Package: idiographic (via r-universe)

October 22, 2026

Type Package

Title Person-Specific (Idiographic) and Heterogeneous Complex Networks

Version 0.3.4

Author Mohammed Saqr [aut, cre, cph], Sonsoles López-Pernas [aut]

Maintainer Mohammed Saqr <saqr@saqr.me>

Description Person-specific and within-person network estimation from intensive longitudinal and panel data. Estimators include ordinary vector autoregression (VAR), graphical vector autoregression (graphical VAR), multilevel vector autoregression (mlVAR), rolling ordinary and graphical VAR, native Bayesian VAR and multilevel Bayesian VAR, unified Structural Equation Modeling (uSEM), and Group Iterative Multiple Model Estimation (GIMME). All estimators are native clean-room implementations. All functions are validated against authoritative literature. Also provides preprocessing audits, edge-stability diagnostics, model-comparison reports, and rolling forecast validation. Methods are described in <doi:10.1007/978-3-031-95365-1_20> and <doi:10.1080/00273171.2018.1454823>.

License GPL-3

Encoding UTF-8

Language en-GB

LazyData TRUE

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

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

Depends R (>= 4.1)

Roxygen list(markdown = TRUE)

Imports stats, utils, parallel

Suggests testthat (>= 3.0.0), lme4, lavaan, cograph, mlVAR, MplusAutomation, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

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

Date/Publication 2026-08-23 16:48:48 UTC

RemoteUrl https://github.com/mohsaqr/idiographic

RemoteRef HEAD

RemoteSha 0c842c6225d85263f23b6a8dd2383f273eeeb779

Contents

idiographic-package . . . 4
argument_coverage . . . 4
as.data.frame.glasso_path_result . . . 5
as.data.frame.glasso_result . . . 6
as_netobject . . . 6
as_netobject.gvar_result . . . 7
as_netobject.net_gimme . . . 7
as_netobject.net_mlvar . . . 8
as_netobject.var_bayes_result . . . 9
coefs.idioml_result . . . 9
compare_idiographic . . . 11
edges.net_usem . . . 12
equivalence . . . 14
equivalence_table . . . 15
esm_srl . . . 16
estimate_stability . . . 17
estimator_info . . . 18
extract_edges . . . 19
fit_gimme . . . 19
fit_graphical_var . . . 23
fit_graphical_var_each . . . 27
fit_idiographic . . . 28
fit_ml . . . 29
fit_mlvar . . . 32
fit_mlvar_bayes . . . 36
fit_mlvar_mplus . . . 39
fit_rolling_graphical_var . . . 41
fit_rolling_var . . . 42
fit_usem . . . 44
fit_var . . . 46
fit_var_bayes . . . 47
fit_var_each . . . 49
get_estimator . . . 50

glasso_fit . . . 50
glasso_kkt . . . 52
glasso_path . . . 53
list_estimators . . . 55
matrices . . . 55
nodes.net_usem . . . 57
plot_gimme . . . 58
plot_idiographic . . . 60
predict.idioml_result . . . 61
preprocess . . . 62
print.forecast_result . . . 65
print.glasso_path_result . . . 65
print.glasso_result . . . 66
print.gvar_list . . . 66
print.gvar_result . . . 67
print.idioml_result . . . 67
print.model_comparison . . . 68
print.net_gimme . . . 68
print.net_mlvar . . . 69
print.net_mlvar_bayes . . . 70
print.net_usem . . . 70
print.preprocess_result . . . 71
print.rolling_gvar_result . . . 71
print.rolling_var_result . . . 72
print.stability_result . . . 72
print.var_bayes_result . . . 73
print.var_list . . . 73
print.var_result . . . 74
register_estimator . . . 74
remove_estimator . . . 75
srl . . . 76
summary.gvar_result . . . 77
summary.idioml_result . . . 78
summary.net_gimme . . . 78
summary.net_mlvar . . . 79
summary.net_usem . . . 80
summary.preprocess_result . . . 80
summary.var_bayes_result . . . 81
summary.var_result . . . 81
validate_forecast . . . 82

Index **84**

idiographic-package *idiographic: Idiographic Network Estimation from Intensive Longitudinal Data*

Description

Person-specific and within-person network estimation from intensive longitudinal / ESM panel data: preprocessing audits (`preprocess()`), edge-stability diagnostics (`estimate_stability()`), model-comparison reports (`compare_idiographic()`), rolling forecast validation (`validate_forecast()`), rolling ordinary vector autoregression (`fit_rolling_var()`), rolling graphical vector autoregression (`fit_rolling_graphical_var()`), ordinary vector autoregression (`fit_var()`), graphical vector autoregression (`fit_graphical_var()`), multilevel vector autoregression (`fit_mlvar()`), unified SEM (`fit_usem()`), Group Iterative Multiple Model Estimation (`fit_gimme()`), and idiographic supervised machine-learning models (`fit_ml()`) for individualized prediction. Use `fit_idiographic()` for registry-driven dispatch or the direct `fit_*()` functions. Every result has tidy `as.data.frame()`, `summary()`, and print methods; network estimators also share `edges()`, `coefs()`, `nodes()`, `matrices()`, and plotting methods. `equivalence()` reports the exact validation scope attached to each method.

Author(s)

Maintainer: Mohammed Saqr <saqr@saqr.me> [copyright holder]

Authors:

- Sonsoles López-Pernas <sonsoles.lopez@uef.fi>

See Also

Useful links:

- https://pak.dynasite.org/idiographic/
- https://github.com/mohsaqr/idiographic
- Report bugs at https://github.com/mohsaqr/idiographic/issues

argument_coverage *Argument-by-argument validation coverage*

Description

Builds a tidy, executable ledger from the actual formals of every registered entry point. Each argument is classified according to the strongest honest evidence contract available for that method: direct oracle/engine equality, frozen statistical fixtures, recovery/internal validation, delegated forwarding, a supported extension, or an explicit rejection boundary. Consequently a newly added formal cannot silently disappear from the audit: package tests require every current formal to occur exactly once here.

Usage

```
argument_coverage(method = NULL)
```

Arguments

`method`	Optional registered method name or alias. `NULL` returns all built-in and currently registered methods.

Value

A data frame with one row per public method argument.

Examples

```
argument_coverage()
argument_coverage("mlvar")
```

`as.data.frame.glasso_path_result` *Tidy a graphical-lasso path*

Description

Tidy a graphical-lasso path

Usage

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

Arguments

`x`	A `glasso_path_result` from `glasso_path()`.
`row.names`, `optional`	Ignored; present for S3 consistency.
`...`	Ignored.

Value

A `data.frame` with one row per variable pair per penalty and columns `rho`, `from`, `to`, `precision` and `weight`.

as.data.frame.glasso_result *Tidy a graphical-lasso fit*

Description

Tidy a graphical-lasso fit

Usage

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

Arguments

x	A glasso_result from glasso_fit().
row.names, optional	Ignored; present for S3 consistency.
...	Ignored.

Value

A data.frame with one row per unique variable pair and columns from, to, precision (the Θ entry) and weight (the partial correlation implied by Θ).

as_netobject *Coerce to a netobject*

Description

Returns netobjects unchanged; promotes a bare cograph_network.

Usage

```
as_netobject(x, ...)
```

Arguments

x	A netobject or cograph_network.
...	Passed to methods.

Value

A c("netobject", "cograph_network") object.

Examples

```
W <- matrix(c(0, 0.3, -0.2, 0), 2, 2,
            dimnames = list(c("A", "B"), c("A", "B")))
x <- structure(list(weights = W, method = "relative", directed = TRUE),
               class = "cograph_network")
as_netobject(x)
```

`as_netobject.gvar_result`

Coerce a gvar_result to plottable netobjects

Description

Returns the temporal lag layer(s) and contemporaneous network as netobjects, so each renders directly with `cograph::splot()` (or any netobject verb) without the caller transposing matrices or dropping intercept columns. The temporal network is oriented `[from = predictor(t-1), to = outcome(t)]`.

Usage

```
## S3 method for class 'gvar_result'
as_netobject(x, ...)
```

Arguments

`x`	A `gvar_result`.
`...`	Ignored.

Value

A `netobject_group`: a named list with `$temporal` for a lag-1 model, or one `temporal_lagN` element per multi-lag model, plus `$contemporaneous`.

`as_netobject.net_gimme`

Plottable netobject(s) from a GIMME fit

Description

Returns the GIMME result as matrix-backed netobjects. By default these encode the **same quantity the `gimme` package plots** — the *proportion of subjects* that have each path (`path_counts / n_subjects`) — not the group-average coefficient (which dilutes toward zero and is not what GIMME displays). For the faithful single mixed network (dashed lag / solid contemporaneous, group/individual colouring, autoregressive self-loops) use `plot_gimme()`.

Usage

```
## S3 method for class 'net_gimme'
as_netobject(x, style = c("pnode", "unified"), weight = c("prop", "coef"), ...)
```

Arguments

x	A net_gimme object.
style	Either "pnode" (default) — a netobject_group of two directed p-node networks, $temporal (lagged; autoregression on the diagonal) and $contemporaneous (same-beep), matching the shape fit_graphical_var() returns — or "unified", a single directed 2p-node network with the *_lag half feeding the current half (the literal uSEM topology).
weight	Either "prop" (default) — edge weight is the proportion of subjects with the path — or "coef", the group-average standardized coefficient.
...	Unused.

Value

For style = "pnode", a netobject_group with $temporal and $contemporaneous. For style = "unified", one c("netobject", "cograph_network") object with 2p nodes.

See Also

plot_gimme() for the faithful gimme-style mixed plot.

---

as_netobject.net_mlvar

Plottable netobjects from an mlVAR fit

Description

Returns the three networks as netobjects oriented for plotting (temporal edges run predictor -> outcome, matching fit_graphical_var), so cograph::splot() renders them consistently. The raw fit$temporal$weights keep mlVAR's [outcome, predictor] layout for equivalence.

Usage

```
## S3 method for class 'net_mlvar'
as_netobject(x, ...)
```

Arguments

x	A net_mlvar object.
...	Unused.

Value

A netobject_group with $temporal, $contemporaneous, $between.

as_netobject.var_bayes_result *Coerce a var_bayes_result to plottable netobjects*

Description

Coerce a var_bayes_result to plottable netobjects

Usage

```
## S3 method for class 'var_bayes_result'
as_netobject(x, ...)
```

Arguments

x	A var_bayes_result.
...	Ignored.

Value

A netobject_group with temporal (directed) and contemporaneous (undirected) netobjects.

coefs.idioml_result *Tidy coefficients from a fitted mlvar model*

Description

Generic accessor for the tidy coefficient table stored on a fit_mlvar() result. Returns a data.frame with one row per (outcome, predictor) pair and columns outcome, predictor, beta, se, t, p, ci_lower, ci_upper, significant.

Usage

```
## S3 method for class 'idioml_result'
coefs(x, ...)

coefs(x, ...)

## S3 method for class 'net_mlvar'
coefs(x, ...)

## Default S3 method:
coefs(x, ...)

## S3 method for class 'net_mlvar_bayes'
```

```
coefs(x, ...)

## S3 method for class 'net_usem'
coefs(x, ...)

## S3 method for class 'var_result'
coefs(x, ...)

## S3 method for class 'var_bayes_result'
coefs(x, ...)

## S3 method for class 'gvar_result'
coefs(x, ...)

## S3 method for class 'net_gimme'
coefs(x, ...)

## S3 method for class 'var_list'
coefs(x, ...)

## S3 method for class 'gvar_list'
coefs(x, ...)
```

Arguments

x	A fitted model object — currently only net_mlvar is supported.
...	Unused.

Details

Only the within-person (temporal) coefficients are tabulated — these are the lagged fixed effects that populate fit$temporal. The between-subjects effects that go into fit$between are handled via the D (I - Gamma) transformation and are not exposed as a separate tidy table.

Value

A tidy data.frame of coefficient estimates.

Examples

```
set.seed(1)
n_id <- 8; n_t <- 30; vars <- c("A", "B", "C")
rows <- lapply(seq_len(n_id), function(i) {
  m <- as.data.frame(matrix(rnorm(n_t * 3), ncol = 3))
  names(m) <- vars
  m$id <- i; m$day <- 1L; m$beep <- seq_len(n_t)
  m
})
d <- do.call(rbind, rows)
```

```
fit <- fit_mlvar(d, vars = vars, id = "id", day = "day", beep = "beep")
print(fit)
summary(fit)
```

`compare_idiographic` *Compare idiographic estimators on one dataset*

Description

Fits one or more idiographic estimators to the same data and returns a tidy per-method/per-network comparison table. This is a reporting layer: it does not define a new model, and each row is computed from the estimator's own `summary()` method plus common edge-table accessors.

Usage

```
compare_idiographic(
  data,
  vars,
  estimators = c("var", "graphical_var"),
  id = NULL,
  day = NULL,
  beep = NULL,
  estimator_args = list(),
  keep_fits = FALSE
)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`estimators`	Character vector naming registered network estimators to fit. Built-in values are `"var"`, `"var_bayes"`, `"graphical_var"`, `"mlvar"`, `"mlvar_bayes"`, `"mlvar_mplus"`, `"usem"`, and `"gimme"`.
`id`	Character. Name of the person-ID column, or `NULL`.
`day`	Character. Name of the day/session column, or `NULL`.
`beep`	Character. Name of the measurement-occasion column, or `NULL`.
`estimator_args`	Named list of per-estimator argument lists, e.g. `list(graphical_var = list(n_lambda = 8), usem = list(temporal = "ar"))`.
`keep_fits`	Logical. Store fitted model objects? Default `FALSE`.

Value

A `model_comparison` object with `$comparison`, `$failures`, and optionally `$fits`. `$comparison` is a tidy `data.frame` with one row per method/network.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = rep(1:4, each = 15),
                beep = rep(1:15, 4),
                A = rnorm(60), B = rnorm(60), C = rnorm(60))
cmp <- compare_idiographic(
  d, vars = c("A", "B", "C"), id = "id", day = "day", beep = "beep",
  estimators = c("var", "graphical_var"),
  estimator_args = list(graphical_var = list(n_lambda = 5))
)
cmp$comparison
```

---

edges.net_usem *Tidy edge table for any idiographic result*

Description

A single tidy verb for every network idiographic produces. Returns one row per edge with columns `network` (e.g. `"temporal"`, `"contemporaneous"`, `"between"`), `from`, `to`, `weight` – and, for GIMME, `level` (`"group"`/`"individual"`). Directed networks (temporal) keep every edge; undirected networks (contemporaneous, between) report each pair once.

