Package 'idionomics'

May 8, 2026

Title Conduct Idionomic Analyses for Time Series Modeling

Version 0.1.0

Description A toolkit for idionomic science, a research philosophy that places the unit of the ensemble (individual/couple/group) at the center of analysis. Rather than assuming a common distribution, a similar enough process for each unit, and fitting a single model to the whole ensemble, idionomic methods model each unit separately, then aggregate upward if sensible. The group-level picture emerges from individual results, not the other way around, while explicitly evaluating whether aggregation is reasonable given the measured level of heterogeneity of effects. The package is built around intensive longitudinal data where each participant contributes a time series. It provides a pipeline from preprocessing through modeling to group-level summaries. Current functions: data quality screening (i_screener()), within-person standardization (pmstandardize()), linear detrending (i_detrender()), per-subject ARIMAX (AutoRegressive Integrated Moving Average with eXogenous inputs) modeling and meta-analysis (iarimax()), individual p-values (i_pval()), Sign Divergence and Equisyncratic Null tests (sden_test()), and directed loop detection (looping_machine()). Methods are described in Hernandez et al. (2024) <doi:10.1007/978-3-030-77644-2_136-1>, Ciarrochi et al. (2024) <doi:10.1007/s10608-024-10486-w>, and Sahdra et al. (2024) <doi:10.1016/j.jcbs.2024.100728>.

URL https://github.com/cristobalehc/idionomics

BugReports https://github.com/cristobalehc/idionomics/issues

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports broom, dplyr, forcats, forecast, ggplot2, metafor, rlang, stats, tibble, tidyr

Suggests knitr, rmarkdown, testthat (>= 3.2.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Cristóbal Hernández [cre, aut],
Joseph Ciarrochi [aut],
Steven Hayes [aut],
Baljinder Sahdra [aut]

Maintainer Cristóbal Hernández <cristobal.ehc@gmail.com>

Repository CRAN

Date/Publication 2026-04-21 18:42:18 UTC

Contents

`idionomics-package` *idionomics: Conduct Idionomic Analyses for Time Series Modeling*

Description

A toolkit for idionomic science, a research philosophy that places the unit of the ensemble (individual/couple/group) at the center of analysis. Rather than assuming a common distribution, a similar enough process for each unit, and fitting a single model to the whole ensemble, idionomic methods model each unit separately, then aggregate upward if sensible. The group-level picture emerges from individual results, not the other way around, while explicitly evaluating whether aggregation is reasonable given the measured level of heterogeneity of effects. The package is built around intensive longitudinal data where each participant contributes a time series. It provides a pipeline from preprocessing through modeling to group-level summaries. Current functions: data quality screening (i_screener()), within-person standardization (pmstandardize()), linear detrending (i_detrender()), per-subject ARIMAX (AutoRegressive Integrated Moving Average with eXogenous inputs) modeling and meta-analysis (iarimax()), individual p-values (i_pval()), Sign Divergence and Equisyncratic Null tests (sden_test()), and directed loop detection (looping_machine()). Methods are described in Hernandez et al. (2024) doi:10.1007/9783030776442_1361, Ciarrochi et al. (2024) doi:10.1007/s1060802410486w, and Sahdra et al. (2024) doi:10.1016/j.jcbs.2024.100728.

Author(s)

Maintainer: Cristóbal Hernández <cristobal.ehc@gmail.com>

Authors:

- Joseph Ciarrochi <ciarrochij@gmail.com>
- Steven Hayes <stevenchayes@gmail.com>
- Baljinder Sahdra <baljinder.sahdra@acu.edu.au>

References

Hernandez, C., Sahdra, B. K., Ciarrochi, J., & Hayes, S. C. (2024). Idionomic Assessment of Mindfulness. *Handbook of Assessment in Mindfulness Research*, 1–27. doi:10.1007/978303077644-2_1361

Ciarrochi, J., Sahdra, B., Hayes, S. C., Hofmann, S. G., Sanford, B., Stanton, C., Yap, K., Fraser, M. I., Gates, K., & Gloster, A. T. (2024). A Personalised Approach to Identifying Important Determinants of Well-being. *Cognitive Therapy and Research*, *48*(4), 1–22. doi:10.1007/s10608024-10486w

Sahdra, B. K., Ciarrochi, J., Klimczak, K. S., Krafft, J., Hayes, S. C., & Levin, M. (2024). Testing the applicability of idionomic statistics in longitudinal studies: The example of 'doing what matters.' *Journal of Contextual Behavioral Science*, *32*, 100728. doi:10.1016/j.jcbs.2024.100728

See Also

Useful links:

- https://github.com/cristobalehc/idionomics
- Report bugs at https://github.com/cristobalehc/idionomics/issues

---

`iarimax` *Run I-ARIMAX algorithm.*

Description

Run I-ARIMAX algorithm.

