Package: publishTC (via r-universe)

March 16, 2025

Type Package

Title Tools to help publish the trend-cycle

Version 0.1.0.9000

Description This package provides several functions to facilitate the computation of trend-cycle component. In particular, the computation can be done: using the Cascade Linear Filter (CLF) Dagum, E. B., & Luati, A. (2008); using the classical Henderson symmetric filter and the surrogate Musgrave asymmetric filters; using a local Parametrization of the Musgrave asymmetric filters (Quartier-la-Tente 2024); extending the Henderson symmetric fiter and the surrogate Musgrave asymmetric filters to take into account additive outliers and level shifts (Quartier-la-Tente 2025). Confidence intervals can be computed and several plots are available.

LazyData true

Depends R (>= 4.1.0)

Imports rjd3filters, rjd3x13, ggplot2, zoo, rJava

Remotes github::rjdverse/rjd3x13 github::rjdverse/rjd3filters

URL https://github.com/AQLT/publishTC

SystemRequirements Java (>= 17)

License EUPL

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

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

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

RemoteRef HEAD

RemoteSha f42864c658c86f323ae619e7d35696c48c23d8cd

Contents

`bandwidth` *Get Bandwidth*

Description

Get the bandwidth of a `"tc_estimates"` object. The length of the filter is then equal to $2 \times \text{bandwidth}(x) + 1$.

Usage

```
bandwidth(x)
```

Arguments

`x`	a `"tc_estimates"` object.

classical-ma *Classical Moving Average*

Description

Classical moving average for trend-cycle extraction.

Usage

```
henderson

CLF

CLF_CN

local_param_est
```

Format

`henderson` is `list()` of "`moving_average`".

`CLF` is a "`finite_filters`".

`CLF_CN` is a "`finite_filters`".

`local_param_est` is `list()` of "`finite_filters`".

Details

`henderson` contains the Henderson moving average of length 5, 7, 9, 13 and 23.

`CLF` contains the Cascade Linear Filter (CLF) of length 13 and the associated Asymmetric Linear Filters (ALF).

`CLF_CN` contains the Cascade Linear Filter (CLF) of length 13 and the associated cut and normalise asymetric filters.

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. https://doi.org/10.1080/07474930802387837.

Henderson, R. (1916). Note on graduation by adjusted average. *Transactions of the actuarial society of America* 17: 43-48.

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. https://doi.org/10.1177/0282423X241283207.

clf_smoothing — *Smoothing using the Cascade Linear Filter*

Description

Smoothing using the Cascade Linear Filter

Usage

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

Arguments

`x`	input time-series.
`endpoints`	Method used for the asymmetric filter. If `endpoints = "cut-and-normalize"` (the default) the cut-and-normalise method is used, otherwise the Asymmetric Linear Filter (ALF) filters are used.
`...`	other unused parameters.

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. https://doi.org/10.1080/07474930802387837

confint-tc — *Confidence Intervals for* `"tc_estimates"`

Description

Confidence Intervals for `"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`	a "`tc_estimates`" object.
`parm`	unused parameter.
`level`	the confidence level required.
`asymmetric_var`	if `asymmetric_var = TRUE` then the variance is estimated for each asymmetric filters instead of using the variance associated the symmetric estimates.
`...`	other (unused) parameters.

`confint_plot` *Confidence Intervals plot*

Description

Confidence Intervals plot

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,
  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	"tc_estimates". The confidence intervals are computed using the confint() function.
xlim, ylim	x and y limits of the plot. If NULL (the default), then the limits determined automatically.
col_sa, col_tc	color of the seasonally adjusted and trend-cycle components.
col_confint	color of the confidence interval.
xlab, ylab	x and y axis labels.
level	the confidence level required.
...	other parameters.
legend_tc, legend_sa, legend_confint	legend of the trend-cycle and seasonally adjusted components and for the confidence intervals.

