Package 'SignalY'

February 4, 2026

Type Package

Title Signal Extraction from Panel Data via Bayesian Sparse Regression and Spectral Decomposition

Version 1.1.1

Author Jose Mauricio Gomez Julian [aut, cre] (ORCID: <`https://orcid.org/0009-0000-2412-3150`>)

Maintainer Jose Mauricio Gomez Julian <`isadore.nabi@pm.me`>

Description Provides a comprehensive toolkit for extracting latent signals from panel data through multivariate time series analysis. Implements spectral decomposition methods including wavelet multiresolution analysis via maximal overlap discrete wavelet transform, Percival and Walden (2000) <doi:10.1017/CBO9780511841040>, empirical mode decomposition for non-stationary signals, Huang et al. (1998) <doi:10.1098/rspa.1998.0193>, and Bayesian trend extraction via the Grant-Chan embedded Hodrick-Prescott filter, Grant and Chan (2017) <doi:10.1016/j.jedc.2016.12.007>. Features Bayesian variable selection through regularized Horseshoe priors, Piironen and Vehtari (2017) <doi:10.1214/17-EJS1337SI>, for identifying structurally relevant predictors from high-dimensional candidate sets. Includes dynamic factor model estimation, principal component analysis with bootstrap significance testing, and automated technical interpretation of signal morphology and variance topology.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, graphics, grDevices, utils, parallel, waveslim (>= 1.8.4), EMD (>= 1.5.9), urca (>= 1.3.3)

Suggests GPArotation, plotly, cmdstanr (>= 0.7.0), posterior (>= 1.5.0), bayesplot (>= 1.10.0), loo (>= 2.6.0), projpred (>= 2.6.0), testthat (>= 3.0.0), knitr, rmarkdown, patchwork

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

VignetteBuilder knitr

RoxygenNote 7.3.3

URL https://github.com/IsadoreNabi/SignalY

BugReports https://github.com/IsadoreNabi/SignalY/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2026-02-04 16:00:14 UTC

Contents

`SignalY-package` *SignalY: Signal Extraction from Panel Data via Bayesian Sparse Regression and Spectral Decomposition*

Description

SignalY provides a comprehensive methodological framework for extracting latent signals from panel data through the integration of spectral decomposition methods, Bayesian variable selection, and automated technical interpretation. The package is designed for researchers working with multivariate time series who seek to distinguish underlying structural dynamics from phenomenological noise.

Philosophical Foundation

The package operationalizes a distinction between **latent structure** and **phenomenological dynamics**. In complex systems, observed variables often represent the superposition of: (1) underlying generative processes that exhibit persistent, structured behavior; and (2) transient perturbations, measurement noise, and stochastic fluctuations. SignalY provides tools to decompose this mixture and identify which candidate variables contribute meaningfully to the latent structure of a target signal.

This framework recognizes that panel data exhibit **multivariate non-linear interdependence**: the relationships between variables may be complex, non-additive, and evolve over time. The methods implemented here are robust to such complexities while remaining interpretable.

Core Methodological Components

1. Spectral Decomposition (Signal Filtering)

The package implements three complementary approaches to extract trend components from time series:

- **Wavelet Multiresolution Analysis**: Using the maximal overlap discrete wavelet transform (MODWT) with configurable Daubechies wavelets, the signal is decomposed into scale-specific components. Lower-frequency detail levels (e.g., D3, D4) capture structural dynamics while higher-frequency levels capture transient noise.
- **Empirical Mode Decomposition (EMD)**: A data-adaptive method that decomposes signals into intrinsic mode functions (IMFs) without requiring pre-specified basis functions. The residual component captures the underlying trend.
- **Grant-Chan Embedded Hodrick-Prescott Filter**: A Bayesian implementation embedding the HP filter within an unobserved components model, allowing for principled uncertainty quantification around the extracted trend via Markov Chain Monte Carlo sampling.

2. Bayesian Variable Selection (Horseshoe Regression)

When the target signal Y is constructed from or influenced by a set of candidate variables X, identifying which candidates are structurally relevant versus informationally redundant is crucial. The regularized Horseshoe prior provides:

- **Adaptive shrinkage**: Coefficients for irrelevant variables are strongly shrunk toward zero (high kappa), while relevant variables escape shrinkage (low kappa).
- **Uncertainty quantification**: Full posterior distributions over coefficients enable credible interval construction.
- **Automatic sparsity detection**: The effective number of non-zero coefficients (m_eff) is estimated as part of the model.

3. Dimensionality Reduction and Factor Analysis

For high-dimensional panels, the package provides:

- **Principal Component Analysis (PCA)**: With bootstrap significance testing to identify which variables load significantly on each component.
- **Dynamic Factor Models (DFM)**: For extracting common factors that drive co-movement in the panel.

- **Entropy-based interpretation**: Shannon entropy of loadings distinguishes between diffuse systemic movement (high entropy) and concentrated structural signals (low entropy).

4. Unit Root and Stationarity Testing

Comprehensive suite of tests to characterize the persistence properties of extracted signals:

- Augmented Dickey-Fuller (ADF) tests with drift and trend options
- Elliott-Rothenberg-Stock (ERS) DF-GLS and P-tests
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests
- Phillips-Perron tests

Interpretation Framework

SignalY generates automated technical interpretations based on:

- **Signal smoothness**: Comparing variance of second differences between original and filtered series
- **Trend persistence**: Whether extracted trends are deterministic or stochastic based on unit root tests
- **Information topology**: Entropy and distributional fit of PCA loadings indicating structural concentration
- **Sparsity ratio**: Proportion of candidate variables shrunk to zero under Horseshoe regression
- **Regime detection**: Identification of structural breakpoints in mean or volatility

Important Caveats

SignalY provides **methodology**, not **theory**. The statistical identification of relevant variables does not establish causal or structural relationships without supporting domain theory. Users must:

1. Justify variable inclusion based on domain knowledge
2. Interpret sparsity results in theoretical context
3. Recognize that statistical significance is necessary but not sufficient for structural claims

Author(s)

Jose Mauricio Gomez Julian <isadore.nabi@pm.me>

References

Daubechies, I. (1992). Ten Lectures on Wavelets. SIAM.

