

Package ‘convergenceDFM’

June 26, 2026

Type Package

Title Convergence and Dynamic Factor Models

Version 0.3.2

Description Tests convergence in macro-financial panels combining Dynamic Factor Models (DFM) and mean-reverting, discrete-time Ornstein-Uhlenbeck/AR(1) factor processes. Provides: (i) static factor extraction with VAR stability checks, Portmanteau tests and rolling out-of-sample R^2 , in the spirit of Stock and Watson (2002) <doi:10.1198/073500102317351921> and the Generalized Dynamic Factor Model of Forni, Hallin, Lippi and Reichlin (2000) <doi:10.1162/003465300559037>; (ii) cointegration analysis a la Johansen (1988) <doi:10.1016/0165-1889(88)90041-3>; (iii) Bayesian factor-OU/AR(1) estimation with convergence and half-life summaries grounded in Uhlenbeck and Ornstein (1930) <doi:10.1103/PhysRev.36.823> and Vasicek (1977) <doi:10.1016/0304-405X(77)90016-2>, with full Markov chain Monte Carlo convergence diagnostics; (iv) heteroskedasticity-consistent (HC) and, when the suggested 'sandwich' (Zeileis (2004) <doi:10.18637/jss.v011.i10>) and 'lmtest' packages are available, heteroskedasticity- and autocorrelation-consistent (HAC) robust inference, with a self-contained HC fallback; (v) coupling significance tests based on time-shift / block-bootstrap nulls that preserve marginal dynamics while breaking cross-series dependence; and (vi) optional PLS-based factor preselection (Mevik and Wehrens (2007) <doi:10.18637/jss.v018.i02>). Functions emphasize reproducibility (explicit seeds throughout) and clear, publication-ready summaries.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1)

Imports stats, methods, parallel, pls, vars, urca, readxl, dplyr, tidyr, stringr, magrittr, zoo, BayesianDisaggregation (>= 0.2.1)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, cmdstanr, posterior, rstan, sandwich, lmtest

Additional_repositories <https://mc-stan.org/r-packages/>

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.0.0

Author José Mauricio Gómez Julián [aut, cre] (ORCID:
<<https://orcid.org/0009-0000-2412-3150>>)

Maintainer José Mauricio Gómez Julián <isadore.nabi@pm.me>

Repository CRAN

Date/Publication 2026-06-26 08:00:13 UTC

Contents

build_cluster_map	3
choose_var_lag	4
compute_wedge	5
deltaR2_ou	6
diagnose_data	7
estimate_DFM	8
estimate_factor_OU	9
make_X_innovations	10
placebo_values	11
plot_error_correction_panel	12
read_cpi	13
rescue_short_run_channel	14
rotation_null_test	15
row_norm1	16
run_complete_factor_analysis_robust	17
run_convergence_robustness_tests	19
run_rotation_null_on_results	20
select_optimal_components_safe	21
test_cointegration_control	22
test_jackknife_sectors	24
test_leave_cluster_out	25
test_permutation_robustness	26
test_reweighting_robustness	28
to_num_commas	29
visualize_factor_dynamics	30
visualize_factor_dynamics_simple	31
wedge_stationarity	31

Index	33
--------------	-----------

build_cluster_map	<i>Build a fallback sector-to-cluster map</i>
-------------------	---

Description

Constructs a sector-to-cluster partition for `test_leave_cluster_out()` when a genuine input-output (Leontief, "MIP") partition is not yet available. This is an explicitly documented **stop-gap proxy**: the canonical clusters are value chains defined by inter-industry demand linkages, which neither correlation nor organic composition reproduces. Supply the real partition through the `cluster_map` argument of `test_leave_cluster_out()` once the MIP is at hand.

Usage

```
build_cluster_map(
  Y_matrix,
  n_clusters = 3,
  method = c("correlation", "com"),
  com = NULL
)
```

Arguments

<code>Y_matrix</code>	Numeric matrix (periods in rows, sectors in columns).
<code>n_clusters</code>	Integer number of clusters (clamped to $1 \leq n_clusters \leq \text{ncol}(Y_matrix)$). Default 3.
<code>method</code>	Fallback rule, "correlation" (default) or "com".
<code>com</code>	Numeric vector of organic-composition values, one per sector (length $\text{ncol}(Y_matrix)$); required only for <code>method = "com"</code> .

Value

A named integer vector of length $\text{ncol}(Y_matrix)$: the cluster label of each sector, named by sector.

Fallback methods

"correlation" Average-linkage hierarchical clustering on the correlation distance $1 - \rho_{ij}$ between the sectoral series (columns of `Y_matrix`), cut into `n_clusters` groups. Sectors that move together over time are grouped; this is a purely statistical proxy for co-movement, not a demand-linkage cluster.

"com" Quantile binning of a per-sector organic-composition vector `com` into `n_clusters` groups. Motivated by the gravitation reading (sectors of similar organic composition share a profit-rate neighbourhood), but again a one-dimensional proxy, not an input-output partition.

See Also

[test_leave_cluster_out\(\)](#)

choose_var_lag

Select optimal VAR lag order with multiple criteria

Description

Determines the optimal lag order for a Vector Autoregression model using information criteria, stability checks, serial correlation tests, and optional out-of-sample validation.

Usage

```
choose_var_lag(
  F_combined,
  lag.max = 4,
  type = "const",
  p_pref = c("SC(n)", "HQ(n)"),
  alpha = 0.05,
  oos_eval = TRUE,
  oos_start = 0.7,
  verbose = TRUE
)
```

Arguments

<code>F_combined</code>	Numeric matrix (T x K) of factor scores to be modeled.
<code>lag.max</code>	Integer. Maximum lag order to consider. Default is 4.
<code>type</code>	Character string. Type of deterministic terms: "const" (default), "trend", "both", or "none".
<code>p_pref</code>	Character vector. Preferred information criteria for initial selection. Default is <code>c("SC(n)", "HQ(n)")</code> .
<code>alpha</code>	Numeric. Significance level for serial correlation test. Default is 0.05.
<code>oos_eval</code>	Logical. Should out-of-sample evaluation be performed? Default is TRUE.
<code>oos_start</code>	Numeric. Proportion of sample to use for training in OOS validation. Default is 0.7.
<code>verbose</code>	Logical; print progress information. Default TRUE.

