

Package: publishTC (via r-universe)

October 13, 2026

Type Package

Title Tools to Help to Publish the Trend-Cycle Component

Version 0.2.3

Description Tools to facilitate the computation of the trend-cycle component of time series. In particular, the computation can be performed using: the Cascade Linear Filter ('CLF', Dagum and Luati, 2008), classical Henderson symmetric filters with surrogate Musgrave asymmetric filters, local parameterizations of Musgrave filters (Quartier-la-Tente, 2024), and robust extensions taking into account additive outliers and level shifts (Quartier-la-Tente, 2025). Computes confidence intervals and provides specialized plotting functions for publication.

LazyData true

Depends R (>= 4.1.0)

Imports rJava, rjd3filters (>= 2.4.0), rjd3x13 (>= 3.3.0), ggplot2, zoo, rjd3jars (>= 0.0.5)

URL <https://github.com/AQLT/publishTC>,
<https://aqlt.github.io/publishTC/>

BugReports <https://github.com/AQLT/publishTC/issues>

SystemRequirements Java (>= 17)

License EUPL

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs make default-jdk libprotobuf-dev protobuf-compiler libprotoc-dev

Repository <https://aqlt.r-universe.dev>

Date/Publication 2026-10-13 19:55:56 UTC

RemoteUrl <https://github.com/AQLT/publishTC>

RemoteRef HEAD

RemoteSha e84bfda5574fdcfb27ddecdbde4062a37a9afb4da

Contents

bandwidth	2
classical-ma	3
clf_smoothing	4
confint_plot	5
confint_tc	6
ggsmoothing_plot	7
growthplot	8
henderson_robust_smoothing	10
henderson_smoothing	12
icr	13
implicit_forecasts.henderson	14
implicit_forecasts_plot	15
lollypop	17
mcd	18
plot.tc_estimates	19
segmented_smoothing	21
smoothing	23
smoothness	24
tc_estimates	25
ts-exemple	27
turning_points	28
underlying_forecasts.henderson	29
underlying_forecasts_plot	30
write.ts	32
x11_trend_selection	33
x13_regarima_outliers	34
Index	35

bandwidth	<i>Get Filter Bandwidth</i>
-----------	-----------------------------

Description

Retrieves the bandwidth of the filter used in a "tc_estimates" object.

Usage

bandwidth(x)

Arguments

x An object of class "tc_estimates".

Details

The bandwidth corresponds to the number of observations on each side of the central value for a symmetric trend-cycle filter. The total filter length is equal to $2 \times \text{bandwidth}(x) + 1$.

Value

An integer representing the bandwidth of the trend-cycle filter.

Examples

```
x <- french_ipi[, "manufacturing"]
tc_clf <- henderson_smoothing(x)
bandwidth(tc_clf)
```

classical-ma	<i>Classical Moving Averages</i>
--------------	----------------------------------

Description

Datasets containing coefficients of classical moving averages used for trend-cycle extraction.

Usage

```
henderson
CLF
CLF_CN
local_param_est
```

Format

henderson is a list of objects of class "moving_average".

CLF is an object of class "finite_filters".

CLF_CN is an object of class "finite_filters".

local_param_est is a nested list of objects of class "finite_filters". The first level corresponds to the length of the filter; the second level corresponds to the degree of the local polynomial model used for the trend-cycle; and the third level corresponds to the target component (1 for bias, 2 for slope, and 3 for curvature).

Details

- henderson: Henderson moving averages of lengths 5, 7, 9, 13, and 23.
- CLF: Cascade Linear Filter (CLF) of length 13 and associated Asymmetric Linear Filters (ALF).
- CLF_CN: Cascade Linear Filter (CLF) of length 13 and associated cut-and-normalize asymmetric filters.
- local_param_est: Moving averages used to locally estimate the bias, slope, or curvature associated with trend-cycle estimation (via local polynomial approximation).

References

Dagum, E. B., & Luati, A. (2008). A Cascade Linear Filter to Reduce Revisions and False Turning Points for Real Time Trend-Cycle Estimation. *Econometric Reviews*, 28(1-3), 40–59. doi:10.1080/07474930802387837

Henderson, R. (1916). Note on graduation by adjusted average. *Transactions of the Actuarial Society of America*, 17, 43–48.

Musgrave, J. (1964). A Set of End Weights to End All End Weights. *US Census Bureau [Custodian]*. <https://www.census.gov/library/working-papers/1964/adrm/musgrave-01.html>

Quartier-la-Tente, A. (2024). Improving Real-Time Trend Estimates Using Local Parametrization of Polynomial Regression Filters. *Journal of Official Statistics*, 40(4), 685–715. doi:10.1177/0282423X241283207

clf_smoothing	<i>Trend-Cycle Estimation using the Cascade Linear Filter</i>
---------------	---

Description

Estimates the trend-cycle component using the Cascade Linear Filter (CLF) of length 13.

Usage

```
clf_smoothing(x, endpoints = c("cut-and-normalize", "ALF"), ...)
```

Arguments

x	An input time series (object of class "ts" or "mts").
endpoints	Character string specifying the method used for asymmetric filters. If "cut-and-normalize" (the default), the cut-and-normalize method is used (cut the symmetric filter and normalize the coefficients, as done by Statistics Canada); otherwise, Asymmetric Linear Filters (ALF) are used.
...	Other unused parameters.

Value

An object of class c("tc_estimates", "clf"). See [tc_estimates\(\)](#) for a full description of the returned object.

References

Dagum, E. B., & Luati, A. (2008). A Cascade Linear Filter to Reduce Revisions and False Turning Points for Real Time Trend-Cycle Estimation. *Econometric Reviews* 28(1-3), 40–59. doi:[10.1080/07474930802387837](https://doi.org/10.1080/07474930802387837)

Examples

```
tc <- clf_smoothing(cars_registrations)
plot(tc, xlim = c(2022, 2025))
```

confint_plot

Plot Confidence Intervals for Trend-Cycle Estimates

Description

Base R (confint_plot()) and **ggplot2** (ggconfint_plot()) graphics for plotting trend-cycle estimates alongside their confidence intervals.

Usage

```
confint_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_confint = "grey",
  xlab = "",
  ylab = "",
  level = 0.95,
  ...
)

ggconfint_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_confint = "grey",
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
  legend_confint = "Confidence interval",
  level = 0.95,
  ...
)
```

Arguments

object	An object of class "tc_estimates".
xlim, ylim	Limits for the x- and y-axes. If xlim is specified and ylim is NULL, ylim is determined automatically based on the truncated series.
col_sa, col_tc	Colors used for the seasonally adjusted and trend-cycle components, respectively.
col_confint	Color of the confidence interval lines or shaded region.
xlab, ylab	Character strings for the x- and y-axis labels.
level	The confidence level required (defaults to 0.95).
...	Additional arguments (currently unused).
legend_tc, legend_sa, legend_confint	Character strings specifying the legend labels for the trend-cycle, the seasonally adjusted series, and the confidence intervals, respectively.

