

# Web quantlets for time series analysis

Wolfgang Härdle<sup>1</sup>, Torsten Kleinow<sup>2</sup>, Rolf Tschernig<sup>3</sup>

Institut für Statistik und Ökonometrie, Wirtschaftswissenschaftliche Fakultät,  
Humboldt-Universität zu Berlin, Spandauer Strasse 1, D-10178 Berlin  
<http://ise.wiwi.hu-berlin.de>

SFB 373 Discussion Paper No. 1, 2000  
available at: <http://sfb.wiwi.hu-berlin.de>

## Abstract

Newly developed and advanced methods for nonlinear time series analysis are in general not available in standard software packages. Moreover, their implementation requires substantial time, computing power as well as programming skills. The recent results on lag and bandwidth selection methods for nonlinear autoregressive time series models provide such a scenario. Application of these methods requires enormous computing resources if larger samples are considered. In this paper we suggest a method to provide empirical researchers with a fast access to new methods as well as to powerful computing environments. It is illustrated with a recently suggested nonparametric lag selection procedure based on CAFPE (Corrected Asymptotic Final Prediction Error). Our approach is based on the XploRe quantlet technology. Its worldwide Web usage is made possible by a specific client/server architecture. It allows researchers to use the quantlet computing service without knowing either the statistical computing language or the server location. Quantlets are accessed via standard WWW browsers or via a Java client which works like a standard desktop environment. This architecture allows a flexible scaling of time consuming computations on either client or server. The XploRe quantlet service is helpful in constructing research books and interactive teaching environments as the electronic version of this paper demonstrates.

**Keywords:** distributed computing, lag selection, nonparametric time series analysis, quantlets, Web based computing

<sup>1</sup> [stat@wiwi.hu-berlin.de](mailto:stat@wiwi.hu-berlin.de)   <sup>2</sup> [kleinow@wiwi.hu-berlin.de](mailto:kleinow@wiwi.hu-berlin.de)   <sup>3</sup> [rolf@wiwi.hu-berlin.de](mailto:rolf@wiwi.hu-berlin.de)  
The authors acknowledge support by the Deutsche Forschungsgemeinschaft via Sonderforschungsbereich 373 "Quantifikation and Simulation ökonomischer Prozesse" at Humboldt-Universität zu Berlin

# 1 Introduction

New methods in time series analysis involve often computationally intensive numerical operations and entail often highly complex mathematical arguments. Bootstrapping time series (e.g. Franke, Kreiss, Mammen & Neumann (1998)) involves quite a bit of computing resources, for example. Nonparametric procedures such as wavelets (e.g. Härdle, Kerkycharian, Picard & Tsybakov (1998), Dahlhaus, Neumann & von Sachs (2000)) assume a certain degree of mathematical skills before they can be used for practical implementation. Both the computational and mathematical complexity slows down the process of real world applications for new time series methods. In this paper we propose an architecture that helps in the proliferation of new time series methods. More precisely, we suggest a design for computing that combines Java applets with the advantages of statistical programming languages. This is particularly useful for nonlinear time series analysis. These methods and the advances in computing technology have made it possible to avoid too restrictive assumptions for model construction. Consider, for example, the problem of modelling a univariate time series. For the last two decades it has been common for analysing time series data to use the well developed toolbox of linear autoregressive moving-average (ARMA) and integrated autoregressive moving-average (ARIMA) models, see e.g. Brockwell & Davis (1991). This model class, however, can only be adequate if one is exclusively interested in linear dependencies through time.

However, in many applications modelling asymmetric effects or regime-dependent dynamics is of central importance. Then linear models are an inadequate choice and one considers, for example in finance, conditional heteroskedastic nonlinear autoregressive processes instead. A general nonparametric time series model for an univariate stochastic process  $\{Y_t\}_{t=1}^n$  is given by

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

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

Quite often, theory in the field of application does not indicate a reasonable choice of these functions and parameters and one has to apply statistical model selection methods which only recently have become available for general function classes (Vieu (1994), Yao & Tong (1994), Auestad & Tjøstheim (1990), Tjøstheim & Auestad (1994), Tschernig & Yang (2000)).