Details

The function combines multiple selection criteria: (1) information criteria (AIC, BIC, HQ), (2) VAR stability (eigenvalue modulus < 1), (3) Portmanteau test for serial correlation, and (4) out-of-sample forecast performance. Returns the model that best balances these considerations.

Value

List with components:

- p Selected optimal lag order.
- fit Fitted VAR model object of class varest.
- roots_ok Logical indicating if stability condition is satisfied.
- serial_ok Logical indicating if serial correlation test passed.
- oos_mse Out-of-sample mean squared error (if oos_eval = TRUE).

compute_wedge	<i>Construct the transformation wedge $W = \Phi - V = K * G' - p$</i>
---------------	--

Description

Builds the sector-by-time wedge that isolates the marxian redistribution of surplus from the shared cost price. The wedge cancels the cost price $k = c + v$ that both the production price $\Phi = k + KG'$ and the value $V = k + p$ share, leaving $W_i = K_i G' - p_i$. By construction of the general profit rate $G' = \sum_i p_i / \sum_i K_i$, the wedge sums to zero across sectors each period (redistribution, not creation of value: the marxian invariance).

Usage

```
compute_wedge(K, Gprime, p, tol = 1e-08, check_zero_sum = TRUE)
```

Arguments

- K Numeric matrix [T, S] of total advanced capital per sector.
- Gprime Numeric vector of length T, the general profit rate per period (typically rowSums(p) / rowSums(K)).
- p Numeric matrix [T, S] of sectoral surplus (plusvalia, EBO).
- tol Relative tolerance for the per-period zero-sum check. Default 1e-8.
- check_zero_sum Logical; if TRUE (default) verify $\sum_i W_{i,t} \approx 0$ for every t and error out otherwise.

Value

A list with: W (the [T, S] wedge), row_sums (per-period cross-sectional sums, expected ~0), max_abs_rowsum_rel (the largest relative zero-sum violation), and markup (the profit markup $K_i G'$).

Examples

```
set.seed(1)
Tn <- 20; S <- 5
K <- matrix(runif(Tn * S, 1, 5), Tn, S)
p <- matrix(runif(Tn * S, 0.1, 1), Tn, S)
Gp <- rowSums(p) / rowSums(K)
w <- compute_wedge(K, Gp, p)
max(abs(w$row_sums)) # ~ 0 by construction
```

deltaR2_ou

*Incremental R-squared from X in OU model***Description**

Computes the incremental explanatory power (delta R-squared) contributed by X factors in predicting Y factors, both in-sample and out-of-sample.

Usage

```
deltaR2_ou(
  results_robust,
  lag = 1,
  oos = TRUE,
  seed = 123,
  ridge = 1e-08,
  verbose = TRUE
)
```

Arguments

<code>results_robust</code>	Output from <code>run_complete_factor_analysis_robust()</code>
<code>lag</code>	Number of lags for the model (default: 1)
<code>oos</code>	Logical, perform out-of-sample validation (default: TRUE)
<code>seed</code>	Random seed for reproducibility (default: 123)
<code>ridge</code>	Ridge regularization parameter (default: 1e-08)
<code>verbose</code>	Logical, print progress messages (default: TRUE)

Value

List with components:

`in_sample` Data frame with `R2_full`, `R2_baseline`, and `deltaR2`.

`per_equation` Vector of `deltaR2` for each Y factor equation.

`OOS` List with out-of-sample RMSE and `deltaR2` (if `oos = TRUE`).

diagnose_data	<i>Diagnose and prepare data matrices</i>
---------------	---

Description

Performs data validation, missing value imputation, and variance checks on input matrices to prepare them for factor analysis. Handles dimension compatibility, NA values, and zero-variance columns.

Usage

```
diagnose_data(
  X_matrix,
  Y_matrix,
  verbose = TRUE,
  seed = NULL,
  max_impute_frac = 0.2
)
```

Arguments

<code>X_matrix</code>	Numeric matrix or data frame of X variables (e.g., labour-value price indices).
<code>Y_matrix</code>	Numeric matrix or data frame of Y variables (e.g., market prices / CPI).
<code>verbose</code>	Logical; print diagnostic information. Default TRUE.
<code>seed</code>	Optional integer. If supplied, the random number generator state is set (and restored on exit) before injecting the jitter used to break exactly zero-variance columns, making the operation reproducible. Default NULL (no seeding).
<code>max_impute_frac</code>	Numeric in (0, 1). If the fraction of imputed entries in either matrix exceeds this value a warning is emitted (heavy imputation can distort downstream covariance structure). Default 0.2.

Details

The function:

- Converts to matrix format if needed
- Validates dimensional compatibility
- Imputes missing values via interpolation (using zoo if available)
- Adds minimal, optionally seeded, noise to zero-variance columns
- Reports diagnostic information and the imputation fraction

Imputation by interpolation with boundary carry-forward (and a mean fallback) is a convenience for exploratory work; for inference on series with substantial missingness, prefer an explicit, model-based imputation upstream.

Value

List with components:

`X_matrix` Cleaned and prepared X matrix.

`Y_matrix` Cleaned and prepared Y matrix.

`impute_frac` Named numeric vector with the fraction of imputed entries in X and Y.

estimate_DFM

Estimate Dynamic Factor Model with VAR dynamics

Description

Estimates a Dynamic Factor Model by extracting factors via PLS and modeling their dynamics with a Vector Autoregression. Includes automatic lag selection, robust inference, and optional out-of-sample evaluation.

Usage

```
estimate_DFM(
  factors_data,
  p = 2,
  compute_oos = TRUE,
  hc_type = "HC3",
  hac = FALSE,
  verbose = TRUE
)
```

Arguments

<code>factors_data</code>	List containing PLS-extracted factor scores (<code>scores_X</code> , <code>scores_Y</code>) and related objects.
<code>p</code>	Integer. VAR lag order. If NULL, selected automatically. Default is 2.
<code>compute_oos</code>	Logical. Should out-of-sample diagnostics be computed? Default is TRUE.
<code>hc_type</code>	Character string. Heteroskedasticity-consistent SE type. Default is "HC3".
<code>hac</code>	Logical. If TRUE and sandwich is installed, use a heteroskedasticity- and autocorrelation-consistent (Newey-West) covariance instead of HC. Default FALSE.
<code>verbose</code>	Logical; print progress and diagnostic information. Default TRUE.

