Diagnosing the trend and bootstrapping the forecasting intervals using a semiparametric ARMA

Yuanhua Feng, Marlon Fritz, Thomas Gries
Sebastian Letmathe and Dominik Schulz
Faculty of Business Administration and Economics, Paderborn University

June 12, 2020

Abstract

Development of theory for existing and new graphical tools based on the data-driven Semi-ARMA model and introducing them into a Version 1.1.0 of 'smoots'. (Feng/Gries)

Keywords: Asymptotics, stationarity test, linearity test, bootstrapping interval forecasting, *smoots* package, R, Semi-Log-GARCH

JEL Codes: C14, C51

1 Introduction

Schulz, Letmathe, Feng, Gries

2 The Semi-ARMA and its estimation procedure

2.1 The common Semi-ARMA model

(Definition, estimation)

Define the time index as $t = 1, 2 \ldots, n$. Consider an additive time series model that consists of a nonparametric trend function $m(x_t)$, where $x_t = t/n$ is the rescaled time on the interval $[0, 1]$, and a zero mean stationary rest ξ_t. Let y_t be the observed time series. The model can now be written as

$$y_t = m(x_t) + \xi_t. \tag{1}$$

A Semi-ARMA model is defined by (1) and the additional assumption that the rest term ξ_t follows an autoregressive-moving-average (ARMA) model with autoregressive and moving-average orders p and q, respectively, i.e.,

$$\beta(B)\xi_t = \alpha(B)\epsilon_t, \tag{2}$$

where ϵ_t is an independentally and identically distributed (*i.i.d.*) White Noise process with zero mean and constant variance σ_ϵ^2, i.e., $\epsilon_t \sim \text{IID}(0, \sigma_\epsilon^2)$. Moreover, $\beta(B) = 1 - \sum_{i=1}^{p} B^i \beta_i$ and $\alpha(B) = \sum_{j=0}^{q} B^j \alpha_j$ with α_j as well as β_i being real numbered coefficients and B being the backshift operator so that $B^k X_t = X_{t-k}$ for $k = 0, 1, 2, \ldots$.

A usual procedure for estimation consists of two seperate estimation steps. At first, the values of the nonparametric trend function are estimated. Commonly used approaches are kernel regression or the more favorable local polynomial regression (references!!!). Secondly, the detrended series can be further analyzed by well-established Maximum Likelihood methods (references!!!) in order to obtain a parametric model for the residual series.

Different variants of the initial decomposition of the observed time series can be implemented as well. Another common approach is the multiplicative decomposition of time series. In this case, let y_t^* be the observed time series and $m^*(x_t)$ be the nonparametric trend function and ξ_t^* the multiplicative error in y_t^*, respectively. Then

$$y_t^* = m^*(x_t) \times \xi_t^* \tag{3}$$

states the multiplicative component model. If we define $y_t = \ln(y_t^*)$, $m(x_t) = \ln[m^*(x_t)]$ and $\xi_t = \ln(\xi_t^*)$, the logarithm of the observed series can be modeled according to an additive component model (1).

2.2 Practical implementation using 'smoots'

Basic idea of the algorithms, different options, ... !!! Noch auf genaue Angabe aller aufgeführten Variablen etc. achtgeben!

The main smoothing method within 'smoots' is local polynomial regression, because the boundary problem in nonparametric smoothing approaches is partially solved. The local polynomial estimator of the ν-th derivative of m, $m^{(\nu)}$, is regarded within this paper. Let m be at least $(p+1)$-times differentiable at a point x_0. Define $\vec{y} = (y_1, \ldots, y_n)^T$. If $\nu \leq p$, solving the locally weighted least squares problem (references!!!)

$$Q = \sum_{t=1}^{n} \left\{ y_t - \sum_{j=0}^{p} \beta_j (x_t - x)^j \right\}^2 W\left(\frac{x_t - x}{h}\right) \Rightarrow \min \tag{4}$$

yields the local polynomial estimator of $m^{(\nu)}$ at a point x with h being the bandwidth, W being a second order kernel function on $[-1, 1]$ and $\hat{\vec{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$. Following this idea, $\hat{m}^{(\nu)}(x) = \nu! \hat{\beta}_\nu$ is an estimator of $m^{(\nu)}(x)$.

To quantify the quality of the local polynomial estimator, the mean integrated squared error (MISE)

$$\text{MISE}(h) = \int_{c_b}^{d_b} [\hat{m}^{(\nu)}(x) - m^{(\nu)}(x)]^2 dx \tag{5}$$

is commonly evaluated, where $0 < c_b < d_b < 1$ are boundary cutoffs to reduce the effect of the boundary within the bandwidth selection.

Let $K_{(\nu,k)}(u)$ be the k-th order equivalent kernel that is realized for the estimation of $m^{(\nu)}$ at an interior point. The asymptotic MISE (AMISE) of the local polynomial estimator is then given by

$$\text{AMISE}(h) = h^{2(k-\nu)} \frac{I[m^{(k)}]\beta^2}{[k!]^2} + \frac{2\pi c_f (d_b - c_b) R(K)}{n h^{2\nu+1}}, \tag{6}$$

where c_f is the variance factor, $k = p+1$, $I[m^{(k)}] = \int_{c_b}^{d_b} [m^{(k)}(x)]^2 dx$, $\beta_{(\nu,k)} = \int_{-1}^{1} u^k K(u) du$ and $R(K) = \int K^2(u) du$. The asymptotically optimal bandwidth is then defined as

$$h_A = C_A n^{-1/(2k+1)} \tag{7}$$

with

$$C_A = \left[\frac{2\nu+1}{2(k-\nu)} \frac{2\pi c_f [k!]^2 (d_b - c_b) R(K)}{I[m^{(k)}] \beta^2_{(\nu,k)}} \right]^{1/(2k+1)}. \tag{8}$$

Within the 'smoots' package, a plug-in algorithm for automated bandwidth selection is implemented. Since only c_f and $I[m^{(k)}]$ are unknown in (8), a selected bandwidth can be acquired by inserting suitable estimates of these two quantities. As was stated by (Feng, Gries, Fritz....Anpassen!!!), the existence of error correlation does not have an impact on $\hat{I}[m^{(k)}]$. Therefore, approaches usually applied to independent error models will also yield a suitable estimate of $I[m^{(k)}]$ in the case of dependent errors. For the estimation of c_f, however, a nonparametric lag-window estimator is applied following the idea of Bühlmann (1996).

2.2.1 Nonparametric lag-window estimation of c_f

In order to estimate the unknown quantity c_f, a slightly adjusted version of the nonparametric lag-window estimation introduced by Bühlmann (1996) is implemented. This adjustment of the algorithm was proposed by Feng, Gries, and Fritz (2020). Let M be the lag-window-width and $\hat{\gamma}(l)$ the sample autocovariances that were obtained from the residuals of a nonparametric trend estimation with the bandwidth h_V. c_f can now be estimated via

$$\hat{c}_{f,M} = \frac{1}{2\pi} \sum_{l=-M}^{M} w_l \hat{\gamma}(l) \tag{9}$$

with $w_l = l/(M + 0.5)$ being Bartlett-window weights. The IPI-algorithm is then defined by the following steps.