Usage

```
## S3 method for class 'net_usem'
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

## S3 method for class 'var_result'
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

edges(x, ...)

## S3 method for class 'netobject'
```

```
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

## S3 method for class 'netobject_group'
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

## S3 method for class 'gvar_result'
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

## S3 method for class 'net_mlvar'
edges(
  x,
  sort_by = "weight",
  include_self = FALSE,
  network = NULL,
  n = NULL,
  ...
)

## S3 method for class 'net_gimme'
edges(
  x,
  sort_by = "weight",
  include_self = TRUE,
  weight = c("prop", "coef"),
  network = NULL,
  n = NULL,
  ...
```

```
)

## S3 method for class 'var_list'
edges(x, ...)

## S3 method for class 'gvar_list'
edges(x, ...)
```

Arguments

x	A gvar_result, net_mlvar, net_gimme, netobject, or netobject_group.
sort_by	"weight" (descending |weight|) or NULL for natural order.
include_self	Keep autoregressive self-loops? Default FALSE (TRUE for GIMME, where the autoregression is the point).
network	Optional character vector selecting the network layer(s) to return (e.g. "temporal", "contemporaneous", "between"). Default NULL returns every layer. An unknown layer errors and lists the available ones.
n	Optional integer. Keep only the first n edges, which are the strongest by absolute weight when sort_by = "weight". Default NULL returns all edges.
...	Passed to methods.
weight	For GIMME only: "prop" (proportion of subjects, default) or "coef" (group-average coefficient) for the edge weight.

Value

A tidy data.frame, one row per edge.

Examples

```
set.seed(1)
d <- data.frame(id = 1, A = rnorm(80), B = rnorm(80), C = rnorm(80))
fit <- fit_graphical_var(d, vars = c("A", "B", "C"), id = "id", n_lambda = 8)
edges(fit)            # tidy: network / from / to / weight
```

---

equivalence *Report method-equivalence evidence*

Description

Returns the equivalence declaration attached by fit_idiographic(). For a result created by a direct fit_*() call, the registry can infer the method from a unique registered result class. The declaration describes the scope of committed validation; it is not a new statistical equivalence test.

Usage

```
equivalence(x)
```

Arguments

x	A fitted object.

Value

An idiographic_equivalence list with method, status, reference, scope, tolerance, notes, and source.

Examples

```
set.seed(2)
d <- data.frame(A = rnorm(40), B = rnorm(40))
fit <- fit_var(d, vars = c("A", "B"), scale = FALSE)
equivalence(fit)
```

---

equivalence_table *Package-wide equivalence and validation ledger*

Description

Returns one tidy row for every registered estimator and workflow. Unlike equivalence(), which refines the declaration for one fitted object, equivalence_table() exposes the package-wide evidence boundary before a model is fitted. A closed evidence status means the declared scope has an executable oracle, engine, recovery, or internal-consistency contract; it does not turn native extensions into claims about an unrelated package.

Usage

```
equivalence_table(method = NULL)
```

Arguments

method	Optional registered method name or alias. NULL returns all built-in and currently registered methods.

Value

A data frame with method, kind, declared status, evidence status, reference, numerical tolerance bounds, scope, and notes.

Examples

```
equivalence_table()
equivalence_table("gimme")
```

`esm_srl` *Momentary self-regulated-learning experience-sampling data*

Description

An anonymized intensive longitudinal data set in which 41 students rated their momentary self-regulation, motivation, and anxiety several times per day over the course of a study, giving roughly 70 to 80 occasions each. Unlike the once-per-day srl panel, the occasions here are within-day momentary assessments, so the series are well suited to person-specific (idiographic) VAR, graphical VAR, and unified SEM. The data are fully anonymized: the participant identifiers are fictional names, and the calendar dates have been shifted by a constant offset (preserving all within-person spacing) so that no real dates or identities remain.

Usage

```
esm_srl
```

Format

A `data.frame` with 2820 rows and 12 columns:

name Fictional participant identifier (41 unique students).

occasion Within-person occasion index, ordered in time.

date Anonymized (constant-shifted) assessment date.

efficacy Momentary self-efficacy (motivation).

value Momentary task value (motivation).

planning Momentary planning (self-regulation).

monitoring Momentary monitoring (self-regulation).

effort Momentary effort regulation (self-regulation).

regulation Momentary strategy regulation (self-regulation).

motivated Momentary felt motivation (motivation).

enjoyment Momentary enjoyment (motivation).

anxiety Momentary anxiety.

Details

The nine indicators span three domains: self-regulation (`planning`, `monitoring`, `effort`, `regulation`), motivation (`efficacy`, `value`, `motivated`, `enjoyment`), and `anxiety`. Each variable is on a 0-100 scale. Rows are one person-occasion each, ordered within person by `occasion`.

Examples

```
data(esm_srl)
summary(esm_srl)
head(esm_srl)
```

`estimate_stability` *Estimate edge stability by block resampling (experimental)*

Description

Experimental. The resampling design is methodologically grounded (block bootstrap for dependent data; edge-stability summaries in the spirit of bootnet), but unlike the estimators in this package it has no external reference implementation to validate against, and its interface, defaults, and reported statistics may change in a future release.

Refit an idiographic estimator across deterministic block resamples and summarize edge-level stability. Blocks preserve within-block time order: subject-day blocks when `id` and `day` are supplied, subjects when only `id` is supplied, days when only `day` is supplied, or consecutive row blocks for a single series. Duplicate blocks receive temporary ids/day labels before fitting so lag construction never connects two sampled copies.

Usage

```
estimate_stability(
  data,
  vars,
  estimator = c("var", "graphical_var", "mlvar", "usem", "gimme"),
  id = NULL,
  day = NULL,
  beep = NULL,
  n_resamples = 100L,
  resample = c("block", "split_half"),
  block_size = NULL,
  threshold = 1e-08,
  seed = NULL,
  keep_fits = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`estimator`	`"var"` (default) for `fit_var()`, `"graphical_var"` for `fit_graphical_var()`, `"mlvar"` for `fit_mlvar()`, `"usem"` for `fit_usem()`, or `"gimme"` for `fit_gimme()`.
`id`	Character. Name of the person-ID column, or `NULL`.
`day`	Character. Name of the day/session column, or `NULL`.
`beep`	Character. Name of the measurement-occasion column, or `NULL`.
`n_resamples`	Integer number of bootstrap/split resamples.

resample	"block" samples blocks with replacement; "split_half" samples half the blocks without replacement on each replicate.
block_size	Integer or NULL. Consecutive block length used only when neither id nor day is supplied. Defaults to floor(sqrt(nrow(data))).
threshold	Numeric. Absolute weight above which an edge is counted as selected. Default 1e-8.
seed	Optional integer seed for deterministic resampling.
keep_fits	Logical. Store successful resampled fits in the returned object? Default FALSE.
...	Further arguments passed to the estimator.

Value

A stability_result with $stability edge statistics, $original fit, $resample_edges, $failures, and $config.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = rep(1:4, each = 12),
                beep = rep(1:12, 4),
                A = rnorm(48), B = rnorm(48), C = rnorm(48))
st <- estimate_stability(d, vars = c("A", "B", "C"), id = "id",
                         day = "day", beep = "beep",
                         n_resamples = 5, seed = 1)
head(st$stability)
```

---

estimator_info *Inspect a registered estimator*

Description

Inspect a registered estimator

Usage

```
estimator_info(method)
```

Arguments

method	A registered method name or alias. Names are case-insensitive; spaces, hyphens, and periods are normalized to underscores.

Value

estimator_info() returns the complete registration as a list.

Examples

```
estimator_info("var")
```

extract_edges *Tidy edge table from a network object*

Description

Returns a one-row-per-edge `data.frame` with node labels, for any netobject / cograph_network (or a `gvar_result` constituent).

Usage

```
extract_edges(model, sort_by = "weight", include_self = FALSE)
```

Arguments

`model`	A `netobject` or `cograph_network`. Multi-network results (a `gvar_result`, `net_mlvar`, or any `netobject_group`) hold more than one network, so pass a single constituent — e.g. `extract_edges(as_netobject(x)$temporal)`.
`sort_by`	Either `"weight"` (descending by absolute weight) or `NULL`.
`include_self`	Keep autoregressive self-loops? Default `FALSE`.

Value

A `data.frame` with columns `from`, `to`, `weight`.

Examples

```
W <- matrix(c(0, 0.3, -0.2, 0), 2, 2,
            dimnames = list(c("A", "B"), c("A", "B")))
x <- structure(list(weights = W, method = "relative", directed = TRUE),
               class = "cograph_network")
extract_edges(x)
```

fit_gimme *GIMME: Group Iterative Multiple Model Estimation*

Description

Estimates person-specific directed networks from intensive longitudinal data using the unified Structural Equation Modeling (uSEM) framework. Implements a data-driven search that identifies:

1. **Group-level paths**: Directed edges present for a majority (default 75\
2. **Individual-level paths**: Additional edges specific to each person, found after group paths are established.

Uses `lavaan` for SEM estimation and modification indices. Accepts a single data frame with an ID column (not CSV directories).

Usage

```
fit_gimme(
  data,
  vars,
  id,
  time = NULL,
  day = NULL,
  beep = NULL,
  min_obs = NULL,
  subject = NULL,
  ar = TRUE,
  standardize = FALSE,
  groupcutoff = 0.75,
  subcutoff = 0.75,
  paths = NULL,
  exogenous = NULL,
  hybrid = FALSE,
  VAR = FALSE,
  rmsea_cutoff = 0.05,
  srmr_cutoff = 0.05,
  nnfi_cutoff = 0.95,
  cfi_cutoff = 0.95,
  n_excellent = 2L,
  seed = NULL,
  group_correct = "Bonferroni Group",
  indiv_correct = "Bonferroni",
  alpha = 0.05,
  stop_crit = "model fit",
  subgroup = FALSE,
  outcome = NULL,
  conv_vars = NULL,
  mult_vars = NULL,
  lv_model = NULL,
  lasso_model_crit = NULL,
  ms_allow = FALSE,
  ordered = NULL,
  dir_prop_cutoff = 0,
  out = NULL,
  sep = NULL,
  header = NULL,
  plot = FALSE,
  sub_feature = "lag & contemp",
  sub_method = "Walktrap",
  sub_sim_thresh = "lowest",
  confirm_subgroup = NULL,
  conv_length = 16,
  conv_interval = 1,
  mean_center_mult = FALSE,
```

```
  diagnos = FALSE,
  ms_tol = 1e-05,
  lv_estimator = "miiv",
  lv_scores = "regression",
  lv_miiv_scaling = "first.indicator",
  lv_final_estimator = "miiv"
)
```

Arguments

data	A data.frame in long format with columns for person ID, time-varying variables, and optionally a time/beep column.
vars	Character vector of variable names to model.
id	Character string naming the person-ID column.
time	Character string naming the time/order column, or NULL. When provided, data is sorted by id then time before lagging.
day	Character string naming the day/session column, or NULL. When supplied, lag-1 pairs are formed only within the same (id, day) block, so a lag never crosses the overnight gap.
beep	Character string naming the measurement-occasion column, or NULL. Used (with day) to order observations when time is not given.
min_obs	Integer or NULL. Keep only subjects with at least this many observations (counts taken from data).
subject	Optional vector naming the exact subject(s) to analyse.
ar	Logical. If TRUE (default), autoregressive paths (each variable predicting itself at lag 1) are included as fixed paths.
standardize	Logical. If TRUE (default FALSE), variables are standardized per person before estimation. Note: the returned coefficient network ($coefs, $psi, $temporal_avg, $contemporaneous_avg, $group_paths) is unaffected because idiographic extracts the standardized lavaan solution (lavInspect(fit, "std")), which is invariant to input scaling. Only the scale-dependent $fit statistics (chisq, aic, bic) change.
groupcutoff	Numeric between 0 and 1. Proportion of individuals for whom a path must be significant to be added at group level. Default 0.75.
subcutoff	Numeric. Subgroup cutoff (default 0.75, matching gimme); only relevant to subgrouping, which is not implemented.
paths	Character vector of lavaan-syntax paths to force into the model (e.g., "V2~V1lag"). Default NULL.
exogenous	Character vector of variable names to treat as exogenous. Default NULL.
hybrid	Logical. If TRUE, also searches residual covariances. Default FALSE.
VAR	Logical. If TRUE, fit a standard VAR: only lagged directed paths are searched and contemporaneous relations are estimated as residual covariances (no directed contemporaneous paths). Matches gimme(VAR = TRUE). Default FALSE.
rmsea_cutoff	Numeric. RMSEA threshold for excellent fit (default 0.05).

`srmr_cutoff` Numeric. SRMR threshold for excellent fit (default 0.05).

`nnfi_cutoff` Numeric. NNFI/TLI threshold for excellent fit (default 0.95).

`cfi_cutoff` Numeric. CFI threshold for excellent fit (default 0.95).

`n_excellent` Integer. Number of fit indices that must be excellent to stop individual search. Default 2.

`seed` Integer or `NULL`. Random seed for reproducibility.

`group_correct` Group-level multiple-comparison correction. Use `"Bonferroni Group"` (the default) to divide `alpha` by the number of people, `"Bonferroni Paths"` to divide it by the number of eligible paths, `"fdr"` for Benjamini-Hochberg correction, or a single number in `(0, 1)` to set the group alpha directly. The legacy misspelling `"Bonferoni Group"` is accepted with a deprecation warning.

`indiv_correct` Individual-level multiple-comparison correction. Use `"Bonferroni"` (the default) or `"fdr"`.

`alpha` Base significance level for group and individual searches. Default `0.05`.

`stop_crit` Individual-search stopping rule. `"standard"` stops when fit is adequate or no significant path remains; `"model fit"` (the default) keeps adding the largest-MI path, regardless of significance, until fit is adequate; and `"significance"` keeps adding significant paths even after fit is adequate.

`subgroup` Logical. Subgrouping (S-GIMME) is not implemented; `TRUE` raises an error pointing to `gimme::gimme()`. Default `FALSE`.

`outcome`, `conv_vars`, `mult_vars`, `lv_model`, `lasso_model_crit`, `ms_allow`, `ordered`, `dir_prop_cutoff`
Accepted for `gimme::gimme()` API parity but not implemented (latent variable / fMRI-convolution / multiplied-term / LASSO / ordinal / multiple-solutions / directionality features). A non-default value raises an error pointing to `gimme::gimme()`.

`out`, `sep`, `header`, `plot`
Accepted for `gimme::gimme()` API parity. idiographic reads a `data.frame` (not a CSV directory), so non-default `out`, `sep`, and `header` values emit a warning and have no effect. It returns an object you plot with `plot_gimme()`; `plot = TRUE` emits a message.

