Package 'IVPP'

May 7, 2026

Type Package

Title Invariance Partial Pruning Test

Version 1.1.2

Description An implementation of the Invariance Partial Pruning (IVPP) approach described in Du, X., Johnson, S. U., Epskamp, S. (2025) The Invariance Partial Pruning Approach to The Network Comparison in Longitudinal Data. IVPP is a two-step method that first test for global network structural difference with invariance test and then inspect specific edge difference with partial pruning. The package also allows you to compute centrality measures and use radar chart to plot. Analysis of bridge centralities by community pairs is also possible (e.g., the bridge strength from depression to anxiety, and from depression to panic disorder).

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.3.0)

Imports bootnet, clusterGeneration, dplyr, mvtnorm, psychonetrics, graphicalVAR, lifecycle, future.apply, future, networktools, qgraph, fmsb

BugReports https://github.com/xinkaidupsy/IVPP/issues

URL https://github.com/xinkaidupsy/IVPP

RoxygenNote 7.3.2

NeedsCompilation no

Author Xinkai Du [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4158-7878>)

Maintainer Xinkai Du <xinkai.du.xd@gmail.com>

Repository CRAN

Date/Publication 2026-02-28 23:40:09 UTC

Contents

bridge_by_community . . . 2
centrality . . . 2
centrality_radar . . . 4
gen_panelGVAR . . . 5
gen_tsGVAR . . . 7
IVPP_panelgvar . . . 8
IVPP_tsgvar . . . 11
sim_panelGVAR . . . 13
sim_tsGVAR . . . 15

Index **17**

`bridge_by_community` *Bridge centrality by target community*

Description

Computes bridge strength by community pairs

Usage

```
bridge_by_community(graph, community)
```

Arguments

`graph`	p x p adjacency matrix (can be directed or undirected)
`community`	character vector of length p in the SAME order as rows/cols of graph e.g., rep(c("dep","anx","mech"), each = 3)

`centrality` *Centrality (degree/EI) and bridge centrality for an network matrix*

Description

Computes node centrality measures from a weighted network matrix. Degree and expected-influence measures are obtained via `centrality` and bridge centrality is obtained via `bridge` when communities are supplied.

Usage

```
centrality(
  graph,
  communities = NULL,
  alpha = 1,
  posfun = abs,
  weighted = TRUE,
  signed = TRUE,
  directed = NULL,
  bridge_normalize = FALSE,
  useCommunities = "all"
)
```

Arguments

graph	A square numeric network matrix (p x p). Can be weighted and/or signed. If directed, provide a non-symmetric matrix. Row and column names should be identical and in the same order (node names).
communities	A character vector indicating community membership of the nodes (e.g., c("Comm1", "Comm1", "Comm2", "Comm2)).
alpha	The tuning parameter. Defaults to 1.
posfun	A function that converts positive and negative values to only positive. Defaults to the absolute value.
weighted	Logical, set to FALSE to set all edge weights to 1 or -1
signed	Logical, set to FALSE to make all edge weights absolute.
directed	Logical. Whether the input network is directed. Automatically detected if set to "NULL" (the default). Symmetric network matrices will be undirected, asymmetric matrices will be directed
bridge_normalize	logical. Bridge centralities are divided by their highest possible value (assuming max edge strength=1) in order to normalize by different community sizes
useCommunities	Character string passed to networktools::bridge(useCommunities = ...).

Value

A list with components:

degree Named list of numeric vectors (length p): OutDegree, InDegree, OutExpectedInfluence, InExpectedInfluence.

bridge If communities is provided, the object returned by networktools::bridge(); otherwise NULL.

meta List with metadata: directed, alpha, weighted, signed, and the expression name of posfun.

See Also

centrality, bridge

Examples

```
# Undirected signed weighted example:
W <- matrix(c(
  0,  0.4, -0.2,
  0.4, 0,   0.1,
 -0.2, 0.1, 0
), nrow = 3, byrow = TRUE)
colnames(W) <- rownames(W) <- c("A","B","C")

cent <- centrality(W)
cent$degree$OutDegree
cent$degree$OutExpectedInfluence

# Bridge centrality:
comm <- c(A = "X", B = "X", C = "Y")
cent_b <- centrality(W, communities = comm)
```

`centrality_radar` *Radar plot for selected centrality measures*

Description

Creates a radar chart (via `radarchart`) from the output of `centrality`.

