Package 'MacroFilters'

June 11, 2026

Title Robust Trend-Cycle Decomposition for Macroeconomic Time Series

Version 0.2.1

Description Provides high-performance tools for macroeconomic trend extraction and filtering, specifically designed to solve the end-point problem in real-time. Implements the MacroBoost Hybrid (MBH) filter using penalized P-splines and gradient boosting. Unlike the standard Hodrick-Prescott filter, 'MacroFilters' utilizes component-wise L2-boosting with robust loss functions (Huber) to handle extreme transient shocks (e.g., COVID-19) without inducing spurious trend shifts. The algorithm includes an automated two-layer diagnostic stage for unit roots and structural breaks, optimized via corrected AICc for computational efficiency. Methodology detailed in Kinel (2026) <doi:10.2139/ssrn.6371138>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

URL https://github.com/michal0091/MacroFilters, https://michal0091.github.io/MacroFilters/

BugReports https://github.com/michal0091/MacroFilters/issues

Imports data.table, ggplot2, Matrix, mboost, tseries

Suggests knitr, rmarkdown, scales, strucchange, testthat (>= 3.0.0), usethis, xts, zoo

VignetteBuilder knitr

Depends R (>= 3.5)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Michal Kinel [aut, cre] (ORCID: <https://orcid.org/0009-0007-3295-7199>)

Maintainer Michal Kinel <michal.kinel@gmail.com>

Repository CRAN

Date/Publication 2026-06-11 21:00:02 UTC

Contents

`autoplot.macrofilter` *Plot a macrofilter decomposition with ggplot2*

Description

Visualises any `macrofilter` object: the observed series (attenuated) over the estimated trend (emphasised). When the object carries bootstrap bands (`$trend_lower` / `$trend_upper`), a 95% confidence ribbon is drawn behind the lines automatically.

Usage

```
## S3 method for class 'macrofilter'
autoplot(object, ...)
```

Arguments

`object`	A `macrofilter` object.
`...`	Currently ignored; present for S3 generic compatibility.

Value

A `ggplot` object.

Examples

```
y <- ts(cumsum(rnorm(120)), start = c(2000, 1), frequency = 4)
autoplot(mbh_filter(y, mstop = 100L, boot_iter = 50L))
```

bhp_filter *Boosted HP Filter*

Description

Iteratively applies the Hodrick-Prescott filter on residuals to better capture stochastic trends. At each iteration the HP smoother is applied to the current residual and the resulting trend increment is added to the cumulative trend estimate. Iteration stops according to one of three rules: BIC minimisation (default), ADF stationarity test on residuals, or a fixed number of iterations.

Usage

```
bhp_filter(
  x,
  lambda = NULL,
  iter_max = 100L,
  stopping = c("bic", "adf", "fixed"),
  sig_level = 0.05,
  freq = NULL,
  boot_iter = 0,
  block_size = "auto"
)
```

Arguments

x	Numeric vector, ts, xts, or zoo object.
lambda	Smoothing parameter. If NULL (default), it is auto-detected using the Ravn-Uhlig rule (6.25 * freq^4).
iter_max	Integer. Maximum number of boosting iterations (default 100).
stopping	Character. Stopping rule: "bic" (default), "adf", or "fixed".
sig_level	Numeric. Significance level for the ADF test when stopping = "adf" (default 0.05).
freq	Numeric frequency override (1 = annual, 4 = quarterly, 12 = monthly). Used only when lambda is NULL and the frequency cannot be inferred from x.
boot_iter	Non-negative integer. Number of block-bootstrap iterations for uncertainty quantification (default 0, bootstrap disabled). When > 0, the function adds $trend_lower and $trend_upper: a 95% normal-approximation band, trend +/- 1.96 * sd(bootstrap trends), centred on the estimated trend. The bootstrap sd is used instead of empirical percentiles because it is smooth and stable at practical boot_iter. Each bootstrap refit uses the same mstop as the base fit, so larger boot_iter raises cost linearly. See also block_size.
block_size	Positive integer or "auto". Block length for the moving-block bootstrap (used only when boot_iter > 0). If "auto" (default), it is set to 2 * stats::frequency(x) (two full cycles), bounded above by floor(length(x) / 3) to keep at least three blocks.