`sub_feature`, `sub_method`, `sub_sim_thresh`, `confirm_subgroup`, `conv_length`, `conv_interval`, `mean_center_mult`, `diagnos`, `ms_tol`, `lv_estimator`, `lv_scores`, `lv_miiv_scaling`, `lv_final_estimator`
Accepted for `gimme::gimme()` API parity. These configure the unsupported subgrouping / convolution / multiplied-term / multiple-solutions / latent-variable features and are inert here (their parent feature is guarded above).

Value

An S3 object of class `"net_gimme"` containing:

`temporal` p x p matrix of group-level temporal (lagged) path counts – entry `[i,j]` = number of individuals with path j(t-1)->i(t).

`contemporaneous` p x p matrix of group-level contemporaneous path counts – entry `[i,j]` = number of individuals with path j(t)->i(t).

`coefs` List of per-person q x (q + p) coefficient matrices (q non-exogenous rows; columns = `[lagged, contemporaneous]`).

`psi` List of per-person q x (q + p) standardized residual covariance matrices, with non-exogenous current variables in rows and `c(lag_names, varnames)` in columns, matching `gimme::gimme()`'s returned `psi` contract.

`fit` Data frame of per-person fit indices (chisq, df, pvalue, rmsea, srmr, nnfi, cfi, bic, aic, logl, status).

`path_counts` p x 2p matrix: how many individuals have each path.

`paths` List of per-person character vectors of lavaan path syntax.

`group_paths` Character vector of group-level paths found.

`individual_paths` List of per-person character vectors of individual-level paths (beyond group).

`syntax` List of per-person full lavaan syntax strings.

`labels` Character vector of variable names.

`n_subjects` Integer. Number of individuals.

`n_obs` Integer vector. Time points per individual.

`config` List of configuration parameters.

See Also

`fit_mlvar`, `fit_graphical_var`, `as_netobject`

Examples

```
# Create simple panel data (3 subjects, 4 variables, 50 time points).
set.seed(42)
n_sub <- 3; n_t <- 50; vars <- paste0("V", 1:4)
rows <- lapply(seq_len(n_sub), function(i) {
  d <- as.data.frame(matrix(rnorm(n_t * 4), ncol = 4))
  names(d) <- vars; d$id <- i; d
})
panel <- do.call(rbind, rows)
res <- fit_gimme(panel, vars = vars, id = "id")
print(res)
```

`fit_graphical_var` *Graphical VAR Estimation*

Description

Estimate a graphical vector autoregressive (GVAR) model from time series or panel data. Jointly estimates a sparse temporal network (L1-penalized VAR coefficients) and a sparse contemporaneous network (graphical lasso on residuals) using EBIC model selection over a lambda grid.

Usage

```
fit_graphical_var(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  lags = 1L,
  n_lambda = 50L,
  gamma = 0.5,
  scale = TRUE,
  center_within = TRUE,
  lambda_min_ratio = 0.05,
  lambda_min_kappa = NULL,
  lambda_min_beta = NULL,
  penalize_diagonal = TRUE,
  lambda_beta = NULL,
  lambda_kappa = NULL,
  regularize_mat_beta = NULL,
  regularize_mat_kappa = NULL,
  maxit_in = 100L,
  maxit_out = 100L,
  delete_missings = TRUE,
  likelihood = c("unpenalized", "penalized"),
  ebic_tol = 1e-04,
  mimic = "current",
  verbose = FALSE,
  min_obs = NULL,
  subject = NULL
)
```

Arguments

data	A data.frame or matrix with columns for variables, and optionally id, day, beep columns for panel/ESM data. A prepared list containing numeric matrices `data_c` (current responses) and `data_l` (lagged design, with or without an intercept column) is also accepted.
vars	Character vector of variable names. May be omitted for prepared input when `data_c` has column names.
id	Character. Name of the person-ID column. If NULL, assumes single subject.
day	Character. Name of the day/session column. Default: NULL.
beep	Character. Name of the beep/measurement column. Default: NULL.
lags	Positive integer vector of explicit lags to include. Default: 1.
n_lambda	Integer scalar, or a two-value vector giving the number of beta and kappa penalties. The latter can be named, for example `c(beta = 30, kappa = 20)`, or unnamed in beta/kappa order. Default: 50.

`gamma` Numeric. EBIC hyperparameter (0 = BIC, higher = sparser). Default: 0.5.

`scale` Logical. Whether to standardize variables. Default: TRUE.

`center_within` Logical. Whether to centre within person when more than one id is present (removes between-person variance). Default: TRUE.

`lambda_min_ratio`
Numeric scalar, or a two-value beta/kappa vector. Ratio of min/max lambda unless overridden per-dimension. Default: 0.05.

`lambda_min_kappa`, `lambda_min_beta`
Numeric or `NULL`. Per-dimension min/max lambda ratios (matching `graphicalVAR`'s `lambda_min_kappa` / `lambda_min_beta`). When `NULL`, fall back to `lambda_min_ratio`.

`penalize_diagonal`
Logical. Penalize the autoregressive diagonal in beta. Default: TRUE (matches `graphicalVAR`).

`lambda_beta` Numeric scalar (or vector), or `NULL`. When supplied, the temporal penalty is pinned to this value instead of being EBIC-selected over a grid – matching `graphicalVAR`'s `lambda_beta` argument (e.g. `lambda_beta = 0.1`). Default `NULL` (EBIC grid).

`lambda_kappa` Numeric scalar (or vector), or `NULL`. As `lambda_beta` but for the contemporaneous (kappa) penalty.

`regularize_mat_beta`
Optional numeric/logical matrix (`p x p` or `p x (p+1)`) of per-element beta penalty multipliers (matches `graphicalVAR`'s `regularize_mat_beta`). `NULL` uses `penalize_diagonal`.

`regularize_mat_kappa`
Optional `p x p` numeric/logical matrix of per-element kappa penalty multipliers (matches `graphicalVAR`'s `regularize_mat_kappa`). `NULL` penalizes all off-diagonals.

`maxit_in`, `maxit_out`
Integer. Max inner (beta) / outer (beta-kappa) iterations. Defaults 100 (matches `maxit.in` / `maxit.out`).

`delete_missings`
Logical. Drop rows with missing current/lagged values. Default TRUE (matches `deleteMissings`).

`likelihood` Either `"unpenalized"` (default; refit precision for the EBIC, matching `graphicalVAR`) or `"penalized"` (use the regularized kappa directly).

`ebic_tol` Numeric. Tolerance for the EBIC tie-break. Default 1e-4.

`mimic` Character. Only `"current"` is supported. Legacy modes error explicitly because idiographic does not claim equivalence to them.

`verbose` Logical. Emit progress messages. Default FALSE.

`min_obs` Integer or `NULL`. Keep only subjects with at least this many observations (counts taken from `data`). Default `NULL`.

`subject` Optional vector naming the exact subject(s) to analyse. Default `NULL` (all subjects).

Details

This is a clean-room reimplementation of the Rothman/Epskamp two-step estimator that is **numerically equivalent to** `graphicalVAR::graphicalVAR()`: identical data preparation (global scaling, optional within-person centring, intercept column, lag-1 construction within id/day blocks), identical lambda grids (`generate_lambdas`), the coupled MRCE beta-update / glasso kappa-update loop, the unpenalized-likelihood EBIC, and the same tie-broken model selection. The committed end-to-end regression tests use tolerance `1e-6`, covering both well-conditioned and numerically difficult fits. That equivalence claim is limited to `mimic = "current"` and `lags = 1`; multiple lags are an idiographic extension and are labelled as such in the returned equivalence metadata.

Value

A list of class `gvar_result` containing:

beta Temporal coefficient matrix, outcome x (intercept + predictors), in `graphicalVAR`'s convention.

temporal The first requested p x p temporal layer as `[outcome, predictor]`; unchanged for the default lag 1 fit.

temporal_layers Named p x p coefficient matrices for every lag.

kappa Precision matrix (p x p, symmetric).

PCC Partial contemporaneous correlations `-cov2cor(kappa)`, diagonal zeroed.

PDC Partial directed correlations.

contemporaneous Alias for `PCC`.

labels Variable names.

n_obs Number of valid lag-pair observations.

lambda_beta, lambda_kappa Selected penalties.

gamma, EBIC EBIC gamma used and the selected EBIC.

References

Epskamp, S., Waldorp, L. J., Mottus, R., & Borsboom, D. (2018). The Gaussian Graphical Model in Cross-Sectional and Time-Series Data. *Multivariate Behavioral Research*, 53(4), 453-480.

Rothman, A. J., Levina, E., & Zhu, J. (2010). Sparse multivariate regression with covariance estimation. *JCGS*, 19(4), 947-962.

Examples

```
set.seed(1)
d <- data.frame(A = rnorm(60), B = rnorm(60))
fit <- fit_graphical_var(d, vars = c("A", "B"), n_lambda = 3,
                         scale = FALSE)
fit$temporal
fit$contemporaneous
```

`fit_graphical_var_each` *Fit a graphical VAR for every subject*

Description

Applies `fit_graphical_var()` to each subject separately, returning one person-specific network per individual — the idiographic "all individuals" workflow. Subjects that cannot be fit (too few lag pairs after listwise deletion) are dropped with a warning.

Usage

```
fit_graphical_var_each(
  data,
  vars,
  id,
  day = NULL,
  beep = NULL,
  min_obs = NULL,
  ...
)
```

Arguments

`data`	A data.frame or matrix with columns for variables, and optionally id, day, beep columns for panel/ESM data. A prepared list containing numeric matrices `data_c` (current responses) and `data_l` (lagged design, with or without an intercept column) is also accepted.
`vars`	Character vector of variable names. May be omitted for prepared input when `data_c` has column names.
`id`	Character. The subject-id column (required here).
`day`	Character. Name of the day/session column. Default: NULL.
`beep`	Character. Name of the beep/measurement column. Default: NULL.
`min_obs`	Integer or `NULL`. Keep only subjects with at least this many observations (counts taken from `data`). Default `NULL`.
`...`	Further arguments passed to `fit_graphical_var()` (e.g. `n_lambda`, `gamma`, `scale`).

Value

A named list of `gvar_result` objects (class `gvar_list`), one element per subject, named by subject id.

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:2, each = 35),
                A = rnorm(70), B = rnorm(70))
fits <- fit_graphical_var_each(d, vars = c("A", "B"), id = "id",
                               n_lambda = 3, scale = FALSE)
names(fits)
```

`fit_idiographic` *Fit an idiographic model through the unified interface*

Description

`fit_idiographic()` dispatches every built-in estimator and workflow through the same entry point. Arguments may be supplied directly or in `params`, which makes a stored configuration directly replayable. Direct arguments and `params` must both be named and cannot overlap; this turns otherwise ambiguous duplicate arguments into an immediate, informative error.

Usage

```
fit_idiographic(data, method, ..., params = list())
```

Arguments

`data`	A data frame or matrix passed to the selected method.
`method`	A registered method name or alias.
`...`	Named arguments passed directly to the selected method.
`params`	A named list of additional method arguments.

Value

The selected method's result, unchanged except for lightweight dispatch and equivalence metadata attributes.

Examples

```
set.seed(1)
d <- data.frame(A = rnorm(80), B = rnorm(80))
fit <- fit_idiographic(d, "var", vars = c("A", "B"), scale = FALSE)
fit2 <- fit_idiographic(d, "ols-var",
                        params = list(vars = c("A", "B"), scale = FALSE))
equivalence(fit)
```

fit_ml *Fit idiographic supervised machine-learning models*

Description

Fits train/test supervised prediction models in an idiographic design: each subject can receive a model trained only on that subject's earlier rows, and the same held-out rows can also be scored by a pooled model trained on all subjects' earlier rows. This mirrors individualized modelling designs where person-specific prediction is compared against a nomothetic pooled baseline.

The implementation is dependency-free beyond base R. Regression supports mean baseline, ordinary least squares, ridge, lasso, elastic net, principal component regression, k-nearest neighbours, and a one-split regression tree. Binary classification supports majority baseline, logistic regression, ridge/lasso/elastic-net logistic regression, linear discriminant analysis, Gaussian naive Bayes, k-nearest neighbours, and a one-split classification tree. Predictors are standardized using training rows only.

Usage

```
fit_ml(
  data,
  outcome,
  predictors,
  id,
  day = NULL,
  beep = NULL,
  task = c("auto", "regression", "classification"),
  model = NULL,
  estimator = NULL,
  compare = c("both", "individual", "pooled"),
  test_prop = 0.2,
  min_train = 10L,
  min_test = 1L,
  lambda = 1,
  alpha = 0.5,
  k = 5L,
  n_components = NULL,
  max_iter = 100L,
  tol = 1e-06,
  standardize = TRUE,
  keep_fits = FALSE,
  ...
)

fit_idiographic_ml(
  data,
  outcome,
```

```
  predictors,
  id,
  day = NULL,
  beep = NULL,
  task = c("auto", "regression", "classification"),
  model = NULL,
  estimator = NULL,
  compare = c("both", "individual", "pooled"),
  test_prop = 0.2,
  min_train = 10L,
  min_test = 1L,
  lambda = 1,
  alpha = 0.5,
  k = 5L,
  n_components = NULL,
  max_iter = 100L,
  tol = 1e-06,
  standardize = TRUE,
  keep_fits = FALSE,
  ...
)

fit_individualized_ml(
  data,
  outcome,
  predictors,
  id,
  day = NULL,
  beep = NULL,
  task = c("auto", "regression", "classification"),
  model = NULL,
  estimator = NULL,
  compare = c("both", "individual", "pooled"),
  test_prop = 0.2,
  min_train = 10L,
  min_test = 1L,
  lambda = 1,
  alpha = 0.5,
  k = 5L,
  n_components = NULL,
  max_iter = 100L,
  tol = 1e-06,
  standardize = TRUE,
  keep_fits = FALSE,
  ...
)
```

Arguments

data	A data.frame or matrix.
outcome	Character. Name of the outcome column.
predictors	Character vector of predictor columns.
id	Character. Name of the subject/person ID column.
day, beep	Optional ordering columns. Rows are ordered by id, day, and beep before the last rows for each subject are held out.
task	"auto", "regression", or "classification". Auto treats a numeric outcome as regression and a two-level non-numeric outcome as binary classification.
model	NULL for the task default, "all" for all native models for the selected task, or a character vector of simple model names. Regression models are "mean", "linear", "ridge", "lasso", "elastic", "pcr", "knn", and "tree". Classification models are "majority", "logistic", "ridge", "lasso", "elastic", "lda", "bayes", "knn", and "tree".
estimator	NULL for each model's default estimator, or a named character vector/list mapping model names to estimator names. The native base-R estimator is "native". This is where package-specific backends belong when the same model can be estimated more than one way.
compare	Which models to fit: "both" (default), "individual", or "pooled".
test_prop	Proportion of each subject's ordered rows held out from the end of the series. Default 0.2.
min_train	Minimum complete training rows required for a model. Default 10.
min_test	Minimum held-out rows required per subject. Default 1.
lambda	Ridge penalty for model = "ridge". The intercept is not penalized. Also used by lasso and elastic-net models. Default 1.
alpha	Elastic-net mixing value in [0, 1]; 0 is ridge and 1 is lasso. Default 0.5.
k	Number of neighbours for model = "knn". Default 5.
n_components	Number of principal components for model = "pcr". Default uses min(5, n_predictors, n_train - 1).
max_iter	Maximum iterations for coordinate-descent penalized models. Default 100.
tol	Convergence tolerance for iterative models. Default 1e-6.
standardize	Logical. Standardize predictors using training-set means and SDs? Default TRUE.
keep_fits	Logical. Store fitted internal model objects? Default FALSE.
...	Optional model controls using the same names as the explicit tuning arguments (lambda, alpha, k, n_components, max_iter, or tol). Unknown names are rejected.

Value

An idioml_result with $predictions, $metrics, $coefficients, $failures, and optionally $fits.

Examples