Usage

```
centrality_radar(
  cent,
  measures = c("outdegree", "indegree", "outEI", "inEI", "bridgestrength",
    "bridgeoutdegree", "bridgeindegree", "bridgebetweenness", "bridgecloseness",
    "bridgeEI_1", "bridgeEI_2"),
  title = NULL,
  axis_range = NULL,
  add_legend = TRUE,
  legend_pos = "bottomright",
  ...
)
```

Arguments

`cent`	A list returned by `centrality`.
`measures`	Character vector of measures to plot. Allowed values: • `outdegree`, `indegree`, `outEI`, `inEI` • `bridgestrength`, `bridgeoutdegree`, `bridgeindegree`, `bridgebetweenness`, `bridgecloseness`, `bridgeEI_1`, `bridgeEI_2`
`title`	Optional character title. If provided and `...` does not already set `title`, it is passed to `fmsb::radarchart(title = ...)`.

axis_range	Numeric length-2 vector c(min, max) for the radial axis. If NULL, computed from the selected measures.
add_legend	Logical. If TRUE, adds a base-R legend with one entry per selected measure.
legend_pos	Character position, e.g. "bottomright", "topright", etc.
...	Additional arguments forwarded to radarchart.

Value

Invisibly returns the data frame used for plotting (in **fmsb** format): the first row is max, the second row is min, and subsequent rows correspond to the requested centrality measures.

See Also

centrality, radarchart

Examples

```
W <- matrix(c(
  0,  0.4, -0.2, 0,
  0.4, 0,   0.1, 0.3,
  -0.2, 0.1, 0,   0.2,
  0,   0.3, 0.2, 0
), nrow = 4, byrow = TRUE)
colnames(W) <- rownames(W) <- c("A", "B", "C", "D")

cent <- centrality(W)

# Plot degree + EI
# pdf('radar_plot.pdf')
centrality_radar(cent, measures = c("outdegree", "outEI"),
                 axis_range = c(0, 1), plwd = 2,
                 seg = 5, caxislabel = seq(0,1,0.2))
# dev.off()

# If bridge metrics were computed:

comm <- c(A="X", B="X", C="Y", D="Y")
cent_b <- centrality(W, communities = comm)
centrality_radar(cent_b, measures = c("bridgestrength", "bridgebetweenness"))
```

---

gen_panelGVAR *Generate a (multi-group) panelGVAR model*

Description

This function generates a (multi-group) panel GVAR model. Currently generating temporal and contemporaneous networks

Usage

```
gen_panelGVAR(
  n_node = 6,
  p_rewire_temp = 0.5,
  p_rewire_cont = 0.5,
  n_group = 1,
  propPos = 0.8
)
```

Arguments

n_node	an integer denoting the number of nodes
p_rewire_temp	a numeric value between 0-1 denoting the extent of group difference in the temporal network
p_rewire_cont	a numeric value between 0-1 denoting the extent of group difference in the contemporaneous network
n_group	an integer denoting the number of groups
propPos	specify the proportion of positive edges in the networks generated

Details

beta can be transposed to obtain the temporal network; PDC is the partial directed correlation matrix, which is a standardized version of temporal network; kappa is the precision matrix denoting conditional (in)dependence, which is a inverse of covariance matrix denoting the (dependence) among variables; kappa can be further standardized to the contemporaneous networks (omega_zeta_within)

Value

A list of beta, PDC, kappa and contemporaneous networks

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)
# Generate the network
net_ls <- gen_panelGVAR(n_node = 6,
                        p_rewire_temp = 0.5,
                        p_rewire_cont = 0,
                        n_group = 2)
```

`gen_tsGVAR` *Generate time-series GVAR model for multiple (heterogeneous) individuals*

Description

This function generates time-series GVAR model for multiple individuals that demonstrates difference or simularity. Currently generating temporal and contemporaneous networks

Usage

