Package 'convergenceDFM'

May 8, 2026

Type Package

Title Convergence and Dynamic Factor Models

Version 0.1.4

Description Tests convergence in macro-financial panels combining Dynamic Factor Models (DFM) and mean-reverting Ornstein-Uhlenbeck (OU) processes. Provides: (i) static/approximate DFMs for large panels with VAR/VECM stability checks, Portmanteau tests and rolling out-of-sample R^2, following 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 à la Johansen (1988) <doi:10.1016/0165-1889(88)90041-3>; (iii) OU-based 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>; (iv) robust inference via 'sandwich' HC/HAC estimators (Zeileis (2004) <doi:10.18637/jss.v011.i10>) and regression diagnostics ('lmtest'); and (v) optional PLS-based factor preselection (Mevik and Wehrens (2007) <doi:10.18637/jss.v018.i02>). Functions emphasize reproducibility and clear, publication-ready summaries.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports stats, methods, pls, vars, urca, readxl, dplyr, tidyr, stringr, magrittr, zoo

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

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

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author José Mauricio Gómez Julián [aut, cre]

Maintainer José Mauricio Gómez Julián <isadorenabi@pm.me>

Repository CRAN

Date/Publication 2025-12-01 13:40:07 UTC

Contents

`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

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

---

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

results_robust	Output from run_complete_factor_analysis_robust()
lag	Number of lags for the model (default: 1)
oos	Logical, perform out-of-sample validation (default: TRUE)
seed	Random seed for reproducibility (default: 123)
ridge	Ridge regularization parameter (default: 1e-08)
verbose	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 *Diagnose and prepare data matrices*

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)
```

Arguments

X_matrix	Numeric matrix or data frame of X variables (e.g., Marxist prices).
Y_matrix	Numeric matrix or data frame of Y variables (e.g., market prices/CPI).
verbose	Logical; print diagnostic information. Default TRUE.

Details

The function:

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

Value

List with components:

`X_matrix` Cleaned and prepared X matrix.

`Y_matrix` Cleaned and prepared Y matrix.

`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",
  verbose = TRUE
)
```

Arguments

`factors_data`	List containing PLS-extracted factor scores (`scores_X`, `scores_Y`) and related objects.
`p`	Integer. VAR lag order. If `NULL`, selected automatically. Default is 2.
`compute_oos`	Logical. Should out-of-sample diagnostics be computed? Default is TRUE.
`hc_type`	Character string. Heteroskedasticity-consistent SE type. Default is "HC3".
`verbose`	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.

Value

List with components:

`var_fit` Fitted VAR model on combined factors.

`p_used` VAR lag order used.

`robust_se` Matrices of robust standard errors.

`diagnostics` List of diagnostic tests (stability, serial correlation).

`oos_metrics` Out-of-sample forecast evaluation (if requested).

`estimate_factor_OU` *Estimate Factor Ornstein-Uhlenbeck model (Stan if available)*

Description

Estimates a multivariate OU with cross-equation coupling Y_t depending on lagged X_{t-1} via a β matrix. 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 = 0.98,
  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.
`adapt1`, `adapt2`, `mtd1`, `mtd2`	Advanced Stan controls.
`verbose`	Logical; print progress/details.

Value

A list with posterior medians (ϕ, μ, β), half-lives, coupling strength, pseudo-R2, and the fitted Stan object.

Examples

```
# Create toy factor data
set.seed(123)
n <- 50
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)

# Estimate OU model (reduce iterations for speed)
ou_result <- estimate_factor_OU(factors_data, chains = 2, iter = 500,
                                verbose = FALSE)

# Check half-lives
print(ou_result$half_lives_Y)
```

---

make_X_innovations *Extract X innovations from VAR model*

Description

Computes structural shocks (innovations) for X factors by fitting a VAR model and extracting residuals.

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 structural shocks with NA padding for initial lags.

var_fit Fitted VAR model object.

lag_used Integer lag order used.

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) or lagged X (time t-1) for computing equilibrium path? Default is FALSE (lagged).
main_left	Character string. Title for left panel. Default is "Error Correction Terms u_t".
main_right	Character string. Title for right panel. Default is "Half-Lives of Error Correction".

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 *Read CPI data from Excel file*

Description

Reads Consumer Price Index data from an Excel file with robust error handling and data validation.

Usage

```
read_cpi(path_cpi)
```

Arguments

path_cpi	Path to the CPI data file

Value

Data frame with CPI data. Returns NULL if file not found or read operation fails.

