Package 'smoots'

December 2, 2019

Type Package

Title Nonparametric Estimation of the Trend and Its Derivatives in TS

Version 1.0.1

Description The nonparametric trend and its derivatives in equidistant time series (TS) with short-memory stationary errors can be estimated. The estimation is conducted via local polynomial regression using an automatically selected bandwidth obtained by a built-in iterative plug-in algorithm or a bandwidth fixed by the user. A Nadaraya-Watson kernel smoother is also built-in as a comparison.
The methods of the package are described in
Feng, Y., and Gries, T., (2017) <http://groups.uni-paderborn.de/wp-wiwi/RePEc/pdf/ciepap/WP102.pdf>.
A current version of the paper that is also referred to in the documentation of the functions is prepared for publication.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.1

Imports stats, graphics

Suggests knitr, rmarkdown, fGarch

Depends R (>= 2.10)

URL `https://wiwi.uni-paderborn.de/en/dep4/feng/`
`https://wiwi.uni-paderborn.de/dep4/gries/`

NeedsCompilation no

Author Yuanhua Feng [aut] (Paderborn University, Germany),
Dominik Schulz [aut, cre] (Paderborn University, Germany),
Thomas Gries [ctb] (Paderborn University, Germany),
Marlon Fritz [ctb] (Paderborn University, Germany),
Sebastian Letmathe [ctb] (Paderborn University, Germany)

Maintainer Dominik Schulz <`schulzd@mail.uni-paderborn.de`>

Repository CRAN

Date/Publication 2019-12-02 15:40:02 UTC

R topics documented:

`dax` *German Stock Market Index (DAX) Financial Time Series Data*

Description

A dataset that contains the daily financial data of the DAX from 1990 to July 2019 (currency in EUR).

Usage

```
dax
```

Format

A data frame with 7475 rows and 9 variables:

Year the observation year
Month the observation month
Day the observation day
Open the opening price of the day
High the highest price of the day
Low the lowest price of the day
Close the closing price of the day
AdjClose the adjusted closing price of the day
Volume the traded volume

Source

The data was obtained from Yahoo Finance.

https://query1.finance.yahoo.com/v7/finance/download/^GDAXI?period1=631148400&period2=1564524000&interval=1d&events=history&crumb=Iaq1EPZAQRb

`dsmooth` *Data-driven Local Polynomial for the Trend's Derivatives in Equidistant Time Series*

Description

This function runs through an iterative process in order to find the optimal bandwidth for the nonparametric estimation of the first or second derivative of the trend in an equidistant time series (with short-memory errors) and subsequently employs the obtained bandwidth via local polynomial regression.

Usage

```
dsmooth(
  y,
  d = c(1, 2),
  mu = c(0, 1, 2, 3),
  pp = c(1, 3),
  bStart.p = 0.15,
  bStart = 0.15
)
```

Arguments

`y` a numeric vector that contains the time series ordered from past to present.

`d` an integer 1 or 2 that defines the order of derivative.

`mu` an integer 0, ..., 3 that represents the smoothness parameter of the kernel weighting function and thus defines the kernel function that will be used within the local polynomial regression; is set to *1* by default.

Number	**Kernel**
0	Uniform Kernel
1	Epanechnikov Kernel
2	Bisquare Kernel
3	Triweight Kernel

`pp` an integer 1 (local linear regression) or 3 (local cubic regression) that indicates the order of polynomial upon which c_[f], i.e. the variance factor, will be calculated.

`bStart.p` a numeric object that indicates the starting value of the bandwidth for the iterative process for the calculation of c_[f]; should be 0 < bStart.p < 0.5; is set to *0.15* by default.

`bStart` a numeric object that indicates the starting value of the bandwidth for the iterative process; should be 0 < bStart < 0.5; is set to *0.15* by default.

Details

The iterative-plug-in (IPI) algorithm, which numerically minimizes the Asymptotic Mean Squared Error (AMISE), was proposed by Feng et al. (2019).

Define I[m^(k)] = int_[c_[b]]^[d_[b]] [m^(k)(x)]^2 dx,

beta_[v,k] = int_[-1]^[1] u^k K(u)du and

R(K) = int_[-1]^[1] K^2(u)du.

The AMISE is then

AMISE(h) = h^(2(k-v)) * (I[m^(k)]beta^2 / [k]^2) + (2pi * c_[f](d_[b]-c_[b])R(K) / nh^(2v+1))

with h being the bandwidth, k = p + 1 being the order of the equivalent kernel, v being the order of derivative, 0 <= c_[b] < d_[b] <= 1, n being the number of observations, c_[f] being the variance factor and K_[(v,k)](u) being the k-th order equivalent kernel obtained for the estimation of m^[(v)] in the interior. m^[(v)] is the v-th order derivative (v = 0, 1, 2, ...) of the nonparametric trend.

The variance factor c_[f] is first obtained from a pilot-estimation of the time series' nonparametric trend (v = 0) with polynomial order p_[p]. The estimate is then plugged into the iterative procedure for estimating the first or second derivative (v = 1 or v = 2). For further details on the asymptotic theory or the algorithm, we refer the user to Feng, Fritz and Gries (2019) and Feng et al. (2019).

The function itself is applicable in the following way: Based on a data input *y*, an order of polynomial *pp* for the variance factor estimation procedure, a starting value for the relative bandwidth *bStart.p* in the variance factor estimation procedure, a kernel function defined by the smoothness parameter *mu* and a starting value for the relative bandwidth *bStart* in the bandwidth estimation procedure, an optimal bandwidth is numerically calculated for the derivative of order *d*. In fact, aside from the input vector *y*, every argument has a default setting that can be adjusted for the individual case. However, it is recommended to initially use the default values for the estimation of the first derivative and adjust the argument *d* to *d = 2* for the estimation of the second derivative. Following Feng, Gries and Fritz (2019), the initial bandwidth does not affect the resulting optimal bandwidth in theory. However in practice, local minima of the AMISE can influence the results. Therefore, the default starting bandwidth is set to 0.15, the suggested starting bandwidth by Feng, Gries and Fritz (2019) for the data-driven estimation of the first derivative. The recommended initial bandwidth for the second derivative, however, is 0.2 and not 0.15. Thus, if the algorithm does not give suitable results (especially for *d = 2*), the adjustment of the initial bandwidth might be a good starting point. Analogously, the default starting bandwith for the trend estimation for the variance factor is *bStart.p = 0.15*, although according to Feng, Gries and Fritz (2019), *bStart.p = 0.1* is suggested for *pp = 1* and *bStart.p = 0.2* for *pp = 3*. The default is therefore a compromise between the two suggested values. For more specific information on the input arguments consult the section *Arguments*.