Grant, A. L., & Chan, J. C. C. (2017). Reconciling output gaps: Unobserved components model and Hodrick-Prescott filter. Journal of Economic Dynamics and Control, 75, 114-121.

Huang, N. E., et al. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of the Royal Society A, 454(1971), 903-995.

Percival, D. B., & Walden, A. T. (2000). Wavelet Methods for Time Series Analysis. Cambridge University Press.

Piironen, J., & Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. Electronic Journal of Statistics, 11(2), 5018-5051.

See Also

- `signal_analysis`: Master function for complete analysis
- `filter_wavelet`: Wavelet multiresolution analysis
- `filter_emd`: Empirical mode decomposition
- `filter_hpgc`: Grant-Chan HP filter
- `fit_horseshoe`: Regularized Horseshoe regression

`apply_to_columns` *Apply Function to Matrix Columns*

Description

Applies a univariate filtering or transformation function to each column of a matrix and returns a consolidated data frame. This utility enables batch processing of panel data where each column represents a different variable or series.

Usage

```
apply_to_columns(X, FUN, extract = NULL, ..., verbose = FALSE)
```

Arguments

`X`	Matrix or data frame where each column is a series to process.
`FUN`	Function to apply to each column. Must accept a numeric vector and return either a numeric vector of the same length or a list with a named element (specified by `extract`).
`extract`	Character string specifying which element to extract from the function output if it returns a list. Default is NULL (use raw output).
`...`	Additional arguments passed to FUN.
`verbose`	Logical indicating whether to print progress messages.

Value

A data frame with the same number of rows as X, containing the processed output for each column.

Examples

```
X <- matrix(rnorm(200), ncol = 4)
colnames(X) <- c("A", "B", "C", "D")
result <- apply_to_columns(X, function(x) cumsum(x))
```

`compute_entropy` *Compute Shannon Entropy*

Description

Calculates the Shannon entropy of a probability distribution or, when applied to loadings, the entropy of the squared normalized loadings. High entropy indicates diffuse/uniform distribution (systemic noise), while low entropy indicates concentrated structure.

Usage

```
compute_entropy(x, base = 2, normalize = FALSE)
```

Arguments

`x`	Numeric vector. Will be squared and normalized to form a probability distribution.
`base`	Base of the logarithm. Default is 2 (bits).
`normalize`	Logical. If TRUE, returns normalized entropy (0 to 1 scale).

Details

The Shannon entropy is defined as:

$$H(p) = -\sum_i p_i \log(p_i)$$

where p_i are the probabilities. For factor loadings, we use squared normalized loadings as the probability distribution:

$$p_i = \lambda_i^2 / \sum_j \lambda_j^2$$

This measures the concentration of explanatory power across variables. Maximum entropy occurs when all loadings are equal (diffuse structure); minimum entropy occurs when a single variable dominates (concentrated structure).

Value

Numeric scalar representing entropy value.

Interpretation in Signal Analysis

In the context of latent structure extraction:

- **High entropy (near maximum)**: Suggests "maximum entropy systemic stochasticity" - the component captures diffuse, undifferentiated movement across all variables (akin to Brownian motion).
- **Low entropy**: Suggests "differentiated latent structure" - the component is driven by a subset of variables, indicating meaningful structural relationships.

Examples

```
uniform_loadings <- rep(1, 10)
compute_entropy(uniform_loadings, normalize = TRUE)

concentrated_loadings <- c(10, rep(0.1, 9))
compute_entropy(concentrated_loadings, normalize = TRUE)
```

---

estimate_dfm *Dynamic Factor Model Estimation*

Description

Estimates a Dynamic Factor Model (DFM) to extract common latent factors from panel data. Uses principal components as initial estimates and optionally refines via EM algorithm.

Usage

```
estimate_dfm(
  X,
  r = NULL,
  p = 1,
  ic = c("IC2", "IC1", "IC3"),
  max_factors = NULL,
  standardize = TRUE,
  verbose = FALSE
)
```

Arguments

`X`	Matrix or data frame where rows are observations and columns are variables.
`r`	Number of factors. If NULL, determined by information criterion.
`p`	Number of lags in factor VAR dynamics. Default 1.
`ic`	Character string specifying information criterion for factor selection: "IC1", "IC2", or "IC3" (Bai & Ng, 2002). Default "IC2".
`max_factors`	Maximum number of factors to consider. Default min(10, floor(ncol(X)/2)).
`standardize`	Logical. Standardize variables before estimation. Default TRUE.
`verbose`	Logical for progress messages.

Details

The DFM assumes:

$$X_{it} = \lambda_i' F_t + e_{it}$$

where F_t are common factors, λ_i are loadings, and e_{it} are idiosyncratic errors. The factors follow VAR dynamics:

$$F_t = A_1 F_{t-1} + ... + A_p F_{t-p} + u_t$$

Factor selection uses the Bai & Ng (2002) information criteria which penalize over-fitting while consistently estimating the true number of factors.

Value

A list of class "signaly_dfm" containing:

factors Matrix of estimated latent factors (T x r)

loadings Matrix of factor loadings (p x r)

var_coefficients VAR coefficient matrices for factor dynamics

idiosyncratic_var Idiosyncratic variance estimates

r_selected Number of factors selected

ic_values Information criterion values

fitted_values Fitted values from the model

residuals Residuals (idiosyncratic components)

References

Bai, J., & Ng, S. (2002). Determining the number of factors in approximate factor models. Econometrica, 70(1), 191-221.

