

Härdle, Wolfgang; Tschernig, Rolf

**Working Paper**

## Flexible time series analysis

SFB 373 Discussion Paper, No. 2000,51

**Provided in Cooperation with:**

Collaborative Research Center 373: Quantification and Simulation of Economic Processes,  
Humboldt University Berlin

*Suggested Citation:* Härdle, Wolfgang; Tschernig, Rolf (2000) : Flexible time series analysis, SFB 373 Discussion Paper, No. 2000,51, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin,  
<https://nbn-resolving.de/urn:nbn:de:kobv:11-10047775>

This Version is available at:

<https://hdl.handle.net/10419/62228>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*

# Flexible Time Series Analysis

Wolfgang Härdle and Rolf Tschernig

In this chapter we present nonparametric methods and available quantlets for **nonlinear modelling of univariate time series**. A general nonlinear time series model for an univariate stochastic process  $\{Y_t\}_{t=1}^T$  is given by the **heteroskedastic nonlinear autoregressive (NAR) process**

$$Y_t = f(Y_{t-i_1}, Y_{t-i_2}, \dots, Y_{t-i_m}) + \sigma(Y_{t-i_1}, Y_{t-i_2}, \dots, Y_{t-i_m})\xi_t, \quad (1)$$

where  $\{\xi_t\}$  denotes an i.i.d. noise with zero mean and unit variance and  $f(\cdot)$  and  $\sigma(\cdot)$  denote the conditional mean function and conditional standard deviation with lags  $i_1, \dots, i_m$ , respectively. In practice, the conditional functions  $f(\cdot)$  and  $\sigma(\cdot)$  as well as the number of lags  $m$  and the lags itself  $i_1, \dots, i_m$  are unknown and have to be estimated.

In Section 1 we discuss nonparametric estimators for the conditional mean function of nonlinear autoregressive processes of order one. While this case has been most intensively studied in theory, in practice models with several lags are often more appropriate. Section 2 covers the estimation of the latter, including the selection of appropriate lags. For all models we discuss methods of bandwidth selection which aim at an optimal trade-off between variance and bias of the presented estimators.

Both sections contain practical examples. The corresponding quantlets for fitting nonlinear autoregressive processes of order one are contained in the quantlib `smoother`. A number of quantlets for fitting higher order models are found in the third party quantlib `tp/cafpe/cafpe`.

Although obvious we would like to mention that in the following we only discuss methods for which quantlets are available. For an overview of alternative methods and models we would like to refer the reader to the surveys of Tjøstheim (1994) or Härdle, Lütkepohl, and Chen (1997).

# 1 Nonlinear Autoregressive Models of Order One

## 1.1 Estimation of the Conditional Mean

```
mh = regxest(x{, h, K, v})
      computes the univariate conditional mean function using the
      Nadaraya-Watson estimator

mh = regest(x{, h, K, v})
      computes the univariate conditional mean function using the
      Nadaraya-Watson estimator and WARPing

mh = lprexest(x{, h, p, v})
      computes the univariate conditional mean function using local
      polynomial estimation

mh = lpregest(x{, h, p, K, d})
      computes the univariate conditional mean function using local
      polynomial estimation and WARPing
```

Let us turn to estimating the conditional mean function  $f(\cdot)$  of a nonlinear autoregressive processes of order one (NAR(1) process)

$$Y_t = f(Y_{t-1}) + \sigma(Y_{t-1})\xi_t \quad (2)$$

using nonparametric techniques. The basic idea is to estimate a Taylor approximation of order  $p$  of the unknown function  $f(\cdot)$  around a given point  $y$ . The simplest Taylor approximation is obtained if its order  $p$  is chosen to be zero. One then approximates the unknown function by a constant. Of course, this approximation may turn out to be very bad if one includes observations  $Y_{t-1}$  that are distant to  $y$  since this might introduce a large approximation bias. One therefore weights those observations less in the estimation. Using the least squares principle, the estimated function value  $\hat{f}(y, h)$  is provided by the estimated constant  $\hat{c}_0$  of a local constant estimate around  $y$

$$\hat{c}_0 = \arg \min_{\{c_0\}} \sum_{t=2}^T \{Y_t - c_0\}^2 K_h(Y_{t-1} - y), \quad (3)$$

where  $K$  denotes the weighting function, which is commonly called a kernel function, and  $K_h(Y_{t-1} - y) = h^{-1}K\{(Y_{t-1} - y)/h\}$ . A number of kernel functions are used in practice, e.g. the Gaussian density function or the quartic kernel  $K(u) = 15/16(1 - u^2)^2$  on the range  $[-1, 1]$  and  $K(u) = 0$  elsewhere.  $\hat{f}(y, h) = \hat{c}_0$  is known as the Nadaraya-Watson or local constant function estimator and can be written as

$$\hat{f}(y, h) = \frac{\sum_{t=2}^T K_h(Y_{t-1} - y)Y_t}{\sum_{t=2}^T K_h(Y_{t-1} - y)}. \quad (4)$$

The parameter  $h$  is called bandwidth parameter and controls the weighting of the lagged variables  $Y_{t-1}$  with respect to their distance to  $y$ . While choosing  $h$  too small and therefore including only few observations in the estimation procedure leads to a too large estimation variance, taking  $h$  too large implies a too large approximation bias. Methods for bandwidth selection are presented in Subsection 1.2.

Before one applies Nadaraya-Watson estimation one should be aware of the conditions that the underlying data generating mechanism has to fulfil such that the estimator has nice asymptotic properties: most importantly, the function  $f(\cdot)$  has to be continuous, the stochastic process has to be stationary and the dependence among the observations must decline fast enough if the distance among the observations increases. For measuring dependence in nonlinear time series one commonly uses various mixing concepts. For example, a sequence is said to be  $\alpha$ -mixing (strong mixing) (Robinson 1983) if

$$\sup_{A \in \mathcal{F}_1^n, B \in \mathcal{F}_{n+k}^\infty} |P(A \cap B) - P(A)P(B)| \leq \alpha_k,$$

where  $\alpha_k \rightarrow 0$  and  $\mathcal{F}_i^j$  is the  $\sigma$ -field generated by  $X_i, \dots, X_j$ . An alternative and stronger condition is given by the  $\beta$ -mixing condition (absolute regularity)

$$E \sup \{|P(B|A) - P(B)|\} \leq \beta(k)$$

for any  $A \in \mathcal{F}_1^n$  and  $B \in \mathcal{F}_{n+k}^\infty$ . An even stronger condition is the  $\phi$ -mixing (uniformly mixing) condition (Billingsley 1968) where

$$|P(A \cap B) - P(A)P(B)| \leq \phi_k P(A)$$

for any  $A \in \mathcal{F}_1^n$  and  $B \in \mathcal{F}_{n+k}^\infty$  and  $\phi_k$  tends to zero for  $k \rightarrow \infty$ . The rate at which  $\alpha_k$ ,  $\beta_k$  or  $\phi_k$  go to zero plays an important role in showing asymptotic properties of the nonparametric smoothing procedures. We note that these

conditions are in general difficult to check. However, if the process follows a stationary Markov chain, then geometric ergodicity implies absolute regularity, which in turn implies strong mixing conditions. Techniques exist for checking geometric ergodicity, see e.g. Doukhan (1994) or Lu (1998). Further and more detailed conditions will be discussed in Subsection 2.2.

The quantlet `regxest` allows to compute Nadarya-Watson estimates of  $f(\cdot)$  for an array of different  $y$ 's. Its syntax is

```
mh = regxest(x{, h, K, v})
```

with the input variables

- x**  
 $(T - 1) \times 2$  matrix, in the first column the independent, in the second column the dependent variable,
- h**  
scalar, bandwidth for which if not given, 20% of the range of the values in the first column of **x** is used,
- K**  
string, kernel function on  $[-1,1]$  or Gaussian kernel "gau" for which if not given, the Quartic kernel "qua" is used,
- v**  
 $m \times 1$  vector of values of the independent variable on which to compute the regression for which if not given, **x** is used.

This quantlet returns a  $(T - 1) \times 2$  or  $m \times 2$  matrix **mh**, where the first column is the sorted first column of **x** or the sorted **v**, the second column contains the regression estimate on the values of the first column.

In order to illustrate the methods presented in this chapter, we model the dynamics underlying the famous annual Canadian lynx trappings in 1821–1934, see e.g. Brockwell and Davis (1991, Appendix, Series G). Figures 1 and 2 of their original and logged time series are obtained with the quantlet

```
library("plot")
setsize(640,480)
lynx      = read("lynx.dat") ; read data
```

```

d1          = createdisplay(1,1)
x1          = #(1821:1934)~lynx
setmaskl (x1, (1:rows(x1))', 0, 1)
show(d1,1,1,x1)          ; plot data
setgopt(d1,1,1,"title","Annual Canadian Lynx
                        Trappings, 1821-1934")
setgopt(d1,1,1,"xlabel","Years","ylabel","Lynx")
d2          = createdisplay(1,1)
x2          = #(1821:1934)~log(lynx)
setmaskl (x2, (1:rows(x2))', 0, 1)
show(d2,1,1,x2)          ; plot data
setgopt(d2,1,1,"title","Logs of Annual Canadian
                        Lynx Trappings, 1821-1934")
setgopt(d2,1,1,"xlabel","Years","ylabel","Lynx")

```

 flts01.xpl

Their inspection indicates that taking logarithms is required to make the time series look stationary. The following quantlet reads the lynx data set, constructs the vectors of the dependent and lagged variables, computes the Nadaraya-Watson estimator and plots the resulting function including the scatter plot which is displayed in Figure 3. For selecting the bandwidth we use here the primitive rule to take one fifth of the data range.

```

library("smoother")
library("plot")
setsize(640,480)
;          data preparation
lynx      = read("lynx.dat")
lynxrows  = rows(lynx)
lag1      = lynx[1:lynxrows-1] ; vector of first lag
y         = lynx[2:lynxrows]   ; vector of dep. var.
data      = lag1~y
data      = log(data)
;          estimation
h         = 0.2*(max(data[,1])-min(data[,1])); crude bandwidth
"Bandwidth used" h
mh        = regxest(data,h)      ; N-W estimation
;          graphics
mh        = setmask(mh,"line","blue")
xy        = setmask(data,"cross","small")

```

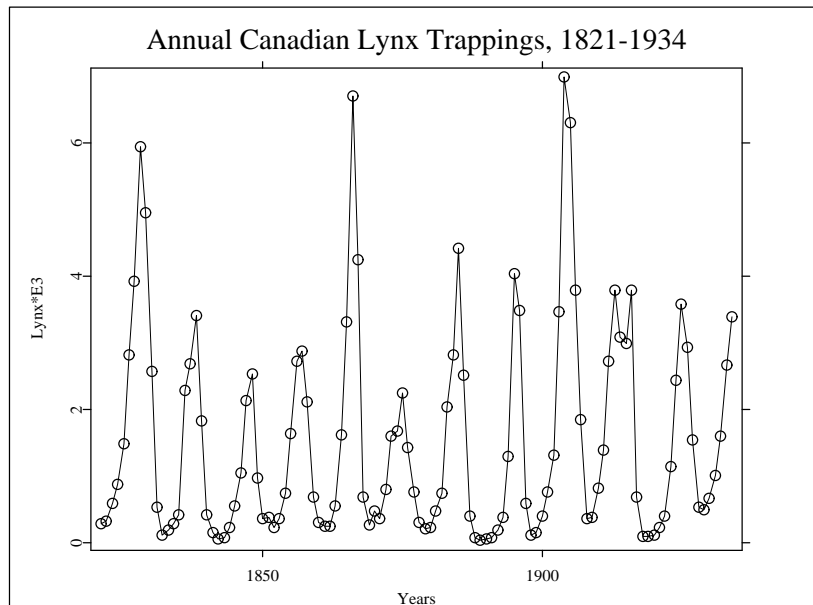



Figure 1: Time series of annual Canadian Lynx Trappings, 1821–1934

```
plot(xy,mh)
setgopt(plotdisplay,1,1,"title","Estimated NAR(1)
                                             mean function")
setgopt(plotdisplay,1,1,"xlabel","First Lag","ylabel","Lynx")
```

 flts02.xpl

For long time series the computation of the Nadaraya-Watson estimates may become quite slow since there are more points at which to estimate the function and each estimation involves more data. In this case one may use the WARPing, **weighted average of rounded points**, technique. The basic idea is the “binning” of the data in bins of length  $d$ . Each observation is then replaced by the bincenter of the corresponding bin which means that each point is rounded to the precision given by  $d$ . A typical choice for  $d$  is  $h/5$  or  $(\max Y_{t-1} - \min Y_{t-1})/100$ . In the latter case, the effective sample size  $r$ , i.e. the number

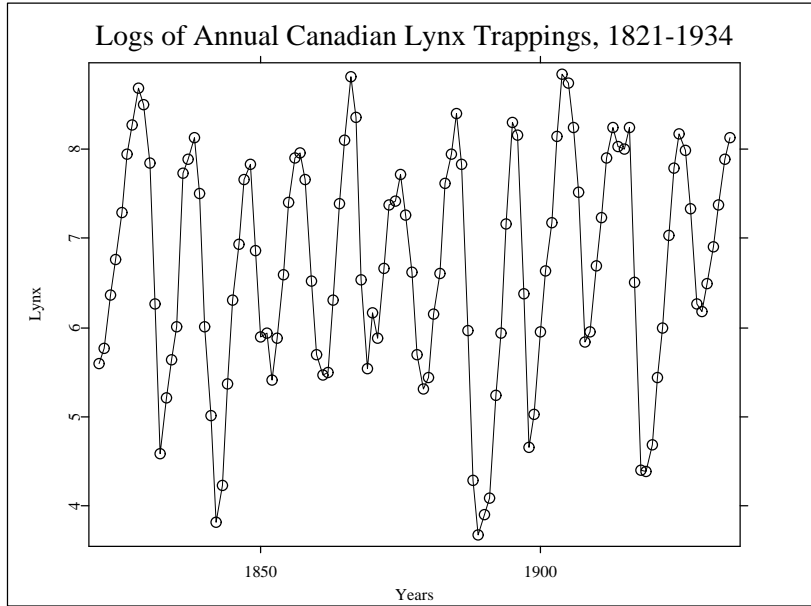


Figure 2: Time series of logarithm of annual Canadian Lynx Trappings, 1821–1934

of nonempty bins, for computation is at most 101. If WARPing is necessary, just call the quantlet `regest` which has the same parameters as the quantlet `regxest`.