Details

The boosted HP filter starts from the standard HP solution and then re-applies the same HP smoother to the residual (cycle) component. The trend increment from each pass is accumulated, and the procedure stops when one of the following criteria is met:

`"bic"` Schwarz information criterion computed as $n \log(\hat{\sigma}^2) + \log(n)\,\mathrm{tr}(S^m)$, where S^m is the iterated smoother. Iteration stops when the BIC increases relative to the previous best.

`"adf"` Augmented Dickey-Fuller test on the residual. Iteration stops when the residual is stationary at level `sig_level`.

`"fixed"` Runs exactly `iter_max` iterations.

Value

A list of class `c("macrofilter", "list")` with `trend`, `cycle`, `data`, and `meta` (`method = "bHP"`, `lambda`, `iterations`, `stopping_rule`, `compute_time`). When `boot_iter > 0` it also carries `trend_lower` and `trend_upper` (95% normal-approximation bootstrap band); each bootstrap refit runs a *fixed* `iterations` passes, conditioning on the complexity selected by the base fit.

References

Phillips, P.C.B. and Shi, Z. (2021). Boosting: Why You Can Use the HP Filter. *International Economic Review*, 62(2), 521–570.

Examples

```
# Quarterly GDP-like series
y <- ts(cumsum(rnorm(200)), start = c(2000, 1), frequency = 4)
result <- bhp_filter(y)
print(result)
```

---

es_gdp *Spain Real GDP — FRED Vintage*

Description

Quarterly Spain Real Gross Domestic Product from the Federal Reserve Bank of St. Louis (FRED) public data API (series **CLVMNACSCAB1GQES**), expressed in millions of chained 2015 EUR, seasonally adjusted. Original source: Eurostat/OECD National Accounts via FRED.

Usage

```
es_gdp
```

Format

A `data.table` with one row per quarter and three columns:

`date Date`. Quarter start date (e.g. `2000-01-01` = 2000 Q1).

`gdp_real numeric`. Real GDP level, millions of chained 2015 EUR.

`gdp_log numeric`. Natural logarithm of `gdp_real`, pre-computed for convenience.

Details

Spain experienced one of the sharpest COVID-19 contractions in the EU: approximately -18% quarter-on-quarter in 2020 Q2, followed by a strong V-shaped recovery in 2020 Q3. The series starts in 1995 Q1 under ESA 2010 methodology.

Source

Federal Reserve Bank of St. Louis — FRED Economic Data, series CLVMNACSCAB1GQES.
Downloaded via the public CSV endpoint `https://fred.stlouisfed.org/graph/fredgraph.csv?id=CLVMNACSCAB1GQE`
See `data-raw/intl_gdp.R` for the reproducible download script.

Examples

```
data("es_gdp", package = "MacroFilters")
head(es_gdp)
```

`fr_gdp` *France Real GDP — FRED Vintage*

Description

Quarterly France Real Gross Domestic Product from the Federal Reserve Bank of St. Louis (FRED) public data API (series **CLVMNACSCAB1GQFR**), expressed in millions of chained 2015 EUR, seasonally adjusted. Original source: Eurostat/OECD National Accounts via FRED.

Usage

```
fr_gdp
```

Format

A `data.table` with one row per quarter and three columns:

`date Date`. Quarter start date (e.g. `2000-01-01` = 2000 Q1).

`gdp_real numeric`. Real GDP level, millions of chained 2015 EUR.

`gdp_log numeric`. Natural logarithm of `gdp_real`, pre-computed for convenience.

Details

France experienced a sharp COVID-19 contraction of approximately -14% quarter-on-quarter in 2020 Q2, followed by a rapid V-shaped recovery. Together with Spain (`es_gdp`), the two series serve as a demanding stress test for trend filters in the introduction vignette.