Stock, J. H., & Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. Journal of the American Statistical Association, 97(460), 1167-1179.

Examples

```
set.seed(123)
n <- 100
p <- 20
X <- matrix(rnorm(n * p), ncol = p)
result <- estimate_dfm(X, r = 2)
print(dim(result$factors))
```

`filters` *Signal Filtering Methods for Trend Extraction*

Description

This module implements three complementary spectral decomposition methods for extracting latent trend signals from time series: wavelet multiresolution analysis, empirical mode decomposition, and the Grant-Chan embedded Hodrick-Prescott filter.

`filter_all` *Apply Multiple Filters to a Series*

Description

Convenience function that applies all three filtering methods (wavelet, EMD, HP-GC) to a time series and returns a consolidated comparison of results.

Usage

```
filter_all(
  y,
  wavelet_wf = "la8",
  wavelet_J = 4,
  wavelet_levels = c(3, 4),
  hpgc_prior = "weak",
  hpgc_chains = 4,
  hpgc_iterations = 20000,
  hpgc_burnin = 5000,
  verbose = FALSE
)
```

Arguments

`y`	Numeric vector of the time series.
`wavelet_wf`	Wavelet filter for wavelet decomposition. Default "la8".
`wavelet_J`	Wavelet decomposition depth. Default 4.
`wavelet_levels`	Levels to combine for wavelet trend. Default c(3, 4).
`hpgc_prior`	Prior configuration for HP-GC. Default "weak".
`hpgc_chains`	Number of MCMC chains. Default 4.
`hpgc_iterations`	MCMC iterations. Default 20000.
`hpgc_burnin`	MCMC burn-in. Default 5000.
`verbose`	Logical for progress messages.

Value

A list of class "signaly_multifilter" containing results from all three methods and a comparison data frame.

Examples

```
y <- cumsum(rnorm(100)) + sin(seq(0, 4*pi, length.out = 100))
result <- filter_all(y, hpgc_iterations = 5000, hpgc_burnin = 1000)
```

filter_emd *Empirical Mode Decomposition Filter*

Description

Applies Empirical Mode Decomposition (EMD) to extract intrinsic mode functions (IMFs) from a time series. Unlike Fourier or wavelet methods, EMD is fully data-adaptive and does not require pre-specified basis functions, making it suitable for non-stationary and non-linear signals.

Usage

```
filter_emd(
  y,
  boundary = "periodic",
  max_imf = NULL,
  stop_rule = "type1",
  tol = NULL,
  max_sift = 20,
  verbose = FALSE
)
```

Arguments

`y`	Numeric vector of the time series to decompose.
`boundary`	Character string specifying boundary handling: "periodic" (default), "symmetric", "none", or "wave".
`max_imf`	Maximum number of IMFs to extract. If NULL, extraction continues until the residue is monotonic.
`stop_rule`	Character string specifying the stopping criterion for sifting: "type1" (default), "type2", "type3", "type4", or "type5".
`tol`	Tolerance for sifting convergence. Default is `sd(y) * 0.1^2`.
`max_sift`	Maximum number of sifting iterations per IMF. Default is 20.
`verbose`	Logical indicating whether to print diagnostic messages.

Details

EMD decomposes a signal x(t) into a sum of Intrinsic Mode Functions (IMFs) and a residue:

$$x(t) = \sum_{j=1}^{n} c_j(t) + r_n(t)$$

where each IMF $c_j(t)$ satisfies two conditions:

1. The number of extrema and zero crossings differ by at most one
2. The mean of upper and lower envelopes is zero at each point

The sifting process iteratively extracts IMFs from highest to lowest frequency until the residue becomes monotonic (representing the trend).

Value

A list of class "signaly_emd" containing:

trend Numeric vector of the extracted trend (original minus residue)

residue Numeric vector of the EMD residue (monotonic trend)

imfs Matrix where each column is an IMF, ordered from highest to lowest frequency

n_imfs Number of IMFs extracted

original Original input series

settings List of parameters used

diagnostics List with IMF statistics

Advantages over Fourier/Wavelet Methods

- **Adaptive basis**: IMFs are derived from the data itself, not pre-specified
- **Handles non-stationarity**: Instantaneous frequency can vary over time
- **Handles non-linearity**: No assumption of linear superposition
- **Preserves local structure**: Better time localization than Fourier methods

Limitations

- **Mode mixing**: Different scales may appear in the same IMF
- **End effects**: Boundary conditions can cause artifacts
- **No formal theory**: Unlike wavelets, lacks rigorous mathematical foundation
- **Reproducibility**: Results can vary with stopping criteria

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C., & Liu, H. H. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of the Royal Society A, 454(1971), 903-995.

Wu, Z., & Huang, N. E. (2009). Ensemble empirical mode decomposition: A noise-assisted data analysis method. Advances in Adaptive Data Analysis, 1(1), 1-41.

See Also

`emd`, `filter_wavelet`, `filter_hpgc`

Examples

```
set.seed(123)
t <- seq(0, 10, length.out = 200)
y <- sin(2*pi*t) + 0.5*sin(8*pi*t) + 0.1*rnorm(200)
result <- filter_emd(y)
plot(y, type = "l", col = "gray")
lines(result$trend, col = "red", lwd = 2)
```

filter_hpgc *Grant-Chan Embedded Hodrick-Prescott Filter*

Description

Implements the Bayesian Hodrick-Prescott filter embedded in an unobserved components model, as developed by Grant and Chan (2017). This approach provides principled uncertainty quantification for the extracted trend through Markov Chain Monte Carlo sampling.

