Package: InterconectaR (via r-universe)

October 16, 2026

Version 1.0.1

Date 2026-10-05

Title Psychological Network Analysis and Psychometric Insights

Encoding UTF-8

Depends R (>= 4.1.0)

Description Estimates psychological networks by group for research in the social sciences and complements them with psychometric modeling. It summarises centrality and bridge centrality indices, evaluates the stability of edges and centrality from bootstrap results, refines items by the stability of exploratory graph analysis (EGA), and plots networks with the variance explained of each node. It also estimates an a priori sample size for EGA through Monte Carlo simulation and includes an interactive application to plan the sample size of a network study. Methods follow Epskamp, Borsboom and Fried (2018) <doi:10.3758/s13428-017-0862-1> and Golino and Epskamp (2017) <doi:10.1371/journal.pone.0174035>.

License GPL (>= 3)

URL https://github.com/jventural/InterconectaR

BugReports https://github.com/jventural/InterconectaR/issues

Imports bootnet, Cairo, correlation, cowplot, dplyr, EGAnet, forcats, future, future.apply, ggforce, ggplot2 (>= 3.4.0), ggplotify, graphics, grDevices, grid, gridExtra, lavaan, magick, mgm, networktools, patchwork, png, progressr, purrr, qgraph, RColorBrewer, rlang, scales, semPlot, stats, tibble, tidyr, tidyselect, tools, utils, wesanderson

Suggests bslib, powerly, psychonetrics, shiny, viridis

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

NeedsCompilation no

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libmagick++-dev gsfonts libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-10-05 11:26:12 UTC

RemoteUrl https://github.com/jventural/interconectar

RemoteRef HEAD

RemoteSha 6d691d0eb780d4b100af5968905734f4ea66cfd6

Contents

boot_and_evaluate . . . 3
calculate_centrality_sd_correlation . . . 4
centrality_bridge_plot . . . 5
centrality_bullet_plot . . . 6
centrality_cleveland_plot . . . 8
centrality_duo_plot . . . 10
centrality_plots2 . . . 11
choose_best_method . . . 12
combine_graphs_centrality . . . 13
combine_graphs_centrality2 . . . 14
combine_groupBy . . . 15
compute_netScores . . . 17
convert_EGA_list_to_df . . . 18
convert_EGA_to_df . . . 19
Density_report . . . 20
estimate_networks_by_group . . . 21
filter_correlation_stability . . . 22
get_edge_weights_summary . . . 23
make_likert_thresholds . . . 24
mgm_error_metrics . . . 25
mgm_errors_groups . . . 26
net_reduce2 . . . 27
optimize_leiden_resolution . . . 28
plot_avg_TEFI . . . 29
plot_centrality_by_group . . . 29
plot_centrality_stability . . . 30
plot_failure_decomposition . . . 32
plot_latent_network_diagram . . . 33
plot_lvm . . . 34
plot_net . . . 35
plot_net_group . . . 38
plot_networks_by_group . . . 41
plot_performance_metrics . . . 43
plot_power_curve_ci . . . 44
plot_sample_size_EGA_montecarlo . . . 45

print.sample_size_EGA_montecarlo 46
print.scale_network_analysis 47
process_LCT 48
qgraph_centrality_panel 48
refine_items_by_stability 50
run_EGA_combinations 52
run_netpowerlab 53
sample_size_EGA_control 54
sample_size_EGA_montecarlo 55
show_plots 58
show_recommendation 59
structure_groups 60
summarise_case_drop_stability 60
summarise_nonparametric_edges 61
summary_metrics 62

Index **64**

`boot_and_evaluate` *Bootstrap and Evaluate Network Analysis*

Description

Performs bootstrap evaluation of network analysis algorithms.

Usage

```
boot_and_evaluate(
  data,
  true_network,
  algorithms,
  correlation_methods,
  sample_sizes = c(100, 250, 500, 1000),
  n_simulations = 100,
  seed = NULL,
  n_cores = 2
)
```

Arguments

`data`	Data frame with the data.
`true_network`	Reference network used as the truth: an object returned by `EGAnet::EGA()` (its `network` element is compared with each estimate).
`algorithms`	Vector of algorithm names to evaluate.
`correlation_methods`	Vector of correlation methods.
`sample_sizes`	Vector of sample sizes to test (default c(100, 250, 500, 1000)).

n_simulations	Number of simulations (default 100).
seed	Optional random seed for reproducible results. NULL (default) leaves the random number generator untouched.
n_cores	Number of cores for parallel processing (default 2). With n_cores = 1 the simulations run sequentially.

Value

Data frame with performance metrics.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

true_net <- EGAnet::EGA(items, plot.EGA = FALSE)
res <- boot_and_evaluate(items, true_network = true_net,
                         algorithms = "walktrap",
                         correlation_methods = "pearson",
                         sample_sizes = c(100, 200),
                         n_simulations = 2, seed = 1, n_cores = 1)
head(res)
```

---

calculate_centrality_sd_correlation *Centrality-SD Correlation*

Description

Calculates correlation between centrality indices and standard deviations.

Usage

```
calculate_centrality_sd_correlation(Data, Centralitys)
```

Arguments

Data	Data frame with the original data (one column per node).
Centralitys	A list with a table element: a data frame with a node column and two centrality indices in its third and fourth columns.

Value

A `correlation` object (from the correlation package) with the correlation of each of the two centrality indices with the standard deviation of the nodes.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
ct <- qgraph::centralityTable(net)
ct <- ct[ct$measure %in% c("Strength", "ExpectedInfluence"),
         c("graph", "node", "measure", "value")]
tab <- tidyr::pivot_wider(ct, names_from = "measure", values_from = "value")
calculate_centrality_sd_correlation(items, list(table = tab))
```

---

centrality_bridge_plot

Bridge Centrality Plot

Description

Creates a comparative bridge centrality plot for two groups.

Usage

```
centrality_bridge_plot(
  networks_groups,
  group_names = c("Mujer", "Varón"),
  measure = "Bridge Expected Influence (1-step)",
  color_palette = c("#FF5733", "#33FFCE"),
  communities = NULL
)
```

Arguments

`networks_groups`	Named list of network objects by group.
`group_names`	Character vector of length 2 with group names.
`measure`	Bridge centrality measure name.
`color_palette`	Character vector of 2 colors for the groups.
`communities`	Community membership vector for bridge calculation.

Value

A list with `table` (a data frame with the bridge centrality of each node in each group, raw and as z-scores) and `plot` (a ggplot object).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
items$sex <- rep(c("Female", "Male"), each = n / 2)

nets <- estimate_networks_by_group(items, group_var = "sex",
                                   columns = names(items)[1:8],
                                   default = "EBICglasso")
res <- centrality_bridge_plot(nets, group_names = c("Female", "Male"),
                              communities = rep(c("A", "B"), each = 4))
res$plot
```

---

centrality_bullet_plot

Bullet Chart para Indices de Centralidad

Description

Crea un grafico tipo Bullet Chart para visualizar dos metricas de centralidad (por ejemplo, Strength y Bridge Strength) de forma compacta y comparativa.

Usage

```
centrality_bullet_plot(
  qgraph_obj,
  network,
  groups = NULL,
  measure0 = "Strength",
  measure1 = "Bridge Strength",
  color_bar = "#3498DB",
  color_marker = "#E74C3C",
  color_background = "#ECF0F1",
  bar_height = 0.6,
  marker_size = 4,
  use_abbrev = TRUE,
  labels = NULL,
```

```
  order_by = "measure0",
  show_values = TRUE,
  text_size = 11
)
```

Arguments

qgraph_obj	Objeto qgraph creado previamente.
network	Objeto de red estimado (por ejemplo, de bootnet::estimateNetwork).
groups	Lista de grupos/comunidades para calcular metricas de puente.
measure0	Metrica principal (barra). Default: "Strength".
measure1	Metrica secundaria (linea marcador). Default: "Bridge Strength".
color_bar	Color de la barra principal. Default: "#3498DB".
color_marker	Color del marcador de referencia. Default: "#E74C3C".
color_background	Color de fondo de la barra. Default: "#ECF0F1".
bar_height	Altura de las barras (0-1). Default: 0.6.
marker_size	Tamano del marcador triangular. Default: 4.
use_abbrev	Usar nombres abreviados. Default: TRUE.
labels	Vector de etiquetas personalizadas para los nodos. Default: NULL.
order_by	Ordenar por: "measure0", "measure1", "name". Default: "measure0".
show_values	Mostrar valores numericos. Default: TRUE.
text_size	Tamano del texto de los nodos. Default: 11.

Value

Lista con dos elementos:

plot	Objeto ggplot con el Bullet Chart.
table	Tabla con los valores de centralidad.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
result <- centrality_bullet_plot(
```

```
  qgraph_obj = g,
  network = net,
  groups = rep(c("A", "B"), each = 4),
  measure0 = "Strength",
  measure1 = "Bridge Strength"
)
result$plot
result$table
```

---

centrality_cleveland_plot

Cleveland Dot Plot para Indices de Centralidad

Description

Crea un grafico tipo Cleveland Dot Plot para visualizar dos metricas de centralidad (por ejemplo, Strength y Bridge Strength). Estilo clasico y elegante para publicaciones.

