Package 'esemifar'

November 3, 2026

Type Package

Title Smoothing Long-Memory Time Series

Version 2.0.2

Description The nonparametric trend and its derivatives in equidistant time series (TS) with long-memory 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.
The smoothing methods of the package are described in Letmathe, S., Beran, J. and Feng, Y., (2023) <doi:10.1080/03610926.2023.2276049>.

License GPL-3

Encoding UTF-8

LazyData true

Imports fracdiff, stats, utils, smoots, graphics, grDevices, Rcpp, future, furrr, ggplot2

Depends R (>= 2.10)

LinkingTo Rcpp, RcppArmadillo

Acknowledgments This work was supported by the German DFG project GZ-FE-1500-2-1.

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Yuanhua Feng [aut] (Paderborn University, Germany), Jan Beran [aut] (University of Konstanz, Germany), Sebastian Letmathe [aut] (formerly Paderborn University, Germany), Dominik Schulz [aut, cre] (formerly Paderborn University, Germany)

Maintainer Dominik Schulz <dominik.schulz.r@gmail.com>

Repository CRAN

Date/Publication 2026-11-03 17:40:02 UTC

Contents

esemifar-package . . . 2
airLDN . . . 3
arma_to_ar . . . 4
arma_to_ma . . . 5
critMatlm . . . 6
dsmoothlm . . . 8
d_to_coef . . . 11
esemifar . . . 12
farima_to_ar . . . 13
farima_to_ma . . . 14
fitted.esemifar . . . 15
gdpG7 . . . 16
gsmooth . . . 16
plot.esemifar . . . 19
plot.esemifar_fc . . . 20
predict.esemifar . . . 21
print.esemifar . . . 23
residuals.esemifar . . . 24
tsmoothlm . . . 24

Index **29**

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

Description

The `esemifar` 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 long-memory errors.

Functions (version 1.0.0)

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

`dsmoothlm` is a function that calculates the derivatives of the trend after obtaining the optimal bandwidth by an iterative plug-in algorithm.

`tsmoothlm` 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. Inflation rate (and other factors) can be manually and individually adjusted as arguments in the function (see also: `tsmoothlm`).

`critMatlm` is a quick tool for the calculation of information criteria for FARIMA(p, d, q) models with different order combinations p and q. The function returns a matrix with the obtained values of the selected criterion for the different combinations of p and q (see also: `critMatlm`).

Datasets

The package includes two datasets: `airLDN` (see also: `airLDN`) with daily observations of individual air pollutants from 2014 to 2020 and `gdpG7` (see also: `gdpG7`) that has data concerning the quarterly G7 GDP between Q1 1962 and Q4 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
- Sebastian Letmathe,
  Author (and Original Package Creator and Maintainer)
- Dominik Schulz,
  Author and Current Package Creator and Maintainer

References

Beran, J. and Y. Feng (2002a). Iterative plug-in algorithms for SEMIFAR models - definition, convergence, and asymptotic properties. Journal of Computational and Graphical Statistics 11(3), 690-713.

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

Beran, J. and Feng, Y. (2002c). SEMIFAR models - a semiparametric approach to modelling trends, longrange dependence and nonstationarity. Computational Statistics & Data Analysis 40(2), 393-419.

Letmathe, S., Beran, J. and Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods: 1-13.

`airLDN` *Daily Observations of the Air Quality Index of London (Britain)*

Description

A dataset that contains daily observations of individual air pollutants from 2014 to 2020.

Usage

```
airLDN
```

Format

A data frame with 2552 rows and 8 variables:

date the observation date

pm25 particle matter with an aerodynamic diameter smaller than 2.5 μm

pm10 particle matter with an aerodynamic diameter smaller than 10 μm

o3 ozone or trioxygen

no2 nitrogen dioxide

so2 sulphur dioxide

co carbon monoxide

AQI composite air quality index

Source

The data can be obtained from the World Air Quality Project.

https://aqicn.org/city/london

---

arma_to_ar *AR Representation of an ARMA Model*

Description

Output has representation with positive signs (on the right-hand side of the equation); inputs are both with positive signs (on right-hand side of equation).

Usage

```
arma_to_ar(ar = numeric(0), ma = numeric(0), max_i = 1000)
```

Arguments

ar	the AR-coefficient series ordered by lag.
ma	the MA-coefficient series ordered by lag.
max_i	the maximum index up until which to return the coefficient series.