Source

Federal Reserve Bank of St. Louis — FRED Economic Data, series CLVMNACSCAB1GQFR. Downloaded via the public CSV endpoint `https://fred.stlouisfed.org/graph/fredgraph.csv?id=CLVMNACSCAB1GQF` See `data-raw/intl_gdp.R` for the reproducible download script.

Examples

```
data("fr_gdp", package = "MacroFilters")
head(fr_gdp)
```

---

hamilton_filter *Hamilton Filter*

Description

Decomposes a time series into trend and cycle components using the regression-based filter proposed by Hamilton (2018). The trend is the fitted value from an OLS regression of y_{t+h} on $(1, y_t, y_{t-1}, \ldots, y_{t-p+1})$, and the cycle is the residual.

Usage

```
hamilton_filter(x, h = NULL, p = 4L, boot_iter = 0, block_size = "auto")
```

Arguments

`x`	Numeric vector, `ts`, `xts`, or `zoo` object.
`h`	Integer horizon (number of periods ahead). If `NULL` (default), auto-detected from the series frequency using Hamilton's rule: annual = 2, quarterly = 8, monthly = 24.
`p`	Integer number of lags in the regression (default 4).
`boot_iter`	Non-negative integer. Number of block-bootstrap iterations for uncertainty quantification (default `0`, bootstrap disabled). When `> 0`, the function adds `$trend_lower` and `$trend_upper`: a 95% normal-approximation band, `trend +/- 1.96 * sd(bootstrap trends)`, centred on the estimated trend. The bootstrap sd is used instead of empirical percentiles because it is smooth and stable at practical `boot_iter`. Each bootstrap refit uses the same `mstop` as the base fit, so larger `boot_iter` raises cost linearly. See also `block_size`.
`block_size`	Positive integer or `"auto"`. Block length for the moving-block bootstrap (used only when `boot_iter > 0`). If `"auto"` (default), it is set to `2 * stats::frequency(x)` (two full cycles), bounded above by `floor(length(x) / 3)` to keep at least three blocks.

Details

Hamilton (2018) proposes replacing the HP filter with a simple regression:

$$y_{t+h} = \beta_0 + \beta_1 y_t + \beta_2 y_{t-1} + \cdots + \beta_p y_{t-p+1} + v_{t+h}$$

The fitted values $\hat{y}_{t+h}$ define the trend and the residuals $\hat{v}_{t+h}$ define the cycle.

The first $h + p - 1$ observations have no computable trend or cycle and are filled with `NA`.

The lag matrix is constructed vectorized via `embed()` and the regression is solved with `stats::lm.fit()` for speed.

When `boot_iter > 0`, the confidence band comes from a residual bootstrap that holds the observed lead-in fixed (conditional on initial values) and resamples residuals only over the valid window. A direct consequence is that the band is narrow at the start of the valid window – where the predictors are entirely the (frozen) lead-in, so only the regression coefficients vary across replicates – and widens forward as the predictors themselves become resampled quantities. This is the correct behaviour of a conditional bootstrap, not an artefact.

Value

A list of class `c("macrofilter", "list")` with `trend`, `cycle`, `data`, and `meta` (`h`, `p`, `coefficients`, `compute_time`). When `boot_iter > 0` it also carries `trend_lower` and `trend_upper` (95% normal-approximation band). The bootstrap is a residual bootstrap conditional on the initial `h + p - 1` observations (the regression lead-in is held fixed); band entries for those lead-in positions are `NA`.

References

Hamilton, J.D. (2018). Why You Should Never Use the Hodrick-Prescott Filter. *Review of Economics and Statistics*, 100(5), 831–843.

Examples

```
# Quarterly GDP-like series
y <- ts(cumsum(rnorm(200)), start = c(2000, 1), frequency = 4)
result <- hamilton_filter(y)
print(result)
```

---

hp_filter *Hodrick-Prescott Filter (Sparse Matrix Implementation)*

Description

Decomposes a time series into trend and cycle components by solving the HP penalized least-squares problem using a sparse Cholesky factorization. This avoids the dense O(n^3) inversion used by other implementations and scales linearly in the number of observations.

