Package ‘ResIN’

March 13, 2026

Type Package

Title Conduct Response Item Network (ResIN) Analysis with Social Response Data

Version 2.3.1

Author Philip Warncke [cre, aut],
Dino Carpentras [aut],
Adrian Lüders [aut]

Maintainer Philip Warncke <philip.warncke@ul.ie>

Description Contains various tools to estimate, analyze, and visualize Response Item Networks. 'ResIN' dummy-codes ordered and qualitative response choices from (survey) data, calculates pairwise associations and maps the location of each item response as a node in a force-directed network. Please refer to <https://www.resinmethod.net/> for more details.

License GPL-3

URL https://pwarncke77.github.io/ResIN/

BugReports https://github.com/pwarncke77/ResIN/issues

Depends R (>= 4.2.0)

Imports utils, graphics, psych, ggplot2 (>= 3.4.4), dplyr (>= 1.1.3), tidyr (>= 1.3.0), fastDummies, qgraph, igraph, DirectedClustering, foreach, parallelly, parallel, doSNOW, readr, ggraph (>= 2.2.0), shadowtext (>= 0.1.4), tidygraph, network

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-03-13 09:20:02 UTC

Contents

as.gephi.ResIN . . . 2
as.graphsjl . . . 3
as.graphsjl.ResIN . . . 4
as.igraph.ResIN . . . 5
as.network.ResIN . . . 6
as.networkx . . . 7
as.networkx.ResIN . . . 7
as.qgraph.ResIN . . . 8
as.tidygraph.ResIN . . . 9
Bootstrap_example . . . 10
BrJSocPsychol_2024 . . . 11
lik_data . . . 11
plot.ResIN . . . 12
print.ResIN . . . 13
ResIN . . . 13
ResIN_boots_draws . . . 18
ResIN_boots_execute . . . 19
ResIN_boots_executed . . . 20
ResIN_boots_extract . . . 21
ResIN_boots_prepare . . . 22
ResIN_boots_prepped . . . 23
ResIN_to_gephi . . . 24
ResIN_to_igraph . . . 25
ResIN_to_qgraph . . . 26
summary.ResIN . . . 27

Index **29**

`as.gephi.ResIN` *Coerce a ResIN object to Gephi CSV table(s)*

Description

Produces Gephi-readable edge (and optionally node) tables from a `ResIN` object and (by default) writes them to CSV. Set `dont_save_csv = TRUE` to return tables without writing files.

Usage

```
## S3 method for class 'ResIN'
as.gephi(
  x,
  file = "ResIN_gephi.csv",
  edges_only = TRUE,
  dont_save_csv = FALSE,
  weight_col = "weight",
  ...
)
```

Arguments

`x`	A `ResIN` object.
`file`	Output file name (legacy style).
`edges_only`	Logical; if TRUE write/return only edges.
`dont_save_csv`	Logical; set TRUE to disable writing.
`weight_col`	Name of the edge-weight column in `x$ResIN_edgelist`.
`...`	Ignored.

Value

If `edges_only = TRUE`, an edge table data.frame. Otherwise a list with `edges` and `nodes`. “When `dont_save_csv = FALSE`, the return value is returned invisibly.”

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Estimate a ResIN network
res <- ResIN(lik_data, plot_ggplot = FALSE)

## Create Gephi edge table without writing files
edges <- as.gephi(res, dont_save_csv = TRUE)
head(edges)

## Not run:
## Write CSV file(s) for import to Gephi
## (writes "ResIN_gephi.csv" by default)
as.gephi(res, file = "ResIN_gephi.csv")

## Write both edges and nodes tables
## (writes "ResIN_gephi_edges.csv" and "ResIN_gephi_nodes.csv")
as.gephi(res, file = "ResIN_gephi.csv", edges_only = FALSE)

## End(Not run)
```

`as.graphsjl` *Julia Graphs.jl coercion helpers*

Description

Generic for exporting objects to a lightweight Graphs.jl-ready representation. For `ResIN` objects, this creates edge and node CSV tables suitable for loading with **CSV.jl/DataFrames.jl**, with integer vertex IDs for direct use in **Graphs.jl**.

Usage

```
as.graphsjl(x, ...)
```

Arguments

`x`	Object to coerce/export.
`...`	Passed to methods.

`as.graphsjl.ResIN` *Export a ResIN object to Graphs.jl (Julia) tables*

Description

Produces Graphs.jl-style edge (and optionally node) tables from a `ResIN` object. Edge endpoints are mapped to integer vertex IDs (`src`, `dst`) to align with **Graphs.jl** in Julia. Node and edge metadata are preserved as table columns. The node table stores the vertex mapping and additional ResIN metadata.

Usage

```
## S3 method for class 'ResIN'
as.graphsjl(
  x,
  file = "ResIN_graphsjl.csv",
  edges_only = TRUE,
  dont_save_csv = FALSE,
  weight_col = "weight",
  ...
)
```

Arguments

`x`	A `ResIN` object.
`file`	Output file name (legacy style). If `edges_only` = `TRUE`, the edge table is written to `file`. If `edges_only` = `FALSE`, `file` is treated as a prefix and `"_edges.csv"` / `"_nodes.csv"` are appended (with any trailing `.csv` removed).
`edges_only`	Logical; if TRUE (default), only write/return edge table.
`dont_save_csv`	Logical; if FALSE (default), write CSV output. If TRUE, no files are written and the resulting table(s) are returned visibly.
`weight_col`	Preferred edge-weight column name. Defaults to `"weight"`.
`...`	Ignored.

Value

If `edges_only` = `TRUE`, an edge table `data.frame`. Otherwise a list with elements `edges` and `nodes`. The node table includes integer `vertex_id` values and preserved node metadata.

Examples