After the bandwidth estimation, the nonparametric derivative of the series is calulated with respect to the obtained optimal bandwidth by means of a local polynomial regression. The output object is then a list that contains, among other components, the original time series, the estimates of the derivative and the estimated optimal bandwidth.

The default print method for this function delivers key numbers such as the iteration steps and the generated optimal bandwidth rounded to the fourth decimal. The exact numbers and results such as the estimated nonparametric trend series are saved within the output object and can be addressed via the *$* sign.

Value

The function returns a list with different components:

b0 the optimal bandwidth chosen by the IPI-algorithm.

bStart the starting bandwidth for the local polynomial regression based derivative estimation procedure; input argument.

bStart.p the starting bandwidth for the nonparametric trend estimation that leads to the variance factor estimate; input argument.

cf0 the estimated variance factor.

InfR the inflation rate setting.

iterations the bandwidths of the single iterations steps

mu the smoothness parameter of the second order kernel; input argument.

n the number of observations.

niterations the total number of iterations until convergence.

orig the original input series; input argument.

p the order of polynomial for the local polynomial regression used within derivative estimation procedure.

pp the order of polynomial for the local polynomial regression used in the variance factor estimation; input argument.

res the estimated residual series.

ws the weighting systems used within the local polynomial regression; only exists, if the final smoothing is done via a local polynomial regression.

ye the nonparametric estimates of the derivative.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: `https://wiwi.uni-paderborn.de/en/dep4/feng/`
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Feng, Y., Gries, T. and Fritz, M. (2019). Data-driven local polynomial for the trend and its derivatives in economic time series. Discussion Paper. Paderborn University. (Not yet publshed)

Feng, Y., Gries, T., Letmathe, S. and Schulz, D. (2019). The smoots package in R for semiparametric modeling of trend stationary time series. Discussion Paper. Paderborn University. (Not yet published)

Examples

```
# Logarithm of test data
test_data <- gdpUS
y <- log(test_data$GDP)
t <- seq(from = 1947, to = 2019.25, by = 0.25)

# Applied dsmooth function for the trend's first derivative
```

```
result_d <- dsmooth(y, d = 1, mu = 1, pp = 1, bStart.p = 0.1, bStart = 0.15)
estim <- result_d$ye

# Plot of the results
plot(t, estim, xlab = "Year", ylab = "First derivative", type = "l",
 main = "Estimated first derivative of the trend for log-quarterly US-GDP, Q1 1947 - Q2 2019",
 cex.axis = 0.8, cex.main = 0.8, cex.lab = 0.8, bty = "n")

# Print result
result_d
```

gdpUS *Quarterly US GDP, Q1 1947 to Q2 2019*

Description

A dataset that contains the (seasonally adjusted) Gross Domestic Product of the US from the first quarter of 1947 to the second quarter of 2019

Usage

```
gdpUS
```

Format

A data frame with 290 rows and 3 variables:

Year the observation year

Quarter the observation quarter in the given year

GDP the Gross Domestic Product of the US in billions of chained 2012 US Dollars

Source

The data was obtained from the Federal Reserve Bank of St. Louis.

https://fred.stlouisfed.org/series/GDPC1

gsmooth — *Estimation of Trends and their Derivatives via Local Polynomial Regression*

Description

This function is an R function for estimating the trend function and its derivatives in an equidistant time series with local polynomial regression and a fixed bandwidth given beforehand.

Usage

```
gsmooth(y, v = 0, p = 1, mu = 1, b = 0.15, bb = c(0, 1))
```

Arguments

`y` a numeric vector that contains the time series data ordered from past to present.

`v` an integer 0, 1, ... that represents the order of derivative that will be estimated.

Number (v)	**Degree of derivative**
0	The function *f(x)* itself
1	The first derivative *f'(x)*
2	The second derivative *f"(x)*
...	...

`p` an integer >= (v + 1) that represents the order of polynomial; if p - v is odd, this approach has automatic boundary correction.

Examplary for v = 0:

Number (p)	**Polynomial**	**p - v**	**p - v odd?**	**p usable?**
1	Linear	1	Yes	Yes
2	Quadratic	2	No	No
3	Cubic	3	Yes	Yes
...	...	...	...	...

`mu` an integer 0, 1, 2, ... that represents the smoothness parameter of the kernel weighting function that will be used; is set to *1* by default.

Number (mu)	**Kernel**
0	Uniform Kernel
1	Epanechnikov Kernel
2	Bisquare Kernel
3	Triweight Kernel
...	...

`b` a real number 0 < b < 0.5; represents the relative bandwidth that will be used for the smoothing process; is set to *0.15* by default.

bb	can be set to *0* or *1*; the parameter controlling the bandwidth used at the boundary; is set to *0* by default.

Number (bb)	**Estimation procedure at boundary points**
0	Fixed bandwidth on one side with possible large bandwidth on the other side at the boundary
1	The k-nearest neighbor method will be used

Details

The trend or its derivatives are estimated based on the additive nonparametric regression model for a time series

y_[t] = m(x_[t]) + eps_[t],

where y_[t] is the observed time series, x_[t] is the rescaled time on [0, 1], m(x_[t]) is a smooth trend function and eps_[t] are stationary errors with E(eps_[t]) = 0 (see also Beran and Feng, 2002).