1. Define $M_0 = [n/2]$, where [.] resembles the integer part, as the initial lag-window-width.
2. Following the idea of Bühlmann (1996), the global step of the IPI-algorithm results in a global lag-window-width $\hat{M}_G$. As opposed to Bühlmann (1996), however, an inflation factor of $n^{-2/21}$ is recommended by Feng, Gries, and Fritz (2020).
3. At the local step, $M' = [\hat{M}_G/n^{2/21}]$ is used for $\lambda = 0$. Insert the needed estimates into equation (5) in Bühlmann (1996) to receive a final lag-window-width $\hat{M}$. $\hat{c}_{f,\hat{M}}$ can now be obtained in accordance to (9).

2.2.2 Nonparametric estimation of m

The IPI-algorithm for the estimation of m is briefly described in this subsection. Denote by CF the bandwidth correction factors given in Table 1 in Feng and Heiler (2009). Furthermore, let p_d be equal to $k + 1$.

1. Select an inital bandwidth h_0. The IPI-algorithm will begin using this bandwidth.
2. For $j = 1, 2, \ldots,$ apply a local polynomial regression for $m^{(0)}(x)$ with the bandwidth CFh_{j-1} and obtain $\hat{c}_f$ using the residuals of the previous estimation and the lag-window estimator (9).
3. Define $h_{d,j} = h_{j-1}^{\alpha}$ with α being a suitable inflation factor. Obtain an estimate of $I[m^{(k)}]$ under consideration of $h_{d,j}$ and a local polyomial of order p_d. Insert $\hat{c}_f$ and $\hat{I}[m^{(k)}]$ as well as all necessary quantities into (7) and define it as h_j.
4. Increase j by 1 and repeat the previous steps 2. and 3 until either convergence or a given number of iterations is reached. Finally, set $\hat{h}_{\text{opt}}$ equal to h_j of the last iteration that was executed.

A more detailed explanation of the underlying algorithms, especially the algorithm for the data-driven estimation of $m^{(1)}(x)$ and $m^{(2)}(x)$, which was omitted at this point, can be found in Feng, Gries, and Fritz (2020).

2.2.3 Application via 'smoots'

The *smoots* package (smoothing time series, version 1.1.0, Feng, Letmathe, and Schulz, 2020) in R (R Core Team, 2020) consists of thirteen directly applicable functions, four S3

methods and four datasets. The package allows for the simple estimation of nonparametric trend functions or their derivatives in trend-stationary time series with short-memory errors. With package version 1.1.0 further highly practical functions were implemented for testing the linearity of the trend function and for forecasting. Moreover, a selection of the built-in functions now uses compiled C++ code that is invoked by the *Rcpp* (Eddelbuettel and François, 2011) and *RcppArmadillo* (Eddelbuettel and Sanderson, 2014) packages for less computation time of the algorithms. *Rcpp* works as an API (application programming interface) in order to call C++ from R, whereas *RcppArmadillo* provides C++ functions of the *Armadillo* library (Sanderson and Curtin, 2016). In the following, a brief overview of the available functions of *smoots* 1.1.0 and their usage will be given.

The core of the *smoots* package consists of the smoothing functions with automatically selected bandwidth *msmooth()*, *tsmooth()* and *dsmooth()*. In trend-stationary time series with short-memory errors, *msmooth()* and *tsmooth()* obtain estimates of the nonparametric trend function $m(x_t)$, whereas with *dsmooth()* the trend function's derivatives can be calculated. *msmooth()* is in fact a wrapper-function for *tsmooth()* that comes with predefined algorithms with different settings for *tsmooth()*. Therefore, the latter is a more sophisticated trend estimation function for experienced users, while the former function is more user-friendly. With the release of *smoots* 1.1.0, the main part of the non-data-driven function *gsmooth()* has been rewritten in C++ and implemented via *Rcpp* (Eddelbuettel and François, 2011). Consequently, the data-driven functions, which repeatedly call *gsmooth()*, have a noticeably improved performance. More details on the three main data-driven functions and their arguments can be found in Feng et al. (2020) or in the package's manual.

Version 1.1.0 of the *smoots* comes with many additional functions. Firstly, derivative estimates obtained by *smoots* that are always valid for the time points scaled to the closed interval [0, 1] can now be easily rescaled to any actual time points by means of the function *rescale()*. Any numeric matrix or vector that needs to be rescaled can be passed to the function's first argument y. With the second argument x, however, the series of actual equidistant time points can be specified. The function then rescales the object passed to y in accordance to x. Secondly, another new function in *smoots* is *critMatrix()*. For a given time series X_t, which is assumed to follow an ARMA(p, q) model, an information criterion described by the argument *criterion* is calculated for different combinations of possible

values for p and q. Available criteria are the Bayesian Information Criterion (*criterion = "bic"*) and Akaike's Information Criterion (*criterion = "aic"*). Further specifying the arguments *p.max* as well as *q.max* allows the adjustement of the maximum orders of p and q to calculate the criterion for. All possible combinations of $p = 0, 1, \ldots, p.max$ and $q = 0, 1, \ldots, q.max$ will be considered. The last argument of the function *include.mean* is a logical value. With this argument the user can decide whether to estimate the mean of the series (*include.mean = TRUE*) or not (*include.mean = FALSE*). The latter option is mainly useful, if the mean of the series is to be excluded from the ARMA model. The third function *orderSelect()* is a quick order selection tool that can be combined with *critMatrix()*. When a numeric matrix is passed to its argument *mat*, the functions selects a value according to the function given to its argument *FUN* and then returns a vector with both the corresponding row and column number reduced by one of the selected value. Further selection restrictions can be implemented as an expression via the function argument *restr*.

The linearity of the (nonparametric) trend function in an additive component model can now be tested graphically with the newly added function *confBounds()*. By passing an object of class *smoots* that was generated by either *msmooth()*, *tsmooth()* or *dsmooth()* to the argument *obj*, an (approximately) unbiased estimate of the trend or its derivatives alongside the corresponding confidence bounds is obtained. For *plot = TRUE*, a graphic of these series is created. For the trend, different parametric regression lines can be selected to be shown alongside the trend and its confidence bounds by the argument p, where 0, 1, 2, 3 are valid options that define the order of the parametric model. This behavior can be suppressed by setting *showPar = FALSE*. For the first derivative, the constant line representing the slope of a theoretical linear representation of the trend can be displayed or omitted, whereas a constant line at value zero can be shown in the plot of the second derivative. These parametric representations, i.e. the linear trend line and the constant values, can be compared graphically with the respective confidence bounds. Moreover, the argument *alpha* represents the confidence level. A 95% confidence level is the default setting with *alpha = 0.95*. By means of ... additional arguments of the standard *plot()* function in R can be implemented, e.g. by defining a vector x, time points different from the standard setting *1:n* can be considered for the x axis of the plots, the limits of the axes can be adjusted by *xlim* and *ylim*, the color of the estimate series can be adjusted by *col* and many other adjustments can be made. The last argument in *confBounds*

that needs to be mentioned is *rescale*, which is only meaningful, if a plot of the first or second derivative is created. For *rescale = FALSE* the estimates for the time scaled on the interval [0, 1] are forced to be plotted against the default plot time points *1:n* or the values passed to *x*. If however *rescale = TRUE* is selected, the estimates are rescaled in accordance to the x axis.