```
gen_tsGVAR(
  n_node = 6,
  p_rewire_temp = 0.5,
  p_rewire_cont = 0.5,
  n_persons = 1,
  propPos = 0.8
)
```

Arguments

`n_node`	an integer denoting the number of nodes
`p_rewire_temp`	a numeric value between 0-1 denoting the extent of individual difference in the temporal network
`p_rewire_cont`	a numeric value between 0-1 denoting the extent of individual difference in the contemporaneous network
`n_persons`	an integer denoting the number of individuals to generate tsGVAR for
`propPos`	specify the proportion of positive edges in the networks generated

Details

`beta` can be transposed to obtain the temporal network; `PDC` is the partial directed correlation matrix, which is a standardized version of temporal network; `kappa` is the precision matrix denoting conditional (in)dependence, which is a inverse of covariance matrix denoting the (dependence) among variables; kappa can be further standardized to the contemporaneous networks (`omega_zeta_within`)

Value

A list of beta, PDC, kappa and contemporaneous networks

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)

# Generate the network
net_ls <- gen_tsGVAR(n_node = 6,
                     p_rewire_temp = 0.5,
                     p_rewire_cont = 0,
                     n_persons = 2)
```

`IVPP_panelgvar` *The invariance partial pruning (IVPP) algorithm for panel GVAR models*

Description

This function implements the IVPP algorithm to compare networks in the multi-group panelGVAR models. The IVPP algorithm is a two-step procedure that first conducts an global invariance test of network difference and then performs partial pruning for the specific edge-level differences. Currently supports the comparison of temporal and contemporaneous networks.

Usage

```
IVPP_panelgvar(
  data,
  vars,
  idvar,
  beepvar,
  groups,
  global = TRUE,
  g_test_net = c("both", "temporal", "contemporaneous"),
  net_type = c("sparse", "saturated"),
  output_equal = FALSE,
  partial_prune = FALSE,
  prune_net = c("both", "temporal", "contemporaneous"),
  prune_alpha = 0.01,
  p_prune_alpha = 0.01,
  estimator = "FIML",
  standardize = c("none", "z", "quantile"),
  ncores = 1,
  ...
)
```

Arguments

`data`	A data frame containing the long-formatted panel data
`vars`	A character vector of variable names
`idvar`	A character string specifying subject IDs

beepvar	A character string specifying the name of wave (time) variable
groups	A character string specifying the name of group variable
global	A logical value default to TRUE. If FALSE, the global invariance test is skipped.
g_test_net	A character vector specifying the network you want to test group-equality on in the global invariance test. Specify "both" if you want to test on both temporal or contemporaneous networks. Specify "temporal" if you want to test only on the temporal network. Specify "contemporaneous" if you want to test only on the contemporaneous network. See the Details section for more information.
net_type	A character vector specifying the type of networks to be compared in the global test. Specify "saturated" if you want to estimate and compare the saturated networks. Specify "sparse" if you want to estimate and compare the pruned networks.
output_equal	Whether to output the networks that are constrained equal across groups. Default to FALSE. Can set to TRUE if the global tests rejects heterogeneity.
partial_prune	A logical value specifying whether to conduct partial pruning test or not.
prune_net	A character vector specifying the network you want to partial prune on. Only works when partial_prune = TRUE.
prune_alpha	A numeric value specifying the alpha level for the pruning (if net_type = "sparse").
p_prune_alpha	A numeric value specifying the alpha level for the partial pruning (if partial_prune = TRUE).
estimator	A character string specifying the estimator to be used. Must be "FIML"
standardize	A character string specifying the type of standardization to be used. "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.
ncores	A numeric value specifying the number of cores you want to use to run the analysis. Default to 1.
...	Additional arguments to be passed to the dlvm1 function

Details

The comparison between the fully unconstrained (free) model and tempEq model is a test for group equality in temporal networks. The comparison between fully constrained model (bothEq) and tempEq is a test for group equality in contemporaneous networks. Similarly, the comparison between the free model and contEq model is a test for group equality in contemporaneous networks, and the comparison between bothEq and contEq is a test for group equality in temporal networks.

Value

A list containing the results of IVPP and networks of all groups.

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)

# Generate the network
net_ls <- gen_panelGVAR(n_node = 6,
                        p_rewire_temp = 0.5,
                        p_rewire_cont = 0.5,
                        n_group = 2)

# Generate the data
data <- sim_panelGVAR(temp_base_ls = net_ls$temporal,
                      cont_base_ls = net_ls$omega_zeta_within,
                      n_person = 200,
                      n_time = 3,
                      n_group = 2,
                      n_node = 6)

# global test on both nets
omnibus_both <- IVPP_panelgvar(data,
                               vars = paste0("V",1:6),
                               idvar = "subject",
                               beepvar = "time",
                               groups = "group",
                               g_test_net = "both",
                               net_type = "sparse",
                               partial_prune = FALSE,
                               ncores = 1)