This function is part of the package *smoots* and is used in the field of analysing equidistant time series data. It applies the local polynomial regression method to the input data with an arbitrarily selectable bandwidth. By these means, the trend as well as its derivatives can be estimated nonparametrically, even though the result will strongly depend on the bandwidth given beforehand as input.

Value

The output object is a list with different components:

b the chosen (relative) bandwidth; input argument.

bb the chosen bandwidth option at the boundaries; input argument.

mu the chosen smoothness parameter for the second order kernel; input argument.

n the number of observations.

orig the original input series; input argument.

p the chosen order of polynomial; input argument.

res a vector with the estimated residual series; only meaningful for *v = 0*.

v the order of derivative; input argument.

ws the obtained weighting system matrix for possible advanced analysis.

ye a vector with the estimates of the selected nonparametric order of derivative.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: https://wiwi.uni-paderborn.de/en/dep4/feng/
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Beran, J. and Feng, Y. (2002). Local polynomial fitting with long-memory, short-memory and antipersistent errors. Annals of the Institute of Statistical Mathematics, 54(2), 291-311.

Feng, Y., Gries, T. and Fritz, M. (2019). Data-driven local polynomial for the trend and its derivatives in economic time series. Discussion Paper. Paderborn University. (Not yet published)

Feng, Y., Gries, T., Letmathe, S. and Schulz, D. (2019). The smoots package in R for semiparametric modeling of trend stationary time series. Discussion Paper. Paderborn University. (Not yet published)

Examples

```
# Logarithm of test data
test_data <- gdpUS
y <- log(test_data$GDP)

# Applied gsmooth function for the trend with two different bandwidths
results1 <- gsmooth(y, v = 0, p = 1, mu = 1, b = 0.28, bb = 1)
results2 <- gsmooth(y, v = 0, p = 1, mu = 1, b = 0.11, bb = 1)
trend1 <- results1$ye
trend2 <- results2$ye

# Plot of the results
t <- seq(from = 1947, to = 2019.25, by = 0.25)
plot(t, y, type = "l", xlab = "Year", ylab = "log(US-GDP)", bty = "n",
    lwd = 2,
    main = "Estimated trend for log-quarterly US-GDP, Q1 1947 - Q2 2019")
points(t, trend1, type = "l", col = 2, lwd = 1)
points(t, trend2, type = "l", col = 4, lwd = 1)
legend("bottomright", legend = c("Trend (b = 0.28)", "Trend (b = 0.11)"),
     fill = c(2, 4), cex = 0.6)
title(sub = expression(italic("Figure 1")), col.sub = "gray47",
     cex.sub = 0.6, adj = 0)
```

`knsmooth` *Estimation of Nonparametric Trend Functions via Kernel Regression*

Description

This function estimates the nonparametric trend function in an equidistant time series with Nadaraya-Watson kernel regression.

Usage

```
knsmooth(y, mu = 1, b = 0.15, bb = c(0, 1))
```

Arguments

`y`	a numeric vector that contains the time series data ordered from past to present.
`mu`	an integer 0, 1, 2, ... that represents the smoothness parameter of the second order kernel function that will be used; is set to *1* by default.

Number (mu)	**Kernel**
0	Uniform Kernel
1	Epanechnikov Kernel
2	Bisquare Kernel
3	Triweight Kernel
...	...

`b`	a real number 0 < b < 0.5; represents the relative bandwidth that will be used for the smoothing process; is set to *0.15* by default.
`bb`	can be set to 0 or 1; the parameter controlling the bandwidth used at the boundary; is set to *0* by default.

Number (bb)	**Estimation procedure at boundary points**
0	Fixed bandwidth on one side with possible large bandwidth on the other side at the boundary
1	The k-nearest neighbor method will be used

Details

The trend or its derivatives are estimated based on the additive nonparametric regression model for a time series

y_[t] = m(x_[t]) + eps_[t],

where y_[t] is the observed time series, x_[t] is the rescaled time on [0, 1], m(x_[t]) is a smooth trend function and eps_[t] are stationary errors with E(eps_[t]) = 0.

This function is part of the package *smoots* and is used for the estimation of trends in equidistant time series. The applied method is a kernel regression with arbitrarily selectable second order kernel, relative bandwidth and boundary method. Especially the chosen bandwidth has a strong impact on the final result and has thus to be selected carefully. This approach is not recommended by the authors of this package.

Value

The output object is a list with different components:

b the chosen (relative) bandwidth; input argument.

bb the chosen bandwidth option at the boundaries; input argument.

mu the chosen smoothness parameter for the second order kernel; input argument.

n the number of observations.

orig the original input series; input argument.

res a vector with the estimated residual series.

ye a vector with the estimates of the nonparametric trend.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: https://wiwi.uni-paderborn.de/en/dep4/feng/
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Feng, Y. (2009). Kernel and Locally Weighted Regression. Verlag für Wissenschaft und Forschung, Berlin.

Examples

```
# Logarithm of test data
test_data <- gdpUS
y <- log(test_data$GDP)

#Applied knmooth function for the trend with two different bandwidths
trend1 <- knsmooth(y, mu = 1, b = 0.28, bb = 1)$ye
trend2 <- knsmooth(y, mu = 1, b = 0.05, bb = 1)$ye

# Plot of the results
t <- seq(from = 1947, to = 2019.25, by = 0.25)
plot(t, y, type = "l", xlab = "Year", ylab = "log(US-GDP)", bty = "n",
    lwd = 2,
    main = "Estimated trend for log-quarterly US-GDP, Q1 1947 - Q2 2019")
points(t, trend1, type = "l", col = 2, lwd = 1)
points(t, trend2, type = "l", col = 4, lwd = 1)
legend("bottomright", legend = c("Trend (b = 0.28)", "Trend (b = 0.05)"),
     fill = c(2, 4), cex = 0.6)
title(sub = expression(italic("Figure 1")), col.sub = "gray47",
     cex.sub = 0.6, adj = 0)
```

`msmooth` *Data-driven Nonparametric Regression for the Trend in Equidistant Time Series*

Description

This function runs an iterative plug-in algorithm to find the optimal bandwidth for the estimation of the nonparametric trend in equidistant time series (with short memory errors) and then employs the resulting bandwidth via either local polynomial or kernel regression.