Aside from the previously mentioned functions, the newly introduced forecasting functions are an integral part of the new package version of *smoots*. There are four different functions: *normCast()*, *bootCast()*, *trendCast()* and *modelCast()*. *normCast()* and *bootCast()* are forecasting functions for series that follow an ARMA process. *trendCast()* implements a forecasting approach for nonparametric trend functions and works with *smoots* objects. These three functions will not be discussed any further. The main forecasting function, however, is *modelCast()*, as it uses *smoots* objects that are the result of a trend estimation as an input for the argument *obj* and returns point forecasts as well as forecasting intervals for the original series that the trend was estimated for. The results are obtained under the assumption that the original series follows a Semi-ARMA model as described in Section 2.1. The arguments p and q define the orders of the ARMA model that will be implemented for the stationary part of the series. If both arguments are set to *NULL*, the function will automatically choose the best ARMA(p, q) model for $p = 0, 1, \ldots, 5$ and $q = 0, 1, \ldots, 5$ following the Bayesian Information Criterion. The forecasting horizon can be set by the argument h, so that point forecasts and forecasting intervals will be returned for the future time points $n+1, n+2, \ldots, n+h$. Furthermore, a linear extrapolation (*np.fcast = "lin"*) and a constant continuation of the last value (*np.fcast = "const"*) are the two available options for the forecasting of the nonparametric trend function. To adjust the confidence level, the argument *alph* can be used, so that a [(1 - *alph*)100]% forecasting interval will be determined. Two different settings are accessible for the forecasting of the error term. Since according to (1) and (2), ξ_t follows an ARMA model, the distribution of the innovations ϵ_t determines the forecasting approach. As will be discussed in more detail in Sections 5.1 and 5.2, approximately normal forecasting intervals can be quickly obtained for normally distributed innovations (*method = "norm"*), whereas non-Gaussian cases are considered via a bootstrap (*method = "boot"*). Some additional arguments can be adjusted, if in fact a bootstrap is applied to the data. Users can control the total number of bootstrap iterations via the argument *it*. Of course, more iterations deliver a more precise bootstrapped empirical distribution

of the forecasting error and thus a more accurate forecasting interval, however, running a bootstrap usually needs a lot of computer memory and computation time. Thus, a compromise between computation time and accuracy has to be made. Nevertheless, we recommend to run at least 100.000 iterations, which is the default setting for *it*. By means of the argument *n.start*, the number of 'burn-in' observations for the simulated ARMA processes within the bootstrap is determined. For a simulated process, usually $n.start + n$ observations are drawn and the first *n.start* observations are subsequently omitted. Due to the high computation time of a bootstrap, *modelCast()* tries to inform the user of the current progress. *msg* is a single-element numeric argument that controls the frequency of messages being printed to the R console. The messages returned are of the type *Iteration: i*, where i is either 1 or divisable by *msg* without rest. If $msg = NA$ is selected, all messages are suppressed. *modelCast* normally returns a matrix with the forecasting results. If a bootstrap is applied, the user also has the choice to return more bootstrap results. For $export.error = TRUE$ a list with two elements is returned. The first component is the usual forecasting matrix. The second element, however, is a matrix, where each column of the matrix corresponds to the bootstrapped empirical distribution of the forecasting error of the future time points $n + 1$ to $n + h$. Like the function *confBounds()*, *modelCast()* allows the automatic creation of a graphic, if the argument *plot* is set to *TRUE*, and for more arguments of the standard *plot()* function in R that can be implemented via the three dots ellipsis ....

For a higher accessibility of the *smoots* version 1.1.0, two new S3 methods were added to the existing roster of *print.smoots()* and *plot.smoots()*. *fitted.smoots()* enables users without knowledge of the name conventions of the elements of smoots objects to quickly access the trend or derivative estimates. Quite similarly, *residuals.smoots()* returns the residuals that are saved within *smoots* objects that are the result of a trend estimation. Since there are no residuals when estimating the derivatives of the nonparametric trend function, *residuals.smoots()* returns *NULL* for *smoots* objects created by these estimation processes.

2.3 Some important variants

Semi-Log-GARCH, Semi-Log-ACD, ...

To model conditional volatility, Engle (1982) and Bollerslev (1986) introduced the ARCH (autoregressive conditional heteroskedasticity) and GARCH (generalized ARCH) models, respectively, of which the latter has gained many extensions during the past decades. After the introduction of the Log-GARCH model (Pantula, 1986; Geweke, 1986; Milhøj, 1988), it has been established that the model has an ARMA representation that allowed for consistent and asymptotically normal estimators of all parameters of the Log-GARCH representation except for the volatility intercept (Psaradakis and Tzavalis, 1999; Francq and Zakoïan, 2006). After Francq, Wintenberger, and Zakoian (2013) focused once more on the logarithmic GARCH model, it were Sucarrat, Grønneberg, and Escribano (2016) who introduced a consistent and asymptotically normal estimator of the volatility intercept via the ARMA representation by means of QMLE (quasi-maximum-likelihood estimation).

In combination with the release of the *smoots* 1.0.1 package, Feng et al. (2020) introduced a semiparametric Log-GARCH extension (Semi-Log-GARCH) model, which will be described hereinafter. Let r_t be the (demeaned) log-returns. The Semi-Log-GARCH model is then defined by

$$\begin{aligned} r_t = \sqrt{\nu(x_t)}\epsilon_t, \quad \epsilon_t = \sqrt{h_t}\eta_t \quad \text{with} \quad \eta_t \sim IID(0,1) \quad \text{and} \\ \ln(h_t) = \omega + \sum_{i=1}^{p} \alpha_i \ln(\epsilon_{t-i}^2) + \sum_{j=1}^{q} \beta_j \ln(h_{t-j}), \end{aligned} \tag{10}$$

where $t = 1, 2, \ldots, n$, $\mathrm{E}(\epsilon_t^2)$=1, $\Pr(\eta_t = 0) = 0$, $\sqrt{\nu(x_t)} > 0$ is a nonparametric scale function that represents the slowly changing standard deviation and $\sqrt{h_t} > 0$ is the local standard deviation. To ensure the positivity of the scale function during the estimation (references!!!), the trend of the log-transformed series $\ln(r_t^2) = \ln[\nu(x_t)] + \mu_{l\epsilon} + \ln(\epsilon^2) - \mu_{l\epsilon}$ with $\mu_{l\epsilon} = \mathrm{E}[\ln(\epsilon_t^2)]$, which corresponds to the additive component model (1) when defining $y_t = \ln(r_t^2)$, $m(x_t) = \ln[\nu(x_t)] + \mu_{l\epsilon}$ and $\xi_t = \ln(\epsilon_t^2) - \mu_{l\epsilon}$, can be estimated.