Details

confint_plot() produces a base R graphic, whereas ggconfint_plot() creates a **ggplot2** object that can be further customized. The confidence intervals are computed internally using [confint\(\)](#).

Value

- confint_plot() returns NULL invisibly and is called for its side effect (drawing a plot).
- ggconfint_plot() returns a [ggplot](#) object.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
confint_plot(tc_mod, xlim = c(2022, 2024.5))
```

confint_tc

Confidence Intervals for Trend-Cycle Estimates

Description

Computes confidence intervals for trend-cycle estimates of class "tc_estimates".

Usage

```
## S3 method for class 'henderson'
confint(object, parm, level = 0.95, asymmetric_var = TRUE, ...)

## S3 method for class 'clf'
confint(object, parm, level = 0.95, asymmetric_var = TRUE, ...)

## S3 method for class 'robust_henderson'
confint(object, parm, level = 0.95, asymmetric_var = TRUE, ...)
```

Arguments

object	An object of class "tc_estimates".
parm	Unused parameter, kept for compatibility with the generic function.
level	The confidence level required (defaults to 0.95).
asymmetric_var	Logical. If TRUE (the default), the variance is estimated for each asymmetric filter. If FALSE, the variance associated with the symmetric filter is used throughout.
...	Additional arguments (currently unused).

Details

See `rjd3filters::confint_filter()` for details on the computation of confidence intervals.

Value

A matrix or mts object with the filtered series and the lower and upper bounds of the confidence interval.

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
confint <- confint(tc)

plot(confint, plot.type = "single",
     col = c("red", "black", "black"),
     lty = c(1, 2, 2), xlab = NULL, ylab = NULL)
lines(x, col = "grey")
legend("topleft", legend = c("x", "Smoothed", "CI (95%)"),
     col = c("grey", "red", "black"), lty = c(1, 1, 2))
```

ggsmoothing_plot

Plot Multiple Trend-Cycle Visualizations

Description

Generates a list of **ggplot2** graphics displaying various diagnostic and comparative plots for a "tc_estimates" object.

Usage

```
ggsmoothing_plot(
  object,
  plots = c("normal", "confint", "lollypop", "implicit_forecasts",
    "underlying_forecasts"),
  level = 0.95,
  ...
)
```

Arguments

<code>object</code>	An object of class "tc_estimates".
<code>plots</code>	A character vector specifying the types of plots to generate. Available options are "normal", "confint", "lollypop", "implicit_forecasts", and "underlying_forecasts". Defaults to generating all available plots.
<code>level</code>	The confidence level required (defaults to 0.95).
<code>...</code>	Additional graphical arguments passed to internal plotting functions.

Details

The following plot types can be produced:

- "normal": Standard plot showing trend-cycle and seasonally adjusted series via `autoplot.tc_estimates()`.
- "confint": Confidence interval plot via `ggconfint_plot()`.
- "lollypop": Lollipop chart comparing estimates via `gglollypop()`.
- "implicit_forecasts": Implicit forecasts plot via `ggimplicit_forecasts_plot()`.
- "underlying_forecasts": Underlying forecasts plot via `ggunderlying_forecasts_plot()`.

Value

A list of `ggplot` objects corresponding to the requested plots.

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
ggsmoothing_plot(tc)
```

growthplot

Plot Growth Rates for Trend-Cycle Estimates

Description

Base R (`growthplot()`) and **ggplot2** (`gggrowthplot()`) graphics for plotting the growth rates of the trend-cycle estimates (solid lines) and the seasonally adjusted series (bar-lines or lines).

Usage

```
growthplot(
  object,
  pct = TRUE,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
```

```

    xlab = "",
    ylab = "",
    sa_bar_line = TRUE,
    ...,
    lag = -1
)

gggrowthplot(
  object,
  pct = TRUE,
  xlim = NULL,
  ylim = NULL,
  sa_bar_line = TRUE,
  col_tc = "#E69F00",
  col_sa = "black",
  col_sa_fill = "grey",
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
  ...,
  lag = -1
)
```

Arguments

object	An object of class "tc_estimates".
pct	Logical. If TRUE (the default), the growth rate is expressed as a percentage, otherwise as a ratio.
xlim, ylim	Limits for the x- and y-axes. If xlim is specified and ylim is NULL, ylim is determined automatically based on the truncated series.
col_sa, col_tc	Colors used for the seasonally adjusted and trend-cycle components, respectively.
xlab, ylab	Character strings for the x- and y-axis labels.
sa_bar_line	Logical. If TRUE (the default), the growth rates of the seasonally adjusted series are presented as bar-lines; otherwise, as lines.
...	Additional graphical parameters passed to internal plotting functions.
lag	Integer specifying the lag used to compute the growth rate. Defaults to -1 (i.e., period-to-period growth rate).
col_sa_fill	Fill color of the bars for the seasonally adjusted series.
legend_tc, legend_sa	Character strings specifying the legend labels for the trend-cycle and seasonally adjusted components, respectively.

Details

growthplot() produces a base R graphic, whereas gggrowthplot() creates a **ggplot2** object that can be further customized.

Value

- `growthplot()` returns NULL invisibly and is called for its side effect (drawing a plot).
- `gggrowthplot()` returns a [ggplot](#) object.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
growthplot(tc_mod, xlim = c(2022, 2024.5))
```

`henderson_robust_smoothing`

Robust Trend-Cycle Estimation using the Henderson Filter

Description

Estimates the trend-cycle component using Henderson moving averages robust to outliers (additive outliers and level shifts).

Usage

```
henderson_robust_smoothing(
  x,
  endpoints = c("Musgrave", "QL", "CQ", "DAF"),
  length = NULL,
  ao = NULL,
  ao_tc = NULL,
  ls = NULL,
  icr = NULL,
  local_icr = FALSE,
  asymmetric_var = FALSE,
  degree = 3,
  ...
)
```

Arguments

<code>x</code>	An input time series (object of class "ts" or "mts").
<code>endpoints</code>	Character string specifying the method used for asymmetric filters. By default, "musgrave" is used.
<code>length</code>	An integer specifying the length of the symmetric filter. If NULL, the length is selected automatically using x11_trend_selection() .
<code>ao</code>	Vector of dates for Additive Outliers (AO) whose effects are allocated to the irregular component.
<code>ao_tc</code>	Vector of dates for Additive Outliers (AO) whose effects are allocated to the trend-cycle component.