Usage

```
msmooth(
  y,
  p = c(1, 3),
  mu = c(0, 1, 2, 3),
  bStart = 0.15,
  alg = c("A", "B", "N", "NA", "NAM", "NM", "O", "OA", "OAM", "OM"),
  method = c("lpr", "kr")
)
```

Arguments

`y`	a numeric vector that contains the input time series ordered from past to present.
`p`	an integer 1 (local linear regression) or 3 (local cubic regression); represents the order of polynomial within the local polynomial regression (see also the 'Details' section); is set to *1* by default; is automatically set to *1* if *method = "kr"*.
`mu`	an integer 0, ..., 3 that represents the smoothness parameter of the kernel weighting function and thus defines the kernel function that will be used within the local polynomial regression; is set to *1* by default.

Number	**Kernel**
0	Uniform Kernel
1	Epanechnikov Kernel
2	Bisquare Kernel
3	Triweight Kernel

`bStart`	a numeric object that indicates the starting value of the bandwidth for the iterative process; should be 0 < bStart < 0.5; is set to *0.15* by default.
`alg`	a control parameter (as character) that indicates the corresponding algorithm used (set to *A* by default).

Algorithm	**Description**
A	Nonparametric estimation of the variance factor with an enlarged bandwidth, optimal inflation rate
B	Nonparametric estimation of the variance factor with an enlarged bandwidth, naive inflation rate
O	Nonparametric estimation of the variance factor, optimal inflation rate
N	Nonparametric estimation of the variance factor, naive inflation rate
OAM	Estimation of the variance factor with ARMA(p,q)-models, optimal inflation rate
NAM	Estimation of the variance factor with ARMA(p,q)-models, naive inflation rate
OA	Estimation of the variance factor with AR(p)-models, optimal inflation rate
NA	Estimation of the variance factor with AR(p)-models, naive inflation rate
OM	Estimation of the variance factor with MA(q)-models, optimal inflation rate
NM	Estimation of the variance factor with MA(q)-models, naive inflation rate

It is proposed to use *alg = "A"* only in combination with *p = 1*. If the user finds that the chosen bandwidth by algortihm *A* is too small, *alg = "B"* with preferably *p = 3* is suggested. For more information on the components of the

	different algorithms, please consult `tsmooth`.
`method`	the smoothing approach; *"lpr"* represents the local polynomial regression, whereas *"kr"* implements a kernel regression; is set to *"lpr"* by default.

Details

The trend or its derivatives are estimated based on the additive nonparametric regression model for a time series

y_[t] = m(x_[t]) + eps_[t],

where y_[t] is the observed time series, x_[t] is the rescaled time on [0, 1], m(x_[t]) is a smooth trend function and eps_[t] are stationary errors with E(eps_[t]) = 0 (see also Beran and Feng, 2002). With this function m(x_[t]) can be estimated without a parametric model assumption for the error series. Thus, after estimating and removing the trend, any suitable parametric model, e.g. an ARMA(p, q) model, can be fitted to the residuals.

The iterative-plug-in (IPI) algorithm, which numerically minimizes the Asymptotic Mean Squared Error (AMISE), was proposed by Feng, Gries and Fritz (2019).

Define I[m^(k)] = int_[c_[b]]^[d_[b]] [m^(k)(x)]^2 dx,

beta_[v,k] = int_[-1]^[1] u^k K(u)du and

R(K) = int_[-1]^[1] K^2(u)du.

The AMISE is then

AMISE(h) = h^(2(k-v)) * (I[m^(k)]beta^2 / [k]^2) + (2pi * c_[f](d_[b]-c_[b])R(K) / nh^(2v+1))

with h being the bandwidth, k = p + 1 being the order of kernel, v being the order of derivative, 0 <= c_[b] < d_[b] <= 1, n being the number of observations, c_[f] being the variance factor and K_[(v,k)](u) being the k-th order equivalent kernel obtained for the estimation of m^[(v)] in the interior. m^[(v)] is the v-th order derivative (v = 0, 1, 2, ...) of the nonparametric trend.

The function calculates suitable estimates for c_[f], the variance factor, and I[m^(k)] over different iterations. In each iteration, a bandwidth is obtained in accordance with the AMISE that once more serves as an input for the following iteration. The process repeats until either convergence or the 40th iteration is reached. For further details on the asymptotic theory or the algorithm, please consult Feng, Gries and Fritz (2019) or Feng et al. (2019).

To apply the function, only few arguments are needed: a data input *y*, an order of polynomial *p*, a kernel function defined by the smoothness parameter *mu*, a starting value for the relative bandwidth *bStart* and a final smoothing method *method*. In fact, aside from the input vector *y*, every argument has a default setting that can be adjusted for the individual case. It is recommended to initially use the default values for *p*, *alg* and *bStart* and adjust them in the rare case of the resulting optimal bandwidth being either too small or too large. Theoretically, the initial bandwidth does not affect the selected optimal bandwidth. However, in practice local minima of the AMISE might exist and influence the selected bandwidth. Therefore, the default setting is *bStart = 0.15*, which is a compromise between the starting values *bStart = 0.1* for *p = 1* and *bStart = 0.2* for *p = 3* that were proposed by Feng, Gries and Fritz (2019). In the rare case of a clearly unsuitable optimal bandwidth, a starting bandwidth that differs from the default value is a first possible approach to obtain a better result. Other argument adjustments can be tried as well. For more specific information on the input arguments consult the section *Arguments*.