Details

Consider the ARMA model

$$X_t = ar_1 X_{t-1} + ... + ar_p X_{t-p} + ma_1 e_{t-1} + ... + ma_q e_{t-q} + e_t,$$

where e_t are the innovations. ar_i, $i = 1, ..., p$, are the AR-coefficients to pass to the argument ar, ma_j, $j = 1, ..., q$, are the MA-coefficients to pass to the argument ma.The function then returns the coefficients from the corresponding infinite-order AR-representation

$$-e_t = c_0 X_t + c_1 X_{t-1} + c_2 X_{t-2} + c_3 X_{t-3} + ...,$$

where c_l, $l = 0, 1, 2, ...$, are the coefficients. Following this notation, $c_0 = -1$ by definition.

Value

A numeric vector is returned.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
arma_to_ar(ar = 0.75, ma = 0.5, max_i = 100)
```

`arma_to_ma` *MA Representation of an ARMA Model*

Description

Output has representation with positive signs (on the right-hand side of the equation); inputs are both with positive signs (on right-hand side of equation).

Usage

```
arma_to_ma(ar = numeric(0), ma = numeric(0), max_i = 1000)
```

Arguments

`ar`	the AR-coefficient series ordered by lag.
`ma`	the MA-coefficient series ordered by lag.
`max_i`	the maximum index up until which to return the coefficient series.

Details

Consider the ARMA model

$$X_t = ar_1 X_{t-1} + ... + ar_p X_{t-p} + ma_1 e_{t-1} + ... + ma_q e_{t-q} + e_t,$$

where e_t are the innovations. ar_i, $i = 1, ..., p$, are the AR-coefficients to pass to the argument `ar`, ma_j, $j = 1, ..., q$, are the MA-coefficients to pass to the argument `ma`.The function then returns the coefficients from the corresponding infinite-order MA-representation

$$X_t = c_0 e_t + c_1 e_{t-1} + c_2 e_{t-2} + c_3 e_{t-3} + ...,$$

where c_l, $l = 0, 1, 2, ...$, are the coefficients. Following this notation, $c_0 = 1$ by definition.

Value

A numeric vector is returned.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
arma_to_ma(ar = 0.75, ma = 0.5, max_i = 100)
```

`critMatlm` *FARIMA Order Selection Matrix*

Description

An information criterion is calculated for different orders of a fractionally integrated autoregressive-moving-average (FARIMA) model.

Usage

```
critMatlm(X, p.max = 5, q.max = 5, criterion = c("bic", "aic"))
```

Arguments

`X`	a numeric vector that contains the observed time series ordered from past to present; the series is assumed to follow an FARIMA process.
`p.max`	an integer value $>= 0$ that defines the maximum autoregressive order to calculate the criterion for; is set to `5` by default; decimal numbers will be rounded off to integers.
`q.max`	an integer value $>= 0$ that defines the maximum moving-average order to to calculate the criterion for; is set to `5` by default; decimal numbers will be rounded off to integers.
`criterion`	a character value that defines the information criterion that will be calculated; the Bayesian Information Criterion (`"bic"`) and Akaike Information Criterion (`"aic"`) are the supported choices; is set to `"bic"` by default.

Details

The series passed to `X` is assumed to follow an FARIMA(p, d, q) model. A `p.max + 1` by `q.max + 1` matrix is calculated for this series. More precisely, the criterion chosen via the argument `criterion` is calculated for all combinations of orders $p = 0, 1, ..., p_{max}$ and $q = 0, 1, ..., q_{max}$.

Within the function, two information criteria are supported: the Bayesian Information Criterion (BIC) and Akaike's Information Criterion (AIC). The AIC is given by

$$AIC_{p,q} := \ln(\hat{\sigma}^2_{p,q}) + \frac{2(p+q)}{n},$$

where $\hat{\sigma}^2_{p,q}$ is the estimated innovation variance, p and q are the ARMA orders and n is the number of observations.

The BIC, on the other hand, is defined by

$$BIC_{p,q} := k\ln(n) - 2\ln(\hat{L})$$

with k being the number of estimated parameters and $\hat{L}$ being the estimated Log-Likelihood. Since the parameter k only differs with respect to the orders p and q for all estimated models, the term $k\ln(n)$ is reduced to $(p+q)\ln(n)$ within the function. Exemplary, if the mean of the series is estimated as well, it is usually considered within the parameter k when calculating the BIC. However, since the mean is estimated for all models, not considering this estimated parameter within the calculation of the BIC will reduce all BIC values by the same amount of $\ln(n)$. Therefore, the selection via this simplified criterion is still valid, if the number of the estimated parameters only differs with respect to p and q between the models that the BIC is obtained for.

The optimal orders are considered to be the ones which minimize either the BIC or the AIC. The use of the BIC is however recommended, because the BIC is consistent, whereas the AIC is not.

NOTE:

Within this function, the `fracdiff` function of the `fracdiff` package is used throughout for the estimation of FARIMA models.