While the Nadaraya-Watson function estimate is simple to compute it may suffer from a substantial estimation bias due to the zero order Taylor expansion. Therefore, it seems natural to increase the order  $p$  of the expansion. For example, by selecting  $p = 1$  one obtains the local linear estimator which corresponds to the following weighted minimization problem

$$\{\hat{c}_0, \hat{c}_1\} = \arg \min_{\{c_0, c_1\}} \sum_{t=2}^T \{Y_t - c_0 - c_1(Y_{t-1} - y)\}^2 K_h(Y_{t-1} - y), \quad (5)$$

where the estimated function value  $\hat{f}_2(y, h)$  is provided as before by the esti-



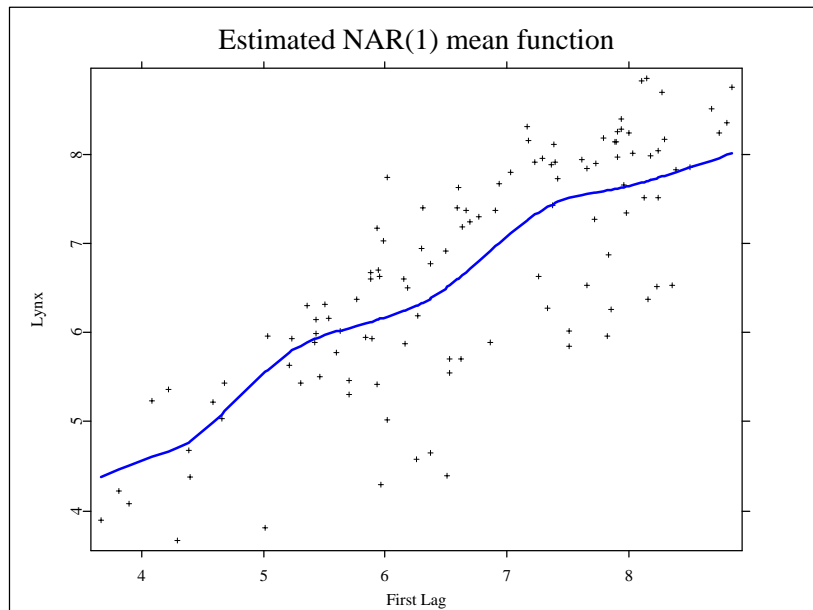


Figure 3: Nadaraya-Watson estimates of NAR(1) mean function for lynx data and scatter plot

mated constant  $\hat{c}_0$ . In a similar way one obtains the local quadratic estimator if one chooses  $p = 2$ . The quantlet `lprexest` allows to compute local linear or local quadratic function estimates using the quartic kernel. Its syntax is

```
y = lprexest (x,h {,p {,v}})
```

where the inputs are:

**x**

$(T - 1) \times 2$  matrix, in the first column the independent, in the second column the dependent variable,

**h**

scalar, bandwidth for which if not given, the rule-of-thumb bandwidth

computed by the quantlet `lpregot` is used,

`p`  
integer, order of polynomial: `p=0` yields the Nadaraya-Watson estimator, `p=1` yields local linear estimation (which is default), `p=2` (local quadratic) is the highest possible order,

`v`  
 $m \times 1$ , values of the independent variable on which to compute the regression for which if not given, `x` is used.

The output is given by the

`mh`  
 $(T - 1) \times 2$  or  $m \times 2$  matrix, the first column is the sorted first column of `x` or the sorted `v`, the second column contains the regression estimate on the values of the first column.

The following quantlet allows to visualize the difference between local constant and local linear estimation of the first order nonlinear autoregressive mean function for the `lynx` data. It produces Figure 4 where the solid and dotted lines display the local linear and local constant estimates, respectively. One notices that the local linear function estimate shows less variation.

```
library("smoother")
library("plot")
setsize(640,480)
;                               data preparation
lynx      = read("lynx.dat")
lynxrows  = rows(lynx)
lag1      = lynx[1:lynxrows-1]   ; vector of first lag
y         = lynx[2:lynxrows]     ; vector of dep. var.
data      = lag1~y
data      = log(data)
;                               estimation
h         = 0.2*(max(data[,1])-min(data[,1])); crude bandwidth
mh        = regxest(data,h)       ; N-W estimation
mhlp      = lprexest(data,h)     ; local linear estimation
;                               graphics
mh        = setmask(mh,"line","blue","dashed")
```

```

mhlp      = setmask(mhlp,"line","red")
xy        = setmask(data,"cross","small")
plot(xy,mh,mhlp)
setgopt(plotdisplay,1,1,"title","Estimated NAR(1)
                                     mean function")
setgopt(plotdisplay,1,1,"xlabel","First Lag","ylabel","Lynx")

```

flts03.xpl

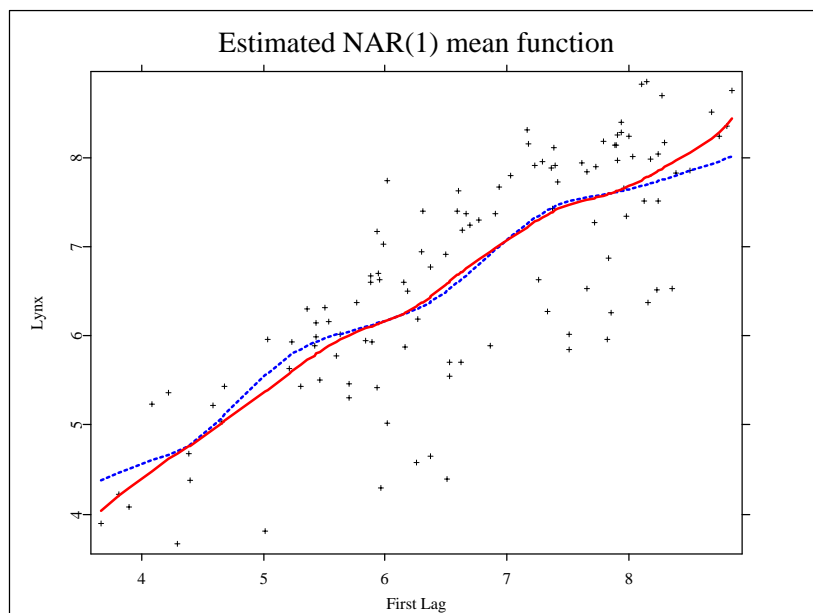


Figure 4: Local linear estimates (solid line) and Nadaraya-Watson estimates (dotted line) of NAR(1) mean function for lynx data and scatter plot

Like Nadaraya-Watson estimation local linear estimation may become slow for long time series. In this case, one may use the quantlet `lpregest` which uses the WARPing technique.

## 1.2 Bandwidth Selection

```
{hcrit, crit} = regxbwsel(x{, h, K})
    interactive tool for bandwidth selection in univariate kernel re-
    gression estimation.

{hcrit, crit} = regbwsel(x{, h, K, d})
    interactive tool for bandwidth selection in univariate kernel re-
    gression estimation using the WARPing method.
```

So far we have used a primitive way of selecting the bandwidth parameter  $h$ . Of course, there are better methods for bandwidth choice. They are all based on minimizing some estimated distance measures. Since we are interested in one bandwidth for various  $y$ , we look at “global” distances like, for instance, the **integrated squared error** (ISE)

$$d_I(h) = \int \left\{ f(y) - \hat{f}(y, h) \right\}^2 w(y) \mu(y) dy. \quad (6)$$

Here  $\mu(\cdot)$  denotes the density of the stationary distribution and  $w(\cdot)$  is a weight function with compact support. Note that the bandwidth which minimizes the ISE  $d_I(h)$  in generally varies from sample to sample. In practice, one may want to avoid the integration and consider an approximation of the ISE, namely the **average squared error** (ASE)

$$d_A(h) = \frac{1}{T-1} \sum_{t=2}^T \left\{ f(Y_{t-1}) - \hat{f}(Y_{t-1}, h) \right\}^2 w(Y_{t-1}). \quad (7)$$

Since the measure of accuracy  $d_A(h)$  involves the unknown autoregression function  $f(\cdot)$ , it cannot be used directly. Instead, one may estimate  $f(Y_{t-1})$  by  $Y_t$ . One then obtains the **average squared error of prediction** (ASEP)

$$d_{AP}(h) = \frac{1}{T-1} \sum_{t=2}^T \left\{ Y_t - \hat{f}(Y_{t-1}, h) \right\}^2 w(Y_{t-1}). \quad (8)$$

This, however, implies the new problem that  $d_{AP}(h)$  can be driven to zero by choosing  $h$  small enough. To see this consider the Nadaraya-Watson estimator (4) and imagine that the bandwidth  $h$  is chosen so small that (4) becomes  $\hat{f}(Y_{t-1}, h) = Y_t$ . This implies  $d_{AP}(h) = 0$ . This estimation problem can easily

be solved by always leaving out  $Y_t$  in computing (4) which leads to

$$\hat{f}_{-t}(y) = \frac{\sum_{i=2, i \neq t}^T K_h(Y_{i-1} - y) Y_i}{\sum_{i=2, i \neq t}^T K_h(Y_{i-1} - y)} \quad (9)$$

and is called the **leave-one-out cross-validation estimate** of the autoregression function. One therefore estimates  $d_{AP}(h)$  with the cross-validation function

$$CV(h) = \frac{1}{T-1} \sum_{t=2}^T \left\{ Y_t - \hat{f}_{-t}(Y_{t-1}, h) \right\}^2 w(Y_{t-1}). \quad (10)$$

Let  $\hat{h}$  be the bandwidth that minimizes  $CV(h)$ . Härdle (1990) and Härdle and Vieu (1992) proved that under an  $\alpha$ -mixing condition,

$$\frac{d_A(\hat{h})}{\inf_h d_A(h)} \rightarrow 1 \quad \text{in probability.}$$

The interactive quantlet `regxbwsel` offers cross-validation and other bandwidth selection methods. The latter may be used in case of independent data. It is called by

```
{hcrit, crit} = regxbwsel(x{, h, K})
```

with the input variables:

**x**

$(T - 1) \times 2$  vector of the data,

**h**

$m \times 1$  vector of bandwidths,

**K**

string, kernel function on  $[-1, 1]$  e.g. quartic kernel "qua" (default) or Gaussian kernel "gau".

The output variables are:

**hcrit**


$p \times 1$  vector, selected bandwidths by the different criteria,

crit

$p \times 1$  string vector, criteria considered for bandwidth selection.

If one wants to use WARPing one has to use the quantlet `regbwsel`. Using the following quantlet one may estimate the cross-validation bandwidth for the lynx data set and obtains  $\hat{h} = 1.12085$ .

```
library("smoother")
library("plot")
setsize(640,480)
;                               data preparation
lynx      = read("lynx.dat")
lynxrows  = rows(lynx)
lag1      = lynx[1:lynxrows-1]      ; vector of first lag
y         = lynx[2:lynxrows]       ; vector of dep. var.
data      = lag1~y
data      = log(data)
;
tmp       = regbwsel(data)
```

 flts04.xpl

It was already noted that the optimal bandwidth with respect to ISE (6) or ASE (7) may vary across samples. In order to obtain a sample independent optimal bandwidth one may consider the **mean integrated squared error** (MISE)

$$d_M(h) = E \left[ \int \left\{ f(y) - \hat{f}(y, h) \right\}^2 w(y) \mu(y) dy \right]. \quad (11)$$

Like  $d_I(h)$  or  $d_A(h)$ , it also cannot be used directly. It is, however, possible to derive the asymptotic expansion of  $d_M(h)$ . This allows to obtain an explicit formula for the asymptotically optimal bandwidth  $h_{opt}$  which, however, contains unknown constants. In Subsection 2.2 we show how one can estimate these unknown quantities in order to obtain a plug-in bandwidth  $\hat{h}_{opt}$ .

### 1.3 Diagnostics

```
acfplot(x)
    generates plot of autocorrelation function of time series contained
    in vector x.

{jb, probjb, sk, k} = jarber(x, 1)
    checks for normality of the data contained in vector x using the
    Jarque-Bera test.
```

It is well known that if a fitted model is misspecified, then resulting inference can be misleading like, for example, for confidence intervals or significance tests. One way to check whether a chosen model is correctly specified is to investigate the resulting residuals. Most importantly, one checks for autocorrelation remaining in the residuals. This can easily be done by inspecting the graph of the autocorrelation function using the quantlet `acfplot`. It only requires the  $(T - 1) \times 1$  vector `x` with the estimated residuals as input variable. The quantlet also draws 95% confidence intervals for the case of no autocorrelation.

Another issue is to check the normality of the residuals. This is commonly done by using the Bera-Jarque test suggested by Bera and Jarque (1982). It is commonly called JB-test and can be computed with the quantlet `jarber` which is called by

```
{jb, probjb, sk, k} = jarber(resid, printout)
```

with input variables

```
resid
     $(T - 1) \times 1$  matrix of residuals,
```

```
printout
    scalar, 0 no printout, 1 printout,
```

and output variables

```
jb
    scalar, test statistic of Jarque-Bera test,
```

```
probjb
    scalar, probability value of test statistics,
```

sk  
 scalar, skewness,

k  
 scalar, kurtosis.

In the following quantlet these diagnostics are applied to the residuals of the NAR(1) model fitted to the lynx data using the Nadaraya-Watson estimator (4) with the cross-validation bandwidth  $\hat{h} = 1.12085$

```

;               load required quantlets
  library("smoother")
  library("plot")
  func("acfplot")
  func("jarber")
  setsize(640,480)
;               data preparation
  lynx         = read("lynx.dat")
  lynxrows    = rows(lynx)
  lag1        = lynx[1:lynxrows-1]      ; vector of first lag
  y           = lynx[2:lynxrows]       ; vector of dep. var.
  data        = lag1~y
  data        = log(data)
  datain      = data~#(1:lynxrows-1)    ; add index to data
  dataso      = sort(datain,1)         ; sorted data
;               estimation
  h           = 1.12085                 ; Cross-validation bandwidth
  mhlp        = regxest(dataso[,1|2],h) ; local constant estimation
;               graphics
  mhlp        = setmask(mhlp,"line","red")
  xy          = setmask(data,"cross","small")
  plot(xy,mhlp)
  setgopt(plotdisplay,1,1,"title",
          "Estimated NAR(1) mean function")
  setgopt(plotdisplay,1,1,"xlabel","First Lag","ylabel","Lynx")
;               diagnostics
  yhatso      = mhlp.data[,2]~dataso[,3] ; sorted est. fct. values
  yhat        = sort(yhatso,2)          ; undo sorting
  eps         = data[,2] - yhat[,1]     ; compute residuals

```



```

acfplot(eps)           ; plot autocorrelation function of res.
setgopt(dacf,1,1,"title","Autocorrelation function of NAR(1)
                                residuals")
;
{jb,probjb,sk,k} = jarber(eps,1)
                    ; compute Jarque-Bera test for normality of residuals
                                flts05.xpl

```

The plot of the resulting autocorrelation function of the residuals is shown in Figure 5. It clearly shows that the residuals are not white noise. This indicates that one should use a higher order nonlinear autoregressive process for modelling the dynamics of the lynx data. This will be discussed in Section 2. Moreover, normality is rejected even at the 1% significance level since the JB-test statistic is 11.779 which implies a  $p$ -value of 0.003.