When applying the function, an optimal bandwidth is obtained based on the IPI algorithm proposed by Feng, Gries and Fritz (2019). In a second step, the nonparametric trend of the series is calulated with respect to the chosen bandwidth and the selected regression method (*lpf* or *kr*). It is notable that *p* is automatically set to 1 for *method = "kr"*. The output object is then a list that contains, among other components, the original time series, the estimated trend values and the series without the trend.

The default print method for this function delivers key numbers such as the iteration steps and the generated optimal bandwidth rounded to the fourth decimal. The exact numbers and results such as the estimated nonparametric trend series are saved within the output object and can be addressed via the *$* sign.

For more information on the use of this function

Value

The function returns a list with different components:

AR.BIC the Bayesian Information Criterion of the optimal AR(p) model when estimating the variance factor via autoregressive models (if calculated; calculated for *alg = "OA"* and *alg = "NA"*).

ARMA.BIC the Bayesian Information Criterion of the optimal ARMA(p,q) model when estimating the variance factor via autoregressive-moving-average models (if calculated; calculated for *alg = "OAM"* and *alg = "NAM"*).

cb the percentage of omitted observations on each side of the observation period; always equal to 0.05.

b0 the optimal bandwidth chosen by the IPI-algorithm.

bb the boundary bandwidth method used within the IPI; always equal to 1.

bStart the starting value of the (relative) bandwidth; input argument.

bvc indicates whether an enlarged bandwidth was used for the variance factor estimation or not; depends on the chosen algorithm.

cf0 the estimated variance factor.

cf0.AR the estimated variance factor obtained by estimation of autoregressive models (if calculated; *alg = "OA"* or *"NA"*).

cf0.ARMA the estimated variance factor obtained by estimation of autoregressive-moving-average models (if calculated; calculated for *alg = "OAM"* and *alg = "NAM"*).

cf0.LW the estimated variance factor obtained by Lag-Window Spectral Density Estimation following Bühlmann (1996) (if calculated; calculated for algorithms *A*, *B*, *O* and *N*).

cf0.MA the estimated variance factor obtained by estimation of moving-average models (if calculated; calculated for *alg = "OM"* and *alg = "NM"*).

I2 the estimated value of I[m(k)].

InfR the setting for the inflation rate according to the chosen algorithm.

iterations the bandwidths of the single iterations steps

L0.opt the optimal bandwidth for the lag-window spectral density estimation (if calculated; calculated for algorithms *A*, *B*, *O* and *N*).

MA.BIC the Bayesian Information Criterion of the optimal MA(q) model when estimating the variance factor via moving-average models (if calculated; calculated for *alg = "OM"* and *alg = "NM"*).

Mcf the estimation method for the variance factor estimation; depends on the chosen algorithm.

mu the smoothness parameter of the second order kernel; input argument.

n the number of observations.

niterations the total number of iterations until convergence.

orig the original input series; input argument.

p.BIC the order p of the optimal AR(p) or ARMA(p,q) model when estimating the variance factor via autoregressive or autoregressive-moving average models (if calculated; calculated for *alg = "OA"*, *alg = "NA"*, *alg = "OAM"* and *alg = "NAM"*).

p the order of polynomial used in the IPI-algorithm; also used for the final smoothing, if *method = "lpr"*; input argument.

q.BIC the order q of the optimal MA(q) or ARMA(p,q) model when estimating the variance factor via moving-average or autoregressive-moving average models (if calculated; calculated for *alg = "OM"*, *alg = "NM"*, *alg = "OAM"* and *alg = "NAM"*).

res the estimated residual series.

ye the nonparametric estimates of the trend.

ws the weighting systems used within the local polynomial regression; only exists, if the final smoothing is done via a local polynomial regression.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: https://wiwi.uni-paderborn.de/en/dep4/feng/
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Beran, J. and Feng, Y. (2002). Local polynomial fitting with long-memory, short-memory and antipersistent errors. Annals of the Institute of Statistical Mathematics, 54(2), 291-311.

Bühlmann, P. (1996). Locally adaptive lag-window spectral estimation. Journal of Time Series Analysis, 17(3), 247-270.

Feng, Y., Gries, T. and Fritz, M. (2019). Data-driven local polynomial for the trend and its derivatives in economic time series. Discussion Paper. Paderborn University. (Not yet published)

Feng, Y., Gries, T., Letmathe, S. and Schulz, D. (2019). The smoots package in R for semiparametric modeling of trend stationary time series. Discussion Paper. Paderborn University. (Not yet published)

Examples

```
### Example 1: US-GDP ###

# Logarithm of test data
# -> the logarithm of the data is assumed to follow the additive model
test_data <- gdpUS
```

```
y <- log(test_data$GDP)

# Applied msmooth function for the trend
results <- msmooth(y, p = 1, mu = 1, bStart = 0.1, alg = "A", method = "lpr")
res <- results$res
ye <- results$ye

# Plot of the results
t <- seq(from = 1947, to = 2019.25, by = 0.25)
matplot(t, cbind(y, ye), type = "ll", lty = c(3, 1), col = c(1, 2),
       xlab = "Years", ylab = "Log-Quartlery US-GDP",
       main = "Log-Quarterly US-GDP vs. Trend, Q1 1947 - Q2 2019")
legend("bottomright", legend = c("Original series", "Estimated trend"),
      fill = c(1, 2), cex = 0.7)
results


### Example 2: German Stock Index ###

# Obtain log-transformation of the returns
returns <- diff(log(dax$Close))
rt <- returns - mean(returns)

# Apply 'smoots' function to the log-data
# -> the log-transformation is assumed to follow the additive model
yt <- log(rt^2)

# Algorithm A delivers better results for local cubic regression in this
# case than the recommended algorithm B.
est <- msmooth(yt, p = 3, alg = "A")
m_xt <- est$ye

# Obtain the standardized returns 'eps' and the scale function 's'
sqrtC_s <- exp(m_xt / 2)
eps_sqrtC <- rt / sqrtC_s
C <- 1 / var(eps_sqrtC)
eps <- eps_sqrtC * sqrt(C)
s <- sqrtC_s / sqrt(C)

# -> 'eps' can now be analyzed by any suitable GARCH-type model.
#    The total volatilities are then the product of the conditional
#    volatilities obtained from 'eps' and the scale function 's'.
```

---

plot.smoots *Plot Method for the Package 'smoots'*

Description

This function regulates how objects created by the package smoots are plotted.