```
## Not run:
data(lik_data)
res <- ResIN(lik_data, generate_ggplot = FALSE, plot_ggplot = FALSE)

# Return tables only (no files written)
jl_tbls <- as.graphsjl(res, dont_save_csv = TRUE, edges_only = FALSE)

# Default behavior writes CSV files
# as.graphsjl(res, file = "ResIN_graphsjl.csv", edges_only = FALSE)

## End(Not run)
```

`as.igraph.ResIN` *Coerce a ResIN object to an igraph graph*

Description

Converts a `ResIN` object to an `igraph` graph using the adjacency matrix stored in `x$aux_objects$adj_matrix`.

Usage

```
## S3 method for class 'ResIN'
as.igraph(x, mode = "undirected", weighted = TRUE, diag = FALSE, ...)
```

Arguments

`x`	A `ResIN` object.
`mode, weighted, diag`	Passed to `igraph::graph_from_adjacency_matrix()`.
`...`	Additional arguments passed to `igraph::graph_from_adjacency_matrix()`.

Value

An `igraph` object.

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Run the function:

igraph_output <-  as.igraph(ResIN(lik_data, plot_ggplot = FALSE))

class(igraph_output)
```

```
## Plot and/or investigate as you wish:

igraph::plot.igraph(igraph_output)
```

as.network.ResIN   *Convert a ResIN object to a statnet/network object*

Description

Coerces a `ResIN` object to a `network` object (from the **network** package used in the **statnet** ecosystem). The method preserves edge-level columns from `x$ResIN_edgelist` as edge attributes and node-level columns from `x$ResIN_nodeframe` as vertex attributes whenever available.

Usage

```
## S3 method for class 'ResIN'
as.network(x, directed = FALSE, loops = FALSE, multiple = FALSE, ...)
```

Arguments

`x`	A `ResIN` object.
`directed`	Logical; should the resulting network be directed? Defaults to `FALSE`.
`loops`	Logical; allow self-loops? Defaults to `FALSE`.
`multiple`	Logical; allow multiple edges? Defaults to `FALSE`.
`...`	Additional arguments passed to `network::as.network()`.

Value

An object of class `network`.

Examples

```
data(lik_data)
res <- ResIN(lik_data, generate_ggplot = FALSE, plot_ggplot = FALSE)

# ResIN re-exports network::as.network()
net <- as.network.ResIN(res) ## alternatively: as.network(res)
class(net)
```

`as.networkx` *Python NetworkX coercion helpers*

Description

Generic for exporting objects to a lightweight NetworkX-ready representation. For `ResIN` objects, this creates edge and node CSV tables that can be read via **pandas** and converted to a **networkx** graph.

Usage

```
as.networkx(x, ...)
```

Arguments

`x`	Object to coerce/export.
`...`	Passed to methods.

`as.networkx.ResIN` *Export a ResIN object to NetworkX (Python) tables*

Description

Produces NetworkX-ready edge (and optionally node) tables from a `ResIN` object for further manipulation in Python. By default, this method writes CSV files that can be imported into Python via **pandas** and **networkx**. Node and edge metadata are preserved as table columns.

Usage

```
## S3 method for class 'ResIN'
as.networkx(
  x,
  file = "ResIN_networkx.csv",
  edges_only = TRUE,
  dont_save_csv = FALSE,
  weight_col = "weight",
  ...
)
```

Arguments

`x`	A `ResIN` object.
`file`	Output file name (legacy style). If `edges_only = TRUE`, the edge table is written to `file`. If `edges_only = FALSE`, `file` is treated as a prefix and `"_edges.csv"` / `"_nodes.csv"` are appended (with any trailing `.csv` removed).
`edges_only`	Logical; if TRUE (default), only write/return edge table.
`dont_save_csv`	Logical; if FALSE (default), write CSV output. If TRUE, no files are written and the resulting table(s) are returned visibly.
`weight_col`	Preferred edge-weight column name. Defaults to `"weight"`.
`...`	Ignored.

Value

If `edges_only = TRUE`, an edge table `data.frame`. Otherwise a list with elements `edges` and `nodes`.

Examples

```
## Not run:
data(lik_data)
res <- ResIN(lik_data, generate_ggplot = FALSE, plot_ggplot = FALSE)

# Return tables only (no files written)
nx_tbls <- as.networkx(res, dont_save_csv = TRUE, edges_only = FALSE)

# Default behavior writes CSV files
# as.networkx(res, file = "ResIN_networkx.csv", edges_only = FALSE)

## End(Not run)
```

`as.qgraph.ResIN` *Coerce a ResIN object to a qgraph object*

Description

Converts a `ResIN` object to a `qgraph` object using the adjacency matrix stored in `x$aux_objects$adj_matrix`.

Usage

```
## S3 method for class 'ResIN'
as.qgraph(
  x,
  layout = "spring",
  maximum = 1,
  vsize = 6,
  DoNotPlot = TRUE,
  sampleSize = NULL,
  title = "ResIN graph in qgraph",
  mar = c(3, 8, 3, 8),
  normalize = FALSE,
  ...
)
```

Arguments

`x`	A `ResIN` object.
`layout`, `maximum`, `vsize`, `DoNotPlot`, `sampleSize`, `title`, `mar`, `normalize`	Passed to `qgraph::qgraph()`.
`...`	Additional arguments passed to `qgraph::qgraph()`.