Usage

```
iarimax(
  dataframe,
  min_n_subject = 20,
  minvar = 0.01,
  y_series,
  x_series,
  focal_predictor = NULL,
  id_var,
  timevar,
```

```
  fixed_d = NULL,
  correlation_method = "pearson",
  keep_models = FALSE,
  verbose = FALSE
)
```

Arguments

dataframe	A data frame containing time-series data for all subjects, with columns for the outcome, predictor(s), subject ID, and time variable.
min_n_subject	Integer. Subjects with fewer than min_n_subject pairwise-complete observations (across y_series and all x_series) are excluded. The threshold is inclusive (>=). n_valid is extracted directly from the fitted Arima object via stats::nobs() and might be smaller than min_n_subject if auto.arima() applies differencing. Defaults to 20.
minvar	Numeric. Last-resort guard against near-zero variance: subjects whose within-person variance on any series (y_series or x_series) falls below minvar are excluded. Defaults to 0.01. This guard protects against constant or near-constant series that would cause model fitting to fail. For substantive data quality screening (e.g., excluding floor/ceiling responders on raw data), use i_screener() before entering the pipeline.
y_series	A string containing the name of your dependent variable y.
x_series	A character vector containing the name(s) of your predictor variable(s).
focal_predictor	A string with the name of the predictor to use in the meta-analysis. Required when x_series has more than one variable. When x_series is a single variable, defaults to that variable.
id_var	A string containing your id variable.
timevar	A string naming the column used to sort observations into chronological order within each subject. Must be complete (no missing values) and numeric.
fixed_d	Optional non-negative integer. If provided, fixes the differencing order to this value for every subject instead of letting auto.arima() select it. NULL (default) means automatic selection. Fixing d ensures all subjects' coefficients are on the same scale (e.g., d = 0 for levels, d = 1 for changes), which is important for comparability in the meta-analysis. AR (p) and MA (q) orders are always selected automatically per subject because they control error autocorrelation and do not affect the scale of the xreg coefficients.
correlation_method	Select method for raw correlations. Options are: 'spearman', 'pearson' or 'kendall'. Defaults to 'pearson'.
keep_models	If TRUE, will keep original arimax models in a list.
verbose	If TRUE, prints progress messages during filtering and model fitting. Defaults to FALSE.

Value

An S3 object of class `iarimax_results` (a named list) with:

results_df Per-subject data frame with ARIMA orders (`nAR`, `nI`, `nMA`), coefficient estimates (`estimate_<feature>`), standard errors (`std.error_<feature>`), `n_valid` (effective n extracted from the fitted Arima object via `stats::nobs()`), `n_params` (number of model coefficients), and `raw_cor` (pairwise correlation between the focal predictor and `y_series`, not partialled for other predictors in `x_series`). Subjects whose `auto.arima()` call failed appear with `NA` for all model statistics (`nAR`, `nI`, `nMA`, `n_valid`, `n_params`, and all coefficient/SE columns), but retain a valid `raw_cor` if the correlation step succeeded before the model failure. `raw_cor` is `NA` only if `cor.test()` also failed. Their IDs are listed in `case_number_detail$error_arimax_skipped` if auto.arima fails.

meta_analysis A `metafor::rma` object from a random-effects meta-analysis on the focal predictor coefficients, or `NULL` if the meta-analysis failed (e.g. fewer than two valid estimates).

case_number_detail A list with `n_original_df` (total unique subjects in the input), `n_filtered_out` (excluded by variance or n thresholds), `error_arimax_skipped` (character vector of IDs whose `auto.arima()` call failed), and `n_used_iarimax` (subjects with a valid fitted model). The four quantities satisfy: `n_original_df == n_filtered_out + length(error_arimax_skipped) + n_used_iarimax`.

models A named list of raw `Arima` objects (one per subject, `NULL` for failed fits) if `keep_models = TRUE`, otherwise `NULL`.