Usage

```
filter_hpgc(
  y,
  prior_config = "weak",
  n_chains = 4,
  iterations = 20000,
  burnin = 5000,
  verbose = FALSE
)
```

Arguments

`y`	Numeric vector of the time series. Will be internally scaled for numerical stability.
`prior_config`	Character string or list specifying prior configuration. Options: "weak" (default), "informative", or "empirical". Alternatively, a named list with prior parameters (see Details).
`n_chains`	Integer number of MCMC chains to run. Default is 4.
`iterations`	Integer total number of MCMC iterations per chain. Default is 20000.
`burnin`	Integer number of burn-in iterations to discard. Default is 5000.
`verbose`	Logical indicating whether to print progress messages.

Details

The Grant-Chan model decomposes the observed series y_t as:

$$y_t = \tau_t + c_t$$

where τ_t is the trend component and c_t is the cyclical component.

Trend Model (Second-Order Markov Process):

$$\Delta^2 \tau_t = u_t^\tau, \quad u_t^\tau \sim N(0, \sigma_\tau^2)$$

This implies the trend growth rate follows a random walk, allowing for time-varying trend growth.

Cycle Model (Stationary AR(2)):

$$c_t = \phi_1 c_{t-1} + \phi_2 c_{t-2} + u_t^c, \quad u_t^c \sim N(0, \sigma_c^2)$$

with stationarity constraints on ϕ.

Value

A list of class "signaly_hpgc" containing:

- `trend`: Numeric vector of posterior mean trend
- `trend_lower`: Numeric vector of 2.5 percent posterior quantile
- `trend_upper`: Numeric vector of 97.5 percent posterior quantile
- `cycle`: Numeric vector of posterior mean cycle component
- `cycle_lower`: Numeric vector of 2.5 percent posterior quantile
- `cycle_upper`: Numeric vector of 97.5 percent posterior quantile
- `draws`: List of posterior draws for all parameters
- `diagnostics`: Convergence diagnostics including R-hat and ESS
- `dic`: Deviance Information Criterion
- `settings`: Parameters used in the analysis

Prior Configurations

weak Diffuse priors allowing data to dominate. Good for initial exploration.

informative Tighter priors based on typical macroeconomic dynamics. Suitable when strong smoothness is desired.

empirical Priors calibrated from data moments. Balances flexibility with data-driven regularization.

Custom priors can be specified as a list with elements:

- `phi_mu`: Mean of phi prior (2-vector)
- `phi_v_i`: Precision matrix for phi prior (2x2)
- `gamma_mu`: Mean of gamma (initial trend growth) prior
- `gamma_v_i`: Precision matrix for gamma prior
- `s_tau`: Upper bound for uniform prior on σ^2_τ
- `s_c_shape`: Shape parameter for inverse-gamma prior on σ^2_c
- `s_c_rate`: Rate parameter for inverse-gamma prior on σ^2_c

Relationship to Standard HP Filter

The standard HP filter solves:

$$\min_\tau \sum_t (y_t - \tau_t)^2 + \lambda \sum_t (\Delta^2 \tau_t)^2$$

The Grant-Chan approach embeds this within a probabilistic model where $\lambda = \sigma^2_c/\sigma^2_\tau$, allowing this ratio to be estimated from data with full uncertainty quantification.

References

Grant, A. L., & Chan, J. C. C. (2017). Reconciling output gaps: Unobserved components model and Hodrick-Prescott filter. Journal of Economic Dynamics and Control, 75, 114-121. doi:10.1016/j.jedc.2016.12.007

Chan, J., Koop, G., Poirier, D. J., & Tobias, J. L. (2019). Bayesian Econometric Methods (2nd ed.). Cambridge University Press.

See Also

`filter_wavelet`, `filter_emd`

Examples

```
set.seed(123)
y <- cumsum(rnorm(100)) + sin(seq(0, 4*pi, length.out = 100))
result <- filter_hpgc(y, prior_config = "weak", n_chains = 2,
                      iterations = 5000, burnin = 1000)
plot(y, type = "l", col = "gray")
lines(result$trend, col = "red", lwd = 2)
```

---

filter_wavelet   *Wavelet Multiresolution Analysis Filter*

Description

Performs wavelet-based signal decomposition using the Maximal Overlap Discrete Wavelet Transform (MODWT) to extract trend components at specified frequency scales. This method decomposes the signal into detail coefficients (D1, D2, ..., DJ) capturing progressively lower frequencies and a smooth coefficient (SJ) representing the underlying trend.

Usage

```
filter_wavelet(
  y,
  wf = "la8",
  J = 4,
  boundary = "periodic",
  levels_to_combine = c(3, 4),
  first_difference = FALSE,
  verbose = FALSE
)
```

Arguments

`y`	Numeric vector of the time series to decompose. Length must be at least `2^J`.
`wf`	Character string specifying the wavelet filter. Options include "la8" (least asymmetric with 8 vanishing moments, 16 coefficients), "la16", "la20", "haar", "d4", "d6", "d8", etc. Default is "la8".
`J`	Integer specifying the decomposition depth (number of levels). Default is 4, yielding D1-D4 detail levels plus S4 smooth level.
`boundary`	Character string specifying boundary handling: "periodic" (default) or "reflection".
`levels_to_combine`	Integer vector specifying which detail levels to combine for the trend estimate. Default is `c(3, 4)` for D3+D4.
`first_difference`	Logical. If TRUE, applies wavelet to first differences and reconstructs via cumulative sum. Default is FALSE.
`verbose`	Logical indicating whether to print diagnostic messages.

Details

The MODWT (Maximal Overlap Discrete Wavelet Transform) is preferred over the classical DWT for several reasons relevant to signal extraction:

1. **Translation invariance**: Unlike DWT, MODWT does not depend on the starting point of the series, producing consistent results regardless of circular shifts.
2. **Any sample size**: MODWT can be applied to series of any length, not just powers of 2.
3. **Additive decomposition**: The MRA (multiresolution analysis) coefficients sum exactly to the original series.