Following Sucarrat, Grønneberg, and Escribano (2016), the Log-GARCH model admits an ARMA representation

$$\phi(B)\xi_t = \theta(B)u_t \tag{11}$$

with u_t following $IID(0, \sigma_u^2)$ defined by $\ln(\eta_t^2) - \mu_{l\eta}$ with $\mu_{l\eta} = \mathrm{E}[\ln(\eta_t^2)]$. As stated by Sucarrat, Grønneberg, and Escribano (2016), the Log-GARCH parameters can be derived from the estimated ARMA parameters by using the relations $\omega = \mu_{l\epsilon}(1 - \sum_{i=1}^{p^*} \phi_i) - (1 - \sum_{j=1}^{q} \beta_j)\mu_{l\eta}$, $\phi_i = \alpha_i + \beta_i$ and $\theta_j = -\beta_j$. μ_{le} and $\mu_{l\eta}$ can be estimated consistently from

both the residuals of the nonparametric trend and the parametric ARMA estimation by taking into account that $\mathrm{E}(\epsilon_t^2) = \mathrm{E}(\eta_t^2) = 1$. In consideration of $\ln(h_t) = \xi_t + \mu_{l\epsilon} - \mu_{l\eta} - u_t$, the local standard deviation series is easily obtained from the fitted ARMA representation.

Another important model with respect to the analysis of non-negative financial time series in general is the ACD (autoregressive conditional duration, Engle and Russell, 1998) model, which admits a logarithmic extension (see the Type I model in Bauwens and Giot, 2000), called a Log-ACD, that corresponds to a squared Log-GARCH process. Moreover, Forstinger (2018) provided the definition of and examples with applications to suitable financial data for a semiparametric expansion(!!!) of the Log-ACD (Semi-Log-ACD) model.
The Semi-Log-ACD model is analogous to the previously described Semi-Log-GARCH model. Define the observed values F_t by $F_t = g(x_t)\psi_t$, where $g(x_t) > 0$ is a nonparametric scale function and $\psi_t > 0$ are the standardized observations. Due to the equivalency of the Semi-Log-GARCH and the Semi-Log-ACD model, i.e., F_t is equivalent to r_t^2, $g(x_t)$ to $\nu(x_t)$ and ψ_t to ϵ_t^2, the same estimation approach as previously described can be implemented.

2.4 Further extensions

3 Asymptotically unbiased and normal estimators

Feng

4 Some graphical tools for testing stationarity and linearity

All coauthors (hereby the existing graphics should also be extended to case with $\alpha = 1\%$, another key points are to improve the quality of the plots and introduce those tools into 'smoots')

4.1 Asymptotically unbiased estimators and confidence bounds

An asymptotically (relatively) unbiased estimate of $m^{(\nu)}$, which is sometimes very helpful in practice, e.g. for conducting reasonable confidence bounds, can be obtained by applying a bandwidth $h_{\mathrm{ub}} = o[n^{-1/(2k+1)}]$, that is of a smaller order of magnitude than the optimal bandwidth. Compared to the variance, the bias is now asymptotically negligible and the error in $\hat{m}^{(\nu)}$ is dominated by the former. Hence, we have

$$\sqrt{nh}[m^{(\nu)}(x) - \hat{m}^{(\nu)}(x)] \xrightarrow{\mathcal{D}} N[0, 2\pi c_f R(x)]. \tag{12}$$

Note that there is no unique definition of h_{ub} and hence there is no rule for selecting it. If the estimation of the regression function m is considered, a reasonable choice of a bandwidth for this purpose is with $h = (h_{\mathrm{A}})^{(2k+1)/(2k)} = O[n^{-1/(2k)}]$ such that the bias in $\hat{m}^{(\nu)}$ achieves the parametric convergence rate $O(n^{-1/2})$. This leads to the choices of $h = O(n^{-1/4})$ for a local linear estimator and $h = O(n^{-1/8})$ for a local cubic estimator of m. The use of a bandwidth of an even smaller order is no longer necessary. Based on the data-driven results, a bandwidth $\hat{h}_{\mathrm{ub}} = (\hat{h})^{(2k+1)/(2k)}$ of the above mentioned order can be used to obtain asymptotically unbiased estimates and the corresponding confidence bounds. This idea will also be employed to deduce bandwidths for calculating asymptotically unbiased estimates of m', m'' as well as their confidence bounds from the corresponding selected bandwidths. This leads to $\hat{h}^d_{1,\mathrm{ub}} = (\hat{h}^d_1)^{7/6}$ for estimating m' and $\hat{h}^d_{2,\mathrm{ub}} = (\hat{h}^d_2)^{9/8}$ for estimating m'', respectively.

4.2 The methodology

The most simple econometric approach for analyzing a growth curve is the use of a linear trend function. In growth theory the linear trend is in accordance with Jones (2002) a good starting point. Thus we use the overall linear trend as a benchmark and show that it could be a misspecification for different periods. Our purpose is to use the confidence bounds proposed in the last sub-section as some new graphical tools to test possible linearity of the trend nonparametrically. Related research on nonparametric test of linearity was e.g. done by Hjellvik et al. (1998). Here, the null-hypothesis to be tested is

$$H_0 : y_t = \beta_0 + \beta_1 x_t + \varepsilon_t, \text{ against } H_1 : \text{“}\textit{A linear trend is not suitable.}\text{”}$$

Three variants to test H_0 based on confidence bounds for m, m' and m'', respectively, will be proposed. Firstly, an unbiased local linear estimator $\hat{m}_{\text{ub}}(x)$ and the corresponding confidence bounds at given confidence level $1-\alpha$ can be calculated using the bias-corrected bandwidth $\hat{h}_{\text{ub}} = (\hat{h})^{5/4}$, where $\hat{h}$ is the selected bandwidth for estimating m with $p = 1$. To this end the selected bandwidth with $p = 3$ can also be used, which will however not be considered to save space. Furthermore, a linear trend $\hat{m}_{\text{L}}$ can be fitted to the data as a comparison. Under H_0 the estimated trend $\hat{m}_{\text{L}}$ is unbiased and $\sqrt{n}$-consistent. Those results can be displayed all together in a figure. If $\hat{m}_{\text{L}}$ lies clearly outside the estimated confidence bounds at least at some observation points, H_0 should be rejected at the significance level α, because the probability for that those phenomena can happen is asymptotically less than α. For an important indicator, the fitted linear regression provides an ideal benchmark of the economic development, such a figure hence indicates where the economic development is beyond and where it is below an ideal level.