Usage

```
hp_filter(x, lambda = NULL, freq = NULL, boot_iter = 0, block_size = "auto")
```

Arguments

x	Numeric vector, ts, xts, or zoo object.
lambda	Smoothing parameter. If NULL (default), it is auto-detected using the Ravn-Uhlig rule (6.25 * freq^4).
freq	Numeric frequency override (1 = annual, 4 = quarterly, 12 = monthly). Used only when lambda is NULL and the frequency cannot be inferred from x.
boot_iter	Non-negative integer. Number of block-bootstrap iterations for uncertainty quantification (default 0, bootstrap disabled). When > 0, the function adds $trend_lower and $trend_upper: a 95% normal-approximation band, trend +/- 1.96 * sd(bootstrap trends), centred on the estimated trend. The bootstrap sd is used instead of empirical percentiles because it is smooth and stable at practical boot_iter. Each bootstrap refit uses the same mstop as the base fit, so larger boot_iter raises cost linearly. See also block_size.
block_size	Positive integer or "auto". Block length for the moving-block bootstrap (used only when boot_iter > 0). If "auto" (default), it is set to 2 * stats::frequency(x) (two full cycles), bounded above by floor(length(x) / 3) to keep at least three blocks.

Details

The HP filter minimises

$$\sum (y_t - \tau_t)^2 + \lambda \sum (\Delta^2 \tau_t)^2$$

which admits the closed-form solution

$$(I + \lambda D' D)\, \tau = y$$

where D is the second-difference operator.

The implementation builds D as a banded sparse matrix (Matrix::bandSparse()) and solves the symmetric positive-definite system with a sparse Cholesky decomposition (Matrix::solve()).

When lambda is not supplied the Ravn-Uhlig (2002) rule is applied: lambda = 6.25 * freq^4, yielding 6.25 (annual), 1600 (quarterly), and 129 600 (monthly).

Value

A list of class c("macrofilter", "list") with trend, cycle, data, and meta. When boot_iter > 0 it also carries trend_lower and trend_upper (95% normal-approximation bootstrap band).

References

Hodrick, R.J. and Prescott, E.C. (1997). Postwar U.S. Business Cycles: An Empirical Investigation. *Journal of Money, Credit and Banking*, 29(1), 1–16.

Ravn, M.O. and Uhlig, H. (2002). On Adjusting the Hodrick-Prescott Filter for the Frequency of Observations. *Review of Economics and Statistics*, 84(2), 371–376.

Examples

```
# Quarterly GDP-like series
y <- ts(cumsum(rnorm(200)), start = c(2000, 1), frequency = 4)
result <- hp_filter(y)
print(result)
```

`mbh_filter` *MacroBoost Hybrid (MBH) Filter*

Description

Decomposes a time series into trend and cycle using a robust boosting algorithm. Unlike the HP filter, MBH uses the Huber loss function to automatically downweight outliers (like the COVID-19 shock), preventing them from distorting the trend.

Usage

```
mbh_filter(
  x,
  d = "auto",
  boot_iter = 0,
  block_size = "auto",
  knots = NULL,
  mstop = 500L,
  nu = 0.1,
  df = 4L,
  select_mstop = FALSE,
  boundary.knots = NULL,
  hp_lambda = NULL
)
```

Arguments

`x`	Numeric vector, `ts`, `xts`, or `zoo` object.
`d`	Numeric or `"auto"`. The delta parameter for Huber loss. If `"auto"` (default), it is calibrated as `stats::mad(.hp_fast(x))`, i.e. the MAD of the HP cyclical residual. This anchors the threshold to the output-gap scale rather than the growth-rate scale, avoiding the under-truncation failure mode of the legacy `mad(diff(y))` heuristic. A `message()` is emitted reporting the exact value chosen. Supply an explicit positive numeric to override.
`boot_iter`	Non-negative integer. Number of block-bootstrap iterations for uncertainty quantification (default `0`, bootstrap disabled). When `> 0`, the function adds `$trend_lower` and `$trend_upper`: a 95% normal-approximation band, `trend +/- 1.96 * sd(bootstrap trends)`, centred on the estimated trend. The bootstrap sd is used instead of empirical percentiles because it is smooth and stable at practical `boot_iter`. Each bootstrap refit uses the same `mstop` as the base fit, so larger `boot_iter` raises cost linearly. See also `block_size`.

