2007-07-11

A Modified Cluster-Weighted Approach to Nonlinear Time Series

Mark Ballatore Lyman
Brigham Young University - Provo

A MODIFIED CLUSTER-WEIGHTED MODEL APPROACH TO NONLINEAR TIME SERIES

by

Mark B. Lyman

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

August 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Mark B. Lyman

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	H. Dennis Tolley, Chair
Date	Scott D. Grimshaw
Date	John S. Lawson

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Mark B. Lyman in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

H. Dennis Tolley
Chair, Graduate Committee

Accepted for the Department

Scott D. Grimshaw
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

A MODIFIED CLUSTER-WEIGHTED MODEL APPROACH TO NONLINEAR TIME SERIES

Mark B. Lyman

Department of Statistics

Master of Science

In many applications involving data collected over time, it is important to get timely estimates and adjustments of the parameters associated with a dynamic model. When the dynamics of the model must be updated, time and computational simplicity are important issues. When the dynamic system is not linear the problem of adaptation and response to feedback are exacerbated. A linear approximation of the process at various levels or "states" may approximate the non-linear system. In this case the approximation is linear within a state and transitions from state to state over time. The transition probabilities are parametrized as a Markov chain, and the within-state dynamics are modeled by an AR time series model. However, in order to make the estimates available almost instantaneously, least squares and weighted least squares estimates are used. This is a modification of the cluster-weighted models proposed by Gershenfeld, Schoner, and Metois (1999). A simulation study compares the models and explores the adequacy of least squares estimators.

CONTENTS

CHAPTER

TABLES

Table

FIGURES

Figure

1. INTRODUCTION

One of the principal tools for modeling stochastic processes is the linear systems approach. Linear systems, or processes, are used in economic time series analysis, forecasting and prediction, stochastic system identification, and stochastic control. There has been considerable success in applying the linear systems approach to real world applications. One reason for this success is, arguably, the mathematical and statistical properties of linear systems modeling. Such questions as bias, prediction error, stability, and, more recently, robustness have all received rigorous attention in the literature. Computational tools to estimate the parameters of a linear system and to assess the order of the process are readily available to the user. Consequently, most of the statistical problems of linear systems may be viewed as either resolved or as having received a unifying rigorous treatment upon which computational tools and examples of implementation will soon emerge.

Nonlinear systems have also received considerable attention in recent years. While many of the major issues of linear systems are identified and most of them resolved, such is not the case of nonlinear systems. Indeed, modeling and implementing linear systems has been so successful that one major method of solving a nonlinear system is to use a linear approximation. One method of using linear system methods to approximate the nonlinear problem is to consider the process at various local levels to be linear. In this thesis the local region used for the linear approximation is described as a "state."[1] Thus, the process is viewed as linear within a state, and it transitions from state to state over time. Such a process is then modeled as a linear time series locally. Figure 1.1 contains an example of a nonlinear system with the two states shown.

[1] Note that this use of state is consistent with the common statistical use but differs from systems engineering where state refers to a different condition.

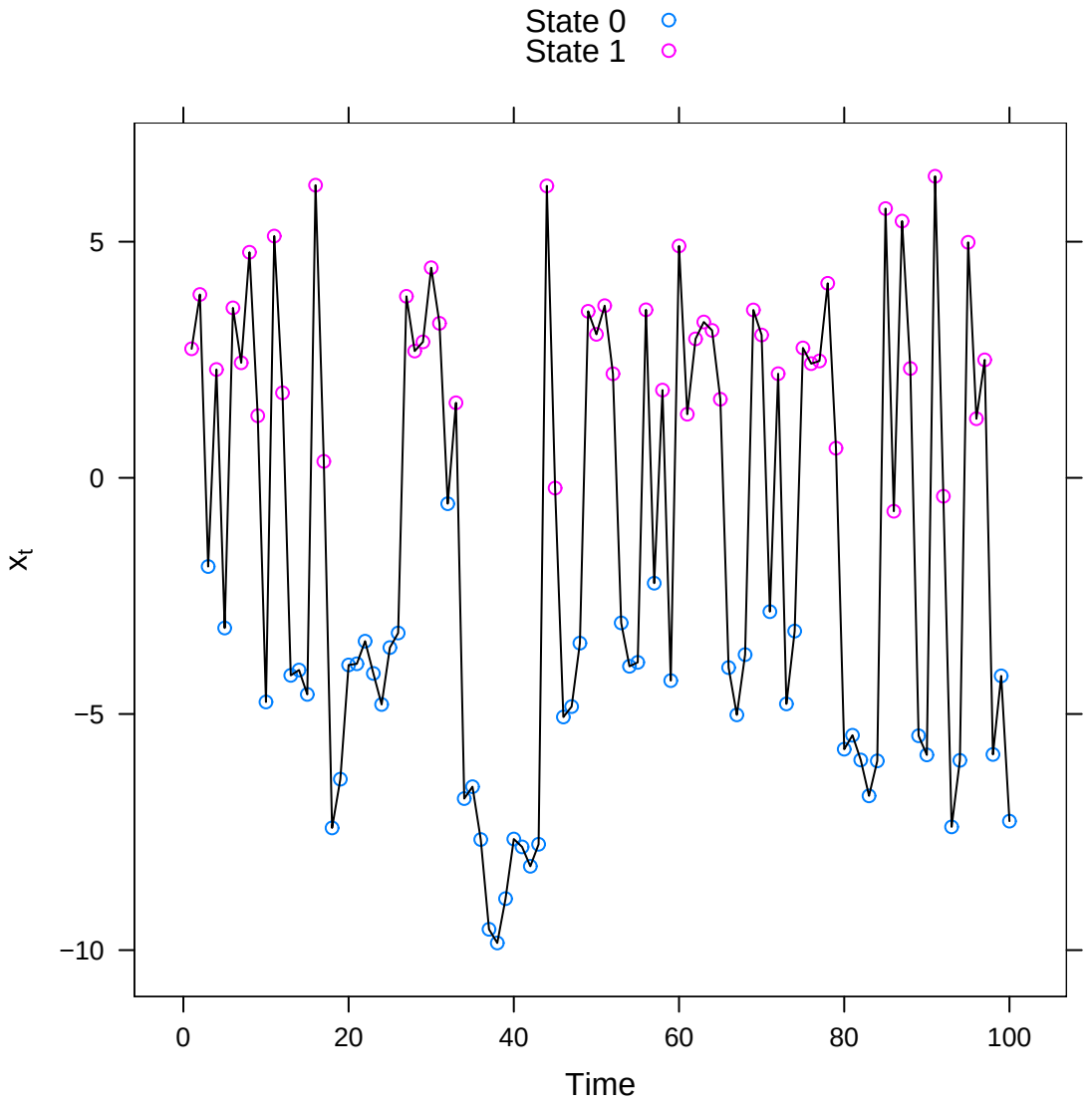

Figure 1.1: Simulated nonlinear time series data with states shown.

This approach is often especially valuable because the states have worth beyond serving as a tool in modeling. For example, it is quite common to model stock returns with a time series model, but estimates of the model parameters, and any subsequent predictions, might benefit if the state of the market, "bear" or "bull," was known or could be predicted. In this example, knowledge of the state and of the system dynamics is useful. It is also apparent that the ability to predict the state along with the value of the return would greatly enhance the worth of this model.

In many applications involving data sampled across time, it is important to get estimates of the system and adjustments of the parameters associated with the dynamic model in a timely manner. For example, in stochastic control, the manner in which feedback is used requires quick decisions. When the dynamics of the model must be updated, time and computational simplicity are important issues. When the dynamic system is nonlinear the problems of adaptation and response to feedback are exacerbated.