Usage

```
centrality_cleveland_plot(
  qgraph_obj,
  network,
  groups = NULL,
  measure0 = "Strength",
  measure1 = "Bridge Strength",
  color0 = "#2C3E50",
  color1 = "#E74C3C",
  point_size = 4,
  segment_size = 0.8,
  segment_color = "grey60",
  use_abbrev = TRUE,
  labels = NULL,
  order_by = "measure0",
  show_zero_line = TRUE,
  text_size = 12
)
```

Arguments

qgraph_obj	Objeto qgraph creado previamente.
network	Objeto de red estimado (por ejemplo, de bootnet::estimateNetwork).
groups	Lista de grupos/comunidades para calcular metricas de puente.
measure0	Metrica principal. Default: "Strength".
measure1	Metrica secundaria. Default: "Bridge Strength".

color0	Color para measure0. Default: "#2C3E50".
color1	Color para measure1. Default: "#E74C3C".
point_size	Tamano de los puntos. Default: 4.
segment_size	Grosor de las lineas de conexion. Default: 0.8.
segment_color	Color de la linea que conecta los puntos. Default: "grey60".
use_abbrev	Usar nombres abreviados. Default: TRUE.
labels	Vector de etiquetas personalizadas para los nodos. Default: NULL.
order_by	Ordenar por: "measure0", "measure1", "difference", "name". Default: "measure0".
show_zero_line	Mostrar linea vertical en cero. Default: TRUE.
text_size	Tamano base del texto. Default: 12.

Value

Lista con dos elementos:

plot	Objeto ggplot con el Cleveland Dot Plot.
table	Tabla con los valores de centralidad.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
result <- centrality_cleveland_plot(
  qgraph_obj = g,
  network = net,
  groups = rep(c("A", "B"), each = 4),
  measure0 = "Strength",
  measure1 = "Bridge Strength"
)
result$plot
result$table
```

centrality_duo_plot *Dual Centrality Plot*

Description

Creates a plot comparing two centrality measures.

Usage

```
centrality_duo_plot(
  qgraph_obj,
  network,
  groups = NULL,
  measure0 = "ExpectedInfluence",
  measure1 = NULL,
  color_palette = c("#6C5CE7", "#00B894"),
  use_abbrev = TRUE,
  labels = NULL
)
```

Arguments

qgraph_obj	A qgraph object.
network	Estimated network object (e.g., from bootnet).
groups	List of communities for bridge calculation.
measure0	Primary centrality measure name.
measure1	Secondary bridge measure name (NULL for single measure).
color_palette	Character vector of colors.
use_abbrev	Logical; use abbreviated node names.
labels	Optional character vector of custom node labels.

Value

A list with `table` (a data frame with the centrality values of each node) and `plot` (a ggplot object).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
```

```
net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
res <- centrality_duo_plot(g, net,
                           groups = rep(c("A", "B"), each = 4),
                           measure1 = "Bridge Expected Influence (1-step)")
res$plot
```

`centrality_plots2` *Centrality Plot (Version 2)*

Description

Creates centrality plots with optional bridge metrics.

Usage

```
centrality_plots2(
  qgraph_obj,
  network,
  groups = NULL,
  measure0 = "ExpectedInfluence",
  measure1 = NULL,
  color_palette = c("#FF0000", "#00A08A"),
  labels = NULL,
  legend_labels = NULL,
  use_abbrev = TRUE
)
```

Arguments

`qgraph_obj`	A qgraph object.
`network`	Estimated network object.
`groups`	List of communities for bridge calculation.
`measure0`	Primary centrality measure name.
`measure1`	Secondary bridge measure name (NULL for single measure).
`color_palette`	Character vector of colors.
`labels`	Named vector mapping node names to custom labels.
`legend_labels`	Custom legend labels.
`use_abbrev`	Logical; use abbreviated node names.

Value

A list with `table` (a data frame with the centrality values of each node) and `plot` (a ggplot object).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
res <- centrality_plots2(g, net,
                         groups = rep(c("A", "B"), each = 4),
                         measure1 = "Bridge Expected Influence (1-step)")
res$table
res$plot
```

---

choose_best_method *Choose Best Method*

Description

Selects the best correlation comparison method based on sample size and average inter-correlation.

Usage

```
choose_best_method(n, r_matrix)
```

Arguments

n	Integer, sample size.
r_matrix	Square correlation matrix.

Value

A list with the best method name and details.

Examples

```
r <- matrix(c(1, 0.40, 0.35,
              0.40, 1, 0.30,
              0.35, 0.30, 1), nrow = 3)
choose_best_method(n = 250, r_matrix = r)
```

combine_graphs_centrality

Combine Network and Centrality Graphs

Description

Combines a qgraph network plot with a centrality line plot.

Usage

```
combine_graphs_centrality(
  Figura1_Derecha,
  network,
  groups,
  error_Model,
  labels = NULL,
  abbreviate_labels = FALSE,
  ncol = 2,
  widths = c(0.5, 0.6),
  dpi = 600,
  legend.cex = 0.1,
  layoutScale = c(1, 1),
  vsize = 18,
  threshold = 0
)
```

Arguments

`Figura1_Derecha`	ggplot object with centrality plot.
`network`	Estimated network object.
`groups`	List of node communities.
`error_Model`	Prediction error values for pie chart overlay.
`labels`	Custom node labels.
`abbreviate_labels`	Logical; abbreviate labels to 3 characters.
`ncol`	Number of columns in layout.
`widths`	Relative widths of panels.
`dpi`	Resolution in dots per inch.
`legend.cex`	Legend text size multiplier.
`layoutScale`	Layout scale vector.
`vsize`	Node size.
`threshold`	Minimum edge weight threshold.

Value

A patchwork object with the network on the left and the centrality plot on the right.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
cent <- centrality_plots2(g, net,
                          groups = rep(c("A", "B"), each = 4),
                          measure1 = "Bridge Expected Influence (1-step)")
r2 <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))$R2
combine_graphs_centrality(cent$plot, net, groups = list(A = 1:4, B = 5:8),
                          error_Model = r2, dpi = 100)
```

---

combine_graphs_centrality2

Combine Network and Centrality Graphs (Version 2)

Description

Combines a qgraph network plot with a centrality line plot.

Usage

```
combine_graphs_centrality2(
  Figura1_Derecha,
  network,
  groups,
  error_Model,
  labels = NULL,
  abbreviate_labels = FALSE,
  ncol = 2,
  widths = c(0.5, 0.6),
  dpi = 600,
  legend.cex = 0.1
)
```

Arguments

`Figura1_Derecha`	ggplot object with centrality plot.
`network`	Estimated network object.
`groups`	List of node communities.
`error_Model`	Prediction error values for pie chart overlay.
`labels`	Custom node labels.
`abbreviate_labels`	Logical; abbreviate labels to 3 characters.
`ncol`	Number of columns in layout.
`widths`	Relative widths of panels.
`dpi`	Resolution in dots per inch.
`legend.cex`	Legend text size multiplier.

Value

A patchwork object with the network on the left and the centrality plot on the right.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
cent <- centrality_plots2(g, net,
                          groups = rep(c("A", "B"), each = 4),
                          measure1 = "Bridge Expected Influence (1-step)")
r2 <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))$R2
combine_graphs_centrality2(cent$plot, net, groups = list(A = 1:4, B = 5:8),
                           error_Model = r2, dpi = 100)
```

`combine_groupBy` *Combine Group Plots*

Description

Combines network, centrality, and bridge plots into a labeled panel.

Usage

```
combine_groupBy(
  red_group,
  plot_centralidad_group,
  bridge_plot_group = NULL,
  width_a = 8,
  width_bc = 4.5,
  show_plot = TRUE
)
```

Arguments

red_group	Network graph object (ggplot, grob, or magick-image).
plot_centralidad_group	ggplot centrality plot.
bridge_plot_group	Optional ggplot bridge centrality plot.
width_a	Width of network panel.
width_bc	Width of centrality panels.
show_plot	Logical; display the combined plot.

Value

A ggplot object with the network (panel A), the centrality plot (panel B) and, when supplied, the bridge centrality plot (panel C).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
r2 <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))$R2
net_plot <- plot_net(net$graph, r2 = r2, groups_list = list(A = 1:4, B = 5:8))
cent <- centrality_plots2(g, net)
bridge <- centrality_cleveland_plot(g, net, groups = rep(c("A", "B"), each = 4))
combine_groupBy(net_plot, cent$plot, bridge$plot, show_plot = FALSE)
```

compute_netScores *Compute Network Scores*

Description

Computes network scores with stability analysis using EGA.

Usage

```
compute_netScores(
  item_prefixes,
  data,
  rename_dims = TRUE,
  custom_names = NULL,
  add_sums = TRUE,
  stability_threshold = 0.7,
  stability_corr = "spearman",
  stability_model = "glasso",
  stability_algorithm = "louvain",
  stability_iter = 1000,
  stability_seed_start = NULL,
  stability_type = "resampling",
  stability_ncores = 2,
  stability_max_iter = 10,
  stability_plot = TRUE,
  ega_plot = TRUE,
  verbose = TRUE
)
```

Arguments

`item_prefixes`	Character vector of item prefix patterns.
`data`	Data frame with item responses.
`rename_dims`	Logical; ask for a name for each dimension. The prompt only appears in interactive sessions; otherwise, or when `FALSE`, the default names are used.
`custom_names`	Optional named list of dimension names.
`add_sums`	Logical; compute sum scores per dimension.
`stability_threshold`	Minimum item stability threshold.
`stability_corr`	Correlation method for stability analysis.
`stability_model`	Network model for stability analysis.
`stability_algorithm`	Community detection algorithm.
`stability_iter`	Number of bootstrap iterations.