<code>ls</code>	Vector of dates for Level Shifts (LS) whose effects are allocated to the trend-cycle component.
<code>icr</code>	Numeric value specifying the Irregular-to-Trend (I/C) ratio used for asymmetric filters.
<code>local_icr</code>	Logical. If TRUE, the I/C ratio is estimated locally (as described in Quartier-la-Tente, 2024) instead of globally.
<code>asymmetric_var</code>	Logical. When <code>local_icr = TRUE</code> , if TRUE, the variance is estimated for each asymmetric filter. If FALSE (the default), the variance associated with symmetric estimates is used throughout.
<code>degree</code>	Integer. When <code>local_icr = TRUE</code> , degree of the polynomial used to estimate the local bias parameter.
<code>...</code>	Additional arguments passed to <code>rjd3filters::lp_filter()</code> .

Details

When outliers are present in a time series, standard linear filters like the Henderson moving average can spread their impact across adjacent trend estimates. This function explicitly accounts for specified Additive Outliers (AO) and Level Shifts (LS) during the filtering process to prevent distortion of the trend-cycle estimates as described in Quartier-la-Tente (2025).

In a nutshell, Henderson smoothing is equivalent to a local polynomial regression of degree 3 (or equivalently 2) estimated with weighted least squares, using specific weights to obtain the Henderson coefficients. Adding regressors to this local regression to account for outliers allows removing their influence on the estimation of polynomial coefficients and thus on the trend-cycle estimates.

Value

An object of class `c("tc_estimates", "robust_henderson")`. See `tc_estimates()` for a full description of the returned object.

References

Quartier-la-Tente, A. (2025). Estimation de la tendance-cycle avec des méthodes robustes aux points atypiques. <https://github.com/AQLT/robustMA>.

Examples

```
x <- cars_registrations
outliers <- x13_regarima_outliers(x)
tc <- henderson_robust_smoothing(x, ao = outliers$ao, ls = outliers$ls, local_icr = TRUE)

tc_henderson <- henderson_smoothing(x)
plot(tc, xlim = c(2019, 2022))
lines(tc_henderson$tc, col = "blue")
```

henderson_smoothing *Trend-Cycle Estimation using the Henderson Filter*

Description

Estimates the trend-cycle component using Henderson moving averages.

Usage

```
henderson_smoothing(
  x,
  endpoints = c("Musgrave", "QL", "CQ", "CC", "DAF", "CN"),
  length = NULL,
  icr = NULL,
  local_icr = FALSE,
  asymmetric_var = FALSE,
  degree = 3,
  ...
)
```

Arguments

<code>x</code>	An input time series (object of class "ts" or "mts").
<code>endpoints</code>	Character string specifying the method used for asymmetric filters. By default, "musgrave" is used.
<code>length</code>	An integer specifying the length of the symmetric filter. If NULL, the length is selected automatically using x11_trend_selection() .
<code>icr</code>	Numeric value specifying the Irregular-to-Trend (I/C) ratio used for asymmetric filters.
<code>local_icr</code>	Logical. If TRUE, the I/C ratio is estimated locally (as described in Quartier-la-Tente, 2024) instead of globally.
<code>asymmetric_var</code>	Logical. When <code>local_icr = TRUE</code> , if TRUE, the variance is estimated for each asymmetric filter. If FALSE (the default), the variance associated with symmetric estimates is used throughout.
<code>degree</code>	Integer. When <code>local_icr = TRUE</code> , degree of the polynomial used to estimate the local bias parameter.
<code>...</code>	Additional arguments passed to rjd3filters::lp_filter() .

Value

An object of class `c("tc_estimates", "henderson")`. See [tc_estimates\(\)](#) for a full description of the returned object.

References

- Henderson, R. (1916). Note on Graduation by Adjusted Average. *Transactions of the Actuarial Society of America* 17: 43-48.
- Musgrave, J. (1964). A Set of End Weights to End All End Weights. *US Census Bureau [Custodian]*. <https://www.census.gov/library/working-papers/1964/adrm/musgrave-01.html>.
- Quartier-la-Tente, A. (2024). Improving Real-Time Trend Estimates Using Local Parametrization of Polynomial Regression Filters. *Journal of Official Statistics*, 40(4), 685-715. doi:10.1177/0282423X241283207.

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
plot(tc, xlim = c(2022, 2025))
```

icr

Compute Irregular-to-Trend-Cycle (I/C) Ratios

Description

`icr()` computes the overall I/C ratio, while `icrs()` computes the I/C ratios for each lag up to the series frequency.

Usage

```
icr(x, tc, mul = FALSE)

icrs(x, tc, mul = FALSE)
```

Arguments

- | | |
|------------------|---|
| <code>x</code> | A seasonally adjusted time series or an object of class "tc_estimates". |
| <code>tc</code> | A time series representing the trend-cycle component. Ignored if <code>x</code> is an object of class "tc_estimates". |
| <code>mul</code> | Logical indicating whether the decomposition is multiplicative (TRUE) or additive (FALSE, the default). |

Details

The I/C ratio measures the relative importance of the irregular component compared to the trend-cycle component in a time series decomposition.

For a time series of frequency p , when the decomposition is additive (`mul = FALSE`), the irregular component is defined as $I_t = X_t - TC_t$. `icrs()` returns a vector containing the ratio of the mean absolute variation of the irregular component to that of the trend-cycle component for each lag k :

$$\frac{\bar{I}_k}{\bar{C}_k} = \frac{\sum |I_t - I_{t-k}|}{\sum |TC_t - TC_{t-k}|} \quad \text{for } k \in \{1, 2, \dots, p\}.$$

If the decomposition is multiplicative (`mul = TRUE`), the irregular component is defined as $I_t = X_t/TC_t$, and the ratio of relative variations is computed as:

$$\frac{\bar{I}_k}{\bar{C}_k} = \frac{\sum |(I_t - I_{t-k})/I_{t-k}|}{\sum |(TC_t - TC_{t-k})/TC_{t-k}|} \quad \text{for } k \in \{1, 2, \dots, p\}.$$

`icr()` returns the overall I/C ratio, which corresponds to the first lag ratio $\bar{I}_1/\bar{C}_1$.

Value

`icr()` returns a single numeric value representing the overall I/C ratio, while `icrs()` returns a named numeric vector of I/C ratios for lags 1 to p .

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
icr(tc)
icrs(tc)
```

implicit_forecasts.henderson

Implicit Forecasts Methods

Description

S3 method of `rjd3filters::implicit_forecasts()` for "tc_estimates" objects.