Value

The function returns a `p.max + 1` by `q.max + 1` matrix, where the rows represent the AR orders from $p = 0$ to $p = p_{max}$ and the columns represent the MA orders from $q = 0$ to $q = q_{max}$. The values within the matrix are the values of the previously selected information criterion for the different combinations of p and q.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
# Simulate a FARIMA(2, 0.3 ,1) process
set.seed(1)
X.sim <- fracdiff::fracdiff.sim(n = 1000, ar = c(1.2, -0.71), ma = -0.46,
                                d = 0.3)$series
# Application of the function with BIC-criterion
BIC_mat <- critMatlm(X.sim)
BIC_mat
# determining the optimal order
smoots::optOrd(BIC_mat)
```

`dsmoothlm` *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 long-memory errors) and subsequently employs the obtained bandwidth via local polynomial regression.

Usage

```
dsmoothlm(
  y,
  d = c(1, 2),
  pmin = c(0, 1, 2, 3, 4, 5),
  pmax = c(0, 1, 2, 3, 4, 5),
  qmin = c(0, 1, 2, 3, 4, 5),
  qmax = c(0, 1, 2, 3, 4, 5),
  mu = c(0, 1, 2, 3),
  mu.p = c(0, 1, 2, 3),
  pp = c(1, 3),
  bStart.p = 0.15,
  InfR.p = c("Opt", "Nai", "Var")
)
```

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; the default is `d = 1`.
`pmin`	an integer value $>= 0$ that defines the minimum autoregressive order to calculate the BIC-criterion for; is set to `0` by default; decimal numbers will be rounded off to integers.
`pmax`	an integer value $>= 0$ that defines the maximum autoregressive order to calculate the BIC-criterion for; is set to `0` by default; decimal numbers will be rounded off to integers.
`qmin`	an integer value $>= 0$ that defines the minimum moving-average order to calculate the BIC-criterion for; is set to `0` by default; decimal numbers will be rounded off to integers.
`qmax`	an integer value $>= 0$ that defines the maximum moving-average order to calculate the BIC-criterion for; is set to `0` by default; decimal numbers will be rounded off to integers.
`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.

`mu.p` an integer `0`, ..., `3` that represents the smoothness parameter of the kernel weighting function for the iterative process to obtain initial estimates for c_f, d and b_0; 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, d and b_0 will be calculated by `tsmoothlm`; the default is `pp = 1`.

`bStart.p` a numeric object that indicates the starting value of the bandwidth for the iterative process to obtain initial estimates for c_f, d and b_0; should be > 0; is set to `0.15` by default.

`InfR.p` a character object that represents the inflation rate in the form $h_d = h^a$ of the bandwidth for the iterative process to obtain initial estimates for c_f, d and b_0; is set to `"Opt"` by default.

Details

The trend is estimated based on the additive nonparametric regression model for an equidistant time series

$$y_t = m(x_t) + \epsilon_t,$$

where y_t is the observed time series, x_t is the rescaled time on the interval $[0, 1]$, $m(x_t)$ is a smooth and deterministic trend function and ϵ_t are stationary errors with $E(\epsilon_t) = 0$ and is assumed to follow a FARIMA(p, d, q) model (see also Beran and Feng, 2002).

The iterative-plug-in (IPI) algorithm, which numerically minimizes the Asymptotic Mean Squared Error (AMISE), is based on the proposal of Beran and Feng (2002a).

The variance factor c_f, the long memory parameter d and the starting bandwidth $b0$ are first obtained from a pilot-estimation of the time series' nonparametric trend ($\nu = 0$) with polynomial order p_p. The estimate is then plugged into the iterative procedure for estimating the first or second derivative ($\nu = 1$ or $\nu = 2$). For further details on the asymptotic theory or the algorithm, we refer the user to Letmathe, Beran and Feng (2023).

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 and a kernel function defined by the smoothness parameter `mu`, an optimal bandwidth is numerically calculated for the trend's 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. The initial bandwidth does not affect the resulting optimal bandwidth in theory. However in practice, local minima of the AMISE can influence the results. For more specific information on the input arguments consult the section *Arguments*.

After the bandwidth estimation, the nonparametric derivative of the series is calculated 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.p the starting bandwidth for the nonparametric trend estimation that leads to the initial estimates; input argument.

cf0 the estimated variance factor.

InfR.p the inflation rate setting.

iterations the bandwidths of the single iterations steps

mu.p 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 pilot estimation; input argument.

v the considered order of the trend's derivative; input argument d.

ws the weighting system matrix used within the local polynomial regression; this matrix is a condensed version of a complete weighting system matrix; in each row of ws, the weights for conducting the smoothing procedure at a specific observation time point can be found; the first $[nb + 0.5]$ rows, where n corresponds to the number of observations, b is the bandwidth considered for smoothing and $[.]$ denotes the integer part, contain the weights at the $[nb + 0.5]$ left-hand boundary points; the weights in row $[nb + 0.5] + 1$ are representative for the estimation at all interior points and the remaining rows contain the weights for the right-hand boundary points; each row has exactly $2[nb + 0.5] + 1$ elements, more specifically the weights for observations of the nearest $2[nb + 0.5] + 1$ time points; moreover, the weights are normalized, i.e. the weights are obtained under consideration of the time points $x_t = t/n$, where $t = 1, 2, ..., n$.

ye the nonparametric estimates of the derivative.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms

- Sebastian Letmathe (formerly Department of Economics, Paderborn University), Package Creator and Maintainer
- Dominik Schulz (formerly Department of Economics, Paderborn University), Author

References

Letmathe, S., Beran, J. and Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods: 1-13.

Examples