`stability_seed_start`
: Optional starting random seed (see `refine_items_by_stability()`). `NULL` (default) leaves the seed to `EGAnet::bootEGA()`.

`stability_type` Bootstrap type.

`stability_ncores`
: Number of CPU cores (default 2).

`stability_max_iter`
: Maximum refinement iterations.

`stability_plot` Logical; plot item stability.

`ega_plot` Logical; plot EGA results.

`verbose` Logical; report the progress of each step as messages.

Value

A list of class `ega_network_scores` with `data_complete` (the data with the network scores and sums appended), `net_scores`, `dimension_sums`, `dimensions_list`, `dimension_names`, `ega_objects`, `scale_columns`, `stability_results` and `removed_items`.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("SC", 1:4), paste0("SC", 5:8))

scores <- compute_netScores("SC", items, rename_dims = FALSE,
                            stability_corr = "pearson",
                            stability_iter = 20, stability_seed_start = 1,
                            stability_ncores = 1, stability_plot = FALSE,
                            ega_plot = FALSE)
head(scores$net_scores)
```

`convert_EGA_list_to_df`

Convert EGA List to Data Frame

Description

Converts a list of EGA results into a combined data frame.

Usage

```
convert_EGA_list_to_df(ega_list)
```

Arguments

ega_list	Named list of EGA result objects whose names follow the pattern "<correlation>.<algorithm>", such as the output of run_EGA_combinations.

Value

A data frame with one row per EGA result: the combination name, the correlation method and algorithm parsed from it, and the metrics returned by convert_EGA_to_df.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

combos <- run_EGA_combinations(items, corr = c("pearson", "spearman"),
                               algorithm = c("louvain", "walktrap"))
convert_EGA_list_to_df(combos)
```

---

convert_EGA_to_df *Convert EGA Result to Data Frame*

Description

Extracts network metrics from a single EGA result object.

Usage

```
convert_EGA_to_df(ega_result)
```

Arguments

ega_result	A single EGA result object, as returned by EGAnet::EGA().

Value

A one-row data frame with the network model, correlation method, community detection algorithm, lambda, number of nodes and edges, density, descriptive statistics of the non-zero edge weights, number of communities and TEFI.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

ega <- EGAnet::EGA(items, plot.EGA = FALSE)
convert_EGA_to_df(ega)
```

Density_report *Network Density Report*

Description

Calculates and reports network density statistics.

Usage

```
Density_report(edge.matrix)
```

Arguments

edge.matrix	Network edge matrix (e.g., from qgraph).

Value

A character string reporting how many edges differ from zero out of the possible edges, and the resulting density as a percentage.

Examples

```
w <- matrix(c(0, 0.3, 0,
              0.3, 0, 0.2,
              0, 0.2, 0), nrow = 3,
            dimnames = list(c("A", "B", "C"), c("A", "B", "C")))
Density_report(w)
```

estimate_networks_by_group

Estimate Networks by Group

Description

Estimates network models separately for each group in the data.

Usage

```
estimate_networks_by_group(
  data,
  group_var,
  columns,
  default = "ggmModSelect",
  stepwise = TRUE,
  corMethod = "spearman",
  abbreviate_vars = FALSE,
  abbr_minlength = 3,
  labels = NULL,
  set_labels_on_graph = TRUE
)
```

Arguments

`data`	Data frame containing the data.
`group_var`	Name of the grouping variable column.
`columns`	Character vector of variable names to include.
`default`	Network estimation method (default: "ggmModSelect").
`stepwise`	Logical; use stepwise model selection.
`corMethod`	Correlation method (default: "spearman").
`abbreviate_vars`	Logical; abbreviate variable names.
`abbr_minlength`	Minimum length for abbreviated names.
`labels`	Optional custom labels for nodes.
`set_labels_on_graph`	Logical; apply labels to graph object.

Value

A named list with one estimated network (a `bootnetResult` object) per group. Each network carries the attribute `var_mapping`, a tibble that maps the original variable names to the labels used.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
items$sex <- rep(c("Female", "Male"), each = n / 2)

nets <- estimate_networks_by_group(items, group_var = "sex",
                                   columns = names(items)[1:8],
                                   default = "EBICglasso")
names(nets)
```

---

filter_correlation_stability

Filter Correlation Stability

Description

Extracts and filters correlation stability indices from bootstrap results.

Usage

```
filter_correlation_stability(caseDroppingBoot)
```

Arguments

`caseDroppingBoot`
: Case-dropping bootstrap results from `bootnet::bootnet(..., type = "case")`, computed with the statistics `"strength"`, `"expectedInfluence"`, `"bridgeStrength"` and `"bridgeExpectedInfluence"` (or a subset of them).

Value

A data frame with the columns `rowname` (the centrality index) and `Index` (its correlation stability coefficient, CS, rounded to two decimals).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
```

```
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
case_boot <- bootnet::bootnet(net, nBoots = 20, type = "case",
                              statistics = c("strength", "expectedInfluence"),
                              nCores = 1, verbose = FALSE)
filter_correlation_stability(case_boot)
```

---

get_edge_weights_summary *Edge Weights Summary*

Description

Computes descriptive statistics for network edge weights.

Usage

```
get_edge_weights_summary(network, abs_weights = TRUE, round_digits = 2)
```

Arguments

network	Network object with a graph element (e.g., from `bootnet::estimateNetwork()`).
abs_weights	Logical; use absolute edge weights.
round_digits	Number of decimal places for rounding.

Value

A one-row data frame with the number of non-zero edges (`N`) and the mean, standard deviation, minimum and maximum of their weights.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
get_edge_weights_summary(net)
```

make_likert_thresholds

Generate Likert-Style Threshold Patterns

Description

Convenience helper to build per-item thresholds for the latent normal scale used by `sample_size_EGA_montecarlo()` when `data_type = "ordinal"`. Useful for simulating realistic Likert scenarios with floor or ceiling effects rather than the unrealistic equiprobable default.

Patterns are parameterized via a Beta distribution over the latent cumulative scale: `shape1 = shape2 = 1` gives equiprobable categories, `shape1 < shape2` concentrates mass on lower categories (floor), `shape1 > shape2` concentrates mass on upper categories (ceiling).

Usage

```
make_likert_thresholds(
  n_items,
  categories,
  pattern = c("equiprobable", "floor", "ceiling", "mixed", "custom"),
  shape1 = NULL,
  shape2 = NULL,
  seed = NULL
)
```

Arguments

`n_items`	Integer number of items.
`categories`	Integer number of ordinal response categories (>= 2).
`pattern`	One of `"equiprobable"`, `"floor"`, `"ceiling"`, `"mixed"` (half of items floor, half ceiling, shuffled), or `"custom"`.
`shape1`	First Beta shape parameter, used when `pattern = "custom"`. Ignored otherwise.
`shape2`	Second Beta shape parameter, used when `pattern = "custom"`. Ignored otherwise.
`seed`	Optional integer seed for the shuffling step in `pattern = "mixed"`.

Value

A list of length `n_items`, each element a numeric vector of `categories - 1` thresholds on the standard normal scale.

Examples

```
# 18 items, 5 categories, all with floor effect
tau_floor <- make_likert_thresholds(18, 5, "floor")
tau_floor[[1]]
```

```
# Half floor, half ceiling, randomly interleaved
tau_mixed <- make_likert_thresholds(18, 5, "mixed", seed = 1)

# Custom skew via Beta(1.5, 4)
tau_custom <- make_likert_thresholds(18, 5, "custom",
                                     shape1 = 1.5, shape2 = 4)
```

`mgm_error_metrics` *MGM Error Metrics*

Description

Fits a Mixed Graphical Model and extracts prediction error metrics.

Usage

```
mgm_error_metrics(data, type, level, k = 2)
```

Arguments

`data`	Data frame or matrix with the data.
`type`	Character vector of variable types ("g", "c", "p").
`level`	Integer vector of variable levels.
`k`	Order of interactions (default: 2).

Value

A list with the node predictability of the model: `errorCon`, a data frame with the RMSE and R2 of the continuous variables; `errorCat`, a data frame with the accuracy (CC) and normalized accuracy (nCC) of the categorical variables; and, when all variables are of the same kind, a named vector `R2` or `nCC` ready to be used as the `pie` argument of qgraph.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

errors <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))
errors$errorCon
```

mgm_errors_groups *MGM Errors by Group*

Description

Fits Mixed Graphical Models and extracts errors for each group.

Usage

```
mgm_errors_groups(data, type, level, group, columns)
```

Arguments

data	Data frame containing the data.
type	Character vector of variable types.
level	Integer vector of variable levels.
group	Unquoted grouping variable name.
columns	Character vector of column names to analyze.

Value

A named list with one tibble per group, holding the RMSE, R2, CC and nCC of each variable (in the order of `columns`).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
items$sex <- rep(c("Female", "Male"), each = n / 2)

mgm_errors_groups(items, type = rep("g", 8), level = rep(1, 8),
                  group = sex, columns = names(items)[1:8])
```

net_reduce2 *Network Reduction*

Description

Reduces redundant node pairs using PCA or best goldbricker selection.

Usage

```
net_reduce2(
  data,
  badpairs,
  method = c("PCA", "best_goldbricker"),
  keep_unpaired = TRUE
)
```

Arguments

data	Data frame with item responses.
badpairs	Goldbricker object or list of redundant node pairs.
method	Reduction method: "PCA" or "best_goldbricker".
keep_unpaired	Logical; keep unpaired items in result.