The choice of wavelet filter affects the trade-off between time and frequency localization:

- **la8 (Daubechies least asymmetric, 8 vanishing moments)**: Good balance of smoothness and localization, recommended for economic data.
- **Higher order (la16, la20)**: Better frequency resolution at cost of temporal smearing.
- **haar**: Maximum time localization but poor frequency resolution.

Value

A list of class "signaly_wavelet" containing:

trend Numeric vector of the extracted trend component

mra Full multiresolution analysis object from waveslim::mra

detail_levels Data frame with all detail level coefficients

smooth_level Vector of the smooth (SJ) coefficients

combined_levels Character string indicating which levels were combined

settings List of parameters used in the analysis

diagnostics List with variance decomposition and energy distribution

Frequency Interpretation

For a series with unit sampling interval, the detail levels correspond to approximate frequency bands:

- D1: periods 2-4 (highest frequency noise)
- D2: periods 4-8 (short-term fluctuations)
- D3: periods 8-16 (medium-term cycles)
- D4: periods 16-32 (longer cycles)
- S4: periods > 32 (smooth trend)

For annual economic data, D3+D4 typically captures business cycle dynamics (8-32 year periods), while D1+D2 captures short-term noise.

References

Daubechies, I. (1992). Ten Lectures on Wavelets. SIAM.

Percival, D. B., & Walden, A. T. (2000). Wavelet Methods for Time Series Analysis. Cambridge University Press.

Gencay, R., Selcuk, F., & Whitcher, B. (2002). An Introduction to Wavelets and Other Filtering Methods in Finance and Economics. Academic Press.

See Also

`mra`, `filter_emd`, `filter_hpgc`

Examples

```
set.seed(123)
y <- cumsum(rnorm(100)) + sin(seq(0, 4*pi, length.out = 100))
result <- filter_wavelet(y, wf = "la8", J = 4)
plot(y, type = "l", col = "gray")
lines(result$trend, col = "red", lwd = 2)
```

`fit_horseshoe` *Fit Regularized Horseshoe Regression Model*

Description

Fits a Bayesian linear regression with regularized Horseshoe prior using Stan via cmdstanr. This version includes improved numerical stability and automatic prior calibration.

Usage

```
fit_horseshoe(
  y,
  X,
  var_names = NULL,
  p0 = NULL,
  slab_scale = 3,
  slab_df = 4,
  tau_scale = NULL,
  use_qr = FALSE,
  standardize = TRUE,
  X_new = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  adapt_delta = 0.95,
  max_treedepth = 12,
  seed = 123,
  verbose = TRUE
)
```

Arguments

`y`	Numeric vector of the response variable.
`X`	Matrix or data frame of predictor variables.
`var_names`	Optional character vector of variable names.
`p0`	Expected number of non-zero coefficients. Default: P/3.
`slab_scale`	Scale for the regularizing slab. Default: 3.
`slab_df`	Degrees of freedom for the slab. Default: 4.
`tau_scale`	Scale multiplier for the global shrinkage prior. Default: NULL (auto-calibrated based on data characteristics). Increase this value (e.g., 10-20) if the model over-shrinks.
`use_qr`	Use QR decomposition? Default: FALSE.
`standardize`	Standardize predictors internally? Default: TRUE.
`X_new`	Optional matrix for out-of-sample prediction.
`iter_warmup`	Warmup iterations per chain. Default: 1000.
`iter_sampling`	Sampling iterations per chain. Default: 1000.
`chains`	Number of MCMC chains. Default: 4.
`adapt_delta`	Target acceptance probability. Default: 0.95.
`max_treedepth`	Maximum tree depth. Default: 12.
`seed`	Random seed.
`verbose`	Print progress messages?

Details

The regularized Horseshoe prior (Piironen & Vehtari, 2017) provides adaptive shrinkage that can distinguish between relevant and irrelevant predictors.

Variable Selection Methods:

After fitting, variables can be selected using different criteria:

- `select_by_credible_interval`: Selects variables whose credible interval excludes zero. **Recommended** - most robust method.
- `select_by_shrinkage`: Selects based on kappa (shrinkage factor). May underselect when tau is very small.
- `select_by_magnitude`: Selects based on coefficient magnitude.

Note on kappa-based selection:

The shrinkage factor kappa depends on the global shrinkage parameter tau. In some datasets, the posterior of tau may concentrate near zero, causing all kappa values to be close to 1 even for truly relevant variables. When this happens, the coefficient estimates (beta) remain valid, but kappa-based selection will fail. The function automatically warns when this occurs and recommends using `select_by_credible_interval()` instead.

Value

A list of class "signaly_horseshoe" with posterior summaries, diagnostics, and model fit object.

`horseshoe` *Regularized Horseshoe Regression for Variable Selection*

Description

Implements Bayesian sparse regression using the regularized Horseshoe prior for identifying structurally relevant predictors from high-dimensional candidate variable sets.

`iplot` *Interactive Plot for Signal Analysis*

Description

Generates an interactive dashboard using plotly to explore the results of the signal analysis. Allows zooming, panning, and toggling traces.

Usage

```
iplot(x)
```

Arguments

`x`	An object of class `signal_analysis`

Value

A plotly object (HTML widget).

`pca_bootstrap` *Principal Component Analysis with Bootstrap Significance Testing*

Description

Performs PCA on panel data with bootstrap-based significance testing for factor loadings. Identifies which variables load significantly on each principal component using a null distribution constructed via block bootstrapping.