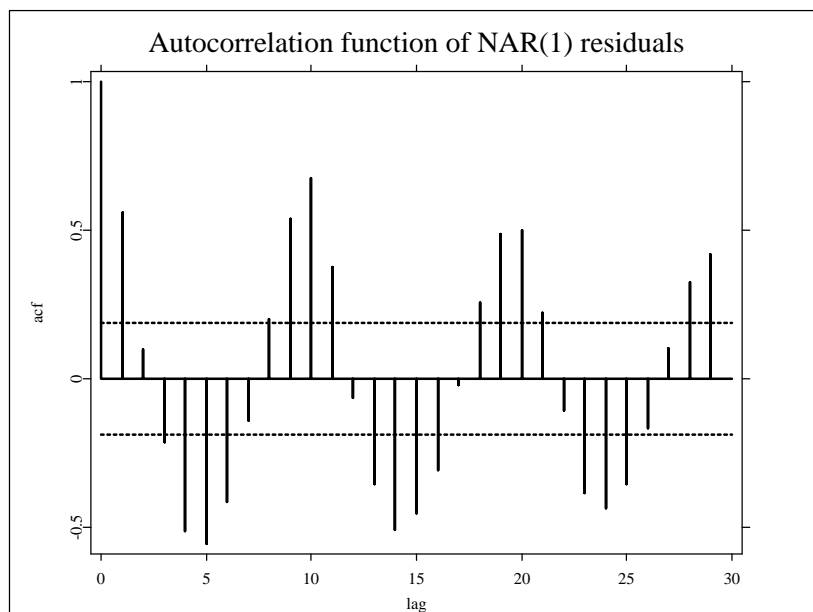


Figure 5: Autocorrelation function of estimated residuals based on a NAR(1) model for the lynx data

## 1.4 Confidence Intervals

```
{mh, clo, cup} = regxci(x{, h, alpha, K, xv})
  computes pointwise confidence intervals with prespecified confidence level for univariate regression using the Nadaraya-Watson estimator.

{mh, clo, cup} = regci(x{, h, alpha, K, d})
  computes pointwise confidence intervals with prespecified confidence level for univariate regression using the Nadaraya-Watson estimator. The computation uses WARPing.
```

Once one selected the bandwidth and checked the residuals one often wants to investigate the variance of estimating the autoregression function. Under appropriate conditions, the variance of both the Nadaraya-Watson and the local linear estimator can be approximated by

$$\text{Var}(\hat{f}(y, h)) \approx \frac{1}{Th} \frac{\sigma^2(y)}{\mu(y)} \|K\|_2^2 \quad (12)$$

as will be seen in Subsection 2.1. (12) can be used for constructing confidence intervals for  $\hat{f}(\cdot)$  since one can estimate the conditional variance  $\sigma^2(y)$  by the kernel estimate

$$\hat{\sigma}^2(y, h) = \frac{\sum_{t=2}^T K_h(Y_{t-1} - y) Y_t^2}{\sum_{t=2}^T K_h(Y_{t-1} - y)} - \hat{f}(y, h) \quad (13)$$

and the density  $\mu(y)$  by the kernel estimate

$$\hat{\mu}(y, h) = \sum_{t=1}^T K_h(Y_t - y). \quad (14)$$

Based on these estimates the quantlet `regxci` computes pointwise confidence intervals using the Nadaraya-Watson estimator. It is called with

```
{mh, clo, cup} = regxci(x{, h, alpha, K, xv})
```

with input variables:

**x**  $(T - 1) \times 2$  matrix of the data with the independent and the dependent variable in the first and second column, respectively,

**h** scalar, bandwidth for which if not given 20% of the range of the values in the first column **x** is used,

**alpha** confidence level with 0.05 as default value,

**K** string, kernel function on  $[-1, 1]$  and the quartic kernel "qua" as default,

**xv**  $m \times 1$  matrix of the values of the independent variable on which to compute the regression and **x** as default.

The output variables are:

**mh**  $(T - 1) \times 2$  or  $m \times 2$  matrix, the first column is the sorted first column of **x** or the sorted **xv**, the second column contains the regression estimate on the values of the first column,

**clo**  $(T - 1) \times 2$  or  $m \times 2$  matrix, the first column is the sorted first column of **x** or the sorted **xv**, the second column contains the lower confidence bounds on the values of the first column,

**cup**  $(T - 1) \times 2$  or  $m \times 2$  matrix, the first column is the sorted first column of **x** or the sorted **xv**, the second column contains the upper confidence bounds on the values of the first column.

If the WARPing technique is required, one uses the quantlet `regci`.

In Subsection 1.3 we found that the NAR(1) model for the lynx data is misspecified. Therefore, it is not appropriate for illustrating the computation of pointwise confidence intervals. Instead we will use a simulated time series. The quantlet below generates 150 observations of a stationary exponential AR(1)

process

$$Y_t = 0.3Y_{t-1} + 2.2Y_{t-1} \exp(-0.1Y_{t-1}^2) + \xi_t, \quad \xi_t \sim N(0, 1), \quad (15)$$

calls the interactive quantlet `regxbwsel` for bandwidth selection where one has to choose for the first time cross-validation and for the second time stop, computes the confidence intervals and plots the true and estimated function (solid and dashed line) as well as the pointwise confidence intervals (dotted line) as shown in Figure 6.

```
library("smoother")
library("plot")
library("times")
setsize(640,480)

;          generate exponential AR(1) process
phi1      = 0.3
phi2      = 2.2
g         = 0.1
randomize(0)
x         = genexpar(1,g,phi1,phi1+phi2,normal(150))


;          data preparation
xrows    = rows(x)
lag1     = x[1:xrows-1]          ; vector of first lag
y        = x[2:xrows]           ; vector of dep. var.
data     = lag1~y

;          true function
f        = sort(lag1^(phi1*lag1 + phi2*lag1.*exp(-g*lag1^2)),1)

;          estimation
{hcrit,crit} = regxbwsel(data)
{mh, clo, cup} = regxci(data,hcrit)

f        = setmask(f,"line","solid","red")
data     = setmask(data,"cross")
mh       = setmask(mh,"line","dashed","blue")
clo      = setmask(clo,"line","blue","thin","dotted")
cup      = setmask(cup,"line","blue","thin","dotted")
```

```
plot(data,f,mh,clo,cup)
setgopt(plotdisplay,1,1,"title","Confidence intervals of
      estimated NAR(1) mean function")
setgopt(plotdisplay,1,1,"xlabel","First Lag","ylabel","Y")
```

 flts06.xpl

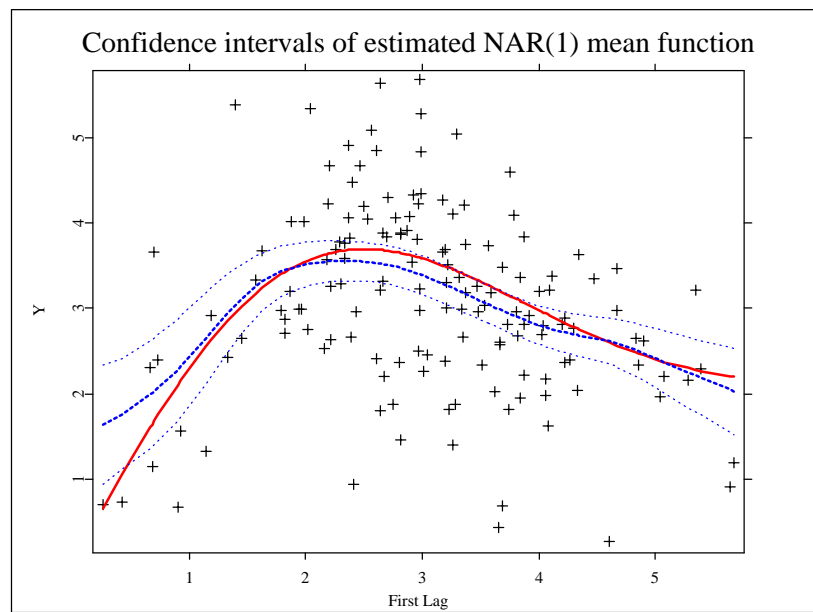


Figure 6: True and estimated mean function plus pointwise confidence intervals for a generated exponential AR(1) process

## 1.5 Derivative Estimation

```

mh = lpderxest(x, h{, q, p, K, v})
      estimates the q-th derivative of a regression function using local
      polynomial kernel regression with quartic kernel.

mh = lpderest(x, h{, q, p, K, d})
      estimates the q-th derivative of a autoregression function using
      local polynomial kernel regression. The computation uses WARP-
      ing.

```

When investigating the properties of a conditional mean function, one is often interested in its derivatives. The estimation of derivatives can be accomplished by using local polynomial estimation as long as the order  $p$  of the polynomial is at least as large as the order  $q$  of the derivative to be estimated. Using a local quadratic estimator

$$\{\hat{c}_0, \hat{c}_1, \hat{c}_2\}$$

$$= \underset{\{c_0, c_1, c_2\}}{\operatorname{arg\,min}} \sum_{t=1}^T \{Y_t - c_0 - c_1(Y_{t-1} - y) - c_2(Y_{t-1} - y)^2\}^2 K_h(Y_{t-1} - y)$$

one estimates the first and second derivative of  $f(y)$  at  $y$  with

$$\hat{f}'(y, h) = \hat{c}_1, \quad \hat{f}''(y, h) = 2\hat{c}_2.$$

In general, one uses a  $q+1$  instead of a  $q$ -th order polynomial for the estimation of the  $q$ -th derivative since this reduces the complexity of the estimation bias, see e.g. Fan and Gijbels (1995). The estimated derivative is then obtained as  $\hat{f}^{(q)} = q!\hat{c}_q$ . The quantlet `lpderxest` allows to estimate first and second order derivatives where maximally a second order polynomial is used. It is called by

```
mh = lpderxest(x, h{, q, p, K, v})
```

with input variables

**x**

$(T-1) \times 2$  matrix of the data with the independent and dependent variable in the first and second column, respectively.

**h** scalar, bandwidth for which if not given the rule-of-thumb bandwidth is computed with `lpderrot`,  
**q** integer  $\leq 2$ , order of derivative for which if not given,  $q=1$  (first derivative) is chosen,  
**p** integer, order of polynomial for which if not given,  $p=q + 1$  is used for  $q < 2$  and  $p=q$  is used for  $q=2$ ,  
**v**  $m \times 1$ , values of the independent variable on which to compute the regression for which if not given, `x` is used.

The output variable is

**mh**  $(T - 1) \times 2$  or  $m \times 2$  matrix where the first column is the sorted first column of `x` or the sorted `v` and the second column contains the derivative estimate on the values of the first column.

The quantlet `lpderest` which applies the WARPing technique (Fan and Marron 1994) allows for  $p \leq 5$  and  $q \leq 4$ . We note, however, that WARPing may waste a lot of information. Bandwidth selection remains an important issue and can be done using the quantlet `lpderrot`.

In the following quantlet we estimate the first and second derivatives of the conditional mean function of the exponential AR(1) process (15) based on 150 observations. The true derivatives (solid lines) and their estimates (dashed lines) are shown in Figures 7 and 8.

```

library("smoother")
library("plot")
library("times")
setsize(640,480)
; generate exponential AR(1) process
phi1 = 0.3
phi2 = 2.2
g = 0.1

```

```


randomize(0)
x      = genexpar(1,g,phi1,phi1+phi2,normal(150))

;
                                data preparation
xrows  = rows(x)
lag1   = x[1:xrows-1]           ; vector of first lag
y      = x[2:xrows]           ; vector of dep. var.
data   = lag1~y
ffder  = sort(lag1^(phi1 + exp(-g*lag1^2).*
                                phi2.*(1-2.*g.*lag1^2)),1)
fsder  = sort(lag1^(exp(-g*lag1^2).*(-2*g.*lag1)*
                                phi2.*(3-2.*g.*lag1^2)),1)

;
                                estimate first derivative
ffder  = setmask(ffder,"line","solid","red")
mhfder = lpderxest(data)
mhfder = setmask(mhfder, "line","dashed","blue")
plotder = createdisplay(1,1)
show(plotder,1,1,ffder,mhfder)
setgopt(plotder,1,1,"title","Estimated first derivative
                                of mean function")
setgopt(plotder,1,1,"xlabel","First lag","ylabel",
                                "First derivative")

;
                                estimate second derivative
fsder  = setmask(fsder,"line","solid","red")
hrot   = 2*lpderrot(data,2)
mhsder = lpderxest(data,hrot,2)
mhsder = setmask(mhsder, "line","dashed","blue")
plot(fsder,mhsder)
setgopt(plotdisplay,1,1,"title","Estimated second
                                derivative of mean function")
setgopt(plotdisplay,1,1,"xlabel","First lag","ylabel",
                                "Second derivative")

```

 flts07.xpl



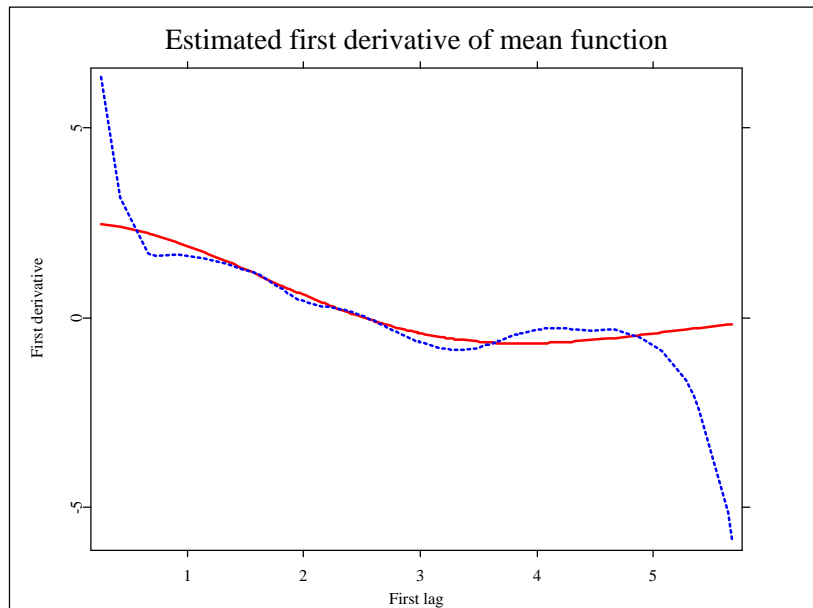


Figure 7: True and estimated first derivative for a generated exponential AR(1) process