---

rescue_short_run_channel

Rescue short-run channel test

Description

Evaluates the short-run causal relationship from X to Y factors using Granger causality tests and out-of-sample forecast comparison.

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 bootstrap iterations (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.

Details

testing. Default is 0.2.

Value

List with components:

`granger_p` P-values from Granger causality tests.

`OOS` List with out-of-sample RMSE comparison.

`p_values` Bootstrap p-values for RMSE differences.

`rotation_null_test` *Rotation null hypothesis test for factor coupling*

Description

Tests whether the observed correlation structure between X and Y factor spaces is significantly stronger than would be expected under random orthogonal rotations.

Usage

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

Arguments

`scores_X`	Factor scores from first dataset
`scores_Y`	Factor scores from second dataset
`lag`	Number of lags for the model (default: 1)
`B`	Number of bootstrap iterations (default: 1000)
`seed`	Random seed for reproducibility (default: 123)
`compute`	Vector of methods to compute: 'procrustes', 'cca', 'principal', 'dynbeta'
`progress`	Logical, show progress bar (default: TRUE)
`rotate`	Which dataset to rotate: 'X' or 'Y' (default: 'Y')

Details

(contemporaneous). "spearman", or "kendall".

Value

List with components:

`observed` Observed correlation statistics.

`null_distribution` Matrix of statistics under null rotations.

`p_values` One-sided p-values for each statistic.

`significant` Logical indicating significance at alpha = 0.05.

`row_norm1` *Normalize matrix rows to sum to one*

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 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 = 10,
  ou_iter = 10000,
  skip_ou = FALSE,
  run_convergence_tests = TRUE,
  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 TMG matrix (default: NULL)
`COM_matrix`	Optional COM matrix (default: NULL)
`SPVR_matrix`	Optional SPVR matrix (default: NULL)
`CA`	Optional CA parameter (default: NULL)
`sector_names`	Vector of sector names (default: NULL)
`max_comp`	Maximum number of components (default: 3)
`dfm_lags`	Number of lags for DFM (default: 1)
`ou_chains`	Number of MCMC chains for OU estimation (default: 10)
`ou_iter`	Number of MCMC iterations (default: 10000)

skip_ou	Logical, skip OU estimation (default: FALSE)
run_convergence_tests	Logical, run convergence tests (default: TRUE)
path_cpi	Path to CPI data (default: NULL)
path_weights	Path to weights data (default: NULL)
verbose	Logical; print progress and diagnostic information. Default TRUE.

Details

uses column names of X_matrix. and reweighting tests). Can be NULL.

This function orchestrates the complete analysis:

1. Data validation and diagnostics
2. Bayesian CPI disaggregation (if applicable)
3. PLS-based factor extraction with optimal component selection
4. Dynamic Factor Model estimation via VAR
5. Factor Ornstein-Uhlenbeck mean-reversion model
6. Formal convergence tests (stationarity, cointegration, speed)
7. Robustness tests (permutation, reweighting, jackknife)

Value

List with components:

factors List containing PLS-extracted factors (scores_X, scores_Y) and related objects (pls_X, pls_Y, ncomp_X, ncomp_Y).

dfm List with DFM estimation results including VAR fit, lag order, diagnostics, and optional impulse responses.

factor_ou List with Factor-OU model estimates: beta, lambda, sigma, half_life, method used, and optional Stan fit object.

convergence_tests List with formal convergence test results (if run_convergence_tests = TRUE).

robustness_tests List with robustness test results (if run_robustness_tests = TRUE).

diagnostics List with data diagnostics (multicollinearity, stationarity, structural breaks).

bayesian_cpi List with Bayesian disaggregation results (if CPI paths provided).

metadata List with analysis metadata (timestamp, versions, parameters).

See Also

estimate_DFM, estimate_factor_OU, run_convergence_robustness_tests, visualize_factor_dynamics

Examples

```
# Basic usage with simulated data
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,
  ou_chains = 4,
  ou_iter = 2000,
  verbose = FALSE
)

# View convergence summary
summary(results$convergence_tests)

# Visualize results
visualize_factor_dynamics(
  dfm_result = results$dfm,
  ou_result = results$factor_ou,
  factors_data = results$factors
)
```

`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 rotation null test on complete analysis results

Description

Wrapper function that extracts factors from a complete analysis object and runs the rotation null test.