Value

A list with data (the reduced data frame), removed_items, added_variables, kept_from_original and pair_summary (a data frame describing what was done with each redundant pair).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
# a near-duplicate of A1
items$A1_copy <- items$A1 + rnorm(n, 0, 0.2)

redundant <- networktools::goldbricker(items, threshold = 0.5)
reduced <- net_reduce2(items, redundant, method = "PCA")
reduced$pair_summary
```

optimize_leiden_resolution

Optimize Leiden Resolution

Description

Optimizes the resolution parameter for Leiden community detection.

Usage

```
optimize_leiden_resolution(
  data,
  corr = "cor_auto",
  gamma_values = c(0.01, 0.05, 0.1, 0.5, 1),
  objective_function = "CPM",
  verbose = TRUE
)
```

Arguments

`data`	Data frame with the data.
`corr`	Correlation method (default: "cor_auto").
`gamma_values`	Numeric vector of gamma resolution values to test.
`objective_function`	Leiden objective function (default: "CPM").
`verbose`	Logical; print progress messages.

Value

A list with `best_gamma` (the resolution with the lowest TEFI), `max_valid_gamma` (the largest resolution tested before TEFI became undefined), `best_model` (the EGA object for `best_gamma`), `all_results` (the EGA objects of the valid resolutions), `comparison` (a data frame with the TEFI and number of communities per resolution) and `optimization_plot` (a ggplot of TEFI against gamma).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
```

```
opt <- optimize_leiden_resolution(items, corr = "pearson",
                                  gamma_values = c(0.05, 0.1, 0.5),
                                  verbose = FALSE)
opt$comparison
```

---

plot_avg_TEFI *Plot Average TEFI by Sample Size*

Description

Plots median TEFI values across different sample sizes, algorithms, and correlation methods.

Usage

```
plot_avg_TEFI(results)
```

Arguments

results	Data frame from boot_and_evaluate with TEFI values.

Value

A ggplot object showing TEFI trends.

Examples

```
# A small table with the structure returned by boot_and_evaluate()
set.seed(1)
res <- expand.grid(Algorithm = c("louvain", "walktrap"),
                   Correlation_Method = c("pearson", "spearman"),
                   Sample_Size = c(100, 250, 500), Simulation = 1:5,
                   stringsAsFactors = FALSE)
res$TEFI <- rnorm(nrow(res), -5, 0.3)

plot_avg_TEFI(res)
```

---

plot_centrality_by_group *Plot centrality measures by group*

Description

Plot centrality measures by group

Usage

```
plot_centrality_by_group(
  networks_groups,
  replacements,
  measure_spec,
  color_palette = c("#FF5733", "#33FFCE")
)
```

Arguments

networks_groups	A named list of two network objects.
replacements	Character vector of length 2 with group replacement names.
measure_spec	Character string specifying the centrality measure.
color_palette	Character vector of two colors for the groups.

Value

A list with elements `plot` (a ggplot object) and `table` (a data frame).

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
items$sex <- rep(c("Female", "Male"), each = n / 2)

nets <- estimate_networks_by_group(items, group_var = "sex",
                                   columns = names(items)[1:8],
                                   default = "EBICglasso")
res <- plot_centrality_by_group(nets, replacements = c("Female", "Male"),
                                measure_spec = "ExpectedInfluence")
res$plot
```

---

plot_centrality_stability

Plot centrality stability diagnostics

Description

Plot centrality stability diagnostics

Usage

```
plot_centrality_stability(
  caseDroppingBoot,
  nonParametricBoot,
  statistics = "bridgeStrength",
  labels = FALSE,
  color_palette = NULL
)
```

Arguments

`caseDroppingBoot`
: Case-dropping bootstrap result object.

`nonParametricBoot`
: Non-parametric bootstrap result object.

`statistics`
: Character string specifying the statistic (default: "bridgeStrength").

`labels`
: Logical; whether to show labels (default: FALSE).

`color_palette`
: Optional character vector of colors for the statistics. If NULL (default), uses bootnet's default colors. Length should match the number of statistics being plotted.

Value

A combined patchwork plot object.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
case_boot <- bootnet::bootnet(net, nBoots = 20, type = "case",
                              statistics = c("bridgeStrength", "strength"),
                              communities = rep(c("A", "B"), each = 4),
                              nCores = 1, verbose = FALSE)
edge_boot <- bootnet::bootnet(net, nBoots = 20, nCores = 1, verbose = FALSE)
plot_centrality_stability(case_boot, edge_boot)
```

plot_failure_decomposition

Plot Replication-Level Failure Decomposition Per Sample Size

Description

Stacked-bar visualization showing what fraction of Monte Carlo replications fails each constraint at every sample size. The bars decompose `1 - power_combined` into mutually exclusive failure modes (in priority order): non-convergence, incorrect dimension recovery, and one bar per target metric whose threshold was not met. The complementary green segment shows the success rate (`power_combined`).

This is an original visualization (not present in powerly) intended to diagnose which constraint is the bottleneck at low `n`: e.g., is it convergence, dimension recovery, or a specific recovery target?

Usage

```
plot_failure_decomposition(x, order_targets = NULL, ...)
```

Arguments

`x`	Object returned by `sample_size_EGA_montecarlo()`.
`order_targets`	Optional character vector reordering the failure-mode priorities for the metric targets. The default order is `convergence -> dimensions -> targets` in the order they appear in `x$targets -> success`.
`...`	Ignored.

Value

A ggplot object.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
plot_failure_decomposition(res)
```

plot_latent_network_diagram

Plot a latent network diagram (SEM path diagram)

Description

Plot a latent network diagram (SEM path diagram)

Usage

```
plot_latent_network_diagram(
  model_object,
  filename,
  output_path = NULL,
  width = 9.5,
  height = 6.5,
  resolution = 1500,
  node_labels = NULL,
  color_scheme = NULL,
  color_palette = "auto",
  plot_settings = NULL,
  device_settings = NULL,
  edge_settings = NULL,
  save_message = TRUE,
  show_manifest = TRUE
)
```

Arguments

`model_object`	A fitted model: a `lavaan` object or a `psychonetrics` latent variable model (the latter needs the psychonetrics package).
`filename`	Output filename (with extension .png, .jpg, or .pdf).
`output_path`	Directory where the file is written. It has no default, so nothing is written unless you choose a location (use `tempdir()` for a temporary one).
`width`	Plot width in inches.
`height`	Plot height in inches.
`resolution`	Plot resolution in DPI.
`node_labels`	Optional character vector of node labels.
`color_scheme`	Optional list with elements `man` and `lat` for colors.
`color_palette`	Color palette type (default: "auto").
`plot_settings`	Optional list of semPaths plot settings.
`device_settings`	Optional list of device settings.
`edge_settings`	Optional list of edge display settings.

`save_message` Logical; report the path of the saved file as a message (default: TRUE).

`show_manifest` Logical; show manifest variables (default: TRUE).

Value

Invisibly, the path of the saved file. Called for its side effect: the diagram is written to `file.path(output_path, filename)`.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

fit <- lavaan::cfa("F1 =~ A1 + A2 + A3 + A4
                    F2 =~ B1 + B2 + B3 + B4", data = items)
plot_latent_network_diagram(fit, filename = "diagram.png",
                            output_path = tempdir(), resolution = 100)
```

---

plot_lvm *Plot Latent Variable Models*

Description

Creates visualizations for psychonetrics latent variable models.

Usage

```
plot_lvm(
  x,
  plot = c("network", "loadings", "residcors", "residpcors", "latents"),
  ask,
  rotation = promax,
  vsize = c(5, 5),
  palette = "Set1",
  ...
)
```

Arguments

x	A psychonetrics model object.
plot	Types of plots to create (network, loadings, residcors, residpcors, latents).
ask	Logical, whether to ask before each plot.
rotation	Rotation function (default promax).
vsize	Node size vector (default c(5, 5)).
palette	RColorBrewer palette name (default "Set1").
...	Additional arguments passed to qgraph.

Value

Invisible list of qgraph objects.

Examples

```
if (requireNamespace("psychonetrics", quietly = TRUE)) {
  set.seed(123)
  n <- 300
  f1 <- rnorm(n)
  f2 <- 0.3 * f1 + rnorm(n)
  items <- data.frame(
    sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
    sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
  )
  names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

  lambda <- matrix(0, 8, 2)
  lambda[1:4, 1] <- 1
  lambda[5:8, 2] <- 1
  mod <- psychonetrics::lvm(items, lambda = lambda, verbose = FALSE)
  mod <- psychonetrics::runmodel(mod, verbose = FALSE)
  plot_lvm(mod, plot = c("loadings", "latents"))
}
```

---

plot_net *Plot a single network with R2 progress rings*

Description

Creates a ggplot of a single network with nodes colored by domain, R2 progress rings, gradient-colored edges, and optional abbreviated labels with a legend mapping.