Value

A `qgraph` object.

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Run the function:
ResIN_qgraph <-  as.qgraph(ResIN(lik_data, plot_ggplot = FALSE))

class(ResIN_qgraph)
```

`as.tidygraph.ResIN` *Coerce a ResIN object to a tidygraph graph*

Description

Converts a `ResIN` object to a `tidygraph::tbl_graph` object while preserving as much node- and edge-level information as possible from `x$ResIN_nodeframe` and `x$ResIN_edgelist`.

Because `tidygraph` stores edge endpoints as integer node indices, the original edge endpoint labels are preserved in additional edge columns `from_name` and `to_name`.

If `ResIN_nodeframe` or `ResIN_edgelist` are unavailable, the method falls back to a simpler conversion via `as.igraph()` followed by `tidygraph::as_tbl_graph()`, which may not preserve all metadata.

Usage

```
## S3 method for class 'ResIN'
as.tidygraph(x, directed = FALSE, ...)
```

Arguments

`x`	A `ResIN` object.
`directed`	Logical; should the resulting graph be treated as directed? Defaults to `FALSE`.
`...`	Ignored.

Value

A `tidygraph::tbl_graph` object. Node data include (when present) all columns from `x$ResIN_nodeframe`; edge data include (when present) all columns from `x$ResIN_edgelist`, plus `from_name`/`to_name` preserving original endpoint labels.

Examples

```
## Load toy data and estimate ResIN
data(lik_data)
res <- ResIN(lik_data, network_stats = TRUE, detect_clusters = TRUE,
             plot_ggplot = FALSE)

## Convert to tidygraph
tg <- as.tidygraph(res)

class(tg)
```

`Bootstrap_example` *Output example for a bootstrapping analysis conducted with the ResIN package*

Description

Output example for a bootstrapping analysis conducted with the ResIN package

Usage

```
data(Bootstrap_example)
```

Format

An object of class `"rda"`

References

This dataset was made available by Lüders et.al. 2024.

Examples

```
data(Bootstrap_example)
names(Bootstrap_example[[1]])
```

`BrJSocPsychol_2024` *Source data for Lüders, A., Carpentras, D. and Quayle, M., 2024. Attitude networks as intergroup realities: Using network-modelling to research attitude-identity relationships in polarized political contexts. British Journal of Social Psychology, 63(1), pp.37-51.*

Description

The sample of N = 402 paid participants through the crowd working platform Prolific Academic. The core of the survey consists of A set of eight political attitude items abortion, immigration, gun control, and gay marriage. Each item followed a 5-point scale ranging from strong disagreement to strong agreement. The survey also includes items on partisanship, affective polarization, and a short vignette experiment.

Usage

```
data(BrJSocPsychol_2024)
```

Format

An object of class `"data.frame"`

References

This dataset was made available by Lüders et.al. 2024.

Examples

```
data(BrJSocPsychol_2024)
head(BrJSocPsychol_2024)
```

`lik_data` *Likert-type, mock-response data for "ResIN" package examples*

Description

An artificially created data-set (n=1000) of 12, 5-point Likert data. Modeled on the basis of a standard normal data-generating process. Likert scales contain 20 percent uncorrelated, homoscedastic measurement error. This data-set is used for the examples in the "ResIN" package vignette.

Usage

```
data(lik_data)
```

Format

An object of class "data.frame"

References

This data set was artificially created for the ResIN package.

Examples

```
data(lik_data)
head(lik_data)
```

`plot.ResIN` *Plot a ResIN object*

Description

Plot a ResIN object

Usage

```
## S3 method for class 'ResIN'
plot(x, which = c("network", "multimodal"), print_plot = TRUE, ...)
```

Arguments

`x`	A ResIN object.
`which`	Which plot to show: "network" (default) or "multimodal".
`print_plot`	Logical; print the plot (TRUE) or just return it (FALSE).
`...`	Ignored.

Value

The plot object (invisibly if `print_plot = TRUE`).

`print.ResIN` *Print a ResIN object*

Description

Print a ResIN object

Usage

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

Arguments

`x`	A ResIN object.
`...`	Ignored.

`ResIN` *Flagship function that implements Response Item Network (ResIN) analysis*

Description

Performs Response Item-Network (ResIN) analysis in one go. Users minimally need to supply a dataframe or matrix of discrete response data. If needed for step-wise analysis, all intermediate outputs can still be accessed as part of the aux_objects output list.

Usage