1.1 Modified Cluster-Weighted Model (MCWM)

This thesis introduces a likelihood-based model, the modified cluster-weighted model (MCWM), that incorporates the local linear process and the state-jumping process. The linear dynamic process applies while the process remains in the same state. The jumping process defines the transition from one state, k_t, to another state over time, which is dictated by the nonlinear process x_t and an additional observed random variable y_t. This jumping process is assumed to follow a Markov chain. The contribution to the likelihood of one observation, k_t, conditional on k_{t-1} and x_{t-1}, for these transitions is shown in Equation 1.1, where the free transition probabilities are $\boldsymbol{\theta} = \left(\theta_{11}, \ldots, \theta_{1i}, \ldots, \theta_{(i-1)i}\right)$ for $i = (1, \ldots, m)$, where m is the number of states and θ_{ij} is the probability of transitioning from state i to state j. The transitions depend on the exogenous variable y_t and x_t through a logistic model of the transition

probabilities shown in Equation 1.2, where $\boldsymbol{\beta}$ is the vector of all β parameters.

$$L\left(\boldsymbol{\theta}; k_t | k_{t-1}, x_{t-1}, y_{t-1}\right) = \theta_{k_{t-1}k_t}^{I(k_t \neq 0)} \left(1 - \sum_{i=1}^{m} \theta_{k_{t-1}i}\right)^{I(k_t=0)} \tag{1.1}$$

$$\begin{aligned} L\left(\boldsymbol{\beta}; k_t | k_{t-1}, x_{t-1}, y_{t-1}\right) = & \left\{ \frac{\exp\left(\beta_0^{k_{t-1}k_t} + \beta_1^{k_{t-1}k_t} x_{t-1} + \beta_2^{k_{t-1}k_t} y_{t-1}\right)}{1 + \sum_{i=1}^{m} \exp\left(\beta_0^{k_{t-1}i} + \beta_1^{k_{t-1}i} x_{t-1} + \beta_2^{k_{t-1}i} y_{t-1}\right)} \right\}^{I(k_t=0)} \\ & \times \left\{ \frac{1}{1 + \sum_{i=1}^{m} \exp\left(\beta_0^{k_{t-1}i} + \beta_1^{k_{t-1}i} x_{t-1} + \beta_2^{k_{t-1}i} y_{t-1}\right)} \right\}^{I(k_t \neq 0)} \end{aligned} \tag{1.2}$$

The within-state dynamics, conditional on k_t, are modeled by an autoregressive (AR) time series model. Equation 1.3 is the contribution to the likelihood of one observation from the nonlinear process, x_t, where $\boldsymbol{\omega} = (\boldsymbol{\phi}, \boldsymbol{\sigma}^2)$. The form of the model is the same for each state; the actual parameters (both AR coefficients and variance), and possibly the model order, p_{k_t}, vary across states.

$$L\left(\boldsymbol{\omega}; x_t | x_{t-1}, \ldots, x_{t-p}; k_t\right) = \left(2\pi\sigma_{k_t}^2\right)^{-1} \exp\left\{ \frac{\left(x_t - \phi^{k_t} + \sum_{i=1}^{p_{k_t}} \phi_i^{k_t} x_{t-i}\right)^2}{2\sigma_{k_t}^2} \right\} \tag{1.3}$$

The product of the AR model and the Markov chain model results in a two-stage joint likelihood for the observed time series data and the observed y_t values (see Equation 1.4, where n^* is the maximum order of the m states, and n is the number of observations from the nonlinear process). By estimating the parameters of the transition probabilities, we can also predict the state of future observations as we would make a prediction with any logistic regression model.

$$\begin{aligned} L\left(\boldsymbol{\omega}, \boldsymbol{\theta} | k_1, \ldots, k_{n^*}, x_1, \ldots, x_{n^*}\right) = & \prod_{n^*}^{n} L\left(\boldsymbol{\omega}; x_t | x_{t-1}, \ldots, x_{t-p}; k_t\right) \\ & \times L\left(\boldsymbol{\beta}; k_t | k_{t-1}, x_{t-1}\right) \end{aligned} \tag{1.4}$$

1.2 MCWM Estimators

Parameter estimates can be obtained using nonlinear maximization algorithms, like the Expectation-Maximization (EM) algorithm; however, such methods cannot provide estimates of these parameters fast enough for many real-time applications. Therefore, a method of moments-type procedure that contains a first-order likelihood correction term makes the estimates available almost instantaneously. The model is broken into pieces that can be fitted using real-time parameter estimates and updates from least squares and weighted least squares. A one-step adjustment to the estimates based on the likelihood is then implemented to establish asymptotic efficiency. However, simulation results indicate that the adjustment is not helpful and, in some cases, results in worse estimates. These non-iterative methods provide a way to estimate the MCWM in a fixed amount of time, whereas the iterative methods converge in an unknown number of iterations.

Chapter 2 contains a review of recent literature on nonlinear systems, including a discussion of cluster-weighted models and functional-coefficient regression models. Also, recent literature on the properties of various time series estimators is discussed. Chapter 3 describes the simulations that show the usefulness of the modified cluster-weighted model and the adequacy of non-iterative estimation methods based on two computer simulations. Chapter 4 contains the results of the simulations discussed in Chapter 3, and Chapter 5 contains conclusions and suggestions for future research.

2. LITERATURE REVIEW

The mathematical and statistical properties of linear systems have been thoroughly presented in standard textbooks by authors such as Box, Jenkins, and Reinsel (1994), Pandit and Wu (1983), and Brockwell and Davis (1983). The properties of nonlinear processes are much less well-known. However, Fan and Yao (2003) and Tong (1990) provide a good discussion of many commonly used approaches to modeling such processes.

2.1 Cluster-Weighted Model (CWM)

One approach to modeling non-linear time series is a cluster-weighted model (CWM) (Gershenfeld, Schoner, and Metois 1999). A CWM is a mixture model in which the distributions of the sub-populations are described by a time series. Let the density of the time series, x, be written as in Equation 2.1, where $p(\boldsymbol{x}|c_m)$ is multivariate normal, $N(\boldsymbol{\mu}_m, \boldsymbol{C}_m)$, $p(c_m)$ is the weight of the cluster, and M is the number of clusters. The parameter, M, is generally determined by cross-validation to control under- and over-fitting (Schoner, Cooper, Douglas, and Gershenfeld 1999). In this thesis, the number of states is known for both the CWM and the MCWM, but cross-validation can be used for the MCWM as well. The mean vectors $\boldsymbol{\mu}_m$ are usually a function of lagged values of $\boldsymbol{x}$. Gershenfeld et al. (1999) proposed these models as a way of fitting linear models to an overall non-linear system because fewer parameters could be used to model a system to achieve a given error. Estimates are computed iteratively. Gershenfeld et al. (1999) use an expectation-maximization algorithm to find the cluster weights and parameter estimates.

$$p(\boldsymbol{x}) = \sum_{m=1}^{M} p(\boldsymbol{x}, c_m) = \sum_{m=1}^{M} p(\boldsymbol{x}|c_m)p(c_m) \tag{2.1}$$

As noted previously, it is often desirable to predict the state or cluster of future observations; however, CWMs lack a method for predicting the cluster of future observations. The state of an observation is only indirectly related to the state of a previous observation through the process variable. The clusters are used only as a tool for modeling a complex system; thus, any interpretation of the clusters themselves is largely coincidental.

CWMs have been applied in several areas. Uses in financial models are given by Wong and Chan (2005) and Ferreira et al. (2003). Schoner et al. (1999) demonstrate uses in acoustics, and He et al. (2002) use a CWM to model heart rates. Ferreira et al. (2003) and Wong and Chan (2005) also compare CWMs with several other approaches to modeling non-linear time series. Anandamohan and Ram (2005) show the strengths of the cluster-weighted model in complex driven systems.