`block_size` Positive integer or `"auto"`. Block length for the moving-block bootstrap (used only when `boot_iter > 0`). If `"auto"` (default), it is set to `2 * stats::frequency(x)` (two full cycles), bounded above by `floor(length(x) / 3)` to keep at least three blocks.

`knots` Integer. Number of interior knots for the P-Spline. If `NULL` (default), it is calculated as `min(max(20, floor(n / 2)), 250)`. High knot density keeps the trend flexible, while the cap of 250 keeps the B-spline basis bounded for long / high-frequency series: in a P-spline the smoothness is governed by the difference penalty (via `df`, `mstop`, `nu`), not by the knot count, so beyond a few hundred knots the extra basis columns only inflate memory and runtime without adding useful flexibility.

`mstop` Integer. Maximum number of boosting iterations (default 500). If `select_mstop = TRUE` this is the upper bound; the actual stopping point is chosen by AICc.

Under-smoothing warning (`mstop` vs `d`): when `d` is small relative to the trend's range – the typical case for long log-level series, where the cycle (hence the auto-calibrated `d`) is tiny but the trend spans a large range – the Huber loss caps the gradient from the first iteration, so each boosting step advances the trend only slightly. Reducing `mstop` then leaves the trend unable to climb its full range: it collapses to a nearly flat curve while the cycle absorbs the long-run variation. Keep the default `mstop = 500` (or higher) for such series; lower it only for short or high-cycle-variance inputs.

`nu` Numeric. The learning rate (shrinkage) for boosting (default 0.1).

`df` Integer. Effective degrees of freedom per boosting step for the P-Spline base learner (default 4). This enforces the *weak-learner* constraint of Bühlmann & Hothorn (2007): each boosting step contributes only a small, smooth update so that the trend is built up gradually over many iterations rather than fitted in one pass.

End-point instability warning: Higher `df` values cause the B-spline basis matrix to shift drastically when the sample size changes by even one observation (the "rubber-band effect"). The last few data points pull the estimated trend non-smoothly, producing unreliable end-of-sample estimates. Keep `df = 4` (the default) unless you have a specific reason to deviate.

`select_mstop` Logical. If `TRUE`, the optimal number of boosting iterations is selected automatically via AICc (corrected AIC), following Bühlmann & Hothorn (2007). The `mstop` argument acts as the search upper bound. Default `FALSE`.

AICc underfitting warning: In the combination of Huber quasi-likelihood + P-splines, AICc penalises model complexity hyper-aggressively. In practice the algorithm stops at iteration ~5–15 instead of the intended ~500. The resulting trend is nearly a straight line; all long-run variance is pushed into the cycle component, defeating the purpose of the filter. Treat `select_mstop = TRUE` as an experimental option and validate visually before relying on it.

`boundary.knots` A numeric vector of length 2 specifying the global domain for the B-spline basis (e.g., `c(1, T_max)`). If `NULL` (default), the range of `time_idx` is used. For real-time stability, fix this to the full-sample domain so the basis does not shift as the sample grows.

`hp_lambda` Numeric or `NULL`. Smoothing parameter for the internal HP filter used to auto-calibrate `d` (only relevant when `d = "auto"`). If `NULL` (default), it is derived from

`stats::frequency(x)` via the Ravn-Uhlig rule. **Supply this when `x` is a plain numeric vector whose true frequency is not annual**, since `frequency()` returns 1 for unclassed vectors and would otherwise under-smooth the calibration cycle (e.g. monthly data: `hp_lambda = 129600`).