Details

This function models the joint dynamics of X and Y factors using a VAR. It performs stability checks, tests for serial correlation, computes robust standard errors, and optionally evaluates forecast performance out-of-sample. Robust covariance uses **sandwich/lmtest** when available and a self-contained HC (HC0-HC4) implementation otherwise. Because the factors are PLS scores (supervised on Y), in-sample fit is optimistic; rely on the out-of-sample diagnostics. See the vignette section "Methodological notes".

Value

List with components:

`var_model` Fitted VAR model (`vars::varest`).

`p_used` VAR lag order used.

`roots`, `is_stable` Companion-root moduli and stability flag.

`half_lives`, `half_life_dominant` Per-root half-lives and the half-life of the largest-modulus (slowest) root, which governs system persistence.

`r2_eq`, `r2_adj_eq`, `r2_global` In-sample R2.

`r2_oos_eq`, `r2_oos_global`, `mse_oos_eq` Out-of-sample diagnostics (if requested), reported *unclipped*.

`robust_se` Per-equation list of robust standard errors.

`robust_inference` Per-equation list of `coef`, `se`, `t`, `p` from HC/HAC robust inference.

`portmanteau_p` Portmanteau serial-correlation p-value.

`hc_type` The HC type used.

<code>estimate_factor_OU</code>	<i>Estimate Factor Ornstein-Uhlenbeck / AR(1) model (Stan if available)</i>
---------------------------------	---

Description

Estimates a Bayesian multivariate, discrete-time mean-reverting model (a first-order vector autoregression with cross-equation coupling) in which Y_t depends on its own lag and on lagged X_{t-1} through a β matrix. This is the discrete-time analogue of a coupled Ornstein-Uhlenbeck system; see the vignette section "Methodological notes" for the exact mapping $\phi = e^{-\kappa\Delta t}$. Uses **cmdstanr** when available, otherwise **rstan**, with a discrete AR(1) fallback.

Usage

```
estimate_factor_OU(
  factors_data,
  data_prep = NULL,
  chains = 4,
  iter = 2000,
  seed = 1234,
  adapt1 = 0.95,
  adapt2 = 0.999,
  mtd1 = 12,
  mtd2 = 15,
  verbose = TRUE
)
```

Arguments

factors_data	List with scores_X, scores_Y.
data_prep	Optional preprocessed data (reserved).
chains, iter, seed	Stan sampling controls. iter is the total per chain; warmup is floor(iter/2) with a floor that guarantees at least 100 post-warmup draws.
adapt1, mtd1	adapt_delta and max_treedepth for the first run.
adapt2, mtd2	adapt_delta and max_treedepth used for an automatic re-run if the first run shows divergent transitions or poor mixing.
verbose	Logical; print progress/details.

Value

A list with posterior medians (ϕ , μ , β), half-lives, coupling strength, pseudo-R2, MCMC convergence diagnostics (diagnostics: max R-hat, min ESS, divergences) and the fitted Stan object.

Convergence is testable, not assumed

The persistence ϕ is *not* constrained to (0, 1): it is given a generous support and a weakly-informative, convergence-neutral prior (normal(0.5, 0.5)), so the posterior can place mass on unit-root or mildly explosive dynamics. Mean reversion (convergence) is therefore a conclusion supported by the data, not an artefact of the parameterization.

Examples

```
set.seed(123)
n <- 60
X_scores <- matrix(rnorm(n * 2), n, 2)
Y_scores <- matrix(rnorm(n * 2), n, 2)
factors_data <- list(scores_X = X_scores, scores_Y = Y_scores)
ou_result <- estimate_factor_OU(factors_data, chains = 2, iter = 600,
                               verbose = FALSE)

print(ou_result$half_lives_Y)
print(ou_result$diagnostics)
```

make_X_innovations	<i>Extract X innovations (reduced-form VAR residuals)</i>
--------------------	---

Description

Computes *reduced-form* innovations for the X factors by fitting a VAR and returning its residuals. These are not structural shocks: identifying structural shocks would require an identification scheme (e.g. a Cholesky or sign restriction); the residuals here are correlated across equations.

Usage

```
make_X_innovations(results, p = NULL)
```

Arguments

results List. Output from main analysis.
p Integer. VAR lag order. If NULL, uses order from DFM estimation.

Value

List with components:

shocks Matrix (T x Kx) of reduced-form innovations with NA padding for the initial lags.
var_fit Fitted VAR model object.
lag_used Integer lag order used.

placebo_values	<i>Aggregate-preserving placebo values (negative controls for gravitation)</i>
----------------	--

Description

Builds theoretically false "values" that respect the SAME aggregate marxian constraints (per-period total surplus $\sum_i p_i$ and the general profit rate G') but BREAK the marxian sectoral structure. Used as negative controls: if the genuine sectoral value gravitates (wedge reverts / OOS) better than placebos that share all aggregate accounting, the specific marxian assignment carries information beyond the shared cost price.

Usage

```
placebo_values(  
  k_cost,  
  K,  
  Gprime,  
  p,  
  scheme = c("uniform", "permute"),  
  seed = 1L  
)
```

Arguments

k_cost Numeric matrix [T, S], the cost price $k = c + v$.
K Numeric matrix [T, S], total advanced capital.
Gprime Numeric vector length T, the general profit rate.
p Numeric matrix [T, S], the true sectoral surplus.
scheme Either "uniform" or "permute".
seed Integer seed for the permutation (reproducibility). Default 1L. Only used by "permute".

Details

Schemes:

"uniform" Every sector earns the average profit rate: $p_i^{plac} = G' K_i$, so $V^{plac} = k + G' K = \Phi$ and the wedge is identically zero (the "no redistribution" null). Tests whether the value structure adds anything over the production price.

"permute" A fixed (seeded) derangement σ reassigns each sector's surplus series to another sector: $p_i^{plac} = p_{\sigma(i)}$. Per-period totals are preserved ($\sum_i p_{\sigma(i)} = \sum_i p_i$) but the $K_i \leftrightarrow p_i$ pairing is destroyed, so the wedge $K_i G' - p_{\sigma(i)}$ mis-pairs capital and surplus.

