Package: silentema (via r-universe)

October 8, 2026

Type Package

Title Dynamic Missingness Graphs and Sensitivity Analysis for EMA Data

Version 1.0.0

Description Tools for diagnosing and correcting informative nonresponse in ecological momentary assessment (EMA) and other experience-sampling designs. Declares the assumed nonresponse mechanism as a dynamic missingness graph built from a taxonomy of seven motifs, following the graphical missing-data framework of Mohan and Pearl (2021) <doi:10.1080/01621459.2021.1874961>; checks by d-separation which within-person and between-person estimands of a two-level vector autoregressive model remain recoverable and by which estimator; tests whether skipped prompts were informative (the silence test and the sensor-gap test, with cluster-robust inference after Cameron and Miller (2015) <doi:10.3368/jhr.50.2.317>); estimates the temporal and contemporaneous networks from answered adjacent prompts with the half-panel jackknife of Dhaene and Jochmans (2015) <doi:10.1093/restud/rdv007>, by inverse-probability weighting on an observed context, and by full-information maximum likelihood with the state-space expectation-maximization (EM) algorithm of Shumway and Stoffer (1982) <doi:10.1111/j.1467-9892.1982.tb00349.x>; profiles the estimates over a self-censoring sensitivity parameter (inverse-probability weighting with a fixed probit selection model whose intercept is calibrated to the response rate); calibrates that parameter from passive sensors, randomized probes, or the post-skip contrast; computes worst-case bounds for person means in the spirit of Manski (2003) <doi:10.1007/b97478>; writes a preregistration-ready missingness declaration; and simulates experience-sampling data under every motif. The methods are described in Yu (2026, manuscript under review); the accompanying materials are archived at <https://osf.io/x6d2t/>.

License GPL (>= 3)

URL https://github.com/hsiutingyu/silentema,

https://hsiutingyu.github.io/silentema/, https://osf.io/x6d2t/

BugReports https://github.com/hsiutingyu/silentema/issues

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.7), stats, graphics, grDevices

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation yes

Author Hsiu-Ting Yu [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3668-8033>)

Maintainer Hsiu-Ting Yu <hsiutingyu@gmail.com>

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

Date/Publication 2026-10-08 16:40:32 UTC

RemoteUrl https://github.com/cran/silentema

RemoteRef HEAD

RemoteSha cfa522410a2a8f281b35cea5f702b06ce5714974

Contents

silentema-package 3
bounds_support 5
break_even 6
calibrate_delta 6
default_params 9
dm_graph 9
dsep 11
fatigue_check 12
fit_fiml 13
fit_ipw 14
fit_pairs 16
fit_tilt 18
loglik_fiml 20
make_pairs 21
missingness_declaration 22
partial_cors 23
plot.delta_calibration 24
plot.dm_graph 24
plot.tilt_profile 25

recoverability 26
sensor_gap_test 27
silence_test 28
simulate_ema 30
simulate_from_fit 33
stationary_cov 34
summary.pairs_fit 34
tilt_profile 36

Index **38**

`silentema-package` *silentema: dynamic missingness graphs and sensitivity analysis for EMA data*

Description

Participants in ecological momentary assessment (EMA) and other experience-sampling studies skip prompts, and the reasons are rarely unrelated to the momentary states being measured. **silentema** provides a graphical language for stating which nonresponse mechanism is assumed, a checker that says which quantities of a two-level vector autoregressive model of order one (VAR(1)) remain estimable under that mechanism and by which estimator, two tests that ask the data whether skipped prompts were informative, estimators for the recoverable case, a sensitivity analysis for self-censoring (the mechanism that estimation alone cannot repair), three calibration designs for the sensitivity parameter, worst-case bounds, a reporting template, and a simulator.

The workflow

The recommended order of analysis is

1. **Declare** the assumed mechanism as a dynamic missingness graph with `dm_graph`, built from the motifs M0 (completely random), M1 (lagged-state dependence), M2 (self-censoring), M3 (burden or fatigue), M4 (person propensity), M5 (context confounding) and M6 (reactivity), with optional sensor, probe and context nodes; inspect it with `plot` and query it with `dsep`.
2. **Check** recoverability with `recoverability`: which estimands (transition kernel, person means, between-person law) are structurally recoverable, and which estimator recovers them.
3. **Test** whether silence was informative with `silence_test` (the state after a skipped prompt) and, when an always-observed channel exists, `sensor_gap_test`; describe response persistence with `fatigue_check`.
4. **Estimate** the within-person VAR(1) from answered adjacent prompts with `fit_pairs` (person intercepts, half-panel jackknife, cluster-robust inference through `summary.pairs_fit`), with covariate adjustment or inverse-probability weighting for an observed context (`fit_ipw`), or by full-information maximum likelihood under missing at random (`fit_fiml`).
5. **Profile** the estimates over a self-censoring sensitivity value with `fit_tilt` and `tilt_profile`, and summarize the profile with `break_even` (sign changes, significance changes, identified sets and bands).

6. **Calibrate** the sensitivity value from a passive sensor, randomized probes or the post-skip contrast with `calibrate_delta`; compare with the worst-case bounds of `bounds_support`.
7. **Report** with `missingness_declaration`, which writes a Markdown declaration for preregistrations and papers.

Simulation and design

`simulate_ema` generates two-level VAR(1) data under any combination of motifs with a target response rate, optional sensor, probe, context and day structure; `simulate_from_fit` simulates from a fitted (tilted) model; `default_params`, `stationary_cov` and `partial_cors` give the population parameters used in the accompanying article and the derived quantities.

Data format

All estimators take a long data frame with one row per *scheduled* prompt, answered or not: a person identifier (`id`), an integer prompt index that increases by one from one scheduled prompt to the next (`time`), a response indicator (`R`, 1 = answered), the state columns (`NA` at skipped prompts), and optionally a `day` column (pairs are then formed within days only), a sensor column, a probe column and context columns. See `make_pairs`.

Author(s)

Maintainer: Hsiu-Ting Yu <hsiutingyu@gmail.com> (ORCID) [copyright holder]

References

Yu, H.-T. (2026). What skipped prompts hide: Detecting, diagnosing, and correcting informative nonresponse in ecological momentary assessment. Manuscript under review. Materials: https://osf.io/x6d2t/.

Mohan, K., & Pearl, J. (2021). Graphical models for processing missing data. *Journal of the American Statistical Association, 116*, 1023-1037. doi:10.1080/01621459.2021.1874961

Dhaene, G., & Jochmans, K. (2015). Split-panel jackknife estimation of fixed-effect models. *The Review of Economic Studies, 82*, 991-1030. doi:10.1093/restud/rdv007

Shumway, R. H., & Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis, 3*, 253-264. doi:10.1111/j.1467-9892.1982.tb00349.x

See Also

The vignettes: `vignette("silentema-workflow")` for a start-to-finish analysis, `vignette("dm-graphs")` for the motif taxonomy and recoverability, `vignette("testing-informativeness")` for the tests, `vignette("sensitivity-analysis")` for tilting and calibration, and `vignette("simulation-and-design")` for the simulator.

`bounds_support` *Worst-case (support) bounds for person means under arbitrary nonresponse*

Description

Worst-case bounds in the spirit of Manski (2003): every skipped prompt could have taken any value on the response scale `[L, U]`. The width, $(1 - \pi_i)(U - L)$, depends only on the response rate and does not shrink with autocorrelation, in contrast to the tilt-bounded identified sets of `break_even`.