Secondly, under the linearity null the first derivative of m should always be β_1. Now, we can conduct an asymptotically unbiased estimate of m', $\hat{m}'_{\text{ub}}(x)$ and the corresponding confidence bounds using the bandwidth $\hat{h}^d_{1,\text{ub}} = (\hat{h}^d_1)^{7/6}$. Then $\hat{m}'_{\text{ub}}$ and the corresponding confidence bounds can be displayed and compared to the average level of $\hat{m}'$, which is an estimate of β_1. If the the upper confidence bounds are below this level or the lower confidence bounds are beyond this level at clearly more than $100\alpha\%$ of the observation points, H_0 should be rejected at the significance level α. Thirdly, an asymptotically unbiased estimate of m'', $\hat{m}''_{\text{ub}}(x)$ and the corresponding confidence bounds can be calculated using the bandwidth $\hat{h}^d_{2,\text{ub}} = (\hat{h}^d_2)^{9/8}$. Then $\hat{m}''_{\text{ub}}$ and the corresponding confidence bounds can be displayed and compared to the level zero, since m'' should be uniformly zero, if H_0 holds. In particular, this method does not involve the use of an auxiliary estimate. Now, if the the upper confidence bounds are below zero or the lower confidence bounds are beyond zero at clearly more than $100\alpha\%$ of the observation points, H_0 should be rejected at the significance level α. Results in the last two cases are not only interesting for carrying out the above mentioned test, but can also indicate the details of the growth rate and the curvature of the trend in a macroeconomic time series under consideration.

Note that the first idea can be adjusted to test whether the time series under consideration is stationary or trend-stationary, provided that a constant mean is simply used as the comparison. For instance, this idea can be employed to show, whether a parametric or a

semiparametric Log-GARCH model should be used for modeling the volatility of a given return series. Moreover, this proposal can also be extended to show if another parametric trend function, e.g. a quadratic or a cubic function, could be a suitable choice.

5 Semiparametric forecasting approaches

All coauthors

5.1 Point and interval forecasting under normal errors

We once more assume a given process to follow a Semi-ARMA model defined by equations (1) and (2). The forecast of y_t for a time point $n+k$ with $k = 1, 2, \ldots$ is defined as $\hat{y}_{n+k}$. Consequently, suitable forecasting approaches based both the estimated nonparametric trend function $\hat{m}(x_t)$ and the corresponding residuals $\hat{\xi} = y_t - \hat{m}(x_t)$ must be implemented. Define D as a dummy variable that is either equal to zero or to one. Feng et al. (2018) proposed

$$\hat{m}(x_{n+k}) = \hat{m}(x_n) + D \times k \times \Delta m \tag{13}$$

with $\Delta m = \hat{m}(x_n) - \hat{m}(x_{n-1})$ as a forcasting approach for the nonparametric trend function. If $D = 1$, equation (13) represents a linear extrapolation, whereas for $D = 0$ the forcasts are constant.

The best linear predictions in ARMA models are well-established (see e.g. Section 3.3 in Brockwell and Davis, 2016). Feng et al. (2018) propose to obtain (approximately) best linear predictions from an ARMA model fitted to $\hat{\xi}_t$, because Feng and Zhou (2015) showed that additional errors within the best linear predictions due to errors in the estimated trend function are asymptotically negligible. Therefore, the (approximately) best linear predicitions $\hat{\xi}_{n+k}$ of $\hat{\xi}_t$ are given by

$$\hat{\xi}_{n+k} = \sum_{i=1}^{p} \beta_i \hat{\xi}_{n+k-i} + \sum_{j=1}^{q} \alpha_j \hat{\epsilon}_{n+k-j}, \tag{14}$$

where only for $k = 1$ all values $\hat{\xi}_{n+k-i}$ and $\hat{\epsilon}_{n+k-j}$ are either known or can be estimated. For $k \geq 2$ the values of $\hat{\xi}_{n+k-i}$ that are unknown have to be replaced by the previously forecasted values. Furthermore, $\mathrm{E}(\epsilon_t) = 0$ is considered for unknown values of

$\hat{\epsilon}_{n+k-j}$(!!!reformulate). In consequence of the two point forecasting approaches to obtain both $\hat{m}(x_{n+k})$ and $\hat{\xi}_{n+k}$, $\hat{y}_{n+k} = \hat{m}(x_{n+k}) + \hat{\xi}_{n+k}$ are the proposed total point predictions for a Semi-ARMA model (Feng et al., 2018).

The variance of the error in $\hat{m}(x_t)$ is of order $O(n^{-2/5})$ and is omitted for simplicity as proposed by Feng et al. (2018). If the process described in equation (2) is stationary, it admits an MA(∞) representation $\xi_t = \sum_{i=0}^{\infty} c_i \epsilon_{t-i}$ with $c_0 = 1$ and $\sum_{i=0}^{\infty} |c_i| < \infty$. An individual forecast derived from this representation is then given by $\xi_{n+j} = \hat{\xi}_{n+j} + \sum_{i=0}^{j-1} c_i \epsilon_{n+i}$ and consequently, the asymptotic mean squared error of a point forecast is $\tilde{\sigma}_k^2 = \sigma_\epsilon^2 \sum_{i=0}^{k-1} c_i^2$ (see e.g. equation (3.3.19) in Brockwell and Davis, 2016).

Under the normality assumption, i.e. $\epsilon_t \sim \mathrm{N}(0, \sigma_\epsilon^2)$, suitable forecasting intervals for $\hat{y}_{n+k}$ can therefore be obtained by considering only the error in $\hat{\xi}_t$, i.e. $\hat{y}_{n+k} \pm \Phi_{1-\alpha/2} \tilde{\sigma}_k$ are the prediction bounds for $\hat{y}_{n+k}$ proposed by Feng et al. (2018), where $(1 - \alpha)$ is the probability that y_{n+k} lies within the bounds and Φ_i is the i-th quantile of the standard normal distribution.

5.2 Bootstrapping the forecasting intervals in non-Gaussian cases

If the error term does not follow a Gaussian white noise or if the error distribution is unknown, forecasting intervals can also be computed with use of standard statistical software. An approach for the estimation of forecasting intervals via forward bootstrap approaches in AR and ARIMA processes were first introduced by Masarotto (1990) and Pascual, Romo, and Ruiz (2004), respectively. Pan and Politis (2016) proposed a forward bootstrap for autoregressive time series model of order p, where the bootstrapped point forecasts are obtained via the original series rather than the simulated data. Subsequently, Lu and Wang (2018) expanded the algorithm to ARMA(p, q) processes to obtain forecasting intervals for the future time point $n + 1$.

The *B-ARMARoots* algorithm (Lu and Wang, 2018) is conducted under the assumption that the orders p and q of the underlying ARMA(p, q) process are known. We propose to expand the algorithm by Lu and Wang (2018) to multiple future time points $n + k$ with $k = 1, 2, \ldots, h$. Let n be the number of observations, X_t the observed series and $s = 1, 2, \ldots, M$ the number of iteration with M being the final iteration.