Usage

```
plot_net(
  mat,
  r2,
  labels = colnames(mat),
  short_labels = NULL,
  full_labels = NULL,
  groups_list = NULL,
  group_cols = NULL,
  palette = "Darjeeling1",
  node_radius = 0.2,
  node_label_size = NULL,
  ring_offset = 0.045,
  ring_stroke = 2.4,
  ring_progress_col = "black",
  edge_min = 0,
  lwd_range = c(0.4, 4.5),
  alpha_range = c(0.15, 1),
  edge_color_pos = NULL,
  edge_color_neg = NULL,
  edge_curvature = 0.2,
  layout = "spring",
  layout_repulsion = 1,
  show_edge_values = TRUE,
  edge_label_digits = 2,
  edge_label_size = 3.2,
  edge_label_offset = 0.05,
  edge_label_bg = TRUE,
  edge_label_bg_fill = "white",
  edge_label_bg_alpha = 0.9,
  show_R2_text = TRUE,
  R2_text_size = NULL,
  R2_text_color = "grey30",
  R2_digits = 2,
  R2_position = "below",
  title = NULL,
  subtitle = NULL,
  caption = "default"
)
```

Arguments

`mat`	Adjacency matrix.
`r2`	Numeric vector of R2 values per node.
`labels`	Character vector of node labels (default: colnames of mat).
`short_labels`	Optional character vector of abbreviations displayed inside nodes.
`full_labels`	Optional character vector of full names for the caption legend.

`groups_list` Named list mapping domain names to node indices.

`group_cols` Named character vector of colors per domain.

`palette` Wes Anderson palette used when `group_cols` or edge colors are not supplied.

`node_radius` Numeric radius of node circles.

`node_label_size` Text size for node labels (NULL = auto).

`ring_offset` Gap between node circle and R2 ring.

`ring_stroke` Line width of the R2 ring.

`ring_progress_col` Color of the R2 progress arc.

`edge_min` Minimum absolute edge weight to display.

`lwd_range` Numeric vector of length 2: min and max line widths.

`alpha_range` Numeric vector of length 2: min and max alpha for edges.

`edge_color_pos` Color for the strongest positive edge.

`edge_color_neg` Color for the strongest negative edge.

`edge_curvature` Curvature of edges (0 = straight).

`layout` Layout algorithm: "spring" (Fruchterman-Reingold), "circle", or a Nx2 matrix.

`layout_repulsion` Repulsion parameter for FR layout (higher = more spread).

`show_edge_values` Logical; show edge weight labels?

`edge_label_digits` Number of decimal digits for edge labels.

`edge_label_size` Text size for edge labels.

`edge_label_offset` Perpendicular offset for edge labels.

`edge_label_bg` Logical; use background box for edge labels?

`edge_label_bg_fill` Fill color of the edge label background.

`edge_label_bg_alpha` Alpha of the edge label background.

`show_R2_text` Logical; show R2 numeric text near nodes?

`R2_text_size` Text size for R2 labels (NULL = auto).

`R2_text_color` Color for R2 labels.

`R2_digits` Decimal digits for R2 labels.

`R2_position` Position of R2 text: "below", "above", "right", "left".

`title` Optional plot title.

`subtitle` Optional plot subtitle.

`caption` Caption text. '"default"' explains the R2 ring; 'NULL' removes the caption.

Value

A ggplot object.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
r2 <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))$R2
plot_net(
  mat = net$graph,
  r2  = r2,
  groups_list = list(Anxiety = 1:4, Depression = 5:8),
  short_labels = names(items),
  full_labels  = paste("Item", 1:8)
)
```

---

plot_net_group *Plot two networks side by side with R2 progress rings*

Description

Creates a faceted ggplot comparing two group networks (e.g., female vs male) with nodes colored by domain, R2 progress rings, gradient-colored edges, and optional abbreviated labels with a legend mapping.

Usage

```
plot_net_group(
  mat1,
  mat2,
  r2_1,
  r2_2,
  group_names = c("Femenino", "Masculino"),
  labels = colnames(mat1),
  short_labels = NULL,
  full_labels = NULL,
  groups_list = NULL,
  group_cols = NULL,
  palette = "Darjeeling1",
```

```
  node_radius = 0.2,
  node_label_size = NULL,
  ring_offset = 0.045,
  ring_stroke = 2.4,
  ring_progress_col = "black",
  edge_min = 0,
  lwd_range = c(0.4, 4.5),
  alpha_range = c(0.15, 1),
  edge_color_pos = NULL,
  edge_color_neg = NULL,
  edge_curvature = 0,
  show_edge_values = TRUE,
  edge_label_digits = 2,
  edge_label_size = 3.2,
  edge_label_offset = 0.05,
  edge_label_bg = TRUE,
  edge_label_bg_fill = "white",
  edge_label_bg_alpha = 0.9,
  show_R2_text = TRUE,
  R2_text_size = NULL,
  R2_text_color = "grey30",
  R2_digits = 2,
  R2_position = "below",
  title = NULL,
  subtitle = NULL,
  caption = "default"
)
```

Arguments

`mat1`	Adjacency matrix for the first group.
`mat2`	Adjacency matrix for the second group.
`r2_1`	Numeric vector of R2 values for the first group.
`r2_2`	Numeric vector of R2 values for the second group.
`group_names`	Character vector of length 2 with facet labels (default: c("Femenino","Masculino")).
`labels`	Character vector of node labels (default: colnames of mat1).
`short_labels`	Optional character vector of abbreviations displayed inside nodes.
`full_labels`	Optional character vector of full names for the caption legend.
`groups_list`	Named list mapping domain names to node indices.
`group_cols`	Named character vector of colors per domain.
`palette`	Wes Anderson palette used when `group_cols` or edge colors are not supplied.
`node_radius`	Numeric radius of node circles.
`node_label_size`	Text size for node labels (NULL = auto).
`ring_offset`	Gap between node circle and R2 ring.

`ring_stroke` Line width of the R2 ring.

`ring_progress_col` Color of the R2 progress arc.

`edge_min` Minimum absolute edge weight to display.

`lwd_range` Numeric vector of length 2: min and max line widths.

`alpha_range` Numeric vector of length 2: min and max alpha for edges.

`edge_color_pos` Color for the strongest positive edge.

`edge_color_neg` Color for the strongest negative edge.

`edge_curvature` Curvature of edges (0 = straight).

`show_edge_values` Logical; show edge weight labels?

`edge_label_digits` Number of decimal digits for edge labels.

`edge_label_size` Text size for edge labels.

`edge_label_offset` Perpendicular offset for edge labels.

`edge_label_bg` Logical; use background box for edge labels?

`edge_label_bg_fill` Fill color of the edge label background.

`edge_label_bg_alpha` Alpha of the edge label background.

`show_R2_text` Logical; show R2 numeric text near nodes?

`R2_text_size` Text size for R2 labels (NULL = auto).

`R2_text_color` Color for R2 labels.

`R2_digits` Decimal digits for R2 labels.

`R2_position` Position of R2 text: "below", "above", "right", "left".

`title` Optional plot title.

`subtitle` Optional plot subtitle.

`caption` Caption text. '"default"' explains the R2 ring; 'NULL' removes the caption.

Value

A ggplot object.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
```

```
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
sex <- rep(c("Female", "Male"), each = n / 2)

net_f <- bootnet::estimateNetwork(items[sex == "Female", ], default = "EBICglasso")
net_m <- bootnet::estimateNetwork(items[sex == "Male", ], default = "EBICglasso")
r2_f <- mgm_error_metrics(items[sex == "Female", ], type = rep("g", 8),
                          level = rep(1, 8))$R2
r2_m <- mgm_error_metrics(items[sex == "Male", ], type = rep("g", 8),
                          level = rep(1, 8))$R2
plot_net_group(
  mat1 = net_f$graph, mat2 = net_m$graph,
  r2_1 = r2_f, r2_2 = r2_m,
  group_names = c("Female", "Male"),
  groups_list = list(Anxiety = 1:4, Depression = 5:8)
)
```

---

plot_networks_by_group

Plot networks by group as a combined panel

Description

Plot networks by group as a combined panel

Usage

```
plot_networks_by_group(
  networks_by_group,
  width = 7,
  height = 7,
  units = "in",
  res = 300,
  groups = list(Factor1 = 1:6, Factor2 = 7:12),
  labels = NULL,
  pie = NULL,
  legend.cex = 0.5,
  GLratio = 1.5,
  ncol = NULL,
  plot_margin = 20,
  title_spacing = 6,
  show_plot = TRUE,
  layoutScale = c(1, 1),
  vsize = 12,
  esize = 20,
  node.width = 1,
  edge.labels = TRUE,
```

```
  edge.label.cex = 1.5,
  curveAll = 2,
  minimum = 0.1,
  border.width = 2.5,
  label.cex = 1,
  title.cex = 1.6,
  palette = "ggplot2",
  color = NULL
)
```

Arguments

networks_by_group	A named list of network objects (one per group).
width	Plot width in inches.
height	Plot height in inches.
units	Units for width/height.
res	Plot resolution in DPI.
groups	Named list mapping factor names to node indices.
labels	Optional character vector of node labels.
pie	Optional named list of pie values per group.
legend.cex	Legend text size.
GLratio	Graph-to-legend ratio.
ncol	Number of columns for the panel layout.
plot_margin	Margin around each plot.
title_spacing	Spacing for titles.
show_plot	Logical; whether to print the plot.
layoutScale	Numeric vector of length 2 for layout scaling.
vsize	Node size.
esize	Edge size.
node.width	Node width.
edge.labels	Logical; show edge labels.
edge.label.cex	Edge label text size.
curveAll	Edge curvature.
minimum	Minimum edge weight to display.
border.width	Node border width.
label.cex	Node label text size.
title.cex	Title text size.
palette	Color palette name.
color	Optional vector of node colors per group.

