Package 'trendseries'

May 8, 2026

Type Package

Title Extract Trends from Time Series

Version 1.2.0

Description Extract trends from monthly and quarterly economic time series. Provides two main functions: augment_trends() for pipe-friendly 'tibble' workflows and extract_trends() for direct time series analysis. Includes established econometric filters such as Hodrick-Prescott (HP), Baxter-King, Christiano-Fitzgerald, and Hamilton, alongside moving averages and smoothing methods. Smart defaults are tuned for common economic frequencies following Ravn and Uhlig (2002) <doi:10.1162/003465302317411604>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

URL https://github.com/viniciusoike/trendseries, https://viniciusoike.github.io/trendseries/

BugReports https://github.com/viniciusoike/trendseries/issues

Imports cli, dlm, glue, hpfilter, lubridate, mFilter, RcppRoll, stats, tibble, tsbox

Depends R (>= 4.1.0)

RoxygenNote 7.3.3

Suggests dplyr, ggplot2, knitr, rmarkdown, scales, testthat (>= 3.0.0), tidyr, xts

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Vinicius Oike [aut, cre, cph]

Maintainer Vinicius Oike <viniciusoike@gmail.com>

Repository CRAN

Date/Publication 2026-05-02 22:50:02 UTC

Contents

`augment_trends` *Add trend columns to data frame*

Description

Pipe-friendly function that adds trend columns to a tibble or data.frame. Designed for exploratory analysis of monthly and quarterly economic time series. Supports multiple trend extraction methods and handles grouped data.

Usage

```
augment_trends(
  data,
  date_col = "date",
  value_col = "value",
  group_cols = NULL,
  group_vars = NULL,
  methods = "stl",
  frequency = NULL,
  suffix = NULL,
  window = NULL,
  smoothing = NULL,
  band = NULL,
  align = NULL,
  params = list(),
```

```
  .quiet = FALSE
)
```

Arguments

data	A data.frame, tibble, or data.table containing the time series data.
date_col	Name of the date column. Defaults to "date". Must be of class Date.
value_col	Name of the value column(s). Defaults to "value". Must be numeric. A character vector of length > 1 is accepted; trends are extracted for each column and named trend_{method}_{col} (e.g. trend_stl_consumption).
group_cols	Optional grouping variables for multiple time series. Can be a character vector of column names.
group_vars	Deprecated. Use group_cols instead.
methods	Character vector of trend methods. Options: "hp", "bk", "cf", "ma", "stl", "loess", "spline", "poly", "bn", "ucm", "hamilton", "spencer", "ewma", "wma", "triangular", "kernel", "kalman", "median", "gaussian". Default is "stl".
frequency	The frequency of the series. Supports 4 (quarterly) or 12 (monthly). Will be auto-detected if not specified.
suffix	Optional suffix for trend column names. If NULL, uses method names.
window	Unified window/period parameter for moving average methods (ma, wma, triangular, stl, ewma, median, gaussian). Must be positive. If NULL, uses frequency-appropriate defaults. For EWMA, specifies the window size when using TTR's optimized implementation. Cannot be used simultaneously with smoothing for EWMA method. For ma and median methods, a numeric vector is accepted (e.g., c(3, 6, 12)), which adds one column per window value named trend_ma_3, trend_ma_6, etc. Other methods ignore extra values (with a warning).
smoothing	Unified smoothing parameter for smoothing methods (hp, loess, spline, ewma, kernel, kalman). For hp: use large values (1600+) or small values (0-1) that get converted. For EWMA: specifies the alpha parameter (0-1) for traditional exponential smoothing. Cannot be used simultaneously with window for EWMA method. For kernel: multiplier of optimal bandwidth (1.0 = optimal, <1 = less smooth, >1 = more smooth). For kalman: controls the ratio of measurement to process noise (higher = more smoothing). For others: typically 0-1 range.
band	Unified band parameter for bandpass filters (bk, cf). Both values must be positive. Provide as c(low, high) where low/high are periods in quarters, e.g., c(6, 32).
align	Unified alignment parameter for moving average methods (ma, wma, triangular, gaussian). Valid values: "center" (default, uses surrounding values), "right" (causal, uses past values only), "left" (anti-causal, uses future values only). Note: triangular only supports "center" and "right". If NULL, uses "center" as default.
params	Optional list of method-specific parameters for fine control.
.quiet	If TRUE, suppress informational messages.