Attributes: `outcome` (the `y_series` name), `focal_predictor`, `id_var`, `timevar`.

References

Ciarrochi, J., Sahdra, B., Hayes, S. C., Hofmann, S. G., Sanford, B., Stanton, C., Yap, K., Fraser, M. I., Gates, K., & Gloster, A. T. (2024). A Personalised Approach to Identifying Important Determinants of Well-being. *Cognitive Therapy and Research*, *48*(4), 1–22. doi:10.1007/s10608024-10486w

Sahdra, B. K., Ciarrochi, J., Klimczak, K. S., Krafft, J., Hayes, S. C., & Levin, M. (2024). Testing the applicability of idionomic statistics in longitudinal studies: The example of 'doing what matters.' *Journal of Contextual Behavioral Science*, *32*, 100728. doi:10.1016/j.jcbs.2024.100728

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, *27*(3), 1–22. doi:10.18637/jss.v027.i03

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, *36*(3), 1–48. doi:10.18637/jss.v036.i03

See Also

`i_screener()`, `pmstandardize()`, `i_detrender()` for preprocessing; `i_pval()` for per-subject p-values; `sden_test()` for sign divergence / equisyncratic null tests; `looping_machine()` for directed loop detection; `summary.iarimax_results()`, `plot.iarimax_results()` for S3 methods.

Examples

```
# Build a small synthetic panel: 6 subjects, 30 observations each
set.seed(42)
panel <- do.call(rbind, lapply(1:6, function(id) {
  x <- rnorm(30)
  data.frame(
    id   = as.character(id),
    time = seq_len(30),
    x    = x,
    y    = 0.5 * x + rnorm(30),
    stringsAsFactors = FALSE
  )
}))

result <- iarimax(panel,
                  y_series  = "y",
                  x_series  = "x",
                  id_var    = "id",
                  timevar   = "time")

summary(result)
plot(result)
```

`i_detrender` *Linearly detrend idiographic time series by the time variable.*

Description

Linearly detrend idiographic time series by the time variable.

Usage

```
i_detrender(
  df,
  cols,
  id_var,
  timevar,
  min_n_subject = 20,
  minvar = 0.01,
  verbose = FALSE,
  append = TRUE
)
```

Arguments

`df`	A dataframe. Any existing grouping is removed before processing.
`cols`	A non-empty character vector of column names to detrend.

id_var	A string naming the ID variable (one variable only).
timevar	A string naming the time variable used for linear detrending. Must be numeric and complete — no missing values allowed.
min_n_subject	Integer. Subjects with fewer than min_n_subject non-NA observations in a given column receive NA in the detrended output. Mirrors the threshold used in iarimax(). Defaults to 20.
minvar	Numeric. Last-resort guard against near-zero variance: subjects whose pre-detrend or post-detrend variance for a given column falls below minvar receive NA in the detrended output. Defaults to 0.01. This guard protects against constant or near-constant series that would produce meaningless residuals. For substantive data quality screening on raw data, use i_screener() before entering the pipeline.
verbose	If TRUE, prints a description of the detrending rules. Defaults to FALSE.
append	If TRUE (default), returns the original dataframe with new <col>_dt columns appended. If FALSE, returns only the ID column, the time variable, and the new _dt columns.

Details

For each subject x column combination, the following rules are applied in order:

- All values NA: returns NA.
- Fewer than min_n_subject non-NA values: returns NA.
- Pre-detrend variance < minvar: returns NA.
- Otherwise: fits lm(col ~ timevar, na.action = na.exclude) and takes the residuals. If the post-detrend residual variance is also < minvar, returns NA.

The thresholds min_n_subject and minvar share the same defaults as iarimax(), but filtering here is applied **per column independently**: a subject can receive NA in one _dt column while still producing valid residuals in another. This differs from iarimax(), which applies a joint AND filter — a subject is excluded if *any* series fails the thresholds. The per-column design is intentional: it avoids unnecessary data loss and lets you detrend variables in different combinations before deciding which to pass to iarimax(). The thresholds still serve the same protective purpose — preventing data-poor or low-variance subjects from being detrended, which would amplify numerical noise to apparent unit variance after person-mean standardization and allow them to slip through iarimax()'s filters.