1. Estimate the coefficients $\mu_X = \mathrm{E}(X_t)$, α_j and β_i of the underlying ARMA process. Obtain the point forecasts $\hat{X}_{n+1}, \hat{X}_{n+2}, \ldots, \hat{X}_{n+h}$.
2. Calculate the residuals $\hat{\epsilon}_t$ based on the fitted model in step 1. Define $\hat{F}_\epsilon$ as the empirical distribution of the demeaned residuals. Set $s = 1$.
3. Draw a sample $\epsilon^*_{t,s}$ from $\hat{F}_\epsilon$ and simulate a series $X^*_{1,s}, X^*_{2,s}, \ldots, X^*_{n,s}$ based on the sampled residuals and the estimated values $\hat{\mu}_X$, $\hat{\alpha}_j$ and $\hat{\beta}_i$.
4. Obtain the estimates $\hat{\mu}^*_{X,s}$, $\hat{\alpha}^*_{j,s}$ and $\hat{\beta}^*_{i,s}$ for the simulated data.
5. Calculate the residual series $\hat{\epsilon}^{**}_{t,s}$ from the original process X_t based on the coefficient estimates $\hat{\mu}^*_{X,s}$, $\hat{\alpha}^*_{j,s}$ and $\hat{\beta}^*_{i,s}$.
6. Obtain the point forecasts $\hat{X}^*_{n+1,s}, \hat{X}^*_{n+2,s}, \ldots, \hat{X}^*_{n+h,s}$ iteratively, where $\hat{X}^*_{n+k,s} = \sum_{i=1}^{p} \hat{\beta}^*_{i,s}(X^*_{n+k-i,s} - \hat{\mu}^*_{X,s}) + \sum_{j=1}^{q} \hat{\alpha}^*_{j,s}\hat{\epsilon}^{**}_{n+k-j,s} + \hat{\mu}^*_{X,s}$ with $X^*_{n+k-i,s} = X_{n+k-i}$ for $t - i \leq n$. Calculate the future bootstrap observations $X_{n+k,s} = \sum_{i=1}^{p} \hat{\beta}_i(X_{n+k-i} - \hat{\mu}_X) + \sum_{j=1}^{q} \hat{\alpha}_j\hat{\epsilon}_{n+k-j} + \epsilon^*_{n+k,s} + \hat{\mu}_X$ iteratively for each k.
7. Calculate the forecasting error $X^*_{n+k,s} - \hat{X}^*_{n+k,s}$ for each k. Increase s by one.
8. Execute steps 3 to 7 a total of M times and obtain the empirical distributions of $X^*_{n+k} - \hat{X}^*_{n+k}$ for each k with their respective $\alpha/2$-quantiles being denoted by $q_k(\alpha/2)$. The forecasting intervals are now given by $[\hat{X}_{n+k} + q_k(\alpha/2); \hat{X}_{n+k} + q_k(1 - \alpha/2)]$.

5.3 Backtesting the results

(?)Idee: in-sample u. out-sample –> one-step-ahead rolling forecast; Häufigkeit der Überschreitungen muss ca. 1 - Konfidenzniveau für alle forecastings sein.

6 Application of linearity and stationarity tests

6.1 Empirical results for the data examples

The proposed methods are applied to the two data examples. For the regression function the two selected examples in figure 1 with $p = 1$ are used. The estimates of m' and m'' are those mentioned at the end of Section 5.2 with the pilot estimate $\hat{c}_f$ obtained by $p_p = 1$. The bandwidth for calculating the asymptotically unbiased estimates and their confidence bounds are calculated according to the formulas given in Section 6.2. Results

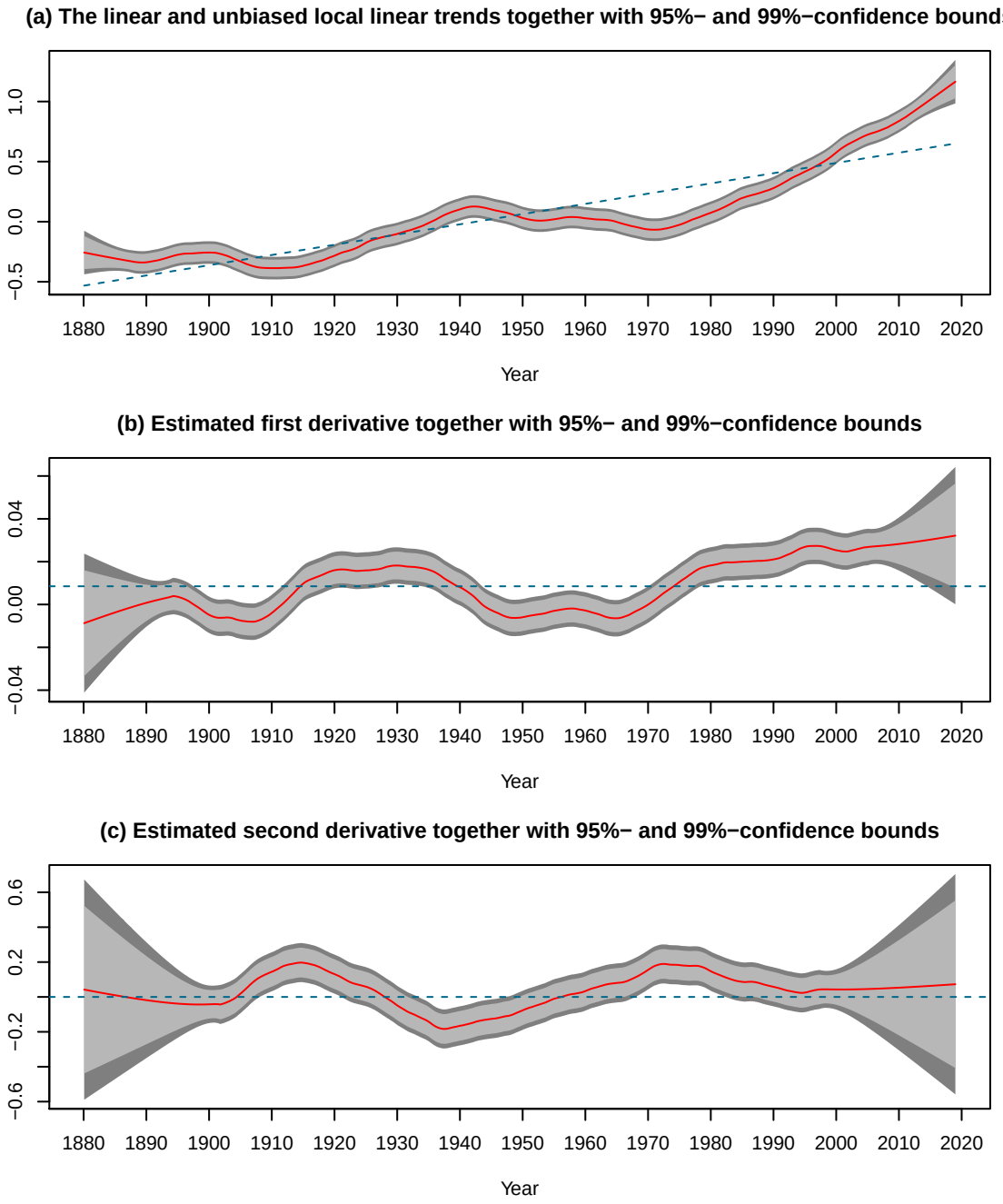

Figure 1: Results of different linearity tests for the temperature changes data.