```
# Logarithm of test data
test_data <- gdpG7
y <- log(test_data$gdp)
n <- length(y)
t <- seq(from = 1962, to = 2020, length.out = n)

# Applied dsmooth function for the trend's first derivative
result_d <- dsmoothlm(y, d = 1, pp = 1, pmax = 1, qmax = 1, InfR.p = "Opt")
estim <- result_d$ye

# Plot of the results
plot(t, estim, xlab = "Year", ylab = "First derivative", type = "l",
 main = paste0("Estimated first derivative of the trend for log-quarterly ",
 "G7-GDP, Q1 1962 - Q4 2019"), cex.axis = 0.8, cex.main = 0.8,
 cex.lab = 0.8, bty = "n")

# Print result
result_d

# The main function "dsmoothlm"----------------------------------------
```

---

d_to_coef *Filter Coefficients of the Fractional Differencing Operator*

Description

Output is with positive signs on the left-hand side of the equation.

Usage

```
d_to_coef(d, max_i = 1000)
```

Arguments

d	the fractional differencing coefficient.
max_i	the maximum index up until which to return the coefficient series.

Details

Consider the FARIMA model

$$(1-B)^d Y_t = ar_1 X_{t-1} + ... + ar_p X_{t-p} + ma_1 e_{t-1} + ... + ma_q e_{t-q} + e_t,$$

where e_t are the innovations and where $X_t = (1-B)^d Y_t$. d is the fractional differencing coefficient.

The fractional differencing operator $(1-B)^d$ can alternatively be expressed as an infinite coefficient series, so that

$$(1-B)^d = \sum_{l=0}^{\infty} b_l B^k,$$

where B is the backshift operator and where b_l, $l = 0, 1, 2, ...$, are the coefficients. Note that $b_0 = 1$ by definition.

The function returns the series of coefficients $\{b_l, l = 0, 1, 2, ...\}$.

Value

A numeric vector is returned.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
d_to_coef(d = 0.3, max_i = 100)
```

`esemifar` *Create an ESEMIFAR Model Object Suitable for Forecasting*

Description

An alternative method to create an ESEMIFAR estimation object stitched together from a nonparametric and a parametric part.

Usage

```
esemifar(nonpar_model, par_model)
```

Arguments

`nonpar_model`	an estimation object returned by `gsmooth`.
`par_model`	an estimation object returned by `fracdiff` fitted to the residuals of `nonpar_model`.

Details

The main function `tsmoothlm` already returns a fully estimated ESEMIFAR model. In some instances, alternative specifications of the nonparametric and parametric model parts are needed, for which `tsmoothlm` with its automated estimation algorithm does not provide sufficient flexibility. Therefore, this function allows to stitch together a nonparametric model part returned by `gsmooth` and a FARIMA part for the residuals obtained via `fracdiff`. The resulting object can then be used for forecasting.

Value

The function returns a list of class `"esemifar"` with elements `nonpar_model` and `par_model`.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
lgdp <- log(esemifar::gdpG7$gdp)
nonpar <- gsmooth(lgdp, b = 0.15)
res <- nonpar$res
par <- fracdiff::fracdiff(res, nar = 1, nma = 1)
model <- esemifar(nonpar_model = nonpar, par_model = par)
model
```

`farima_to_ar` *AR Representation of a FARIMA Model*

Description

Output has representation with positive signs (on the right-hand side of the equation); inputs are both with positive signs (on right-hand side of equation).

Usage

