, 64*, *72*, *74*, *76, 77*, *80*, *86*
net_bridge, 45
net_bridge(), *40, 41*, *50*, *52*, *59*
net_casedrop_reliability, 47
net_casedrop_reliability(), *42*, *45*, *63, 64*, *74*, *77*, *80*, *86*
net_centralities, 50
net_centralities(), *43*, *45–47*, *52, 53*, *59*, *61*, *72*, *75*, *77*, *80*, *86*
net_clustering, 52
net_clustering(), *61, 62*
net_compare, 54
net_compare(), *12, 13*, *42*, *79–81*, *91*
net_crosswalk, 57
net_crosswalk(), *79*
net_edge_betweenness, 59
net_edge_betweenness(), *50*, *52*
net_predict, 60
net_predict(), *30*, *32, 33*, *38, 39*, *52*, *65*, *80*, *86*
net_smallworld, 61
net_smallworld(), *52, 53*
net_split_reliability, 63
net_split_reliability(), *42*, *45*, *48, 49*, *80*, *86*
net_stability (plot.psychnet_stability), 74
net_stability(), *42*, *45*, *48–50*, *52*, *63, 64*, *72*, *80*, *86*
node_predictability, 65
node_predictability(), *60, 61*

panel_gvar, 66
panel_gvar(), *58*, *77*, *82*, *84*, *86*, *92, 93*
pcor_network, 69
pcor_network(), *8*, *10*, *14*, *16*, *18*, *77*, *89*, *91*
plot.psychnet, 72
plot.psychnet_bootstrap (net_boot), 41
plot.psychnet_bridge (net_bridge), 45
plot.psychnet_casedrop (net_casedrop_reliability), 47
plot.psychnet_centrality (net_centralities), 50
plot.psychnet_difference (difference_test), 11
plot.psychnet_multilevel (psychnet_multilevel), 82
plot.psychnet_nct (net_compare), 54
plot.psychnet_panelgvar (psychnet_panelgvar), 84
plot.psychnet_reliability (net_split_reliability), 63
plot.psychnet_stability, 74
print.psychnet (plot.psychnet), 72
print.psychnet_bootstrap (net_boot), 41
print.psychnet_casedrop (net_casedrop_reliability), 47
print.psychnet_group (psychnet_group), 80
print.psychnet_moderated (condition), 4
print.psychnet_multilevel (psychnet_multilevel), 82
print.psychnet_nct (net_compare), 54
print.psychnet_panelgvar (psychnet_panelgvar), 84
print.psychnet_redundancy (redundancy), 87
print.psychnet_reliability (net_split_reliability), 63
print.psychnet_result_group (psychnet_result_group), 86
print.psychnet_stability (plot.psychnet_stability), 74
psychnet, *3*, *5, 6*, *9, 10*, *15, 16*, *20*, *27–33*, *35–38*, *41*, *46*, *51*, *53*, *59, 60*, *62*, *65*, *71*, *77*, *79–81*, *83*, *90, 91*, *94*, *96*
psychnet(), *4*, *10*, *16–18*, *20*, *28*, *30*, *33*, *36*, *39–41*, *43*, *45*, *48*, *58*, *63*, *66*, *69*, *71*, *73*, *75*, *80–83*, *91, 92*, *96*
psychnet-class (plot.psychnet), 72
psychnet_group, *43, 44*, *48, 49*, *51*, *55*, *57*, *60, 61*, *63, 64*, *72, 73*, *75, 76*, *78, 79*, 80, *86, 87*
psychnet_multilevel, *18*, *79*, 82
psychnet_panelgvar, *66*, *68, 69*, 84
psychnet_result_group, *44*, *51*, *61*, *76*, *80*, 86

redundancy, 87
redundancy(), *40, 41*
relimp_network, 89
relimp_network(), *33, 34*, *50*, *60*, *72*, *77*

set.seed(), *44*, *64*, *76*
SRL, 91, *93*
SRL_Claude (SRL), 91
SRL_Gemini (SRL), 91

SRL_GPT *(SRL)*, 91
SRL_LLaMa *(SRL)*, 91
SRL_Mistral *(SRL)*, 91
srl_panel, *92*, 92
stats::p.adjust(), *9*, *12*, *43*, *55*, *71*
stats::p.adjust.methods, *32*, *67*
subset(), *78*
summary.psychnet *(plot.psychnet)*, 72
summary.psychnet_group
(psychnet_group), 80
summary.psychnet_multilevel
(psychnet_multilevel), 82
summary.psychnet_nct *(net_compare)*, 54
summary.psychnet_panelgvar
(psychnet_panelgvar), 84

tmfg_certificate, 93
tmfg_certificate(), *4*, *34*, *96*
tmfg_network, 95
tmfg_network(), *8*, *10*, *34*, *36*, *61*, *77*, *93*, *94*