Value

A list with p_placebo, V_placebo ($= k + p^{plac}$), W_placebo ($= K G' - p^{plac}$), the scheme, and (for "permute") the perm used.

Examples

```
set.seed(1); Tn <- 12; S <- 6
k <- matrix(runif(Tn * S, 1, 3), Tn, S)
K <- k + matrix(runif(Tn * S, 1, 3), Tn, S)
p <- matrix(runif(Tn * S, 0.1, 1), Tn, S)
Gp <- rowSums(p) / rowSums(K)
pl <- placebo_values(k, K, Gp, p, scheme = "permute", seed = 42)
max(abs(rowSums(pl$p_placebo) - rowSums(p))) # ~0: per-period total preserved
```

```
plot_error_correction_panel
```

Plot error correction panel

Description

Creates a two-panel plot showing (1) error correction terms over time and (2) their estimated half-lives, providing visual assessment of convergence speed.

Usage

```
plot_error_correction_panel(
  results_robust,
  use_contemporaneous_X = TRUE,
  main_left = expression(u[t] == Y[t] - hat(mu)[Y](X[t])),
  main_right = "Half-life de u[t]"
)
```

Arguments

`results_robust` List. Results from main analysis pipeline.

`use_contemporaneous_X` Logical. Use contemporaneous X (time t, the default) or lagged X (time t-1) for computing the equilibrium path.

`main_left` Expression/character. Title for the left panel.

`main_right` Character string. Title for the right panel.

Value

Invisibly returns a list with components:

`U` Matrix of error correction terms (T x Ky).

`half_lives` Vector of estimated half-lives for each Y factor.

read_cpi	<i>Read CPI data from an Excel file</i>
----------	---

Description

Reads an aggregate Consumer Price Index series from an Excel file, with robust column detection and data validation. The date/year and CPI columns are located by pattern-matching on lower-cased header names; localized numerics (comma decimals) are parsed with `to_num_commas()`; duplicate years are collapsed by averaging.

Usage

```
read_cpi(path_cpi)
```

Arguments

`path_cpi` Path to the CPI Excel file.

Value

A `data.frame` with two columns: Year (integer) and CPI (numeric), sorted ascending by Year. The function errors (it does not return NULL) when the file is missing or no date/CPI column can be identified.

See Also

`to_num_commas()`

rescue_short_run_channel

Rescue short-run channel test

Description

Evaluates the short-run relationship from X to Y factors using three complementary, valid pieces of evidence: (1) a coupling null obtained by circularly time-shifting the X residual (breaking its alignment with Y while preserving its own autocorrelation – the rotation-based null used previously was invariant and therefore vacuous); (2) an expanding-window out-of-sample comparison of forecasts with and without lagged X, with a Clark-West test of the improvement; and (3) a Granger causality test.

Usage

```
rescue_short_run_channel(
  results_robust,
  lag = 1,
  B = 1000,
  seed = NULL,
  ridge = 0.001,
  oos_start = 0.6,
  verbose = TRUE
)
```

Arguments

results_robust	Output from run_complete_factor_analysis_robust()
lag	Number of lags for the model (default: 1)
B	Number of null replicates (default: 1000)
seed	Random seed for reproducibility (default: NULL)
ridge	Ridge regularization parameter (default: 0.001)
oos_start	Proportion of data to use for training (default: 0.6)
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

List with components:

p_values, p_values_fdr Coupling p-values (time-shift null) and their Benjamini-Hochberg adjustment.

OOS Out-of-sample RMSE/R2 comparison plus the Clark-West test (cw_stat, cw_p).

granger_p Granger causality p-value.

observed, sim_quantiles, null_stats Coupling statistics and their null distributions.

rotation_null_test	<i>Null hypothesis test for X->Y factor coupling</i>
--------------------	---

Description

Tests whether the observed association between the lagged X factor space and the Y factor space is stronger than expected when the X->Y temporal alignment is broken.

Usage

```
rotation_null_test(
  scores_X,
  scores_Y,
  lag = 1,
  B = 1000,
  seed = 123,
  compute = c("procrustes", "cca", "principal", "dynbeta"),
  null_method = c("circular_shift", "rotation"),
  progress = TRUE
)
```

Arguments

scores_X	Factor scores from the first dataset (T x Kx).
scores_Y	Factor scores from the second dataset (T x Ky).
lag	Number of lags for the X->Y relationship (default 1).
B	Number of null replicates (default 1000).
seed	Random seed for reproducibility (default 123). The RNG state is restored on exit.
compute	Statistics to compute: any of 'procrustes', 'cca', 'principal', 'dynbeta'.
null_method	Null-generating mechanism: "circular_shift" (default) or "rotation" (invariance diagnostic only).
progress	Logical, show progress bar (default TRUE).

Value

List with components:

observed Observed statistics.

null_stats List of length-B null vectors per statistic.

p_values Monte Carlo one-sided p-values $(1 + \#\{\text{null} \geq \text{obs}\}) / (B + 1)$.

p_values_fdr Benjamini-Hochberg FDR-adjusted p-values.

params Test settings, including the seed and null method.

Why not a rotation null

The coupling statistics used here (Procrustes nuclear-norm ratio, canonical correlations, principal angles, dynamic-beta Frobenius norm) are *invariant* to orthogonal rotation of either factor space: rotating Y by an orthogonal matrix leaves all of them unchanged, so a rotation-based null is degenerate (its "distribution" is a point mass at the observed value and every p-value is ~ 1). The default null (`null_method = "circular_shift"`) instead applies a random circular time shift to Y , which preserves each series' own autocorrelation while destroying the cross-series alignment – the statistics then genuinely vary under the null. `null_method = "rotation"` is retained only as an invariance diagnostic and will (correctly) yield non-significant results.

row_norm1	<i>Normalize matrix rows to sum to one</i>
-----------	--

Description

Scales each row of a matrix so that its elements sum to 1. Handles zero-sum rows by leaving them unchanged (avoiding division by zero).

Usage

```
row_norm1(M)
```

Arguments

`M` Numeric matrix to be row-normalized.

Value

Matrix of the same dimensions as `M` with each row summing to 1 (except rows that originally summed to zero).