# global test on temporal
omnibus_temp <- IVPP_panelgvar(data,
                               vars = paste0("V",1:6),
                               idvar = "subject",
                               beepvar = "time",
                               groups = "group",
                               g_test_net = "temporal",
                               net_type = "sparse",
                               partial_prune = FALSE,
                               ncores = 1)

# global test on cont
omnibus_cont <- IVPP_panelgvar(data,
                               vars = paste0("V",1:6),
                               idvar = "subject",
                               beepvar = "time",
                               groups = "group",
                               g_test_net = "contemporaneous",
                               net_type = "sparse",
                               partial_prune = FALSE,
                               ncores = 1)

# partial prune on both networks
pp_both <- IVPP_panelgvar(data,
```

```
  vars = paste0("V",1:6),
  idvar = "subject",
  beepvar = "time",
  groups = "group",
  global = FALSE,
  partial_prune = TRUE,
  prune_net = "both",
  ncores = 1)
```

IVPP_tsgvar *The invariance partial pruning (IVPP) algorithm for idiographic GVAR models*

Description

This function implements the IVPP algorithm to compare networks in the multi-group panelGVAR models. The IVPP algorithm is a two-step procedure that first conducts an global invariance test of network difference and then performs partial pruning for the specific edge-level differences. Currently supports the comparison of temporal and contemporaneous networks.

Usage

```
IVPP_tsgvar(
  data,
  vars,
  idvar,
  dayvar,
  beepvar,
  global = TRUE,
  g_test_net = c("both", "temporal", "contemporaneous"),
  net_type = c("sparse", "saturated"),
  output_equal = FALSE,
  partial_prune = FALSE,
  prune_net = c("both", "temporal", "contemporaneous"),
  prune_alpha = 0.01,
  p_prune_alpha = 0.01,
  estimator = "FIML",
  standardize = c("none", "z", "quantile"),
  ncores = 1,
  ...
)
```

Arguments

data	A data frame containing the long-formatted panel data
vars	A character vector of variable names

idvar	A character string specifying the IDs of subjects you want to compare
dayvar	A character string specifying the name of day variable
beepvar	A character string specifying the name of variable indicating the measurement number at each day
global	A logical value default to TRUE. If FALSE, the global invariance test is skipped.
g_test_net	A character vector specifying the network you want to test group-equality on in the global invariance test. Specify "both" if you want to test on both temporal or contemporaneous networks. Specify "temporal" if you want to test only on the temporal network. Specify "contemporaneous" if you want to test only on the contemporaneous network. See the Details section for more information.
net_type	A character vector specifying the type of networks to be compared in the global test. Specify "saturated" if you want to estimate and compare the saturated networks. Specify "sparse" if you want to estimate and compare the pruned networks.
output_equal	Whether to output the networks that are constrained equal across groups. Default to FALSE. Can set to TRUE if the global tests rejects heterogeneity.
partial_prune	A logical value specifying whether to conduct partial pruning test or not.
prune_net	A character vector specifying the network you want to partial prune on. Only works when partial_prune = TRUE.
prune_alpha	A numeric value specifying the alpha level for the pruning (if net_type = "sparse").
p_prune_alpha	A numeric value specifying the alpha level for the partial pruning (if partial_prune = TRUE).
estimator	A character string specifying the estimator to be used. Must be "FIML"
standardize	A character string specifying the type of standardization to be used. "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.
ncores	A numeric value specifying the number of cores you want to use to run the analysis. Default to 1.
...	Additional arguments to be passed to the dlvm1 function

Details

The comparison between the fully unconstrained (free) model and tempEq model is a test for group equality in temporal networks. The comparison between fully constrained model (bothEq) and tempEq is a test for group equality in contemporaneous networks. Similarly, the comparison between the free model and contEq model is a test for group equality in contemporaneous networks, and the comparison between bothEq and contEq is a test for group equality in temporal networks.

Value

A list containing the results of IVPP and networks of all groups.

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)

# Generate the network
net_ls <- gen_tsGVAR(n_node = 6,
                     p_rewire_temp = 0.5,
                     p_rewire_cont = 0.5,
                     n_persons = 2)

# Generate the data
data <- sim_tsGVAR(beta_base_ls = net_ls$beta,
                   kappa_base_ls = net_ls$kappa,
                   # n_person = 2,
                   n_time = 100)

# global test on temporal
omnibus_temp <- IVPP_tsgvar(data,
                            vars = paste0("V",1:6),
                            idvar = "id",
                            g_test_net = "temporal",
                            net_type = "sparse",
                            partial_prune = FALSE,
                            ncores = 1)

# global test on cont
omnibus_cont <- IVPP_tsgvar(data,
                            vars = paste0("V",1:6),
                            idvar = "id",
                            g_test_net = "contemporaneous",
                            net_type = "sparse",
                            partial_prune = FALSE,
                            ncores = 1)

# partial prune on both networks
pp_both <- IVPP_tsgvar(data,
                       vars = paste0("V",1:6),
                       idvar = "id",
                       global = FALSE,
                       partial_prune = TRUE,
                       prune_net = "both",
                       ncores = 1)
```