for the US example are displayed in Figure 3. From this figure we see that all of the three methods indicate the similar conclusion that a linear trend is not suitable and should not be used. The fact is more clear at both ends of this time series, and in particular at the current end, where a decreasing growth rate of the US economy is observable. We can also see that the economic development between the middle of 1970s and 2000 was likely

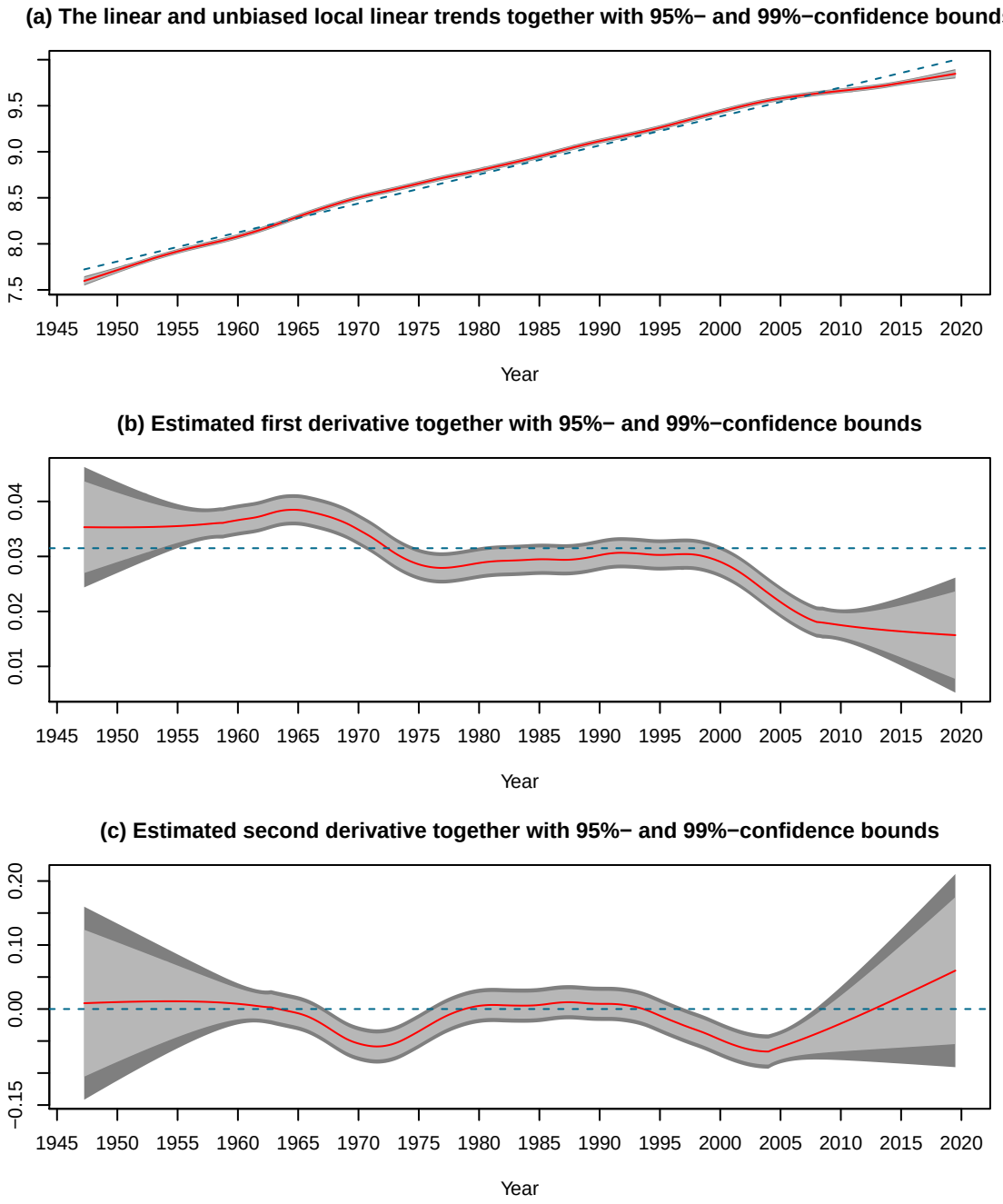

Figure 2: Results of different linearity tests for the US-data.

with a linear trend (in the log-data). Furthermore, also local polynomial regression has automatic boundary correction (of the order of magnitude of the bias), the estimation quality at a boundary point is poorer than that in the interior. As expected, the higher the order of the derivative the more clear this effect so that the length of the confidence bounds for m' and in particular for m'' at points near to the end-points becomes several

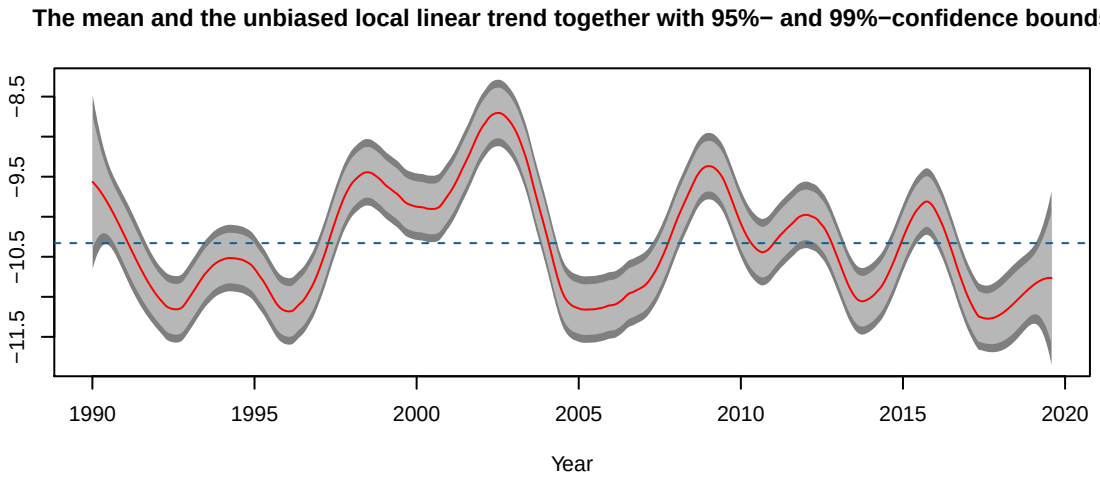

Figure 3: Result of a stationarity test for the DAX returns.