Usage

```
## S3 method for class 'smoots'
plot(x, type = "l", main = NULL, xlab = NULL, ylab = NULL, ...)
```

Arguments

`x`	an input object of class *smoots*.
`type`	the usual 'type' argument of the plot function; is set to "l" per default.
`main`	the usual 'main' argument of the plot function; is set to NULL per default and then uses a predefined title that depends on the chosen smoots plot.
`xlab`	the usual 'xlab' argument of the plot function; is set to NULL per default and then uses a predefined label for the x-axis that depends on the chosen smoots plot.
`ylab`	the usual 'ylab' argument of the plot function; is set to NULL per default and then uses a predefined label for the y-axis that depends on the chosen smoots plot.
`...`	additional arguments of the standard plot method

Value

None

Author(s)

- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University), Package Creator and Maintainer

---

`print.smoots` *Print Method for the Package 'smoots'*

Description

This function regulates how objects created by the package smoots are printed.

Usage

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

Arguments

`x`	an input object of class *smoots*.
`...`	additional arguments of the standard print method

Value

None

Author(s)

- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University), Package Creator and Maintainer

---

`smoots` *smoots: A package for data-driven nonparametric estimation of the trend and its derivatives in equidistant time series.*

Description

The *smoots* package provides different applicable functions for the estimation of the trend or its derivatives in equidistant time series. The main functions include an automated bandwidth selection method for time series with short-memory errors.

Functions

The smoots functions are either meant for calculating nonparametric estimates of the trend of a time series or its derivatives.

msmooth is the central function of the package. It allows the user to conduct a local polynomial regression of the trend based on an optimal bandwidth that is obtained by an iterative plug-in algorithm. There are also different algorithms implemented concerning the inflation rate and other factors that can be chosen from (see also: `msmooth`).

dsmooth is a function that calculates the derivatives of the trend after obtaining the optimal bandwidth by an itertive plug-in algorithm (see also: `dsmooth`).

tsmooth is similar to *msmooth* as it also calculates the trend of the series. Instead of using the name of a predefined algorithm that settles the inflation rate (and other factors), these factors can be manually and individually adjusted as arguments in the function (see also: `tsmooth`).

gsmooth is a standard smoothing function that applies the local polynomial regression method. A bandwidth can be chosen freely (see also: `gsmooth`).

knsmooth is a standard smoothing function that applies the kernel regression method. A bandwidth can be chosen freely (see also: `knsmooth`).

Datasets

The package includes four datasets: *gdpUS* (see also: `gdpUS`) that has data concerning the quarterly US GDP between Q1 1947-01 and Q2 2019, *tempNH* (see also: `tempNH`) with mean monthly Northern Hemisphere temperature changes from 1880 to 2018, *dax* (see also: `dax`) with daily financial time series data of the German stock index (DAX) from 1990 to July 2019 and *vix* (see also: `vix`) with daily financial time series data of the CBOE Volatility Index (VIX) from 1990 to July 2019.

License

The package is distributed under the General Public License v3 ([GPL-3](https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3))).

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: `https://wiwi.uni-paderborn.de/en/dep4/feng/`
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Feng, Y., Gries, T. and Fritz, M. (2019). Data-driven local polynomial for the trend and its derivatives in economic time series. Discussion Paper. Paderborn University. (Not yet published)

Feng, Y., Gries, T., Letmathe, S. and Schulz, D. (2019). The smoots package in R for semiparametric modeling of trend stationary time series. Discussion Paper. Paderborn University. (Not yet published)

`tempNH` *Mean Monthly Northern Hemisphere Temperature Changes*

Description

A dataset that contains seasonally adjusted mean monthly Northern Hemisphere temperature changes from 1880 to 2018.

Usage

```
tempNH
```

Format

A data frame with 1668 rows and 3 variables:

Year the observation year

Month the observation month

Change the observed mean monthly temperature changes in degrees Celsius

Source

The data was obtained from the Goddard Institute for Space Studies (part of the National Aeronautics and Space Administration [NASA]).

`https://data.giss.nasa.gov/gistemp/tabledata_v4/NH.Ts+dSST.txt`

tsmooth — *Advanced Data-driven Nonparametric Regression for the Trend in Equidistant Time Series*

Description

This function runs an iterative plug-in algorithm to find the optimal bandwidth for the estimation of the nonparametric trend in equidistant time series (with short-memory errors) and then employs the resulting bandwidth via either local polynomial or kernel regression. This function allows for more flexibility in its arguments than *msmooth*.

Usage

```
tsmooth(
  y,
  p = c(1, 3),
  mu = c(0, 1, 2, 3),
  Mcf = c("NP", "ARMA", "AR", "MA"),
  InfR = c("Opt", "Nai", "Var"),
  bStart = 0.15,
  bvc = c("Y", "N"),
  bb = c(0, 1),
  cb = 0.05,
  method = c("lpr", "kr")
)
```

Arguments

`y`	a numeric vector that contains the time series ordered from past to present.
`p`	an integer 1 (local linear regression) or 3 (local cubic regression); represents the order of polynomial within the local polynomial regression (see also the 'Details' section); is set to *1* by default; is automatically set to *1* if *method = "kr"*.
`mu`	an integer 0, ..., 3 that represents the smoothness parameter of the kernel weighting function and thus defines the kernel function that will be used within the local polynomial regression; is set to *1* by default.

Number	**Kernel**
0	Uniform Kernel
1	Epanechnikov Kernel
2	Bisquare Kernel
3	Triweight Kernel

`Mcf`	method for estimating the variance factor c_[f] by the IPI (see also the 'Details' section); is set to *NP* by default.