```
ResIN(
  df,
  node_vars = NULL,
  left_anchor = NULL,
  cor_method = "pearson",
  weights = NULL,
  missing_cor = "pairwise",
  offset = 0,
  ResIN_scores = TRUE,
  remove_nonsignificant = FALSE,
  remove_nonsignificant_method = "default",
  sign_threshold = 0.05,
  node_covars = NULL,
  node_costats = NULL,
  network_stats = TRUE,
  detect_clusters = FALSE,
  cluster_method = NULL,
  cluster_arglist = NULL,
  cluster_assignment = TRUE,
  generate_ggplot = TRUE,
  plot_ggplot = TRUE,
  plot_whichstat = NULL,
  plot_edgestat = NULL,
  color_palette = "RdBu",
  direction = 1,
  plot_responselabels = TRUE,
  response_levels = NULL,
  plot_title = NULL,
  multimodal = FALSE,
  multimodal_edge_overlay = c("multimodal", "none", "ResIN", "both"),
  save_input = TRUE,
  remove_negative = TRUE,
  EBICglasso = FALSE,
  EBICglasso_arglist = NULL,
  bipartite = FALSE,
  seed = NULL
)
```

Arguments

`df`	A data-frame object containing the raw data.
`node_vars`	An optional character vector detailing the attitude item columns to be selected for ResIN analysis (i.e. the subset of attitude variables in df).
`left_anchor`	An optional character scalar indicating a particular response node which determines the spatial orientation of the ResIN latent space. If this response node does not appear on the left-hand side, the x-plane will be inverted. This ensures consistent interpretation of the latent space across multiple iterations (e.g. in bootstrapping analysis). Defaults to NULL (no adjustment to orientation is taken.)
`cor_method`	Which correlation method should be used? Current implementation supports "pearson" (default) and "polychoric". Please note that polychoric correlations are currently unsupported for weighted analysis.
`weights`	Optional survey weights. Can be either `NULL` (default), a numeric vector of length `nrow(df)`, or a character scalar naming a weights column in `df`. If a column name is supplied and `node_vars = NULL`, the weights column is automatically excluded from the response-node variables used for ResIN estimation.
`missing_cor`	Character scalar controlling missing-data handling for correlation estimation. Either `"pairwise"` (default) or `"listwise"`.
`offset`	Optional off-set to correlation edges to manually adjust for over- or under-fitting the network. Defaults to `0`. Supplying a value between -1 and 0 globally reduces edge values by that amount, leading to the elimination of all positive edges below that value, resulting in a more sparse network. (However, we strongly recommend setting `remove_nonsignificant=TRUE` instead for a more principled

approach to ensuring optimal network sparsity as global thresholds have heuristic value at best). Alternatively, a value between 0 and 1 enforces a positive offset, resulting in more dense (but potentially over-fitted) networks.

`ResIN_scores`
: Logical; should spatial scores be calculated for every individual. Defaults to TRUE. Function obtains the mean positional score on the major (x-axis) and minor (y-axis). Current package implementation also provides empirical Bayesian scores via James-Stein shrinkage (`eb_x`) and heuristic shrinkage (`heur_x`) scores. Please refer to the package [vignette]https://pwarncke77.github.io/ResIN/articles/ResIN-VIGNETTE.html#spatial-interpretation-and-individual-latent-space-scores for further details.

`remove_nonsignificant`
: Logical; should non-significant edges be removed from the ResIN network? Defaults to FALSE. For weighted Pearson correlations, p-values are approximated using a weighted effective sample size. For currently unsupported polychoric configurations, ResIN falls back to Pearson and issues a warning.

`remove_nonsignificant_method`
: Character scalar specifying how p-values are thresholded when `remove_nonsignificant = TRUE`. Defaults to `"default"`, which prunes edges with raw p-values greater than `sign_threshold` (i.e., retains edges with `p <= sign_threshold`). If set to `"bh"`, p-values are adjusted using the Benjamini–Hochberg procedure; edges are retained only if the adjusted p-value is less than or equal to `sign_threshold`, interpreted as the target false discovery rate q. This provides multiplicity control across all tested edges and is typically more principled than using unadjusted p-values, but may be slightly slower. See `p.adjust` for details.

`sign_threshold`
: Numeric scalar controlling the pruning threshold used when `remove_nonsignificant = TRUE`. For `remove_nonsignificant_method = "default"`, this is the raw p-value cutoff (e.g., `0.05`). For `remove_nonsignificant_method = "bh"`, this is the target false discovery rate level q (e.g., `0.05`), applied to Benjamini–Hochberg adjusted p-values.

`node_covars`
: An optional character string selecting quantitative co-variates that can be used to enhance ResIN analysis. Typically, these covariates provide grouped summary statistics for item response nodes. (E.g.: What is the average age or income level of respondents who selected a particular item response?) Variable names specified here should match existing columns in `df`.

`node_costats`
: If any `node_covars` are selected, what summary statistics should be estimated from them? Argument should be a character vector and call a base-R function. (E.g. `"mean"`, `"median"`, `"sd"`). Each element specified in `node_costats` is applied to each element in `node_covars` and the out-put is stored as a node-level summary statistic in the `ResIN_nodeframe`. The extra columns in `ResIN_nodeframe` are labeled according to the following template: "covariate name"_"statistic". So for the respondents mean age, the corresponding column in `ResIN_nodeframe` would be labeled as "age_mean".

`network_stats`
: Should common node- and graph level network statistics be extracted? Calls `qgraph::centrality_auto` and `DirectedClustering::ClustF` to the ResIN graph object to extract node-level betweenness, closeness, strength centrality, as well as the mean and standard deviation of these scores at the network level.

Also estimates network expected influence, average path length, and global clustering coefficients. Defaults to TRUE. Set to FALSE if estimation takes a long time.

`detect_clusters`
: Optional, should community detection be performed on item response network? Defaults to FALSE. If set to TRUE, performs a clustering method from the [igraph](https://igraph.org/r/doc/cluster_leading_eigen.html) library and stores the results in the `ResIN_nodeframe` output.

`cluster_method`