`sim_panelGVAR` *Simulate data for a (multi-group) panelGVAR model*

Description

This function generates data for the input (multi-group) panelGVAR model

Usage

```
sim_panelGVAR(
  temp_base_ls,
  beta_base_ls,
  cont_base_ls,
  n_node,
  n_person = 500,
  n_time = 3,
  n_group,
  mean_trend = 0,
  p_rewire_temp = 0,
  p_rewire_cont = 0,
  save_nets = FALSE
)
```

Arguments

temp_base_ls	a list of temporal networks of all groups
beta_base_ls	a list of beta matrices of all groups
cont_base_ls	a list of contemporaneous networks of all groups
n_node	number of nodes
n_person	an integer denoting the sample size of each group, default to 500
n_time	number of waves, default to 3
n_group	number of groups
mean_trend	a numeric value indicating the extent of mean trends in data, default to 0
p_rewire_temp	a numeric value between 0 and 1 indicating the extent of non-stationarity in temporal networks, default to 0
p_rewire_cont	a numeric value between 0 and 1 indicating the extent of non-stationarity in contemporaneous networks, default to 0
save_nets	a logical value indicating whether to save the data-generating networks, default to FALSE

Value

A list of temporal and contemporaneous networks

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)
# Generate the network
net_ls <- gen_panelGVAR(n_node = 6,
                        p_rewire_temp = 0.5,
```

```
                        p_rewire_cont = 0,
                        n_group = 3)

# Generate the data
data <- sim_panelGVAR(temp_base_ls = net_ls$temporal,
                      cont_base_ls = net_ls$omega_zeta_within,
                      n_person = 500,
                      n_time = 4,
                      n_group = 3,
                      n_node = 6)
```

`sim_tsGVAR` *Simulate data for a (multi-group) N = 1 GVAR model*

Description

This function generates data for the input (multi-group) N = 1 GVAR model

Usage

```
sim_tsGVAR(
  temp_base_ls,
  beta_base_ls,
  cont_base_ls,
  kappa_base_ls,
  n_node,
  n_time = 50,
  n_person,
  save_nets = FALSE
)
```

Arguments

`temp_base_ls`	a list of temporal networks of all individuals
`beta_base_ls`	a list of beta matrices of all individuals
`cont_base_ls`	a list of contemporaneous networks of all individuals
`kappa_base_ls`	a list of precision matricies of all individuals
`n_node`	number of nodes
`n_time`	number of measurements per person, default to 50
`n_person`	number of individuals to generate data for
`save_nets`	a logical value indicating whether to save the data-generating networks, default to FALSE

Value

A list of temporal and contemporaneous networks

Author(s)

Xinkai Du Maintainer: Xinkai Du xinkai.du.xd@gmail.com

Examples

```
library(IVPP)
# Generate the network
net_ls <- gen_tsGVAR(n_node = 6,
                     p_rewire_temp = 0.5,
                     p_rewire_cont = 0,
                     n_persons = 3)

# Generate the data
data <- sim_tsGVAR(beta_base_ls = net_ls$beta,
                   kappa_base_ls = net_ls$kappa,
                   # n_person = 3,
                   n_time = 50)
```

Index

bridge, *2, 3*
bridge_by_community, 2

centrality, *2*, 2, *3–5*
centrality_radar, 4

dlvm1, *9*, *12*

gen_panelGVAR, 5
gen_tsGVAR, 7

IVPP_panelgvar, 8
IVPP_tsgvar, 11

radarchart, *4, 5*

sim_panelGVAR, 13
sim_tsGVAR, 15