Details

This function is designed for monthly (frequency = 12) and quarterly (frequency = 4) economic data. It uses economic-appropriate defaults for all trend extraction methods.

For grouped data, the function applies trend extraction to each group separately, maintaining the original data structure while adding trend columns.

Value

A tibble with original data plus trend columns named `trend_{method}` or `trend_{method}_{suffix}` if suffix is provided.

Examples

```
# Simple STL decomposition on quarterly GDP construction data
gdp_construction |> augment_trends(value_col = "index")

# Multiple smoothing methods with unified parameter
gdp_construction |>
  augment_trends(
    value_col = "index",
    methods = c("hp", "loess", "ewma"),
    smoothing = 0.3
  )

# Moving averages with unified window on monthly data
vehicles |>
  tail(60) |>
  augment_trends(
    value_col = "production",
    methods = c("ma", "wma", "triangular"),
    window = 8
  )

# Economic indicators with different methods
ibcbr |>
  tail(48) |>
  augment_trends(
    value_col = "index",
    methods = c("median", "kalman", "kernel"),
    window = 9,
    smoothing = 0.15
  )

# Moving average with right alignment (causal filter)
vehicles |>
  tail(60) |>
  augment_trends(
    value_col = "production",
    methods = "ma",
    window = 12,
    align = "right"
```

```
  )

# Advanced: fine-tune specific methods
electric |>
  tail(72) |>
  augment_trends(
    value_col = "consumption",
    methods = "median",
    window = 7
  )

# Multiple MA windows in a single call (adds trend_ma_3, trend_ma_6, trend_ma_12)
vehicles |>
  tail(60) |>
  augment_trends(
    value_col = "production",
    methods = "ma",
    window = c(3, 6, 12)
  )
```

`coffee_arabica` *Daily Arabica Coffee Price*

Description

Daily Arabica coffee price data from CEPEA/ESALQ (USP) with inflation adjustment.

Prices are provided in Brazilian reais, with USD values calculated using the daily USD/BRL exchange rate at 16:30. All reported prices include taxes and freight costs.

The data tracks prices for standard 60kg sacks of type 6 Arabica coffee negotiated in São Paulo (SP), representing production from five major regions: Cerrado, Sul de Minas Gerais, Mogiana (SP), Garça (SP), and Northeast Paraná. Regional prices are weighted by production volume to create the final value.

Usage

```
coffee_arabica
```

Format

A tibble with daily observations:

date Date column

spot_rs Spot price in Brazilian Reais per 60-kg bag

spot_us Spot price in US Dollars per 60-kg bag

usd_2022 US Dollar price adjusted for inflation (base year 2022)

trend 22-day rolling mean of inflation-adjusted prices

Source

Center for Advanced Studies in Applied Economics (CEPEA - ESALQ/USP).

See Also

coffee_robusta

`coffee_robusta` *Daily Robusta Coffee Price*

Description

Daily Robusta coffee price data from CEPEA/ESALQ (USP) with inflation adjustment.

Prices are provided in Brazilian reais, with USD values calculated using the daily USD/BRL exchange rate at 16:30. All reported prices include taxes and freight costs.

The data tracks prices for standard 60kg sacks of type 6 Arabica coffee negotiated in São Paulo (SP), representing production from two major regions: Colatina (ES) and São Gabriel da Palha (ES). The final value is an arithmetic average of the regional prices.