Method	**Explanation**
NP	Nonparametric estimation using the Bartlett window
ARMA	Estimation on the assumption that the residuals follow an ARMA model
AR	Estimation on the assumption that the residuals follow an AR model
MA	Estimation on the assumption that the residuals follow an MA model

`InfR`	a character object that represents the inflation rate in the form h_[d] = h^[a] for the bandwidth in the estimation of I[m^(k)] (see also the 'Details' section); is set to *Opt* by default.

Inflation rate	**Description**
Opt	Optimal inflation rate a_[p,O] (5/7 for p = 1; 9/11 for p = 3)
Nai	Naive inflation rate a_[p,N] (5/9 for p = 1; 9/13 for p = 3)
Var	Stable inflation rate a_[p,S] (1/2 for p = 1 and p = 3)

`bStart`	a numeric object that indicates the starting value of the bandwidth for the iterative process; should be 0 < bStart < 0.5; is set to *0.15* by default.
`bvc`	a character object that indicates whether an enlarged bandwidth is being used for the estimation of the variance factor c_[f] (see also the 'Details' section) or not; is set to *Y* by default.

Bandwidth	**Description**
Y	Using an enlarged bandwidth
N	Using a bandwidth without enlargement

`bb`	can be set to *0* or *1*; the parameter controlling the bandwidth used at the boundary; is set to *1* by default.

Number (bb)	**Estimation procedure at boundary points**
0	Fixed bandwidth on one side with possible large bandwidth on the other side at the boundary
1	The k-nearest neighbor method will be used

`cb`	a numeric value that indicates the percentage of omitted observations on each side of the observation period for the automated bandwidth selection; is set to *0.05* by default.
`method`	the final smoothing approach; *"lpr"* represents the local polynomial regression, whereas *"kr"* implements a kernel regression; is set to *"lpr"* by default.

Details

A nonparametric regression of the trend in a time series is based on the additive model

y_[t] = m(x_[t]) + eps_[t],

where y_[t] is the observed time series in question, x_[t] is the rescaled time on [0, 1], m(x_[t]) is the nonparametric trend and eps_[t] are the errors with E(eps_[t]) = 0 (see also Beran and Feng, 2002). With this function m(x_[t]) can be estimated without a parametric model assumption for the error series. Thus, after estimating and removing the trend, any suitable parametric model, e.g. an ARMA(p, q) model, can be fitted to the residuals.

The iterative-plug-in (IPI) algorithm, which numerically minimizes the Asymptotic Mean Squared Error (AMISE), was proposed by Feng, Gries and Fritz (2019).

Define I[m^(k)] = int_[c_[b]]^[d_[b]] [m^(k)(x)]^2 dx,

beta_[v,k] = int_[-1]^[1] u^k K(u)du and

R(K) = int_[-1]^[1] K^2(u)du.

The AMISE is then

AMISE(h) = h^(2(k-v)) * (I[m^(k)]beta^2 / [k]^2) + (2pi * c_[f](d_[b]-c_[b])R(K) / nh^(2v+1))

with h being the bandwidth, k = p + 1 being the order of the equivalent kernel, v being the order of derivative, 0 <= c_[b] < d_[b] <= 1, n being the number of observations, c_[f] being the variance factor and K_[(v,k)](u) being the k-th order equivalent kernel obtained for the estimation of m^[(v)] in the interior. m^[(v)] is the v-th order derivative (v = 0, 1, 2, ...) of the nonparametric trend.

The function calculates suitable estimates for c_[f], the variance factor, and I[m^(k)] over different iterations. In each iteration, a bandwidth is obtained in accordance with the AMISE that once more serves as an input for the following iteration. The process repeats until either convergence or the 40th iteration is reached. For further details on the asymptotic theory or the algorithm, please consult Feng, Gries and Fritz (2019) or Feng et al. (2019).

To apply the function, more arguments are needed compared to the similar function *msmooth*: a data input *y*, an order of polynomial *p*, a kernel weighting function defined by the smoothness parameter *mu*, a variance factor estimation method *Mcf*, an inflation rate setting *InfR* (see also Beran and Feng, 2002), a starting value for the relative bandwidth *bStart*, an inflation setting for the variance factor estimation *bvc*, a boundary method *bb*, a boundary cut-off percentage *cb* and a final smoothing method *method*. In fact, aside from the input vector *y*, every argument has a default setting that can be adjusted for the individual case. Theoretically, the initial bandwidth does not affect the selected optimal bandwidth. However, in practice local minima of the AMISE might exist and influence the selected bandwidth. Therefore, the default setting is *bStart = 0.15*, which is a compromise between the starting values *bStart = 0.1* for *p = 1* and *bStart = 0.2* for *p = 3* that were proposed by Feng, Gries and Fritz (2019). In the rare case of a clearly unsuitable optimal bandwidth, a starting bandwidth that differs from the default value is a first possible approach to obtain a better result. Other argument adjustments can be tried as well. For more specific information on the input arguments consult the section *Arguments*.

When applying the function, an optimal bandwidth is obtained based on the IPI algorithm proposed by Feng, Gries and Fritz (2019). In a second step, the nonparametric trend of the series is calulated with respect to the chosen bandwidth and the selected regression method (*lpf* or *kr*). Please note that *method = "lpf"* is strongly recommended by the authors. Moreover, it is notable that *p* is automatically set to 1 for *method = "kr"*. The output object is then a list that contains, among other components, the original time series, the estimated trend values and the series without the trend.

The default print method for this function delivers only key numbers such as the iteration steps and the generated optimal bandwidth rounded to the fourth decimal. The exact numbers and results such as the estimated nonparametric trend series are saved within the output object and can be addressed via the *$* sign.

Value

The function returns a list with different components:

AR.BIC the Bayesian Information Criterion of the optimal AR(p) model when estimating the variance factor via autoregressive models (if calculated; calculated for *Mcf = "AR"*).