All of these procedures are rather complex and computer intensive especially for large data sets. Both factors substantially hinder the proliferation and application of these techniques. In the best case, the developer of a new method of this kind provides public access to the algorithm via his own home page or a method basis which is accessible via FTP, e.g. <http://lib.stat.cmu.edu/S/>, for programs written in S-Plus. However, in order to execute the algorithm, the user needs access to the software system that can execute or compile the program (e.g. S-Plus, GAUSS,

FORTRAN). Moreover, their implementation may not always be an easy task nor may these programs always be well checked and documented.

These difficulties can be circumvented by a more Web based approach in which case a new method is implemented in Java. While this allows potential users to access an algorithm from almost any platform, this method suffers from three drawbacks. First, due to its generality, Java implemented programs may be slower than those written in more platform dependent languages. Thus, the user's computing facilities may be occupied more than necessary. Second, a developer of a method faces double programming since he has to write the code both in a proprietary software language and in Java. Third, if Java programs are accessed via any browser one does not have access to the resources of the local machine. For example, one cannot load data files from the local hard disk. Nakano (1998) already suggested a client/server architecture that avoids the latter problems but does not provide the advantages of Java applets.

Here we suggest a new approach that combines the advantages of Java applets with the advantages of statistical programming languages in a distributed computing framework. Our ideas are illustrated with advanced statistical model selection methods for time series data.

In Section 2 we present a nonparametric identification method for nonlinear time series models. Section 3 presents our new Web based computing architecture. Section 4 provides an application of these ideas using the algorithm of Section 2.

## 2 Nonparametric identification of nonlinear time series models

Fitting a nonlinear time series model like (1) requires two steps:

- 1) lag selection: one has to choose the relevant lags  $i_1, \dots, i_m$  including their number  $m$  for the autoregression function  $f$  and the conditional standard deviation  $\sigma$ .
- 2) function estimation: estimate  $f$  and, if desired,  $\sigma$  for the chosen lag vector.

Here we present a nonparametric lag and estimation procedure that has recently been suggested by Tschernig & Yang (2000). For all details on the assumptions as well as on the implementation the reader is referred to their paper. We first describe Step 2 of nonparametrically estimating  $f$ . Let  $\mathbf{X}_t = (Y_{t-i_1}, Y_{t-i_2}, \dots, Y_{t-i_m})^T$  denote the vector that contains all selected lags. The idea is to estimate a first order Taylor approximation of the unknown function  $f$  around a given point  $\mathbf{x}$ . Since including observations  $\mathbf{X}_t$  that are distant to  $\mathbf{x}$  would introduce a large approximation bias, one weights those observations less in the estimation. Using the least squares principle the estimated function value  $\hat{f}(\mathbf{x}, h)$  is provided by the estimated constant  $\hat{c}_0$  of a

local polynomial estimate around  $\mathbf{x}$

$$\{\hat{c}_0, \hat{\mathbf{c}}\} = \arg \min_{\{c_0, \mathbf{c}\}} \sum_{t=1}^T \left\{ Y_t - c_0 - (\mathbf{X}_t - \mathbf{x})^T \mathbf{c} \right\}^2 K_h(\mathbf{X}_t - \mathbf{x})$$

where  $K$  denotes the weighting function which is commonly called a kernel function and  $K_h(\mathbf{X}_t - \mathbf{x}) = h^{-m} \prod_{i=1}^m K\{(X_{t,i} - x_i)/h\}$  is a product kernel.  $\hat{f}(\mathbf{x}, h) = \hat{c}_0$  is known as a local linear function estimator. Similarly one can estimate the conditional standard deviation  $\sigma$ , see Härdle & Tsybakov (1997) for details.