Value

A dataframe with new <col>_dt columns containing the within-person linear-detrended residuals.

See Also

i_screener() for upstream data quality screening, pmstandardize() for within-person standardization, iarimax() for per-subject modeling.

Examples

```
local({
# Build a panel with a linear time trend embedded in x
set.seed(2)
panel <- do.call(rbind, lapply(1:3, function(id) {
  data.frame(
    id   = as.character(id),
    time = seq_len(25),
    x    = seq_len(25) + rnorm(25),  # linear trend + noise
    stringsAsFactors = FALSE
  )
}))

# Detrend x within each person (returns original df + x_dt)
result <- i_detrender(panel, cols = "x", id_var = "id", timevar = "time")
head(result)

# Return only id, time, and detrended columns
result_slim <- i_detrender(panel, cols = "x", id_var = "id",
                           timevar = "time", append = FALSE)
head(result_slim)
})
```

---

i_pval *Case by case p-value calculation based on t-distribution.*

Description

Case by case p-value calculation based on t-distribution.

Usage

```
i_pval(iarimax_object, feature = NULL)
```

Arguments

`iarimax_object`	An iarimax object.
`feature`	Feature name to calculate p-value. Defaults to iarimax focal predictor. Use your original name, function will automatically append "estimate_"

Value

Returns an updated version of results_df within the iarimax_object with p-values for the specific feature. p-values use ML-based degrees of freedom (n_valid - n_params) and will differ from `lm()` even for an ARIMA(0,0,0) model because `stats::arima` estimates residual variance via maximum likelihood (dividing by n) rather than the OLS unbiased estimator (dividing by n - k), which affects the standard errors used to compute the t-statistic. For higher-order models, differencing further reduces n_valid and AR/MA parameters increase n_params, widening the gap. If both estimate and SE are exactly zero, the p-value is set to `NA` (not `NaN`).

See Also

iarimax() for the modeling step that produces the input object, sden_test() for group-level inference on the p-values, looping_machine() which calls i_pval() internally.

Examples

```
# Fit I-ARIMAX on a small synthetic panel
set.seed(42)
panel <- do.call(rbind, lapply(1:4, function(id) {
  x <- rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             x = x, y = 0.5 * x + rnorm(30),
             stringsAsFactors = FALSE)
}))

result <- iarimax(panel, y_series = "y", x_series = "x",
                  id_var = "id", timevar = "time",
                  min_n_subject = 20)

# Add per-subject p-values for the focal predictor x
result_pval <- i_pval(result)
result_pval$results_df[, c("id", "estimate_x", "pval_x")]
```

---

i_screener *Screen subjects for data quality before entering the idionomic pipeline.*

Description

Applies per-subject data quality filters on raw (unstandardized) time series designed to be implemented before pmstandardize() or iarimax(). After pmstandardize(), all non-constant series have within-person variance = 1 by construction, making iarimax()'s minvar filter ineffective. Running i_screener() on raw data prevents low-quality subjects from entering the pipeline.

Usage

```
i_screener(
  df,
  cols,
  id_var,
  min_n_subject = 20,
  min_sd = NULL,
  max_mode_pct = NULL,
  filter_type = "joint",
  mode = "filter",
  verbose = FALSE
)
```

Arguments

df	A dataframe. Any existing grouping is removed before processing.
cols	A non-empty character vector of column names to screen.
id_var	A string naming the ID variable (one variable only).
min_n_subject	Integer. Subjects or columns with fewer than min_n_subject non-NA observations in a given column fail this criterion. Defaults to 20, matching iarimax()'s default threshold.
min_sd	Numeric or NULL. If provided, subjects or columns whose within-person SD for a given column is below min_sd (in raw units) fail this criterion. Must be a finite positive number. NULL (default) skips this check. Use this before pmstandardize() to exclude floor/ceiling responders and near-constant series.
max_mode_pct	Numeric in (0, 1] or NULL. If provided, subjects for whom more than max_mode_pct of their non-NA responses fall on the same value fail this criterion. Computed as max(table(x)) / sum(!is.na(x)). Useful for Likert/ordinal EMA data to detect "stuck" or floor/ceiling responders. NULL (default) skips this check.
filter_type	One of "joint" (default) or "per_column". Controls how criteria are applied across multiple columns: • "joint": a subject is excluded or flagged if they fail *any* criterion on *any* variable. Mirrors iarimax()'s AND filter. Recommended when all cols will be analyzed together in the same model. • "per_column": each column is evaluated independently. A subject can pass for one variable and fail for another. Useful for exploratory quality inspection across a broad set of variables. The pass_overall column in reports always uses a joint criterion.
mode	One of "filter" (default), "flag", or "report". Controls the type of output returned: • "filter": returns the dataframe with failing subjects' rows removed ("joint") or their failing column values set to NA ("per_column"). • "flag": returns the original dataframe with a logical pass_overall column added ("joint") or one <col>_pass column per variable ("per_column"). • "report": returns a per-subject summary dataframe with quality metrics and pass/fail indicators for each column and overall (pass_overall).
verbose	If TRUE, prints the criteria applied and subject counts before and after screening. Defaults to FALSE.