```
set.seed(1)
d <- data.frame(
  id = rep(1:4, each = 40),
  beep = rep(seq_len(40), 4),
  x1 = rnorm(160),
  x2 = rnorm(160)
)
d$y <- 0.4 * d$x1 - 0.2 * d$x2 + rep(c(-1, 0, 1, 0.5), each = 40) +
  rnorm(160, sd = 0.4)
fit <- fit_ml(d, outcome = "y", predictors = c("x1", "x2"),
              id = "id", beep = "beep",
              model = c("linear", "ridge", "knn"))
fit$metrics
coefs(fit)
```

`fit_mlvar` *Build a Multilevel Vector Autoregression (mlVAR) network*

Description

Estimates three networks from ESM/EMA panel data, matching validated `mlVAR::mlVAR()` configurations at machine precision: (1) a directed temporal network of fixed-effect lagged regression coefficients, (2) an undirected contemporaneous network of partial correlations among residuals, and (3) an undirected between-subjects network of partial correlations derived from the person-mean fixed effects.

Usage

```
fit_mlvar(
  data,
  vars,
  id,
  day = NULL,
  beep = NULL,
  lags = 1L,
  estimator = c("lmer", "default", "lm", "Mplus"),
  temporal = c("fixed", "correlated", "orthogonal", "unique", "default"),
  contemporaneous = c("fixed", "correlated", "orthogonal", "unique", "default"),
  AR = FALSE,
  scale = FALSE,
  scaleWithin = FALSE,
  nCores = 1L,
  verbose = FALSE,
  lag = NULL,
  standardize = NULL,
  min_obs = NULL,
  subject = NULL,
```

```
  engine = c("frequentist", "bayes", "mplus", "reference"),
  standardize_mode = NULL,
  missing = c("omit", "fail", "model"),
  compare_to_lags = NULL,
  true_means = NULL,
  detrend = c("none", "position"),
  na_rm = TRUE,
  orthogonal = NULL,
  ...
)
```

Arguments

data	A data.frame containing the panel data.
vars	Character vector of variable column names to model.
id	Character string naming the person-ID column.
day	Character string naming the day/session column, or NULL. When provided, lag pairs are only formed within the same day.
beep	Character string naming the measurement-occasion column, or NULL. When NULL, row position within each (id, day) is used.
lags	One or more unique positive integer lag orders (mlVAR's lags).
estimator	Character. Frequentist estimator: "lmer" (multilevel) or "lm" (separate person-specific models, requiring temporal = "unique"). The legacy value "Mplus" selects engine = "mplus".
temporal, contemporaneous	Character random-effect structure. The native frequentist engine supports fixed, correlated, orthogonal, and unique person-specific effects. The Bayesian engine maps correlated/orthogonal/ unique temporal effects to its full random-slope model.
AR	Logical. If TRUE, estimate only autoregressive (own-lag) temporal effects, giving a diagonal temporal matrix (matches mlVAR(AR = TRUE)). For the native frequentist/reference path this requires estimator = "lmer". Default FALSE.
scale	Logical. If TRUE, each variable is grand-mean centred and divided by its pooled SD before augmentation (mlVAR's scale). Default FALSE. (The deprecated standardize is an alias.)
scaleWithin	Logical. If TRUE, additionally scale within person (mlVAR's scaleWithin). Default FALSE.
nCores	Positive integer number of outcome models to fit in parallel. Uses forked workers on Unix-like systems and a PSOCK cluster on Windows.
verbose	Logical. Emit progress messages. Default FALSE.
lag	Deprecated alias for lags.
standardize	Deprecated alias for scale.
min_obs	Integer or NULL. Keep only subjects with at least this many observations (counts taken from data).

subject	Optional vector naming the exact subject(s) to analyse.
engine	Estimation engine: "frequentist" (native lme4/base R), "bayes" (the native DSEM sampler), "mplus" (licensed Mplus through fit_mlvar_mplus()), or "reference" (direct mlVAR::mlVAR() followed by conversion to idiographic's tidy result contract).
standardize_mode	Easy standardization vocabulary: "none", "global", "within", or "both". When supplied it sets scale and scaleWithin; the logical legacy arguments remain supported.
missing	Missing-data policy: "omit", "fail", or "model". "fail" checks model variables and ID/day/beep ordering keys. Within-model imputation is available with the Bayesian random-slope engine.
compare_to_lags	Optional positive lag vector used only to align the analysis rows when comparing models with different lag orders. It must include every fitted value in lags; for example, use c(1, 2) while fitting lag 1 for a comparison with a lag-2 model.
true_means	Optional data frame containing id and all vars, used as known person means instead of sample means.
detrend	"none" or "position". Position detrending standardizes each measurement position across subjects before model standardization.
na_rm	Logical legacy spelling for whether incomplete model rows are omitted. FALSE is equivalent to missing = "fail".
orthogonal	Deprecated upstream compatibility flag. When supplied it sets temporal = "orthogonal" (TRUE) or "correlated" (FALSE).
...	Engine-specific controls. For example n_iter, n_chains, and residual for the Bayesian engine, or Mplus controls for the Mplus engine.

Details

The algorithm follows mlVAR's lmer pipeline exactly:

1. Drop rows with NA in id/day/beep and optionally grand-mean standardize each variable.
2. Expand the per-(id, day) beep grid and right-join original values, producing the augmented panel (augData).
3. Add within-person lagged predictors (L1_*) and person-mean predictors (PM_*).
4. For each outcome variable fit lmer(y ~ within + between-except-own-PM + (1 | id)) with REML = FALSE. Collect the fixed-effect temporal matrix B, between-effect matrix Gamma, random-intercept SDs (mu_SD), and lmer residual SDs.
5. Contemporaneous network: cor2pcor(D %*% cov2cor(cor(resid)) %*% D).
6. Between-subjects network: cor2pcor(pseudoinverse(forcePositive(D (I - Gamma)))).

The committed oracle matrix validates fixed temporal/contemporaneous lmer fits at lags 1 and 1+2, preprocessing controls (scale, scaleWithin, compareToLags, trueMeans, and position detrending), and lag-1 estimator = "lm", temporal = "unique" fits across every supported contemporaneous structure. Other configurations carry a narrower declaration available through equivalence().

Value

A dual-class `c("net_mlvar", "netobject_group")` object — a named list of three full netobjects, one per network, plus model-level metadata stored as attributes. Each element is a standard `c("netobject", "cograph_network")` weight-matrix wrapper (no raw `$data`), so `print()`, `summary()`, `coefs()`, and `cograph::splot(fit$temporal)` work directly. The three constituents are matrix-wrapped and carry no underlying panel data, so any data-resampling workflow (bootstrap, reliability, stability) must start from the original panel rather than from these wrappers. Structure:

`fit$temporal` Directed netobject for the `d x d` matrix of fixed-effect lagged coefficients. `$weights[i, j]` is the effect of variable j at t-lag on variable i at t. `method = "mlvar_temporal"`, `directed = TRUE`.

`fit$contemporaneous` Undirected netobject for the `d x d` partial-correlation network of within-person lmer residuals. `method = "mlvar_contemporaneous"`, `directed = FALSE`.

`fit$between` Undirected netobject for the `d x d` partial-correlation network of person means, derived from `D (I - Gamma)`. `method = "mlvar_between"`, `directed = FALSE`. **Convention:** when a random-intercept SD is 0 the between network is not estimable; idiographic returns an all-zero matrix (with a warning) as a plotting-oriented convention, whereas `mlVAR` returns an all-`NA` matrix. The contemporaneous network follows the same zero-on-degeneracy convention. This is a deliberate departure from strict reference equivalence in the singular case.

`attr(fit, "coefs")` / `coefs()` Tidy `data.frame` with one row per `(outcome, predictor)` pair and columns `outcome`, `predictor`, `beta`, `se`, `t`, `p`, `ci_lower`, `ci_upper`, `significant`. Filter, sort, or plot with base R or the tidyverse. Retrieve with `coefs(fit)`.

`attr(fit, "n_obs")` Number of rows in the augmented panel after na.omit.

`attr(fit, "n_subjects")` Number of unique subjects remaining.

`attr(fit, "lag")` Lag order used.

`attr(fit, "standardize")` Logical; whether pre-augmentation standardization was applied.

Observation keys

When `beep` is supplied, every complete `(id, day, beep)` key (or `(id, beep)` when `day = NULL`) must be unique. Duplicate keys often indicate that a study-period/session column was lost during data conversion. Because upstream join behaviour is row-order dependent in that case, `fit_mlvar()` errors and asks you to resolve or explicitly deduplicate the source data.

See Also

`fit_gimme()`, `fit_graphical_var()`, `as_netobject()`

Examples

```
set.seed(1)
n_id <- 8; n_t <- 30; vars <- c("A", "B", "C")
rows <- lapply(seq_len(n_id), function(i) {
  m <- as.data.frame(matrix(rnorm(n_t * 3), ncol = 3))
  names(m) <- vars
```

```
  m$id <- i; m$day <- 1L; m$beep <- seq_len(n_t)
  m
})
d <- do.call(rbind, rows)
fit <- fit_mlvar(d, vars = vars, id = "id", day = "day", beep = "beep")
print(fit)
summary(fit)
```

`fit_mlvar_bayes` *Build a Bayesian multilevel VAR network (Mplus DSEM-targeted)*

Description

Native, pure-R Bayesian estimator for a two-level VAR(1) that statistically reproduces Mplus DSEM output (the estimator behind `mlVAR::mlVAR(estimator = "Mplus")`) without needing Mplus installed. A conjugate Gibbs sampler estimates a fixed temporal matrix, a within-person residual (contemporaneous) network, and a between-person network, using latent mean centring and Mplus's default priors. Point estimates are posterior medians with posterior SDs and 95% credible intervals.

Usage

```
fit_mlvar_bayes(
  data,
  vars,
  id,
  day = NULL,
  beep = NULL,
  lags = 1L,
  temporal = c("fixed", "default", "random"),
  contemporaneous = c("fixed", "default"),
  residual = c("fixed", "random"),
  scale = TRUE,
  scaleWithin = FALSE,
  tinterval = NULL,
  impute = FALSE,
  n_iter = 4000L,
  n_burnin = NULL,
  n_chains = 2L,
  thin = 1L,
  seed = NULL,
  min_obs = NULL,
  subject = NULL,
  verbose = FALSE
)
```

Arguments

data	A data.frame containing the panel data.
vars	Character vector of variable column names to model (length >= 2).
id	Character string naming the person-ID column.
day	Character string naming the day/session column, or NULL.
beep	Character string naming the measurement-occasion column, or NULL. When NULL, row position within each (id, day) block is used.
lags	Integer lag order; only 1 is supported (matches Mplus DSEM defaults here).
temporal	Character. "fixed" (default) fits fixed temporal effects with random intercepts (Mplus DSEM temporal = "fixed"). "random" fits the full DSEM with person-specific temporal matrices B_i and a full random-effect covariance over (mu_i, vec(B_i)); the temporal network then reports the posterior mean transition matrix and attr(fit, "slope_sd") holds the per-coefficient random-slope SDs. "random" needs more subjects estimable random-effect covariance: at least 2 * (p + p^2) + 1 subjects.
contemporaneous	Character. Only "fixed" is implemented.
residual	Character. "fixed" (default) uses one shared population within-person residual covariance. "random" (only with temporal = "random") gives each subject their own residual covariance Sigma_W_i via a conjugate hierarchical inverse-Wishart (Sigma_W_i ~ IW(Lambda, p + 2), Lambda ~ Wishart), matching DSEM person-specific innovation variances; the reported contemporaneous network is then the population-average residual covariance.
scale	Logical. Global grand-mean/SD standardization of each variable before fitting (Mplus/mlVAR scale = TRUE). Default TRUE.
scaleWithin	Logical. Additionally within-person scale each variable. Default FALSE.
tinterval	Numeric or NULL. When supplied, beep is treated as a continuous time variable and binned onto a regular grid of this width (Mplus TINTERVAL); the integer bin becomes the occasion index for gap-aware lagging, and multiple observations in one (id, day, bin) slot are collapsed to the first. Lagging is gap-aware in all cases: lag-1 pairs are only formed between consecutive occasions, so missing occasions never create spurious lag pairs. Default NULL.
impute	Logical. If TRUE (only with temporal = "random"), missing observations are imputed **within the model** each MCMC iteration (data augmentation), rather than dropped: each person's series is expanded to a full occasion grid and every latent cell is drawn from its Gaussian full conditional (as an outcome at t and a predictor at t+1), using a vectorised even/odd (checkerboard) block sweep. This matches how Mplus / Stan / JAGS handle missing data and removes the listwise-deletion bias under heavy missingness, at extra computational cost. Default FALSE.
n_iter	Integer. Total MCMC iterations per chain. Default 4000.
n_burnin	Integer. Burn-in iterations discarded per chain. Default n_iter / 2 (Mplus's first-half burn-in convention).
n_chains	Integer. Number of independent chains. Default 2.

thin	Integer. Keep every thin-th post-burn-in draw. Default 1.
seed	Integer or NULL. Base RNG seed (chain c uses seed + c).
min_obs	Integer or NULL. Keep only subjects with at least this many observations before fitting.
subject	Optional vector naming the exact subject(s) to analyse.
verbose	Logical. Emit progress messages. Default FALSE.

Details

The sampler alternates five conjugate full-conditional draws per iteration: the latent person means mu_i (Gaussian), the fixed temporal matrix B (matrix-normal), the within residual covariance Sigma_W (inverse-Wishart), the grand mean alpha (Gaussian), and the between covariance Sigma_B (inverse-Wishart). The lagged predictor is recentred on the current mu_i draw every iteration (latent mean centring). Data are globally standardized first (matching mlVAR's scale = TRUE); the first observation of each block is used only as a lag (condition-on-first).

Validated to statistical (Monte-Carlo-error) equivalence against real Mplus 9 DSEM output on standardized synthetic panels: posterior medians of B, Sigma_W, Sigma_B agree with Mplus to well within a posterior SD.

Value

A net_mlvar_bayes object (also inheriting net_mlvar), a named list of three netobjects (temporal, contemporaneous, between) with posterior-summary attributes. coefs() returns a tidy table with estimate (posterior median), posterior_sd, ci_lower, ci_upper, p (one-tailed), and significant (95% CI excludes 0). Posterior draws and the max Gelman-Rubin PSR are kept in attributes.

See Also

fit_mlvar() (frequentist lmer path), fit_mlvar_mplus() (true-Mplus wrapper).

Examples