---

ggsmoothing_plot *Produce several plots*

Description

Produce several plots

Usage

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

Arguments

object	"tc_estimates". The confidence intervals are computed using the confint() function.
plots	list of plots to use.
level	the confidence level required.
...	other unused parameters.

henderson_robust_smoothing
Smoothing using the Henderson filter

Description

Smoothing using the Henderson filter

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

`x`	input time-series.
`endpoints`	Method used for the asymmetric filter. By default the Musgrave method is used
`length`	the length of the
`ao`	Dates of the Additive Outliers (AO) which effects are associated to the irregular component.
`ao_tc`	Dates of the Additive Outliers (AO) which effects are associated to the trend-cycle component.
`ls`	Dates of the Level Shifts (LS) which effects are associated to the trend-cycle component.
`icr`	I/C ratio used for the asymmetric filter.
`local_icr`	if `TRUE`, the I/C ratio is estimated locally (as described in Quartier-la-Tente, A. (2024)) instead of globally.
`asymmetric_var`	when `local_icr = TRUE`, if `asymmetric_var = TRUE` then the variance is estimated for each asymmetric filters instead of using the variance associated to the symmetric estimates.
`degree`	if `local_icr = TRUE`, degree of polynomial used to estimate the local bias parameter.
`...`	other parameters passed to `rjd3filters::lp_filter()`.

henderson_smoothing *Smoothing using the Henderson filter*

Description

Smoothing using the Henderson filter

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

`x`	input time-series.
`endpoints`	Method used for the asymmetric filter. By default the Musgrave method is used
`length`	the length of the
`icr`	I/C ratio used for the asymmetric filter.
`local_icr`	if `TRUE`, the I/C ratio is estimated locally (as described in Quartier-la-Tente, A. (2024)) instead of globally.
`asymmetric_var`	when `local_icr = TRUE`, if `asymmetric_var = TRUE` then the variance is estimated for each asymmetric filters instead of using the variance associated to the symmetric estimates.
`degree`	if `local_icr = TRUE`, degree of polynomial used to estimate the local bias parameter.
`...`	other parameters passed to `rjd3filters::lp_filter()`.

References

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. https://doi.org/10.1177/0282423X241283207.

icr — *Compute IC-Ratio*

Description

`icr()` compute the overall I/C ratio, while `icrs()` compute the I/C ratios for each period.

Usage

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

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

Arguments

`x, tc`	seasonally adjusted and trend-cycle components. If `x` is a `"tc_estimates"` object then `tc` is ignored.
`mul`	boolean indicating if the decomposition is multiplicative or additive.

Examples

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

implicit_forecasts — *Compute Implicit Forecasts*

Description

Compute Implicit Forecasts

Usage

```
implicit_forecasts(x, ...)
```

Arguments

`x`	a `"tc_estimates"` object otherwise uses the `rjd3filters::implicit_forecast()` function.
`...`	other unused parameters.

`implicit_forecasts_plot` *Implicit Forecasts plot*

Description

Implicit Forecasts plot

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,
  ...
)

ggimplicit_forecasts_plot(
  object,
  col_tc = "#E69F00",
  col_sa = "black",
  col_i_f = col_sa,
  lty_last_tc = 2,
  lty_i_f = 3,
  legend_tc = "Trend-cycle",
  legend_sa = "Seasonally adjusted",
  legend_i_f = "Implicit forecasts",
  ...
)
```

Arguments

`object`	`"tc_estimates"` object. If `object` is a `"tc_estimates"` object then `sa` is optional.
`xlim, ylim`	x and y limits of the plot. If `NULL` (the default), then the limits determined automatically.
`col_sa, col_tc`	color of the seasonally adjusted and trend-cycle components.
`col_i_f`	color of the implicit forecasts.
`xlab, ylab`	x and y axis labels.

`lty_last_tc`, `lty_i_f`
: line type of the last values of the trend-cycle component and for the implicit forecasts.

`...`
: other parameters.

`legend_tc`, `legend_sa`, `legend_i_f`
: legend of the trend-cycle and seasonally adjusted components and for implicit forecasts.

`lollypop` *Lollypop plot*

Description

Lollypop plot

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 = "",
  ...
)

gglollypop(
  object,
  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,
  ...
)
```

Arguments

object	"tc_estimates" object. If object is a "tc_estimates" object then sa is optional.
xlim, ylim	x and y limits of the plot. If NULL (the default), then the limits determined automatically.
col_sa, col_tc	color of the seasonally adjusted and trend-cycle components.
color_points, cex_points	color and size of the points associated to the seasonnaly adjusted component.
pch_points	point type of the seasonally adjusted component.
xlab, ylab	x and y axis labels.
...	other parameters.
legend_tc, legend_sa	legend of the trend-cycle and seasonally adjusted components.
lty_last_tc	line type of the last values of the trend-cycle component.