: A character scalar specifying the [igraph-based](https://igraph.org/r/doc/communities.html) community detection function. Current ResIN (v. 2.3.1) implementation `"cluster_leading_eigen"`, `"cluster_fast_greedy"`, `"cluster_spinglass"`, `"cluster_edge_betweenness"`, `"cluster_louvain"`, `"cluster_walktrap"`, `"cluster_infomap"`, `"cluster_optimal"` and `"cluster_fluid_communities"`. Note that `"cluster_fluid_communities"` additionally requires a pre-supplied `"no.of.communities"` argument which can be supplied via the `"cluster_arglist"` argument (defaults to 2). Please consult the [igraph](https://igraph.org/r/doc/cluster_leading_eigen.html) community detection algorithm library for more information on each algorithm and their requirements.

`cluster_arglist`
: An optional list specifying additional arguments to the selected [igraph](https://igraph.org/r/doc/communi clustering method.

`cluster_assignment`
: Should individual (survey) respondents be assigned to different clusters? If set to TRUE, function will generate an n*c matrix of probabilities for each respondent to be assigned to one of c clusters. Furthermore, a vector of length `n` is generated displaying the most likely cluster respondents belong to. In case of a tie between one or more clusters, a very small amount of random noise determines assignment. Both matrix and vectors are added to the `aux_objects` list. Defaults to FALSE and will be ignored if `detect_clusters` is set to FALSE.

`generate_ggplot`
: Logical; should a ggplot-based visualization of the ResIN network be generated? Defaults to TRUE.

`plot_ggplot`
: Logical; should the ggplot of the ResIN network be plotted? Defaults to TRUE. If set to FALSE, the ggplot object will not be directly returned to the console. (However, if generate_ggplot=TRUE, the plot will still be generated and stored alongside the other output objects.)

`plot_whichstat`
: Should a particular node-level metric be color-visualized in the ggplot output? For node cluster, specify "cluster". For the same Likert response choices or options, specify "choices". For a particular node-level co-variate please specify the name of the particular element in `node_covars` followed by a "_" and the specific `node_costats` you would like to visualize. For instance if you want the visualize average age at the node-level, you should specify "age_mean". To colorize by node centrality statistics, possible choices are "Strength", "Betweenness", "Closeness", and "ExpectedInfluence". Defaults to NULL. Make sure to supply appropriate choices to `node_covars`, `node_costats`, `detect_clusters`, and/or `network_stats` prior to setting this argument.

`plot_edgestat`	Should the thickness of the edges be adjusted according to a particular co-statistic? Defaults to NULL. Possible choices are "weight" for the bi-variate correlation strength, and "edgebetweenness"
`color_palette`	Optionally, you may specify the ggplot2 color palette to be applied to the plot. All options contained in [`RColorBrewer`](https://cran.r-project.org/web/packages/RColorBrewer/RColor (for discrete colors such as cluster assignments) and [`ggplot2::scale_colour_distiller`](https://ggpl are supported. Defaults to "RdBu".
`direction`	Which direction should the color palette be applied in? Defaults to 1. Set to -1 if the palette should appear in reverse order.
`plot_responselabels`	Should response labels be plotted via `geom_text`? Defaults to TRUE. It is recommended to set to FALSE if the network possesses a lot of nodes and/or long response choice names.
`response_levels`	An optional character vector specifying the correct order of global response levels. Only useful if all node-items follow the same convention (e.g. ranging from "strong disagreement" to "strong agreement"). The supplied vector should have the same length as the total number of response options and supply these (matching exactly) in the correct order. E.g. c("Strongly Agree", "Somewhat Agree", "Neutral", "Somewhat Disagree", "Strongly Disagree"). Defaults to NULL.
`plot_title`	Optionally, a character scalar specifying the title of the ggplot output. Defaults to "ResIN plot".
`multimodal`	Logical; should a multimodal graph which jointly incorporates respondents/ data rows and response choices be produced in addition to classic ResIN graph? Defaults to FALSE. If set to TRUE, an [igraph](https://igraph.org/r/doc/) multimodal graph with response options as node type 1 and participants as node type 2 will be generated and included in the output list. Further, an object called `coordinate_df` with spatial coordinates of respondents and a plot-able `ggraph`-object called `multimodal_ggraph` are generated if set to TRUE.
`multimodal_edge_overlay`	Character scalar controlling which edges are drawn in the multimodal plot when `multimodal = TRUE`. One of `"none"` (no edges), `"ResIN"` (only the original ResIN edges among response nodes, styled via `plot_edgestat` when supplied), `"multimodal"` (only participant–response edges), or `"both"` (ResIN edges as a base layer plus multimodal edges on top).
`save_input`	Logical; should input data and function arguments be saved (this is necessary for running ResIN_boots_prepare function). Defaults to TRUE.
`remove_negative`	Logical; should all negative correlations be removed? Defaults to TRUE (highly recommended). Setting to FALSE makes it impossible to estimate a force-directed network layout. Function will use igraph::layout_nicely instead.
`EBICglasso`	Retired as of ResIN 2.3.0 and ignored.
`EBICglasso_arglist`	Retired as of ResIN 2.3.0 and ignored.
`bipartite`	Retired as of ResIN 2.3.1 and ignored. Please set `multimodal` argument instead.
`seed`	Random seed for force-directed algorithm. Defaults to NULL (no seed is set.) If scalar integer is supplied, that seed will be set prior to analysis.

Value

An edge-list type data-frame, `ResIN_edgelist`, a node-level data-frame, `ResIN_nodeframe`, an n*2 data-frame of individual-level spatial scores along the major (x) and minor(y) axis, `ResIN_scores` a list of graph-level statistics `graph_stats` including (`graph_structuration`), and centralization (`graph_centralization`). Further, a `multimodal_output` list which includes an `igraph` class multimodal graph (`multimodal_igraph`), a data frame, `coordinate_df`, with spatial coordinates of respondents, and a plot-able `ggraph`-object called `multimodal_ggraph` is optionally generated. Lastly, the output includes a list of auxiliary objects, `aux_objects`, including the ResIN adjacency matrix (`adj_matrix`), an alternate version of the adjacency matrix with all negative edges retained, a numeric vector detailing which item responses belong to which item (`same_items`), and the dummy-coded item-response data-frame (`df_dummies`). For reproducibility, (`aux_objects$meta` stores a numeric dataframe identifier (`df_id`, the random seed, call, and the (`ResIN` package version used to create the object."

Examples