Examples

```
M <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
M_norm <- row_norm1(M)
rowSums(M_norm) # Should be c(1, 1)
```

run_complete_factor_analysis_robust

Complete factor-OU convergence analysis pipeline

Description

Executes the end-to-end analysis workflow: data preparation, PLS-based factor extraction, DFM estimation, Factor-OU/AR(1) estimation, convergence tests, and robustness checks. This is the main user-facing function.

Usage

```
run_complete_factor_analysis_robust(
  X_matrix,
  Y_matrix,
  TMG = NULL,
  COM_matrix = NULL,
  SPVR_matrix = NULL,
  CA = NULL,
  sector_names = NULL,
  max_comp = 3,
  dfm_lags = 1,
  ou_chains = 4,
  ou_iter = 2000,
  skip_ou = FALSE,
  run_convergence_tests = TRUE,
  make_plots = FALSE,
  plot_file = NULL,
  seed = 123,
  path_cpi = NULL,
  path_weights = NULL,
  verbose = TRUE
)
```

Arguments

X_matrix	Matrix of first set of variables
Y_matrix	Matrix of second set of variables
TMG	Optional total Marxist gross output (default: NULL)
COM_matrix	Optional commodity-composition matrix (default: NULL)
SPVR_matrix	Optional surplus-value-rate matrix (default: NULL)
CA	Optional additional covariates (default: NULL)
sector_names	Vector of sector names (default: NULL)
max_comp	Maximum number of components (default: 3)

<code>dfm_lags</code>	Number of lags for DFM (default: 1)
<code>ou_chains</code>	Number of MCMC chains for OU estimation (default: 4)
<code>ou_iter</code>	Number of MCMC iterations (default: 2000)
<code>skip_ou</code>	Logical, skip OU estimation (default: FALSE)
<code>run_convergence_tests</code>	Logical, run convergence tests (default: TRUE)
<code>make_plots</code>	Logical, draw the factor-dynamics figure (default: FALSE). When TRUE and <code>plot_file</code> is set, the figure is written to that PDF; otherwise it is drawn on the current device.
<code>plot_file</code>	Optional path for the saved figure (default: NULL).
<code>seed</code>	Integer master seed for the stochastic steps (default: 123).
<code>path_cpi</code>	Path to CPI data (default: NULL)
<code>path_weights</code>	Path to weights data (default: NULL)
<code>verbose</code>	Logical; print progress and diagnostic information. Default TRUE.

Details

This function orchestrates the complete analysis: data validation, PLS factor extraction, VAR-based DFM, Bayesian factor-OU/AR(1), convergence tests (stationary band on phi, Johansen cointegration, residual unit-root with caveats) and robustness tests (time-shift permutation, reweighting, delete-one-sector jackknife). Plotting is opt-in (`make_plots`); by default no graphics device is opened and no file is written, so the function is safe in batch/non-interactive use.

Value

List with components:

`data_prep` Scaled (possibly auxiliary-augmented) X and Y.
`selection` PLS component selection (see [select_optimal_components_safe](#)).
`factors` PLS-extracted factors (`scores_X`, `scores_Y`) and interpretation.
`dfm` DFM/VAR estimation (see [estimate_DFM](#)).
`factor_ou` Factor-OU/AR(1) estimates (see [estimate_factor_OU](#)), including MCMC diagnostics.
`convergence_tests` Convergence and robustness test results (if `run_convergence_tests` = TRUE).

See Also

[estimate_DFM](#), [estimate_factor_OU](#), [run_convergence_robustness_tests](#), [visualize_factor_dynamics](#)

Examples

```
set.seed(123)
X <- matrix(rnorm(100 * 10), 100, 10)
Y <- X + matrix(rnorm(100 * 10, 0, 0.5), 100, 10)

results <- run_complete_factor_analysis_robust(
```

```

X_matrix = X, Y_matrix = Y,
max_comp = 3, dfm_lags = 1,
skip_ou = TRUE,          # set FALSE to run the Bayesian factor-OU model
verbose = FALSE
)
str(results$dfm$r2_global)

```

run_convergence_robustness_tests

Run comprehensive robustness test suite

Description

Executes all available robustness tests (permutation, reweighting, jackknife) and synthesizes results into an integrated assessment.

Usage

```

run_convergence_robustness_tests(
  results_robust,
  X_matrix,
  Y_matrix,
  path_cpi = NULL,
  path_weights = NULL,
  sector_names = NULL,
  run_permutation = TRUE,
  run_reweighting = FALSE,
  run_jackknife = TRUE,
  run_leadlag = FALSE,
  run_common_factor = FALSE,
  sensitivity_analysis = TRUE,
  verbose = TRUE
)

```

Arguments

results_robust	Output from run_complete_factor_analysis_robust()
X_matrix	Matrix of first set of variables
Y_matrix	Matrix of second set of variables
path_cpi	Path to CPI data (default: NULL)
path_weights	Path to weights data (default: NULL)
sector_names	Vector of sector names (default: NULL)
run_permutation	Logical, run permutation test (default: TRUE)

```

run_reweighting      Logical, run reweighting test (default: FALSE)
run_jackknife        Logical, run jackknife test (default: TRUE)
run_leadlag          Logical, run lead-lag test (default: FALSE)
run_common_factor    Logical, run common factor test (default: FALSE)
sensitivity_analysis Logical, run sensitivity analysis (default: TRUE)
verbose              Logical; print progress and diagnostic information. Default TRUE.

```

Value

List with components:

```

permutation Results from permutation test.
reweighting Results from reweighting test (if applicable).
jackknife   Results from jackknife test (if requested).
summary     Data frame summarizing all tests.
overall_robust Logical indicating if convergence is robust across all tests.

```

```
run_rotation_null_on_results
```

Run the coupling null test on complete analysis results

Description

Wrapper that extracts factors from a complete analysis object and runs `rotation_null_test` with the corrected time-shift null.