Usage

```
bounds_support(data, vars, L, U, id = "id", R = "R")
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`L, U`	Lower and upper limits of the response scale (scalars or one value per variable).
`id`	Name of the person column.
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.

Value

A data frame with one row per person and variable and the columns `id`, `variable`, `response_rate`, `mean_observed` (`NA` for a person without any answered prompt, whose bounds are then the whole scale), `lower` and `upper`.

References

Manski, C. F. (2003). *Partial identification of probability distributions*. Springer. doi:10.1007/b97478

Examples

```
sim <- simulate_ema(N = 5, n_prompts = 20, motifs = "M2", seed = 1)
b <- bounds_support(sim$data, sim$vars, L = -3, U = 8)
b[b$variable == "NegA", ]
# the width is (1 - response rate) x (U - L)
with(b, all.equal(upper - lower, (1 - response_rate) * 11))
```

break_even — *Break-even sensitivity values and identified sets from a tilt profile*

Description

For each lagged coefficient reports the smallest $|\delta|$ on the grid at which the sign of the estimate changes or the confidence interval starts (or stops) covering zero, and the identified set (range of the estimate) over $|\delta| \leq \delta_{max}$.

Usage

```
break_even(profile, delta_max = max(abs(profile$delta_grid)))
```

Arguments

profile	A tilt_profile.
delta_max	Half-width of the plausible sensitivity interval.

Value

A data frame with one row per lagged coefficient and the columns coef, estimate_0 (the estimate at $\delta = 0$), significant_0 (whether its confidence interval excludes zero there), sign_flip_delta and significance_flip_delta (the grid values nearest zero at which the sign and the significance status change; NA = never on the grid), set_lower and set_upper (the identified set: the range of the estimate over $|\delta| \leq \delta_{max}$) and band_lower and band_upper (the band: the range of the confidence limits over the same interval). A significance change is a dichotomous criterion; the sign change and the band are the primary quantities.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, seed = 1)
prof <- tilt_profile(sim$data, sim$vars, delta_grid = c(-1, -0.5, 0, 0.5))
be <- break_even(prof, delta_max = 1)
be[be$coef == "NegA<-NegA", ]
# a narrower plausible interval gives a narrower identified set
break_even(prof, delta_max = 0.5)[1, c("set_lower", "set_upper")]
```

calibrate_delta — *Calibrate the self-censoring sensitivity parameter from a design feature*

Description

Maps an observed statistic that is sensitive to self-censoring onto the tilt profile and returns the value of the sensitivity parameter at which the fitted model reproduces it. Three statistics are supported: the sensor gap (the coefficient of R_t in `sensor_gap_test`; needs an always-observed sensor), the probe contrast (the within-person difference between the states reported at randomized forced-response probes and at ordinary answered prompts; needs probes), and the post-skip contrast of the `silence_test` (needs no extra channel but relies entirely on the functional form). All three are parametric calibrations, not identification results (Yu, 2026).

Usage

```
calibrate_delta(
  profile,
  data,
  method = c("sensor", "probe", "postskip"),
  sensor = "S",
  probe = "Z",
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  n_sim = 20L,
  burden = c("none", "fit"),
  kappa_grid = c(0, 0.5, 1, 1.5, 2.5),
  seed = NULL,
  level = 0.95,
  boot = 0L,
  boot_unit = c("person", "day"),
  boot_grid = NULL
)
```

Arguments

`profile`	A `tilt_profile`.
`data`	The data used for the profile.
`method`	One of `"sensor"`, `"probe"`, `"postskip"`.
`sensor`, `probe`	Names of the sensor and probe columns.
`id`, `time`, `day`, `R`	Column names as in `make_pairs`.
`n_sim`	Number of simulated data sets per grid value (and per burden value) for `method = "postskip"`; the Monte Carlo error of the model-implied curve is roughly the standard error of the post-skip coefficient divided by `sqrt(n_sim)`.
`burden`	For `method = "postskip"`: `"none"` simulates from the fitted self-censoring model alone; `"fit"` adds a burden term calibrated to the observed response persistence (use when M3 is declared).
`kappa_grid`	Grid of burden values (probit drop in the response propensity after a skipped prompt) searched by `burden = "fit"`.

seed	Optional seed for the post-skip simulation and the bootstrap; the caller's random-number state is restored afterwards. NULL uses (and advances) the current stream.
level	Confidence level for the interval obtained by inverting the observed statistic's confidence limits through the predicted curve (the uncertainty of the curve itself is ignored; Yu, 2026, reports the empirical coverage of this interval).
boot	Number of cluster-bootstrap resamples for a percentile interval that refits the profile in every resample (0 = none).
boot_unit	Resampling unit: persons, or days (for single-person data).
boot_grid	Optional grid for the bootstrap profiles; by default seven points around the point estimate.

Details

The post-skip calibration compares the observed post-skip contrast with the contrast implied by data simulated from the fitted tilt model at every grid value. When burden (motif M3) is declared together with self-censoring, the post-skip contrast also carries the collider contribution of $R_t \rightarrow R_{t+1} \leftarrow X_{t+1}$, which works against the self-censoring signal; with `burden = "fit"` the simulated model includes a burden term whose size is calibrated, at every grid value, so that the simulated response persistence (the response rate after an answered minus after a skipped prompt) matches the observed persistence. Without a burden term the model-implied curve is the M2-only curve, which understates $|\delta|$ when burden is present.

Value

An object of class `"delta_calibration"`: a list with `method`, `delta_hat` (the calibrated value; `NA` when the model-implied curve does not cross the observed value on the grid), `interval` (the `level` interval obtained by inverting the confidence limits of the observed statistic; an endpoint outside the range of the curve leaves that side open, `-Inf` or `Inf`), `n_crossings` (when the curve crosses more than once the crossing nearest zero is used), `observed` and `se` (the observed statistic and its standard error), `curve` (a data frame of the grid values and the model-implied statistics, with the calibrated burden value `kappa_hat` per grid point when `burden = "fit"` and a `note` column when a grid point could not be evaluated), `variable`, `level`, `burden`, `persistence_observed`, and, when `boot > 0`, `boot` (the bootstrap replicates of `delta_hat`), `boot_se` and `boot_interval` (percentile interval). Methods: `print` and `plot`.

See Also

`tilt_profile`, `sensor_gap_test`, `silence_test`, `simulate_from_fit`

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, sensor_cor = 0.6,
                    p_probe = 0.05, seed = 3)
vars <- sim$vars
prof <- tilt_profile(sim$data, vars, delta_grid = c(-1.5, -1, -0.5, 0), probe = "Z")
cs <- calibrate_delta(prof, sim$data, method = "sensor")
cs
cs$curve
calibrate_delta(prof, sim$data, method = "probe")
```

```
# the post-skip calibration simulates from every fitted model
calibrate_delta(prof, sim$data, method = "postskip", n_sim = 5, seed = 1)
```

`default_params` *Default population parameters used in the simulation studies of Yu (2026)*

Description

A four-variable within-person system with autoregressive coefficients between .30 and .45, a few cross-lagged effects, an innovation covariance with a sparse contemporaneous network, and stationary variances equal to one.

Usage

```
default_params()
```

Value

A list with `Phi` (4 x 4 lagged coefficients), `Psi` (innovation covariance, scaled so that the stationary variances equal one), `mu` (population means), `Sigma_mu` (between-person covariance of the person means) and `vars`, the variable names `c("NegA", "PosA", "Stress", "Fatigue")`.

Examples

```
p <- default_params()
p$Phi
# contemporaneous network: NegA--PosA -.30, NegA--Stress .30, Stress--Fatigue .20
round(partial_cors(p$Psi), 2)
```

`dm_graph` *Declare a dynamic missingness graph (dm-graph)*

Description

A dm-graph is a directed acyclic graph over the unrolled within-person process of an experience-sampling study: person-level latent nodes (`eta`, the person's typical state; `zeta`, a person-level response propensity), momentary states `X_t`, response indicators `R_t`, and optional context (`C_t`), sensor (`S_t`) and probe (`Z_t`) nodes. The structural edges `X_{t-1} -> X_t` and `eta -> X_t` are always present. The missingness mechanism is declared as a set of motifs, each of which adds edges into `R_t`.

Usage