## 2 Nonlinear Autoregressive Models of Higher Order

In Subsection 1.3 we briefly discussed diagnostics to check for the correct specification of a time series model. There we found for the lynx data set that the nonlinear autoregressive model of order one (2) is of too low order to capture the linear correlation in the data. For practical flexible time series modelling it is therefore necessary to allow for higher order nonlinear autoregressive models (1). Their estimation and the selection of relevant lags will be discussed in this section. To simplify notation, we introduce the vector of lagged variables  $X_t = (Y_{i_1}, Y_{i_2}, \dots, Y_{i_m})^T$  such that (1) can be written as

$$Y_t = f(X_t) + \sigma(X_t)\xi_t \quad (16)$$

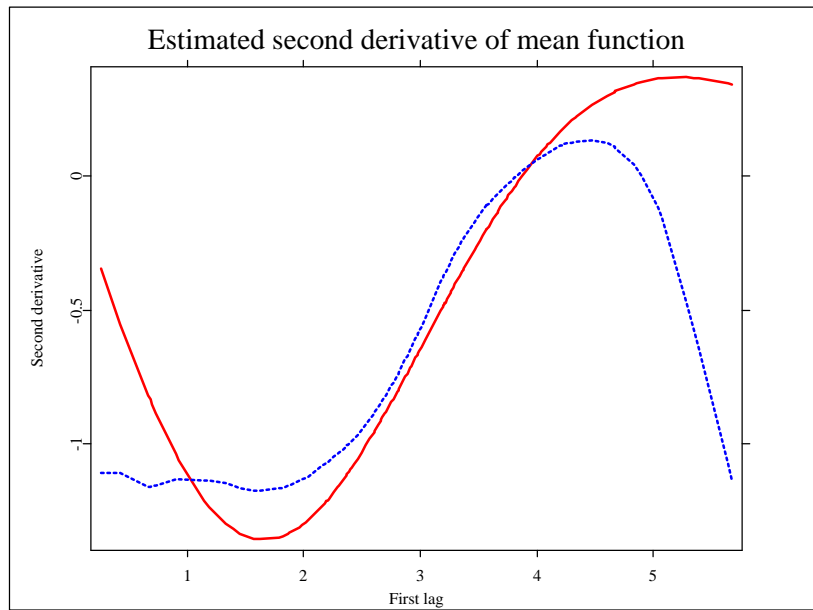


Figure 8: True and estimated second derivative a generated exponential AR(1) process

## 2.1 Estimation of the Conditional Mean

```

mh = regxestp(x{, h, K, v})
    computes the Nadaraya-Watson estimator for multivariate au-
    toregression.

mh = regestp(x{, h, K, d})
    Nadaraya-Watson estimator for multivariate regression. The
    computation uses WARPing.

```

```

mh = lpregxestp(x{, h, K, v})
      estimates a multivariate regression function using local polynomial
      kernel regression with quartic kernel.

mh = lpregestp(x{, h, K, d})
      estimates a multivariate regression function using local polynomial
      kernel regression. The computation uses WARPing.

{mA, gsqA, denA, err} = fvllc(Xsj, Yorig, h, Xtj,
                             kernreg, lorq, fandg, loo)
      estimates a multivariate regression function using local linear re-
      gression with Gaussian kernel.

```

It is not difficult to extend the Nadaraya-Watson estimator (4) and local linear estimator (5) to several lags in the conditional mean function  $f(\cdot)$ . One then simply uses Taylor expansions of order  $p$  for several variables. In the weighted minimization problem of the local constant estimator (3) one has to extend the kernel function  $K_h(\cdot)$  for several lagged variables. The simplest way of doing this is to use a product kernel

$$K_h(X_t - x) = \prod_{j=1}^m h_j^{-1} K\left(\frac{X_{t,j} - x_j}{h_j}\right) \quad (17)$$

where one  $h = (h_1, h_2, \dots, h_m)^T$  is a vector of bandwidths for each lag or variable. Of course, one may also use the same bandwidth  $h = (h, h, \dots, h)^T$  for all lags in which case we write  $K_h(X_t - x)$ . Using a scalar bandwidth, (3) becomes

$$\hat{c}_0 = \arg \min_{\{c_0\}} \sum_{t=i_m+1}^T \{Y_t - c_0\}^2 K_h(X_t - x) \quad (18)$$

and the Nadaraya-Watson estimator is given by

$$\hat{f}_1(x, h) = \hat{c}_0 = \frac{\sum_{t=i_m+1}^T K_h(X_t - x) Y_t}{\sum_{t=i_m+1}^T K_h(X_t - x)}. \quad (19)$$

Note that from now on we indicate the Nadaraya-Watson estimator and local linear estimator by the indices 1 and 2, respectively.

The local linear estimator with  $p = 1$  is derived from the weighted minimization

$$\{\hat{c}_0, \hat{c}_1\} = \arg \min_{\{c_0, c_1\}} \sum_{t=i_m+1}^T \{Y_t - c_0 - c_1(X_t - x)\}^2 K_h(X_t - x). \quad (20)$$

Using the notation

$$Z_2 = \begin{pmatrix} 1 & \cdots & 1 \\ X_{i_m+1} - x & \cdots & X_T - x \end{pmatrix}^T, \quad Y = (Y_{i_m+1}, \dots, Y_T)^T$$

$$e = (1, 0_{1 \times m})^T, \quad W = \text{diag} \left\{ \frac{1}{T - i_m} K_h(X_t - x) \right\}_{t=i_m+1}^T,$$

the estimate  $\hat{f}_2(x, h) = \hat{c}_0$  can be written for any  $x \in \mathbb{R}^m$  as

$$\hat{f}_2(x) = e^T (Z_2^T W Z_2)^{-1} Z_2^T W Y. \quad (21)$$

Under suitable conditions which are listed in Subsection 2.2 the Nadaraya-Watson estimator (19) and local linear estimator (21) have an asymptotic normal distribution

$$T^{2/(4+m)} \left( \hat{f}_a(x, h) - f(x) \right) \rightarrow N \left( \beta^2 \frac{\sigma_K^2}{2} r_a, \beta^{-m} \frac{\sigma(x)}{\mu(x)} \|K\|_2^2 \right), \quad a = 1, 2 \quad (22)$$

where

$$r_1(x) = \text{Tr} \{ \nabla^2 f(x) \} + 2 \nabla^T \mu(x) \nabla f(x) / \mu(x), \quad r_2(x) = \text{Tr} \{ \nabla^2 f(x) \}. \quad (23)$$

Thus, the rate of convergence deteriorates with the number of lags. This feature is commonly called the ‘curse of dimensionality’ and often viewed as a substantial drawback of nonparametric methods. One should keep in mind, however, that the  $\sqrt{T}$ -rate of parametric models only holds if one estimates a model with an a priori chosen finite number of parameters which may imply a large estimation bias in case of misspecified models. If, however, one allows the number of parameters of parametric models to grow with sample size,  $\sqrt{T}$ -convergence may no longer hold.

The quantlets `regxestp` and `lregxestp` compute the Nadaraya-Watson estimator (19) and local linear estimator (21) for higher order autoregressions. They are called by

`mh = regxestp(x{, h, K, v})`

or

`mh = lregxestp(x{, h, K, v})`

with input variables

**x**

$(T - i_m) \times (m + 1)$  matrix of the data with the  $m$  lagged variables in the first  $m$  columns and the dependent variable in the last column,

**h**

scalar or  $m \times 1$  or  $1 \times m$  vector of bandwidth for which if not given 20% of the range of the values in the first column of **x** is used,

**K**

string, kernel function on  $[-1,1]$  or Gaussian kernel "gau" for which if not given, the quartic kernel "qua" is used,

**v**

$n \times m$  matrix of values of the independent variable on which to compute the regression for which if not given, a grid of length 100 ( $m = 1$ ), length 30 ( $m = 2$ ) and length 8 ( $m = 3$ ) is used in case of  $m < 4$ . When  $m \geq 4$  then **v** is set to **x**.

The output variable is a

**mh**

$(T - i_m) \times (m + 1)$  or  $n \times (m + 1)$  matrix where the first  $m$  columns contain the grid or the sorted first  $m$  columns of **x**, the  $m + 1$  column contains the regression estimate on the values of the first  $m$  columns.

As before, there are also quantlets which apply WARPing. They are called `regestp` and `lregestp`, respectively.

Since we found in Subsection 1.3 that a NAR(1) model is not sufficient to capture the dynamics of the lynx trappings, we compute and plot in the following quantlet the autoregression function for lag 1 and 2 for both estimators using the crude bandwidth of 20% of the data range. Note that you have to click on the graph and rotate it in order to see the regression surface.

```


library("smoother")
library("plot")
setsize(640,480)

;                               data preparation
lynx      = read("lynx.dat")
lynxrows  = rows(lynx)
lag1      = lynx[1:lynxrows-2]   ; vector of first lag
lag2      = lynx[2:lynxrows-1]   ; vector of second lag
y         = lynx[3:lynxrows]     ; vector of dep. var.
data      = lag1~lag2~y
data      = log(data)

;                               estimation
h         = 0.2*(max(data[,1])-min(data[,1])) ; crude bandwidth
mh        = regxestp(data,h)       ; local constant estimation
mhlp      = lregxestp(data,h)     ; local constant estimation

;                               graphics
mhplot    = createdisplay(1,1)
mh        = setmask(mh,"surface","blue")
show(mhplot,1,1,data,mh)         ; surface plot
setgopt(mhplot,1,1,"title",
        "Nadaraya-Watson estimate -- ROTATE!")
mhlpplot  = createdisplay(1,1)
mhlp      = setmask(mhlp,"surface","red")
show(mhlpplot,1,1,data,mhlp)    ; surface plot
setgopt(mhlpplot,1,1,"title",
        "Local linear estimate -- ROTATE!")

```

 flts08.xpl

Figures 9 and 10 show three-dimensional plots of the observations and the estimated regression function. In Figure 9 one can clearly see the problem of boundary effects, i.e. in regions where there are no or only few data points the estimated function values may easily become erratic if the bandwidth is too small. Therefore, a selected bandwidth may be appropriate for regions with plenty of observations while inappropriate elsewhere. As can be seen from Figure 10, this boundary problem turns out to be worse for the local linear estimator where

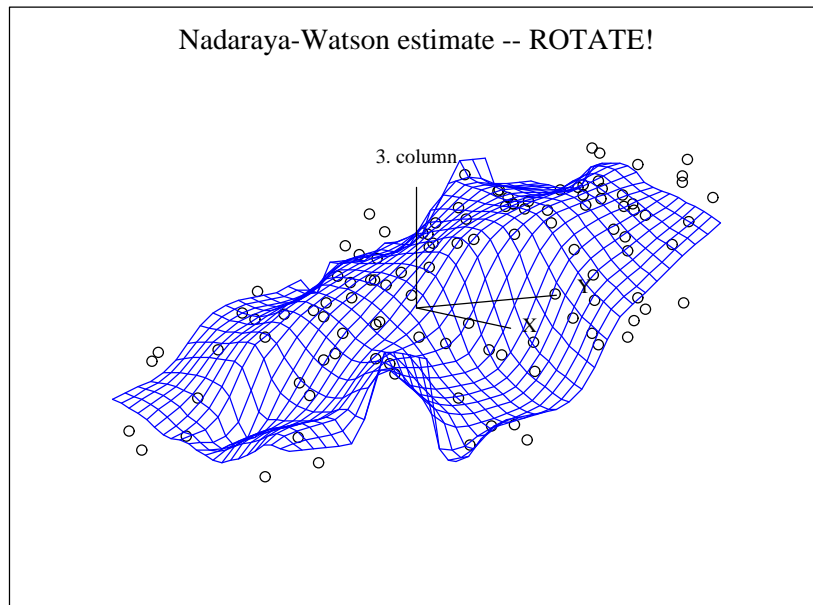


Figure 9: Observations and Nadaraya-Watson estimate of NAR(2) regression function for the lynx data

one observes a large outlier for one grid point. Such terrible estimates happen if the inversion in (20) is imprecise due to a too small bandwidth. One then has to increase the bandwidth. Try the quantlet `f1ts08.xpl` with replacing in the crude bandwidth choice the factor 0.2 by 2. Note that increasing the bandwidth makes the estimated regression surfaces of the two estimators look flat and closer to linearity, respectively. This, however, can increase the estimation bias. Therefore, an appropriate bandwidth choice is important. It will be discussed in the next section.

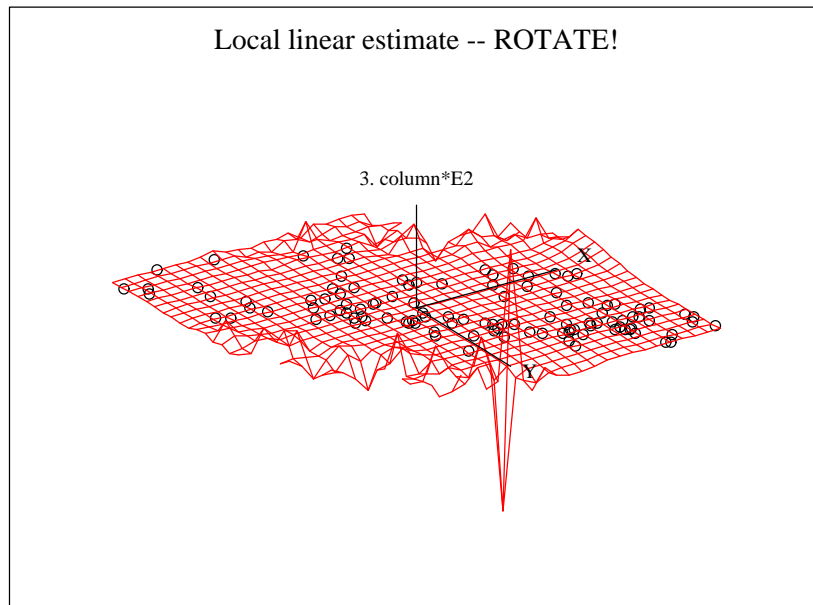


Figure 10: Observations and local linear estimate of NAR(2) regression function for the lynx data

## 2.2 Bandwidth and Lag Selection

```
{Bhat, Bhatr, hB, Chat, sumwc, hC, hA} = hoptest (xsj,
  yorig, xtj, estimator, kernel, ntotal, sigy2, perB,
  lagmax, robden)
quantlet to compute plug-in bandwidth for multivariate regres-
sion or nonlinear autoregressive processes of higher order.
```



```

{crmin, crpro} = cafpe(y, truedat, xdataIn, xdatadif,
                    xdatastand, lagmax, searchmethod, dmax)
quantlet for local linear lag selection for the conditional mean
function based on the Asymptotic Final Prediction Error
(AFPE2) or its corrected versions (CAFPE2) using default set-
tings.

{crmin, crpro, crstore, crstoreadd, hstore, hstoretest}
= cafpefull(y, truedat, xresid, trueres, xdataIn,
            xdatadif, xdatastand, lagmax, volat, searchmethod,
            dmax, selcrit, robden, perA, perB, startval,
            noutputf, outputpath)
quantlet for local linear lag selection for the conditional mean or
volatility function based on the asymptotic final prediction error
(AFPE2) or its corrected version (CAFPE2).

{mA, gsqA, denA, err} = fvllc(Xsj, Yorig, h, Xtj,
                             kernreg, lorq, fandg, loo)
can estimate the multivariate regression function, first or second
direct derivatives using local linear or partial local quadratic re-
gression with Gaussian kernel.