2.2 Hidden Markov Models

A model similar to CWMs is a Markov-switching model. A Markov-switching model is an extension of hidden Markov models in which the conditional distribution of a random variable depends not only on the unobserved state but also on previous realizations of the random variable (Cappé, Moulines, and Rydén 2005). As with ordinary hidden Markov models, the unobserved state is a Markov chain. Parameters are estimated with an iterative algorithm similar to the EM algorithm. Hamilton (1989) proposed the use of these models for modeling the U.S. business cycle. For example, the quarterly GNP comes from one of two autoregressive processes, depending on the state of the economy (expansion or contraction). Markov-switching models, unlike CWMs, allow prediction and interpretation of states, but the transitions between

states do not depend on the linear processes within the states or on an exogenous variable, like they do in the MCWM.

Cox (1981) discusses two similar model types, those that are "observation-driven" and those that are "parameter-driven." Zeger (1988) describes a type of "observation-driven" model, Markov regression models. In Markov regression models, the distribution of the observations at time t is determined by a linear model of a function of observations at previous times. Unlike the MCWM, the parameters are constant; that is, the observations are always related to past observations in the same manner.

The parameter-driven models, on the other hand, do not relate the observations to past observations directly. In these models, the parameters of the observations are related to past parameter values. Thus, observations are related only through the parameter process. Keenan (1982), Azzalini (1982), and Zeger (1988) describe some examples of parameter-driven models.

2.3 Functional Coefficient Models

Functional-coefficient regression models are another development in the area of nonlinear time series (Cai, Fan, and Yao 2000). Functional-coefficient models are a rich set of models in which the coefficients of the model are functions rather than constants. Included in this general model framework are functional-coefficient autoregressive models (Chen and Tsay 1993), threshold autoregressive models (Tong 1990), exponential autoregressive models (Haggan and Ozaki 1981; Ozaki 1982), and regression with random coefficients (Granger and Teräsvirta 1993). In their 2000 paper, Cai et al. developed many asymptotic properties of these models. Harvill and Ray (2005) study the use of these models in forecasting.

2.4 Time Series Estimation

As mentioned above, one of the goals of fitting models in this fashion is to obtain estimates of the parameters and get predictions of future observations quickly. Most applications of the CWM use the EM algorithm to estimate model parameters, and, while the EM algorithm generally performs well, there is no guarantee it will converge in a small number of iterations. Spitzer (1979) and Alpargu and Dutilleul (2001) extensively compared several estimation methods for AR(1) models. Least squares estimators performed significantly worse than maximum likelihood and Yule-Walker estimators only for highly correlated series. With autocorrelations between -0.5 and 0.5, least squares estimators actually had a better Mean-Square Error (MSE) than maximum likelihood and Yule-Walker estimators. Only when autocorrelations were greater than 0.9 was the MSE for least squares twice as large as that of Yule-Walker and maximum likelihood estimators.

2.5 Root-n Adjustment

While iterative methods are potentially too slow for real-time applications, the convergence properties these methods provide are desirable. In cases where correlation in the random variable over time is bounded away from 1 and a uniform mixing condition applies, least squares estimates of a fixed finite set of parameters is $\sqrt{(}n)$ consistent. The least squares estimates are assumed to be $\sqrt{(}n)$ consistent by adjusting least squares estimates, $\tilde{\boldsymbol{\theta}}_n$, by the score function, $S\left(\tilde{\boldsymbol{\theta}}_n\right)$, and the Fisher information, $I^{-1}\left(\tilde{\boldsymbol{\theta}}_n\right)$, as in Equation 2.2. If the likelihood is regular, then $\sqrt{n}(\boldsymbol{\delta}_n - \boldsymbol{\theta}) \xrightarrow{L} N\left(0, 1/I(\boldsymbol{\theta})\right)$. Equation 2.2 is clearly the first step of the Newton-Raphson method with the least squares estimator as a starting value. By only using the first step, there is little time penalty, but because the likelihood of the within-state dynamics is regular, it ensures that our estimator converges at least at a rate of

$1/\sqrt{n}$. This is ideal if the within-state dynamics are of primary importance. Lehmann (2001) discusses this adjustment and proves the convergence property.

$$\boldsymbol{\delta}_n = \tilde{\boldsymbol{\theta}}_n - S\left(\tilde{\boldsymbol{\theta}}_n\right) I^{-1}\left(\tilde{\boldsymbol{\theta}}_n\right) \tag{2.2}$$

3. METHODOLOGY

3.1 Comparison of Time Series Estimators

Because many applications require nearly instantaneous results, one major objective was to develop quick methods of parameter estimation. A simulation study and theoretical results from Shaman and Stine (1988) illustrate the adequacy of the least squares method compared to maximum likelihood and Yule-Walker methods of estimating autoregressive model parameters in a standard AR model. Maximum likelihood estimation was performed using a quasi-Newton-Raphson algorithm implemented in the `R` function `optim` (R Development Core Team 2006). The data were simulated using the `R` function `arima.sim` (R Development Core Team 2006), from a Gaussian distribution with variance of 1. The settings for the simulation were comprised of sample sizes of 20, 50, and 100 with all possible permutations of coefficients -0.9, -0.5, -0.2, 0.2, 0.5, 0.9 that provide a stationary series for an AR(p) series where $p = 1, 2$. Each combination of parameters and sample sizes was replicated 1000 times. The least squares estimators were compared, analytically for large series and by simulation for practical finite samples. The criteria for comparison were MSE and bias of the parameter estimates and bias of one-step-ahead predictions.

3.2 Modified Cluster Weighted Model Estimators

A simulation of an MCWM with two states, state 0 and state 1, and a Markov chain modeling the probability of a transition between them was done to investigate the MCWM and its estimators (see Equation 1.4). The two local dynamic processes are AR(p) processes with $p = 1, 2$. The same sample sizes, except for 20, and coefficient values as in the pure AR simulation comprised the settings for this simulation

as in the pure AR simulation. Samples of size 20 were excluded because all of the simulated observations too often fell into one state. However, there were two series to permute the coefficients between. The innovations for both processes were normally distributed with mean 0 and variance 1. The series in state 0 was centered at -3 and the series in state 1 was centered at 3. Thus, the two series overlapped slightly in the tails, but they were still separate enough to distinguish between them.

The parameters for the transition probabilities were chosen such that, on average, a change in state will occur every 10 observations ($\theta_{11} = 10/11, \theta_{01} = 1/11$) and every 30 observations ($\theta_{11} = 30/31, \theta_{01} = 1/31$) when the process variable and the exogenous variable are equal to their respective means. The coefficients for the process itself were set equal to the intercepts so that both would have equal weight in determining transitions. The coefficient relating the exogenous process y_t was set to be 1.5 times as large as the x_t coefficients. For example, the parameters determining θ_{01} (β_0^{01}, the intercept; β_1^{01}, the parameter relating the process to the state transition; and β_2^{01}, the parameter relating the exogoneous process to the state transition) are determined as follows (see Equation 3.1). Setting the process and the exogoneous process to their means, $x_{t-1} = -3$ and $y_{t-1} = 0$,

$$\begin{aligned} \log \frac{\theta_{01}}{1-\theta_{01}} &= \exp\left(\beta_0^{01} + \beta_1^{01} x_{t-1} + \beta_2^{01} y_{t-1}\right), \\ \log \frac{0.09}{0.91} &\approx 1.15 + 1.15(-3) + 1.73(0). \end{aligned} \tag{3.1}$$

The parameters were estimated under the assumption that the CWM and the MCWM fit with four different methods: least squares, least squares with first order likelihood correction, Yule-Walker, and maximum likelihood. The number of states was known in estimating the parameters, but the state of each observation was unknown. Initial estimates of state membership were obtained using the k-means clustering. The first p observations were assumed to be known, both value and

state, analogous to the common practice for least squares estimators of time series parameters. The criteria for comparison of all five methods was the proportion of misclassified states, MSE and bias of the estimators, and bias of the one-step-ahead prediction, where the final three criteria will assume the correct state was estimated. The simulation results showed that the least squares estimates perform adequately when compared with the iterative maximum likelihood method and that the MCWM performs well when compared to the CWM and allows a prediction of a future observation.