The parameter  $h$  is called bandwidth parameter and controls the weighting of the data  $\mathbf{X}_t$  with respect to their distance to  $\mathbf{x}$ . 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. An optimal tradeoff is obtained if the sum of the estimation variance and squared bias, called the mean squared error is minimized. This optimal bandwidth is unknown since it depends on the unknown function  $f$ . It is, however, possible to estimate the unknown quantities of the asymptotically optimal bandwidth  $h_{opt}$  using similar methods to those described in Yang & Tschernig (1999) in order to obtain the plug-in bandwidth  $\hat{h}_{opt}$ .

We now turn to the problem of selecting the relevant lags (Step 1). For this step it is necessary to 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  $\mathbf{X}_{t,M} = (Y_{t-1}, Y_{t-2}, \dots, Y_{t-M})^T$ . The lag selection task is now to eliminate from the full lag vector  $\mathbf{X}_{t,M}$  all lags that are redundant. For comparing the quality of competing lag specifications, one needs an appropriate measure of fit, as for example the final prediction error (FPE)

$$FPE(h, i_1, \dots, i_m) = E \left[ \check{Y}_t - \hat{f}(\check{X}_t, h)^2 \right].$$

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. The  $FPE$  measures the average squared error from using  $\hat{f}(\check{X}_t)$  instead of  $f(\check{X}_t^*)$  where  $X_t^*$  denotes the correct lag vector. In the literature mainly two approaches were suggested for estimating the unknown  $FPE$  or variants thereof, namely cross-validation (Vieu (1994), Yao & Tong (1994)) or estimation of an asymptotic expression of the  $FPE$  (Auestad & Tjøstheim (1990), Tjøstheim & Auestad (1994) or Tschernig & Yang (2000)). The latter requires to compute

$$AFPE = \hat{A}(\hat{h}_{opt}) + 2K(0)^m T^{-1} \hat{h}_{opt}^{-m} \hat{B}(\hat{h}_B) \quad (2)$$

where the bandwidth  $\hat{h}_B$  is defined in (3) below. In the  $AFPE$  the unknown constants  $A$  and  $B$  which have to be estimated are the integrated mean squared error and an element of the integrated estimation variance, respectively. One therefore can interpret the second term in (2) as a penalty term to punish overfitting or choosing superfluous lags. Note that the penalty term decreases with sample size as  $h_{opt}$

is of order  $T^{-1/(m+4)}$ . The integrated mean squared error  $A$  can be estimated by the sample average

$$\hat{A}(h) = T^{-1} \sum_{t=1}^T \left\{ y_t - \hat{f}(\mathbf{X}_t, h) \right\}^2 w(\mathbf{X}_{t,M})$$

based on the local linear estimator  $\hat{f}(\mathbf{X}_t, h)$  and where  $w(\cdot)$  denotes a weight function. The asymptotic properties of the lag selection method rely on the fact that the argument of  $w(\cdot)$  is the full lag vector  $\mathbf{X}_{t,M}$ . A nonparametric estimate of  $B$  is obtained from

$$\hat{B}(\hat{h}_B) = T^{-1} \sum_{t=1}^T \left\{ Y_t - \hat{f}(\mathbf{X}_t, \hat{h}_B) \right\}^2 w(\mathbf{X}_{t,M}) / \hat{\mu}(\mathbf{X}_t, \hat{h}_B)$$

where  $\hat{\mu}(\cdot)$  is a Gaussian kernel estimator of the density  $\mu(\mathbf{x})$  using Silverman's (1986) rule-of-thumb bandwidth

$$\hat{h}_B = \hat{\sigma} \left( \frac{4}{T+2} \right)^{1/(m+4)} T^{-1/(m+4)} \quad (3)$$

and where  $\hat{\sigma} = \left( \prod_{j=1}^m \sqrt{\text{Var}(\mathbf{X}_j)} \right)^{1/m}$  denotes the geometric mean of the standard deviation of the regressors. In order to select the adequate lag vector, one computes (2) for all possible lag combinations with  $m \leq M$  and chooses the lag vector with the smallest *AFPE*. Tschernig & Yang (2000) 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.