```
set.seed(1)
n_id <- 10; n_t <- 40; vars <- c("A", "B")
rows <- lapply(seq_len(n_id), function(i) {
  y <- matrix(0, n_t, 2)
  for (t in 2:n_t) y[t, ] <- c(0.3, 0.15) * y[t - 1, ] + rnorm(2)
  data.frame(id = i, beep = seq_len(n_t), A = y[, 1], B = y[, 2])
})
d <- do.call(rbind, rows)
fit <- fit_mlvar_bayes(d, vars = vars, id = "id", beep = "beep",
                           n_iter = 500, seed = 1)
print(fit)
coefs(fit)
```

`fit_mlvar_mplus` *Build an Mplus-backed multilevel VAR network*

Description

Runs the Mplus Bayesian estimator exposed by `mlVAR::mlVAR(estimator = "Mplus")` and converts the returned posterior summaries into idiographic's network/tidy accessors. This is a true Mplus backend: Mplus must be installed and discoverable by `MplusAutomation::detectMplus()`.

Usage

```
fit_mlvar_mplus(
  data,
  vars,
  id,
  day = NULL,
  beep = NULL,
  lags = 1L,
  temporal = c("fixed", "correlated", "orthogonal", "default"),
  contemporaneous = c("fixed", "correlated", "orthogonal", "default"),
  nCores = 1L,
  scale = TRUE,
  scaleWithin = FALSE,
  MplusSave = TRUE,
  MplusName = "mlVAR_mplus",
  iterations = "(2000)",
  chains = nCores,
  signs,
  min_obs = NULL,
  subject = NULL,
  workdir = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	A `data.frame` containing the panel data.
`vars`	Character vector of variable column names to model.
`id`	Character string naming the person-ID column.
`day`	Character string naming the day/session column, or `NULL`. Mplus estimation in `mlVAR` does not directly support `day`; when supplied it is passed through so `mlVAR` can prepare the row order, but `mlVAR` will warn about the Mplus limitation.
`beep`	Character string naming the measurement-occasion column, or `NULL`.

lags	Integer lag order. The Mplus backend currently supports 1.
temporal, contemporaneous	Random-effect structure passed to mlVAR. Supported Mplus values are "fixed", "correlated", "orthogonal", and "default".
nCores	Number of Mplus processors/chains.
scale, scaleWithin	Standardization options passed to mlVAR.
MplusSave	Logical. Keep Mplus input/output files in the working directory? Default TRUE.
MplusName	File stem for Mplus input/output files.
iterations	Mplus BITERATIONS string, e.g. "(2000)".
chains	Number of Mplus chains. Defaults to nCores.
signs	Optional sign matrix for contemporaneous random effects.
min_obs	Integer or NULL. Keep only subjects with at least this many observations before fitting.
subject	Optional vector naming the exact subject(s) to analyse.
workdir	Directory in which Mplus files should be written/run. Default uses the current working directory.
verbose	Logical. Show progress from mlVAR/Mplus.
...	Additional arguments passed to mlVAR::mlVAR().

Value

A net_mplus object, also inheriting from net_mlvar, with temporal, contemporaneous, and between networks plus Mplus metadata in attributes. The original mlVAR/Mplus object is available as attr(x, "mplus").

See Also

fit_mlvar()

Examples

```
## Not run:
fit <- fit_mlvar_mplus(
  data, vars = c("A", "B", "C"), id = "id", beep = "time",
  temporal = "fixed", contemporaneous = "fixed",
  MplusName = "my_mplus_mlvar"
)
edges(fit)
attr(fit, "mplus")$output$summaries

## End(Not run)
```

`fit_rolling_graphical_var` *Estimate rolling-window graphical VAR networks*

Description

Fits `fit_graphical_var()` over ordered, overlapping windows within each subject. This is the time-varying graphical VAR companion to `fit_rolling_var()`: every window uses graphical VAR's lag construction, EBIC/penalty settings, and tidy coefficient access, then returns one coefficient table per window.

Usage

```
fit_rolling_graphical_var(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  window_size,
  step = 1L,
  scale = TRUE,
  center_within = TRUE,
  delete_missings = TRUE,
  min_obs = NULL,
  subject = NULL,
  keep_fits = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`id`	Character. Name of the person-ID column, or `NULL` for a single series.
`day`	Character. Name of the day/session column, or `NULL`.
`beep`	Character. Name of the measurement-occasion column, or `NULL`.
`window_size`	Integer number of ordered rows per rolling window.
`step`	Integer number of rows to advance between windows. Default `1`.
`scale`	Logical. Whether to standardize variables inside each window. Default `TRUE`.
`center_within`	Logical. Whether to centre within person inside each window when more than one id is present. Default `TRUE`.
`delete_missings`	Logical. Drop incomplete current/lagged rows. Default `TRUE`.

min_obs	Integer or NULL. Keep only subjects with at least this many observations before rolling.
subject	Optional vector naming the subject(s) to analyse.
keep_fits	Logical. Store successful gvar_result fits? Default FALSE.
...	Further arguments passed to fit_graphical_var(), such as n_lambda, gamma, lambda_beta, or lambda_kappa.

Value

A rolling_gvar_result with $estimates, $windows, $failures, and optionally $fits. $estimates is a tidy coefficient table with subject/window metadata plus network, from, to, and weight.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = rep(1:5, each = 20),
                beep = rep(1:20, 5),
                A = rnorm(100), B = rnorm(100), C = rnorm(100))
tv <- fit_rolling_graphical_var(d, vars = c("A", "B", "C"), id = "id",
                            day = "day", beep = "beep",
                            window_size = 50, step = 25,
                            scale = FALSE, n_lambda = 5)
head(tv$estimates)
```

---

fit_rolling_var *Estimate rolling-window ordinary VAR networks*

Description

Fits fit_var() over ordered, overlapping windows within each subject. This is a simple time-varying idiographic baseline: every window uses the same lag construction, scaling, within-person centring, and tidy coefficient access as fit_var(), but returns one coefficient table per window.

Usage

```
fit_rolling_var(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  window_size,
  step = 1L,
  scale = TRUE,
  center_within = TRUE,
  delete_missings = TRUE,
  min_obs = NULL,
```

```
  subject = NULL,
  keep_fits = FALSE
)
```

Arguments

data	A data.frame or matrix with columns for variables and optional id/day/beep columns.
vars	Character vector of variable names.
id	Character. Name of the person-ID column, or NULL for a single series.
day	Character. Name of the day/session column, or NULL.
beep	Character. Name of the measurement-occasion column, or NULL.
window_size	Integer number of ordered rows per rolling window.
step	Integer number of rows to advance between windows. Default 1.
scale	Logical. Whether to standardize variables inside each window. Default TRUE.
center_within	Logical. Whether to centre within person inside each window when more than one id is present. Default TRUE.
delete_missings	Logical. Drop incomplete current/lagged rows. Default TRUE.
min_obs	Integer or NULL. Keep only subjects with at least this many observations before rolling.
subject	Optional vector naming the subject(s) to analyse.
keep_fits	Logical. Store successful var_result fits? Default FALSE.

Value

A rolling_var_result with $estimates, $windows, $failures, and optionally $fits. $estimates is a tidy coefficient table with subject/window metadata plus network, from, to, and weight.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = rep(1:5, each = 20),
                beep = rep(1:20, 5),
                A = rnorm(100), B = rnorm(100), C = rnorm(100))
tv <- fit_rolling_var(d, vars = c("A", "B", "C"), id = "id",
                  day = "day", beep = "beep",
                  window_size = 40, step = 20, scale = FALSE)
head(tv$estimates)
```

`fit_usem` *Build a user-specified unified SEM network*

Description

Fits person-specific unified Structural Equation Models (uSEM) for intensive longitudinal data. A uSEM combines lagged directed effects, optional contemporaneous directed effects, and optional residual covariances in one SEM. Unlike `fit_gimme()`, this function does no automated path search: the model is fixed by `temporal`, `contemporaneous`, `residual_cov`, and `paths`. With `trim = TRUE`, idiographic uses an independent clean-room modification-index entry and z-value pruning layer over the declared candidate set.

Usage

```
fit_usem(
  data,
  vars,
  id,
  time = NULL,
  day = NULL,
  beep = NULL,
  min_obs = NULL,
  subject = NULL,
  temporal = c("ar", "all", "none"),
  contemporaneous = c("none", "all"),
  residual_cov = TRUE,
  trim = FALSE,
  trim_alpha = 0.05,
  trim_fit_criteria = 3L,
  cfi_cutoff = 0.95,
  tli_cutoff = 0.95,
  rmsea_cutoff = 0.08,
  srmr_cutoff = 0.08,
  paths = NULL,
  exogenous = NULL,
  standardize = FALSE,
  estimator = "ml",
  seed = NULL
)
```

Arguments

`data`	A `data.frame` in long format.
`vars`	Character vector of time-varying variables.
`id`	Character string naming the person-ID column.
`time`	Character string naming the within-person ordering column, or `NULL`.

`day` Character string naming the day/session column, or `NULL`. When supplied, lag pairs are formed only within the same `(id, day)` block.

`beep` Character string naming the measurement-occasion column, or `NULL`. Used with `day` to order observations when `time` is not supplied.

`min_obs` Integer or `NULL`. Keep only subjects with at least this many observations.

`subject` Optional vector naming the subject(s) to analyse.

`temporal` `"ar"` (own-lag only; default), `"all"` (all lagged predictors), `"none"`, or a character vector of lavaan regressions such as `"A ~ Blag"`.

`contemporaneous` `"none"` (default), `"all"` (all directed lag-0 predictors except self-regressions), or lavaan regressions such as `"B ~ A"`.

`residual_cov` Logical. Estimate residual covariances among current endogenous variables? Default `TRUE`.

`trim` Logical. If `TRUE`, treat `temporal`, `contemporaneous`, and `residual_cov` as an eligible search space: start from the structural baseline, add paths by modification index until fit criteria are met, then prune weak paths. This is an idiographic clean-room search layer, not a clone of any external package. Default `FALSE` fits the exact fixed syntax.

`trim_alpha` Significance level used for modification-index entry and z-value pruning when `trim = TRUE`. Default `0.05`.

`trim_fit_criteria` Number of fit criteria that must pass before forward search stops. Default `3`.

`cfi_cutoff`, `tli_cutoff`, `rmsea_cutoff`, `srmr_cutoff` Fit thresholds used by trimmed uSEM.

`paths` Extra lavaan syntax lines to include unchanged.

`exogenous` Optional subset of `vars` to treat as exogenous current variables. They can predict endogenous variables but are not outcomes.

`standardize` Logical. Standardize variables per person before fitting.

`estimator` Lavaan estimator. Default `"ml"`.

`seed` Optional random seed.

Value

A `net_usem` object with average `$temporal`, `$contemporaneous`, and `$residual_cov` matrices, per-subject matrices in `$subjects`, a tidy coefficient table from `coefs()`, fit indices, syntax, labels, and configuration metadata.

See Also

`fit_gimme()`, `fit_graphical_var()`, `fit_mlvar()`

Examples

```
set.seed(1)
d <- data.frame(
  id = rep(1:4, each = 30),
  t = rep(seq_len(30), 4),
  A = rnorm(120), B = rnorm(120), C = rnorm(120)
)
fit <- fit_usem(d, vars = c("A", "B", "C"), id = "id", time = "t")
edges(fit)
```

---

fit_var *Build an ordinary least-squares VAR network*

Description

Fits a transparent VAR(1) baseline from intensive longitudinal data using ordinary least squares: current variables are regressed on an intercept and lag-1 predictors. The lag construction, scaling, within-person centring, and day-boundary behaviour match `fit_graphical_var()`, but no regularization or EBIC model selection is applied.

Usage

```
fit_var(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  lags = 1L,
  scale = TRUE,
  center_within = TRUE,
  delete_missings = TRUE,
  min_obs = NULL,
  subject = NULL
)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`id`	Character. Name of the person-ID column, or `NULL` for a single series.
`day`	Character. Name of the day/session column, or `NULL`.

beep	Character. Name of the measurement-occasion column, or NULL.
lags	Integer. Only 1 is supported.
scale	Logical. Whether to standardize variables before lagging. Default TRUE.
center_within	Logical. Whether to centre within person when more than one id is present. Default TRUE.
delete_missings	Logical. Drop incomplete current/lagged rows. Default TRUE.
min_obs	Integer or NULL. Keep only subjects with at least this many observations.
subject	Optional vector naming the subject(s) to analyse.

Value

A var_result object with temporal OLS coefficients, residual covariance, residual precision, contemporaneous partial correlations, and tidy access through edges(), coefs(), nodes(), and summary().

Examples

```
set.seed(1)
d <- data.frame(id = 1, A = rnorm(80), B = rnorm(80), C = rnorm(80))
fit <- fit_var(d, vars = c("A", "B", "C"), id = "id")
edges(fit)
```

---

fit_var_bayes *Build a Bayesian VAR(1) network (unregularized, Mplus-targeted)*

Description

Native, pure-R Bayesian VAR(1) that reproduces Mplus's Bayesian (DSEM/time-series) estimates without needing Mplus. It is the unregularized Bayesian counterpart of fit_graphical_var(): instead of a graphical-lasso / EBIC sparse fit, it estimates a full VAR(1) with a flat prior on the temporal coefficients and an inverse-Wishart prior on the residual precision, then reports the temporal network B and the contemporaneous partial-correlation network derived from the residual covariance. With more than one subject the data are within-person centred and pooled (as in fit_graphical_var()).