Usage

```
coffee_robusta
```

Format

A tibble with daily observations:

date Date column

spot_rs Spot price in Brazilian Reais per 60-kg bag

spot_us Spot price in US Dollars per 60-kg bag

usd_2022 US Dollar price adjusted for inflation (base year 2022)

trend 22-day rolling mean of inflation-adjusted prices

Source

Center for Advanced Studies in Applied Economics (CEPEA - ESALQ/USP).

See Also

coffee_arabica

converters *Data Format Conversion Utilities*

Description

Functions for converting between different time series formats, frequency detection, and data frame manipulation for the trendseries package. These functions handle the interface between tibble/data.frame workflows and time series objects.

df_to_ts *Convert a data.frame into a time series (ts)*

Description

Converts a series, stored in a data.frame or tibble, into a ts object.

Usage

```
df_to_ts(x, date_col = "date", value_col = "value", frequency = 12)
```

Arguments

x	A `data.frame`, `tibble` or `data.table`.
date_col	Name of the date column. Defaults to `'date'`. Must be of class `Date`.
value_col	Name of the value column. Defaults to `'value'`. Must be `numeric`.
frequency	The frequency of the series. Can be a shortened string (e.g. "M" for monthly) or a number (e.g. 12).

Value

A `ts` object

Examples

```
ibc <- df_to_ts(ibcbr, value_col = "index", frequency = "M")
class(ibc)
plot(ibc)
```

electric *Electric Consumption Residential*

Description

Monthly residential electric consumption in Brazil (GWh).

Usage

```
electric
```

Format

A tibble with monthly observations:

date Date column

consumption Electric consumption in GWh

Source

Brazilian Central Bank SGS (code 1403)

extract_trends *Extract trends from time series objects*

Description

Extract trend components from time series objects using various econometric methods. Designed for monthly and quarterly economic data analysis. Returns trend components as time series objects or a list of time series.

Usage

```
extract_trends(
  ts_data,
  methods = "stl",
  window = NULL,
  smoothing = NULL,
  band = NULL,
  align = NULL,
  params = list(),
  .quiet = FALSE
)
```

Arguments

ts_data	A time series object (ts, xts, or zoo) or any object convertible via tsbox.
methods	Character vector of trend methods. Options: "hp", "bk", "cf", "ma", "stl", "loess", "spline", "poly", "bn", "ucm", "hamilton", "spencer", "ewma", "wma", "triangular", "kernel", "kalman", "median", "gaussian". Default is "stl".
window	Unified window/period parameter for moving average methods (ma, wma, triangular, stl, ewma, median, gaussian). Must be positive. If NULL, uses frequency-appropriate defaults. For EWMA, specifies the window size when using TTR's optimized implementation. Cannot be used simultaneously with smoothing for EWMA method. For ma and median methods, a numeric vector is accepted (e.g., c(3, 6, 12)), which runs the method once per window value and returns a named list with keys like ma_3, ma_6, ma_12. Other methods ignore extra values (with a warning).
smoothing	Unified smoothing parameter for smoothing methods (hp, loess, spline, ewma, kernel, kalman). For hp: use large values (1600+) or small values (0-1) that get converted. For EWMA: specifies the alpha parameter (0-1) for traditional exponential smoothing. Cannot be used simultaneously with window for EWMA method. For kernel: multiplier of optimal bandwidth (1.0 = optimal, <1 = less smooth, >1 = more smooth). For kalman: controls the ratio of measurement to process noise (higher = more smoothing). For others: typically 0-1 range.
band	Unified band parameter for bandpass filters (bk, cf). Both values must be positive. For bk/cf: Provide as c(low, high) where low/high are periods in quarters, e.g., c(6, 32).
align	Unified alignment parameter for moving average methods (ma, wma, triangular, gaussian). Valid values: "center" (default, uses surrounding values), "right" (causal, uses past values only), "left" (anti-causal, uses future values only). Note: triangular only supports "center" and "right". If NULL, uses "center" as default.
params	Optional list of method-specific parameters for fine control: • **HP Filter**: hp_onesided (logical, default FALSE) - Use one-sided (real-time) filter instead of two-sided • **STL**: stl_s_window or s.window (numeric/"periodic", default "periodic") - Seasonal window, stl_t_window or t.window (numeric/NULL, default NULL) - Trend window, stl_robust or robust (logical, default FALSE) - Use robust fitting. Note: Both dot notation (s.window) and underscore notation (stl_s_window) are accepted. • **Spline**: spline_cv (logical/NULL) - Cross-validation method: NULL (none), TRUE (leave-one-out), FALSE (GCV) • **Polynomial**: poly_degree (integer, default 1), poly_raw (logical, default FALSE for orthogonal polynomials) • **UCM**: ucm_type (character, default "level") - Model type: "level", "trend", or "BSM" • **Others**: bn_ar_order, hamilton_h, hamilton_p, kernel_type, kalman_measurement_noise, kalman_process_noise, median_endrule, gaussian_sigma, wma_weights.