```
dm_graph(
  motifs = "M0",
  context = c("none", "latent", "observed"),
  context_persistent = FALSE,
  sensor = FALSE,
  probe = FALSE,
  window = 5L
)
```

Arguments

motifs	Character vector of motif codes. "M0" (no edges into R), "M1" (X_{t-1} -> R_t, lagged-state dependence), "M2" (X_t -> R_t, self-censoring), "M3" (R_{t-1} -> R_t, burden or fatigue), "M4" (zeta -> R_t with zeta associated with eta, person propensity), "M5" (C_t -> X_t and C_t -> R_t, context confounding), "M6" (R_{t-1} -> X_t, reactivity). Motifs may be combined.
context	One of "none", "latent", "observed". Relevant for "M5"; "observed" means the context variable is recorded at every prompt (for example by a phone sensor).
context_persistent	Logical; if TRUE the context process has edges C_{t-1} -> C_t.
sensor	Logical; add an always-observed passive-sensor node S_t with X_t -> S_t.
probe	Logical; add a randomized probe indicator Z_t with Z_t -> R_t (a probe forces a response).
window	Integer number of prompts in the unrolled graph (at least 4).

Value

An object of class "dm_graph": a list with elements nodes (character vector of node names), edges (two-column character matrix from, to), motifs, context, context_persistent, sensor, probe, window, latent (names of the latent nodes) and observed_always (names of the nodes observed at every prompt). It has print and plot methods.

See Also

recoverability, dsep, plot.dm_graph, simulate_ema (which generates data under the same motifs)

Examples

```
g <- dm_graph(c("M1", "M4"))
g
recoverability(g)
g2 <- dm_graph(c("M2", "M3"), sensor = TRUE)
recoverability(g2)
# an observed context adds M5 and observed context nodes
g5 <- dm_graph("M1", context = "observed", context_persistent = TRUE)
g5$edges[g5$edges[, "from"] == "C2", ]
```

dsep *d-separation in a dm-graph*

Description

Tests whether two node sets are d-separated given a conditioning set, using the reachability ("Bayes-ball") algorithm (Koller & Friedman, 2009, Algorithm 3.1).

Usage

```
dsep(g, x, y, z = character(0))
```

Arguments

g	A dm_graph (or any list with an edges matrix with columns from, to).
x, y	Character vectors of node names (window indices: "X3" is the state at the third prompt of the window; recoverability uses the third prompt as t).
z	Character vector of conditioning nodes (may be empty).

Value

A single logical value: TRUE if every node in x is d-separated from every node in y given z, FALSE otherwise.

References

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Shachter, R. D. (1998). Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 480-487). Morgan Kaufmann.

Examples

```
g <- dm_graph("M1")
dsep(g, "R3", "X3", c("X2", "eta"))      # TRUE: the kernel is recoverable
g2 <- dm_graph("M2")
dsep(g2, "R3", "X3", c("X2", "eta"))     # FALSE: self-censoring
# any edge list works, not only dm-graphs: a collider A -> B <- C
coll <- list(nodes = c("A", "B", "C"),
             edges = cbind(from = c("A", "C"), to = c("B", "B")))
dsep(coll, "A", "C")          # TRUE
dsep(coll, "A", "C", "B")     # FALSE: conditioning on the collider opens the path
```

`fatigue_check` *Descriptive check for burden or fatigue in the response sequence*

Description

Reports the response rate after an answered and after a skipped prompt (pooled over consecutive prompts, within day when `day` is given, and as the mean over persons of the person-specific rates) and the response rate by study quarter. Serial dependence in R is expected under M3, but some serial dependence also arises under M1, M2 and M4 through the autocorrelated states and the person propensities, so this is a descriptive check, not a test of M3 against the other motifs.

Usage

```
fatigue_check(data, id = "id", time = "time", day = NULL, R = "R")
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`); required, because the check has no state columns from which to infer it.

Value

An object of class `"fatigue_check"`: a list with `after_answered` and `after_skipped` (pooled response rates after an answered and after a skipped prompt), `difference` (their difference, pooled), `difference_person` (mean of the person-specific differences), `by_quarter` (response rate by study quarter) and `n_transitions`. It has a `print` method.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M3", seed = 1)
fatigue_check(sim$data)
# compare with a mechanism without burden
fatigue_check(simulate_ema(N = 40, n_prompts = 30, motifs = "M0", seed = 1)$data)
```

`fit_fiml` *Full-information maximum likelihood under missing at random (state-space EM)*

Description

Fits the two-level VAR(1) with random person means by maximum likelihood, treating skipped prompts as missing at random. The likelihood is computed by the Kalman filter over the augmented state (X_t, μ_i) and maximized by the EM algorithm of Shumway and Stoffer (1982). This is the likelihood that dynamic structural equation modeling and related software maximize (with a diffuse or informative prior instead of the flat likelihood), and it is the "default pipeline" against which Yu (2026) measures the damage of state-dependent nonresponse.

Usage

```
fit_fiml(
  data,
  vars,
  id = "id",
  time = "time",
  R = "R",
  start = NULL,
  max_iter = 200L,
  tol = 1e-05,
  verbose = FALSE
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`id`, `time`	Names of the person and prompt-index columns.
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.
`start`	Optional list with `Phi`, `Psi`, `mu`, `Sigma_mu`; by default the answered-pairs estimates are used.
`max_iter`, `tol`	EM control.
`verbose`	Print the EM trace.

Details

The prompts of a person are treated as one equally spaced series (the overnight transition is a lag-1 transition, as in the default specification of dynamic structural equation models with equally spaced occasions); there is no day argument. Standard errors are not computed (the estimator serves as the

MAR benchmark in the simulation studies of Yu, 2026). `loglik` holds the log-likelihood evaluated at the parameters entering each EM iteration, so its last element belongs to the penultimate iterate; `loglik_fiml` evaluates the returned estimates.

Value

An object of class `"fiml_fit"`: a list with `Phi`, `Psi`, `pcor`, `mu`, `Sigma_mu`, `Sigma0` (the stationary covariance of the within-person deviations), `loglik` (the trace of log-likelihood values over the EM iterations), `iterations`, `converged`, `N`, `n_prompts` (the number of prompts spanned by the prompt index) and `vars`. It has a `print` method.

References

Shumway, R. H., & Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis, 3*, 253-264. doi:10.1111/j.1467-9892.1982.tb00349.x

Examples

```
sim <- simulate_ema(N = 30, n_prompts = 20, motifs = "M1", seed = 1)
f <- fit_fiml(sim$data, sim$vars)
f
f$converged; length(f$loglik)
# under self-censoring the MAR likelihood is biased like the answered-pairs estimator
sim2 <- simulate_ema(N = 30, n_prompts = 20, motifs = "M2", delta = -1, seed = 1)
c(fiml = fit_fiml(sim2$data, sim2$vars)$Phi[1, 1],
  pairs = fit_pairs(sim2$data, sim2$vars)$Phi[1, 1],
  truth = default_params()$Phi[1, 1])
```

`fit_ipw` *Inverse-probability-weighted within-person VAR(1) for observed context confounding*

Description

For motif M5 with an observed context variable (`C_t -> X_t` and `C_t -> R_t`), complete adjacent pairs are reweighted by the stabilized weight $w_t = P(R_t = 1 \mid x_{t-1}) / P(R_t = 1 \mid c_t, x_{t-1})$ so that the context-marginal transition kernel is recovered (the observed-context case of the recoverability results in Yu, 2026). Both response models are probit regressions fitted on all prompts whose predecessor was answered (the response indicator is always observed), with person-specific intercepts absorbed by including the person's observed response rate as an offset-like covariate when `propensity = "person"`.