Details

For each subject x column combination, three quality metrics are computed:

- n_valid: number of non-NA observations.
- sd: within-person standard deviation (with na.rm = TRUE); NA when fewer than 2 non-NA values exist.
- mode_pct: proportion of non-NA responses equal to the modal value, computed as max(table(x)) / length(x[!is.na(x)]). Uses table() internally, so values are compared as characters; best suited to integer or Likert-scale data where floating-point rounding is not a concern.

Criteria are applied in order: `min_n_subject`, then `min_sd`, then `max_mode_pct`. A subject-column fails as soon as any active criterion is not met.

When `filter_type = "per_column"` and `mode = "filter"`, failing subjects are not removed entirely — their values in the failing column are set to `NA`. Subject-column combinations consisting solely of NA values will fail downstream function's `min_n_subject` filters.

Note that `i_screener()` evaluates each column's non-`NA` count independently (`min_n_subject`), whereas `iarimax()` filters on *pairwise-complete* observations across all series jointly. A subject that passes `i_screener()`'s `min_n_subject` threshold may still be excluded by `iarimax()` if the non-`NA` rows in the outcome and predictor series do not sufficiently overlap.

When `mode = "report"`, `filter_type` does not affect the output shape — the report always contains one row per subject with quality metrics for all columns. `pass_overall` always reflects the joint AND across all columns, matching `filter_type = "joint"` semantics.

The `minvar` filters in `iarimax()` and `i_detrender()` serve a different, complementary role: they are technical last-resort guards that protect each function when called independently. In particular, `i_detrender()`'s post-detrend variance check catches series that pass `i_screener()` on raw data but produce near-zero residuals after removing a near-perfect linear trend — a case `i_screener()` cannot anticipate.

Value

Depends on `mode`:

- `"filter"`: a dataframe with the same columns as input but possibly fewer rows (`"joint"`) or `NA` values in screened columns (`"per_column"`).
- `"flag"`: the original dataframe with `pass_overall` (`"joint"`) or `<col>_pass` columns (`"per_column"`) appended.
- `"report"`: a per-subject summary dataframe with one row per subject and columns grouped by metric: all `<col>_n_valid`, then all `<col>_sd`, then all `<col>_mode_pct`, then all `<col>_pass`, and finally `pass_overall`.

See Also

`pmstandardize()` for the next pipeline step, `i_detrender()` for linear detrending, `iarimax()` for per-subject modeling.

Examples

```
local({
set.seed(42)
panel <- do.call(rbind, lapply(1:4, function(id) {
  data.frame(
    id   = as.character(id),
    time = seq_len(25),
    x    = if (id == 2) rep(3L, 25) else sample(1:7, 25, replace = TRUE),
    y    = sample(1:7, 25, replace = TRUE),
    stringsAsFactors = FALSE
  )
}))
```

```
# Remove subjects failing default min_n_subject = 20 or a minimum SD criterion
result <- i_screener(panel, cols = c("x", "y"), id_var = "id", min_sd = 0.5)

# Inspect which subjects would be removed without committing
flagged <- i_screener(panel, cols = c("x", "y"), id_var = "id",
                      min_sd = 0.5, mode = "flag")
table(flagged$pass_overall)

# Retrieve a per-subject quality summary
report <- i_screener(panel, cols = c("x", "y"), id_var = "id",
                     min_sd = 0.5, mode = "report")
print(report)
})
```

`looping_machine` *Run dynamic looping I-ARIMAX algorithm.*

Description

Fits three I-ARIMAX procedures forming a directed loop (a -> b -> c -> a), applies per-subject p-values, and flags subjects where all three focal coefficients are positive and significant.