3.2.1 Non-iterative Methods

The two least squares methods estimate the AR coefficients with ordinary least squares computed separately based on the state membership of the observation at time t. Thus, the autoregressive parameter estimates are the solution to the normal equations in Equation 3.2, and the variance of x_t was estimated by the mean-square error. For convenience, a weight matrix is used with an indicator function of membership in a given state as the weight. For $i = 0, \ldots, m-1$ and $t = p_{k_t} + 1, \ldots, n$, where $\boldsymbol{X}$ is the model matrix formed with a column of ones, and the observations lagged one step and, if appropriate, the observations lagged two steps. And $\boldsymbol{x}_t$ contains observations $p_{k_t} + 1, \ldots, n$ from the process,

$$\boldsymbol{X}_{t-1}^T \text{diag}\left(I\left(k_t = i\right)\right) \boldsymbol{X}_{t-1} \boldsymbol{\phi} = \boldsymbol{X}_{t-1}^T \text{diag}\left(I\left(k_t = i\right)\right) \boldsymbol{x}_t. \tag{3.2}$$

The estimates of the logistic model coefficients in these methods were found using weighted least squares with the binary response of membership of the observation at time t in state 1 or state 0. The estimates were computed separately based on the observation's state at time $t-1$. First, starting values are estimated with ordinary least squares as in Equation 3.3, where $\boldsymbol{k}_t$ is the state membership at time t

for observations $p_{k_t}+1,\ldots,n$. A weight matrix is used for convenience. Then, using the solution $\hat{\boldsymbol{\beta}}_i^*$, $\hat{\pi}_i$ is calculated as in Equation 3.4. Next, the working vector, $\boldsymbol{y}_i$ is calculated, Equation 3.5. Finally, the logistic model coefficients are the solutions of the normal equations in Equation 3.7 using the working vector and appropriate weights (see Section A.5).

$$\hat{\boldsymbol{\beta}}_i^* = \left(\boldsymbol{X}_{t-1}^T \text{diag}\left(I\left(\boldsymbol{k}_{t-1}=i\right)\right)\boldsymbol{X}_{t-1}\right)^{-1}\boldsymbol{X}_{t-1}^T \text{diag}\left(I\left(\boldsymbol{k}_{t-1}=i\right)\right)\boldsymbol{k}_t \tag{3.3}$$

$$\hat{\pi}_i = \frac{\exp\left(\hat{\boldsymbol{\beta}}^T\boldsymbol{X}_{t-1}\right)}{1+\exp\left(\hat{\boldsymbol{\beta}}^T\boldsymbol{X}_{t-1}\right)} \tag{3.4}$$

$$\boldsymbol{y}_i = \hat{\pi}_i + \frac{\boldsymbol{k}_t-\hat{\pi}_i}{\hat{\pi}_i\left(1-\hat{\pi}_i\right)} \tag{3.5}$$

$$\boldsymbol{W}_i = \text{diag}\left(\hat{\pi}_i\boldsymbol{y}_i\right) \tag{3.6}$$

$$\hat{\boldsymbol{\beta}}_i = \left(\boldsymbol{X}_{t-1}^T\boldsymbol{W}_i\boldsymbol{X}_{t-1}\right)^{-1}\boldsymbol{X}_{t-1}^T\boldsymbol{W}_i\boldsymbol{y}_i \tag{3.7}$$

The other non-iterative methods use the above calculations with slight differences. The first-order likelihood correction of the Root-n method is one iteration of the Newton-Raphson algorithm. The score function and the Fisher Information are evaluated at the least squares parameter estimates. The product of the score function and inverse Fisher Information is then subtracted from the vector of least squares parameter estimates (see Equation 2.2 and Appendix A.7). The Yule-Walker method is similar to the least squares method, except the AR parameters are estimated using the Yule-Walker equations (see Appendix A.6).

3.2.2 Iterative Methods

The MCWM maximum likelihood method and the CWM were both fitted using the EM algorithm. Least squares estimates were used as initial parameter estimates. The cluster-weighted algorithm was implemented as described by Gershenfeld et al. (1999) (see Appendix A.3). The steps of the EM algorithm for the MCWM are as follows (see Appendix A.4):

(1) Estimate $\hat{P}(k_t = i) = \frac{p(\hat{\boldsymbol{\theta}}^0|k_t=i)}{p(\hat{\boldsymbol{\theta}}^0|k_t=1)+p(\hat{\boldsymbol{\theta}}^0|k_t=2)}$.

(2) Set $\hat{k}_t = \hat{P}(k_t = i)$

(3) Maximize the likelihood for each set of parameters using the least squares methodology with the updated probabilities of state membership as weights.

(4) Repeat steps 1–3 until likelihood converges.

3.2.3 Shock Adjustment

It seems logical that the nonlinear process experiences a shock following a transition. The shock is strongest immediately following a transition to a state and decreases as long as the process remains in that state. An adjustment is implemented in order to correct for this shock, as shown in Equation 3.8:

$$z_t^i = x_t^i - \hat{\delta}_i^{q_t} * s_{ij}, \tag{3.8}$$

where x_t^i is the observation at time t in state i, z_t^i is the corresponding adjusted observation, q_t is the number of consecutive observations in state i up to time t, and $s_{ij} = \bar{x}_j - \bar{x}_i$ is the shock estimate of an observation following a transition from state i to state j. $\hat{\delta}_i$ is the estimated coefficient of the AR(1) model, $(x_t^i - \bar{x}_i) = \delta_i(x_{t-1}^i - \bar{x}_i) + \epsilon$; that is, the response is the observation at time t that falls in state i and x_{t-1}^i is the preceding observation regardless of state. All of the methods described above use data that is adjusted for such a shock. Initial state membership is determined by k-means clustering.

3.2.4 Computer Algorithms

All of the algorithms described above were implemented using `R`. Because the states are not completely independent of each other, the parameters could not be

estimated by simply separating the observations into states and estimating the parameters independently. While current software can handle missing values in time series, the values in the other states are not truly missing and have some affect on the parameters of the state in question. To estimate the MCWM, some values must be used to estimate parameters but not be considered part of the current series of interest. Thus, it was necessary to write functions for performing all of the above methods, including least squares and Yule-Walker methods. The algorithms were written specifically to handle the two-state situation. Of course, they could be modified to handle a greater number of states.

The root-n method uses the analytical score vector and information matrix because this gives the algorithm a "better" chance of succeeding than using numerical approximations. Thus, the root-n methods failures can not be attributed to poor approximations of the score function and Fisher Information.

4. RESULTS

The data from both simulations is available on a DVD for reproduction of the results or for further investigation.

4.1 Time Series Estimators

The first simulation compares the maximum likelihood, Yule-Walker, and least squares estimators of the standard linear AR(p) model. Shaman and Stine (1988) developed an algorithm for deriving the T^{-1} order bias of least squares autoregressive estimators as a function of the true parameters, where T is the series length. For a first-order AR model, the bias of the least squares estimator is $\frac{(1-3\phi_1)}{T}$ and, for a second-order AR model, the biases for the first and second coefficients are $\frac{1-\phi_1-\phi_2}{T}$ and $\frac{2-4\phi_2}{T}$, respectively. Table 4.1 and 4.2 contain the bias estimates for each combination of autoregressive parameters used in the simulations for the first- and second-order models respectively. The least squares estimators are less biased when the true autoregressive parameters are positive or near zero. Some of the bias estimates are large enough to be worrisome, especially the negative parameters and those with large absolute values.