Value

A combined ggplot (patchwork) object.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))
items$sex <- rep(c("Female", "Male"), each = n / 2)

nets <- estimate_networks_by_group(items, group_var = "sex",
                                   columns = names(items)[1:8],
                                   default = "EBICglasso")
p <- plot_networks_by_group(nets, groups = list(A = 1:4, B = 5:8),
                            res = 100, show_plot = FALSE)
p
```

---

plot_performance_metrics

Plot Network Performance Metrics

Description

Creates faceted plots comparing performance metrics across algorithms and correlation methods.

Usage

```
plot_performance_metrics(averaged_results)
```

Arguments

`averaged_results`
: Data frame with averaged metrics from summary_metrics.

Value

A ggplot object with performance comparison.

Examples

```
# A small table with the structure returned by boot_and_evaluate()
set.seed(1)
res <- expand.grid(Algorithm = c("louvain", "walktrap"),
                   Correlation_Method = c("pearson", "spearman"),
                   Sample_Size = c(100, 250, 500), Simulation = 1:5,
                   stringsAsFactors = FALSE)
k <- nrow(res)
res$sensitivity <- runif(k, 0.6, 0.9)
res$specificity <- runif(k, 0.7, 0.95)
res$precision <- runif(k, 0.6, 0.9)
res$correlation <- runif(k, 0.7, 0.95)
res$abs_cor <- runif(k, 0.7, 0.95)
res$bias <- runif(k, 0.01, 0.05)
res$TEFI <- rnorm(k, -5, 0.3)
res$FDR <- runif(k, 0.05, 0.2)

plot_performance_metrics(summary_metrics(res))
```

---

plot_power_curve_ci *Plot the Power Curve with Bootstrap Confidence Band*

Description

Visualizes the isotonic power curve produced by `sample_size_EGA_montecarlo()` together with a 95 percent bootstrap band, the recommended sample size and its confidence interval as a bracket on the x axis, and (optionally) the empirical validation point.

Conceptually related to the bootstrap-spline plot in the powerly package (Constantin et al., 2023), but distinct in styling, color palette, annotations (bracket on the x axis, validation marker) and underlying object (combined power, not an arbitrary statistic).

Usage

```
plot_power_curve_ci(
  x,
  show_validation = TRUE,
  show_observed = TRUE,
  ribbon_alpha = 0.25,
  palette = c(median = "#2A6F4D", ribbon = "#7FB99B", accent = "#C25E2A"),
  ...
)
```

Arguments

`x`	Object returned by `sample_size_EGA_montecarlo()` run with `recommendation_method = "interpolated"`.

show_validation	Logical; overlay the empirical power at the validation n (when validation_n_rep > 0 was used).
show_observed	Logical; overlay the observed (non-bootstrapped) power per grid point.
ribbon_alpha	Numeric in (0, 1); transparency of the bootstrap band.
palette	Character vector of length 3 with colors for: median curve, bootstrap band, validation/recommendation accents.
...	Ignored. For plot() compatibility.

Value

A ggplot object.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
plot_power_curve_ci(res)
```

---

plot_sample_size_EGA_montecarlo

Plot Monte Carlo EGA Sample Size Results

Description

Creates a visual summary of the output from sample_size_EGA_montecarlo(). The plot shows selected performance metrics across candidate sample sizes, target criteria, and the recommended sample size when one is available. Optionally adds 95 percent Monte Carlo uncertainty ribbons for ARI and edge correlation.

Usage

```
plot_sample_size_EGA_montecarlo(
  x,
  metrics = c("p_dimensions", "median_ari", "median_edge_correlation",
   "median_sensitivity", "median_specificity", "median_precision", "median_fdr"),
  show_criteria = TRUE,
  show_recommended = TRUE,
  show_uncertainty = TRUE,
  point_size = 2.4,
  line_width = 0.8,
  ...
)
```

Arguments

`x`	Object returned by `sample_size_EGA_montecarlo()`.
`metrics`	Character vector with summary metrics to plot. Defaults to the main recovery metrics bounded between 0 and 1.
`show_criteria`	Logical; draw horizontal target lines when criteria are available.
`show_recommended`	Logical; draw a vertical line at the recommended sample size when available.
`show_uncertainty`	Logical; draw 95 percent Monte Carlo bands when the relevant quantile columns are present in `x$summary`.
`point_size`	Numeric point size.
`line_width`	Numeric line width.
`...`	Ignored. Included for compatibility with `plot()`.

Value

A ggplot object.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
plot_sample_size_EGA_montecarlo(res)
```

`print.sample_size_EGA_montecarlo` *Narrative summary of a sample-size simulation*

Description

Prints a one-paragraph dynamic summary of the recommended sample size together with the structural assumptions that led to it (number of items, number of dimensions, factor sizes, replications, power target, validation result and sensitivity sweep).

Usage

```
## S3 method for class 'sample_size_EGA_montecarlo'
print(x, digits = 3, show_plots = FALSE, ...)
```

Arguments

x	A sample_size_EGA_montecarlo object.
digits	Number of significant digits for non-integer values.
show_plots	Logical; if TRUE and the object carries plots (plots = TRUE at fit time), each plot is rendered after the narrative. Default FALSE.
...	Ignored.

Value

Invisibly returns x.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
print(res)
```

---

print.scale_network_analysis

Print method for scale_network_analysis objects

Description

Print method for scale_network_analysis objects

Usage

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

Arguments

x	An object of class scale_network_analysis.
...	Additional arguments passed to print methods.

Value

Invisibly returns the input object.

process_LCT — *Process LCT (Loadings Comparison Test) results*

Description

Process LCT (Loadings Comparison Test) results

Usage

```
process_LCT(...)
```

Arguments

`...`	One or more LCT result objects (from `EGAnet::LCT()`), each with the elements `empirical`, `bootstrap` and `proportion`. The name of each argument is used as its identifier.

Value

A tibble summarizing the LCT results.

Examples

```
# Two objects with the structure returned by EGAnet::LCT()
scale_a <- list(empirical = "Factor", bootstrap = "Factor",
                proportion = c(Factor = 0.86, Network = 0.14))
scale_b <- list(empirical = "Network", bootstrap = "Network",
                proportion = c(Factor = 0.22, Network = 0.78))
process_LCT(scale_a, scale_b)
```

qgraph_centrality_panel — *Create a combined panel with qgraph network and centrality plot*

Description

Create a combined panel with qgraph network and centrality plot

Usage

```
qgraph_centrality_panel(
  Figura1_Derecha,
  network,
  groups,
  error_Model,
  labels = NULL,
```

```
  abbreviate_labels = FALSE,
  ncol = 2,
  widths = c(0.5, 0.6),
  dpi = 600,
  legend.cex = 0.1,
  legend_title = "Métrica",
  keep_centrality_scale = TRUE,
  layoutScale = c(0.9, 0.9),
  GLratio = 2,
  vsize = 18,
  esize = 18,
  node.width = 0.8,
  curveAll = 2,
  palette = "pastel",
  layout = "spring",
  edge.labels = TRUE,
  edge.label.cex = 1
)
```

Arguments

`Figura1_Derecha`
: A ggplot object for the right panel (centrality plot).

`network`
: A network object with elements `graph` and `labels`.

`groups`
: Named list mapping factor names to node indices.

`error_Model`
: Pie values for error model visualization.

`labels`
: Optional character vector of node labels.

`abbreviate_labels`
: Logical; abbreviate labels to 3 characters.

`ncol`
: Number of columns in the layout.

`widths`
: Relative widths of the panels.

`dpi`
: Resolution in DPI for the qgraph rendering.

`legend.cex`
: Legend text size for qgraph.

`legend_title`
: Title for the legend.

`keep_centrality_scale`
: Logical; keep original scale of centrality plot.

`layoutScale`
: Numeric vector of length 2 for layout scaling.

`GLratio`
: Graph-to-legend ratio.

`vsize`
: Node size.

`esize`
: Edge size.

`node.width`
: Node width.

`curveAll`
: Edge curvature.

`palette`
: Color palette name.

layout	Layout algorithm.
edge.labels	Logical; show edge labels.
edge.label.cex	Edge label text size.

Value

A combined patchwork plot object.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
g <- qgraph::qgraph(net$graph, DoNotPlot = TRUE)
cent <- centrality_plots2(g, net,
                          groups = rep(c("A", "B"), each = 4),
                          measure1 = "Bridge Expected Influence (1-step)")
r2 <- mgm_error_metrics(items, type = rep("g", 8), level = rep(1, 8))$R2
qgraph_centrality_panel(cent$plot, net, groups = list(A = 1:4, B = 5:8),
                        error_Model = r2, dpi = 100)
```

---

refine_items_by_stability

Iteratively refine items by bootstrap stability

Description

Iteratively refine items by bootstrap stability

Usage

```
refine_items_by_stability(
  data,
  threshold = 0.8,
  corr = "spearman",
  model = "glasso",
  algorithm = "louvain",
  iter = 1000,
  seed_start = NULL,
  type = "resampling",
```

```
  ncores = 2,
  max_iter = 10,
  plot.itemStability = FALSE
)
```

Arguments

`data`	A data frame of items to analyze.
`threshold`	Stability threshold (default: 0.80).
`corr`	Correlation method (default: "spearman").
`model`	Network model (default: "glasso").
`algorithm`	Community detection algorithm (default: "louvain").
`iter`	Number of bootstrap iterations (default: 1000).
`seed_start`	Optional starting seed. When supplied, refinement round `i` uses the seed `seed_start + i - 1`, which makes the results reproducible. `NULL` (default) leaves the seed to `EGAnet::bootEGA()`.
`type`	Bootstrap type (default: "resampling").
`ncores`	Number of cores for parallel processing (default: 2).
`max_iter`	Maximum number of refinement iterations (default: 10).
`plot.itemStability`	Logical; plot item stability (default: FALSE).

Value

A list containing the final bootstrap result, stability data, removed items, plots, and dimension consistency tables.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

refined <- refine_items_by_stability(items, corr = "pearson", iter = 20,
                                     seed_start = 1, ncores = 1)
refined$stability_df
refined$removed_items
```

run_EGA_combinations *Run EGA Combinations*

Description

Runs EGA with multiple combinations of correlation methods and algorithms.