Usage

```

run_rotation_null_on_results(
  results_robust,
  lag = 1,
  B = 1000,
  seed = 42,
  null_method = c("circular_shift", "rotation"),
  compute = c("procrustes", "cca", "principal", "dynbeta")
)

```

Arguments

results_robust	Output from <code>run_complete_factor_analysis_robust()</code> .
lag	Number of lags for the X->Y relationship (default 1).
B	Number of null replicates (default 1000).
seed	Random seed for reproducibility (default 42).
null_method	Null mechanism passed to <code>rotation_null_test()</code> ; "circular_shift" (default) or "rotation" (diagnostic).
compute	Statistics to compute: 'procrustes', 'cca', 'principal', 'dynbeta'.

Value

List. Output from [rotation_null_test](#).

`select_optimal_components_safe`

Select optimal number of PLS components with cross-validation

Description

Determines the optimal number of Partial Least Squares (PLS) components using k-fold or leave-one-out cross-validation, with robust error handling. The number of components is chosen with the one-standard-error rule (the most parsimonious model whose cross-validated RMSEP is within one standard error of the minimum), falling back to an R2 elbow and then to the RMSEP minimizer.

Usage

```
select_optimal_components_safe(
  X,
  Y,
  max_comp = 15,
  validation = c("CV", "LOO"),
  segments = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

X	First data matrix (predictors), T x p.
Y	Second data matrix (responses), T x q (or a vector).
max_comp	Maximum number of components to test. Default 15. It is internally capped at <code>min(nrow(X) - 1, ncol(X))</code> .
validation	Character; cross-validation type, one of "CV" (k-fold, default) or "LOO" (leave-one-out, deterministic).

segments	Integer; number of folds when validation = "CV". Default min(10, nrow(X)).
seed	Optional integer. With validation = "CV" the folds are random; supplying a seed makes the selection reproducible. The RNG state is restored on exit. Default NULL.
verbose	Logical; print progress and diagnostic information. Default TRUE.

Details

The function fits PLS models with 1 to max_comp components, evaluates each via cross-validation, and selects the number that minimizes prediction error while avoiding overfitting (one-SE rule).

Value

A list with components:

optimal_ncomp Integer, number of components selected.
 pls_model Fitted pls::plsr model (all max_comp components, validated).
 R2_cv Cross-validated R2 at optimal_ncomp.
 var_explained Percent of Y variance explained (CV) at optimal_ncomp.
 rmsep_cv Numeric vector of cross-validated RMSEP for 1..max_comp components.
 validation The validation method actually used.

test_cointegration_control

Classical cointegration control (Johansen trace or eigen)

Description

Runs Johansen's cointegration test on the first min(2, ncol(X), ncol(Y)) factors from scores_X and scores_Y, and counts the number of cointegrating relations at the 5% level.

Usage

```
test_cointegration_control(
  factors_data,
  max_lag = 4,
  type = "trace",
  ecdet = "const",
  verbose = TRUE
)
```

Arguments

factors_data	A list with matrices scores_X and scores_Y (T x Kx) and (T x Ky), respectively.
max_lag	Integer; maximum lag K passed to <code>urca::ca.jo()</code> .
type	Character; Johansen test type, one of "trace" or "eigen". Defaults to "trace".
ecdet	Character; deterministic terms, e.g. "const", "trend", or "none". Defaults to "const".
verbose	Logical; print progress and a summary of the test. Default TRUE.

Details

This function requires the optional package **urca** (declared in Suggests). It does not attempt to install packages at runtime; if **urca** is unavailable, an informative error is thrown.

The 5% critical values are taken from the "5pct" column of the cval slot returned by `urca::ca.jo()`.

Value

A list with:

- test: the `urca::ca.jo` fitted object,
- n_coint: integer number of cointegrating relations at 5%,
- vectors: matrix of cointegrating vectors (or NULL if none).

References

Johansen, S. (1991). Estimation and Hypothesis Testing of Cointegration Vectors in Gaussian Vector Autoregressive Models. *Econometrica*, 59(6), 1551-1580.

Johansen, S. (1995). *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

See Also

[ca.jo](#)

Examples

```
if (requireNamespace("urca", quietly = TRUE)) {
  set.seed(1)
  T <- 120
  X <- cbind(cumsum(rnorm(T)), cumsum(rnorm(T)))
  Y <- cbind(cumsum(rnorm(T)), cumsum(rnorm(T)))
  fd <- list(scores_X = X, scores_Y = Y)
  out <- test_cointegration_control(fd, max_lag = 2, verbose = FALSE)
  str(out)
}
```

test_jackknife_sectors

Delete-one-sector jackknife of the X->Y coupling

Description

Genuine leave-one-sector-out jackknife: each sector (column) is dropped in turn from both X and Y, the coupling is re-estimated, and the delete-one replicates are used to compute the jackknife bias and standard error and to rank sectors by influence. (The previous version dropped the top-k highest-variance sectors a single time and was not a jackknife.)

Usage

```
test_jackknife_sectors(
  X_matrix,
  Y_matrix,
  sector_names = NULL,
  k_exclude = 3,
  max_comp = 3,
  seed = 123,
  verbose = TRUE
)
```

Arguments

X_matrix	Matrix of first set of variables (sectors in columns)
Y_matrix	Matrix of second set of variables (sectors in columns)
sector_names	Vector of sector names (default: column names of Y)
k_exclude	Number of most-influential sectors to report (default: 3)
max_comp	Maximum number of components (default: 3)
seed	Optional integer for reproducible component selection/diagnostics.
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

List with components:

baseline Full-sample coupling.

jackknife_estimates Named vector of delete-one coupling estimates.

bias Jackknife bias estimate $(n-1)(\text{mean}(\text{jk}) - \text{full})$.

se Jackknife standard error.

influence Per-sector change $\text{full} - \text{jk}_i$ (signed).

retention Per-sector $\text{jk}_i / \text{full}$.

influential_sectors The k_exclude most influential sectors.

robust TRUE if no single sector changes the coupling by more than 50 percent.

test_leave_cluster_out

Leave-cluster-out robustness of the X->Y coupling

Description

Generalizes the delete-one-sector jackknife (`test_jackknife_sectors()`) to delete-one-**cluster**: each cluster of sectors is dropped in turn from both X and Y, and the coupling is re-estimated on the remaining sectors. Under cross-sectional dependence of the input-output ("MIP") kind, a naive leave-one-out is optimistic because the excluded sector's information leaks in through intermediate demand; dropping an entire value chain forces the prediction to rely on the general gravitation rather than on near-collinear neighbours. This is the cross-sectional complement of the temporal time-shift / block-bootstrap nulls already in the package (`rotation_null_test()`, `test_permutation_robustness()`); it reuses the same coupling pipeline (`.coupling_estimate`), it does not reimplement it.

Usage