Usage

```
fit_ipw(
  data,
  vars,
```

```
  context = "C",
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  propensity = c("common", "person"),
  pairs = NULL
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`context`	Name(s) of the observed context column(s).
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.
`propensity`	`"common"` or `"person"` (adds the person's response rate on other prompts as a covariate of the response model).
`pairs`	Optional precomputed result of `make_pairs`.

Value

An object of class `c("ipw_fit", "pairs_fit")`: a `fit_pairs` object (weighted) with the additional elements `response_model` (the fitted denominator probit model, a `glm`) and `ess` (the effective number of pairs); the weights are in `weights`. The weights are exact only when the context is serially independent; under a persistent context the pair selection through R_{t-1} depends on C_{t-1}, and covariate adjustment (`fit_pairs(covariates = )`) is preferable.

See Also

`fit_pairs` with `covariates` for covariate adjustment, which is preferable under a persistent context.

Examples

```
# the context is recorded in column C
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M5", seed = 1)
f <- fit_ipw(sim$data, sim$vars, context = "C")
f
f$ess / f$n_pairs
summary(f$response_model)$coefficients
```

`fit_pairs` *Within-person VAR(1) estimated from complete adjacent pairs*

Description

Estimates the lagged-coefficient matrix Φ, the innovation covariance Ψ, person intercepts and person means from complete adjacent prompt pairs, with person-specific intercepts (the within estimator). This is the estimator that recovers the transition kernel under the recoverable motifs (M0, M1, M3, M4 and their combinations, and M5 with an observed context; see `recoverability`). Optional weights turn it into the inverse-probability-weighted or tilted estimator. By default the half-panel (split-panel) jackknife of Dhaene and Jochmans (2015) removes the finite-T (Nickell) bias of the within estimator of Φ.

Usage

```
fit_pairs(
  data,
  vars,
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  weights = NULL,
  pairs = NULL,
  min_pairs = 5L,
  bias_correct = TRUE,
  covariates = NULL
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.
`weights`	Optional non-negative weights, one per complete pair (in the order returned by `make_pairs`).
`pairs`	Optional precomputed result of `make_pairs`.
`min_pairs`	Persons with fewer complete pairs are dropped from the between-person summaries (they still contribute to Φ). A person can have at most $T - 1$ pairs, so `min_pairs` must be smaller than the number of prompts per person.

`bias_correct` Logical; apply the half-panel jackknife to Φ, Ψ, the person intercepts and means, and the between-person covariance (all of which carry $O(1/T)$ incidental-parameter bias).

`covariates` Optional names of observed context columns measured at prompt t that enter the within regression as covariates; the returned Φ is then the context-conditional transition kernel (the recoverable estimand under M5 with an observed context), and `B_cov` holds the covariate coefficients. Person means are evaluated at each person's mean covariate level.

Details

Person intercepts are handled by weighted within-person demeaning on the sample of complete pairs (equivalent to person dummies), not by centering on the observed person means as in the two-step `mlVAR` approach; the two coincide for Φ only in large samples. Persons contribute to Φ and Ψ whatever their number of pairs; the between-person summaries use persons with at least `min_pairs` pairs, and the number dropped is reported in `n_dropped`.

Value

An object of class `"pairs_fit"`: a list with `Phi` (lagged coefficients, rows = outcome at t, columns = predictor at $t-1$), `Psi` (innovation covariance), `pcor` (contemporaneous partial correlations), `B_cov` (covariate coefficients, or `NULL`), `intercepts` (person intercepts c_i), `mu_dyn` (dynamics-recovered person means $(I-\Phi)^{-1}c_i$), `mu_obs` (observed person means), `mu` and `Sigma_mu` (person-weighted between-person law based on `mu_dyn`), `mu_obs_person` (person-weighted mean of the observed means), `mu_obs_pooled` (prompt-weighted mean of answered prompts), `Sigma_mu_obs`, the uncorrected (least-squares) versions `Phi_ls`, `Psi_ls`, `mu_ls`, `Sigma_mu_ls`, `vcov_Phi` (cluster-robust covariance of the stacked rows of `Phi`), `n_pairs`, `n_persons` (persons with at least one complete pair), `n_pairs_person`, `n_dropped`, `weights`, `pairs` (the `make_pairs` object), `vars`, `residuals` and `bias_correct`. Methods: `print`, `summary.pairs_fit`, `coef`, `vcov`, `confint`, `nobs`.

References

Dhaene, G., & Jochmans, K. (2015). Split-panel jackknife estimation of fixed-effect models. *The Review of Economic Studies, 82*, 991-1030. doi:10.1093/restud/rdv007

Nickell, S. (1981). Biases in dynamic models with fixed effects. *Econometrica, 49*, 1417-1426. doi:10.2307/1911408

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M1", seed = 1)
f <- fit_pairs(sim$data, sim$vars)
f
summary(f)
coef(f)[1:4]; confint(f)[1:2, ]
# compare with the population values used by the simulator
round(f$Phi - default_params()$Phi, 2)
# covariate adjustment for an observed context (motif M5)
sim5 <- simulate_ema(N = 40, n_prompts = 30, motifs = "M5", seed = 2)
f5 <- fit_pairs(sim5$data, sim5$vars, covariates = "C")
```

```
f5$B_cov
```

`fit_tilt` *Tilted (self-censoring-adjusted) within-person VAR(1) at a fixed value of the sensitivity parameter*

Description

Sensitivity analysis for motif M2 (self-censoring). The selection model $P(R_t = 1 \mid X_t = x) = \Phi_N(\alpha + \delta' x)$ is held at a fixed sensitivity vector δ that is never estimated from the fit. The response intercept is calibrated so that the model-implied response rate given the observed history matches the observed one, and complete pairs are reweighted by $w_t = P(R_t = 1 \mid x_{t-1}) / P(R_t = 1 \mid x_t)$ (inverse-probability weighting with a stabilized numerator). Weighted within-person regression and intercept calibration are iterated to a fixed point; the population parameters are a fixed point when δ is the true selection vector.

Usage

```
fit_tilt(
  data,
  vars,
  delta,
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  propensity = c("common", "person"),
  pairs = NULL,
  probe = NULL,
  max_iter = 200L,
  tol = 1e-05,
  start = NULL,
  damping = 0.5,
  min_pairs = 5L
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`delta`	The sensitivity parameter: a numeric vector (length = number of variables) of selection coefficients in probit units per unit of the state; a scalar is applied to the first variable. Negative values encode "high states are skipped".
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).

R	Name of the response-indicator column (0/1, no NA). If the default name is not among the columns of data, a prompt counts as answered when all vars are non-missing; any other name must exist.
propensity	"common" (one intercept) or "person" (one intercept per person; use when a person-propensity motif M4 is declared together with M2).
pairs	Optional precomputed result of make_pairs.
probe	Optional name of a randomized-probe indicator column; probe prompts are answered by design, receive weight one and are excluded from the intercept calibration.
max_iter, tol	Fixed-point iteration control.
start	Optional pairs_fit (typically the fit at a neighboring sensitivity value) used as the starting point of the iteration.
damping	Step size of the damped fixed-point update (1 = undamped).
min_pairs	Passed to fit_pairs.