Usage

```
## S3 method for class 'henderson'
implicit_forecasts(x, ...)

## S3 method for class 'clf'
implicit_forecasts(x, ...)

## S3 method for class 'robust_henderson'
implicit_forecasts(x, ...)
```

Arguments

`x` An object of class "tc_estimates".

`...` Additional arguments passed to internal methods (currently unused).

Details

See `rjd3filters::implicit_forecasts()` for details on the computation of implicit forecasts.

Value

A matrix or mts object containing the implicit forecasts for the series.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
implicit_forecasts(tc_mod)
```

`implicit_forecasts_plot`*Plot Implicit Forecasts for Trend-Cycle Estimates*

Description

Base R (`implicit_forecasts_plot()`) and **ggplot2** (`ggimplicit_forecasts_plot()`) graphics for plotting trend-cycle estimates alongside their implicit forecasts.

Usage

```
implicit_forecasts_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_i_f = col_sa,
  xlab = "",
  ylab = "",
  lty_last_tc = 2,
  lty_i_f = 3,
  n_last_tc = 4,
  ...
)

ggimplicit_forecasts_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_i_f = col_sa,
  lty_last_tc = 2,
  lty_i_f = 3,
  n_last_tc = 4,
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
```

```

    legend_i_f = "Implicit forecasts",
    ...
  )

```

Arguments

<code>object</code>	An object of class "tc_estimates".
<code>xlim, ylim</code>	Limits for the x- and y-axes. If <code>xlim</code> is specified and <code>ylim</code> is NULL, <code>ylim</code> is determined automatically based on the truncated series.
<code>col_sa, col_tc</code>	Colors used for the seasonally adjusted and trend-cycle components, respectively.
<code>col_i_f</code>	Color of the forecasts lines.
<code>xlab, ylab</code>	Character strings for the x- and y-axis labels.
<code>lty_last_tc, lty_i_f</code>	Line types used for the last values of the trend-cycle component and for the forecasts, respectively.
<code>n_last_tc</code>	Number of final values of the trend-cycle component to plot with a distinct line type (<code>lty_last_tc</code>), emphasizing higher uncertainty in recent estimates. If NULL (the default), <code>n_last_tc</code> is set to the Month of Cyclical Dominance (MCD) statistic.
<code>...</code>	Additional graphical parameters passed to internal plotting functions.
<code>legend_tc, legend_sa, legend_i_f</code>	Character strings specifying the legend labels for the trend-cycle, the seasonally adjusted series, and the forecasts, respectively.

Details

`implicit_forecasts_plot()` produces a base R graphic, whereas `ggimplicit_forecasts_plot()` creates a **ggplot2** object that can be further customized.

Value

- `implicit_forecasts_plot()` returns NULL invisibly and is called for its side effect (drawing a plot).
- `ggimplicit_forecasts_plot()` returns a **ggplot** object.

Examples

```

tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
implicit_forecasts_plot(tc_mod, xlim = c(2022, 2025))

```

lollypop

*Lollipop Plot for Trend-Cycle Estimates***Description**

Base R (`lollypop()`) and **ggplot2** (`gglollypop()`) graphics for creating lollipop plots of trend-cycle estimates alongside seasonally adjusted series.

Usage

```
lollypop(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  color_points = col_sa,
  cex_points = 1,
  pch_points = 16,
  xlab = "",
  ylab = "",
  lty_last_tc = 2,
  n_last_tc = 4,
  ...
)

gglollypop(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  color_points = col_sa,
  cex_points = 1,
  pch_points = 16,
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
  lty_last_tc = 2,
  n_last_tc = 4,
  ...
)
```

Arguments

<code>object</code>	An object of class "tc_estimates".
<code>xlim, ylim</code>	Limits for the x- and y-axes. If <code>xlim</code> is specified and <code>ylim</code> is <code>NULL</code> , <code>ylim</code> is determined automatically based on the truncated series.

<code>col_sa, col_tc</code>	Colors used for the seasonally adjusted and trend-cycle components, respectively.
<code>color_points, cex_points</code>	Color and size (character/numeric) of the points associated with the seasonally adjusted component.
<code>pch_points</code>	Point symbol type (integer or character) for the seasonally adjusted component.
<code>xlab, ylab</code>	Character strings for the x- and y-axis labels.
<code>lty_last_tc</code>	Line type for the last values of the trend-cycle component.
<code>n_last_tc</code>	Number of final values of the trend-cycle component to plot with a distinct line type (<code>lty_last_tc</code>), emphasizing higher uncertainty in recent estimates. If NULL (the default), <code>n_last_tc</code> is set to the Month of Cyclical Dominance (MCD) statistic.
<code>...</code>	Additional graphical parameters passed to internal plotting functions.
<code>legend_tc, legend_sa</code>	Character strings specifying the legend labels for the trend-cycle and seasonally adjusted components, respectively.

Details

`lollypop()` produces a base R graphic, whereas `ggollypop()` creates a **ggplot2** object that can be further customized.

Value

- `lollypop()` returns NULL invisibly and is called for its side effect (drawing a plot).
- `ggollypop()` returns a **ggplot** object.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
lollypop(tc_mod, xlim = c(2022, 2024.5))
```

mcd

Month of Cyclical Dominance (MCD)

Description

Computes the Month of Cyclical Dominance (MCD) for a time series decomposition.

Usage

```
mcd(x, tc, mul = FALSE)
```

Arguments

x	A seasonally adjusted time series or an object of class "tc_estimates".
tc	A time series representing the trend-cycle component. Ignored if x is an object of class "tc_estimates".
mul	Logical indicating whether the decomposition is multiplicative (TRUE) or additive (FALSE, the default).

Details

The Month of Cyclical Dominance (MCD) represents the minimum number of months required for the average absolute variation of the trend-cycle component to dominate that of the irregular component. In other words, it indicates the shortest time span needed on average for the cyclical signal to outweigh irregular fluctuations.

It is computed as the smallest integer k (where $k \geq 1$) such that the I/C ratio remains less than or equal to 1 for all lags $j \geq k$:

$$k = \min \left\{ k \in \{1, \dots, p\} : \frac{\bar{I}_j}{\bar{C}_j} \leq 1 \text{ for all } j \geq k \right\}$$

If the I/C ratio never falls below or equal to 1 within the frequency p , the MCD is set to p .

Value

An integer specifying the Month of Cyclical Dominance (MCD).