```

The example of the previous section showed that the bandwidth choice is very important for higher order autoregressive models. Equally important is the selection of the relevant lags. Both will be discussed in this section. The presented procedures are based on Tschernig and Yang (2000). We start with the problem of selecting the relevant lags. For this step it is necessary a priori specify a set of possible lag vectors by choosing the maximal lag  $M$ . Denote the full lag vector containing all lags up to  $M$  by  $X_{t,M} = (Y_{t-1}, Y_{t-2}, \dots, Y_{t-M})^T$ . The lag selection task is now to eliminate from the full lag vector  $X_{t,M}$  all lags that are redundant. Let us first state the assumptions that Tschernig and Yang (2000) require:

- (A1) For some  $M \geq i_m$  the vector process  $X_{t,M}$  is strictly stationary and  $\beta$ -mixing with  $\beta(T) \leq k_0 T^{-(2+\delta)/\delta}$  for some  $\delta > 0$ ,  $k_0 > 0$ .
- (A2) The stationary distribution of the process  $X_{t,M}$  has a continuous density  $\mu_M(x_M)$ ,  $x_M \in \mathbb{R}^M$ . Note that  $\mu(\cdot)$  is used for denoting  $\mu_M(\cdot)$  and all of its marginal densities.

- (A3) The function  $f(\cdot)$  is twice continuously differentiable while  $\sigma(\cdot)$  is continuous and positive on the support of  $\mu(\cdot)$ .
- (A4) The errors  $\{\xi_t\}_{t \geq i_m}$  have a finite fourth moment  $m_4$ .
- (A5) The support of the weight function  $w(\cdot)$  is compact with nonempty interior. The function  $w(\cdot)$  is continuous, nonnegative and  $\mu(x_M) > 0$  for  $x_M$  in the support of  $w(\cdot)$ .
- (A6) The kernel function  $K : \mathbb{R} \rightarrow \mathbb{R}$  is a symmetric probability density and the bandwidth  $h$  is a positive number with  $h \rightarrow 0$ ,  $nh^m \rightarrow \infty$  as  $n \rightarrow \infty$ .

For the definition of  $\beta$ -mixing see Section 1.1 or Doukhan (1994). Conditions (A1) and (A2) can be checked using e.g. Doukhan (1994, Theorem 7 and Remark 7, pp. 102, 103). Further conditions can be found in Lu (1998).

For comparing the quality of competing lag specifications, one needs an appropriate measure of fit, as for example the **final prediction error** (FPE)

$$FPE_a(h, i_1, \dots, i_m) = E \left[ \left( \check{Y}_t - \hat{f}_a(\check{X}_t, h) \right)^2 w(\check{X}_{t,M}) \right], \quad a = 1, 2. \quad (24)$$

In the definition of the  $FPE(\cdot)$  the process  $\{\check{Y}_t\}$  is assumed to be independent of the process  $\{Y_t\}$  but to have the same stochastic properties. If we now indicate the vector of lagged values of the data generating process by the superscript  $*$  and assume its largest lag is smaller than the chosen  $M$ , we can easily relate the definition of the FPE (24) to the MISE

$$d_{a,M}(h, i_1, \dots, i_m) = E \left[ \int \left\{ f(x^*) - \hat{f}_a(x) \right\}^2 w(x_M) \mu(x_M) dx_M \right], \quad (25)$$

which here extends (11) to functions with several lags. First note that

$$\begin{aligned} FPE_a(h, i_1, \dots, i_m) &= E \left\{ E \left[ \left( \check{Y}_t - \hat{f}_a(\check{X}_t, h) \right)^2 w(\check{X}_{t,M}) \mid Y_1, \dots, Y_T \right] \right\} \\ &= E \left\{ \int \left( y - \hat{f}_a(x) \right)^2 w(x_M) \mu(y, x_M) dy dx_M \right\}. \end{aligned}$$

Using  $\left\{ y - \hat{f}_a(x) \right\}^2 = \left\{ y - f(x)^* + f(x)^* - \hat{f}_a(x) \right\}^2$  one obtains the decomposition

$$FPE_a(h, i_1, \dots, i_m) = A + d_{a,M}(h, i_1, \dots, i_m), \quad (26)$$

where

$$A = \int \sigma^2(x^*)w(x_M)\mu(x_M)dx_M \quad (27)$$

denotes the mean variance or final prediction error for the true function  $f(x^*)$ . Therefore, it follows from (26) that the FPE measures the sum of the mean variance and the MISE.

In the literature mainly two approaches were suggested for estimating the unknown  $FPE_a(\cdot)$  or variants thereof, namely cross-validation (Vieu 1994), (Yao and Tong 1994) or estimation of an asymptotic expression of the  $FPE_a(\cdot)$  (Auestad and Tjøstheim 1990), (Tjøstheim and Auestad 1994), (Tschernig and Yang 2000). Given Assumptions (A1) to (A6), Tschernig and Yang (2000, Theorem 2.1) showed that for the local constant estimator,  $a = 1$ , and the local linear estimator,  $a = 2$ , one has  $FPE_a(h, i_1, \dots, i_m) = AFPE_a(h, i_1, \dots, i_m) + o\{h^4 + (T - i_m)^{-1}h^{-m}\}$  where

$$AFPE_a(h, i_1, \dots, i_m) = A + b(h)B + c(h)C_a \quad (28)$$

denotes the **asymptotic final prediction error**. The terms  $b(h)B$  and  $c(h)C$  denote the expected variance and squared bias of the estimator, respectively, with the constants

$$B = \int \sigma^2(x^*)w(x_M)\mu(x_M)/\mu(x)dx_M, \quad (29)$$

$$C_a = \int r_a(x)^2w(x_M)\mu(x_M)dx_M \quad (30)$$

and the variable terms

$$b(h) = \|K\|_2^{2m}(T - i_m)h^{-m}, \quad c(h) = \sigma_K^4 h^4/4 \quad (31)$$

with  $\|K\|_2^2 = \int K(u)^2 du$  and  $\sigma_K^2 = \int K(u)u^2 du$ . The sum of the expected variance and squared bias of the estimator just represents the asymptotic mean squared error. Note that if the vector of correct lags  $X_t^*$  is included in  $X_t$ , then  $AFPE_a(h, \cdot)$  tends to  $A$  as both  $b(h)B$  and  $c(h)C_a$  tend to zero.

From (28) it is possible to determine the asymptotically optimal bandwidth  $h_{opt}$  by minimizing the asymptotic MISE, i.e. solving the variance-bias tradeoff between  $b(h)B$  and  $c(h)C$ . The asymptotically optimal bandwidth is given by

$$h_{a,opt} = \{m\|K\|_2^{2m}B(T - i_m)^{-1}C_a^{-1}\sigma_K^{-4}\}^{1/(m+4)}. \quad (32)$$

Note that for a finite asymptotically optimal bandwidth to exist one has to assume that

(A7)  $C_a$  defined in (30) is positive and finite.

This requirement implies that in case of local linear estimation there does not exist a finite  $h_{2,opt}$  for linear processes. This is because there does not exist an approximation bias and thus a larger bandwidth has no cost.

In order to obtain the plug-in bandwidth  $\hat{h}_{a,opt}$  one has to estimate the unknown constants  $B$  and  $C_a$ . A local linear estimate of  $B$  (29) is obtained from

$$\hat{B}_2(h_B) = T^{-1} \sum_{t=1}^T \left\{ Y_t - \hat{f}_2(X_t, h_B) \right\}^2 w(X_{t,M}) / \hat{\mu}(X_t, h_B),$$

where  $\hat{\mu}(\cdot)$  is the Gaussian kernel estimator (40) of the density  $\mu(x)$ . For estimating  $h_B$  one may use Silverman's (1986) rule-of-thumb bandwidth

$$\hat{h}_B = \hat{\sigma} \left( \frac{4}{T+2} \right)^{1/(m+4)} T^{-1/(m+4)} \quad (33)$$

with  $\hat{\sigma} = \left( \prod_{j=1}^m \sqrt{\text{Var}(X_j)} \right)^{1/m}$  denoting the geometric mean of the standard deviation of the regressors.

For the local linear estimator (21),  $C_2$  (30) can be consistently estimated by

$$\hat{C}_2(h_C) = \frac{1}{T} \sum_{t=i_m+1}^T \left[ \sum_{j=1}^m \hat{f}^{(jj)}(X_t, h_C) \right]^2 w(X_{t,M}), \quad (34)$$

where  $f^{(jj)}(\cdot)$  denotes the second direct derivative of the function  $f(\cdot)$ . It can be estimated using the partial local quadratic estimator

$$\{\hat{c}_0, \hat{c}_{11}, \dots, \hat{c}_{1m}, \hat{c}_{21}, \dots, \hat{c}_{2m}\} = \arg \min_{\{c_0, c_{11}, \dots, c_{1m}, c_{21}, \dots, c_{2m}\}} \quad (35)$$

$$\sum_{t=i_m+1}^T \left\{ Y_t - c_0 - c_{11}(X_{t1} - x_1) - \dots - c_{1m}(X_{tm} - x_m) - c_{21}(X_{t1} - x_1)^2 - \dots - c_{2m}(X_{tm} - x_m)^2 \right\}^2 K_h(X_t - x).$$

The estimates of the direct second derivatives are then given by  $\hat{f}^{(jj)}(x, h) = 2\hat{c}_{2j}$ ,  $j = 1, \dots, m$ . Excluding all cross terms has no asymptotic effects while keeping the increase in the 'parameters'  $c_0, c_{1j}, c_{2j}$ ,  $j = 1, \dots, m$  linear in the number of lags  $m$ . This approach is a simplification of the partial cubic

estimator proposed by Yang and Tschernig (1999) who also showed that the rule-of-thumb bandwidth

$$\hat{h}_C = 2\hat{\sigma} \left( \frac{4}{T+4} \right)^{1/(m+6)} T^{-1/(m+6)} \quad (36)$$

has the optimal rate. We note that for the estimation of  $C_1$  of the Nadaraya-Watson estimator one has additionally to estimate the derivative of the density as it occurs in (23). Therefore, we exclusively use the local linear estimator (21). The direct second derivatives  $f^{(jj)}(x)$  can be estimated with the quantlet `tp/capfe/fvllc`.

The plug-in bandwidth  $\hat{h}_{2,opt}$  is then given by

$$\hat{h}_{2,opt} = \left\{ m \|K\|_2^{2m} \hat{B}_2(\hat{h}_B) (T - i_m)^{-1} \hat{C}_2(\hat{h}_C)^{-1} \sigma_K^{-4} \right\}^{1/(m+4)}. \quad (37)$$

It now turns out that when taking into account the estimation bias of  $A$ , the local linear estimator of  $AFPE_2(h, \cdot)$  (28) becomes

$$AFPE_2 = \hat{A}_2(h_{2,opt}) + 2K(0)^m (T - i_m)^{-1} h_{2,opt}^{-m} \hat{B}_2(h_B) \quad (38)$$

and the expected squared bias of estimation drops out. In practice,  $h_{2,opt}$  is replaced by the plug-in bandwidth (37). Note that one can interpret the second term in (38) as a penalty term to punish overfitting or choosing superfluous lags. This penalty term decreases with sample size as  $h_{2,opt}$  is of order  $T^{-1/(m+4)}$ . The final prediction error for the true function  $A$  (27) is estimated by taking the sample average

$$\hat{A}_2(h) = T^{-1} \sum_{t=1}^T \left\{ y_t - \hat{f}_2(X_t, h) \right\}^2 w(X_{t,M})$$

of the residuals from the local linear estimator  $\hat{f}_2(X_t, h)$ . The asymptotic properties of the lag selection method rely on the fact that the argument of  $w(\cdot)$  is the full lag vector  $X_{t,M}$ .

In order to select the adequate lag vector, one computes (38) for all possible lag combinations with  $m \leq M$  and chooses the lag vector with the smallest  $AFPE_2$ . Given Assumptions (A1) to (A7) and a further technical condition, Tschernig and Yang (2000, Theorem 3.2) showed that this procedure is weakly consistent, i.e. the probability of choosing the correct lag vector if it is included in the set of lags considered approaches one with increasing sample size. This

consistency result may look surprising since the linear FPE is known to be inconsistent. However, in the present case the rate of the penalty term in (38) depends on the number of lags  $m$ . Thus, if one includes  $l$  lags in addition to  $m^*$  correct ones, the rate of the penalty term becomes slower which implies that too large models are ruled out asymptotically. Note that this feature is intrinsic to the local estimation approach since the number of lags influence the rate of convergence, see (22). We remark that the consistency result breaks down if Assumption (A7) is violated e.g. if the stochastic process is linear. In this case overfitting (including superfluous lags in addition to the correct ones) is more likely. The breakdown of consistency can be avoided if one uses the Nadaraya-Watson instead of the local linear estimator since the former is also biased in case of linear processes.

Furthermore, Tschernig and Yang (2000) show that asymptotically it is more likely to overfit than to underfit (miss some correct lags). In order to reduce overfitting and therefore increase correct fitting, they suggest to correct the AFPE and estimate the Corrected Asymptotic FPE

$$CAFPE_a = AFPE_a \left\{ 1 + m(T - i_m)^{-4/(m+4)} \right\}, \quad a = 1, 2. \quad (39)$$

The correction does not affect consistency under the stated assumptions while additional lags are punished more heavily in finite samples. One chooses the lag vector with the smallest  $CAFPE_a$ ,  $a = 1, 2$ .

We note that if one allows the maximal lag  $M$  to grow with sample size, then one has a doubled nonparametric problem of nonparametric function estimation and nonparametric lag selection.