These authors show that asymptotically it is more likely to overfit (include superfluous lags in addition to the correct ones) 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

$$CAFPE = AFPE \left\{ 1 + mT^{-4/(m+4)} \right\}. \quad (4)$$

The correction does not affect consistency while additional lags are punished more heavily in finite samples. One chooses the lag vector with the smallest *CAFPE*.

We note that if one allows the maximal lag  $M$  to grow with sample size Step 1) and Step 2) together constitute a doubled nonparametric problem of nonparametric function estimation and nonparametric lag selection.

### 3 Web quantlets

To facilitate access to and usage of new statistical methods such as for example *CAFPE*, we propose a Web based client/server architecture. This architecture combines advantages of Web based statistical computing which is becoming increasingly popular with the well known advantages of distributed computing and central data

storage. Central data storage concepts are implemented in a variety of database servers. Web based statistical computing has been fueled by collections of Java applets designed for usage in HTML pages. Examples are the VESTAC project at KU Leuven (Darius, Ottoy, Solomin, Thas, Raeymaekers & Michiels ) or some applets at statlab Heidelberg. Mostly, these applets provide more or less interactive programs for particular tasks. While methods exclusively encoded in Java can be easily used on almost every computer platform, they must be written in Java leading frequently to doubled programming effort for a method provider. Moreover, for computationally intensive statistical procedures Java applets occupy substantial computational resources on the client side as programs written in Java are slower compared to platform-dependent code and all the computations are conducted on the client's machine.

For these reasons we propose an approach that shares with Java applets their easy access to methods but shifts the computational burden to a server by incorporating distributed computing. Our approach provides *access* to (statistical) methods which are implemented on a server and data via the Internet. The methods which are called quantlets can be accessed by standard HTML browsers (Netscape Navigator 4.5, MS Internet Explorer 4.0). The Graphical User Interface (GUI) is implemented in Java, thus ready to run on every Java enabled platform, such as Java enabled Web browsers, as well as on most popular operating systems. The statistical plugins are implemented on the server side in C++ or a matrix oriented programming language, providing fast computation on the Web. From the user's point of view the quantlet server behaves like a collection of Java classes but, due to server side computation, it is able to handle complex statistical problems. The natural matrix oriented programming language makes it easy to extend it by new quantlets.

### 3.1 Providing methods via the Web

One of the most important advantages of the proposed architecture is the easy and standardized way for publication of new methods. It contains three parts: i) an environment for the method provider, ii) access to computational resources to apply the new methods to data and iii) a user interface ensuring a comfortable usage of the developed methods. One therefore has relationships between the method provider, the Quantlet Server (QS) and the Quantlet Client (QC).

This architecture equips the method provider with a tool to develop and test his methods as well as to apply them to his own data. This tool enables him to use a familiar statistical programming language as well as the opportunity to use native code (DLL for Windows for QS, so for Unix QS) written in any programming language. In our environment this part is ensured by a QS/QC combination running on a variety of operating systems and behaving like a traditional statistical computing environment.

In addition, our proposed architecture offers the possibility for the method developer to provide Web based access to his new methods where the design of the

QC should allow to control the options of method users. This can be achieved if the QC is configurable to flexibly restrict access to the methods, e.g. by allowing the user only to change parameters and execute the method or by permitting him to edit the quantlet and to execute its modification. In the first case the QC could behave like a Java applet designed for a particular task, while in the second case it can serve as a simple computing environment. For accessing quantlets, Excel users can in addition plug in the ReX client.

The third way for a method provider is to write his own client. This is useful for developers who want to provide a specialized environment for particular customers, e.g. an environment for analyzing financial data.

### 3.2 Technical background

The proposed client server architecture has been implemented in XploRe (Härdle, Klinke & Müller 1999) and consists of three parts: i) an XploRe Quantlet Server (XQS) providing statistical computing power, ii) a Java based XploRe Quantlet Client (XQC) providing a user friendly graphical interface to interact with the XQS and iii) the middleware program MD\*Serv. Figure 1 shows the structure of this architecture.