Usage

```
fit_var_bayes(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  lags = 1L,
  scale = TRUE,
  center_within = TRUE,
```

```
  n_iter = 4000L,
  n_burnin = NULL,
  n_chains = 2L,
  thin = 1L,
  seed = NULL,
  min_obs = NULL,
  subject = NULL,
  verbose = FALSE
)
```

Arguments

data	A data.frame or matrix.
vars	Character vector of variable names (length >= 2).
id	Character. Person-ID column, or NULL for a single series.
day	Character. Day/session column, or NULL.
beep	Character. Beep/measurement column, or NULL.
lags	Integer lag order; only 1 is supported.
scale	Logical. Global standardization of each variable. Default TRUE.
center_within	Logical. Within-person centre when >1 id (removes between-person variance, as in fit_graphical_var()). Default TRUE.
n_iter, n_burnin, n_chains, thin	MCMC controls. Defaults 4000, n_iter/2, 2, 1.
seed	Integer or NULL. Base seed (chain c uses seed + c).
min_obs	Integer or NULL. Keep only subjects with at least this many observations.
subject	Optional vector naming the exact subject(s) to analyse.
verbose	Logical. Progress messages. Default FALSE.

Value

A var_bayes_result object (a cograph group with temporal and contemporaneous netobjects) carrying beta, temporal, kappa, PCC, PDC, posterior draws, and a tidy coefs() table (posterior median, SD, 95% CI, one-tailed p, significance by CI excluding 0).

See Also

fit_graphical_var() (regularized GLASSO/EBIC), fit_var() (OLS), fit_mlvar_bayes() (multilevel Bayesian VAR).

Examples

```
set.seed(1)
y <- matrix(0, 200, 2)
for (t in 2:200) y[t, ] <- c(0.4, 0.3) * y[t - 1, ] + rnorm(2)
d <- data.frame(A = y[, 1], B = y[, 2])
fit <- fit_var_bayes(d, vars = c("A", "B"), n_iter = 500, seed = 1)
```

```
print(fit)
coefs(fit)
```

`fit_var_each` *Fit an ordinary least-squares VAR for every subject*

Description

Applies `fit_var()` to each subject separately, returning one transparent person-specific OLS VAR result per individual. This is the unregularized companion to `fit_graphical_var_each()` and is useful as an equivalence baseline for checking lag construction, scaling, and temporal coefficient direction.

Usage

```
fit_var_each(data, vars, id, day = NULL, beep = NULL, min_obs = NULL, ...)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`id`	Character. Name of the person-ID column; required.
`day`	Character. Name of the day/session column, or `NULL`.
`beep`	Character. Name of the measurement-occasion column, or `NULL`.
`min_obs`	Integer or `NULL`. Keep only subjects with at least this many observations.
`...`	Further arguments passed to `fit_var()`.

Value

A named list of `var_result` objects (class `var_list`), one element per subject, named by subject id. Subjects that cannot be fit are dropped with a warning.

Examples

```
set.seed(1)
d <- data.frame(
  id = rep(1:3, each = 40),
  day = rep(1, 120),
  beep = rep(seq_len(40), 3),
  A = rnorm(120), B = rnorm(120), C = rnorm(120)
)
fits <- fit_var_each(d, vars = c("A", "B", "C"), id = "id",
                     day = "day", beep = "beep")
fits[["1"]]
```

get_estimator *Get a registered estimator function*

Description

Get a registered estimator function

Usage

```
get_estimator(method)
```

Arguments

method	A registered method name or alias. Names are case-insensitive; spaces, hyphens, and periods are normalized to underscores.

Value

`get_estimator()` returns the registered fitting function.

Examples

```
var_fitter <- get_estimator("var")
is.function(var_fitter)
```

glasso_fit *Fit a graphical lasso at a fixed penalty*

Description

Solves the graphical lasso problem

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

with a pure-R block-coordinate-descent solver (Friedman, Hastie and Tibshirani, 2008). No compiled code and no external dependency is involved, so the estimator is available wherever R is.

The objective is strictly convex, so its minimiser is unique and a sufficiently converged fit is *the* global optimum regardless of solver. `glasso_kkt()` certifies that directly from the stationarity conditions, without reference to any other implementation.

Usage

```
glasso_fit(
  S,
  rho,
  penalize_diagonal = FALSE,
  zero = NULL,
  max_outer = 10000,
  tol_outer = 1e-08,
  max_inner = 10000,
  tol_inner = 1e-10
)
```

Arguments

S	Covariance or correlation matrix (p x p, symmetric, finite).
rho	Either a single non-negative penalty applied to every off-diagonal entry, or a p x p non-negative matrix of element-wise penalties.
penalize_diagonal	Logical. Penalise the diagonal of Theta as well? Default FALSE, matching the usual EBIC-glasso convention.
zero	Optional two-column matrix of (row, col) index pairs whose Theta entries are hard-constrained to zero. Used for the unpenalised refit of a selected edge set. The diagonal is never constrained.
max_outer, tol_outer	Outer (column-sweep) iteration cap and convergence tolerance.
max_inner, tol_inner	Inner (coordinate-descent) iteration cap and convergence tolerance.

Value

An object of class glasso_result: a list with

wi p x p estimated precision matrix Θ.

w p x p estimated (regularised) covariance Θ^{-1}.

beta p x p matrix of lasso coefficients from the column sweep, reusable as a warm start.

rho The penalty as supplied – a scalar or a p x p matrix.

penalize_diagonal The diagonal-penalty flag used.

zero The hard zero constraints applied, or NULL.

S The covariance the fit was made to, retained so glasso_kkt() can certify the result on its own. For large p in a resampling loop this is a real retention cost; glasso_path() does not retain it.

The wi and w element names are deliberately identical to those returned by glasso::glasso(), so existing call sites port unchanged. Use as.data.frame() for a tidy one-row-per-edge partial-correlation table.

References

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

See Also

`glasso_path()` for a whole penalty path, `glasso_kkt()` for an optimality certificate, and `fit_graphical_var()` for the estimator that consumes this kernel.

Examples

```
set.seed(1)
z <- rnorm(200)
x <- cbind(A = z + rnorm(200, 0, 0.5), B = z + rnorm(200, 0, 0.5),
           C = rnorm(200), D = rnorm(200))
fit <- glasso_fit(cov(x), rho = 0.05)
fit
as.data.frame(fit)
glasso_kkt(fit)
```

`glasso_kkt` *Certify a graphical-lasso solution from its optimality conditions*

Description

Returns the largest violation of the graphical-lasso stationarity (KKT) conditions. Because the objective is strictly convex, a value near zero certifies the global optimum *directly from the estimand*, with no appeal to any reference solver. For `W = Theta^{-1}` the subgradient conditions are

- diagonal, unpenalised: $W_{ii} = S_{ii}$;
- diagonal, penalised: $W_{ii} - S_{ii} = \rho_{ii}$;
- off-diagonal with $\Theta_{ij} \neq 0$: $W_{ij} - S_{ij} = \rho_{ij} \operatorname{sign}(\Theta_{ij})$;
- off-diagonal with $\Theta_{ij} = 0$: $|W_{ij} - S_{ij}| \leq \rho_{ij}$.

Usage

```
glasso_kkt(
  x,
  S = NULL,
  rho = NULL,
  penalize_diagonal = NULL,
  zero = NULL,
  active_tol = 1e-08
)
```

Arguments

x	A glasso_result from glasso_fit(), or a precision matrix.
S	Covariance the model was fit to. Required only when x is a bare matrix; taken from the fit otherwise.
rho	Penalty (scalar or p x p matrix). Required only when x is a bare matrix.
penalize_diagonal	Logical, whether the diagonal was penalised. Required only when x is a bare matrix.
zero	Two-column matrix of hard-constrained (row, col) index pairs, as passed to glasso_fit(). Taken from the fit when x is a glasso_result. Constrained entries are excluded from the check: at a hard-constrained edge the inactive-edge inequality does not apply, because the equality constraint carries its own multiplier that absorbs the residual. Checking them anyway reports optimal fits as non-optimal.
active_tol	Magnitude above which an off-diagonal entry counts as active. Default 1e-8.

Value

A single non-negative number: the maximum absolute stationarity violation. Values near zero certify optimality.

See Also

glasso_fit()

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 4), ncol = 4)
fit <- glasso_fit(cov(x), rho = 0.05)
glasso_kkt(fit)
```

---

glasso_path *Fit a graphical lasso over a path of penalties*

Description

Runs glasso_fit() at every penalty in rho, warm-starting each fit from the previous one. This is the kernel behind bootstrap and resampling workflows that need many refits of the same covariance.

Usage

```
glasso_path(
  S,
  rho,
  penalize_diagonal = FALSE,
  max_outer = 10000,
  tol_outer = 1e-04,
  max_inner = 10000,
  tol_inner = 1e-04
)
```

Arguments

S	Covariance or correlation matrix (p x p, symmetric, finite).
rho	Numeric vector of non-negative penalties, used in the order supplied.
penalize_diagonal	Logical. Penalise the diagonal of Theta as well? Default FALSE, matching the usual EBIC-glasso convention.
max_outer, max_inner	Outer (column-sweep) and inner (coordinate-descent) iteration caps, as in glasso_fit().
tol_outer, tol_inner	Convergence tolerances. The defaults are the looser path tolerances (1e-4), matching glasso::glassopath(): path scanning selects among well-separated models, so machine precision at every rung buys nothing. Refit the selected penalty with glasso_fit() when the returned matrix itself must be at full precision.

Value

An object of class glasso_path_result: a list with w and wi, each a p x p x length(rho) array indexed in the order rho was supplied, plus the rho vector itself. Use as.data.frame() for a tidy one-row-per-edge-per-penalty table.

See Also

glasso_fit()

Examples

```
set.seed(1)
z <- rnorm(200)
x <- cbind(A = z + rnorm(200, 0, 0.5), B = z + rnorm(200, 0, 0.5),
           C = rnorm(200), D = rnorm(200))
path <- glasso_path(cov(x), rho = c(0.01, 0.05, 0.2))
path
as.data.frame(path)
```

list_estimators *Registered idiographic estimators*

Description

idiographic uses a small registry to give estimators and workflows one stable dispatch interface. Package methods are registered lazily, so the registry does not depend on source-file load order. Third-party methods can register either a function or the name of a function available in the package namespace or calling environment.

Usage

```
list_estimators(kind = NULL)
```

Arguments

kind	Optional character vector selecting "estimator" and/or "workflow" registrations.

Value

list_estimators() returns one row per registration.

Examples

```
list_estimators()
list_estimators("workflow")
```

matrices *Print model matrices for idiographic results*

Description

matrices() is the matrix-oriented companion to summary() and edges(). It returns the core estimated matrices and, by default, prints each one compactly with rounding, so users can inspect coefficients without digging through object internals.

Usage

```
matrices(x, ...)

## Default S3 method:
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'cograph_network'
matrices(x, digits = 3, ..., print = TRUE)
```

```
## S3 method for class 'netobject'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'netobject_group'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'gvar_result'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'var_result'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'net_mlvar'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'net_usem'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'net_gimme'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'preprocess_result'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'rolling_var_result'
matrices(x, fit = 1L, digits = 3, ..., print = TRUE)

## S3 method for class 'rolling_gvar_result'
matrices(x, fit = 1L, digits = 3, ..., print = TRUE)

## S3 method for class 'stability_result'
matrices(x, digits = 3, ..., print = TRUE)

## S3 method for class 'model_comparison'
matrices(x, fit = 1L, digits = 3, ..., print = TRUE)

## S3 method for class 'var_list'
matrices(x, subject = 1L, digits = 3, ..., print = TRUE)

## S3 method for class 'gvar_list'
matrices(x, subject = 1L, digits = 3, ..., print = TRUE)
```

Arguments

x	An idiographic result or cograph network/group.
...	Passed to methods.
digits	Number of digits used for printing. Default 3.

print	Logical. Print the matrices to the console? Default TRUE, which also returns the list *invisibly*. Use print = FALSE for programmatic extraction: the function itself prints nothing and returns the list visibly (so a bare call at the console still auto-prints the returned value – assign it, or wrap in invisible(), to see nothing at all). print follows ... in every method, so it must be named in full; it can never be matched positionally or by partial name.
fit	Stored fit name or index for result containers that optionally keep fitted models, such as rolling results and model comparisons.
subject	Subject name or index for per-subject VAR/GVAR result lists.

Details

Pass print = FALSE to suppress the console output and get the named list back visibly. That is the form other code should call: extracting matrices inside a loop, a bootstrap, or a dependent package should not write to the console.

Value

A named list of matrices: invisibly when print = TRUE (the default), visibly when print = FALSE.

Examples

```
W <- matrix(c(0, 0.3, -0.2, 0), 2, 2,
            dimnames = list(c("A", "B"), c("A", "B")))
x <- structure(list(weights = W, method = "relative", directed = TRUE),
               class = "cograph_network")
matrices(as_netobject(x))

# programmatic extraction: silent, and returned visibly
str(matrices(as_netobject(x), print = FALSE))
```

---

nodes.net_usem *Tidy per-node strength table for any idiographic result*

Description

One row per node per network with strength (sum of absolute incident edge weights) and, for directed networks, out_strength / in_strength (NA for undirected). Self-loops are excluded.

Usage

```
## S3 method for class 'net_usem'
nodes(x, ...)

## S3 method for class 'var_result'
nodes(x, ...)

nodes(x, ...)
```

```
## S3 method for class 'netobject'
nodes(x, ...)

## S3 method for class 'netobject_group'
nodes(x, ...)

## S3 method for class 'gvar_result'
nodes(x, ...)

## S3 method for class 'net_mlvar'
nodes(x, ...)

## S3 method for class 'net_gimme'
nodes(x, ...)

## S3 method for class 'var_list'
nodes(x, ...)

## S3 method for class 'gvar_list'
nodes(x, ...)
```

Arguments

x	A gvar_result, net_mlvar, net_gimme, netobject, or netobject_group.
...	Passed to methods.

Value

A tidy data.frame.