```
farima_to_ar(ar = numeric(0), ma = numeric(0), d = 0, max_i = 1000)
```

Arguments

`ar`	the AR-coefficient series ordered by lag.
`ma`	the MA-coefficient series ordered by lag.
`d`	the fractional differencing coefficient.
`max_i`	the maximum index up until which to return the coefficient series.

Details

Consider the FARIMA model

$$(1-B)^d Y_t = ar_1 X_{t-1} + ... + ar_p X_{t-p} + ma_1 e_{t-1} + ... + ma_q e_{t-q} + e_t,$$

where e_t are the innovations and where $X_t = (1-B)^d Y_t$. ar_i, $i = 1, ..., p$, are the AR-coefficients to pass to the argument `ar`, ma_j, $j = 1, ..., q$, are the MA-coefficients to pass to the argument `ma`. d is the fractional differencing coefficient. The function then returns the coefficients from the corresponding infinite-order AR-representation

$$-e_t = c_0 Y_t + c_1 Y_{t-1} + c_2 Y_{t-2} + c_3 Y_{t-3} + ...,$$

where c_l, $l = 0, 1, 2, ...$, are the coefficients. Following this notation, $c_0 = -1$ by definition.

Value

A numeric vector is returned.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
farima_to_ar(ar = 0.75, ma = 0.5, d = 0.3, max_i = 100)
```

`farima_to_ma` *MA Representation of a FARIMA Model*

Description

Output has representation with positive signs (on the right-hand side of the equation); inputs are both with positive signs (on right-hand side of equation).

Usage

```
farima_to_ma(ar = numeric(0), ma = numeric(0), d = 0, max_i = 1000)
```

Arguments

`ar`	the AR-coefficient series ordered by lag.
`ma`	the MA-coefficient series ordered by lag.
`d`	the fractional differencing coefficient.
`max_i`	the maximum index up until which to return the coefficient series.

Details

Consider the FARIMA model

$$(1-B)^d Y_t = ar_1 X_{t-1} + ... + ar_p X_{t-p} + ma_1 e_{t-1} + ... + ma_q e_{t-q} + e_t,$$

where e_t are the innovations and where $X_t = (1-B)^d Y_t$. ar_i, $i = 1, ..., p$, are the AR-coefficients to pass to the argument `ar`, ma_j, $j = 1, ..., q$, are the MA-coefficients to pass to the argument `ma`. d is the fractional differencing coefficient. The function then returns the coefficients from the corresponding infinite-order AR-representation

$$Y_t = c_0 e_t + c_1 e_{t-1} + c_2 e_{t-2} + c_3 e_{t-3} + ...,$$

where c_l, $l = 0, 1, 2, ...$, are the coefficients. Following this notation, $c_0 = 1$ by definition.

Value

A numeric vector is returned.

Author(s)

- Dominik Schulz (Scientific Employee) (Department of Economics, Paderborn University), Author

Examples

```
farima_to_ma(ar = 0.75, ma = 0.5, d = 0.3, max_i = 100)
```

`fitted.esemifar` *Extract Model Fitted Values*

Description

Generic function which extracts fitted values from a `esemifar` class object. Both `fitted` and `fitted.values` can be called.

Usage

```
## S3 method for class 'esemifar'
fitted(object, ...)
```

Arguments

`object`	an object from the `esemifar` class.
`...`	included for consistency with the generic function.

Value

Fitted values extracted from a `esemifar` class object.

Author(s)

- Sebastian Letmathe (Scientific Employee) (Department of Economics, Paderborn University),

---

gdpG7 *Quarterly G7 GDP, Q1 1962 to Q4 2019*

Description

A dataset that contains the (seasonally adjusted) Gross Domestic Product of the G7 nations from the first quarter of 1962 to the fourth quarter of 2019

Usage

```
gdpG7
```

Format

A data frame with 232 rows and 3 variables:

year the observation year

quarter the observation quarter in the given year

gdp the volume Index of the Gross Domestic Product of the G7 nations

Source

The data was obtained from the Organization for Economic Co-operation and Development (OECD)

https://data.oecd.org/gdp/quarterly-gdp.htm#indicator-chart

---

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 = v + 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; is set to `v = 0` by default.

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; `p - v` must be an odd number; is set to `v + 1` by default.

Exemplary 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 `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

Details

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

$$y_t = m(x_t) + \epsilon_t,$$

where y_t is the observed time series, x_t is the rescaled time on the interval $[0, 1]$, $m(x_t)$ is a smooth and deterministic trend function and ϵ_t are stationary errors with $E(\epsilon_t) = 0$ (see also Beran and Feng, 2002).

This function is part of the package `smoots` and is used in the field of analyzing 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 non-parametrically, even though the result will strongly depend on the bandwidth given beforehand as an input.

NOTE:

The estimates are obtained with regard to the rescaled time points on the interval $[0, 1]$. Thus, if $\nu > 0$, the estimates might not reflect the values for the actual time points. To rescale the estimates, we refer the user to the `rescale` function of the `smoots` package.

With package version 1.1.0, this function implements C++ code by means of the Rcpp and `RcppArmadillo` packages for better performance.