Value

An object of class c("tilt_fit", "pairs_fit"): a fit_pairs object with additional elements delta, alpha (calibrated intercept(s)), iterations, converged, weights, max_weight, propensity, and ess, the effective number of complete pairs $(\sum w)^2 / \sum w^2$ (Kish). The weights are not truncated: profiles in which ess falls far below the number of pairs, or in which a few pairs carry very large weights, rest on few observations and should be read with caution. The cluster-robust covariance treats the weights as fixed (the uncertainty of the calibrated intercept and of the parameters entering the weights is not propagated).

See Also

tilt_profile fits a grid of sensitivity values; calibrate_delta chooses one from a design feature.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, seed = 1)
f0 <- fit_pairs(sim$data, sim$vars)      # treats skips as ignorable
f <- fit_tilt(sim$data, sim$vars, delta = -1)
f
f$ess / f$n_pairs                          # effective share of pairs
c(untilted = f0$Phi[1, 1], tilted = f$Phi[1, 1], truth = default_params()$Phi[1, 1])
# person-specific response intercepts (motif M4 declared with M2)
fp <- fit_tilt(sim$data, sim$vars, delta = -1, propensity = "person")
summary(fp$alpha)
```

`loglik_fiml` *Log-likelihood of the two-level VAR(1) at given parameters (MAR)*

Description

Evaluates, without fitting, the likelihood that `fit_fiml` maximizes: the Gaussian likelihood of the answered states under the two-level VAR(1) with random person means, skipped prompts integrated out as missing at random. Useful for comparing fits, for profile likelihoods, and for checking a fit against the trace of the EM iterations.

Usage

```
loglik_fiml(
  data,
  vars,
  Phi,
  Psi,
  mu,
  Sigma_mu,
  Sigma0 = stationary_cov(Phi, Psi),
  id = "id",
  time = "time",
  R = "R"
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`Phi`, `Psi`, `mu`, `Sigma_mu`, `Sigma0`	Parameter values: the $p \times p$ lagged-coefficient matrix, the $p \times p$ innovation covariance, the length-p population mean, the $p \times p$ between-person covariance of the person means, and the $p \times p$ covariance of the within-person deviation at the first prompt (by default the stationary covariance implied by `Phi` and `Psi`).
`id`, `time`	Names of the person and prompt-index columns.
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.

Value

A single number, the log-likelihood of the observed states under the two-level VAR(1) with the given parameters, skipped prompts treated as missing at random.

Examples

```
sim <- simulate_ema(N = 20, n_prompts = 15, seed = 1)
f <- fit_fiml(sim$data, sim$vars)
loglik_fiml(sim$data, sim$vars, f$Phi, f$Psi, f$mu, f$Sigma_mu)
# the likelihood at the population parameters is lower than at the ML estimates
p <- default_params()
loglik_fiml(sim$data, sim$vars, p$Phi, p$Psi, p$mu, p$Sigma_mu)
```

---

make_pairs *Build adjacent prompt pairs from long experience-sampling data*

Description

A complete pair is two adjacent scheduled prompts of one person (within a day, when a day column is given) that were both answered. Complete pairs are the building block of every estimator in the package: `fit_pairs`, `fit_tilt` and `fit_ipw` regress the states at the second prompt on the states at the first. This function forms the pairs once and also collects the prompts whose predecessor was answered, which the response models of the weighted estimators use.

Usage

```
make_pairs(data, vars, id = "id", time = "time", day = NULL, R = "R")
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.

Details

The `time` column must be an integer prompt index that increases by one from one scheduled prompt to the next within a person (and within a day when `day` is given); two rows whose indices differ by more than one are not treated as adjacent. Every scheduled prompt, answered or not, needs a row.

Value

A list with `y` (matrix of states at t), `x` (states at $t-1$), `id`, `time`, `row_t`, `row_tm1` (row indices in `data`) for the complete pairs; `lag_obs`, a list describing every prompt whose predecessor is answered (its row `row_t`, its predecessor's row `row_tm1`, the predecessor's states `x`, its own response indicator `R`, `id` and `time`), which response models on the observed history use; and `vars`, `n_prompts`, `response_rate` and `Rv` (the response indicator in the row order of `data`).

Examples

```
sim <- simulate_ema(N = 10, n_prompts = 12, motifs = "M1", seed = 1)
pr <- make_pairs(sim$data, sim$vars)
nrow(pr$y); pr$response_rate
# pairs are formed within days only when a day column is given
simd <- simulate_ema(N = 10, n_prompts = 12, days = 4, seed = 1)
nrow(make_pairs(simd$data, simd$vars)$y)
nrow(make_pairs(simd$data, simd$vars, day = "day")$y)
```

`missingness_declaration` *Missingness declaration for preregistrations and reports*

Description

Writes a Markdown declaration that records the declared dm-graph, the recoverability verdicts, the tests that were run, the sensitivity interval and its source, and the reported band, in the order that Yu (2026) recommends for preregistrations and reports. Sections for which the corresponding object is supplied are filled from it; the others are left as prompts to complete.

Usage

```
missingness_declaration(
  g,
  rec = NULL,
  silence = NULL,
  sensor_gap = NULL,
  fatigue = NULL,
  profile = NULL,
  calibration = NULL,
  plausible = NULL,
  file = NULL
)
```

Arguments

`g`	A `dm_graph`.
`rec`	Optional `recoverability` report for `g`.
`silence`	Optional `silence_test` result.
`sensor_gap`	Optional `sensor_gap_test` result.
`fatigue`	Optional `fatigue_check` result.
`profile`	Optional `tilt_profile`.
`calibration`	Optional `calibrate_delta` result (or a list of them).
`plausible`	Optional numeric vector of length 2: the plausible interval used for the band.
`file`	Optional path; if `NULL` the declaration is returned as a character vector.

Value

Invisibly, a character vector with one element per line of the Markdown declaration; when `file` is given the lines are also written to that file.

Examples

```
g <- dm_graph(c("M2", "M4"), sensor = TRUE)
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = c("M2", "M4"), sensor_cor = 0.6, seed = 1)
st <- silence_test(sim$data, sim$vars)
prof <- tilt_profile(sim$data, sim$vars, delta_grid = c(-1, -0.5, 0), propensity = "person")
cal <- calibrate_delta(prof, sim$data, method = "sensor")
cat(missingness_declaration(g, silence = st, profile = prof, calibration = cal,
                            plausible = c(-1, 0)), sep = "\n")
# a bare template for a preregistration, written to a file
tf <- tempfile(fileext = ".md")
missingness_declaration(dm_graph(c("M1", "M3")), file = tf)
readLines(tf)[1:12]
```

---

partial_cors *Partial correlations from a covariance matrix*

Description

Inverts the covariance matrix (or takes the precision matrix as given) and standardizes its negative off-diagonal elements. Applied to the innovation covariance of the VAR(1), the result is the contemporaneous network: the partial correlations of the states at the same prompt given the previous prompt and the other states.

Usage

```
partial_cors(S, precision = FALSE)
```

Arguments

`S`	A covariance (or precision, with `precision = TRUE`) matrix.
`precision`	Logical; is `S` already a precision matrix?

Value

The matrix of partial correlations (the contemporaneous network when `S` is an innovation covariance) with unit diagonal.

Examples

```
round(partial_cors(default_params()$Psi), 2)
# from a precision matrix directly
K <- solve(default_params()$Psi)
all.equal(partial_cors(K, precision = TRUE), partial_cors(default_params()$Psi))
```

plot.delta_calibration *Plot the model-implied curve of a calibration*

Description

Draws the model-implied statistic against the sensitivity value (points and line), the observed statistic (solid horizontal line) with its confidence limits (dashed), and the calibrated value (dotted vertical line).

Usage

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

Arguments

x	A delta_calibration.
...	Passed to plot.

Value

Invisibly, the curve data frame (x$curve sorted by the sensitivity value). Called for its side effect, the plot.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, sensor_cor = 0.6, seed = 3)
prof <- tilt_profile(sim$data, sim$vars, delta_grid = c(-1.5, -1, -0.5, 0))
cs <- calibrate_delta(prof, sim$data, method = "sensor")
plot(cs, main = "Sensor calibration")
```

plot.dm_graph *Plot a dm-graph*

Description

Draws the unrolled graph with prompts on the horizontal axis and node types in rows (person-level latents on top, then context, states, sensor, response indicators and probes). Structural edges are gray, edges added by the declared motifs are black and dashed, so the plot remains readable in grayscale. Latent nodes are drawn as open circles.