Usage

```
run_EGA_combinations(
  data,
  corr = c("cor_auto", "pearson", "spearman"),
  algorithm = c("leiden", "louvain", "walktrap"),
  leiden_args = list(objective_function = "CPM", resolution_parameter = 1)
)
```

Arguments

data	Data frame with the data.
corr	Character vector of correlation methods.
algorithm	Character vector of community detection algorithms.
leiden_args	List of additional arguments for the Leiden algorithm.

Value

A list of EGA results with class "EGA_combinations".

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

combos <- run_EGA_combinations(items, corr = c("pearson", "spearman"),
                               algorithm = c("louvain", "walktrap"))
names(combos)
```

`run_netpowerlab` *Launch the NetPowerLab Shiny Application*

Description

Opens an interactive application to plan the sample size of a psychological network study (Gaussian graphical model). The reference network is declared block by block, one block per instrument, and acts as the effect size of the design. The application then wraps the two simulation approaches: `bootnet::netSimulator()`, which reports how much of the true network is recovered at each sample size (sensitivity, specificity and the correlation between estimated and true edge weights), and `powerly::powerly()`, which recommends the sample size needed to reach a declared performance with a given probability. It also writes the reproducible R script and a draft of the Participants section.

Usage

```
run_netpowerlab(launch.browser = TRUE, port = NULL, ...)
```

Arguments

`launch.browser`	Logical. Open the application in the default browser (default TRUE).
`port`	Port to run the application on. `NULL` (default) lets Shiny pick a free one.
`...`	Further arguments passed to `shiny::runApp()`.

Details

The application needs `shiny`, `bslib` and `powerly`, which are suggested rather than required by InterconectaR: install them with `install.packages(c("shiny", "bslib", "powerly"))` the first time. Both simulations are computationally intensive; the app distributes them across cores (`nCores` in bootnet, `cores` in powerly), which turns minutes into seconds.

Value

Invisibly `NULL`. Called for its side effect: the running app.

References

Constantin, M. A., Schuurman, N. K., & Vermunt, J. K. (2026). A general Monte Carlo method for sample size analysis in the context of network models. *Psychological Methods, 31*(3), 385-405. doi:10.1037/met0000555

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

Examples

```
# The application only runs in an interactive session
if (interactive()) {
  # opens the app in the browser
  run_netpowerlab()

  # on a fixed port, without opening a browser window
  run_netpowerlab(launch.browser = FALSE, port = 7799)
}
```

---

sample_size_EGA_control

Default control parameters for `sample_size_EGA_montecarlo()`

Description

Returns a fully populated list of advanced defaults. Pass a partial list through the `control` argument of `sample_size_EGA_montecarlo()` to override only the entries you care about.

Usage

```
sample_size_EGA_control(control = list())
```

Arguments

`control`	Optional named list with overrides.

Details

Defaults are chosen from the EGA / network-psychometrics literature (Christensen et al., 2024; Golino et al., 2020). Each entry supplied in `control` replaces the default as a whole; for instance, `control = list(targets = list(ari = 0.80))` leaves ARI as the only target.

Value

A named list with all advanced parameters resolved.

Examples

```
# Defaults
str(sample_size_EGA_control(), max.level = 1)

# Use only community recovery as the decisional target
sample_size_EGA_control(list(targets = list(ari = 0.80)))$targets
```

sample_size_EGA_montecarlo *A Priori Sample Size Estimation for EGA Using Monte Carlo Simulation*

Description

Estimates an a priori sample size for psychometric network designs using Monte Carlo simulation. Data are generated from a population correlation matrix or from a hypothesized partial correlation network, then `EGAnet::EGA()` is fitted across candidate sample sizes.

When the input is a correlation matrix, the implied partial correlation network is computed and used as the ground truth for edge-recovery metrics (since `EGAnet::EGA()` returns a partial correlation network).

Optionally runs in parallel using `future.apply` with a per-replication progress bar via `progressr`.

Usage

```
sample_size_EGA_montecarlo(
  community_sizes = NULL,
  within_edge = 0.1,
  bridges = NULL,
  community_names = NULL,
  population_network = NULL,
  communities = NULL,
  sample_sizes = c(150, 250, 400, 600, 900, 1300),
  n_rep = 500,
  network_type = c("partial", "correlation"),
  data_type = c("ordinal", "continuous"),
  power_target = 0.8,
  validation_n_rep = 0,
  sensitivity = FALSE,
  plots = FALSE,
  output_dir = NULL,
  parallel = FALSE,
  n_workers = NULL,
  seed = NULL,
  verbose = TRUE,
  control = list(),
  ...
)
```

Arguments

`community_sizes`
: Integer vector with the number of items per community (e.g. `c(4, 4, 4, 4)` for a 4-community scale with 4 items each). When set, the function builds the population partial- correlation network and the community assignment internally;

population_network and communities can be left at NULL. This is the recommended simple API.

within_edge
: Partial-correlation weight assigned to every within- community edge when the network is built internally. Default 0.10 (a realistic, weak-loading scenario consistent with Christensen et al., 2024). Ignored if population_network is supplied.

bridges
: Optional data.frame (or matrix coercible to one) with columns i, j, weight describing inter-community bridge edges (1-based item indices). Default NULL = no bridges.

community_names
: Optional character vector of community labels used when building communities internally. Default NULL = c("F1", "F2", ...).

population_network
: Optional square matrix supplied directly (advanced API): either the population correlation matrix (network_type = "correlation") or the hypothesized partial correlation network (network_type = "partial"). When provided, communities must also be provided and factor_sizes is ignored.

communities
: Expected community assignment when using the advanced API. Either a named list of item indices per dimension or a vector with one community label per item.

sample_sizes
: Numeric vector with candidate sample sizes. Default c(150, 250, 400, 600, 900, 1300) brackets the typical EGA range reported in the literature (Christensen, Garrido, Guerra-Pena & Golino, 2024; Golino et al., 2020).

n_rep
: Number of Monte Carlo replications per sample size. Default 500 (500-1000 is the working range used in published EGA simulations).

network_type
: "partial" (default, recommended) or "correlation".

data_type
: "ordinal" (default, the standard in psychometric scales) or "continuous".

power_target
: Numeric in (0, 1). Target probability for the combined power curve. Default 0.80 (Cohen-style convention).

validation_n_rep
: Number of additional replications at the recommended n for empirical power validation. Set to 0 (default) to skip; 500 is a good value when reporting in a paper.

sensitivity
: Logical. If TRUE the function automatically runs a sensitivity sweep over the most influential arbitrary choices of the pipeline: true_edge_threshold (0.01 / 0.05 / 0.10), edge_threshold_method ("absolute" vs "match_true"), combination ("and" vs "primary") and, for ordinal data, a floor pattern of thresholds. The two edge-threshold sweeps only run when a target depends on the edge thresholds (sensitivity, specificity, precision, fdr, edge_correlation_true_edges or mae_true_edges). Each row is compared with the recommendation of the main run. Results are stored in $sensitivity. Default FALSE.

plots
: Logical. If TRUE, generates the standard plots (plot_power_curve_ci, plot_failure_decomposition, plot_sample_size_EGA_montecarlo and the bootstrap distribution of n*). Plots are stored in $plots and saved to disk if output_dir is set. Default FALSE.

output_dir
: Optional path to a directory. When set, the function creates the directory if needed and writes result.rds, summary.csv, sensitivity.csv (if applicable) and PNGs of all plots. Default NULL (no files written).

parallel	Logical; run replications in parallel via future.apply::future_lapply(). Default FALSE.
n_workers	Optional integer with the number of parallel workers when parallel = TRUE. Default min(8L, future::availableCores() - 1L). The cap at 8 is deliberate: on Windows each future::multisession worker spawns a fresh Rscript that re-loads EGAnet (~400 MB RAM and several seconds of startup each). Going wider routinely freezes the user-visible progress bar at "0-7%" while the workers initialise. Override explicitly (e.g. n_workers = 11 on a 12-core box) if you have ~5 GB of free RAM and want the extra throughput.
seed	Optional random seed. With parallel = TRUE the seed is forwarded to future.seed for reproducible parallel RNG.
verbose	Logical; print progress messages and show a progress bar.
control	Optional named list to override advanced defaults. See sample_size_EGA_control for the full list of tunable parameters and their literature-backed defaults (categories, corr, model, algorithm, uni.method, criteria, targets, recommendation_method, boots, require_convergence, require_correct_dimensions, combination, primary_metric, true_edge_threshold, estimated_edge_threshold, edge_threshold_method, thresholds).
...	Additional arguments passed to EGAnet::EGA().

Value

A list with class sample_size_EGA_montecarlo containing the recommended sample size, summary table (with median, mean, 95 percent Monte Carlo bands and per-metric power curves), raw replication results, criteria, targets, isotonic interpolation, bootstrap distribution and CI for the recommended n, validation results (when requested), the population correlation matrix, the true partial-correlation network used as ground truth, and the expected communities.

Bootstrap interpretation

The 95 percent CI returned in recommended_n_ci (and the band available via plot_power_curve_ci()) captures Monte Carlo uncertainty of the recommended-n estimator *conditional on the assumed population model and the chosen pipeline*. It is a within-model uncertainty band, not a population-level inferential statement. In particular, it does NOT account for: (1) misspecification of population_network (the wrong "truth"); (2) departures from the simulated data-generating process such as real ordinal scales with floor / ceiling effects, missingness, or non-normality; or (3) differences between the chosen EGA pipeline (corr, model, algorithm) and what users will actually run on their data.

For robust reporting, complement the CI with sensitivity analyses: vary the population network (denser or sparser), edge_threshold_method, combination, true_edge_threshold (e.g. 0.01 vs 0.05 vs 0.10, usually the most influential arbitrary choice in the pipeline when the targets include edge-recovery metrics), and thresholds (use make_likert_thresholds() to build floor / ceiling patterns). The validation list provides an empirical check that the recommended n actually delivers the target power on a fresh batch of replications.

Partial-to-Sigma convention

When network_type = "partial", the implied population correlation matrix Sigma is reconstructed from the partial-correlation network P by setting the precision matrix to K = I - P (unit residual vari-

ances on the diagonal, off-diagonal -P), inverting to obtain a covariance, and rescaling to a correlation. This is one valid parameterisation but *not* the only one: different residual-variance choices are compatible with the same partial structure and produce different Sigma, different simulated data, and ultimately different recommended_n values. The recommendation is therefore conditional on this generative convention. Report it explicitly in the manuscript and, if the partial network admits a natural alternative scaling, run a sensitivity analysis under that alternative.

Edge-recovery metrics under network_type = "correlation"

When network_type = "correlation" the implied true partial network is computed by inverting Sigma and is typically dense. Because model = "glasso" produces sparse estimates, binary edge-recovery metrics (sensitivity, specificity, precision, fdr) become *structurally biased* regardless of sample size: the design itself imposes a sparsity mismatch. In this regime, prefer targets based on ari (community recovery) or edge_correlation (continuous agreement) and treat binary edge targets as untrustworthy. The function emits a warning if you combine network_type = "correlation" with binary edge metrics in targets.

Examples