Examples

```
W <- matrix(c(0, 0.3, -0.2, 0), 2, 2,
            dimnames = list(c("A", "B"), c("A", "B")))
x <- structure(list(weights = W, method = "relative", directed = TRUE),
               class = "cograph_network")
nodes(as_netobject(x))
```

---

plot_gimme *Faithful GIMME network plot (the* gimme*-package convention, via cograph)*

Description

Draws a GIMME result the way the gimme package does: a single p-node network where **dashed edges are lag-1 (temporal)** and **solid edges are lag-0 (contemporaneous)**, **edge width is the proportion of subjects** that have the path, **black edges are group-level** paths and grey edges individual-level, and autoregression shows as a dashed self-loop. Rendered with `cograph::splot()`, so a lag and a contemporaneous effect between the same pair are drawn as two parallel edges.

Usage

```
plot_gimme(
  x,
  weight = c("prop", "coef"),
  group_color = "black",
  individual_color = "grey60",
  layout = "circle",
  curvature = 0.25,
  edge_scale = 5,
  ...
)
```

Arguments

`x`	A `net_gimme` object from `fit_gimme()`.
`weight`	`"prop"` (default, proportion of subjects) or `"coef"` (group-average standardized coefficient) for edge width.
`group_color, individual_color`	Edge colours for group- vs individual-level paths. Defaults `"black"` / `"grey60"`.
`layout`	cograph layout passed to `cograph::splot()`. Default `"circle"`, matching gimme.
`curvature`	Edge curvature (separates parallel lag/contemp edges). Default `0.25`.
`edge_scale`	Multiplier mapping weight to drawn line width. Default `5`.
`...`	Further arguments forwarded to `cograph::splot()`.

Value

Invisibly, the mixed `cograph_network` object that was plotted.

See Also

`as_netobject()` for the matrix view.

Examples

```
set.seed(1)
panel <- data.frame(
  id = rep(1:5, each = 30),
  t  = rep(seq_len(30), 5),
```

```
  A  = rnorm(150), B = rnorm(150), C = rnorm(150)
)
gm <- fit_gimme(panel, vars = c("A", "B", "C"), id = "id", time = "t")
plot_gimme(gm)
```

---

plot_idiographic *Plot an idiographic network result*

Description

S3 `plot()` methods that render any idiographic result with `cograph::splot()`. Call `plot(fit)` to draw the full result (every network panel) or pass `layer` to draw a single network – `"temporal"`, `"contemporaneous"`, `"between"` (mlVAR), or `"residual_cov"` (uSEM) – without indexing into the object.

Usage

```
## S3 method for class 'var_result'
plot(x, layer = NULL, mixed = FALSE, ...)

## S3 method for class 'gvar_result'
plot(x, layer = NULL, mixed = FALSE, ...)

## S3 method for class 'var_bayes_result'
plot(x, layer = NULL, mixed = FALSE, ...)

## S3 method for class 'net_mlvar'
plot(x, layer = NULL, mixed = FALSE, ...)

## S3 method for class 'net_usem'
plot(x, layer = NULL, mixed = FALSE, ...)

## S3 method for class 'net_gimme'
plot(x, layer = NULL, weight = c("prop", "coef"), ...)

## S3 method for class 'var_list'
plot(x, subject = 1L, layer = NULL, ...)

## S3 method for class 'gvar_list'
plot(x, subject = 1L, layer = NULL, ...)

## S3 method for class 'rolling_var_result'
plot(x, fit = 1L, layer = NULL, ...)

## S3 method for class 'rolling_gvar_result'
```

```
plot(x, fit = 1L, layer = NULL, ...)

## S3 method for class 'stability_result'
plot(x, layer = NULL, ...)
```

Arguments

x	An idiographic result (var_result, gvar_result, net_mlvar, net_usem, net_gimme, var_list, rolling_var_result, rolling_gvar_result, or stability_result).
layer	Optional network name to draw on its own. NULL (default) draws the whole result. Available names are reported if an unknown one is given.
mixed	If TRUE, draw two layers as a single mixed network via cograph::plot_mixed_network() — the directed layer as curved arrows and the undirected layer as straight edges. For VAR, graphical VAR, and multilevel VAR this combines the directed temporal network with the undirected contemporaneous network; for uSEM it combines the directed contemporaneous paths with the undirected residual covariances. Default FALSE draws one panel per layer.
...	Further arguments forwarded to cograph::splot().
weight	For GIMME: "prop" (proportion of subjects, default) or "coef" (group-average coefficient) for edge width.
subject	For a var_list / gvar_list: the subject name (or index) to draw. Defaults to the first subject.
fit	For rolling results: the stored window fit (name or index) to draw. Requires keep_fits = TRUE at fit time. Defaults to the first window.

Value

Invisibly, the object that was plotted (a cograph/ggplot object).

Examples

```
set.seed(1)
d <- data.frame(id = 1, A = rnorm(80), B = rnorm(80), C = rnorm(80))
fit <- fit_var(d, vars = c("A", "B", "C"), id = "id")
plot(fit)
plot(fit, layer = "temporal")
```

---

predict.idioml_result *Predict from an idiographic ML result*

Description

Predict from an idiographic ML result

Usage

```
## S3 method for class 'idioml_result'
predict(
  object,
  newdata = NULL,
  scope = c("pooled", "individual"),
  model = NULL,
  estimator = NULL,
  type = c("response", "class"),
  ...
)
```

Arguments

object	An idioml_result.
newdata	Optional new data. If NULL, the stored test-set predictions are returned.
scope	"pooled" or "individual".
model	Fitted model name to use. Default is the first fitted model.
estimator	Fitted estimator/backend for model. Default is that model's first fitted estimator.
type	"response" for numeric predictions/probabilities or "class" for classification labels.
...	Ignored.

Value

A data.frame of predictions.

---

preprocess *Preprocess and audit idiographic time-series data*

Description

Builds the same lag-1 design used by fit_graphical_var() and fit_var(), optionally detrends or differences each series, and returns tidy diagnostics for missingness, day-boundary drops, simple linear trends, AR(1) persistence, split-half mean/variance drift, an ADF-style unit-root screen, and zero-variance variables. It makes the modelling input explicit before estimating VAR, graphical VAR, uSEM, GIMME, or mlVAR models; with detrend it also cleans a non-stationary series in place so the flags can be rechecked on the transformed data.

Usage

```
preprocess(
  data,
  vars,
  id = NULL,
  day = NULL,
  beep = NULL,
  scale = TRUE,
  center_within = TRUE,
  detrend = "none",
 checks = c("trend", "high_ar", "unit_root", "mean_shift", "sd_shift", "zero_variance"),
  delete_missings = TRUE,
  min_obs = NULL,
  subject = NULL,
  trend_alpha = 0.05,
  ar_threshold = 0.95,
  mean_shift_threshold = 0.8,
  sd_ratio_threshold = 2,
  unit_root_t_cutoff = -2.86
)
```

Arguments

data	A data.frame or matrix with columns for variables and optional id/day/beep columns.
vars	Character vector of variable names.
id	Character. Name of the person-ID column, or NULL for a single series.
day	Character. Name of the day/session column, or NULL.
beep	Character. Name of the measurement-occasion column, or NULL.
scale	Logical. Whether to standardize variables before lagging. Default TRUE.
center_within	Logical. Whether to centre within person when more than one id is present. Default TRUE.
detrend	How to remove non-stationarity from each series before lagging. Either a single string applied to every variable, or a **named character vector** giving a per-variable method (unlisted variables are left untouched, e.g. c(planning = "difference", value = "linear")). The available methods are: "none" Default; diagnose only, transform nothing. "auto" Detrend only the subject-series that are flagged, leaving the stationary ones untouched: differencing a stochastic trend (unit root or near-unit-root persistence) and linearly detrending a deterministic trend. The "clean whoever needs it" option – no subsetting, one call over everyone. Can be set per variable too. "linear" Replace the series with the residuals of a within-person regression on a linear time index. "difference" First-difference the series within id/day blocks.

The diagnostics and the returned design reflect the detrended series, so the trend and unit-root flags can be rechecked after cleaning.

`checks` Character vector selecting which stationarity checkups to run: any of `"trend"`, `"high_ar"`, `"unit_root"`, `"mean_shift"`, `"sd_shift"`, `"zero_variance"`. Defaults to all of them. Deselecting a check turns its flag off in the report, in the `flag_stationarity_risk` roll-up, and in the `"auto"` detrend decision, so you can screen for only what you care about.

`delete_missings`
Logical. If `TRUE`, `$pairs` contains only complete current/lagged rows; if `FALSE`, first rows of blocks and incomplete rows are retained with `NA` lags, matching `.gvar_tsdata()`. Default `TRUE`.

`min_obs` Integer or `NULL`. Keep only subjects with at least this many observations.

`subject` Optional vector naming the subject(s) to preprocess.

`trend_alpha` Numeric p-value cutoff for the trend flag. Default `0.05`.

`ar_threshold` Numeric absolute AR(1) cutoff for the high-persistence flag. Default `0.95`.

`mean_shift_threshold`
Numeric absolute standardized split-half mean shift cutoff. Default `0.8`.

`sd_ratio_threshold`
Numeric split-half SD ratio cutoff. Default `2`.

`unit_root_t_cutoff`
Numeric cutoff for the ADF-style lag-level t-statistic. Values greater than this cutoff are flagged as unit-root risk. Default `-2.86`, a common large-sample intercept-only screening cutoff.

Value

A `preprocess_result` object with:

`pairs` The ordered current/lagged design table, including `intercept` and `L1_*` columns.

`counts` Per-subject/per-day row and lag-pair counts.

`diagnostics` Per-subject/per-variable missingness, trend, AR(1), split-half drift, unit-root screen, and stationarity risk indicators.

`matrices` The exact `data_c` and `data_l` matrices returned by the VAR/GVAR preprocessing path.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = 1, beep = 1:40,
                A = cumsum(rnorm(40)), B = rnorm(40))
pp <- preprocess(d, vars = c("A", "B"), id = "id", day = "day", beep = "beep")
pp$counts
pp$diagnostics
# Difference the trending series and recheck the flags:
preprocess(d, vars = c("A", "B"), id = "id", day = "day", beep = "beep",
           detrend = "difference")$diagnostics
```

print.forecast_result *Print method for forecast validation results*

Description

Print method for forecast validation results

Usage

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

Arguments

x	A forecast_result object.
...	Ignored.

Value

x, invisibly.

print.glasso_path_result *Print a graphical-lasso path*

Description

Print a graphical-lasso path

Usage

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

Arguments

x	A glasso_path_result from glasso_path().
...	Ignored.

Value

x, invisibly.

print.glasso_result *Print a graphical-lasso fit*

Description

Print a graphical-lasso fit

Usage

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

Arguments

x	A glasso_result from glasso_fit().
digits	Number of digits used for the printed matrices.
...	Ignored.

Value

x, invisibly.

print.gvar_list *Print a list of per-subject graphical VARs*

Description

Print a list of per-subject graphical VARs

Usage

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

Arguments

x	A gvar_list.
...	Unused.

Value

x, invisibly.

print.gvar_result *Print Method for gvar_result*

Description

Print Method for gvar_result

Usage

```
## S3 method for class 'gvar_result'
print(x, digits = 2, ...)
```

Arguments

x	A gvar_result object.
digits	Number of digits used for printed network matrices.
...	Additional arguments (ignored).

Value

The input object, invisibly.

print.idioml_result *Print method for idiographic ML fits*

Description

Print method for idiographic ML fits

Usage

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

Arguments

x	An idioml_result.
...	Ignored.

Value

x, invisibly.

print.model_comparison *Print method for model comparisons*

Description

Print method for model comparisons

Usage

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

Arguments

x	A model_comparison object.
...	Ignored.

Value

x, invisibly.

print.net_gimme *Print Method for net_gimme*

Description

Print Method for net_gimme

Usage

```
## S3 method for class 'net_gimme'
print(x, digits = 2, ...)
```

Arguments

x	A net_gimme object.
digits	Number of digits used for printed network matrices.
...	Additional arguments (ignored).

Value

The input object, invisibly.

Examples

```
set.seed(1)
panel <- data.frame(
  id = rep(1:5, each = 20),
  t  = rep(seq_len(20), 5),
  A  = rnorm(100), B = rnorm(100), C = rnorm(100)
)
gm <- fit_gimme(panel, vars = c("A","B","C"), id = "id", time = "t")
print(gm)
```

`print.net_mlvar` *Print method for net_mlvar*

Description

Print method for net_mlvar

Usage

```
## S3 method for class 'net_mlvar'
print(x, digits = 2, ...)
```

Arguments

`x`	A `net_mlvar` object returned by `fit_mlvar()`.
`digits`	Number of digits used for printed network matrices.
`...`	Unused; present for S3 consistency.

Value

Invisibly returns `x`.

Examples

```
set.seed(1)
n_id <- 8; n_t <- 30; vars <- c("A", "B", "C")
rows <- lapply(seq_len(n_id), function(i) {
  m <- as.data.frame(matrix(rnorm(n_t * 3), ncol = 3))
  names(m) <- vars
  m$id <- i; m$day <- 1L; m$beep <- seq_len(n_t)
  m
})
d <- do.call(rbind, rows)
fit <- fit_mlvar(d, vars = vars, id = "id", day = "day", beep = "beep")
```

```
print(fit)
summary(fit)
```

`print.net_mlvar_bayes` *Print method for net_mlvar_bayes*

Description

Print method for net_mlvar_bayes

Usage

```
## S3 method for class 'net_mlvar_bayes'
print(x, digits = 2, ...)
```

Arguments

`x`	A `net_mlvar_bayes` object from `fit_mlvar_bayes()`.
`digits`	Digits for printed network matrices.
`...`	Unused.

Value

Invisibly returns `x`.

`print.net_usem` *Print method for uSEM fits*

Description

Print method for uSEM fits

Usage

```
## S3 method for class 'net_usem'
print(x, digits = 2, ...)
```

Arguments

`x`	A `net_usem` object.
`digits`	Number of digits used for printed network matrices.
`...`	Ignored.

Value

`x`, invisibly.

print.preprocess_result

Print method for preprocessing results

Description

Print method for preprocessing results

Usage

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

Arguments

x	A preprocess_result object.
...	Ignored.

Value

x, invisibly.

print.rolling_gvar_result

Print method for rolling graphical VAR results

Description

Print method for rolling graphical VAR results

Usage

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

Arguments

x	A rolling_gvar_result object.
...	Ignored.

Value

x, invisibly.

---

print.rolling_var_result

Print method for rolling VAR results

Description

Print method for rolling VAR results

Usage

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

Arguments

x	A rolling_var_result object.
...	Ignored.

Value

x, invisibly.

---

print.stability_result

Print method for stability results

Description

Print method for stability results

Usage

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

Arguments

x	A stability_result object.
...	Ignored.

Value

x, invisibly.

print.var_bayes_result *Print method for var_bayes_result*

Description

Print method for var_bayes_result

Usage