```
test_leave_cluster_out(
  X_matrix,
  Y_matrix,
  cluster_map = NULL,
  n_clusters = 3,
  fallback = c("correlation", "com"),
  com = NULL,
  max_comp = 3,
  seed = 123,
  verbose = TRUE
)
```

Arguments

X_matrix, Y_matrix	Numeric matrices (sectors in columns; Y carries the sector identities used by the cluster map).
cluster_map	Optional sector-to-cluster map (see "Cluster map"). NULL triggers the fallback.
n_clusters	Integer; number of clusters for the fallback (ignored when cluster_map is supplied). Default 3.
fallback	Fallback rule when cluster_map is NULL: "correlation" (default) or "com".
com	Optional per-sector organic-composition vector for fallback = "com".
max_comp	Maximum number of PLS components (default 3).
seed	Optional integer for reproducible component selection/diagnostics.
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

List with components:

baseline Full-sample coupling.
 cluster_estimates Named vector of leave-one-cluster coupling estimates.
 cluster_members Named list of the sectors in each cluster.
 cluster_sizes Named integer vector of cluster sizes.
 bias,se Delete-a-group jackknife bias and standard error.
 influence Per-cluster signed change baseline - lco_c.
 retention Per-cluster lco_c / baseline.
 influential_clusters Clusters ordered by absolute influence.
 n_clusters Number of clusters.
 cluster_map The resolved per-sector cluster labels.
 map_source "user" or "fallback:<method>".
 robust TRUE if no single cluster changes the coupling by more than 50 percent.

Cluster map (pluggable)

Supply cluster_map as the genuine input-output partition (a per-sector label vector, or a named list mapping each cluster to its sector names/indices). When it is NULL, a documented **fallback** partition is built with `build_cluster_map()` ("correlation" or "com"); the fallback is a stopgap proxy, not a demand-linkage partition, and a message flags its use.

Statistical layer (do not over-read)

bias and se are the delete-a-group (block) jackknife estimates with the $(g - 1)/g$ factor over the g cluster-deletion replicates. They are well-calibrated for roughly balanced clusters; with strongly unequal MIP clusters they are an approximate, conservative summary, and the primary outputs are the per-cluster influence/retention and the robust verdict. The verdict is purely a robustness diagnostic, not a coupling point estimate.

See Also

`test_jackknife_sectors()`, `build_cluster_map()`

test_permutation_robustness

Permutation-based robustness test for X->Y coupling

Description

Tests whether the estimated X->Y coupling strength is stronger than expected when the temporal alignment between X and Y is broken. The null is generated by circularly time-shifting the Y factors, which preserves Y's own autocorrelation while destroying the cross-series relationship.

Usage

```
test_permutation_robustness(
  factors_data,
  data_prep,
  n_perms = 100,
  seed = 123,
  use_stan = TRUE,
  chains = 4,
  iter = 2000,
  verbose = TRUE
)
```

Arguments

factors_data	List with scores_X, scores_Y.
data_prep	Prepared data object (passed to the OU estimator).
n_perms	Number of permutations (default: 100).
seed	Random seed for reproducibility (default: 123). Now honoured.
use_stan	Logical, use Stan for estimation (default: TRUE).
chains	Number of MCMC chains (default: 4).
iter	Number of MCMC iterations (default: 2000).
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

List with components:

baseline	Observed coupling strength.
permuted_mean, permuted_median, permuted_sd	Summary of the null coupling distribution.
p_value	Monte Carlo one-sided p-value $(1 + \#\{\text{null} \geq \text{baseline}\}) / (n + 1)$.
z_score	Standardized effect size.
robust	TRUE if $p_value < 0.05$.
n_successful	Number of usable replicates.

Why not a column permutation

The previous version permuted the *columns* of the Y factors. The coupling strength is a Frobenius norm of the coupling matrix and is invariant to such a relabelling, so the old null was degenerate (every replicate reproduced the baseline). Shuffling time, not factor labels, is what tests coupling.

test_reweighting_robustness

Reweighting-based robustness test

Description

Tests sensitivity of the coupling result to alternative sectoral weighting schemes. For each perturbed weight vector the sectoral price levels are obtained from the canonical closed-form conjugate disaggregation ([BayesianDisaggregation::disaggregate_conjugate\(\)](#)), a Kalman/RTS smoother that conditions genuinely on the aggregate CPI, and the (fallback) factor-OU coupling is re-estimated. The coefficient of variation of the coupling across schemes summarizes the sensitivity.

Usage

```
test_reweighting_robustness(
  path_cpi,
  path_weights,
  X_matrix,
  max_comp = 3,
  seed = 123,
  verbose = TRUE
)
```

Arguments

path_cpi	Path to CPI data file
path_weights	Path to weights data file
X_matrix	Matrix of variables
max_comp	Maximum number of components (default: 3)
seed	Optional integer for reproducible priors (default: 123).
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

List with components:

results Per-scheme list with coupling and half_life_mean.

cv_coupling Coefficient of variation of coupling across schemes.

robust TRUE if $cv_coupling < 0.3$.

Canonical disaggregation

Earlier versions used a deterministic weight blend (`run_disaggregation_custom_prior`) that did not condition on the CPI. That method is removed; the single source of truth is now the conjugate engine in **BayesianDisaggregation**. A constant-in-time custom prior is replicated across periods to form the value-added (VAB) weight matrix W , and the point estimate used is the posterior median (`$phi_summary$median`), i.e. the RTS-smoothed sectoral levels, now genuinely conditioned on the CPI.

to_num_commas	<i>Convert localized number strings to numeric</i>
---------------	--

Description

Converts character strings that may use either the European convention (comma decimal separator, period thousands separator, e.g. "1.234,56") or the Anglo-Saxon convention (period decimal separator, comma thousands separator, e.g. "1,234.56") to numeric. Plain numbers and already-numeric inputs are returned unchanged. The function is vectorized.

Usage