ϕ_1	Bias$\left(\hat{\phi}_1\right)$
-0.90	3.70
-0.50	2.50
-0.20	1.60
0.20	0.40
0.50	-0.50
0.90	-1.70

Table 4.1: AR(1) Least Squares Estimator $Bias \times T$

The results of the simulation comparing the maximum likelihood, Yule-Walker,

ϕ_1	ϕ_2	Bias$\left(\hat{\phi}_1\right)$	Bias$\left(\hat{\phi}_2\right)$
-0.90	-0.90	2.80	5.60
-0.50	-0.90	2.40	5.60
-0.20	-0.90	2.10	5.60
0.20	-0.90	1.70	5.60
0.50	-0.90	1.40	5.60
0.90	-0.90	1.00	5.60
-0.90	-0.50	2.40	4.00
-0.50	-0.50	2.00	4.00
-0.20	-0.50	1.70	4.00
0.20	-0.50	1.30	4.00
0.50	-0.50	1.00	4.00
0.90	-0.50	0.60	4.00
-0.90	-0.20	2.10	2.80
-0.50	-0.20	1.70	2.80
-0.20	-0.20	1.40	2.80
0.20	-0.20	1.00	2.80
0.50	-0.20	0.70	2.80
0.90	-0.20	0.30	2.80
-0.50	0.20	1.30	1.20
-0.20	0.20	1.00	1.20
0.20	0.20	0.60	1.20
0.50	0.20	0.30	1.20
-0.20	0.50	0.70	0.00
0.20	0.50	0.30	0.00

Table 4.2: AR(2) Least Squares Estimator $Bias \times T$

and least squares time series estimators indicate that the least squares estimators are adequate even for small sample sizes. Figures 4.1 and 4.2 contain boxplots of the absolute bias and mean-square error of each simulation. Figure 4.3 contains boxplots of the absolute bias of one-step-ahead predictions for each simulation.

The least squares method is the worst estimator of the three, but it does not appear to be significantly worse than the maximum likelihood or the Yule-Walker method. Increasing the sample size improves the performance of each of the estimators greatly. It is interesting to note that the results from the simulations seem to indicate that the bias of the estimators is closer to zero than Shaman and Stine (1988) proposed. However, there is no real difference among the methods with respect to prediction (see Figure 4.3), and increasing the sample size does not seem to alter the prediction accuracy.

4.2 Comparison of Modified Cluster Weighted Estimators

There are 12 to 14 parameters to be estimated for each of the models described in Section 3.2. There are 6 parameters determining the transition probabilities for all models but the CWM (one intercept parameter, one parameter associated with the process itself, and one parameter associated with the exogenous process for each transition probability), 6–8 parameters determining the within-state AR processes (one variance parameter and one or two autoregressive parameters for each within state process). Each dot in the boxplots of this section represents the 10% trimmed sample mean for all simulations that fit the given criteria. The trimmed mean was used because of some very large outliers (see Figure 4.5–Figure 4.9).

Figure 4.4 shows that all of the methods, except the root-n, did not change much at all by removing the outliers. The root-n method, however, changed quite a bit. Because the other methods hardly changed at all, only the root-n method had extreme outliers. Surprisingly, the bias of the least squares transition parameter