```
## Load the 12-item simulated Likert-type toy dataset
data(lik_data)

# Apply the ResIN function to toy Likert data:
ResIN_obj <- ResIN(lik_data, network_stats = TRUE, remove_nonsignificant = TRUE)
```

`ResIN_boots_draws` *Extract bootstrap draws from ResIN objects*

Description

`"ResIN_boots_draws"` objects are typically returned by `ResIN_boots_extract` when the requested quantity is scalar per bootstrap iteration (e.g., `"global_clustering"`). The object is a numeric vector with attributes describing the bootstrap context.

Details

Common attributes include:

`what` Name of the extracted quantity.

`n_total`, `n_ok`, `n_failed` Counts of total, successful, and failed iterations.

`plan` The `"ResIN_boots_prepped"` plan used to generate the results (if attached).

`created` Timestamp of execution (if attached).

Methods

`print(x)` Compact display including median and 95% CI.

`summary(object)` Return descriptive statistics and quantiles.

`confint(object)` Quantile-based confidence intervals.

`plot(x)` Histogram with CI markers.

`ResIN_boots_execute` *Carry out prepared bootstrap analyses on ResIN networks*

Description

Executes a bootstrap plan created by `ResIN_boots_prepare` by repeatedly re-estimating ResIN on resampled or permuted versions of the original data. Can optionally leverage CPU parallelism.

Usage

```
ResIN_boots_execute(
  ResIN_boots_prepped,
  parallel = FALSE,
  detect_cores = TRUE,
  core_offset = 1L,
  n_cores = 2L,
  inorder = FALSE,
  verbose = TRUE
)
```

Arguments

`ResIN_boots_prepped`	A `"ResIN_boots_prepped"` bootstrap plan (output of `ResIN_boots_prepare`).
`parallel`	Logical; should execution use parallelism via `foreach` + a PSOCK cluster? Defaults to FALSE.
`detect_cores`	Logical; should available CPU cores be detected automatically? Defaults to TRUE (ignored when `parallel = FALSE`).
`core_offset`	Integer offset subtracted from the number of detected cores. Defaults to 1L. Change to 0L on low-overhead systems or if sure that system won't stall.
`n_cores`	Manually specify number of cores (ignored if `detect_cores = TRUE` or `parallel = FALSE`).
`inorder`	Logical; should parallel execution preserve sequential ordering? Defaults to FALSE.
`verbose`	Logical; should the type of computational execution (parallel or sequential), the parallel engine (if any) and the number of cores be returned to the dashboard while the function is running?

Value

An object of class `"ResIN_boots_executed"` containing `n` bootstrapped `ResIN` fits. Use `print()`, `summary()`, `length()`, and `[` to inspect or subset results. See `ResIN_boots_executed` for details.

Examples

```
## Load the 12-item simulated Likert-type toy dataset
data(lik_data)

# Apply the ResIN function to toy Likert data:
ResIN_obj <- ResIN(lik_data, network_stats = TRUE,
                      generate_ggplot = FALSE, plot_ggplot = FALSE)


# Prepare for bootstrapping
prepped_boots <- ResIN_boots_prepare(ResIN_obj, n=50, boots_type="resample")

# Execute the prepared bootstrap list
executed_boots <-  ResIN_boots_execute(prepped_boots, parallel = TRUE,
                      detect_cores = TRUE, verbose = FALSE)

# Extract results - here for example, the network (global)-clustering coefficient
ResIN_boots_extract(executed_boots, what = "global_clustering", summarize_results = TRUE)
```

`ResIN_boots_executed` *Bootstrapped results*

Description

The `"ResIN_boots_executed"` object is returned by `ResIN_boots_execute`. It contains a list of fitted `"ResIN"` objects, typically produced by refitting ResIN on resampled/permuted versions of the original data.

Details

The object is a list of length `n` where each element is (typically) a `ResIN` fit. For reproducibility, the following attributes may be present:

`plan` A `"ResIN_boots_prepped"` plan used to generate the results.

`boot_inputs` Optional list of bootstrap input data sets (only if `save_input` = `TRUE` in the plan).

`created` Time stamp when the executed object was created.

Methods

`print(x)` Print a compact overview of the bootstrap results.

`summary(object)` Summarize the results (iterations, successes/failures, plan details).

`length(x)` Return the number of iterations.

`x[i]` Subset the results while preserving attached attributes.

See Also

`ResIN_boots_prepare`, `ResIN_boots_execute`, `ResIN_boots_extract`, `ResIN_boots_prepped`

`ResIN_boots_extract` *Extract and summarize the results of a bootstrap simulation undertaken on a ResIN object*

Description

Extract bootstrap draws from `"ResIN_boots_executed"` results. Failed iterations (stored as `NULL`) are skipped automatically.

Usage