Usage

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

Arguments

x	A dm_graph.
...	Further arguments passed to plot; main replaces the default title.

Value

Invisibly, a numeric matrix with one row per node and the columns x and y (the plotting coordinates). Called for its side effect, the plot.

Examples

```
plot(dm_graph(c("M2", "M4"), sensor = TRUE))
plot(dm_graph("M5", context = "observed", context_persistent = TRUE))
```

---

plot.tilt_profile *Plot a tilt profile*

Description

Draws the estimates of selected lagged coefficients (with confidence bands) against the sensitivity value, marks the plausible interval when supplied, and marks a calibrated value when supplied.

Usage

```
## S3 method for class 'tilt_profile'
plot(x, coefs = NULL, plausible = NULL, calibrated = NULL, ...)
```

Arguments

x	A tilt_profile.
coefs	Character vector of coefficient names ("Y<-X" form); default: the row of the self-censoring variable.
plausible	Optional numeric vector of length 2 (plausible interval).
calibrated	Optional numeric value(s) to mark.
...	Passed to plot.

Value

Invisibly, the data frame of plotted estimates and limits (the rows of x$Phi for the selected coefficients). Called for its side effect, the plot.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, seed = 1)
prof <- tilt_profile(sim$data, sim$vars, delta_grid = c(-1, -0.5, 0, 0.5))
# 'calibrated' marks a value such as the one returned by calibrate_delta()
plot(prof, plausible = c(-1, 0), calibrated = -0.6)
# cross-lagged effects on stress instead of the row of the self-censoring variable
plot(prof, coefs = c("Stress<-NegA", "Stress<-Fatigue"), main = "Effects on Stress")
```

`recoverability` *Recoverability report for a dm-graph*

Description

Applies the d-separation conditions derived in Yu (2026) to a declared dm-graph and reports, estimand by estimand, whether the estimand is structurally recoverable and by which estimator, and whether the silence test and the sensor-gap test are valid tests of the recoverable class under the declared mechanism. Recoverability is used in the sense of Mohan and Pearl (2021): a consistent estimator exists that uses only the observed part of the data.

Usage

```
recoverability(g, t = 3L)
```

Arguments

`g`	A `dm_graph`.
`t`	Index of the focal prompt inside the window (default: the third; it must satisfy `3 <= t <= window - 1`, so that the prompts $t-2$ to $t+1$ exist).

Details

The within-person conditioning set always contains every person-level latent node (`eta`, and `zeta` when declared), which is what person-specific intercepts (fixed effects) realize in estimation. Observed context nodes are added to the conditioning set for the context-conditional transition kernel. The probe rule (`Z_t = 1` forces `R_t = 1`) is a deterministic relation that d-separation cannot express; it is applied as a separate rule.

Value

A data frame of class `"recoverability"` with one row per estimand and the columns `estimand`, `recoverable` (logical; `NA` for the two test rows, which report expected behavior rather than recoverability), `estimator` (the estimator that recovers the estimand, or the reason it is not recoverable), `condition` (the d-separation statement that was checked), and the attributes `motifs` and `silence_null` (logical: whether the null hypothesis of the silence test is implied by the graph).

References

Mohan, K., & Pearl, J. (2021). Graphical models for processing missing data. *Journal of the American Statistical Association, 116*, 1023-1037. doi:10.1080/01621459.2021.1874961

Yu, H.-T. (2026). What skipped prompts hide: Detecting, diagnosing, and correcting informative nonresponse in ecological momentary assessment. Manuscript under review.

See Also

dm_graph, dsep, missingness_declaration

Examples

```
recoverability(dm_graph("M1"))
recoverability(dm_graph(c("M1", "M3")))
recoverability(dm_graph("M6"))
r <- recoverability(dm_graph("M2", sensor = TRUE))
r$recoverable
attr(r, "silence_null")
```

---

sensor_gap_test *The sensor-gap test: does an always-observed sensor differ at skipped prompts?*

Description

Regresses a passive-sensor channel at prompt t on the state at $t-1$ and the response indicator at t, within person, over all prompts whose predecessor was answered. Under the recoverable motifs the coefficient of R_t is zero, including under M1 + M3 (no collider is conditioned on, because the sensor is always observed). Under self-censoring the gap identifies the direction of the tilt and, through calibrate_delta, its magnitude.

Usage

```
sensor_gap_test(
  data,
  vars,
  sensor = "S",
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  se = NULL,
  B = 500,
  seed = NULL
)
```

Arguments

data	Long data frame with one row per scheduled prompt.
vars	Names of the state columns.
sensor	Name of the sensor column.
id, time	Names of the person and prompt-index columns.
day	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
R	Name of the response-indicator column (0/1, no NA). If the default name is not among the columns of data, a prompt counts as answered when all vars are non-missing; any other name must exist.
se	"cluster" (cluster-robust by person with $t(G-1)$ and $F(q, G-1)$ reference distributions; the default when there are at least 10 persons) or "dayblock" (block bootstrap that resamples person-day blocks with replacement; the default with fewer than 10 persons; requires day).
B	Number of bootstrap resamples for se = "dayblock".
seed	Optional seed for the bootstrap (the caller's random-number state is restored).

Value

A one-row data frame of class c("sensor_gap_test", "data.frame") with the columns coef_R (coefficient of R_t), se, z and p, and the attributes loading (the regression coefficients of the sensor on the state variables, estimated within person from answered prompts), n (the number of prompts used) and se (the method used). It has a print method.

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, sensor_cor = 0.6, seed = 1)
sg <- sensor_gap_test(sim$data, sim$vars, sensor = "S")
sg
attr(sg, "loading")
# under a recoverable mechanism the gap is null
sim1 <- simulate_ema(N = 40, n_prompts = 30, motifs = "M1", sensor_cor = 0.6, seed = 1)
sensor_gap_test(sim1$data, sim1$vars, sensor = "S")
```

---

silence_test *The silence test: does the state after a skipped prompt differ?*

Description

Regresses the state at prompt $t+1$ on the state at $t-1$ and the response indicator at t, within person, using all triples in which prompts $t-1$ and $t+1$ were answered (prompt t may or may not have been). Under every combination of the recoverable motifs (M0, M1, M3, M4) that does not contain both M1 and M3, R_t is conditionally independent of X_{t+1} given X_{t-1} and the person (Yu, 2026), so the population coefficient of R_t is zero when the conditional mean of X_{t+1} given X_{t-1} is linear (exactly under M0, M3 and M4; under M1 the selection on X_t through R_{t+1} makes it slightly

nonlinear, and `poly = 2` adds squared lagged states to absorb the curvature). Under self-censoring (M2), latent context (M5) or reactivity (M6) the coefficient is not zero. When burden (M3) is combined with a state-dependent motif (M1 + M3, M2 + M3), the conditioning on $R_{t+1} = 1$ opens a collider at R_{t+1}, whose parents are R_t (burden) and either X_t (under M1, and X_t drives X_{t+1}) or X_{t+1} itself (under M2): under M1 + M3 the test rejects in large samples although the kernel is recoverable (with a coefficient of the opposite sign to the self-censoring signature), and under M2 + M3 the collider works against the self-censoring signal and the test loses power. The sign of the coefficient on the self-censoring variable is the sign of the tilt: a negative value means that the states hidden by skips were higher than the states that were reported.