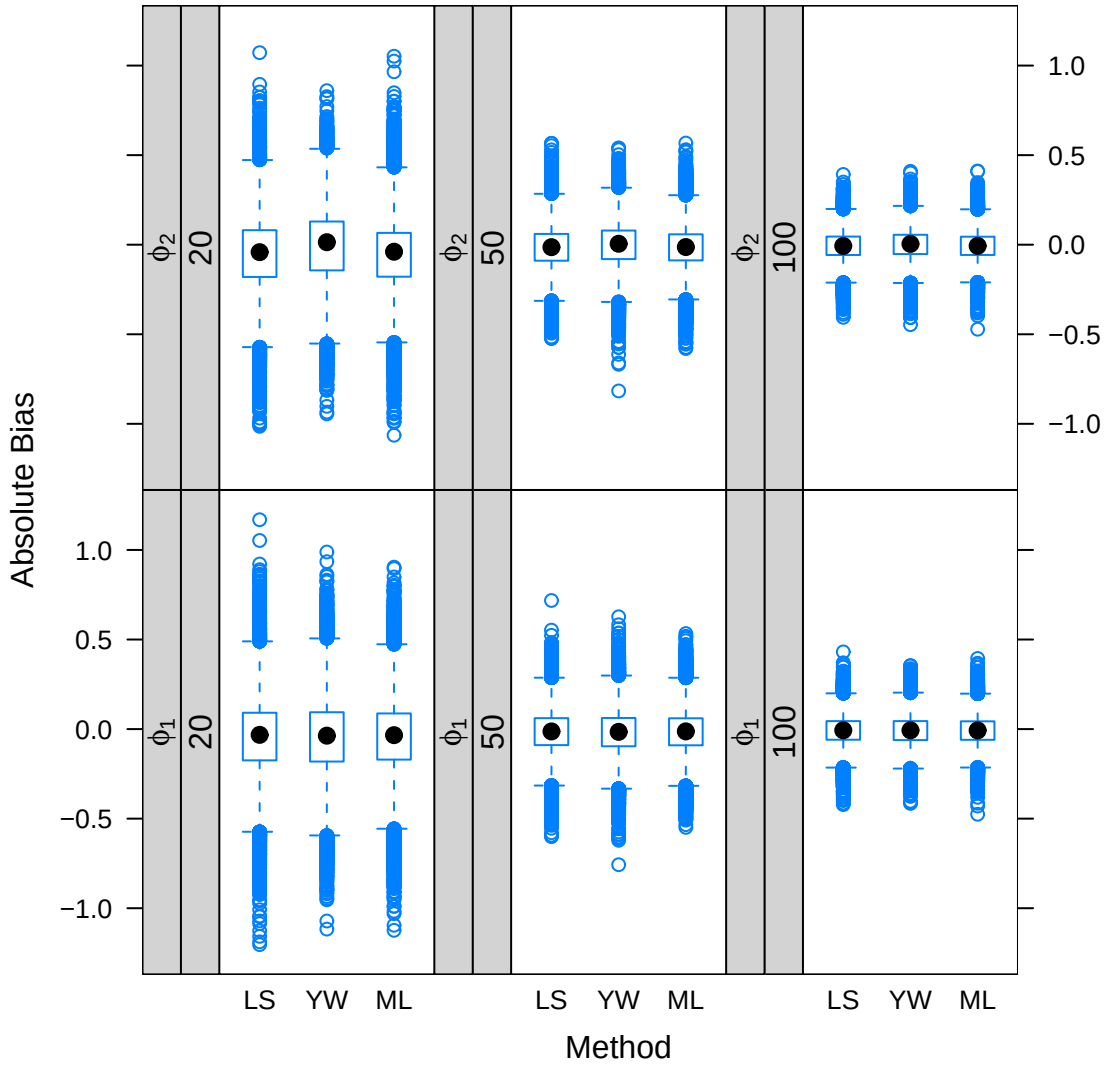

Figure 4.1: Bias for AR Model Parameters

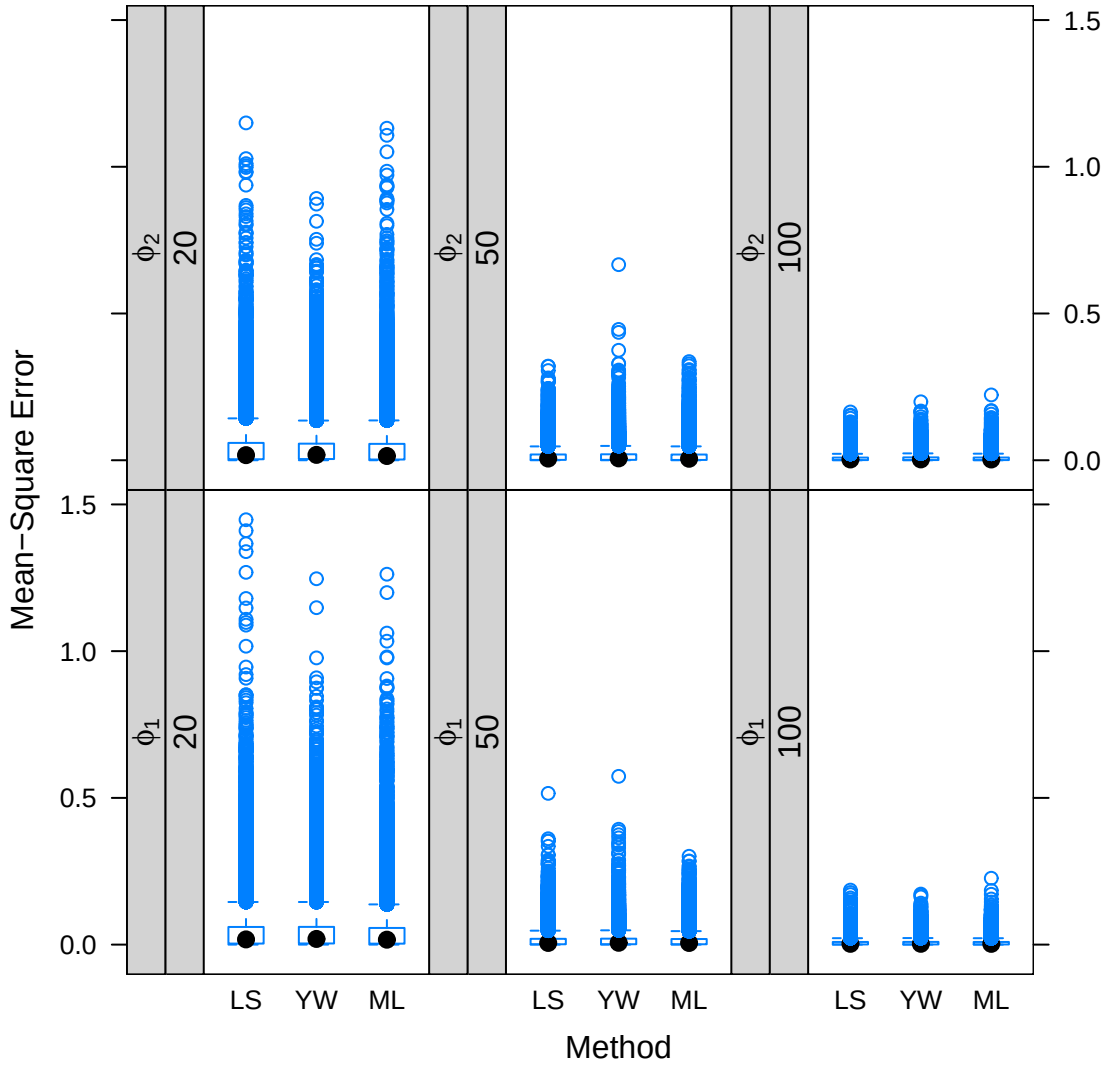

Figure 4.2: MSE for AR Model Parameters

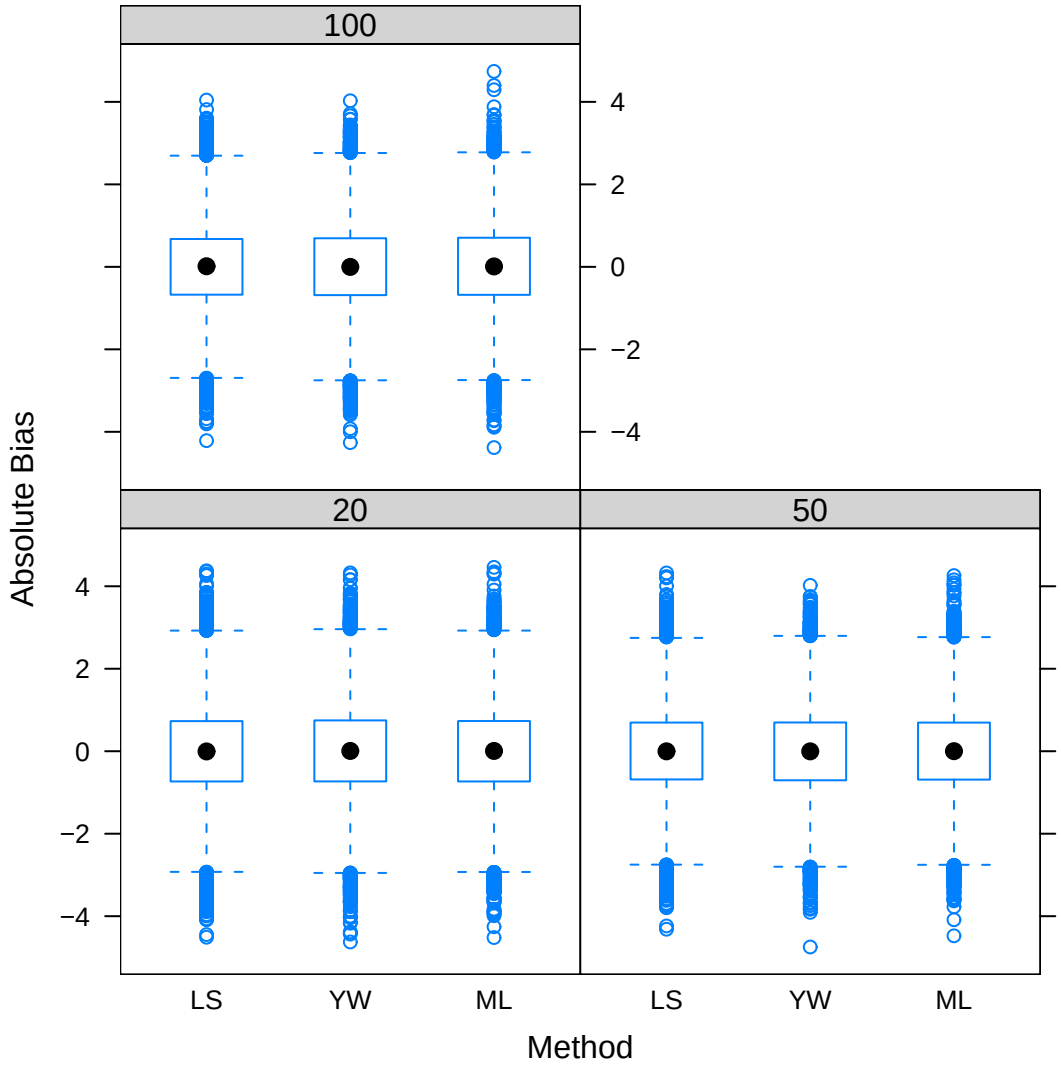

Figure 4.3: Absolute Bias for AR Model One-Step-Ahead Predictions

estimators changed less than the bias of the maximum likelihood estimators, but all of the other parameters were about the same for both methods.

4.2.1 State Transition Parameters

Figure 4.5 contains boxplot summaries of the distribution of the 10% trimmed mean absolute bias of each state transition parameter estimator by sample size, and Table 4.3 contains the average 10% trimmed mean-square errors. It is evident that all of the methods generally underestimate the state transition parameters. However, the least squares method performs more favorably than the other methods.

The mean-square error for the maximum likelihood method is often higher than the other methods. Also, the mean-square error and the absolute bias do not seem to be affected by the sample size. An increase of sample size from 50 to 100 does not improve the estimates of the transition parameters. It is unclear why this is so. Perhaps the advantages of having more observations to estimate the parameters are outweighed by the increased difficulty to correctly classify the observations (see Section 4.2.4). The problems with classification might be caused by too many transitions that accompany the increase in sample size, but there is also a problem when there are too few transitions because there are not enough observations to estimate some of the transition parameters.