Usage

```
pca_bootstrap(
  X,
  n_components = NULL,
  center = TRUE,
  scale = TRUE,
  n_boot = 200,
  block_length = NULL,
  alpha = 0.05,
  use_fdr = FALSE,
  rotation = c("varimax", "none", "oblimin"),
  verbose = FALSE
)
```

Arguments

`X`	Matrix or data frame where rows are observations (time points) and columns are variables.
`n_components`	Number of principal components to extract. If NULL, determined by eigenvalue threshold or explained variance.
`center`	Logical. Center variables before PCA. Default TRUE.
`scale`	Logical. Scale variables to unit variance. Default TRUE.
`n_boot`	Number of bootstrap replications for significance testing. Default 200.
`block_length`	Block length for block bootstrap. If NULL, defaults to `ceiling(sqrt(nrow(X)))`.
`alpha`	Significance level for loading tests. Default 0.05.
`use_fdr`	Logical. Apply Benjamini-Hochberg FDR correction. Default FALSE.
`rotation`	Character string specifying rotation method: "none", "varimax", or "oblimin". Default "varimax".
`verbose`	Logical for progress messages.

Details

The analysis proceeds in several stages:

1. Standard PCA: Eigendecomposition of the correlation (if scaled) or covariance matrix to extract principal components.

2. Rotation (optional): Varimax rotation maximizes the variance of squared loadings within components, producing cleaner simple structure. Oblimin allows correlated factors.

3. Bootstrap Significance Testing: For each bootstrap replicate:

1. Resample rows using block bootstrap (preserving temporal dependence)
2. Perform PCA on resampled data
3. Apply Procrustes rotation to align with original
4. Record absolute loadings

The empirical p-value for each loading is the proportion of bootstrap loadings exceeding the original in absolute value.

4. Entropy Calculation: Shannon entropy of squared loadings indicates whether explanatory power is concentrated (low entropy) or diffuse (high entropy). High entropy on PC1 suggests systemic co-movement rather than differentiated structure.

Value

A list of class "signaly_pca" containing:

loadings Matrix of factor loadings (rotated if specified)

scores Matrix of component scores

eigenvalues Vector of eigenvalues

variance_explained Proportion of variance explained by each component

cumulative_variance Cumulative proportion of variance explained

significant_loadings Matrix of logical values indicating significance

p_values Matrix of bootstrap p-values for loadings

thresholds Cutoff values for significance by component

entropy Shannon entropy of loadings for each component

summary_by_component Data frame summarizing each component

assignments Data frame mapping variables to their dominant component

Interpretation in Signal Analysis

- **High PC1 entropy**: "Maximum entropy systemic stochasticity" - the dominant factor captures undifferentiated movement, suggesting noise rather than latent structure.
- **Low PC1 entropy**: "Differentiated latent structure" - specific variables dominate, indicating meaningful groupings.
- **Significant loadings**: Variables with p < alpha after bootstrap testing reliably load on that component.

References

Jolliffe, I. T. (2002). Principal Component Analysis (2nd ed.). Springer.

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. Psychometrika, 23(3), 187-200.

Examples

```
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), ncol = p)
colnames(X) <- paste0("V", 1:p)
result <- pca_bootstrap(X, n_components = 3, n_boot = 50)
print(result$summary_by_component)
```

`pca_dfm` *Principal Component Analysis and Dynamic Factor Models*

Description

Implements dimensionality reduction techniques for panel data including PCA with bootstrap significance testing and Dynamic Factor Models (DFM) for extracting common latent factors.

`plot.signal_analysis` *Plot Method for signal_analysis Objects*

Description

Generate diagnostic visualizations for signal analysis results.

Usage

```
## S3 method for class 'signal_analysis'
plot(x, which = "all", ask = NULL, ...)
```

Arguments

`x`	An object of class `signal_analysis`
`which`	Character vector specifying which plots to create. Options: `"all"`, `"filters"`, `"horseshoe"`, `"pca"`, `"dfm"`, `"unitroot"`. Default is `"all"`.
`ask`	Logical, whether to prompt before each plot (default: TRUE in interactive mode)
`...`	Additional arguments passed to plotting functions

Value

Invisibly returns the input object

print.signal_analysis *Print Method for signal_analysis Objects*

Description

Print a concise summary of signal analysis results.

Usage

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

Arguments

`x`	An object of class `signal_analysis`
`...`	Additional arguments (ignored)

Value

Invisibly returns the input object

select_by_credible_interval *Select Variables Based on Credible Intervals*

Description

Robust variable selection method using posterior credible intervals. A variable is selected if its credible interval excludes zero, indicating a statistically meaningful effect. This method is more robust than kappa-based selection when the global shrinkage parameter tau collapses.

Usage

```
select_by_credible_interval(hs_fit, prob = 0.95, verbose = FALSE)
```

Arguments

`hs_fit`	Object returned by `fit_horseshoe`.
`prob`	Probability level for the credible interval. Default 0.95 uses the 95% credible interval (2.5% to 97.5% quantiles).
`verbose`	Logical for messages.

Details

This function selects variables whose posterior credible interval does not include zero. This is analogous to checking if a confidence interval excludes zero in frequentist statistics, but with a Bayesian interpretation.

The method is particularly useful when:

- The kappa-based selection returns no variables
- The posterior tau is very small (< 0.05)
- You want a more interpretable selection criterion

Value

List with selected variable names and details.

See Also

`select_by_shrinkage` for kappa-based selection.

Examples

```
## Not run:
hs_fit <- fit_horseshoe(y, X, p0 = 5)
selected <- select_by_credible_interval(hs_fit, prob = 0.95, verbose = TRUE)
print(selected$selected)

## End(Not run)
```

`select_by_magnitude` *Select Variables Based on Effect Magnitude*

Description

Simple variable selection based on the magnitude of posterior mean coefficients. Useful as a quick screening method.

Usage

```
select_by_magnitude(hs_fit, threshold = 0.1, verbose = FALSE)
```

Arguments

`hs_fit`	Object returned by `fit_horseshoe`.
`threshold`	Minimum absolute coefficient value. Default 0.1.
`verbose`	Logical for messages.