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
mcd(tc)
icrs(tc)
```

plot.tc_estimates	<i>Plotting Methods for Trend-Cycle Estimates</i>
-------------------	---

Description

Plotting methods for objects of class "tc_estimates". plot() and lines() produce base R graphics, whereas autoplot() creates a **ggplot2** object.

Usage

```

## S3 method for class 'tc_estimates'
plot(
  x,
  y = NULL,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  xlab = "",
  ylab = "",
  lty_last_tc = 2,
  n_last_tc = 4,
  ...
)

## S3 method for class 'tc_estimates'
lines(x, col_tc = "#E69F00", lty_last_tc = 2, n_last_tc = 4, ...)

## S3 method for class 'tc_estimates'
autoplot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
  lty_last_tc = 2,
  n_last_tc = 4,
  ...
)

## S3 method for class 'ts'
autoplot(object, xlim = NULL, ylim = NULL, ...)

```

Arguments

<code>x</code>	An object of class "tc_estimates".
<code>y</code>	Unused parameter, kept for compatibility with the generic function.
<code>xlim, ylim</code>	Limits for the x- and y-axes. If <code>xlim</code> is specified and <code>ylim</code> is <code>NULL</code> , <code>ylim</code> is determined automatically based on the truncated series.
<code>col_sa, col_tc</code>	Colors used for the seasonally adjusted and trend-cycle components, respectively.
<code>xlab, ylab</code>	Character strings for the x- and y-axis labels.
<code>lty_last_tc</code>	Line type for the last values of the trend-cycle component.

n_last_tc	Number of final values of the trend-cycle component to plot with a distinct line type (lty_last_tc), emphasizing higher uncertainty in recent estimates. If NULL (the default), n_last_tc is set to the Month of Cyclical Dominance (MCD) statistic.
...	Additional graphical parameters passed to internal plotting functions.
object	An object of class "tc_estimates".
legend_tc, legend_sa	Character strings specifying the legend labels for the trend-cycle and seasonally adjusted components, respectively.

Details

plot.tc_estimates() and autoplot.tc_estimates() produce a plot showing both the seasonally adjusted series and the estimated trend-cycle component. lines.tc_estimates() adds only the trend-cycle component to an existing base R plot.

Value

- plot() and lines() return NULL invisibly and are called for their side effect (drawing a plot).
- autoplot() returns a [ggplot](#) object.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
plot(tc_mod, xlim = c(2022, 2024.5))
```

segmented_smoothing	<i>Segmented Smoothing around Breakpoints</i>
---------------------	---

Description

Estimates the trend-cycle component by segmenting a time series around known structural breakpoints.

Usage

```
segmented_smoothing(
  x,
  breaks = NULL,
  break_method = c("one-side", "two-sides"),
  smoothing_method = c("henderson_smoothing", "clf_smoothing"),
  ...
)
```

Arguments

<code>x</code>	An input time series (object of class "ts" or "mts").
<code>breaks</code>	A vector or list of breakpoints (dates) where the series should be split for smoothing.
<code>break_method</code>	Character string specifying the method used to handle boundaries at breakpoints: "one-side" or "two-sides" (the default). See details.
<code>smoothing_method</code>	Function specifying the smoothing method to apply to each segment. Must be henderson_smoothing() (the default) or clf_smoothing() .
<code>...</code>	Additional arguments passed to <code>smoothing_method</code> .

Details

Two methods are available to estimate the trend-cycle around breakpoints:

1. "one-side": The trend-cycle is estimated across the entire dataset, and the estimates after each breakpoint are overwritten with the estimates obtained by smoothing only the data following that breakpoint.
2. "two-sides": The trend-cycle is estimated independently on each segment defined by the breakpoints, and the resulting estimates are combined to form the final trend-cycle.

The `parameters` field of the returned object corresponds to the parameters of the last segment (used to compute confidence intervals and implicit forecasts).

Value

An object of class `c("tc_estimates", "henderson")` if `smoothing_method` is [henderson_smoothing\(\)](#), or `c("tc_estimates", "clf")` if `smoothing_method` is [clf_smoothing\(\)](#).

Examples

```
x <- window(publishTC::french_ipi[, "manufacturing"], start = 2015)
tc_h <- henderson_smoothing(x, length = 13)$tc
breaks <- list(c(2020, 3))
tc_h_2s <- segmented_smoothing(x, breaks = breaks, break_method = "two-sides", length = 13)$tc
tc_h_rob <- henderson_robust_smoothing(x, ls = c(2020 + (3 - 1) / 12, 2020 + (4 - 1) / 12))$tc

plot(window(x, start = 2019.5, end = 2021),
     main = "Smoothing around COVID-19",
     xlab = NULL, ylab = NULL,
     lty = 3
)
lines(tc_h, col = "purple")
lines(tc_h_2s, col = "lightblue")
lines(tc_h_rob, col = "orange")
legend(
  "bottomright",
  legend = c("y", "Henderson", "Two-Sides Segmented", "Robust H. (2 LS)"),
  col = c("black", "purple", "lightblue", "orange"),
```

```

    lty = c(3, 1, 1, 1, 1),
    cex = 0.7
)

```

smoothing

Multi-Method Trend-Cycle Estimation

Description

Computes trend-cycle estimates using multiple smoothing methods simultaneously and returns a list of "tc_estimates" objects.

Usage

```

smoothing(
  x,
  methods = c("henderson", "henderson_localic", "henderson_robust",
    "henderson_robust_localic", "clf_cn", "clf_alf"),
  endpoints = "Musgrave",
  length = NULL,
  icr = NULL,
  asymmetric_var = FALSE,
  degree = 3,
  ao = NULL,
  ao_tc = NULL,
  ls = NULL,
  ...
)

```

Arguments

x	An input time series (object of class "ts" or "mts").
methods	A list of character strings specifying the smoothing methods to apply. Defaults to c("henderson", "henderson_localic", "clf").
endpoints	Character string specifying the method used for asymmetric filters. By default, "musgrave" is used.
length	An integer specifying the length of the symmetric filter. If NULL, the length is selected automatically using x11_trend_selection() .
icr	Numeric value specifying the Irregular-to-Trend (I/C) ratio used for asymmetric filters.
asymmetric_var	Logical. When local_icr = TRUE, if TRUE, the variance is estimated for each asymmetric filter. If FALSE (the default), the variance associated with symmetric estimates is used throughout.
degree	Integer. When local_icr = TRUE, degree of the polynomial used to estimate the local bias parameter.

ao	Vector of dates for Additive Outliers (AO) whose effects are allocated to the irregular component.
ao_tc	Vector of dates for Additive Outliers (AO) whose effects are allocated to the trend-cycle component.
ls	Vector of dates for Level Shifts (LS) whose effects are allocated to the trend-cycle component.
...	Additional parameters passed to the underlying smoothing functions.