4.2.2 Autoregressive Parameters

Figure 4.6 contains boxplot summaries of the distribution of the 10% trimmed mean absolute bias of each of the autoregressive parameter estimators by sample size, and Table 4.4 contains the average 10% trimmed mean-square errors. The estimators all appear to be equally biased. It is unclear why the intercept parameters are estimated so much worse than the other parameters.

Surprisingly, the root-n method does significantly worse than the other estima-

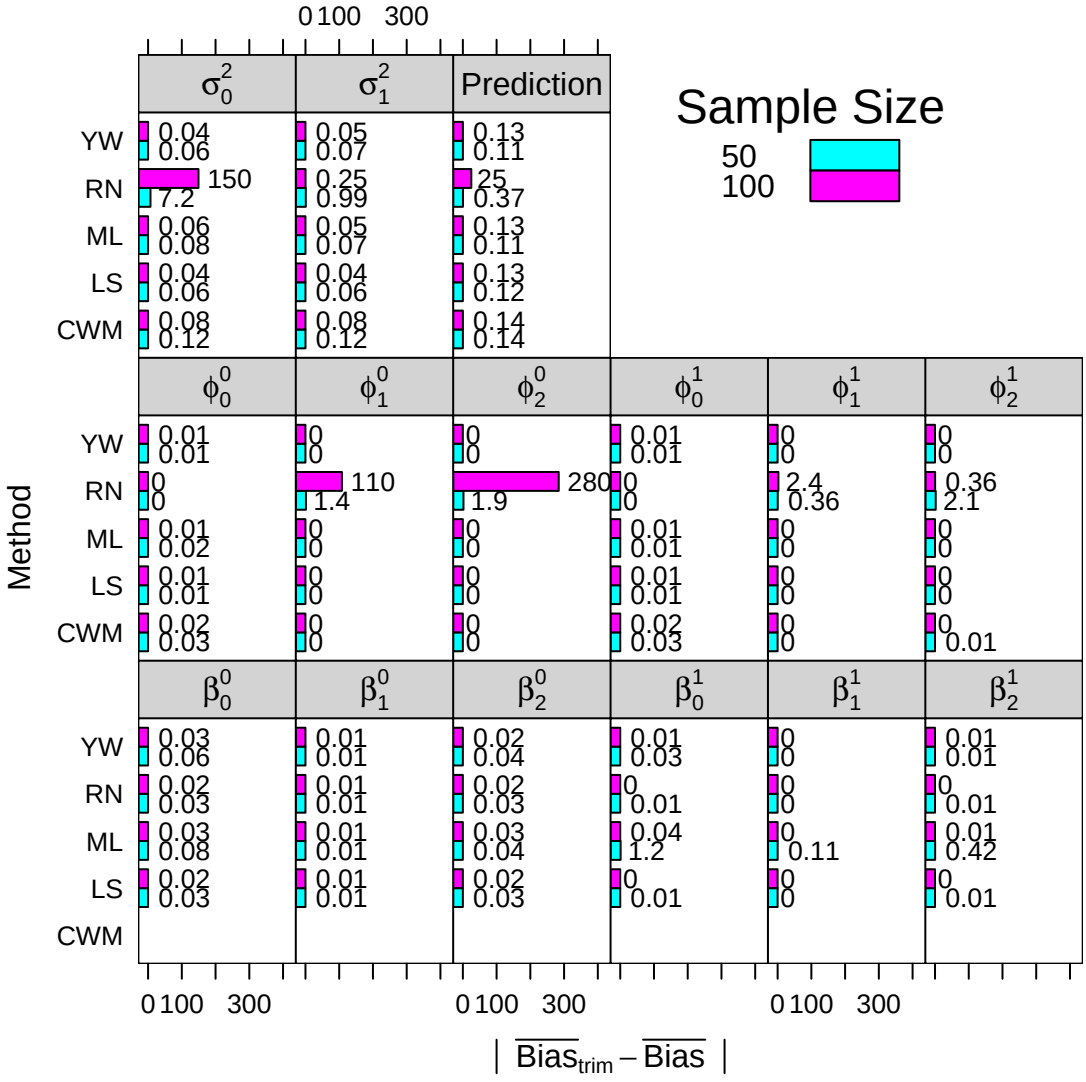

Figure 4.4: Change in Absolute Bias of Parameters

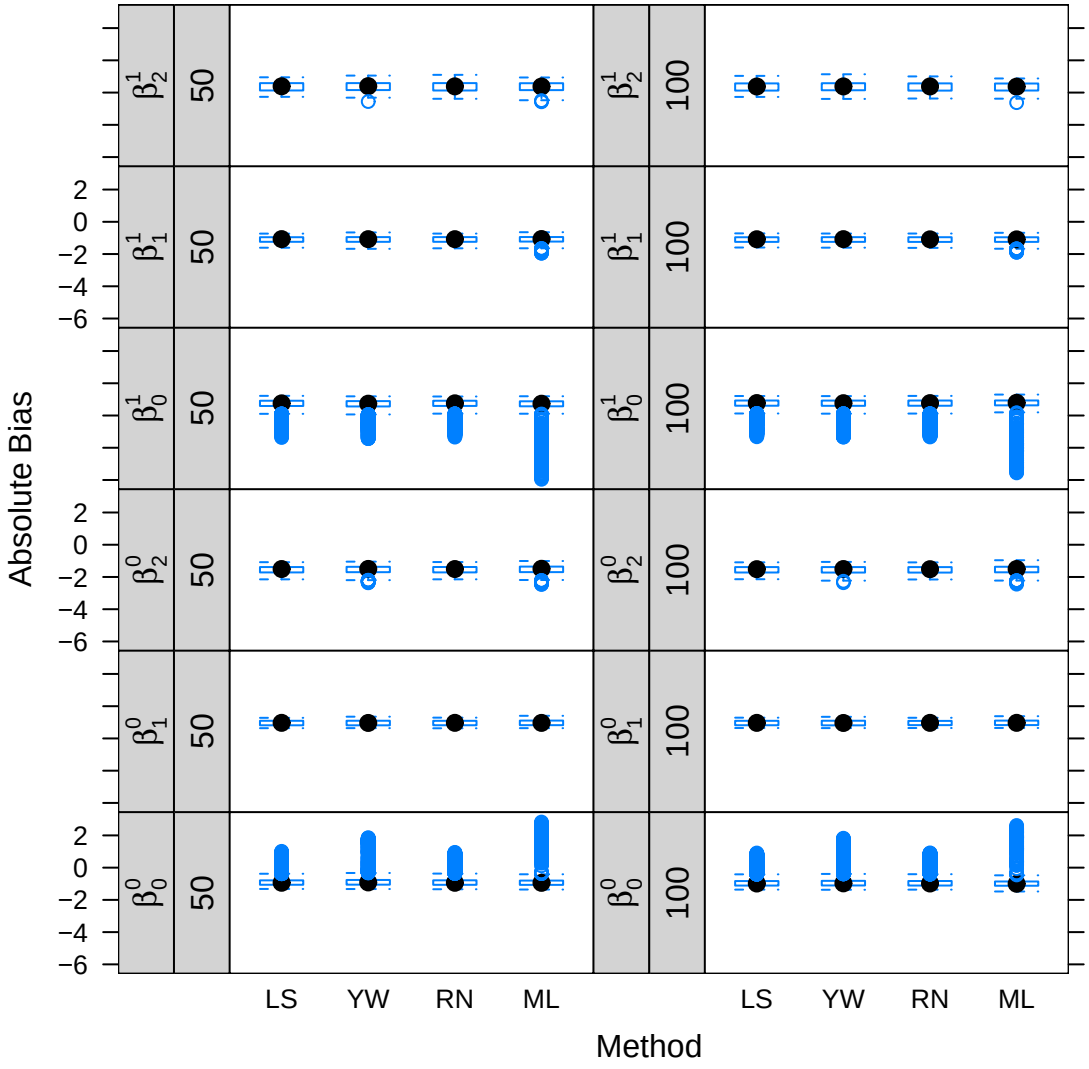

Figure 4.5: State Transition Parameter Absolute Bias—boxplots of the 10% trimmed mean absolute bias by sample size. Note: "0" superscript indicates a parameter for the probability of the process transitioning from state 0 to state 1, and a "1" superscript indicates a parameter for the probability of the process remaining in state 1.