The nonparametric lag selection criterion  $CAFPE_2$  can be computed using the quantlet `tp/cafpe/cafpe`. The quantlet `tp/cafpe/cafpefull` also allows to use  $AFPE_a$ . Both are part of the third party quantlib `tp/cafpe/cafpe` which contains various quantlets for lag and bandwidth selection for nonlinear autoregressive models (16). The quantlet `tp/cafpe/cafpe` is called as

```
{crmin, crpro} = cafpe(y, truedat, xdata1n, xdatadif,
                      xdatastand, lagmax, searchmethod, dmax)
```

with the input variables:

$y$   
 $T \times 1$  matrix of the observed time series or set to zero if `truedat` is used,

**truedat**  
character variable that contains path and name of ascii data file if  $y=0$ ,

**xdata1n**  
character variable where "yes" takes natural logs, "no" doesn't,

**xdatadif**  
character variable where the value "yes" takes first differences of data, "no" doesn't,

**xdatastand**  
character variable where "yes" standardizes data, "no" doesn't,

**lagmax**  
scalar variable, largest lag to be considered,

**searchmethod**  
character variable where "full" considers all possible lag combinations, "directed" does directed search (recommended if  $\text{lagmax} > 10$ ),

**dmax**  
scalar variable with maximum number of possible lags,

and output variables

**crmin**  
 $(\text{dmax}+1) \times 1$  vector that stores for all considered lag combinations in the first  $\text{dmax}$  columns the selected lag vector, in the  $\text{dmax}+1$  column the estimated  $CAFPE_2$ , in the  $\text{dmax}+2$  column  $\hat{A}$ , in the  $\text{dmax}+3$  column the bias corrected estimate of  $A$ , see TY (equation 3.3),

**crpro**  
 $(\text{dmax}+1) \times (\text{dmax}+6)$  matrix that stores for each number of lags  $(0, 1, \dots, \text{dmax})$  in the first  $\text{dmax}$  columns the selected lag vector, in the  $\text{dmax}+1$  column the plug-in bandwidth  $\hat{h}_{2,opt}$  for estimating the final prediction error for the true function  $A$  and  $CAFPE_2$ , in the  $\text{dmax}+2$  column the bandwidth  $\hat{h}_B$  for estimating the constant  $B$  which is used for computing  $CAFPE_2$  and the plug-in bandwidth  $\hat{h}_{2,opt}$ , in the  $\text{dmax}+3$  column the bandwidth  $\hat{h}_C$  for estimating the constant  $C$  which is used for computing the plug-in bandwidth  $\hat{h}_{2,opt}$ , in the  $\text{dmax}+4$  column the estimated  $CAFPE_2$ , in the  $\text{dmax}+5$  column  $\hat{A}$ , in the  $\text{dmax}+6$  column the bias corrected estimate of  $A$ , see TY (equation 3.3).

Some comments may be appropriate. The weight function  $w(\cdot)$  is the indicator function on the range of the observed data. If  $M$  is large or the time series is long, then conducting a full search over all possible lag combinations may take extraordinarily long. In this case, one should use the directed search suggested by Tjøstheim and Auestad (1994): lags are added as long as they reduce the selection criterion and one adds that lag from the remaining ones which delivers the largest reduction.

For computing  $CAFPE_2$  TY follow Tjøstheim and Auestad (1994) and implement two additional features for robustification. For estimating  $\mu(x, h)$  the kernel estimator

$$\hat{\mu}(x, h) = (T - i_m + i_1)^{-1} \sum_{i=i_m+1}^{T+i_1} K_h(X_i - x) \quad (40)$$

is used where the vectors  $X_i, i = T + 1, \dots, T + i_1$  are all available from the observations  $Y_t, t = 1, \dots, T$ . For example,  $X_{T+i_1}$  is given by  $(Y_T, \dots, Y_{T+i_1-i_m})^T$ . This robustification is switched off if the sum stops at  $T$ . Furthermore, 5% of those observations whose density values  $\hat{\mu}(\cdot)$  are the lowest, are screened off. These features can be easily switched off or modified in the quantlet `tp/cafpe/cafpefull`. This quantlet also allows to select the lags of the conditional standard deviation  $\sigma(\cdot)$  and is therefore discussed in detail in Subsection 2.4.

If one is only interested in computing the plug-in bandwidth  $\hat{h}_{2,opt}$ , then one can directly use the quantlet `tp/cafpe/hoptest`. However, before it can be called it requires to prepare the time series accordingly so that it is easier to run the lag selection which automatically delivers the plug-in bandwidth for the chosen lag vector as well. For the definition of its variables the reader is referred to the helpfile of `tp/cafpe/hoptest`.

We are now ready to run the quantlet `tp/cafpe/cafpe` on the lynx data set. The following quantlet conducts a full search among the first six lags

```

pathcafpe      = "tp/cafpe/" ; path of CAFPE quantlets
;      load required quantlibs
library("xplore")
library("times")
func(pathcafpe + "cafpe_load") ; load XploRe files of CAFPE
cafpe_load(pathcafpe)

setenv("outheadline","") ; no header for each output file


```



```

setenv("outlineno","") ; no numbering of output lines
; set parameters
truedat = "lynx.dat" ; name of data file
y = 0
xdataln = "yes"; ; take logarithms
xdatadif = "no"; ; don't take first differences
xdatastand = "no"; ; don't standardize data
lagmax = 6 ; the largest lag considered is 6
searchmethod = "full" ; consider all possible lag comb.
dmax = 6 ; consider at most 6 lags
; conduct lag selection
{ crmin,crpro } = cafpe(y,truedat,xdataln,xdatadif,xdatastand,
lagmax,searchmethod,dmax)
"selected lag vector, estimated CAFPE "
crmin[,1:dmax+1]
"number of lags, chosen lag vector, estimated CAFPE,
plug-in bandwidth"
(0:dmax)~crpro[,1:dmax|(dmax+4)|(dmax+1)]

```

 flts09.xpl

A screenshot of the output which shows the criteria for all other number of lags is contained in Figure 11. The selected lags are 1 to 4 with plug-in bandwidth  $\hat{h}_{2,opt} = 0.90975$  and  $CAFPE_2 = 0.2163$ . However, the largest decrease in  $CAFPE_2$  occurs if one allows for two lags instead of one and lag 2 is added. In this case,  $CAFPE_2$  drops from 0.64125 to 0.24936. Therefore lag 2 seems to capture the autocorrelation in the residuals of the NAR(1) model which was estimated in Subsections 1.1 to 1.3. For this reason a NAR(2) model could be sufficient for the lynx data. Its graphical representation is discussed in the next section.

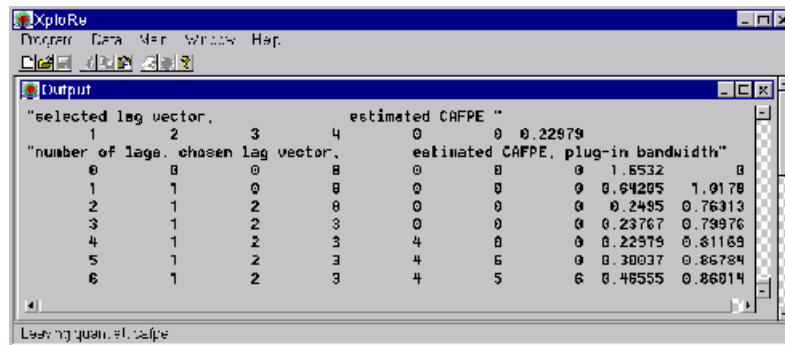


Figure 11: Results of the lag selection procedure using  $CAFPE_2$  for lynx data

### 2.3 Plotting and Diagnostics

```
{hplugin, hB, hC, xs, resid} = plotloclin(xdata, xresid,
    xdataIn, xdataDif, xdataStand, volat, lags, h,
    xsconst, gridnum, gridmax, gridmin)
computes 1- or 2-dimensional plot of regression function of a non-
linear autoregressive process for a given lag vector on the range
of the data; if more than 2 lags are used, then only two lags are
allowed to vary, the others have to be fixed
```

Once the relevant lags and an appropriate bandwidth are determined, one would like to have a closer look at the implied conditional mean function as well as checking the residuals for potential model misspecification as discussed in Subsection 1.3. The latter may be done by inspecting the autocorrelation function and testing the normality of the residuals. The quantlet `tp/caufe/plotloclin` of the quantlib `tp/caufe/caufe` allows to do both. It generates two- or three-dimensional plots of the autoregression function on a grid that covers the range of data and computes the residuals for the given time series. Both is done either with a bandwidth specified by the user or the plug-in bandwidth  $\hat{h}_{2,opt}$  which is automatically computed if required. The quantlet `tp/caufe/plotloclin` also allows to compute three-dimensional plots of functions with more than two lags by keeping  $m - 2$  lags fixed at user-selected values. It is called by

```
{hplugin,hB,hC,xs,resid} = plotloclin(xdata,xresid,xdataIn,
                                     xdatadif,xdatastand,volat,lags,h,xsconst,
                                     gridnum,gridmax,gridmin)
```

with the input variables

**xdata**

$T \times 1$  vector of the observed time series

**xresid**

$T' \times 1$  vector of residuals or observations for plotting conditional volatility function, if not needed set `xresid = 0`,

**xdataIn**

character variable, "yes" takes natural logs, "no" doesn't,

**xdatadif**

character variable, "yes" takes first differences of data, "no" doesn't,

**xdatastand**

character variable, "yes" standardizes data, "no" doesn't,

**volat**

character variable, "no" plots conditional mean function, "resid" plots conditional volatility function, the residuals of fitting a conditional mean function have to be contained in `xresid`,

**lags**

$m \times 1$  vector of lags,

**h**

scalar bandwidth for which if set to zero a scalar plug-in bandwidth using `hoptest` is computed or a  $m \times 1$  vector bandwidth

**xsconst**

$m \times 1$  vector (only needed if  $m > 2$ ) indicates which lags vary and which are kept fixed for those keeping fixed, the entry in the corresponding row contains the value at which it is fixed for those to be varied, the entry in the corresponding row is `1e-100`,

**gridnum**

scalar, number of grid points in one direction,

**gridmax**  
 scalar, maximum of grid,

**gridmin**  
 scalar, minimum of grid,

and output variables

**hplugin**  
 scalar plug-in bandwidth  $\hat{h}_{2,opt}$  (37) or chosen scalar or vector bandwidth,

**hB**  
 scalar, rule-of-thumb bandwidth (33) for nonparametrically estimating the constant  $B$  in  $CAFPE_2$  and for computing the plug-in bandwidth,

**hC**  
 scalar, rule-of-thumb bandwidth (36) for nonparametrically estimating the constant  $C$  for computing the plug-in bandwidth,

**xs**  
 $T' \times m$  matrix with lagged values of time series which are used to compute plug-in bandwidth and residuals for potential diagnostics,

**resid**  
 $T' \times 1$  vector with residuals after fitting a local linear regression at **xs**.

Figure 12 shows the plot of the conditional mean function for an NAR(2) model of the lynx data on a grid covering all observations. The autocorrelation function of the residuals is shown in Figure 13. These graphs and a plot of the standardized residuals are computed with the following quantlet. It also returns the Jarque-Bera test statistic of 2.31 with  $p$ -value of 0.32.

```

pathcafpe = "tp/cafpe/" ; path of CAFPE quantlets
; load required quantlibs
library("xplore")
library("times")
func("jarber")
func(pathcafpe + "cafpeload"); load XploRe files of CAFPE
cafpeload(pathcafpe)

setenv("outheadline","") ; no header for each output file

```


```

setenv("outlineno","") ; no numbering of output lines
; set parameters
lynx = read("lynx.dat");
xresid = 0
xdataln = "yes"; ; take logarithms
xdatadif = "no"; ; don't take first differences
xdatastand = "no"; ; don't standardize data
lags = 1|2 ; lag vector for regression function
h = 0
xsconst = 1e-100|1e-100 ; 1e-100 for the lags which are
; varied for those kept fixed it
; includes the chosen constant

gridnum = 30 ; number of gridpoints in one dir.
gridmax = 9 ; maximum of grid
gridmin = 4 ; minimum of grid
; compute opt. bandwidth and plot regression fct. for given lags
{ hplugin,hB,hC,xs,resid } = plotloclin(lynx,xresid,xdataln,
xdatadif,xdatastand,volat,lags,h,
xsconst,gridnum,gridmax,gridmin)

"plug-in bandwidth" hplugin
; diagnostics
acfplot(resid) ; compute and plot acf of residuals
{jb,probb,sk,k} = jarber(resid,1)
; compute Jarque-Bera test for normality of residuals

```

 flts10.xpl

From inspecting Figure 13 one can conclude that a NAR(2) model captures most of the linear correlation structure. However, the autocorrelation at lags 3 and 4 is close to the boundaries of the confidence intervals of white noise and explains why the CAFPE procedure suggests lags one to four. The regression surface in Figure 12 nicely shows the nonlinearity in the conditional mean function which may be difficult to capture with standard parametric nonlinear models.