Details

This convenience function applies multiple trend-cycle estimation algorithms to the same input series, facilitating direct comparison between different filtering approaches (e.g., standard Henderson, Henderson with local I/C ratio, or CLF smoothing).

Value

A named list of objects of class "tc_estimates", where each element corresponds to one of the specified methods in methods.

Examples

```
x <- french_ipi[, "manufacturing"]
outliers <- x13_regarima_outliers(x)
all_methods <- smoothing(x, ao = outliers$ao, ao_tc = outliers$ao_tc, ls = outliers$ls)

plot(all_methods$henderson, xlim = c(2023, 2025))
lines(all_methods$henderson_localic, col_tc = "blue")
```

smoothness

Measure Smoothness of Trend-Cycle Estimates

Description

Computes the smoothness ratio of trend-cycle estimates relative to the seasonally adjusted series following Picard and Matthews (2016).

Usage

```
smoothness(x)
```

Arguments

x An object of class "tc_estimates".

Details

Smoothness is evaluated as the ratio of month-to-month growth rate variances:

$$100 \times \sqrt{\frac{\sum_t \left(\frac{TC_t - TC_{t-1}}{TC_{t-1}} \right)^2}{\sum_t \left(\frac{SA_t - SA_{t-1}}{SA_{t-1}} \right)^2}}$$

This metric quantifies the relative variability of the trend-cycle compared to the seasonally adjusted series. A lower value indicates a smoother trend-cycle.

Value

A numeric value representing the smoothness index of the trend-cycle estimates.

References

Picard, F., & Matthews, S. (2016). The Addition of Trend-Cycle Estimates to Selected Publications at Statistics Canada. *Proceedings of the Survey Methods Section, Statistical Society of Canada (SSC) Annual Meeting*. https://ssc.ca/sites/default/files/imce/pdf/picard_ssc2016.pdf

Examples

```
x <- french_ipi[, "manufacturing"]
tc_clf <- clf_smoothing(x)
tc_henderson <- henderson_smoothing(x)
smoothness(tc_clf)
smoothness(tc_henderson)
```

tc_estimates	<i>Trend-Cycle Estimates Class</i>
--------------	------------------------------------

Description

Constructor and S3 methods for objects of class "tc_estimates".

Usage

```
tc_estimates(tc, sa, parameters = NULL, extra_class = NULL, ...)

is_tc_estimates(x)

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

Arguments

tc	Time series object ("ts") representing the estimated trend-cycle component.
sa	Time series object ("ts") representing the original or seasonally adjusted series.
parameters	A list of parameters used to compute the trend-cycle estimates (e.g., filter coefficients, variance, or degree of polynomial).
extra_class	Character string specifying an additional class to append to the returned object.
...	Additional arguments passed to or from other methods (currently unused).
x	An object of class "tc_estimates".
object	An object of class "tc_estimates".

Details

Objects of class "tc_estimates" store trend-cycle estimates along with the original series and metadata required for summary statistics, confidence intervals, and implicit forecasts.

Value

tc_estimates() returns an object of class c("tc_estimates", extra_class), which is a list containing the following components:

- tc: The estimated trend-cycle time series.
- x: The original (seasonally adjusted) time series.
- parameters: A list of parameters used during estimation.

The summary() method returns a list of class "summary.tc_estimates" containing:

- I/C ratio: The overall Irregular-to-Trend-Cycle ratio (see [icr\(\)](#)).
- I/C ratios: The I/C ratio per period (see [icrs\(\)](#)).
- MCD: The Month of Cyclical Dominance statistic (see [mcd\(\)](#)).
- Length: The length of the symmetric trend-cycle filter.

Examples

```
x <- cars_registrations
tc <- henderson_smoothing(x)
summary(tc)
print(tc)
```

`ts-exemple`*Example Datasets*

Description

Example datasets used in Quartier-la-Tente (2025).

Usage`cars_registrations``french_ipi``fred``simulated_data``etip`**Format**

An object of class `ts` of length 177.

An object of class `mts` (inherits from `ts`, `matrix`, `array`) with 416 rows and 3 columns.

An object of class `mts` (inherits from `ts`, `matrix`, `array`) with 766 rows and 2 columns.

An object of class `mts` (inherits from `ts`, `matrix`, `array`) with 84 rows and 6 columns.

An object of class `ts` of length 590.

Details

- `cars_registrations`: Monthly new passenger car registrations in France, published in October 2024.
- `french_ipi`: Monthly Industrial Production Index (IPI) in France for crude petroleum, motor vehicles, and manufacturing, published in October 2024.
- `fred`: The series CE160V (Civilian Employment Level) and RETAILx (Retail and Food Services Sales) from the FRED-MD database, published in November 2022.
- `simulated_data`: Simulated trends of degree 0, 1, and 2 with an Additive Outlier (AO) or Level Shift (LS) in January 2022.
- `etip`: Expected trend in production (balance of opinion) in the French manufacturing industry, published in May 2024 in the monthly business survey in goods-producing industries by INSEE.

References

Quartier-la-Tente, A. (2025). Estimation de la tendance-cycle avec des méthodes robustes aux points atypiques. <https://github.com/AQLT/robustMA>. McCracken, Michael W., et Serena Ng. 2016. FRED-MD: A Monthly Database for Macroeconomic Research. Journal of Business & Economic Statistics 34 (4): 574-89. doi:10.1080/07350015.2015.1086655.

turning_points

Detect Turning Points and Unwanted Ripples in a Time Series

Description

Identifies turning points (upturns and downturns) and unwanted short-term ripples in a time series object using local extremum criteria.

Usage