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; is set to `NULL` for `v > 0`.

v the order of derivative; input argument.

ws the weighting system matrix used within the local polynomial regression; this matrix is a condensed version of a complete weighting system matrix; in each row of `ws`, the weights for conducting the smoothing procedure at a specific observation time point can be found; the first $[nb + 0.5]$ rows, where n corresponds to the number of observations, b is the bandwidth considered for smoothing and $[.]$ denotes the integer part, contain the weights at the $[nb + 0.5]$ left-hand boundary points; the weights in row $[nb + 0.5] + 1$ are representative for the estimation at all interior points and the remaining rows contain the weights for the right-hand boundary points; each row has exactly $2[nb + 0.5] + 1$ elements, more specifically the weights for observations of the nearest $2[nb + 0.5] + 1$ time points; moreover, the weights are normalized, i.e. the weights are obtained under consideration of the time points $x_t = t/n$, where $t = 1, 2, ..., n$.

ye a vector with the estimates of the selected nonparametric order of derivative on the rescaled time interval $[0, 1]$.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms

- Dominik Schulz (formerly 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. (2020). Data-driven local polynomial for the trend and its derivatives in economic time series. Journal of Nonparametric Statistics, 32:2, 510-533.

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. Unpublished.

Examples

```
# Logarithm of test data
y <- log(esemifar::gdpG7$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 = 1962, to = 2019.75, 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 = "red", lwd = 1)
points(t, trend2, type = "l", col = "blue", lwd = 1)
legend("bottomright", legend = c("Trend (b = 0.28)", "Trend (b = 0.11)"),
 fill = c("red", "blue"), cex = 0.6)
title(sub = expression(italic("Figure 1")), col.sub = "gray47",
 cex.sub = 0.6, adj = 0)
```

---

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

Description

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

Usage

```
## S3 method for class 'esemifar'
plot(x, t = NULL, rescale = TRUE, which = NULL, ...)
```

Arguments

x	an input object of class esemifar.
t	an optional vector with time points that will be considered for the x-axis within the plot; is set to NULL by default and uses a vector 1:length(x$ye) for time points.
rescale	a single logical value; is set to TRUE by default; if the output of a derivative estimation process is passed to x and if rescale = TRUE, the estimates will be rescaled according to t.
which	one of 1 (observations), 2 (trend), 3 (residuals) or 4 (observations with trend); the default NULL starts an interactive selection via the console.
...	additional arguments of the standard plot method.

Value

None

Author(s)

- Dominik Schulz (formerly Department of Economics, Paderborn University),

---

plot.esemifar_fc *Plot Method for Class "esemifar_fc"*

Description

Create basic R plots for forecasting objects of class "esemifar_fc".

Usage

```
## S3 method for class 'esemifar_fc'
plot(x, y = NULL, t = NULL, ...)
```

Arguments

x	an object of class "esemifar_fc", for example generated by a call to predict.esemifar.
y	currently without use; for compatibility only.
t	a numeric vector with series of time points for observations and forecasts.
...	further arguments of plot.ts to adjust for example the axis limits via xlim and ylim.

Details

This is a plot method to visualize the forecasting results for an ESEMIFAR model. Common plot arguments can be implemented to change the appearance.

Value

This method returns NULL.

Author(s)

- Dominik Schulz (formerly Department of Economics, Paderborn University), Author and Package Creator

Examples

```
lgdp <- log(esemifar::gdpG7$gdp)
est <- tsmoothlm(lgdp, pmax = 1, qmax = 1)
# Under normality
fc <- predict(est, n.ahead = 10, method = "norm", expo = TRUE)
plot(fc)
```

---

predict.esemifar *ESEMIFAR Prediction Method*

Description

Point and interval forecasts (under the normality assumption or via a bootstrap) for fitted ESEMIFAR models.

Usage

```
## S3 method for class 'esemifar'
predict(
  object,
  n.ahead = 5,
  alpha = c(0.95, 0.99),
  method = c("norm", "boot"),
  bootMethod = c("simple", "advanced"),
  npaths = 5000,
  quant.type = 8,
  boot_progress = TRUE,
  expo = FALSE,
  trend_extrap = c("linear", "constant"),
  future = TRUE,
  num_cores = max(future::availableCores() - 1, 1),
  biasadj = TRUE,
  ...
)
```