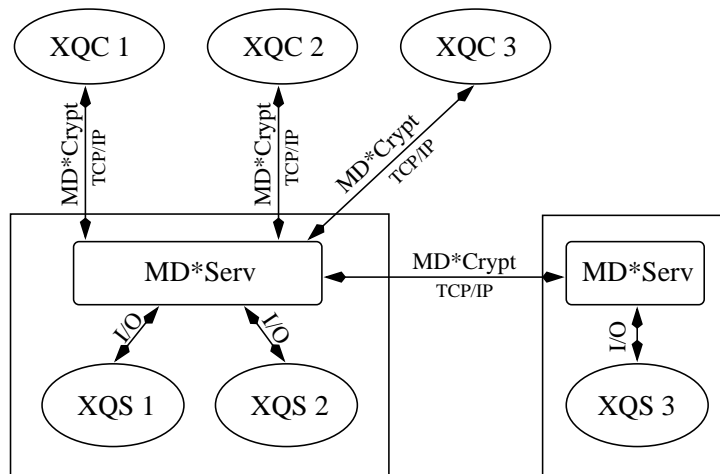


Figure 1: The client/server architecture of XploRe

As described, the XQS provides a statistical programming language (XploRe) and a variety of numerical quantlets for statistical analysis. New quantlets can be written either exclusively in the XploRe language or also as a combination of native code written in a programming language of one's choice via dynamically linked libraries (DLL, so) and XploRe. The XQS writes the results of the computations into its standard Output stream. This stream is encoded in MD\*Crypt, a special communication protocol. The client via MD\*Serv is able to decode MD\*Crypt and displays the results in a readable form on the respective user platform.

The results are passed through the middleware. MD\*Serv is a small Java written program that manages the communication between server and client. Its main task is to handle data coming from the server via the standard Output stream and to pass this data to the client using a TCP/IP stream. MD\*Serv makes it possible to map several server protocols to MD\*Crypt, i.e. it opens the opportunity to connect the XQC to a variety of different servers. This allows to use every batch program as a server by transmitting data via its standard I/O streams. Programming the middleware in Java ensures platform independence and gives an easy way for adding protocol mapping classes. At the moment we have one server and two clients. Besides the Java client there exists the ReX (XploRe, Excel) client based on Visual Basic for use of quantlets in Excel.

## 4 Illustration

In this section we illustrate the suggested Web quantlet architecture for the non-parametric model selection procedure which was presented in Section 2. First, we illustrate the statistical lag selection method via the Web. In this case, our architecture allows users or readers of this paper to replicate the results and modify chosen parameters of the present example. Using the XploRe quantlet [genexpar](#) we generated 50 observations of an exponential autoregressive process of order 2

$$Y_t = 0.3Y_{t-1} + 0.6Y_{t-2} + (1.9Y_{t-1} - 1.1Y_{t-2}) \exp(-0.1 * Y_{t-1}^2) + \xi_t, \quad \xi_t \sim N(0, 1)$$