Usage

```
looping_machine(
  dataframe,
  a_series,
  b_series,
  c_series,
  id_var,
  timevar,
  covariates = NULL,
  include_third_as_covariate = FALSE,
  min_n_subject = 20,
  minvar = 0.01,
  fixed_d = NULL,
  correlation_method = "pearson",
  alpha = 0.05,
  keep_models = FALSE,
  verbose = FALSE
)
```

Arguments

`dataframe`	A data frame containing time-series data for all subjects, with columns for the loop variables, subject ID, and time variable.
`a_series, b_series, c_series`	Strings naming the three loop variables.

`id_var` String naming the subject ID variable.

`timevar` String naming the time variable (must be complete, no NAs, and numeric see `iarimax()`).

`covariates` Optional character vector of additional predictors added to all three legs. Defaults to NULL.

`include_third_as_covariate` If TRUE, the third loop variable is added as a covariate in each leg. Note: this also applies `minvar` filtering to the third variable, which may silently reduce the subject count. Defaults to FALSE.

`min_n_subject` Minimum pairwise-complete observations per subject per I-ARIMAX run. Defaults to 20.

`minvar` Numeric. Last-resort guard against near-zero variance, passed to each `iarimax()` leg. Defaults to 0.01. For substantive data quality screening on raw data, use `i_screener()` before entering the pipeline.

`fixed_d` Optional non-negative integer passed to `iarimax()` for all three legs. See `iarimax()` for details.

`correlation_method` Raw correlation method: 'pearson', 'spearman', or 'kendall'. Defaults to 'pearson'.

`alpha` Significance threshold for the loop condition; must be in (0, 1). Defaults to 0.05.

`keep_models` If TRUE, raw ARIMAX objects are kept in each leg's result. Defaults to FALSE.

`verbose` If TRUE, prints per-leg progress messages. Defaults to FALSE.

Value

A named list:

loop_df Per-subject dataframe with coefficients, SEs, n_valid, n_params, p-values, and `Loop_positive_directed` (1 = positive loop, 0 = not, NA = ARIMAX failed in at least one leg).

alpha Significance threshold used.

covariates Covariate vector (NULL if none).

include_third_as_covariate Logical; whether the third variable was added as a covariate.

loop_case_detail List with `n_in_loop_df`, `n_complete`, `n_na_indicator` (subjects with a failed model in any leg), and `n_dropped_by_join` (subjects present in at least one leg but dropped from `loop_df` because they were absent from another leg due to filtering or model failure).

iarimax_a_to_b,iarimax_b_to_c,iarimax_c_to_a `iarimax_results` objects for each leg, with `i_pval` already applied.

Statistical notes

Conjunction criterion and Type I error for `Loop_positive_directed`. `Loop_positive_directed` requires all three legs to be significant at `alpha` and all three focal coefficients to be positive. Because the positive directed loop is a one-sided criterion (positive only), each leg's probability of contributing to the loop under the global null is $\alpha/2$ (half the two-tailed rate). The conjunction (intersection) of three events cannot be more probable than any single event, so $P(T_1 \cap T_2 \cap T_3) \leq \alpha/2$

regardless of the correlation between legs. Under independence, the rate is exactly $(\alpha/2)^3$. The criterion can be therefore very conservative.

Comparison with possibly differenced outcomes. Coefficients are partial regression effects conditioned on the ARMA structure chosen by `auto.arima()`. When differencing is selected ($d > 0$), the coefficient describes the relationship on the differenced scale, not the raw level. Check the ARIMA orders in each leg's `results_df` before interpreting `Loop_positive_directed`, and make sure that differenced and undifferenced legs are indeed substantively comparable.

See Also

`iarimax()` and `i_pval()` which are called internally for each leg; `i_screener()`, `pmstandardize()`, `i_detrender()` for preprocessing.