Arguments

object	an object returned by either tsmoothlm or esemifar.
n.ahead	a single numeric value that represents the forecasting horizon.
alpha	a numeric vector with confidence levels for the forecasting intervals; the default c(0.95, 0.99) represents 95-percent and 99-percent forecasting interval bounds that will be computed.
method	whether to obtain the forecasting intervals under the normality assumption ("norm") or via a bootstrap ("boot").
bootMethod	only for method = "boot": whether to simulate future paths only ("simple") or whether to re-estimate the FARIMA model for the re-sampled series and to then obtain simulated predictive roots ("advanced").
npaths	only for method = "boot": the number of bootstrap iterations.
quant.type	only for method = "boot": the quantile type as in the argument type of the function quantile.
boot_progress	only for method = "boot": whether to show a progress bar in the console.
expo	whether to exponentiate all results at the end.
trend_extrap	how to extrapolate the estimated trend into the future: linearly ("linear") or constantly ("constant").
future	only for method = "boot": use parallel programming for the bootstrap via the future framework?
num_cores	only for method = "boot" and future = TRUE: how many cores to use in the parallel programming.
biasadj	a logical value indicating whether or not to do a bias adjustment on the exponentiated point forecasts; the default is expo = TRUE.
...	no purpose; for compatibility only.

Details

Produce point and interval forecasts based on ESEMIFAR models. Throughout, the infinite-order AR-representation of the parametric FARIMA part is considered to produce point forecasts and future paths of the series. The trend is usually extrapolated linearly (or constantly as an alternative).

Value

The function returns a list of class "esemifar" with elements nonpar_model and par_model.

A list with various elements is returned.

obs the observed series.

mean the point forecasts.

lower the lower bounds of the forecasting intervals.

upper the upper bounds of the forecasting intervals.

model the fitted ESEMIFAR model object.

level the confidence levels for the forecasting intervals.

Author(s)

- Dominik Schulz (formerly Department of Economics, Paderborn University), Author

Examples

```
lgdp <- log(esemifar::gdpG7$gdp)
est <- tsmoothlm(lgdp, pmax = 1, qmax = 1)
# Under normality
fc <- predict(est, n.ahead = 10, method = "norm", expo = TRUE)
fc$mean
fc$lower
fc$upper
```

---

print.esemifar *Print Method for the Package 'esemifar'*

Description

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

Usage

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

Arguments

`x`	an input object of class `esemifar`.
`...`	included for compatibility; additional arguments will however not affect the output.

Value

None

Author(s)

- Dominik Schulz (formerly Department of Economics, Paderborn University),

`residuals.esemifar` *Extract Model Residuals*

Description

Generic function which extracts model residuals from a `esemifar` class object. Both `residuals` and its abbreviated form `resid` can be called.

Usage

```
## S3 method for class 'esemifar'
residuals(object, ...)
```

Arguments

`object`	an object from the `esemifar` class.
`...`	included for consistency with the generic function.

Value

Residuals extracted from a `esemifar` class object.

Author(s)

- Sebastian Letmathe (formerly Department of Economics, Paderborn University),

---

`tsmoothlm` *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 long-memory errors) and then employs the resulting bandwidth via either local polynomial or kernel regression.