which are shown in Figure 2. For this time series CAFPE (4) is computed for all

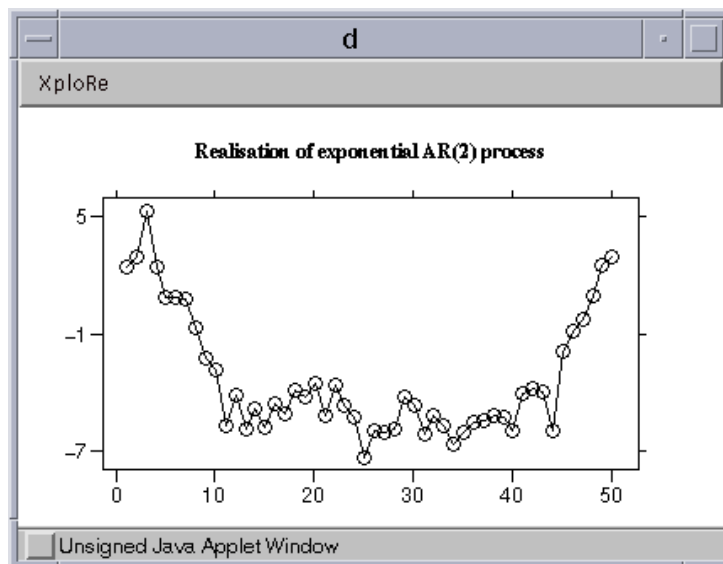


Figure 2: 50 observations of the exponential AR(2) process

combinations of lags up to  $M = 4$  as well as white noise and then the lag vector with the smallest CAFPE is chosen. All this can be done by the quantlet `cafpe`. Both quantlets are called from the public program `cafpeexample`. This example program can be viewed, modified and executed from any Java capable Web browser such that a reader of the electronic version of this paper can obtain a similar output to Figure 3. For the generated series, the method correctly selects lags 1 and 2 with  $CAFPE = 1.5065$

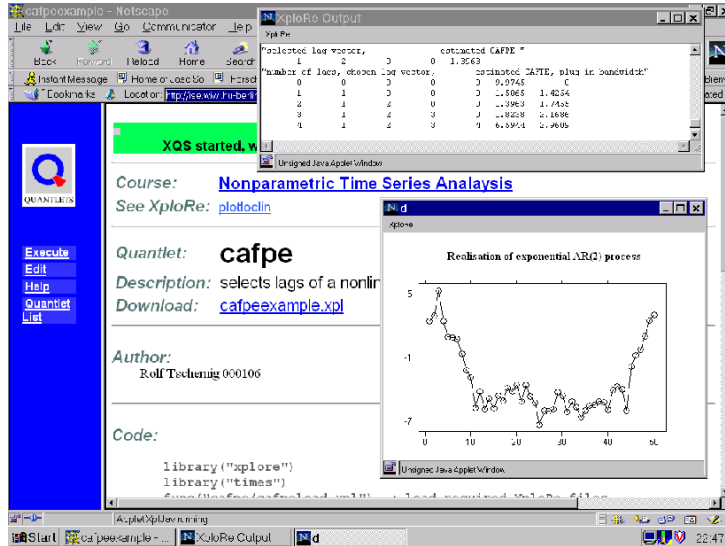


Figure 3: Screen shot of running the example via the Web

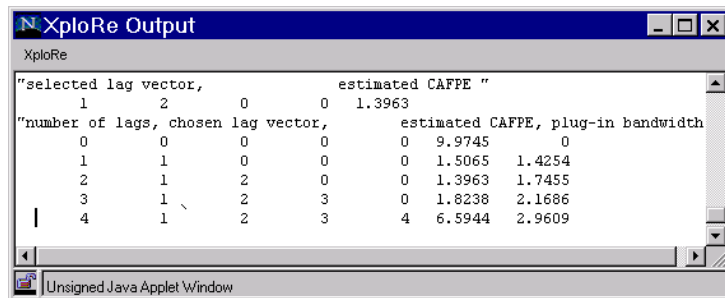


Figure 4: Results for the lag selection procedure

Once the lags are selected, one may estimate the regression function on a grid of the data by calling `plotloclin` from the example program `plotloclinexample`. This produces a 3-dimensional surface plot as shown in Figure 5 which can be rotated by the user. Here the rotation is done by the Java applet and computed on the client.

If a reader is also interested in trying the CAFPE quantlet with his own data, he can download the XploRe client. This allows the user to work and access XploRe