- **Note**: Alignment parameters (`ma_align`, `wma_align`, `triangular_align`, `gaussian_align`) can still be passed via `params` but it's recommended to use the unified `align` parameter instead.

`.quiet` If `TRUE`, suppress informational messages.

Details

This function focuses on monthly (frequency = 12) and quarterly (frequency = 4) economic data. It uses established econometric methods with appropriate defaults:

- **HP Filter**: lambda=1600 (quarterly), lambda=14400 (monthly). Supports both two-sided and one-sided (real-time) variants
- **Baxter-King**: Bandpass filter for business cycles (6-32 quarters default)
- **Christiano-Fitzgerald**: Asymmetric bandpass filter
- **Moving Average**: Centered, frequency-appropriate windows
- **STL**: Seasonal-trend decomposition
- **Loess**: Local polynomial regression
- **Spline**: Smoothing splines
- **Polynomial**: Linear/polynomial trends
- **Beveridge-Nelson**: Permanent/transitory decomposition
- **UCM**: Unobserved Components Model (local level)
- **Hamilton**: Regression-based alternative to HP filter
- **Advanced MA**: EWMA with various implementations
- **Kernel Smoother**: Non-parametric regression with various kernel functions
- **Kalman Smoother**: Adaptive filtering for noisy time series
- **Median Filter**: Robust filtering using running medians to remove outliers
- **Gaussian Filter**: Weighted average with Gaussian (normal) density weights

Parameter Usage Notes:

- **HP Filter**: Use `hp_onesided=TRUE` for real-time analysis or when future data should not influence current estimates. One-sided filter is appropriate for nowcasting, policy analysis, and avoiding look-ahead bias. Default two-sided filter is optimal for historical analysis.
- **EWMA**: Use either `window` (TTR optimization) OR `smoothing` (alpha parameter), not both
- **Kalman**: Use `smoothing` parameter or `params` list for fine control of noise parameters
- **Spline**: Use `spline_cv` to control cross-validation (NULL=none, TRUE=LOO-CV, FALSE=GCV)
- **Polynomial**: Use `poly_raw=FALSE` for orthogonal polynomials (more stable for degree > 2) or `poly_raw=TRUE` for raw polynomials. Warning issued for degree > 3 (overfitting risk).
- **UCM**: Choose model type - "level" (simplest), "trend" (time-varying slope), or "BSM" (with seasonal component, requires seasonal data)

Value

If single method, returns a `ts` object. If multiple methods, returns a named list of `ts` objects.