Usage

```
tsmoothlm(
  y,
  pmin = c(0, 1, 2, 3, 4, 5),
  pmax = c(0, 1, 2, 3, 4, 5),
  qmin = c(0, 1, 2, 3, 4, 5),
  qmax = c(0, 1, 2, 3, 4, 5),
  p = c(1, 3),
```

```
  mu = c(0, 1, 2, 3),
  InfR = c("Opt", "Nai", "Var"),
  bStart = 0.15,
  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.
pmin	an integer value $>= 0$ that defines the minimum autoregressive order to calculate the BIC-criterion for; is set to 0 by default; decimal numbers will be rounded off to integers.
pmax	an integer value $>= 0$ that defines the maximum autoregressive order to calculate the BIC-criterion for; is set to 0 by default; decimal numbers will be rounded off to integers.
qmin	an integer value $>= 0$ that defines the minimum moving-average order to calculate the BIC-criterion for; is set to 0 by default; decimal numbers will be rounded off to integers.
qmax	an integer value $>= 0$ that defines the maximum moving-average order to calculate the BIC-criterion for; is set to 0 by default; decimal numbers will be rounded off to integers.
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

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 $a_{p,O}$ ($(5-2d)/(7-2d)$ for p = 1; $(9-2d)/(11-2d)$ for p = 3)
"Nai"	Naive inflation rate $a_{p,N}$ ($(5-2d)/(9-2d)$ for p = 1; $(9-2d)/(13-2d)$ 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; 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 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

The trend is estimated based on the additive nonparametric regression model for an equidistant time series

$$y_t = m(x_t) + \epsilon_t,$$

where y_t is the observed time series, x_t is the rescaled time on the interval $[0, 1]$, $m(x_t)$ is a smooth and deterministic trend function and ϵ_t are stationary errors with $E(\epsilon_t) = 0$ and is assumed to follow a FARIMA(p, d, q) model (see also Beran and Feng, 2002a, Beran and Feng, 2002b and Beran and Feng, 2002c).

The iterative-plug-in (IPI) algorithm, which numerically minimizes the Asymptotic Mean Squared Error (AMISE), is based on the proposal of Beran and Feng (2002a).

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 see Letmathe et al., 2023.

To apply the function, the following arguments are needed: a data input y, an order of polynomial p, a kernel weighting function defined by the smoothness parameter mu, an inflation rate setting InfR (see also Beran and Feng, 2002b), a starting value for the relative bandwidth bStart, 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. 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 a strongly modified version of the IPI algorithm of Beran and Feng (2002a). In a second step, the nonparametric trend of the series is calculated 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:

FARIMA.BIC the Bayesian Information Criterion of the optimal FARIMA(p, d, q) model.

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

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.

cf0 the estimated variance factor; in contrast to the definitions given in the *Details* section, this object actually contains an estimated value of $2\pi c_f$, i.e. it corresponds to the estimated sum of autocovariances.

d.BIC the long-memory parameter of the optimal FARIMA(p, d, q) model.

FARMA.BIC the model fit of the selected FARIMA(p, d, q model.

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

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 FARIMA(p, d, q) model.

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 FARIMA(p, d, q) model.

res the estimated residual series.

v the considered order of derivative of the trend; is always zero for this function.

ws the weighting system matrix used within the local polynomial regression; this matrix is a condensed version of a complete weighting system matrix; in each row of `ws`, the weights for conducting the smoothing procedure at a specific observation time point can be found; the first $[nb + 0.5]$ rows, where n corresponds to the number of observations, b is the bandwidth considered for smoothing and $[.]$ denotes the integer part, contain the weights at the $[nb + 0.5]$ left-hand boundary points; the weights in row $[nb + 0.5] + 1$ are representative for the estimation at all interior points and the remaining rows contain the weights for the right-hand boundary points; each row has exactly $2[nb + 0.5] + 1$ elements, more specifically the weights for observations of the nearest $2[nb + 0.5] + 1$ time points; moreover, the weights are normalized, i.e. the weights are obtained under consideration of the time points $x_t = t/n$, where $t = 1, 2, ..., n$.

ye the nonparametric estimates of the trend.

Author(s)

- Yuanhua Feng (Department of Economics, Paderborn University),
  Author of the Algorithms

- Sebastian Letmathe (formerly Department of Economics, Paderborn University),
  Package Creator and Maintainer
- Dominik Schulz (formerly Department of Economics, Paderborn University),
  Author

References

Beran, J. and Y. Feng (2002a). Iterative plug-in algorithms for SEMIFAR models - definition, convergence, and asymptotic properties. Journal of Computational and Graphical Statistics 11(3), 690-713.

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

Beran, J. and Feng, Y. (2002c). SEMIFAR models - a semiparametric approach to modelling trends, longrange dependence and nonstationarity. Computational Statistics & Data Analysis 40(2), 393-419.

Letmathe, S., Beran, J. and Feng, Y. (2023). An extended exponential SEMIFAR model with application in R. Communications in Statistics - Theory and Methods: 1-13.

Examples

```
### Example 1: G7-GDP ###

# Logarithm of test data
# -> the logarithm of the data is assumed to follow the additive model
test_data <- gdpG7
y <- log(test_data$gdp)
n <- length(y)

# Applied tsmooth function for the trend
result <- tsmoothlm(y, p = 1, pmax = 1, qmax = 1, InfR = "Opt")
trend1 <- result$ye

# Plot of the results
t <- seq(from = 1962, to = 2020, length.out = n)
plot(t, y, type = "l", xlab = "Year", ylab = "log(G7-GDP)", bty = "n",
 lwd = 1, lty = 3,
 main = "Estimated trend for log-quarterly G7-GDP, Q1 1962 - Q4 2019")
points(t, trend1, type = "l", col = "red", lwd = 1)
title(sub = expression(italic("Figure 1")), col.sub = "gray47",
 cex.sub = 0.6, adj = 0)
result
```

Index

∗ **datasets**
airLDN, 3
gdpG7, 16

airLDN, *3*, 3
arma_to_ar, 4
arma_to_ma, 5

critMatlm, *2*, 6

d_to_coef, 11
dsmoothlm, 8

esemifar, 12, *22*
esemifar-package, 2

farima_to_ar, 13
farima_to_ma, 14
fitted.esemifar, 15
fracdiff, *7*, *12, 13*

gdpG7, *3*, 16
gsmooth, *12, 13*, 16

plot.esemifar, 19
plot.esemifar_fc, 20
plot.ts, *20*
predict.esemifar, *20*, 21
print.esemifar, 23

quantile, *22*

Rcpp, *18*
residuals.esemifar, 24

tsmoothlm, *2*, *9*, *13*, *22*, 24