Usage

```
silence_test(
  data,
  vars,
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  se = NULL,
  B = 500,
  poly = 1L,
  seed = NULL
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`id`, `time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.
`se`	`"cluster"` (cluster-robust by person with $t(G-1)$ and $F(q, G-1)$ reference distributions; the default when there are at least 10 persons) or `"dayblock"` (block bootstrap that resamples person-day blocks with replacement; the default with fewer than 10 persons; requires `day`).
`B`	Number of bootstrap resamples for `se = "dayblock"`.
`poly`	Degree of the polynomial in the lagged states (1 = linear).
`seed`	Optional seed for the bootstrap (the caller's random-number state is restored).

Value

An object of class `c("silence_test", "data.frame")`: a data frame with one row per state variable and the columns `variable`, `coef_R` (coefficient of R_t), `se`, `z` (the ratio of the coefficient to

its standard error, referred to $t(G-1)$ under `se = "cluster"`) and `p`, and the attributes `joint` (a named vector with the Wald statistic over all variables, its degrees of freedom and its p value), `n_triples`, `n_skipped` (triples with a skipped middle prompt) and `se` (the method used). It has a `print` method.

References

Cameron, A. C., & Miller, D. L. (2015). A practitioner's guide to cluster-robust inference. *Journal of Human Resources, 50*, 317-372. doi:10.3368/jhr.50.2.317

Yu, H.-T. (2026). What skipped prompts hide: Detecting, diagnosing, and correcting informative nonresponse in ecological momentary assessment. Manuscript under review.

See Also

`sensor_gap_test`, `fatigue_check`, `recoverability` (which reports whether the null is expected under a declared graph)

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, seed = 1)
st <- silence_test(sim$data, sim$vars)
st
attr(st, "joint")
sim0 <- simulate_ema(N = 40, n_prompts = 30, motifs = "M1", seed = 1)
silence_test(sim0$data, sim0$vars)
# quadratic terms in the lagged states, and a day structure
simd <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, days = 5, seed = 1)
silence_test(simd$data, simd$vars, day = "day", poly = 2)
```

`simulate_ema` *Simulate experience-sampling data with a declared missingness mechanism*

Description

Generates a two-level VAR(1) process for `N` persons over `n_prompts` prompts and then deletes whole prompts according to one or more missingness motifs. Response probabilities use a probit link. Intercepts are calibrated numerically so that the realized response rate matches `compliance`.

Usage

```
simulate_ema(
  N = 100,
  n_prompts = 56,
  params = default_params(),
  motifs = "M0",
  compliance = 0.75,
  gamma = -0.5,
```

```
  delta = -1,
  kappa_R = 1,
  burden = 0,
  rho_propensity = 0.4,
  sd_propensity = 0.5,
  p_context = 0.3,
  rho_context = 0,
  gamma_C = 0.6,
  kappa_C = 1,
  rho_react = 0.2,
  sensor_cor = NULL,
  p_probe = 0,
  burnin = 50,
  seed = NULL,
  days = NULL
)
```

Arguments

N	Number of persons.
n_prompts	Number of prompts per person (called T in the archived versions 0.2.x of the package).
params	List with Phi, Psi, mu, Sigma_mu (see default_params).
motifs	Character vector of motif codes (see dm_graph).
compliance	Target overall response rate.
gamma	Coefficient vector of X_{t-1} in the response model (motif M1); a scalar is applied to the first variable.
delta	Coefficient vector of X_t in the response model (motif M2); a scalar is applied to the first variable. Negative values mean that high states are skipped. Both gamma and delta act on the raw (uncentered) state, so a self-censoring person with a high typical state also responds less often overall.
kappa_R	Coefficient of R_{t-1} in the response model (M3); the term enters as kappa_R * (R_{t-1} - compliance), so the propensity after an answered prompt exceeds that after a skipped prompt by kappa_R probit units.
burden	Coefficient of a linear time trend in the response model (M3); the term enters as burden * trend, where the trend runs from -0.5 to 0.5 over the burn-in and the recorded prompts together, so a negative value gives declining compliance over the study and the realized change over the recorded prompts is about burden * n_prompts / (n_prompts + burnin) probit units.
rho_propensity	Correlation between the person-level response propensity and the person's mean on the first variable (M4).
sd_propensity	Standard deviation of the person-level propensity (M4), in probit units.
p_context	Probability that the (binary) context C_t = 1 (M5).
rho_context	Persistence of the context: with this probability C_t copies C_{t-1}, otherwise it is drawn afresh with probability p_context; 0 gives serially independent contexts (the marginal rate is p_context either way).

gamma_C	Shift of the state vector when C_t = 1 (M5); a scalar is applied to the first variable.
kappa_C	Coefficient of C_t in the response model (M5).
rho_react	Shift of the state vector at prompt t when the previous prompt was answered (M6, reactivity); the term enters as rho_react * (R_{t-1} - compliance), so the contrast between an answered and a skipped predecessor is rho_react; a scalar applies to the first variable.
sensor_cor	Correlation between an always-observed sensor channel and the within-person fluctuation of the first state variable (its deviation from the person mean, standardized); NULL for no sensor.
p_probe	Probability that a prompt is a randomized probe that forces a response; 0 for no probes.
burnin	Number of burn-in prompts discarded before recording.
seed	Optional integer seed (the caller's random-number state is restored afterwards).
days	Optional number of prompts per day; when given, a day column is added (the response model itself does not use the day).

Value

A list of class "ema_sim" with data (long data frame: id, time, optional day, R, the state columns with NA at skipped prompts, and the optional columns C (context, M5), S (sensor) and Z (probe)), full (the state columns without deletion, with id and time), mu_i (the true person means), alpha0 (the calibrated response intercept), a_i (the person propensities, M4), params, motifs, settings (the design and the response-model arguments) and vars. It has a print method.

See Also

dm_graph for the same motifs as a graph, simulate_from_fit to simulate from a fitted model, default_params.

Examples

```
sim <- simulate_ema(N = 20, n_prompts = 15, motifs = c("M2", "M4"), compliance = 0.7, seed = 1)
sim
head(sim$data)
# the deleted states are kept for checking estimators
head(sim$full)
# a sensor, probes, a persistent observed context and a day structure
sim2 <- simulate_ema(N = 20, n_prompts = 20, motifs = c("M2", "M5"), sensor_cor = 0.6,
                     p_probe = 0.1, rho_context = 0.5, days = 5, seed = 2)
names(sim2$data)
mean(sim2$data$R[sim2$data$Z == 1])   # probes are always answered
```

`simulate_from_fit` *Simulate data from a fitted tilt model*

Description

Generates a data set from the fitted VAR(1) with the fitted person means and (when the fit has person-specific intercepts) response intercepts, both resampled jointly with replacement so that the association between a person's level and propensity is preserved, and the probit self-censoring model at the fit's sensitivity value and calibrated intercept. An optional burden term lowers the propensity after a skipped prompt.

Usage

```
simulate_from_fit(fit, N, n_prompts, days = NULL, kappa_R = 0, seed = NULL)
```

Arguments

`fit`	A `tilt_fit` (or a plain `pairs_fit`, which is treated as a fit at `delta = 0`).
`N`	Number of persons.
`n_prompts`	Number of prompts per person (called `T` in the archived versions 0.2.x of the package).
`days`	Optional number of prompts per day; adds a `day` column and restarts the burden term at the first prompt of each day.
`kappa_R`	Burden term: probit drop in the response propensity at a prompt whose predecessor (within the day) was skipped.
`seed`	Optional seed; the caller's random-number state is restored.