Examples

```
# Build a panel with three correlated processes
set.seed(7)
panel <- do.call(rbind, lapply(1:6, function(id) {
  a_process <- rnorm(30)
  b_process <- 0.4 * a_process + rnorm(30)
  c_process <- 0.4 * b_process + rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             a_process = a_process, b_process = b_process,
             c_process = c_process, stringsAsFactors = FALSE)
}))

loop_result <- looping_machine(panel,
                               a_series = "a_process", b_series = "b_process",
                               c_series = "c_process",
                               id_var   = "id", timevar  = "time")

# Proportion of subjects with a detected positive directed loop
mean(loop_result$loop_df$Loop_positive_directed, na.rm = TRUE)
```

---

plot.iarimax_results *Caterpillar plot method for iarimax_results objects.*

Description

Caterpillar plot method for iarimax_results objects.

Usage

```
## S3 method for class 'iarimax_results'
plot(
  x,
  feature = NULL,
  y_series_name = NULL,
```

```
  x_series_name = NULL,
  alpha_crit_t = 0.05,
  lims = c(-1, 1),
  ...
)
```

Arguments

x	An object of class iarimax_results.
feature	Feature to plot. Defaults to the focal predictor attribute. Use your original variable name — the function appends estimate_ and std.error_ internally.
y_series_name	Optional: substantive name for the outcome variable, used in the plot title.
x_series_name	Optional: substantive name for the predictor variable, used in the plot title.
alpha_crit_t	Critical value for per-subject significance coloring. Defaults to 0.05.
lims	Numeric vector of length 2 setting the effect size axis limits. Defaults to c(-1, 1) given current person mean standardization pipeline.
...	Additional arguments (ignored).

Value

A ggplot2 object. Subjects with NA estimates (e.g., from failed auto.arima fits) are silently excluded from the plot.

See Also

iarimax(), summary.iarimax_results(), i_pval()

Examples

```
set.seed(42)
panel <- do.call(rbind, lapply(1:6, function(id) {
  x <- rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             x = x, y = 0.5 * x + rnorm(30), stringsAsFactors = FALSE)
}))
result <- iarimax(panel, y_series = "y", x_series = "x",
                  id_var = "id", timevar = "time")
plot(result)
plot(result, y_series_name = "Mood", x_series_name = "Stress")
```

`pmstandardize` *Person-level standardize (within-person z-score) time series.*

Description

Computes within-person z-scores for each variable in `cols`: for every subject, subtracts that subject's mean and divides by that subject's SD.

Usage

```
pmstandardize(df, cols, id_var, verbose = FALSE, append = TRUE)
```

Arguments

`df`	A dataframe. Any existing grouping is removed before processing.
`cols`	A non-empty character vector of column names to standardize.
`id_var`	A string naming the ID variable (one variable only).
`verbose`	If `TRUE`, prints a description of the transformation rules (default `FALSE`).
`append`	If `TRUE` (default), returns the original dataframe with the new standardized (`_psd`) columns appended. If `FALSE`, returns only the ID column plus the new `_psd` columns.

Details

For each subject, and each column:

- All values `NA`: returns `NA`.
- Fewer than 2 non-`NA` values: returns `0` (SD undefined; treated as zero deviation).
- Within-person SD near 0 (constant series): returns `0`.
- Otherwise: returns `(x - mean(x)) / sd(x)`, computed with `na.rm = TRUE`.

Zero-variance subjects are retained as all-zero rows rather than dropped. If you plan to pass output to `iarimax()`, those subjects will be filtered out by its `minvar` threshold.

Value

A dataframe with new `<col>_psd` columns containing the within-person z-scores.

See Also

`i_screener()` for upstream data quality screening, `i_detrender()` for the next pipeline step, `iarimax()` for per-subject modeling.

Examples

```
local({
# Build a small panel: 3 subjects, 10 observations each
set.seed(1)
panel <- do.call(rbind, lapply(1:3, function(id) {
  data.frame(
    id   = as.character(id),
    time = seq_len(10),
    x    = rnorm(10, mean = id * 2),  # different person-means
    y    = rnorm(10),
    stringsAsFactors = FALSE
  )
}))

# Standardize x and y within each person (append = TRUE, the default)
result <- pmstandardize(panel, cols = c("x", "y"), id_var = "id")
head(result)  # original columns + x_psd + y_psd

# Return only the ID and standardized columns
result_slim <- pmstandardize(panel, cols = "x", id_var = "id", append = FALSE)
head(result_slim)
})
```

`sden_test` *Run Sign Divergence or Equisyncratic Null tests.*

Description

Run Sign Divergence or Equisyncratic Null tests.