```
ResIN_boots_extract(
  ResIN_boots_executed,
  what,
  summarize_results = FALSE,
  allow_missing = FALSE
)
```

Arguments

`ResIN_boots_executed`	An object of class `"ResIN_boots_executed"` (output of `ResIN_boots_execute`).
`what`	Character scalar naming the quantity to extract (e.g., `"global_clustering"`).
`summarize_results`	Logical; if TRUE return a small summary table, otherwise return draws.
`allow_missing`	Logical; if FALSE (default) the function errors if `what` is missing in any successful fit. If TRUE, missing iterations are kept as `NULL` and summaries are computed from available values.

Value

If the extracted quantity is scalar per iteration, returns an object of class `"ResIN_boots_draws"` (a numeric vector with attributes). If `summarize_results = TRUE`, returns a one-row data.frame of summary statistics.

Examples

```
## Load the 12-item simulated Likert-type toy dataset
data(lik_data)

# Apply the ResIN function to toy Likert data:
ResIN_obj <- ResIN(lik_data, network_stats = TRUE,
                      generate_ggplot = FALSE, plot_ggplot = FALSE)


# Prepare for bootstrapping
prepped_boots <- ResIN_boots_prepare(ResIN_obj, n=50, boots_type="resample")
```

```
# Execute the prepared bootstrap list
executed_boots <-  ResIN_boots_execute(prepped_boots, parallel = TRUE,
                       detect_cores = TRUE, verbose = FALSE)

# Extract results - here for example, the network (global)-clustering coefficient
ResIN_boots_extract(executed_boots, what = "global_clustering", summarize_results = TRUE)
```

ResIN_boots_prepare *Create a bootstrap plan for re-estimating ResIN objects to derive statistical uncertainty estimates*

Description

Provides instructions for how to bootstrap a ResIN network to derive uncertainty estimates around core quantities of interest. Requires output of `ResIN` function.

Usage

```
ResIN_boots_prepare(
  ResIN_object,
  n = 1000,
  boots_type = "resample",
  resample_size = NULL,
  weights = NULL,
  save_input = FALSE,
  seed_boots = 42
)
```

Arguments

`ResIN_object`	A ResIN object to prepare bootstrapping workflow.
`n`	Bootstrapping sample size. Defaults to 10.000.
`boots_type`	What kind of bootstrapping should be performed? If set to "resample", function performs row-wise re-sampling of raw data (useful for e.g., sensitivity or power analysis). If set to "permute", function will randomly reshuffle raw item responses (useful e.g., for simulating null-hypothesis distributions). Defaults to "resample".
`resample_size`	Optional parameter determining sample size when `boots_type` is set to "resample". Defaults of to number of rows in raw data.
`weights`	An optional weights vector that can be used to adjust the re-sampling of observations. Should either be NULL (default) or a positive numeric vector of the same length as the original data.

`save_input`	Should all input information for each bootstrap iteration (including re-sampled/permuted data) be stored. Set to FALSE by default to save a lot of memory and disk storage.
`seed_boots`	Random seed for bootstrap samples

Value

An object of class `"ResIN_boots_prepped"` containing a bootstrap plan (specification) used by `ResIN_boots_execute`. Use `print()`, `summary()`, `length()`, and `[` to inspect or subset the plan. See `ResIN_boots_prepped` for details.

Examples

```
## Load the 12-item simulated Likert-type toy dataset
data(lik_data)

# Apply the ResIN function to toy Likert data:
ResIN_obj <- ResIN(lik_data, network_stats = TRUE,
                      generate_ggplot = FALSE, plot_ggplot = FALSE)


# Prepare for bootstrapping
prepped_boots <- ResIN_boots_prepare(ResIN_obj, n=50, boots_type="resample")

# Execute the prepared bootstrap list
executed_boots <-  ResIN_boots_execute(prepped_boots, parallel = TRUE,
                      detect_cores = TRUE, verbose = FALSE)

# Extract results - here for example, the network (global)-clustering coefficient
ResIN_boots_extract(executed_boots, what = "global_clustering", summarize_results = TRUE)
```

`ResIN_boots_prepped` *Prepared bootstrap plan*

Description

The `"ResIN_boots_prepped"` object is created by `ResIN_boots_prepare` and provides a reproducible specification for running a bootstrap with `ResIN_boots_execute`.

Details

A `"ResIN_boots_prepped"` object is a list with (at least) the following elements:

call The matched call used to create the plan.

boots_type Character; `"resample"` or `"permute"`.

n Integer; number of bootstrap iterations.

resample_size Integer; sample size used when resampling rows.

weights Optional numeric vector of sampling weights (or `NULL`).

save_input Logical; whether to store bootstrap inputs during execution.

seed_boots Integer; seed used to generate per-iteration seeds.

iter_seeds Integer vector; per-iteration RNG seeds (useful for parallel execution).

arglist A list of arguments passed to `ResIN` for each re-fit (typically based on the original ResIN call, with plotting disabled).

df_id Character; MD5 hash identifying the raw data used to create the plan.

ResIN_version Character; ResIN package version used to create the plan.

Methods

`print(x)` Print a compact summary of the bootstrap plan.

`summary(object)` Return a structured summary of the plan.

`length(x)` Return the number of iterations `n`.

`x[i]` Subset the plan to selected iteration indices (also updates `n`).

See Also

`ResIN_boots_prepare`, `ResIN_boots_execute`, `ResIN_boots_extract`

`ResIN_to_gephi` *(Deprecated.) Convert ResIN networks to Gephi-readable csv tables. Use* `as.gephi()` *method instead.*

Description

Deprecated/legacy function. Saves a ResIN graph as a series of csv files readable by Gephi. Now supplanted by `as.gephi()` method.

Usage