---

mcd — *Month of Cyclical Dominance*

Description

Month of Cyclical Dominance

Usage

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

Arguments

x, tc	seasonally adjusted and trend-cycle components. If x is a "tc_estimates" object then tc is ignored.
mul	boolean indicating if the decomposition is multiplicative or additive.

plot.tc_estimates *Default* "tc_estimates" *plot*

Description

Default "tc_estimates" plot

Usage

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

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

Arguments

y	unused parameter.
col_sa, col_tc	color of the seasonally adjusted and trend-cycle components.
xlab, ylab	x and y axis labels.
lty_last_tc	line type of the last values of the trend-cycle component.
...	other (unused) parameters.
object, x	"tc_estimates" object.
legend_tc, legend_sa	legend of the trend-cycle and seasonally adjusted components.

smoothing *Smoothing using several methods*

Description

Smoothing using several methods

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	input time-series.
methods	list of methods to use.
endpoints	Method used for the asymmetric filter. By default the Musgrave method is used
length	the length of the
icr	I/C ratio used for the asymmetric filter.
asymmetric_var	when `local_icr = TRUE`, if `asymmetric_var = TRUE` then the variance is estimated for each asymmetric filters instead of using the variance associated to the symmetric estimates.
degree	if `local_icr = TRUE`, degree of polynomial used to estimate the local bias parameter.
ao	Dates of the Additive Outliers (AO) which effects are associated to the irregular component.
ao_tc	Dates of the Additive Outliers (AO) which effects are associated to the trend-cycle component.
ls	Dates of the Level Shifts (LS) which effects are associated to the trend-cycle component.
...	other unused parameters.

ts-exemple — *Data set examples*

Description

Data set examples

Usage

```
cars_registrations

french_ipi

fred

simulated_data
```

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.

turning_points — *Detect turning points in a time series*

Description

Detect turning points in a time series

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)
```

Arguments

`x`	the input time series.
`start, end`	the interval where to find turning points.
`digits`	number of digits used for the comparison of the values.
`k, m`	number of observation before and after the turning point (see details).

Details

Zellner, Hong, et Min (1991) definition is used $k = 3$, $m = 1$:

- we have an upturn at date t when

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

- we have a downturn at date t when

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

---

write.ts *Export and Import time series object to/from CSV*

Description

Export and Import time series object to/from CSV

Usage

```
write.ts(x, file)

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

Arguments

`x`	a time series object
`file`	a character string giving the name of the file to write to.
`frequency`	an integer giving the number of observations per unit of time. By default it is guessed from the data.
`list`	boolean, if `TRUE`, the function returns a list of time series objects.

---

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

Description

Perform X-11 selection of the length of Henderson (`x11_trend_selection()`) and compute the associated I/C ratio used to build Musgrave fuilters (`find_icr()`).

Usage

```
x11_trend_selection(x)

find_icr(length, freq = 12)
```

Arguments

`x`	a `"ts"` object.
`length`	length of the filter.
`freq`	frequency of the time series used to compute the I/C ratio.

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 an Henderson filter of length the frequency plus 1.
2. The length depends on the value or icr:
   - if $icr < 1$ then the selected length is 9 for monthly data and 5 otherwise;
   - if $1 \leq icr < 3.5$ then the selected length is $freq + 1$ where $freq$ is the frequency of data (12 for monthly data, 4 for quarterly data...).
   - if $icr \geq 3.5$ then the selected length is 23 for monthly data and 7 otherwise.
3. The value of icr is then fixed to build Musgrave filters (`find_icr()`) :
   - for quarterly data, if the length is 5 then $icr = 0.001$, otherwide $icr = 4.5$;
   - if the length if less or equal to 9 then $icr = 1$;
   - else if the length if less or equal to 13 then $icr = 3.5$;
   - else $icr = 4.5$.

Index