```
## S3 method for class 'var_bayes_result'
print(x, digits = 2, ...)
```

Arguments

x	A var_bayes_result.
digits	Digits for printed networks.
...	Unused.

Value

Invisibly x.

print.var_list *Print a list of per-subject ordinary VARs*

Description

Print a list of per-subject ordinary VARs

Usage

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

Arguments

x	A var_list.
...	Ignored.

Value

x, invisibly.

print.var_result *Print method for ordinary VAR fits*

Description

Print method for ordinary VAR fits

Usage

```
## S3 method for class 'var_result'
print(x, digits = 2, ...)
```

Arguments

x	A var_result object.
digits	Number of digits used for printed network matrices.
...	Ignored.

Value

x, invisibly.

register_estimator *Register an idiographic estimator or workflow*

Description

Register an idiographic estimator or workflow

Usage

```
register_estimator(
  name,
  fit,
  aliases = character(),
  kind = c("estimator", "workflow"),
  description = "",
  result_class = character(),
  equivalence = list(),
  overwrite = FALSE
)
```

Arguments

name	Unique canonical method name.
fit	A function, or a single character string naming a function.
aliases	Optional alternative method names.
kind	Either "estimator" or "workflow".
description	Short human-readable description.
result_class	Optional result classes used to infer equivalence metadata for objects produced by direct calls to the estimator.
equivalence	A named list describing the validation status, reference, scope, tolerance, and notes. Missing fields receive conservative defaults.
overwrite	Logical. Replace an existing registration with the same canonical name. Aliases owned by another method are never overwritten.

Value

register_estimator() invisibly returns the new registration.

Examples

```
demo_fitter <- function(data, ...) structure(list(data = data),
                                             class = "demo_result")
register_estimator("demo", demo_fitter, result_class = "demo_result")
get_estimator("demo")
remove_estimator("demo")
```

---

remove_estimator *Remove a registered estimator*

Description

Remove a registered estimator

Usage

```
remove_estimator(method, missing_ok = FALSE)
```

Arguments

method	A registered method name or alias. Names are case-insensitive; spaces, hyphens, and periods are normalized to underscores.
missing_ok	Logical. If TRUE, silently do nothing when method is not registered.

Value

Invisibly returns the removed registration, or NULL when missing_ok = TRUE and no registration exists.

Examples

```
temp_fitter <- function(data, ...) data
register_estimator("temporary", temp_fitter)
remove_estimator("temporary")
remove_estimator("temporary", missing_ok = TRUE)
```

`srl` *Self-regulated learning intensive longitudinal data (Chapter 20)*

Description

The self-regulated learning (SRL) experience-sampling data used in the Learning Analytics Methods book, Chapter 20 (Vector Autoregression). Each of 36 students reported nine self-regulated-learning indicators once per study occasion for 156 occasions, giving a balanced person-by-time panel suitable for the idiographic time-series methods in this package.

Usage

```
srl
```

Format

A `data.frame` with 5616 rows and 11 columns:

name Student name (36 unique students).

day Within-person occasion index, 1-156.

efficacy Self-efficacy.

value Task value.

planning Planning.

monitoring Monitoring.

effort Effort regulation.

control Control of learning.

help Help seeking.

social Social support.

organizing Organizing.

Details

The columns have already been tidied for modelling: rows are ordered by `name` then `day`, and `day` is a within-person occasion index (1-156) you can pass as the `time` argument to `fit_usem()` and `fit_gimme()`. No further ordering, indexing, or column selection is needed before fitting a model.

Source

Learning Analytics Methods, Book 2, Chapter 20 (VAR): https://lamethods.org/book2/chapters/ch20-var/ch20-var.html. Original data: https://github.com/lamethods/data2/raw/main/srl/srl.RDS, licensed under CC BY-NC-SA 4.0. See the package COPYRIGHTS file for attribution and transformation details.

Examples

```
data(srl)
summary(srl)
head(srl)
```

---

summary.gvar_result *Summary Method for gvar_result*

Description

Summary Method for gvar_result

Usage

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

Arguments

object	A gvar_result object.
...	Additional arguments (ignored).

Value

A tidy data.frame of per-network metrics: one row per network (temporal, contemporaneous) with n_nodes, n_edges, density, mean_abs_weight, n_positive, n_negative. Use edges(object) / coefs(object) for the estimates and nodes(object) for node strengths.

summary.idioml_result *Summary method for idiographic ML fits*

Description

Summary method for idiographic ML fits

Usage

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

Arguments

object	An idioml_result.
...	Ignored.

Value

The metrics table.

summary.net_gimme *Summary Method for net_gimme*

Description

Summary Method for net_gimme

Usage

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

Arguments

object	A net_gimme object.
...	Additional arguments (ignored).

Value

A tidy data.frame of per-network metrics (one row per network: temporal, contemporaneous), with n_edges/density/etc. computed from the proportion-of-subjects networks. Per-subject fit indices are in object$fit; coefs(object) gives the per-person estimates, edges(object) the tidy edge list, and nodes(object) node strengths.

Examples

```
set.seed(1)
panel <- data.frame(
  id = rep(1:5, each = 20),
  t  = rep(seq_len(20), 5),
  A  = rnorm(100), B = rnorm(100), C = rnorm(100)
)
gm <- fit_gimme(panel, vars = c("A","B","C"), id = "id", time = "t")
summary(gm)
```

`summary.net_mlvar` *Summary method for net_mlvar*

Description

Summary method for net_mlvar

Usage

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

Arguments

`object`	A `net_mlvar` object returned by `fit_mlvar()`.
`...`	Unused; present for S3 consistency.

Value

A tidy `data.frame` of per-network metrics (one row per network: `temporal`, `contemporaneous`, `between`). Use `coefs(object)` for the fixed-effect coefficient table, `edges(object)` for the edge list, and `nodes(object)` for node strengths.

Examples

```
set.seed(1)
n_id <- 8; n_t <- 30; vars <- c("A", "B", "C")
rows <- lapply(seq_len(n_id), function(i) {
  m <- as.data.frame(matrix(rnorm(n_t * 3), ncol = 3))
  names(m) <- vars
  m$id <- i; m$day <- 1L; m$beep <- seq_len(n_t)
  m
})
d <- do.call(rbind, rows)
fit <- fit_mlvar(d, vars = vars, id = "id", day = "day", beep = "beep")
```

```
print(fit)
summary(fit)
```

`summary.net_usem` *Summary method for uSEM fits*

Description

Summary method for uSEM fits

Usage

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

Arguments

`object`	A `net_usem` object.
`...`	Ignored.

Value

A tidy per-network metrics `data.frame`.

`summary.preprocess_result`
Summary method for preprocessing results

Description

Compact per-variable roll-up of the diagnostics: one row per variable with its mean spread and the number of subject-series that tripped each stationarity flag. Use `x$diagnostics` for the full per-subject table.

Usage

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

Arguments

`object`	A `preprocess_result` object.
`...`	Ignored.

Value

A tidy per-variable `data.frame`.

---

summary.var_bayes_result *Summary method for var_bayes_result*

Description

Summary method for var_bayes_result

Usage

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

Arguments

`object`	A `var_bayes_result`.
`...`	Unused.

Value

A tidy per-network metrics `data.frame`.

---

summary.var_result *Summary method for ordinary VAR fits*

Description

Summary method for ordinary VAR fits

Usage

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

Arguments

`object`	A `var_result` object.
`...`	Ignored.

Value

A tidy per-network metrics `data.frame`.

`validate_forecast` *Validate one-step forecasts from idiographic VAR models (experimental)*

Description

Experimental. The rolling-origin design follows standard time-series cross-validation practice, but unlike the estimators in this package it has no external reference implementation to validate against, and its interface, defaults, and reported metrics may change in a future release.

Performs rolling-origin one-step prediction from `fit_var()` or `fit_graphical_var()`. Each split fits the estimator on earlier blocks and predicts current variables in the next block from their lag-1 values. Scaling and within-person centring parameters are learned from the training split only, then applied to the assessment split before prediction.

Usage

```
validate_forecast(
  data,
  vars,
  estimator = c("var", "graphical_var"),
  id = NULL,
  day = NULL,
  beep = NULL,
  initial = NULL,
  assess = 1L,
  step = 1L,
  n_splits = NULL,
  block_size = NULL,
  scale = TRUE,
  center_within = TRUE,
  delete_missings = TRUE,
  keep_fits = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` or matrix with columns for variables and optional id/day/beep columns.
`vars`	Character vector of variable names.
`estimator`	`"var"` (default) for `fit_var()` or `"graphical_var"` for `fit_graphical_var()`.
`id`	Character. Name of the person-ID column, or `NULL`.
`day`	Character. Name of the day/session column, or `NULL`.
`beep`	Character. Name of the measurement-occasion column, or `NULL`.

initial	Integer number of ordered blocks in the first training split. Default uses 60 percent of blocks, leaving at least one assessment block.
assess	Integer number of blocks to assess per split. Default 1.
step	Integer number of blocks to advance between splits. Default 1.
n_splits	Optional maximum number of rolling splits.
block_size	Integer or NULL. Consecutive block length used only when neither id nor day is supplied. Defaults to floor(sqrt(nrow(data))).
scale	Logical. Whether to standardize using training-split means and SDs. Default TRUE.
center_within	Logical. Whether to centre within person using training-split person means when more than one id is present. Default TRUE.
delete_missings	Logical. Drop incomplete current/lagged assessment rows. Default TRUE.
keep_fits	Logical. Store fitted split models? Default FALSE.
...	Further arguments passed to the estimator.

Value

A forecast_result with $predictions, $metrics, $splits, $failures, and optionally $fits.

Examples

```
set.seed(1)
d <- data.frame(id = 1, day = rep(1:5, each = 12),
                beep = rep(1:12, 5),
                A = rnorm(60), B = rnorm(60), C = rnorm(60))
fc <- validate_forecast(d, vars = c("A", "B", "C"), id = "id",
                        day = "day", beep = "beep",
                        initial = 3, n_splits = 2, scale = FALSE)
fc$metrics
```

Index

∗ **datasets**
esm_srl, 16
srl, 76

argument_coverage, 4
as.data.frame(), *51*, *54*
as.data.frame.glasso_path_result, 5
as.data.frame.glasso_result, 6
as_netobject, 6, *23*
as_netobject(), *35*, *59*
as_netobject.gvar_result, 7
as_netobject.net_gimme, 7
as_netobject.net_mlvar, 8
as_netobject.var_bayes_result, 9

coefs *(*coefs.idioml_result*)*, 9
coefs(), *4*, *35*, *45*, *47*
coefs.idioml_result, 9
compare_idiographic, 11
compare_idiographic(), *4*

edges *(*edges.net_usem*)*, 12
edges(), *4*, *47*, *55*
edges.net_usem, 12
equivalence, 14
equivalence(), *4*, *15*, *34*
equivalence_table, 15
esm_srl, 16
estimate_stability, 17
estimate_stability(), *4*
estimator_info, 18
extract_edges, 19

fit_gimme, 19
fit_gimme(), *4*, *17*, *35*, *44, 45*, *59*, *76*
fit_graphical_var, 23, *23*
fit_graphical_var(), *4*, *8*, *17*, *27*, *35*, *41, 42*, *45–48*, *52*, *62*, *82*
fit_graphical_var_each, 27
fit_graphical_var_each(), *49*
fit_idiographic, 28
fit_idiographic(), *4*, *14*
fit_idiographic_ml *(*fit_ml*)*, 29
fit_individualized_ml *(*fit_ml*)*, 29
fit_ml, 29
fit_ml(), *4*
fit_mlvar, *23*, 32
fit_mlvar(), *4*, *9*, *17*, *38*, *40*, *45*, *69*, *79*
fit_mlvar_bayes, 36
fit_mlvar_bayes(), *48*, *70*
fit_mlvar_mplus, 39
fit_mlvar_mplus(), *34*, *38*
fit_rolling_graphical_var, 41
fit_rolling_graphical_var(), *4*
fit_rolling_var, 42
fit_rolling_var(), *4*, *41*
fit_usem, 44
fit_usem(), *4*, *17*, *76*
fit_var, 46
fit_var(), *4*, *17*, *42*, *48, 49*, *62*, *82*
fit_var_bayes, 47
fit_var_each, 49

get_estimator, 50
glasso_fit, 50
glasso_fit(), *6*, *53, 54*, *66*
glasso_kkt, 52
glasso_kkt(), *50–52*
glasso_path, 53
glasso_path(), *5*, *51, 52*, *65*

idiographic *(*idiographic-package*)*, 4
idiographic-package, 4

list_estimators, 55

matrices, 55
matrices(), *4*

nodes *(*nodes.net_usem*)*, 57
nodes(), *4*, *47*

nodes.net_usem, 57

plot.gvar_list *(*plot_idiographic*)*, 60
plot.gvar_result *(*plot_idiographic*)*, 60
plot.net_gimme *(*plot_idiographic*)*, 60
plot.net_mlvar *(*plot_idiographic*)*, 60
plot.net_usem *(*plot_idiographic*)*, 60
plot.rolling_gvar_result
*(*plot_idiographic*)*, 60
plot.rolling_var_result
*(*plot_idiographic*)*, 60
plot.stability_result
*(*plot_idiographic*)*, 60
plot.var_bayes_result
*(*plot_idiographic*)*, 60
plot.var_list *(*plot_idiographic*)*, 60
plot.var_result *(*plot_idiographic*)*, 60
plot_gimme, 58
plot_gimme(), *7, 8*, *22*
plot_idiographic, 60
predict.idioml_result, 61
preprocess, 62
preprocess(), *4*
print.forecast_result, 65
print.glasso_path_result, 65
print.glasso_result, 66
print.gvar_list, 66
print.gvar_result, 67
print.idioml_result, 67
print.model_comparison, 68
print.net_gimme, 68
print.net_mlvar, 69
print.net_mlvar_bayes, 70
print.net_usem, 70
print.preprocess_result, 71
print.rolling_gvar_result, 71
print.rolling_var_result, 72
print.stability_result, 72
print.var_bayes_result, 73
print.var_list, 73
print.var_result, 74

register_estimator, 74
remove_estimator, 75

srl, *16*, 76
summary(), *4*, *11*, *47*, *55*
summary.gvar_result, 77
summary.idioml_result, 78
summary.net_gimme, 78
summary.net_mlvar, 79
summary.net_usem, 80
summary.preprocess_result, 80
summary.var_bayes_result, 81
summary.var_result, 81

validate_forecast, 82
validate_forecast(), *4*