```
turning_points(x, start = NULL, end = NULL, digits = 6, k = 3, m = 1)
```

```
upturn(x, start = NULL, end = NULL, digits = 6, k = 3, m = 1)
```

```
downturn(x, start = NULL, end = NULL, digits = 6, k = 3, m = 1)
```

```
unwanted_ripples(x, start = NULL, end = NULL, digits = 6, k = 3, m = 1)
```

Arguments

<code>x</code>	An input time series (object of class "ts").
<code>start, end</code>	Numeric vectors specifying the start and end of the time interval (e.g., <code>c(2020, 1)</code>) in which to search for turning points.
<code>digits</code>	Integer specifying the number of decimal digits used when comparing values to determine turning points. Defaults to NULL (no rounding).
<code>k, m</code>	Integers specifying the required number of preceding (k) and succeeding (m) observations to define a local extremum. See details.

Details

Turning points are identified following the operational definition adapted from Zellner, Hong, and Min (1991), typically with $k = 3$ and $m = 1$:

- An **upturn** occurs at time t if:

$$y_{t-k} \geq \dots \geq y_{t-1} < y_t \leq y_{t+1} \leq \dots \leq y_{t+m}$$

- A **downturn** occurs at time t if:

$$y_{t-k} \leq \dots \leq y_{t-1} > y_t \geq y_{t+1} \geq \dots \geq y_{t+m}$$

An **unwanted ripple** is defined as any occurrence where two turning points of the same type (two upturns or two downturns) occur within a 10-month window (i.e., short cycles lasting less than 11 months).

Value

- `turning_points()` returns a named list with two components: "upturn" and "downturn", each containing a vector of dates (or empty vectors if none found).
- `upturn()` and `downturn()` return a vector of dates corresponding to the detected upturns or downturns, respectively.
- `unwanted_ripples()` returns an integer representing the number of unwanted ripples detected in the series.

References

Zellner, A., Hong, C., & Min, C. (1991). Forecasting Turning Points in International Growth Rates using Bayesian Exponential Smoothing Methods. *Journal of Econometrics*, 49(1–2), 275–304. doi:[10.1016/03044076\(91\)900167](https://doi.org/10.1016/03044076(91)900167)

Examples

```
tc <- henderson_smoothing(french_ipi[, "manufacturing"])
turning_points(tc)
unwanted_ripples(tc)
```

underlying_forecasts.henderson
Underlying Forecasts Methods

Description

S3 method of `rjd3filters::underlying_forecasts()` for "tc_estimates" objects.

Usage

```
## S3 method for class 'henderson'
underlying_forecasts(x, ...)

## S3 method for class 'clf'
underlying_forecasts(x, ...)

## S3 method for class 'robust_henderson'
underlying_forecasts(x, ...)
```

Arguments

`x` An object of class "tc_estimates".
`...` Additional arguments passed to internal methods (currently unused).

Details

See `rjd3filters::underlying_forecasts()` for details on the computation of underlying forecasts.

Value

A matrix or mts object containing the underlying forecasts for the series.

Examples

```
tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
underlying_forecasts(tc_mod)
```

underlying_forecasts_plot

Plot Underlying Forecasts for Trend-Cycle Estimates

Description

Base R (`underlying_forecasts_plot()`) and **ggplot2** (`ggunderlying_forecasts_plot()`) graphics for plotting trend-cycle estimates alongside their underlying forecasts.

Usage

```
underlying_forecasts_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_i_f = col_sa,
  xlab = "",
  ylab = "",
  lty_last_tc = 2,
  lty_i_f = 3,
  n_last_tc = 4,
  ...
)

ggunderlying_forecasts_plot(
  object,
  xlim = NULL,
  ylim = NULL,
  col_tc = "#E69F00",
  col_sa = "black",
  col_i_f = col_sa,
```

```

    lty_last_tc = 2,
    lty_i_f = 3,
    n_last_tc = 4,
    legend_tc = "Trend-cycle",
    legend_sa = "Seasonally adjusted",
    legend_i_f = "Implicit forecasts",
    ...
  )

```

Arguments

<code>object</code>	An object of class "tc_estimates".
<code>xlim, ylim</code>	Limits for the x- and y-axes. If <code>xlim</code> is specified and <code>ylim</code> is NULL, <code>ylim</code> is determined automatically based on the truncated series.
<code>col_sa, col_tc</code>	Colors used for the seasonally adjusted and trend-cycle components, respectively.
<code>col_i_f</code>	Color of the forecasts lines.
<code>xlab, ylab</code>	Character strings for the x- and y-axis labels.
<code>lty_last_tc, lty_i_f</code>	Line types used for the last values of the trend-cycle component and for the forecasts, respectively.
<code>n_last_tc</code>	Number of final values of the trend-cycle component to plot with a distinct line type (<code>lty_last_tc</code>), emphasizing higher uncertainty in recent estimates. If NULL (the default), <code>n_last_tc</code> is set to the Month of Cyclical Dominance (MCD) statistic.
<code>...</code>	Additional graphical parameters passed to internal plotting functions.
<code>legend_tc, legend_sa, legend_i_f</code>	Character strings specifying the legend labels for the trend-cycle, the seasonally adjusted series, and the forecasts, respectively.

Details

`underlying_forecasts_plot()` produces a base R graphic, whereas `ggunderlying_forecasts_plot()` creates a **ggplot2** object that can be further customized.

Value

- `underlying_forecasts_plot()` returns NULL invisibly and is called for its side effect (drawing a plot).
- `ggunderlying_forecasts_plot()` returns a **ggplot** object.

Examples

```

tc_mod <- henderson_smoothing(french_ipi[, "manufacturing"])
underlying_forecasts_plot(tc_mod, xlim = c(2022, 2025))

```

write.ts

*Export and Import Time Series to and from CSV***Description**

`write.ts()` exports a time series object ("ts") or a list of time series to a CSV file, adding a "time" column. `read.ts()` imports a CSV file whose first column contains dates and reconstructs the corresponding ts object.

Usage

```
write.ts(x, file, ...)
```

```
read.ts(file, frequency = NULL, list = FALSE)
```

Arguments

<code>x</code>	A time series object (class "ts") or a list of "ts" objects to export.
<code>file</code>	A character string specifying the path to the CSV file to write or read.
<code>...</code>	Additional arguments passed to <code>utils::write.csv()</code> or <code>utils::read.csv()</code> .
<code>frequency</code>	An integer specifying the number of observations per unit of time (e.g., 12 for monthly, 4 for quarterly). If NULL (the default), it is automatically inferred from the date column.
<code>list</code>	Logical. If TRUE, <code>read.ts()</code> returns a list of time series objects; otherwise, it returns a single "ts" or "mts" object.

Details

`write.ts()` formats the time index into a dedicated date column (e.g., YYYY-MM-DD or fractional years) alongside the data series. When reading back with `read.ts()`, the time index is parsed to reconstruct regular R time series objects.

Value

- `write.ts()` returns NULL invisibly and is called for its side effect (writing a CSV file).
- `read.ts()` returns a "ts" object, a multiple time series ("mts"), or a list of "ts" objects depending on `list`.

Examples

```
file <- tempfile(fileext = ".csv")
write.ts(AirPassengers, file)
read.ts(file)
```

x11_trend_selection *X-11 Selection of Trend-Cycle Filter*

Description

Performs X-11 selection for the length of the Henderson filter (`x11_trend_selection()`) and computes the associated I/C ratio used to build Musgrave filters (`find_icr()`).

Usage