ARMA.BIC the Bayesian Information Criterion of the optimal ARMA(p,q) model when estimating the variance factor via autoregressive-moving-average models (if calculated; calculated for *Mcf = "ARMA"*).

cb the percentage of omitted observations on each side of the observation period; input argument.

b0 the optimal bandwidth chosen by the IPI-algorithm.

bb the boundary bandwidth method used within the IPI; input argument.

bStart the starting value of the (relative) bandwidth; input argument.

bvc indicates whether an enlarged bandwidth was used for the variance factor estimation or not; input argument.

cf0 the estimated variance factor.

cf0.AR the estimated variance factor obtained by estimation of autoregressive models (if calculated; calculated for *Mcf = "AR"*).

cf0.ARMA the estimated variance factor obtained by estimation of autoregressive-moving-average models (if calculated; calculated for *Mcf = "ARMA"*).

cf0.LW the estimated variance factor obtained by Lag-Window Spectral Density Estimation following Bühlmann (1996) (if calculated; calculated for *Mcf = "NP"*).

cf0.MA the estimated variance factor obtained by estimation of moving-average models (if calculated; calculated for *Mcf = "MA"*).

I2 the estimated value of I[m(k)].

InfR the setting for the inflation rate; input argument.

iterations the bandwidths of the single iterations steps.

L0.opt the optimal bandwidth for the lag-window spectral density estimation (if calculated).

MA.BIC the Bayesian Information Criterion of the optimal MA(q) model when estimating the variance factor via moving-average models (if calculated; calculated for *Mcf = "MA"*).

Mcf the estimation method for the variance factor estimation; input argument.

mu the smoothness parameter of the second order kernel; input argument.

n the number of observations.

niterations the total number of iterations until convergence.

orig the original input series; input argument.

p.BIC the order p of the optimal AR(p) or ARMA(p,q) model when estimating the variance factor via autoregressive or autoregressive-moving average models (if calculated; calculated for *Mcf = "AR"* and *Mcf = "ARMA"*).

p the order of polynomial used in the IPI-algorithm; also used for the final smoothing, if *method = "lpr"*; input argument.

q.BIC the order q of the optimal MA(q) or ARMA(p,q) model when estimating the variance factor via moving-average or autoregressive-moving average models (if calculated; calculated for *Mcf = "MA"* and *Mcf = "ARMA"*).

res the estimated residual series.

ws the weighting systems used within the local polynomial regression; only exists, if the final smoothing is done via a local polynomial regression.

ye the nonparametric estimates of the trend.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms
  Website: https://wiwi.uni-paderborn.de/en/dep4/feng/
- Dominik Schulz (Student Assistant) (Department of Economics, Paderborn University),
  Package Creator and Maintainer

References

Beran, J. and Feng, Y. (2002). Local polynomial fitting with long-memory, short-memory and antipersistent errors. Annals of the Institute of Statistical Mathematics, 54(2), 291-311.

Bühlmann, P. (1996). Locally adaptive lag-window spectral estimation. Journal of Time Series Analysis, 17(3), 247-270.

Feng, Y., Gries, T. and Fritz, M. (2019). Data-driven local polynomial for the trend and its derivatives in economic time series. Discussion Paper. Paderborn University. (Not yet pubslished)

Feng, Y., Gries, T., Letmathe, S. and Schulz, D. (2019). The smoots package in R for semiparametric modeling of trend stationary time series. Discussion Paper. Paderborn University. (Not yet published)

Examples

```
### Example 1: US-GDP ###

# Logarithm of test data
test_data <- gdpUS
y <- log(test_data$GDP)

# Applied tsmooth function for the trend
result <- tsmooth(y, p = 1, mu = 1, Mcf = "NP", InfR = "Opt",
                     bStart = 0.1, bvc = "Y")
trend1 <- result$ye

# Plot of the results
t <- seq(from = 1947, to = 2019.25, by = 0.25)
plot(t, y, type = "l", xlab = "Year", ylab = "log(US-GDP)", bty = "n",
    lwd = 1, lty = 3,
    main = "Estimated trend for log-quarterly US-GDP, Q1 1947 - Q2 2019")
points(t, trend1, type = "l", col = 2, lwd = 1)
title(sub = expression(italic("Figure 1")), col.sub = "gray47",
     cex.sub = 0.6, adj = 0)
result


### Example 2: German Stock Index ###

# Obtain log-transformation of the returns
returns <- diff(log(dax$Close))
rt <- returns - mean(returns)

# Apply 'smoots' function to the log-data
# -> the log-transformation is assumed to follow the additive model
yt <- log(rt^2)

# In this case, the optimal inflation rate is used for p = 3.
est <- tsmooth(yt, p = 3, InfR = "Opt")
m_xt <- est$ye

# Obtain the standardized returns 'eps' and the scale function 's'
sqrtC_s <- exp(m_xt / 2)
eps_sqrtC <- rt / sqrtC_s
C <- 1 / var(eps_sqrtC)
eps <- eps_sqrtC * sqrt(C)
s <- sqrtC_s / sqrt(C)

# -> 'eps' can now be analyzed by any suitable GARCH-type model.
#    The total volatilities are then the product of the conditional
#    volatilities obtained from 'eps' and the scale function 's'.
```

`vix` *CBOE Volatility Index (VIX) Financial Time Series Data*

Description

A dataset that contains the daily financial data of the VIX from 1990 to July 2019 (currency in USD).

Usage

```
vix
```

Format

A data frame with 7452 rows and 9 variables:

Year the observation year

Month the observation month

Day the observation day

Open the opening price of the day

High the highest price of the day

Low the lowest price of the day

Close the closing price of the day

AdjClose the adjusted closing price of the day

Volume the traded volume

Source

The data was obtained from Yahoo Finance.

https://query1.finance.yahoo.com/v7/finance/download/^VIX?period1=631148400&period2=1564524000&interval=1d&events=history&crumb=Iaq1EPZAQRb

Index