## 2.4 Estimation of the Conditional Volatility

So far we have considered the estimation and lag selection for the conditional mean function  $f(x)$ . Finally, we turn our attention to modelling the function of the conditional standard deviation  $\sigma(x)$ . The conditional standard deviation


plays an important role in financial modelling, e.g. for computing option prices. As an example we consider 300 logged observations `dmus58-300` of a 20 minutes spaced sample of the Deutschemark/US-Dollar exchange rate. Figures 14 and 15 display the logged observations and its first differences. The figures are generated with the quantlet

```

library("plot")
library("times")
setsize(640,480)
fx      = read("dmus58-300.dat"); read data
d1      = createdisplay(1,1)
x1      = #(1:300)~fx
setmaskl(x1, (1:rows(x1))', 0, 1)
show(d1,1,1,x1)      ; plot data
setgopt(d1,1,1,"title",
        "20 min. spaced sample of DM/US-Dollar rate")
setgopt(d1,1,1,"xlabel","Periods","ylabel","levels")

d2      = createdisplay(1,1)
x2      = #(2:300)~tdiff(fx)
setmaskl(x2, (1:rows(x2))', 0, 1)
show(d2,1,1,x2)      ; plot data
setgopt(d2,1,1,"title","20 min. spaced sample of
        DM/US-Dollar rate - first differences")
setgopt(d2,1,1,"xlabel","Periods","ylabel","first differences")

```

 `flts11.xpl`

In the following we assume that the conditional mean function  $f(\cdot)$  is known and subtracted from  $Y_t$ . Thus, we obtain  $\tilde{Y}_t = Y_t - f(X_t)$ . After squaring (16) and rearranging we have

$$\tilde{Y}_t^2 = \sigma^2(X_t) + \sigma^2(X_t)(\xi_t^2 - 1). \quad (41)$$

Since  $\sigma^2(X_t)(\xi_t^2 - 1)$  has expectation zero, the stochastic process (41) can be modelled with the methods described in Subsections 2.1 and 2.2 by simply replacing the dependent variable  $Y_t$  by its squares. However, we have to remark that the existence of the expectation  $E \left[ \left( \tilde{Y}_t^2 - \sigma^2(X_t) \right)^2 \right]$  is a necessary condition for applying  $CAFPE_2$ . Otherwise, the FPE cannot be finite. We

note that if  $f(x)$  has to be estimated, the asymptotic properties of  $CAFPE_2$  are expected to remain the same. Therefore, it may be used in practice, however, after replacing  $\tilde{Y}_t$  by the residuals  $Y_t - \hat{f}_2(X_t)$ . This is possible with the quantlet `tp/capfe/cafpefull` which extends the functionality of the quantlet `tp/cafpe/cafpe` and allows the user to change additional tuning parameters. The quantlet `tp/cafpe/cafpefull` is called by

```
{crmin, crpro, crstore, crstoreadd, hstore, hstoretest} =
cafpefull(y, truedat, xresid, trueres, xdataln, xdatadif, xdatastand,
          lagmax, volat, searchmethod, dmax, selcrit, robden, perA,
          perB, startval, noutputf, outputh)
```

and has input variables

**y**

$T \times 1$  vector of univariate time series,

**truedat**

character variable that contains path and name of ascii data file if `y=0`,

**xresid**

$T' \times 1$  vector of residuals or observations for selecting lags of conditional volatility function, if not needed set `xresid = 0`,

**trueres**

character variable, "yes" takes natural logs, "no" doesn't,

**xdatadif**

character variable, "yes" takes first differences of data, "no" doesn't,

**xdatastand**

character variable, "yes" standardizes data, "no" doesn't,

**lagmax**

scalar, largest lag to be considered,

**volat**

character variable, "no" conducts lag selection for conditional mean function, "resid" conducts lag selection for conditional volatility function, the residuals of fitting a conditional mean function have to be contained in `xresid` or a file name has to be given in `trueres`,

**searchmethod**  
character variable for determining search method, "full" conducts full search over all possible input variable combinations, "directed" does directed search,

**dmax**  
scalar, maximal number of lags

**selcrit**  
character variable to select lag selection criterion, "lqafpe" estimates the asymptotic Final Prediction Error  $AFPE_2$  (38) using local linear estimation and the plug-in bandwidth  $\hat{h}_{2,opt}$  (37), "lqcafpe" estimates the corrected asymptotic Final Prediction Error  $CAFPE_2$  (39) using local linear estimation and the plug-in bandwidth  $\hat{h}_{2,opt}$  (37)

**robden**  
character variable, "yes" and "no" switch on and off robustification in density estimation (40),

**perA**  
scalar, parameter used for screening off a fraction of  $0 \leq \text{perA} \leq 1$  observations with the lowest density in computing  $\hat{A}_2$

**perB**  
scalar, parameter like **perA** but for screening off a fraction of **perB** observations with lowest density in computing  $\hat{B}_2$ ,

**startval**  
character variable to control treatment of starting values, "different" uses for each lag vector as few starting values as necessary, "same" uses for each lag vector the same starting value which is determined by the largest lag used in the lag selection quantlet `tp/cafpe/xorigxe`,

**noutputf**  
character variable, name of output file,

**outpath**  
character variable, path for output file.

The output variables are



**crmin**

vector that stores for all considered lag combinations in the first  $d_{\max}$  rows the selected lag vector, in the  $d_{\max}+1$  row the estimated criterion, in the  $d_{\max}+2$  row  $\hat{A}_2$ , in the  $d_{\max}+3$  row the bias corrected estimate of  $A$ ,

**crpro**

matrix that stores for each number of lags in the first  $d_{\max}$  rows the selected lag vector, in the  $d_{\max}+1$  row the plug-in bandwidth  $\hat{h}_{2,opt}$  for estimating  $A$  and  $(C)AFPE$ , in the  $d_{\max}+2$  row the bandwidth  $\hat{h}_B$  used for estimating  $B$ , in the  $d_{\max}+3$  row the bandwidth  $\hat{h}_C$  for estimating  $C$ , in the  $d_{\max}+4$  row the estimated criterion  $AFPE_2$  or  $CAFPE_2$ , in the  $d_{\max}+5$  row  $\hat{A}_2$ , in the  $d_{\max}+6$  row the bias corrected estimate of  $A$ ,

**crstore**

matrix that stores lag vector and criterion value for all lag combinations and bandwidth values considered, in the first  $d_{\max}$  rows all considered lag vector are stored, in the  $d_{\max}+1$  rows the estimated criterion for each lag vector is stored,

**crstoreadd**

matrix that stores those criteria that are evaluated in passing for all lag combinations where all values for one lag combination are stored in one column (see program for details),

**hstore**

row vector that stores the bandwidths used in computing  $(C)AFPE$  for each lag vector

**hstoretest**

matrix that stores for each lag vector in one column the plug-in bandwidth  $\hat{h}_{2,opt}$ ,  $\hat{h}_B$  and  $\hat{h}_C$ .

The quantlet `flts12.xpl` (for brevity not shown) conducts a lag selection for the conditional mean function  $f(x)$  and finds lag 1 and 3 with bandwidth  $\hat{h}_{2,opt} = 0.000432$ . If you run the quantlet, you will obtain the XploRe warning “quantlet fvllc: inversion in local linear estimator did not work because probably the bandwidth is too small”. This means that for one of the checked combinations of lags, one of the rule-of-thumb bandwidths or the plug-in bandwidth was too small so that the matrix  $Z_2^T W Z_2$  in the local linear estimator

(21) is near singular and the matrix inversion failed. In this case, the relevant bandwidth is doubled (at most 30 times) until the near singularity disappears. Therefore, lag selection for the conditional volatility function  $\sigma(x)$  is done with replacing the observations  $Y_t$  in model (41) by the estimated residuals  $Y_t - \hat{f}(X_t)$ . The computations are carried out with the following quantlet which also generates a plot of the conditional mean function on the range  $[-0.0015, 0.0015]$  displayed in Figure 16 and plots the autocorrelation function of the residuals (not shown). The latter plot does not show significant autocorrelation.

```


    pathcafpe      = "tp/cafpe/"      ; path of CAFPE quantlets
;   load required quantlibs
    library("xplore")
    library("times")
    func("jarber")
    func(pathcafpe + "cafpeload") ;load XploRe files of CAFPE
    cafpeload(pathcafpe)
;   set output format
    setenv("outheadline","") ; no header for each output file
    setenv("outlineno","")   ; no numbering of output lines
;   load data
    x                = read("dmus58-300.dat") ; name of data file
    y                = tdiff(x) ; compute first differences
    xresid           = 0
    truedat          = ""             ; name of potential data file
    trueres          = ""             ; name of potential residuals file
    xdataln          = "no"           ; don't take logarithms
    xdatadif         = "no"           ; don't take first differences
    xdatastand       = "no"           ; don't standardize data
    lagmax           = 6               ; the largest lag considered is 6
    searchmethod     = "full"         ; consider all possible lag comb.
    dmax             = 6               ; consider at most 6 lags
    volat            = "no"           ; plot cond. mean function
    selcrit          = "lqcafpe"      ; use CAFPE with plug-in bandwidth
    robden           = "yes"          ; robustify density estimation
    perA             = 0
    perB             = 0.05           ; screen off data with lowest density
    startval         = "different"
    noutputf         = ""             ; name of output file
    outpath          = "test"         ; path for output file

```

```

lags          = 1|3          ; lag vector for regression function
h             = 0
xsconst       = 1e-10|1e-100 ; 1e-100 for the lags which are
                                ; varied for those kept fixed it
                                ; includes the chosen constant
gridnum       = 30          ; number of gridpoints in one direction
gridmax       = 0.0015     ; maximum of grid
gridmin       = -0.0015    ; minimum of grid
; compute optimal bandwidth and plot cond. mean for given lags
{ hplugin,hB,hC,xs,resid } = plotloclin(y,xresid,xdataIn,
                                xdatadif,xdatastand,volat,lags,h,xsconst,gridnum,
                                gridmax,gridmin)
"plug-in bandwidth for conditional mean" hplugin
; diagnostics
acfplot(resid); compute and plot acf of residuals
{jb,probjb,sk,k} = jarber(resid,1)
                ; compute Jarque-Bera test for normality of residuals
; conduct lag selection for cond. standard deviation
xresid        = resid
volat         = "resid" ; conduct lat selection for cond. vol.
{crmin,crpro,crstore,crstoreadd,hstore,hstoretest}
              = cafpefull(y,truedat,xresid,trueres,xdataIn,
                          xdatadif,xdatastand,lagmax,volat,
                          searchmethod,dmax,selcrit,robden,
                          perA,perB,startval,noutputf,outpath)
"Lag selection for cond. standard deviation using residuals"
"selected lag vector,          estimated CAFPE "
crmin[,1:dmax+1]
"number of lags, chosen lag vector, estimated CAFPE,
                                plug-in bandwidth"
(0:dmax)~crpro[,1:dmax|(dmax+4)|(dmax+1)]

```

 flts13.xpl

For the conditional standard deviation one obtains lags 2 and 6 with bandwidth  $\hat{h}_{2,opt} = 0.000456$ . Figures 17, 18 and 19 display the plot of the estimated conditional standard deviation  $\hat{\sigma}_2(x)$ , of the standardized residuals of the modified model (41) and of their autocorrelation. The plots are generated with the following quantlet

```

pathcafpe = "tp/cafpe/" ; path of CAFPE quantlets

; load required quantlets
library("xplore")
library("times")
func("jarber")
func(pathcafpe + "cafpe.load"); load XploRe files of CAFPE
cafpe.load(pathcafpe)

setenv("outheadline","") ; no header for each output file
setenv("outlineno","") ; no numbering of output lines

; set parameters
x = read("dmus58-300.dat");
y = tdiff(x)
xresid = 0
xdataaln = "no" ; don't take logarithms
xdatadif = "no" ; don't take first differences
xdatastand = "no" ; don't standardize data
volat = "no" ; compute cond. standard deviation
lags = 1|3 ; lag vector for regression function
h = 0 ; compute plug-in bandwidths
xsconst = 1e-100|1e-100
; 1e-100 for the lags which are varied
; for those kept fixed it includes the
; chosen constant
gridnum = 30 ; number of gridpoints in one direction
gridmax = 0.0015 ; maximum of grid
gridmin = -0.0015 ; minimum of grid

; compute optimal bandwidth and plot cond. mean for given lags
{ hplugin,hB,hC,xs,resid } = plotloclin(y,xresid,xdataaln,
xdatadif,xdatastand,volat,lags,h,xsconst,gridnum,
gridmax,gridmin)

"plug-in bandwidth for mean" hplugin

; compute plug-in bandwidth and
; plot cond. standard deviation for given lags
lags = 2|6 ; lags for cond. volatility
xresid = resid

```


```

volat      = "resid"
gridmax    = 0.0008      ; maximum of grid
gridmin    = -0.0008    ; minimum of grid

{ hplugin,hB,hC,xs,resid } = plotloclin(y,xresid,xdata1n,
                                       xdatadif,xdatastand,volat,lags,h,xsconst,gridnum,
                                       gridmax,gridmin)
"plug-in bandwidth for conditional volatility" hplugin

; diagnostics
acfplot(resid); compute and plot acf of residuals
{jb,probjb,sk,k} = jarber(resid,1)
                   ; compute Jarque-Bera test for normality of residuals

```

 flts14.xpl

The surface plot of the conditional standard deviation is computed on the range  $[-0.0008, 0.0008]$  in order to avoid boundary effects. Inspecting the range of the standardized residuals in Figure 18 indicates that the analysis may be strongly influenced by outliers which also may explain the extreme increase of the conditional standard deviation in Figure 17 in one corner. Moreover, Figure 19 shows some significant autocorrelation in the residuals. One explanation for this finding could be the presence of long memory in the squared observations. This topic is treated in detail in Chapter ???. Therefore, one should continue to improve the current function estimates by excluding extreme observations and using models that allow for many lags in the function of the conditional standard deviation such as, for example, Yang, Härdle and Nielsen (1999).

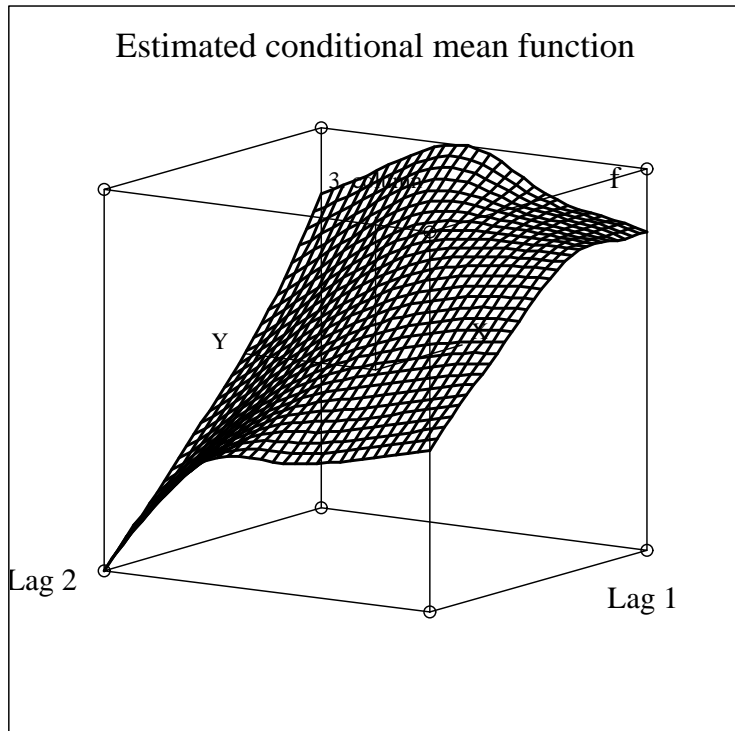


Figure 12: Plot of the conditional mean function of a NAR(2) model for the logged lynx data

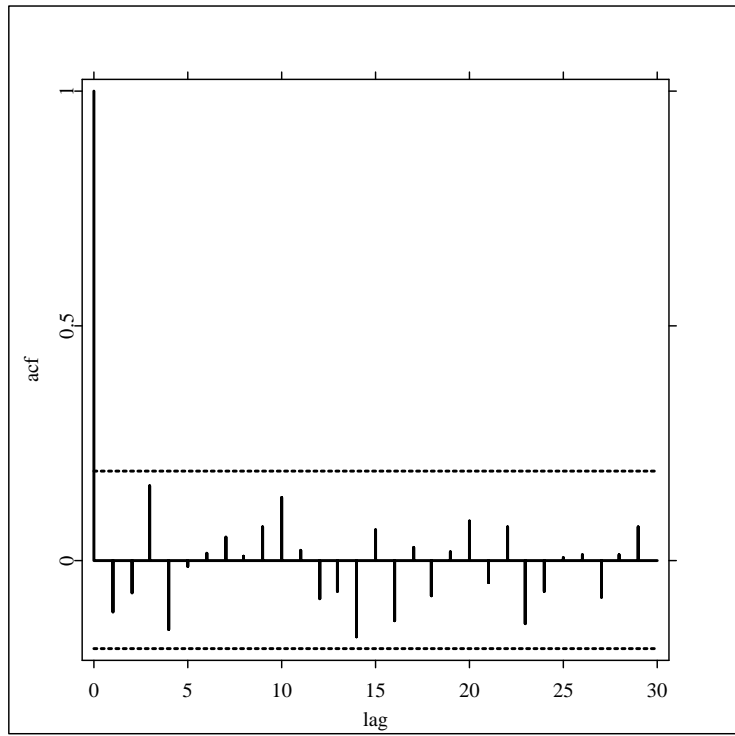


Figure 13: Plot of the autocorrelation function of the residuals of a NAR(2) model for the logged lynx data