```
x11_trend_selection(x)
```

```
find_icr(length, freq = 12)
```

Arguments

<code>x</code>	A "ts" object representing the time series.
<code>length</code>	Integer specifying the length of the filter.
<code>freq</code>	Integer specifying the frequency of the time series used to compute the I/C ratio (e.g., 12 for monthly, 4 for quarterly).

Details

The following procedure is used in X-11 to select the length of the trend filter:

1. Computes the I/C ratio, icr , with a Henderson filter of length equal to the series frequency plus 1.
2. The selected length depends on the value of icr :
 - If $icr < 1$, the selected length is 9 for monthly data and 5 otherwise.
 - If $1 \leq icr < 3.5$, the selected length is $freq + 1$ (i.e., 13 for monthly data, 5 for quarterly data).
 - If $icr \geq 3.5$, the selected length is 23 for monthly data and 7 otherwise.
3. The value of icr is then mapped to build Musgrave filters (`find_icr()`):
 - For quarterly data, if the length is 5, then $icr = 0.001$; otherwise, $icr = 4.5$.
 - For other frequencies, if the length is less than or equal to 9, then $icr = 1.0$.
 - Else, if the length is less than or equal to 13, then $icr = 3.5$.
 - Else, $icr = 4.5$.

Value

- `x11_trend_selection()` returns a named numeric vector containing the selected length and the associated I/C ratio.
- `find_icr()` returns a single numeric value corresponding to the I/C ratio associated with the specified filter length and frequency in the X-11 algorithm.

Examples

```
x11_trend_selection(cars_registrations)
find_icr(length = 13, freq = 12)
```

x13_regarima_outliers *Outlier Detection with RegARIMA Model*

Description

Wrapper around `rjd3x13::regarima_outliers()` to detect Additive Outliers (AO) or Level Shifts (LS) in a seasonally adjusted series.

Usage

```
x13_regarima_outliers(
  y,
  order = c(0, 1, 1),
  mean = FALSE,
  ao = TRUE,
  ls = TRUE
)
```

Arguments

<code>y</code>	A time series object (class "ts").
<code>order</code>	An integer vector of length 3 specifying the non-seasonal ARIMA orders (p, d, q) .
<code>mean</code>	Logical. If TRUE, the model includes a constant term.
<code>ao, ls</code>	Logical flags indicating whether Additive Outliers (AO) and Level Shifts (LS) should be detected, respectively.

Details

This function uses an underlying RegARIMA model via **rjd3x13** to automatically identify structural breaks and temporary shocks in the input series, returning their locations for downstream robust trend-cycle estimation.

Value

A list with two components:

- `ao`: A vector of dates where Additive Outliers were detected (or NULL if none).
- `ls`: A vector of dates where Level Shifts were detected (or NULL if none).

Examples

```
x13_regarima_outliers(cars_registrations)
```

Index

- * **datasets**
 - classical-ma, 3
 - ts-exemple, 27
- autoplot.tc_estimates
 - (plot.tc_estimates), 19
- autoplot.tc_estimates(), 8
- autoplot.ts (plot.tc_estimates), 19
- bandwidth, 2
- cars_registrations (ts-exemple), 27
- classical-ma, 3
- CLF (classical-ma), 3
- CLF_CN (classical-ma), 3
- clf_smoothing, 4
- clf_smoothing(), 22
- confint(), 6
- confint.clf (confint_tc), 6
- confint.henderson (confint_tc), 6
- confint.robust_henderson (confint_tc), 6
- confint_plot, 5
- confint_tc, 6
- downturn (turning_points), 28
- etip (ts-exemple), 27
- find_icr (x11_trend_selection), 33
- fred (ts-exemple), 27
- french_ipi (ts-exemple), 27
- ggconfint_plot (confint_plot), 5
- ggconfint_plot(), 8
- gggrowthplot (growthplot), 8
- ggimplicit_forecasts_plot
 - (implicit_forecasts_plot), 15
- ggimplicit_forecasts_plot(), 8
- gglollypop (lollypop), 17
- gglollypop(), 8
- ggplot, 6, 8, 10, 16, 18, 21, 31
- ggsmoothing_plot, 7
- ggunderlying_forecasts_plot
 - (underlying_forecasts_plot), 30
- ggunderlying_forecasts_plot(), 8
- growthplot, 8
- henderson (classical-ma), 3
- henderson_robust_smoothing, 10
- henderson_smoothing, 12
- henderson_smoothing(), 22
- icr, 13
- icr(), 26
- icrs (icr), 13
- icrs(), 26
- implicit_forecasts.clf
 - (implicit_forecasts.henderson), 14
- implicit_forecasts.henderson, 14
- implicit_forecasts.robust_henderson
 - (implicit_forecasts.henderson), 14
- implicit_forecasts_plot, 15
- is_tc_estimates (tc_estimates), 25
- lines.tc_estimates (plot.tc_estimates), 19
- local_param_est (classical-ma), 3
- lollypop, 17
- mcd, 18
- mcd(), 26
- plot.tc_estimates, 19
- read.ts (write.ts), 32
- rjd3filters::confint_filter(), 7
- rjd3filters::implicit_forecasts(), 14
- rjd3filters::lp_filter(), 11, 12
- rjd3filters::underlying_forecasts(), 29, 30

`rjd3x13::regarima_outliers()`, [34](#)

`segmented_smoothing`, [21](#)

`simulated_data(ts-exemple)`, [27](#)

`smoothing`, [23](#)

`smoothness`, [24](#)

`summary.tc_estimates(tc_estimates)`, [25](#)

`tc_estimates`, [25](#)

`tc_estimates()`, [4](#), [11](#), [12](#)

`ts-exemple`, [27](#)

`turning_points`, [28](#)

`underlying_forecasts.clf`
 (`underlying_forecasts.henderson`),
 [29](#)

`underlying_forecasts.henderson`, [29](#)

`underlying_forecasts.robust_henderson`
 (`underlying_forecasts.henderson`),
 [29](#)

`underlying_forecasts_plot`, [30](#)

`unwanted_ripples(turning_points)`, [28](#)

`upturn(turning_points)`, [28](#)

`utils::read.csv()`, [32](#)

`utils::write.csv()`, [32](#)

`write.ts`, [32](#)

`x11_trend_selection`, [33](#)

`x11_trend_selection()`, [10](#), [12](#), [23](#)

`x13_regarima_outliers`, [34](#)