Details

The model estimated is an additive model:

$$y_t = \text{Linear}(t) + \text{Smooth}(t) + \epsilon_t$$

It is fitted using `mboost::mboost()` with:

- **Base Learners:** A linear time trend (`mboost::bols()`) to capture the global path, plus a B-spline (`mboost::bbs()`) to capture local curvature.
- **Loss Function:** Huber loss (`mboost::Huber()`) with parameter `d`. This is the key to robustness.

The default parameters (`knots = min(n/2, 250)`, `mstop = 500`) are calibrated to mimic the flexibility of a standard HP filter while retaining the robustness of the Huber loss.

Value

A list of class `c("macrofilter", "list")` with:

`$trend` Numeric trend vector.

`$cycle` Numeric cycle vector.

`$data` Original input as numeric.

`$meta` Named list: `method`, `knots`, `d`, `mstop`, `nu`, `df`, `select_mstop`, `compute_time`.

`$trend_lower`, `$trend_upper` 95% normal-approximation bootstrap band (`trend +/- 1.96 * sd`). Present only when `boot_iter > 0`.

Calibration Guidance

Three failure modes were discovered through empirical stress-testing. The defaults guard against all three:

1. **Huber delta scale mismatch** (`d`) The automatic fallback `mad(diff(y))` operates on the scale of growth rates, not the output gap. For log-level input this sets `d` one to two orders of magnitude too small, causing ordinary business-cycle swings to be treated as outliers. If the estimated cycle looks implausibly large or the trend is nearly linear, override with `d = mad(hp_filter(x)$cycle)` as a starting point.
2. **AICc underfitting** (`select_mstop`) AICc + Huber quasi-likelihood + P-splines stops boosting at iteration ~5–15. The trend degenerates to a near-straight line and the cycle absorbs all long-run variance. Leave `select_mstop = FALSE` (the default) and set `mstop` explicitly instead.
3. **End-point instability** (`df`) Values above 4 shift the B-spline basis matrix non-smoothly as the sample grows, producing a "rubber-band" distortion in the final observations. Keep `df = 4` (the default) for real-time applications.

Examples

```
# Fast example with reduced series and iterations
set.seed(42)
y <- ts(cumsum(rnorm(80)), start = c(2000, 1), frequency = 4)
result <- mbh_filter(y, mstop = 100L)
print(result)


# Full example with default parameters
y2 <- ts(cumsum(rnorm(200)), start = c(2000, 1), frequency = 4)
result2 <- mbh_filter(y2)
print(result2)
```

`us_gdp_vintage` *US Real GDP — FRED Vintage*

Description

Quarterly US Real Gross Domestic Product from the Federal Reserve Bank of St. Louis (FRED) public data API (series **GDPC1**), expressed in billions of chained 2017 US dollars, seasonally adjusted annual rate.

Usage

```
us_gdp_vintage
```

Format

A `data.table` with one row per quarter and three columns:

`date Date`. Quarter start date (e.g. `1947-01-01` = 1947 Q1).

`gdp_real numeric`. Real GDP level, billions of chained 2017 USD.

`gdp_log numeric`. Natural logarithm of `gdp_real`, pre-computed for convenience.

Details

The dataset covers 1947 Q1 through the latest vintage available at download time (approximately 316 rows as of 2025). Rows with `NA` in `gdp_real` are excluded.

The log-level column (`gdp_log`) is particularly useful for trend-cycle decomposition because log-differences approximate quarter-on-quarter percentage growth rates:

$$\Delta \log(\text{GDP}_t) \approx g_t$$

Source

Federal Reserve Bank of St. Louis — FRED Economic Data, series GDPC1. Downloaded via the public CSV endpoint `https://fred.stlouisfed.org/graph/fredgraph.csv?id=GDPC1`. See `data-raw/us_gdp_vintage.R` for the reproducible download script.

Examples

```
data("us_gdp_vintage", package = "MacroFilters")
head(us_gdp_vintage)
plot(us_gdp_vintage$date, us_gdp_vintage$gdp_log,
     type = "l", xlab = "Date", ylab = "Log Real GDP",
     main = "US Real GDP (log level)")
```

Index