Value

List with selected variable names and details.

`select_by_shrinkage` *Select Variables Based on Shrinkage*

Description

Variable selection method using shrinkage factors (kappa). Note: This method may underselect when tau collapses to small values. Consider using `select_by_credible_interval` as an alternative.

Usage

```
select_by_shrinkage(hs_fit, threshold = 0.5, verbose = FALSE)
```

Arguments

`hs_fit`	Object returned by `fit_horseshoe`.
`threshold`	Kappa threshold. Variables with kappa < threshold are considered relevant. Default 0.5.
`verbose`	Logical for messages.

Value

List with selected variable names and details.

See Also

`select_by_credible_interval` for a more robust alternative.

`signal_analysis` *Comprehensive Signal Analysis for Panel Data*

Description

Master function that orchestrates the complete signal extraction pipeline, integrating spectral decomposition (wavelets, EMD, HP-GC), Bayesian variable' selection (regularized Horseshoe), dimensionality reduction (PCA, DFM), and stationarity testing into a unified analytical framework.

The function constructs a target signal Y from candidate variables X in panel data and applies multiple complementary methodologies to extract the latent structure from phenomenological dynamics.

Usage

```
signal_analysis(
  data,
  y_formula,
  time_var = NULL,
  group_var = NULL,
  methods = "all",
  filter_config = list(),
  horseshoe_config = list(),
  pca_config = list(),
  dfm_config = list(),
  unitroot_tests = "all",
  na_action = c("interpolate", "omit", "fail"),
  standardize = TRUE,
  first_difference = FALSE,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

`data`	A data.frame or matrix containing the panel data. For data.frames, time should be in rows and variables in columns.
`y_formula`	Formula specifying how to construct Y from X variables, or a character string naming the pre-constructed Y column in data.
`time_var`	Character string naming the time variable (optional, assumes rows are ordered by time if NULL).
`group_var`	Character string naming the group/panel variable for panel data (optional for single time series).
`methods`	Character vector specifying which methods to apply. Options: `"wavelet"`, `"emd"`, `"hpgc"`, `"horseshoe"`, `"pca"`, `"dfm"`, `"unitroot"`, or `"all"` (default).
`filter_config`	List of configuration options for filtering methods: **wavelet_filter** Wavelet filter type (default: "la8") **wavelet_levels** Which detail levels to combine (default: c(3,4)) **emd_max_imf** Maximum IMFs for EMD (default: 10) **hpgc_prior** Prior configuration: "weak", "informative", "empirical" (default: "weak") **hpgc_chains** Number of MCMC chains (default: 4) **hpgc_iterations** Total iterations per chain (default: 20000)
`horseshoe_config`	List of configuration for Horseshoe regression: **p0** Expected number of relevant predictors (default: NULL for auto) **chains** Number of MCMC chains (default: 4) **iter_sampling** Sampling iterations per chain (default: 2000) **iter_warmup** Warmup iterations (default: 1000)

	adapt_delta Target acceptance rate (default: 0.95) **use_qr** Use QR decomposition (default: TRUE) **kappa_threshold** Shrinkage threshold for selection (default: 0.5)
`pca_config`	List of configuration for PCA: **n_components** Number of components (default: NULL for auto) **rotation** Rotation method: "none", "varimax", "oblimin" (default: "none") **n_boot** Bootstrap replications (default: 1000) **block_length** Block length for bootstrap (default: NULL for auto) **alpha** Alpha for bootstrap tests (default: 0.05)
`dfm_config`	List of configuration for Dynamic Factor Models: **r** Number of factors (default: NULL for auto via IC) **max_factors** Maximum factors to consider (default: 10) **p** VAR lags for factor dynamics (default: 1) **ic** Information criterion: "IC1", "IC2", "IC3" (default: "bai_ng_2")
`unitroot_tests`	Character vector of unit root tests to apply. Options: `"adf"`, `"ers"`, `"kpss"`, `"pp"`, or `"all"` (default).
`na_action`	How to handle missing values: "interpolate", "omit", "fail" (default: "interpolate").
`standardize`	Logical, whether to standardize variables before analysis (default: TRUE).
`first_difference`	Logical, whether to first-difference data (default: FALSE).
`verbose`	Logical, whether to print progress messages (default: TRUE).
`seed`	Random seed for reproducibility (default: NULL).

Details

Methodological Framework

The signal extraction pipeline distinguishes between latent structure (the underlying data-generating process) and phenomenological dynamics (observed variability). This is achieved through:

1. **Spectral Decomposition**: Separates signal frequencies
   - Wavelets: Multi-resolution analysis via MODWT
   - EMD: Data-adaptive decomposition into intrinsic modes
   - HP-GC: Bayesian unobserved components (trend + cycle)
2. **Sparse Regression**: Identifies relevant predictors
   - Regularized Horseshoe: Adaptive shrinkage with slab regularization
   - Shrinkage factors (kappa) quantify predictor relevance
3. **Dimensionality Reduction**: Extracts common factors
   - PCA: Static factor structure with bootstrap significance
   - DFM: Dynamic factors with VAR transition dynamics
4. **Stationarity Testing**: Characterizes persistence properties

- Integrated battery of ADF, ERS, KPSS, PP tests
- Synthesized conclusion on stationarity type

Interpretation Framework

The automated interpretation assesses:

- **Signal Smoothness**: Variance of second differences
- **Trend Persistence**: Deterministic vs. stochastic via unit roots
- **Information Topology**: Entropy of PC1 loadings (concentrated vs. diffuse)
- **Sparsity Ratio**: Proportion of predictors shrunk to zero
- **Factor Structure**: Number of significant common factors

Value

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

call The matched function call

data Processed input data

Y The constructed target signal

X The predictor matrix

filters Results from spectral decomposition methods

horseshoe Results from Bayesian variable selection

pca Results from PCA with bootstrap

dfm Results from Dynamic Factor Model

unitroot Results from unit root tests

interpretation Automated technical interpretation

config Configuration parameters used

References

Piironen, J., & Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. Electronic Journal of Statistics, 11(2), 5018-5051. doi:10.1214/17EJS1337SI

Bai, J., & Ng, S. (2002). Determining the Number of Factors in Approximate Factor Models. Econometrica, 70(1), 191-221. doi:10.1111/14680262.00273

See Also

`filter_wavelet`, `filter_emd`, `filter_hpgc`, `fit_horseshoe`, `pca_bootstrap`, `estimate_dfm`, `test_unit_root`

Examples