Examples

```
# Single method
hp_trend <- extract_trends(AirPassengers, methods = "hp")

# Multiple methods with unified smoothing
smooth_trends <- extract_trends(
  AirPassengers,
  methods = c("hp", "loess", "ewma"),
  smoothing = 0.3
)

# EWMA with window (uses TTR optimization)
ewma_window <- extract_trends(AirPassengers, methods = "ewma", window = 12)

# EWMA with alpha (traditional formula)
ewma_alpha <- extract_trends(AirPassengers, methods = "ewma", smoothing = 0.2)

# Moving averages with unified window
ma_trends <- extract_trends(
  AirPassengers,
  methods = c("ma", "wma", "triangular"),
  window = 8
)

# Bandpass filters with unified band
bp_trends <- extract_trends(
  AirPassengers,
  methods = c("bk", "cf"),
  band = c(6, 32)
)

# Moving average with right alignment (causal filter)
ma_causal <- extract_trends(
  AirPassengers,
  methods = "ma",
  window = 12,
  align = "right"
)

# Signal processing methods with specific parameters
finance_trends <- extract_trends(
  AirPassengers,
  methods = c("kalman", "gaussian"),
  window = 9,  # For Gaussian filter
  params = list(kalman_measurement_noise = 0.1)  # Kalman-specific parameter
)

# Spline with cross-validation options
spline_trends <- extract_trends(
  AirPassengers,
  methods = "spline",
  params = list(spline_cv = FALSE)  # Use GCV instead of default
```

```
)

# Polynomial with orthogonal vs raw polynomials
poly_trends <- extract_trends(
  AirPassengers,
  methods = "poly",
  params = list(poly_degree = 2, poly_raw = FALSE)  # Orthogonal (default)
)

# UCM with different model types
ucm_trends <- extract_trends(
  AirPassengers,
  methods = "ucm",
  params = list(ucm_type = "BSM")  # Basic Structural Model with seasonality
)

# HP Filter: One-sided (real-time) vs Two-sided (historical)
hp_realtime <- extract_trends(
  AirPassengers,
  methods = "hp",
  params = list(hp_onesided = TRUE)  # For nowcasting and real-time analysis
)

# STL with custom parameters via params (both notations work)
stl_custom1 <- extract_trends(
  AirPassengers,
  methods = "stl",
  params = list(s.window = 21, robust = TRUE)  # Dot notation
)

stl_custom2 <- extract_trends(
  AirPassengers,
  methods = "stl",
  params = list(stl_s_window = 21, stl_robust = TRUE)  # Underscore notation
)

# Advanced: fine-tune specific methods
custom_trends <- extract_trends(
  AirPassengers,
  methods = c("median", "kalman"),
  window = 7,
  params = list(median_endrule = "constant")
)
```

`gdp_construction` *GDP Construction Index*

Description

Quarterly Brazilian GDP construction sector index (Base: average 1995 = 100).

Usage

```
gdp_construction
```

Format

A tibble with quarterly observations:

date Date column

index Construction index value

Source

DEPEC/GEATI/COACE. Fetched from Brazilian Central Bank SGS (code 22087)

---

ibcbr *Central Bank Economic Activity Index*

Description

Monthly Central Bank Economic Activity Index (IBC-Br). The IBC-Br was built based on proxies for the evolution of agriculture, industry and service-sector products. The proxies are aggregated with weights derived from the tables of supply and use of the Brazilian National Accounts.

Usage

```
ibcbr
```

Format

A tibble with monthly observations:

date Date column

index Index (2003 = 100)

Source

BACEN. Fetched from Brazilian Central Bank SGS (code 24363).

list_datasets *List Available Datasets*

Description

Returns a tibble with metadata for all datasets included in the trendseries package.

Usage

```
list_datasets()
```

Value

A tibble with the following columns:

name Dataset name

description Brief description of the dataset

frequency Data frequency (D = daily, M = monthly, Q = quarterly)

n_obs Number of observations

first_date First observation date

last_date Last observation date

value_cols Main value column(s) in the dataset

source Data source

Examples

```
# List all available datasets
list_datasets()

# Filter for monthly data
list_datasets() |>
  dplyr::filter(frequency == "M")
```

oil_derivatives *Oil Derivatives Production*

Description

Monthly production of petroleum derivatives in Brazil (thousand barrels/day).