Value

A long data frame with `id`, `time`, optional `day`, `R` (1 = answered) and the state columns (`NA` at skipped prompts), in the format accepted by every estimator of the package.

Examples

```
sim <- simulate_ema(N = 30, n_prompts = 20, motifs = "M2", seed = 1)
f <- fit_tilt(sim$data, sim$vars, delta = -1)
d <- simulate_from_fit(f, N = 30, n_prompts = 20, seed = 2)
mean(d$R)
head(d)
# with a day structure and a burden term
d2 <- simulate_from_fit(f, N = 30, n_prompts = 20, days = 5, kappa_R = 1, seed = 2)
fatigue_check(d2, day = "day")
```

stationary_cov *Stationary covariance of a VAR(1) process*

Description

Solves $\Sigma = \Phi\Sigma\Phi' + \Psi$ by vectorization.

Usage

```
stationary_cov(Phi, Psi)
```

Arguments

Phi	Lagged-coefficient matrix.
Psi	Innovation covariance matrix.

Value

The stationary covariance matrix (same dimension as Phi). An error is raised when Phi is not stable.

Examples

```
p <- default_params()
round(diag(stationary_cov(p$Phi, p$Psi)), 3)
S <- stationary_cov(p$Phi, p$Psi)
all.equal(S, p$Phi %*% S %*% t(p$Phi) + p$Psi)
```

summary.pairs_fit *Standard errors and confidence intervals for the lagged coefficients*

Description

Cluster-robust (by person) standard errors with a $t(G-1)$ reference distribution, G being the number of persons, as in silence_test. With few persons (single-case or small samples) the cluster-robust intervals are unreliable; a bootstrap over days of the estimates (refitting fit_pairs on resampled day blocks) is the alternative, and silence_test implements it for the test.

Usage

```
## S3 method for class 'pairs_fit'
summary(object, level = 0.95, ...)

## S3 method for class 'pairs_fit'
coef(object, ...)

## S3 method for class 'pairs_fit'
vcov(object, ...)

## S3 method for class 'pairs_fit'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'pairs_fit'
nobs(object, ...)
```

Arguments

`object`	A `pairs_fit`.
`level`	Confidence level.
`...`	Ignored.
`parm`	Coefficient names or indices (default: all).

Value

`summary` returns a data frame with one row per lagged coefficient (`"Y<-X"` names the effect of `X` at $t-1$ on `Y` at t) and the columns `coef`, `estimate`, `se`, `lower`, `upper`, `z` (the ratio of the estimate to its standard error, referred to $t(G-1)$) and `p`.

`coef` returns the stacked rows of Φ as a named vector, `vcov` their cluster-robust covariance matrix, `confint` a two-column matrix of limits and `nobs` the number of complete pairs.

References

Cameron, A. C., & Miller, D. L. (2015). A practitioner's guide to cluster-robust inference. *Journal of Human Resources, 50*, 317-372. doi:10.3368/jhr.50.2.317

Examples

```
sim <- simulate_ema(N = 30, n_prompts = 20, seed = 2)
f <- fit_pairs(sim$data, sim$vars)
summary(f)
coef(f)
confint(f, parm = "NegA<-NegA", level = 0.9)
nobs(f)
```

`tilt_profile` *Sensitivity profile over a grid of self-censoring values*

Description

Fits `fit_tilt` over a grid of sensitivity values and collects the lagged coefficients with cluster-robust confidence limits, the contemporaneous partial correlations and the between-person means.

Usage

```
tilt_profile(
  data,
  vars,
  delta_grid = seq(-2, 2, by = 0.5),
  which = 1L,
  id = "id",
  time = "time",
  day = NULL,
  R = "R",
  propensity = c("common", "person"),
  level = 0.95,
  pairs = NULL,
  probe = NULL,
  min_pairs = 5L
)
```

Arguments

`data`	Long data frame with one row per scheduled prompt.
`vars`	Names of the state columns.
`delta_grid`	Numeric vector of values of the sensitivity parameter for the self-censoring variable (the first of `vars` unless `which` says otherwise).
`which`	Index (in `vars`), or name, of the variable that drives self-censoring.
`id, time`	Names of the person and prompt-index columns.
`day`	Optional name of a day column; pairs are formed only within a day (the overnight gap is not a lag-1 transition).
`R`	Name of the response-indicator column (0/1, no `NA`). If the default name is not among the columns of `data`, a prompt counts as answered when all `vars` are non-missing; any other name must exist.
`propensity`	`"common"` (one intercept) or `"person"` (one intercept per person; use when a person-propensity motif M4 is declared together with M2).
`level`	Confidence level for the limits.
`pairs`	Optional precomputed result of `make_pairs`.
`probe`	Optional probe column name passed to `fit_tilt`.
`min_pairs`	Passed to `fit_pairs`.

Value

An object of class "tilt_profile": a list with delta_grid, which, vars, a data frame Phi (one row per grid value and lagged coefficient, with coef, estimate, se, lower, upper, z, p and delta), a data frame pcor (contemporaneous partial correlations by grid value and edge), a matrix mu (between-person means by grid value), fits (the list of fit_tilt objects), level, propensity, converged and ess (one entry per grid value) and n_pairs. Methods: print and plot.

See Also

break_even, calibrate_delta, plot.tilt_profile

Examples

```
sim <- simulate_ema(N = 40, n_prompts = 30, motifs = "M2", delta = -1, seed = 1)
prof <- tilt_profile(sim$data, sim$vars, delta_grid = c(-1.5, -1, -0.5, 0, 0.5))
prof
head(prof$Phi)
prof$mu
break_even(prof, delta_max = 1)
plot(prof, plausible = c(-1.5, 0))
```

Index

bounds_support, *4*, 5
break_even, *3*, *5*, 6, *37*

calibrate_delta, *4*, 6, *19*, *22*, *27*, *37*
coef.pairs_fit *(*summary.pairs_fit*)*, 34
confint.pairs_fit *(*summary.pairs_fit*)*, 34

default_params, *4*, 9, *31, 32*
dm_graph, *3*, 9, *27*, *31, 32*
dsep, *3*, *10*, 11, *27*

fatigue_check, *3*, 12, *22*, *30*
fit_fiml, *3*, 13, *20*
fit_ipw, *3*, 14, *21*
fit_pairs, *3*, *15*, 16, *19*, *21*, *34*, *36*
fit_tilt, *3*, 18, *21*, *36, 37*

loglik_fiml, *14*, 20

make_pairs, *4*, *7*, *15–17*, *19*, 21, *36*
missingness_declaration, *4*, 22, *27*

nobs.pairs_fit *(*summary.pairs_fit*)*, 34

partial_cors, *4*, 23
plot, *8*, *10*, *37*
plot.delta_calibration, 24
plot.dm_graph, *10*, 24
plot.tilt_profile, 25, *37*

recoverability, *3*, *10*, *16*, *22*, 26, *30*

sensor_gap_test, *3*, *7, 8*, *22*, 27, *30*
silence_test, *3*, *7, 8*, *22*, 28, *34*
silentema *(*silentema-package*)*, 3
silentema-package, 3
simulate_ema, *4*, *10*, 30
simulate_from_fit, *4*, *8*, *32*, 33
stationary_cov, *4*, 34
summary.pairs_fit, *3*, *17*, 34

tilt_profile, *3*, *7, 8*, *19*, *22*, 36

vcov.pairs_fit *(*summary.pairs_fit*)*, 34