```
# Generate example panel data
set.seed(42)
n_time <- 50
n_vars <- 10

# Create correlated predictors with common factor structure
factors <- matrix(rnorm(n_time * 2), n_time, 2)
loadings <- matrix(runif(n_vars * 2, -1, 1), n_vars, 2)
X <- factors %*% t(loadings) + matrix(rnorm(n_time * n_vars, 0, 0.5), n_time, n_vars)
colnames(X) <- paste0("X", 1:n_vars)

# True signal depends on only 3 predictors
true_beta <- c(rep(1, 3), rep(0, 7))
Y <- X %*% true_beta + rnorm(n_time, 0, 0.5)

# Combine into data frame
data <- data.frame(Y = Y, X)

# Run comprehensive analysis
# We pass specific configs to make MCMC very fast just for the example
result <- signal_analysis(
  data = data,
  y_formula = "Y",
  methods = "all",
  verbose = TRUE,
  # Configuration for speed (CRAN policy < 5s preferred)
  filter_config = list(
     hpgc_chains = 1,
     hpgc_iterations = 50,
     hpgc_burnin = 10
  ),
  horseshoe_config = list(
     chains = 1,
     iter_sampling = 50,
     iter_warmup = 10
  ),
  pca_config = list(
     n_boot = 50
  )
)

# View interpretation
print(result)

# Plot results
plot(result)
```

`summary.signal_analysis` *Summary Method for signal_analysis Objects*

Description

Generate a detailed summary of signal analysis results.

Usage

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

Arguments

`object`	An object of class `signal_analysis`
`...`	Additional arguments (ignored)

Value

A list containing detailed summaries (invisibly)

`test_unit_root` *Comprehensive Unit Root Test Suite*

Description

Applies multiple unit root and stationarity tests to a time series, providing an integrated assessment of persistence properties. Implements Augmented Dickey-Fuller (ADF), Elliott-Rothenberg-Stock (ERS), Kwiatkowski-Phillips-Schmidt-Shin (KPSS), and Phillips-Perron tests.

Usage

```
test_unit_root(y, max_lags = NULL, significance_level = 0.05, verbose = FALSE)
```

Arguments

`y`	Numeric vector of the time series to test.
`max_lags`	Maximum number of lags for ADF-type tests. If NULL, defaults to `floor(12 * (length(y)/100)^0.25)`.
`significance_level`	Significance level for hypothesis testing. Default is 0.05.
`verbose`	Logical indicating whether to print detailed results.

Details

The battery of tests addresses different null hypotheses and specifications:

Augmented Dickey-Fuller (ADF) tests the null of a unit root against the alternative of stationarity. Three specifications are tested:

- **none**: No constant, no trend (random walk)
- **drift**: Constant included (random walk with drift)
- **trend**: Constant and linear trend

Elliott-Rothenberg-Stock (ERS) tests provide more power than ADF by using GLS detrending. Two variants:

- **DF-GLS**: GLS-detrended Dickey-Fuller test
- **P-test**: Point-optimal test

KPSS reverses the hypotheses: null is stationarity, alternative is unit root. This allows testing the stationarity hypothesis directly.

Phillips-Perron uses non-parametric corrections for serial correlation, avoiding lag selection issues.

Value

A list of class "signaly_unitroot" containing:

adf Results from ADF tests (none, drift, trend specifications)

ers Results from ERS tests (DF-GLS and P-test)

kpss Results from KPSS tests (level and trend)

pp Results from Phillips-Perron tests

summary Data frame summarizing all test results

conclusion Integrated conclusion about stationarity

persistence_type Classification: stationary, trend-stationary, difference-stationary, or inconclusive

Interpretation Strategy

The function synthesizes results using the following logic:

1. If ADF/ERS reject unit root AND KPSS fails to reject stationarity: Series is likely **stationary**
2. If ADF/ERS fail to reject AND KPSS rejects stationarity: Series likely has **unit root** (difference-stationary)
3. If only trend-ADF rejects: Series is likely **trend-stationary**
4. Conflicting results indicate **inconclusive** or structural breaks

References

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. Journal of the American Statistical Association, 74(366), 427-431.

Elliott, G., Rothenberg, T. J., & Stock, J. H. (1996). Efficient Tests for an Autoregressive Unit Root. Econometrica, 64(4), 813-836.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. Journal of Econometrics, 54(1-3), 159-178.

Phillips, P. C. B., & Perron, P. (1988). Testing for a unit root in time series regression. Biometrika, 75(2), 335-346.

See Also

`ur.df`, `ur.ers`, `ur.kpss`, `ur.pp`

Examples

```
set.seed(123)
stationary <- arima.sim(list(ar = 0.5), n = 100)
result <- test_unit_root(stationary)
print(result$conclusion)

nonstationary <- cumsum(rnorm(100))
result2 <- test_unit_root(nonstationary)
print(result2$conclusion)
```

`unit_root` *Unit Root and Stationarity Tests*

Description

Comprehensive suite of unit root and stationarity tests for characterizing the persistence properties of time series and extracted signals.

Index