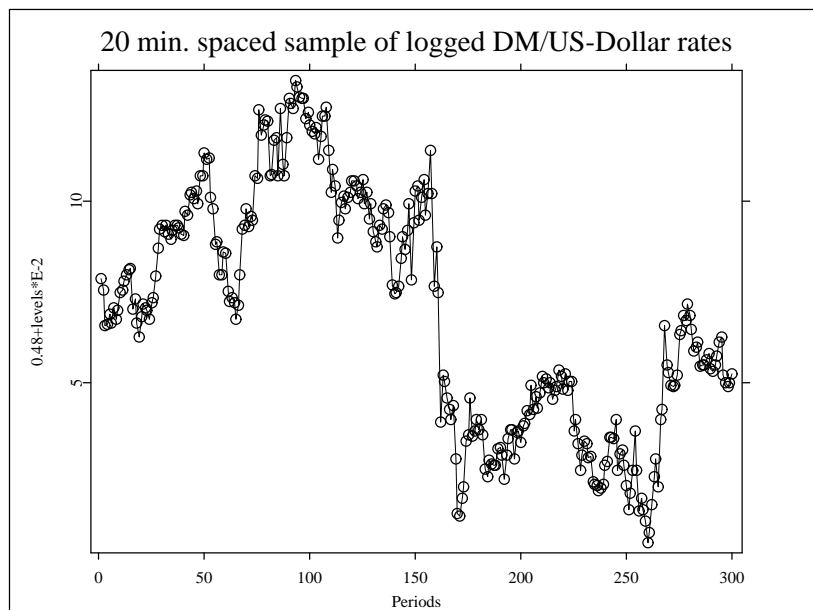


Figure 14: Time series of logarithm of 20 minutes spaced sample of DM/US-Dollar rate



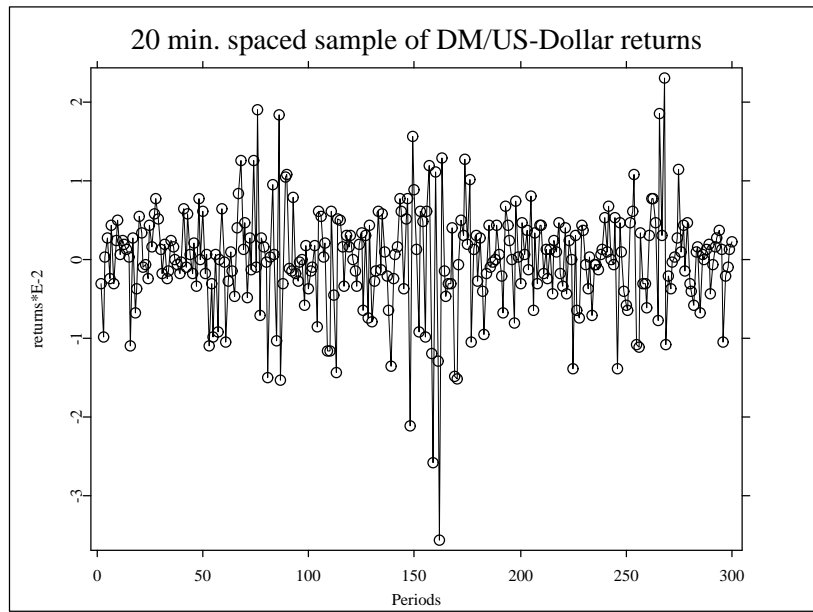


Figure 15: Time series of 20 minutes spaced sample of exchange rate returns

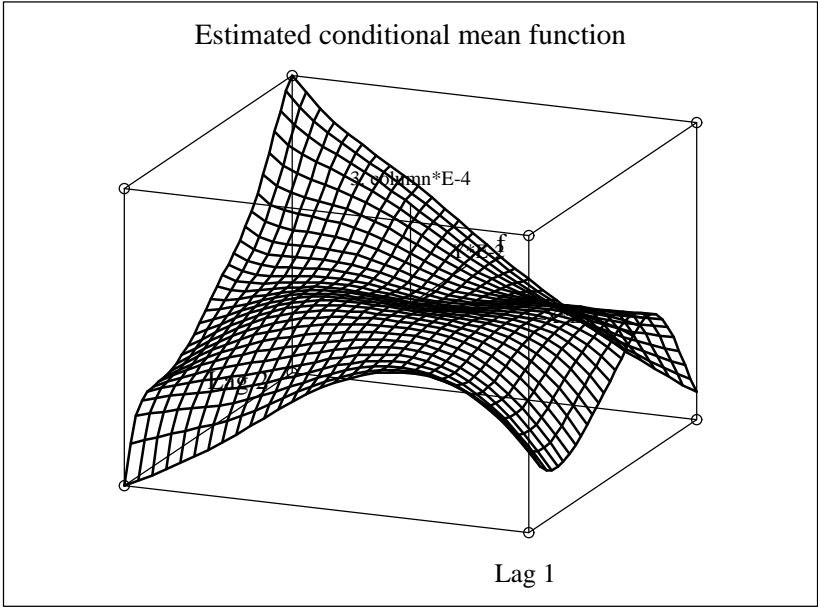


Figure 16: Plot of the conditional mean function of a NAR model with lags 1 and 3 for the returns of the Deutschemark/US-Dollar exchange rate

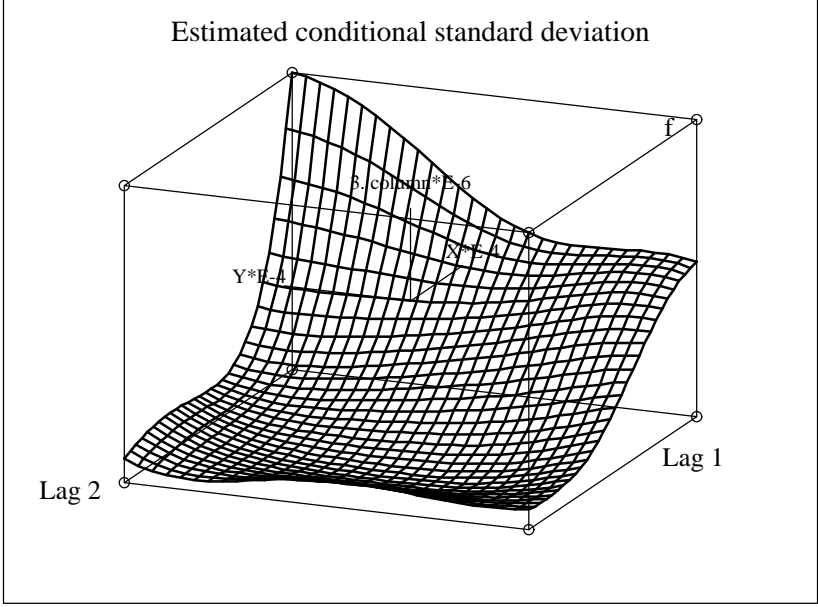


Figure 17: Plot of the conditional standard deviation of a NAR model with lags 2 and 6 for the returns of the Deutschemark/US-Dollar exchange rate

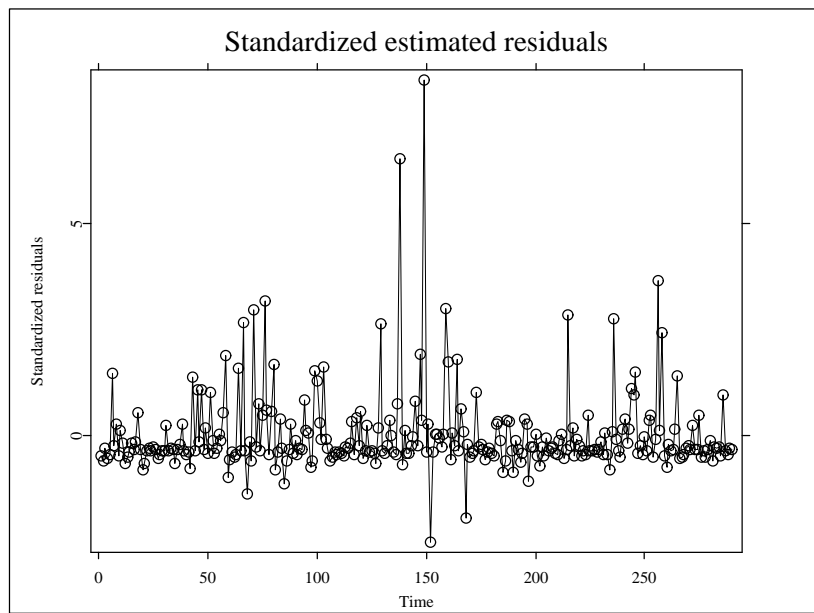


Figure 18: Plot of the standardized residuals of the modified model (41)

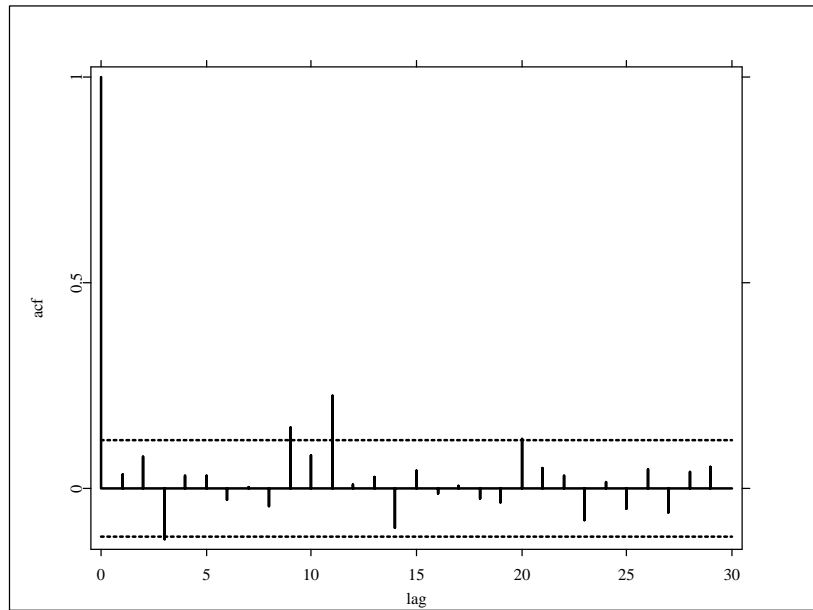


Figure 19: Plot of the autocorrelation function of residuals of the modified model (41)

## References

- Auestad, B. and Tjøstheim, D. (1990). ‘Identification of nonlinear time series: first order characterization and order determination’, *Biometrika* **77**: 669–687.
- Bera, A.K. and Jarque, C.M. (1982). Model Specification Tests: a Simultaneous Approach, *Journal of Econometrics*, **20**: 59–82.
- Billingsley, P. (1968). *Convergence of Probability Measures*, New York: Wiley.
- Brockwell, P.J. and Davis, R.A. (1991). *Time Series: Theory and Methods*, Springer, New York.
- Doukhan, P. (1994). *Mixing. Properties and Examples*, Springer-Verlag, New York et al.
- Fan, J. and Gijbels, I. (1995). ‘Data-driven bandwidth selection in local polynomial fitting: Variable bandwidth and spatial adaption’, *Journal of the Royal Statistical Society B* **57**(2): 371–394.
- Fan and Marron (1994). ‘Fast implementations of nonparametric curve estimators’, *Journal of Computation and Graphical Statistics* **3**, 35 – 56.
- Franke, J., Kreiss, J.-P., Mammen, E., and Neumann, M. H. (1998). ‘Properties of the nonparametric autoregressive bootstrap’, Discussion Paper 54/98, SFB 373, Humboldt University, Berlin.
- Härdle, W. (1990). *Applied Nonparametric Regression*, Cambridge University Press: Cambridge
- Härdle, W. and Tsybakov, A. (1997). ‘Local polynomial estimators of the volatility function in nonparametric autoregression’, *Journal of Econometrics* **81**, 223–242.
- Härdle, W. and Vieu, P. (1992). ‘Kernel regression smoothing of time series’, *Journal of Time Series Analysis* **13**: 209–232.
- Härdle, W., Klinke, S., and Müller, M. (2000), *XploRe — The Statistical Computing Environment*, Springer, New York.
- Härdle, W., Lütkepohl, H., and Chen, R. (1997). ‘A review of nonparametric time series analysis’, *International Statistical Review* **65**(1): 49–72.

- Lu, Z. (1998). 'On the geometric ergodicity of a non-linear autoregressive model with an autoregressive conditional heteroscedastic term', *Statistica Sinica* **8**: 1205–1217.
- Robinson, P. M. (1983). 'Non-parametric estimation for time series models', *Journal of Time Series Analysis* **4**: 185–208.
- Silverman, B. (1986). *Density estimation for Statistics and Data Analysis*, Chapman and Hall, London.
- Tjøstheim, D. (1994). 'Nonlinear time-series - a selective review', *Scandinavian Journal of Statistics* **21**(2): 97–130.
- Tjøstheim, D. and Auestad, B. (1994). 'Nonparametric identification of nonlinear time-series - selecting significant lags', *Journal of the American Statistical Association* **428**: 1410–1419.
- Tschernig, R. and Yang, L. (2000). 'Nonparametric lag selection for time series', *Journal of Time Series Analysis*, forthcoming.
- Vieu, P. (1994). 'Order choice in nonlinear autoregressive models', *Statistics* **24**: 1–22.
- Yang, L., Härdle, W. and Nielsen, J.P. (1999). 'Nonparametric autoregression with multiplicative volatility and additive mean', *Journal of Time Series Analysis* **20**: 579–604.
- Yang, L. and Tschernig, R. (1999). 'Multivariate bandwidth selection for local linear regression', *Journal of the Royal Statistical Society, Series B*, **61**, 793–815.
- Yao, Q. and Tong, H. (1994). 'On subset selection in non-parametric stochastic regression', *Statistica Sinica* **4**: 51–70.