```
# A toy run: two communities of four items, three candidate sample sizes
# and 10 replications each. Real studies need n_rep = 500 or more.
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4),
  within_edge = 0.20,
  sample_sizes = c(100, 200, 400),
  n_rep = 10,
  data_type = "continuous",
  seed = 1,
  verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
res
res$summary[, c("Sample_Size", "power_combined")]
```

---

show_plots *Visualize all plots stored in a* sample_size_EGA_montecarlo *object*

Description

Prints each plot in x$plots sequentially. Use this after calling sample_size_EGA_montecarlo(..., plots = TRUE) to inspect the simulation graphically without having to write each print().

Usage

```
show_plots(x, which = NULL)
```

Arguments

x	A sample_size_EGA_montecarlo object that was produced with plots = TRUE, or its $plots list.
which	Optional character vector with the names of the plots to show (subset of names(x$plots)). Default NULL = all.

Value

Invisibly returns the list of plots.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE, plots = TRUE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
show_plots(res, which = "power_curve_ci")
```

---

show_recommendation *One-line dynamic recommendation message*

Description

Reports, as a message, a single sentence with the recommended sample size and the structural assumptions that produced it (items, communities, sizes, data type, power target, bootstrap CI when available). Use this when you want the answer fast and don't need the full print() report.

Usage

```
show_recommendation(x)
```

Arguments

x	A sample_size_EGA_montecarlo object.

Value

Invisibly returns the recommended n.

Examples

```
res <- sample_size_EGA_montecarlo(
  community_sizes = c(4, 4), within_edge = 0.20,
  sample_sizes = c(100, 200, 400), n_rep = 10,
  data_type = "continuous", seed = 1, verbose = FALSE,
  control = list(boots = 100, corr = "pearson", algorithm = "walktrap")
)
show_recommendation(res)
```

structure_groups *Structure Groups*

Description

Creates a named list of sequential integer groups.

Usage

```
structure_groups(names, values)
```

Arguments

names	Character vector of group names.
values	Integer vector of group sizes.

Value

A named list of integer sequences.

Examples

```
# Two scales: the first four items and the next five
structure_groups(c("Anxiety", "Depression"), c(4, 5))
```

summarise_case_drop_stability *Summarise Case Drop Stability*

Description

Summarises case-dropping bootstrap stability results from bootnet.

Usage

```
summarise_case_drop_stability(
  caseDroppingBoot,
  probs = c(0.025, 0.975),
  correlation_use = "pairwise.complete.obs",
  quiet = TRUE
)
```

Arguments

`caseDroppingBoot`
: A bootnet case-dropping bootstrap object.

`probs`
: Numeric vector of length 2 for confidence interval quantiles.

`correlation_use`
: Character string for cor() use argument.

`quiet`
: Logical, suppress printing.

Value

A data frame with stability summary statistics.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
case_boot <- bootnet::bootnet(net, nBoots = 20, type = "case",
                              statistics = c("strength", "expectedInfluence"),
                              nCores = 1, verbose = FALSE)
summarise_case_drop_stability(case_boot)
```

---

summarise_nonparametric_edges

Summarise Nonparametric Edges

Description

Summarises nonparametric bootstrap results for network edges from bootnet.

Usage

```
summarise_nonparametric_edges(
  nonParametricBoot,
  stat_type = "edge",
  probs = c(0.025, 0.975),
  quiet = TRUE
)
```

Arguments

nonParametricBoot	A bootnet nonparametric bootstrap object.
stat_type	Character, type of statistic to summarise (default "edge").
probs	Numeric vector of length 2 for confidence interval quantiles.
quiet	Logical, suppress printing.

Value

A data frame with edge bootstrap summary statistics.

Examples

```
set.seed(123)
n <- 300
f1 <- rnorm(n)
f2 <- 0.3 * f1 + rnorm(n)
items <- data.frame(
  sapply(1:4, function(i) 0.7 * f1 + rnorm(n, 0, 0.7)),
  sapply(1:4, function(i) 0.7 * f2 + rnorm(n, 0, 0.7))
)
names(items) <- c(paste0("A", 1:4), paste0("B", 1:4))

net <- bootnet::estimateNetwork(items, default = "EBICglasso")
edge_boot <- bootnet::bootnet(net, nBoots = 20, nCores = 1, verbose = FALSE)
summarise_nonparametric_edges(edge_boot)
```

---

summary_metrics *Summarize Network Performance Metrics*

Description

Calculates median values of network performance metrics grouped by algorithm, correlation method, and sample size.

Usage

```
summary_metrics(results)
```

Arguments

results	Data frame from boot_and_evaluate with raw metrics.

Value

A data frame with averaged metrics.

Examples

```
# A small table with the structure returned by boot_and_evaluate()
set.seed(1)
res <- expand.grid(Algorithm = c("louvain", "walktrap"),
                   Correlation_Method = c("pearson", "spearman"),
                   Sample_Size = c(100, 250, 500), Simulation = 1:5,
                   stringsAsFactors = FALSE)
k <- nrow(res)
res$sensitivity <- runif(k, 0.6, 0.9)
res$specificity <- runif(k, 0.7, 0.95)
res$precision <- runif(k, 0.6, 0.9)
res$correlation <- runif(k, 0.7, 0.95)
res$abs_cor <- runif(k, 0.7, 0.95)
res$bias <- runif(k, 0.01, 0.05)
res$TEFI <- rnorm(k, -5, 0.3)
res$FDR <- runif(k, 0.05, 0.2)

summary_metrics(res)
```

Index

boot_and_evaluate, 3

calculate_centrality_sd_correlation, 4
centrality_bridge_plot, 5
centrality_bullet_plot, 6
centrality_cleveland_plot, 8
centrality_duo_plot, 10
centrality_plots2, 11
choose_best_method, 12
combine_graphs_centrality, 13
combine_graphs_centrality2, 14
combine_groupBy, 15
compute_netScores, 17
convert_EGA_list_to_df, 18
convert_EGA_to_df, 19, *19*

Density_report, 20

estimate_networks_by_group, 21

filter_correlation_stability, 22

get_edge_weights_summary, 23

make_likert_thresholds, 24
mgm_error_metrics, 25
mgm_errors_groups, 26

net_reduce2, 27

optimize_leiden_resolution, 28

plot_avg_TEFI, 29
plot_centrality_by_group, 29
plot_centrality_stability, 30
plot_failure_decomposition, 32
plot_latent_network_diagram, 33
plot_lvm, 34
plot_net, 35
plot_net_group, 38
plot_networks_by_group, 41
plot_performance_metrics, 43
plot_power_curve_ci, 44
plot_sample_size_EGA_montecarlo, 45
print.sample_size_EGA_montecarlo, 46
print.scale_network_analysis, 47
process_LCT, 48

qgraph_centrality_panel, 48

refine_items_by_stability, 50
refine_items_by_stability(), *18*
run_EGA_combinations, *19*, 52
run_netpowerlab, 53

sample_size_EGA_control, 54, *57*
sample_size_EGA_montecarlo, 55
shiny::runApp(), *53*
show_plots, 58
show_recommendation, 59
structure_groups, 60
summarise_case_drop_stability, 60
summarise_nonparametric_edges, 61
summary_metrics, 62