```
ResIN_to_gephi(
  ResIN_object,
  file = "ResIN_gephi.csv",
  edges_only = TRUE,
  dont_save_csv = FALSE
)
```

Arguments

`ResIN_object`	The output of the ResIN function (a list with class ResIN).
`file`	The name with .csv extension for the Gephi readable file to be output at. Defaults to "ResIN_gephi.csv".
`edges_only`	Logical; if TRUE write/return only edges.
`dont_save_csv`	Logical; set TRUE to disable writing.

Value

A series of csv files readable by Gephi

References

Source code of original function (< version 2.2.0) had been adapted from: https://github.com/RMHogervorst/gephi?tab=MIT-1-ov-file#readme

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Estimate a ResIN network
res <- ResIN(lik_data, generate_ggplot = FALSE)

## Create Gephi edge table without writing files
edges <- as.gephi(res, dont_save_csv = TRUE)
head(edges)

## Not run:
## Write CSV file(s) for import to Gephi
## (writes "ResIN_gephi.csv" by default)
as.gephi(res, file = "ResIN_gephi.csv")

## Write both edges and nodes tables
## (writes "ResIN_gephi_edges.csv" and "ResIN_gephi_nodes.csv")
as.gephi(res, file = "ResIN_gephi.csv", edges_only = FALSE)

## End(Not run)
```

`ResIN_to_igraph` *(Deprecated.) Convert a ResIN network into an igraph object. Use* `as.igraph()` *method instead.*

Description

Deprecated/legacy function. Transforms the output of the `ResIN` function into an [igraph](https://igraph.org/r/doc/cluster_lead
object. Now simply a wrapper for the `as.igraph()` method.

Usage

```
ResIN_to_igraph(ResIN_object, igraph_arglist = NULL)
```

Arguments

`ResIN_object`	the output of the ResIN function (a list with class ResIN).
`igraph_arglist`	an optional argument list to be supplied to the igraph::graph_from_adjacency_matrix function. If NULL, default is: list(mode = "undirected", weighted = TRUE, diag = FALSE).

Value

A class `igraph` object.

References

Csardi G, Nepusz T (2006). “The igraph software package for complex network research.” InterJournal, Complex Systems, 1695. https://igraph.org.

See Also

`as.igraph` as the recommended interface.

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Run the function:

igraph_output <-  as.igraph(ResIN(lik_data, plot_ggplot = FALSE))

class(igraph_output)

## Plot and/or investigate as you wish:

igraph::plot.igraph(igraph_output)
```

`ResIN_to_qgraph` *(Deprecated.) Convert a ResIN network into an qgraph object. Use* `as.qgraph()` *method instead.*

Description

Deprecated/legacy function. Transforms the output of the `ResIN` function into an `qgraph` object. Use `as.qgraph()` method instead.

Usage

```
ResIN_to_qgraph(ResIN_object, qgraph_arglist = NULL)
```

Arguments

`ResIN_object`	the output of the ResIN function (a list with class ResIN).
`qgraph_arglist`	an optional argument list to be supplied to the igraph::graph_from_adjacency_matrix function. If NULL, defaults are: `list(layout = "spring", maximum = 1, vsize = 6, DoNotPlot = TRUE, sampleSize = nrow(df_nodes), mar = c(3,3,3,3), normalize = FALSE)`

Value

A [qgraph]https://cran.r-project.org/web/packages/qgraph/index.html graph object.

References

Epskamp S, Cramer AOJ, Waldorp LJ, Schmittmann VD, Borsboom D (2012). "qgraph: Network Visualizations of Relationships in Psychometric Data." Journal of Statistical Software, 48(4), 1–18.

See Also

`as.qgraph` as the recommended interface.

Examples

```
## Load the 12-item simulated Likert-type ResIN toy dataset
data(lik_data)

## Run the function:
ResIN_qgraph <-  as.qgraph(ResIN(lik_data, plot_ggplot = FALSE))

class(ResIN_qgraph)
```

`summary.ResIN` *Summarize a ResIN object*

Description

Summarize a ResIN object

Usage

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

Arguments

`object`	A ResIN object.
`...`	Ignored.

Value

An object of class `summary.ResIN`.

Index

∗ **datasets**
    Bootstrap_example, 10
    BrJSocPsychol_2024, 11
    lik_data, 11

as.gephi.ResIN, 2
as.graphsjl, 3
as.graphsjl.ResIN, 4
as.igraph, *26*
as.igraph.ResIN, 5
as.network.ResIN, 6
as.networkx, 7
as.networkx.ResIN, 7
as.qgraph, *27*
as.qgraph.ResIN, 8
as.tidygraph.ResIN, 9

Bootstrap_example, 10
BrJSocPsychol_2024, 11

lik_data, 11

p.adjust, *15*
plot.ResIN, 12
print.ResIN, 13

ResIN, 13, *20*, *24*
ResIN_boots_draws, 18
ResIN_boots_execute, 19, *20, 21*, *23, 24*
ResIN_boots_executed, *19*, 20
ResIN_boots_extract, *18*, *20*, 21, *24*
ResIN_boots_prepare, *19, 20*, 22, *23, 24*
ResIN_boots_prepped, *20*, *23*, 23
ResIN_to_gephi, 24
ResIN_to_igraph, 25
ResIN_to_qgraph, 26

summary.ResIN, 27