Usage

```
sden_test(
  iarimax_object,
  alpha_arimax = 0.05,
  alpha_binom = NULL,
  test = "auto",
  feature = NULL
)
```

Arguments

`iarimax_object`	Your iarimax object.
`alpha_arimax`	Significance threshold for classifying individual subjects' ARIMAX coefficients as significant (`pval < alpha_arimax`). Also used to derive the null probability for the binomial test: `alpha_arimax` in ENT mode, `alpha_arimax / 2` in SDT mode (unless overridden by `alpha_binom`). Defaults to 0.05. In ENT mode this creates a double role: raising `alpha_arimax` simultaneously increases the

count of significant effects and raises the null expectation against which they are tested. Use `alpha_binom` to decouple the two.

`alpha_binom` Null probability for the binomial test. Defaults to `alpha_arimax` for ENT and `alpha_arimax / 2` for SDT. Supply this to override the default without changing the per-subject significance threshold.

`test` Type of test, default to "auto". "auto", "SDT", and "ENT" are supported. When `"auto"`, the selection between SDT and ENT is based on whether the pooled REMA effect is statistically significant at a **fixed threshold of 0.05**, regardless of `alpha_arimax`. This is intentional: the pooled-effect pivot and the per-subject significance criterion are treated as separate inferential decisions. Note that this threshold also appears in the advisory messages printed under `"SDT"` and `"ENT"`, but does not affect any computation in those modes.

`feature` Feature name to run sden test. Defaults to iarimax focal predictor. Use your original name, function will automatically append "estimate_"

Value

An S3 object of class `sden_results` with two elements: `sden_parameters` (a named list with test type, selection mechanism, REMA beta, p-null, significance counts, and binomial p-value) and `binomial_test` (the `htest` object from `stats::binom.test`). Attributes `focal_predictor`, `id_var`, and `timevar` are inherited from the input `iarimax_object`.

See Also

`iarimax()` for per-subject modeling, `i_pval()` for per-subject p-values (called internally by `sden_test()`), `summary.sden_results()` for printing results.

Examples

```
set.seed(42)
panel <- do.call(rbind, lapply(1:6, function(id) {
  x <- rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             x = x, y = 0.5 * x + rnorm(30),
             stringsAsFactors = FALSE)
}))

result <- iarimax(panel, y_series = "y", x_series = "x",
                  id_var = "id", timevar = "time",
                  min_n_subject = 20)

# Automatic test selection based on pooled REMA effect
sden <- sden_test(result)
summary(sden)

# Force ENT regardless of REMA
sden_ent <- sden_test(result, test = "ENT")
```

summary.iarimax_results *Summary method for iarimax_results objects, based on focal predictor.*

Description

Summary method for iarimax_results objects, based on focal predictor.

Usage

```
## S3 method for class 'iarimax_results'
summary(object, alpha = 0.05, ...)
```

Arguments

object	An object of class iarimax_results.
alpha	Significance threshold used for per-subject positive/negative significant counts. Defaults to 0.05.
...	Additional arguments (ignored).

Value

Invisibly returns object, called for its side effect of printing.

See Also

iarimax(), plot.iarimax_results(), i_pval(), sden_test()

Examples

```
set.seed(42)
panel <- do.call(rbind, lapply(1:6, function(id) {
  x <- rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             x = x, y = 0.5 * x + rnorm(30), stringsAsFactors = FALSE)
}))
result <- iarimax(panel, y_series = "y", x_series = "x",
                  id_var = "id", timevar = "time")
summary(result)
```

summary.sden_results *Summary method for sden_results objects.*

Description

Summary method for sden_results objects.

Usage

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

Arguments

object	An object of class sden_results.
...	Additional arguments (ignored).

Value

Invisibly returns object, called for its side effect of printing.

See Also

sden_test(), iarimax(), i_pval()

Examples

```
set.seed(42)
panel <- do.call(rbind, lapply(1:6, function(id) {
  x <- rnorm(30)
  data.frame(id = as.character(id), time = seq_len(30),
             x = x, y = 0.5 * x + rnorm(30), stringsAsFactors = FALSE)
}))
result <- iarimax(panel, y_series = "y", x_series = "x",
                  id_var = "id", timevar = "time")
sden <- sden_test(result)
summary(sden)
```

Index