Usage

```
run_rotation_null_on_results(
  results_robust,
  lag = 1,
  B = 1000,
  seed = 42,
  rotate = "Y",
  compute = c("procrustes", "cca", "principal", "dynbeta")
)
```

Arguments

results_robust	Output from run_complete_factor_analysis_robust()
lag	Number of lags for the model (default: 1)
B	Number of bootstrap iterations (default: 1000)
seed	Random seed for reproducibility (default: 42)
rotate	Which dataset to rotate: 'X' or 'Y' (default: 'Y')
compute	Vector of methods 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 components using leave-one-out or k-fold cross-validation, with robust error handling.

Usage

```
select_optimal_components_safe(X, Y, max_comp = 15, verbose = TRUE)
```

Arguments

X	First data matrix
Y	Second data matrix
max_comp	Maximum number of components to test (default: 15)
verbose	Logical; print progress and diagnostic information. Default TRUE.

Details

default) or "CV" (k-fold cross-validation). validation = "LOO".

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.

Value

List with components:

ncomp_optimal Optimal number of components selected.

RMSEP Root Mean Squared Error of Prediction for each component.

R2 R-squared values for each component.

validation_method Validation method used.

pls_fit Fitted PLS model object with optimal components.

---

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 urca::ca.jo().
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` *Jackknife robustness test by sector*

Description

Assesses the influence of individual sectors by systematically dropping each sector and re-estimating convergence parameters. Identifies influential sectors and checks stability.

Usage

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

Arguments

`X_matrix`	Matrix of first set of variables
`Y_matrix`	Matrix of second set of variables
`sector_names`	Vector of sector names (default: NULL)
`k_exclude`	Number of sectors to exclude in each iteration (default: 3)
`max_comp`	Maximum number of components (default: 3)
`verbose`	Logical; print progress and diagnostic information. Default `TRUE`.

Details

column names.

Value

List with components:

`jackknife_estimates` Matrix of lambda estimates (iterations x factors).

`original_estimate` Original lambda from full sample.

`bias` Estimated jackknife bias.

`se` Jackknife standard errors.

`influential_sectors` Character vector of highly influential sectors.

`dfbetas` Matrix of influence measures (DFBETAS).

`test_permutation_robustness` *Permutation-based robustness test*

Description

Tests the robustness of factor-OU convergence findings by randomly permuting the Y factor space and re-estimating the model. Generates empirical null distribution for convergence statistics.

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`	Data frame with factor information
`data_prep`	Prepared data object
`n_perms`	Number of permutations (default: 100)
`seed`	Random seed for reproducibility (default: 123)
`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`.

Details

(too slow for many iterations).

Value

List with components:

`observed_lambda` Original mean-reversion speeds.

`null_distribution` Matrix of permutation-based lambda values.

`p_values` One-sided p-values for each factor.

`significant` Logical vector indicating significance at alpha = 0.05.

`effect_size` Standardized effect sizes (z-scores).

`test_reweighting_robustness` *Reweighting-based robustness test*

Description

Tests sensitivity of convergence results to alternative sectoral weighting schemes by re-running Bayesian disaggregation and full pipeline with perturbed weights.

Usage

```
test_reweighting_robustness(
  path_cpi,
  path_weights,
  X_matrix,
  max_comp = 3,
  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)
`verbose`	Logical; print progress and diagnostic information. Default `TRUE`.

Value

List with components:

`lambda_distribution` Matrix of lambda estimates across schemes.

`original_lambda` Baseline lambda from original weights.

`robust_factors` Logical vector indicating robust convergence.

`sensitivity_metrics` Summary statistics of sensitivity.

to_num_commas *Convert European number format to numeric*

Description

Converts character strings in European number format (using comma as decimal separator and period as thousands separator) to numeric values. Already numeric inputs are returned unchanged.

Usage

```
to_num_commas(x)
```

Arguments

x	Numeric value or character string in European format (e.g., "1.234,56").

Value

Numeric value. Returns NA if conversion fails.

Examples

```
to_num_commas("1.234,56")
to_num_commas("1234.56")
to_num_commas(1234.56)
```

visualize_factor_dynamics *Visualize factor dynamics comprehensively*

Description

Creates a multi-panel visualization summarizing all aspects of the factor model: scores over time, loadings, correlations, OU dynamics, and convergence patterns.

Usage

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

Arguments

dfm_result	Result object from DFM analysis
ou_result	Result object from OU estimation
factors_data	Data frame with factor information
save_plot	Logical, save plot to file (default: FALSE)
plot_file	File name for saved plot (default: 'factor_dynamics.pdf')
use_device	Graphics device to use: 'default', 'pdf', 'png' (default: 'default')
verbose	Logical; print progress and diagnostic information. Default TRUE.

Details

Default is NULL (display only).

Value

Invisibly returns NULL. Called for side effect of creating plots.

---

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)
```

Index