Usage

```
oil_derivatives
```

Format

A tibble with monthly observations:

date Date column

production Oil derivatives production

Source

ANP. Fetched from Brazilian Central Bank SGS (code 1391).

---

retail_autofuel *UK Retail Sales - Automotive Fuel*

Description

Chained volume of retail sales for automotive fuel in the UK, non-seasonally adjusted. Index numbers of sales per week (100 = 2023). The monthly period consists of 4 weeks except for March, June, September and December which are 5 weeks. January 2025 is also a 5 week period. The series considers the "All Businesses" specification and covers Great Britain from 1998 to 2025.

Usage

```
retail_autofuel
```

Format

A tibble with monthly observations:

date Date column

value Retail sales index (chained volume)

name Series name

frequency Frequency ("M")

source Data source ("ONS")

Source

UK Office for National Statistics (ONS). (Table 3M).

See Also

retail_volume

retail_volume *UK Retail Index*

Description

Chained volume of retail sales, non-seasonally adjusted, selected sub-series. Index numbers of sales per week (100 = 2023). The monthly period consists of 4 weeks except for March, June, September and December which are 5 weeks. January 2025 is also a 5 week period. The included series consider the "All Businesses" specification and cover Great Britain from 1998 to 2025.

Usage

```
retail_volume
```

Format

A tibble with monthly observations:

date Date column

name_series Series name derived from the unofficial SIC

value Retail sales index (chained volume)

Source

UK Office for National Statistics (ONS). (Table 3M).

See Also

retail_autofuel

series_metadata *Series Metadata*

Description

Metadata for all economic series included in the package.

Usage

```
series_metadata
```

Format

A tibble with metadata:

series_name Short series identifier

description Full series description

value_column Main value column(s) in the dataset

frequency Data frequency (D = daily, M = monthly, Q = quarterly)

first_obs First observation date

last_obs Last observation date

n_obs Number of observations

source Data source

Source

Various (BCB-SGS, ONS, CEPEA/ESALQ)

---

transit_london_avgs *London Transit - Average Daily Journeys*

Description

Average daily journeys on London's bus and tube networks, split by business day and non-business day. Aggregated from daily Transport for London (TfL) network demand data using the UK calendar.

Usage

```
transit_london_avgs
```

Format

A tibble with monthly observations:

date_month First day of each month (Date)

transit_mode Transit mode: `"bus"` or `"tube"`

is_business_day 1 for business days, 0 for weekends/holidays

avg_daily_journeys Average daily journeys for the given day type

Source

Transport for London (TfL) network demand data.

See Also

transit_london_monthly

transit_london_monthly

London Transit - Monthly Journey Totals

Description

Monthly total journeys on London's bus and tube (underground) networks. Aggregated from daily Transport for London (TfL) network demand data.

Usage

```
transit_london_monthly
```

Format

A tibble with monthly observations:

date_month First day of each month (Date)

transit_mode Transit mode: `"bus"` or `"tube"`

journey_monthly Total journeys in the month

Source

Transport for London (TfL) network demand data.

See Also

transit_london_avgs

ts_to_df

Convert time series to tibble

Description

Convert time series to tibble

Usage

```
ts_to_df(x, date_col = NULL, value_col = NULL)
```

Arguments

`x`	A time series as a `ts` object
`date_col`	Optional name for the date column
`value_col`	Optional name for the value column

Value

a tibble

Examples

```
# example code
ts_to_df(AirPassengers)

# Using a custom name for the value column
ts_to_df(AirPassengers, value_col = "passengers")
```

---

vehicles *Vehicle Production*

Description

Monthly vehicle production in Brazil (units).

Usage

```
vehicles
```

Format

A tibble with monthly observations:

date Date column

production Vehicle production in absolute units

Source

Anfavea. Fetched from Brazilian Central Bank SGS (code 1378).

Index