```
to_num_commas(x)
```

Arguments

`x` Numeric value or character vector with localized number strings.

Details

Disambiguation rules, applied per element after stripping whitespace:

- If both `.` and `,` are present, the *rightmost* of the two is treated as the decimal mark and the other as the thousands separator.
- If only commas are present: a single comma is treated as a decimal mark; multiple commas are treated as thousands separators and removed.
- If only periods are present: a single period is treated as a decimal mark (Anglo-Saxon); multiple periods are treated as thousands separators and removed.

The single ambiguous case "1.234" (one period, no comma) is interpreted as the decimal 1.234, not as 1234.

Value

Numeric vector the same length as `x`. Entries that cannot be parsed become `NA`.

Examples

```
to_num_commas("1.234,56") # European -> 1234.56
to_num_commas("1,234.56") # US       -> 1234.56
to_num_commas("1234.56")  # plain    -> 1234.56
to_num_commas(1234.56)    # numeric  -> 1234.56
```

```
visualize_factor_dynamics
```

Visualize factor dynamics comprehensively

Description

Creates a multi-panel visualization summarizing the factor model: X and Y factor scores over time and the DFM and OU half-lives. Plotting is a side effect; nothing is written unless `save_plot = TRUE`.

Usage

```
visualize_factor_dynamics(
  dfm_result,
  ou_result,
  factors_data,
  save_plot = FALSE,
  plot_file = NULL,
  use_device = "default",
  verbose = TRUE
)
```

Arguments

<code>dfm_result</code>	Result object from DFM analysis
<code>ou_result</code>	Result object from OU estimation
<code>factors_data</code>	Data frame with factor information
<code>save_plot</code>	Logical, save plot to a PDF file (default: FALSE)
<code>plot_file</code>	File name for the saved plot. If <code>save_plot = TRUE</code> and this is NULL, a file in <code>tempdir()</code> is used (with a message). Default NULL.
<code>use_device</code>	Graphics device to use: 'default', 'pdf', 'none' (default: 'default')
<code>verbose</code>	Logical; print progress and diagnostic information. Default TRUE.

Value

Invisibly returns TRUE when all four panels were created.

```
visualize_factor_dynamics_simple
```

Simple factor dynamics visualization

Description

Creates a simplified multi-panel plot of factor scores over time with minimal dependencies. Useful for quick diagnostics.

Usage

```
visualize_factor_dynamics_simple(
  factors_data,
  output_file = NULL,
  verbose = TRUE
)
```

Arguments

factors_data	List containing factor scores.
output_file	Character string. Optional file path for saving. Default is NULL.
verbose	Logical; print progress and diagnostic information. Default TRUE.

Value

Invisibly returns NULL.

Examples

```
data <- list(scores_X = matrix(rnorm(100), ncol = 2),
             scores_Y = matrix(rnorm(100), ncol = 2))
tmp_file <- file.path(tempdir(), "test_plot.pdf")
visualize_factor_dynamics_simple(data, output_file = tmp_file)
```

```
wedge_stationarity
```

Per-sector and panel stationarity / mean-reversion of the wedge

Description

Tests whether the transformation wedge is a bounded, mean-reverting series (evidence that the redistribution gravitates and does not diverge) rather than a random walk. For each sector it fits a discrete AR(1) $W_t = c + \rho W_{t-1} + e_t$ (reporting ρ , the implied mean-reversion speed $\kappa = -\log \rho$ and half-life $\log(0.5)/\log \rho$) and an augmented Dickey-Fuller test (urca::ur.df, drift). The panel summary reports the fraction of sectors rejecting the unit-root null and the distribution of reversion speeds. Boundedness is reported as the range and the max-abs over standard deviation. This is a descriptive diagnostic; no single pooled p-value is claimed (anti-overreach: panel unit-root nulls require their own tabulated distribution).

Usage

```
wedge_stationarity(W, adf_lags = 2, alpha = 0.05, verbose = TRUE)
```

Arguments

<code>W</code>	Numeric matrix $[T, S]$, the wedge (e.g. from <code>compute_wedge</code>).
<code>adf_lags</code>	Maximum lag order for the ADF auxiliary regression (<code>urca::ur.df lags=</code> , with <code>selectlags = "AIC"</code>). Default 2.
<code>alpha</code>	Significance level for the ADF unit-root rejection. Default 0.05 (uses the 5pct critical value; 0.01/0.10 supported).
<code>verbose</code>	Logical; print a short panel summary. Default TRUE.

Value

A list with `per_sector` (a data frame: `rho`, `kappa`, `half_life`, `adf_stat`, `adf_crit`, `reject_unit_root`, `range`, `maxabs_over_sd`) and `panel` (fraction rejecting, median/IQR of `rho` and `half_life`, and the boundedness summary).

Examples

```
if (requireNamespace("urca", quietly = TRUE)) {
  set.seed(1); Tn <- 60; S <- 4
  W <- sapply(1:S, function(s) as.numeric(stats::arima.sim(list(ar = 0.6), Tn)))
  W <- W - rowMeans(W) # impose zero-sum like a real wedge
  wedge_stationarity(W, verbose = FALSE)$panel$frac_reject
}
```


Index

BayesianDisaggregation::disaggregate_conjugate(), num_commas, 29
28
to_num_commas(), 13
build_cluster_map, 3
build_cluster_map(), 26
ca.jo, 23
choose_var_lag, 4
compute_wedge, 5
deltaR2_ou, 6
diagnose_data, 7
estimate_DFM, 8, 18
estimate_factor_OU, 9, 18
make_X_innovations, 10
placebo_values, 11
plot_error_correction_panel, 12
read_cpi, 13
rescue_short_run_channel, 14
rotation_null_test, 15, 20, 21
rotation_null_test(), 25
row_norm1, 16
run_complete_factor_analysis_robust, 17
run_convergence_robustness_tests, 18, 19
run_rotation_null_on_results, 20
select_optimal_components_safe, 18, 21
test_cointegration_control, 22
test_jackknife_sectors, 24
test_jackknife_sectors(), 25, 26
test_leave_cluster_out, 25
test_leave_cluster_out(), 3
test_permutation_robustness, 26
test_permutation_robustness(), 25
test_reweighting_robustness, 28
visualize_factor_dynamics, 18, 30
visualize_factor_dynamics_simple, 31
wedge_stationarity, 31