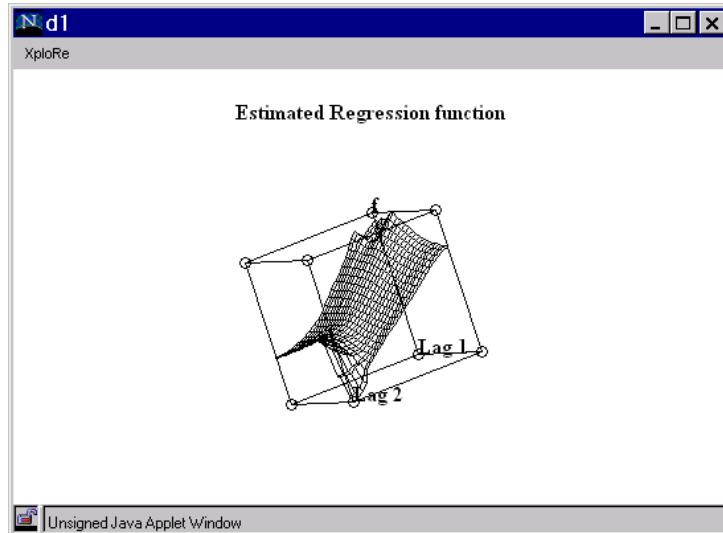


Figure 5: Surface of the estimated two-dimensional regression function

quantlets as if it were installed on his local machine. Whatever approach is used, the client's machine is not occupied by the potentially computationally demanding quantlets. Moreover, the server side computing provides the user with a powerful computing environment even if the client is somewhat outdated. The example with 5000 observations, for example, takes 10.5 hours on a Pentium 200 Mhz in contrast to 2.5 hours on the Sun Ultra 2 sparc server.

## References

- Auestad, B. & Tjøstheim, D. (1990), 'Identification of nonlinear time series: first order characterization and order determination', *Biometrika* **77**, 669–687.
- Brockwell, P.J. & Davis, R.A. (1991), *Time Series: Theory and Methods*, Springer, New York.
- Dahlhaus, R., Neumann, M. H. & von Sachs, R. (2000). 'Nonlinear wavelet estimation of time-varying autoregressive processes', *Bernoulli*.
- Darius, P., Ottoy, J.-P., Solomin, A., Thas, O., Raeymaekers, B. & Michiels, S. (n.d.), A collection of applets for visualizing statistical concepts, KU Leuven.
- Franke, J., Kreiss, J.-P., Mammen, E. & Neumann, M. H. (1998). 'Properties of the nonparametric autoregressive bootstrap'. Discussion Paper 54/98, SFB 373, Humboldt University, Berlin.

- Härdle, W. & Tsybakov, A. (1997), 'Local polynomial estimators of the volatility function in nonparametric autoregression', *Journal of Econometrics* **81**, 223–242.
- Härdle, W., Kerkycharian, G., Picard, D. & Tsybakov, A. (1998) *Wavelets, Approximations, and Statistical Applications*, Springer, Heidelberg.
- Härdle, W., Klinke, S. & Müller, M. (1999), *XploRe -The Statistical Computing Environment*, Springer, New York.
- Nakano, J. (1997), 'An internet user interface for statistical software', in 'Bulletin of the International Statistical Institute', Vol. 2, 35–36.
- Nakano, J. (1998), 'Graphical user interface for statistical software using internet', in R. Payne & P. Green, eds, 'COMPSTAT 1998 Proceedings in Computational Statistics', 407–412.
- Nakano, J. & White, K. (1997), 'WWW interface of the statistical package shazam', in 'Proceedings of the Institute of Statistical Mathematics', Vol. 45, Institute of Statistical Mathematics, 41–47.
- Nakano, J. & Yamamoto, Y. (1997), 'An object oriented graphical user interface for time series analysis', in 'Bulletin of the International Statistical Institute, 51st session, Contributed papers', Vol. 2, 449–450.
- Silverman, B. (1986), *Density estimation for Statistics and Data Analysis*, Chapman and Hall, London.
- Tjøstheim, D. & Auestad, B. (1994), 'Nonparametric identification of nonlinear time-series - selecting significant lags', *Journal of the American Statistical Association* **428**, 1410–1419.
- Tschernig, R. & 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. & Tschernig, R. (1999), 'Multivariate bandwidth selection for local linear regression', *Journal of the Royal Statistical Society, Series B*, **61**, 793–815.
- Yao, Q. & Tong, H. (1994), 'On subset selection in non-parametric stochastic regression', *Statistica Sinica* **4**, 51–70.