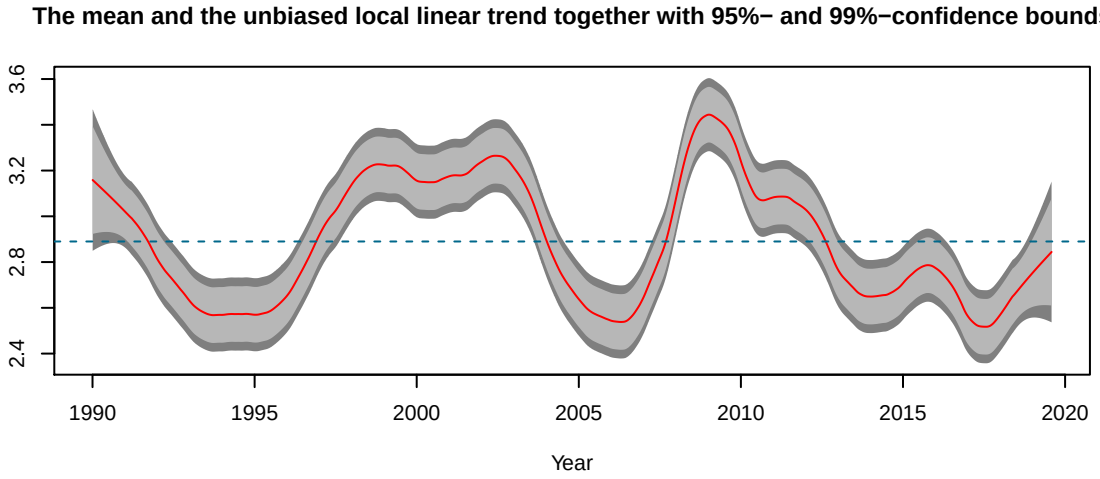

Figure 4: Result of a stationarity test for the VIX.

times of that at an interior point. This leads to slightly different conclusions according to different approaches. Both of the first two panels in Figure 3 indicate that the growth rate in the US is decreasing for more than a decade. However, the confidence bounds for m'' show that a linear trend at the current end cannot be rejected, but this is due to an increase in the variance at the boundary. The results for the UK data are shown in Figure 4. Similar conclusions as for the US example can be drawn here. A clear difference between the two economies might be that the non-linearity in the UK economic development is more obvious than the non-linearity in the US economic development.

7 Application of the forecasting techniques

All coauthors

8 Concluding remarks

Feng, Gries

References

Bauwens, L. and P. Giot (2000). The logarithmic ACD model: an application to the bid-ask quote process of three NYSE stocks. In: *Annales d'Economie et de Statistique*, pp. 117–149.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. In: *Journal of econometrics* 31.3, pp. 307–327.

Brockwell, P. J. and R. A. Davis (2016). *Introduction to time series and forecasting*. springer.

Bühlmann, P. (1996). Locally adaptive lag-window spectral estimation. In: *Journal of Time Series Analysis* 17.3, pp. 247–270.

Eddelbuettel, D. and R. François (2011). Rcpp: Seamless R and C++ Integration. In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08. URL: http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, D. and C. Sanderson (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. In: *Computational Statistics and Data Analysis* 71, pp. 1054–1063. URL: http://dx.doi.org/10.1016/j.csda.2013.02.005.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. In: *Econometrica: Journal of the Econometric Society*, pp. 987–1007.

Engle, R. F. and J. R. Russell (1998). Autoregressive conditional duration: a new model for irregularly spaced transaction data. In: *Econometrica*, pp. 1127–1162.

Feng, Y., T. Gries, and M. Fritz (2020). Data-Driven Local Polynomial for the Trend and Its Derivatives in Economic Time Series. In: *Journal of Nonparametric Statistics* 32.2, pp. 510–533. DOI: 10.1080/10485252.2020.1759598.

Feng, Y. and S. Heiler (2009). A simple bootstrap bandwidth selector for local polynomial fitting. In: *Journal of Statistical Computation and Simulation* 79.12, pp. 1425–1439.

Feng, Y., S. Letmathe, and D. Schulz (2020). *smoots: Nonparametric Estimation of the Trend and Its Derivatives in TS*. R package version 1.1.0. URL: https://CRAN.R-project.org/package=smoots.

Feng, Y. and C. Zhou (2015). Forecasting financial market activity using a semiparametric fractionally integrated Log-ACD. In: *International Journal of Forecasting* 31.2, pp. 349–363.

Feng, Y. et al. (2018). Forecasting Economic Growth Processes for Developing Economies. Forthcoming.

Feng, Y. et al. (2020). The smoots Package in R for Semiparametric Modeling of Trend Stationary Time Series. forthcoming.

Forstinger, S. (2018). Modelling and Forecasting Financial and Economic Time Series Using Different Semiparametric ACD Models. PhD thesis. Paderborn, Universität Paderborn.

Francq, C., O. Wintenberger, and J.-M. Zakoian (2013). GARCH models without positivity constraints: Exponential or Log GARCH? In: *Journal of Econometrics* 177.1, pp. 34–46.

Francq, C. and J.-M. Zakoïan (2006). Linear-representation based estimation of stochastic volatility models. In: *Scandinavian Journal of Statistics* 33.4, pp. 785–806.

Geweke, J (1986). Comment On: Modelling the Persistence of Conditional Variances. In: *Econometric Reviews* 5, pp. 57–61. DOI:10.1080/07474938608800097.

Lu, X. and L. Wang (2018). Bootstrap prediction interval for ARMA models with unknown orders. In: *Revstat*. URL: `https://www.ine.pt/revstat/pdf/BootstrappredictionintervalforARMA` `pdf`. Forthcoming, date of acceptance: March 2018.

Masarotto, G. (1990). Bootstrap prediction intervals for autoregressions. In: *International Journal of Forecasting* 6.2, pp. 229–239.

Milhøj, A. (1988). *A Multiplicative Parametrization of ARCH Models*. Universitetets Statistiske Institut.

Pan, L. and D. N. Politis (2016). Bootstrap prediction intervals for linear, nonlinear and nonparametric autoregressions. In: *Journal of Statistical Planning and Inference* 177, pp. 1–27.

Pantula, S. G. (1986). Modeling the Persistence of Conditional Variances: A Comment. In: *Econometric Reviews* 5, pp. 79–97.

Pascual, L., J. Romo, and E. Ruiz (2004). Bootstrap predictive inference for ARIMA processes. In: *Journal of Time Series Analysis* 25.4, pp. 449–465.

Psaradakis, Z. and E. Tzavalis (1999). On regression-based tests for persistence in logarithmic volatility models. In: *Econometric Reviews* 18.4, pp. 441–448.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Sanderson, C. and R. Curtin (2016). Armadillo: a template-based C++ library for linear algebra. In: *Journal of Open Source Software* 1.2, p. 26.

Sucarrat, G., S. Grønneberg, and A. Escribano (2016). Estimation and inference in univariate and multivariate log-GARCH-X models when the conditional density is unknown. In: *Computational statistics & data analysis* 100, pp. 582–594.