n	parameter	LS	YW	RN	ML
50	β_0^0	0.20	0.35	0.20	0.46
100	β_0^0	0.11	0.17	0.11	0.20
50	β_1^0	0.11	0.11	0.11	0.12
100	β_1^0	0.10	0.10	0.10	0.10
50	β_2^0	0.48	0.60	0.48	0.73
100	β_2^0	0.45	0.50	0.45	0.57
50	β_0^1	0.63	0.72	0.63	1.00
100	β_0^1	0.59	0.64	0.59	0.79
50	β_1^1	0.17	0.18	0.17	0.24
100	β_1^1	0.18	0.19	0.18	0.23
50	β_2^1	0.32	0.44	0.32	0.66
100	β_2^1	0.27	0.35	0.27	0.44

Table 4.3: Transition Probability Parameters' Average Mean-Square Error

tors. It appears that the likelihood is irregular enough that the one-step adjustment actually converges to a local maximum. However, as with the transition parameter estimators, the least squares estimator has as small a bias as the other estimators.

The mean-square error for the root-n estimator is, of course, much larger than that of the other estimators, but the other methods are all very comparable. The MCWM methods, including the least squares method, estimate the parameters as well as the CWM method, indicating that the least squares MCWM estimator is generally sufficient for modeling a nonlinear process. Again, it is interesting to note that overall the sample size has little effect except that the root-n method actually performs worse with a larger sample size.

The variance of the autoregressive process is also estimated well by all of the methods, except the root-n method. In this case, the bias of the root-n method is several orders of magnitude higher than the other methods. In order to allow a graphical comparison of the remaining methods, Figure 4.7 does not contain the data from the root-n method. Again, the reader may access the raw data from the simulation contained on the included DVD. All of the plotted methods overestimate

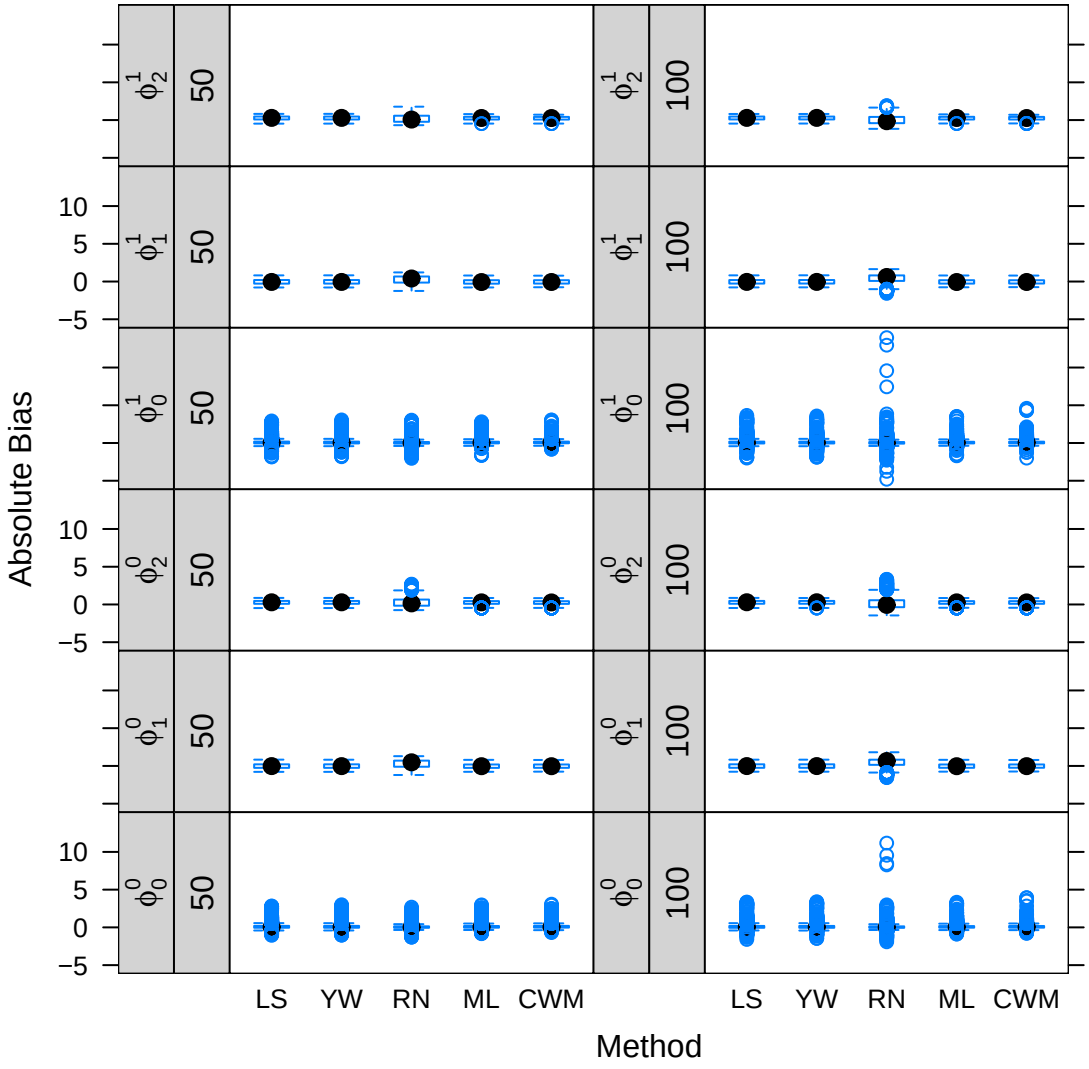

Figure 4.6: Autoregressive Parameter Absolute Bias—boxplots of the 10% trimmed mean absolute bias by sample size. Note: "0" superscript indicates a parameter for the autoregressive process in state 0, and a "1" superscript indicates a parameter for the autoregressive process in state 1.

n	parameter	LS	YW	RN	ML	CWM
50	ϕ_0^0	0.16	0.16	0.17	0.19	0.23
100	ϕ_0^0	0.11	0.11	0.53	0.13	0.15
50	ϕ_1^0	0.09	0.09	3.26	0.09	0.09
100	ϕ_1^0	0.09	0.09	3.78	0.08	0.08
50	ϕ_2^0	0.05	0.05	4.51	0.05	0.06
100	ϕ_2^0	0.05	0.05	5.36	0.05	0.05
50	ϕ_0^1	0.16	0.16	0.21	0.19	0.23
100	ϕ_0^1	0.11	0.11	0.59	0.13	0.16
50	ϕ_1^1	0.09	0.09	3.27	0.09	0.09
100	ϕ_1^1	0.09	0.09	3.88	0.08	0.08
50	ϕ_2^1	0.13	0.13	4.39	0.13	0.13
100	ϕ_2^1	0.13	0.13	4.92	0.13	0.14

Table 4.4: Autoregressive Coefficients' Average Mean-Square Error

the variance, and the least squares method is generally worse than the other methods. The least squares estimator is more biased than the maximum likelihood estimator, but the majority of the results for both methods are similar. Also, the average mean-square errors for the two methods are not very different, (Table 4.5).

4.2.3 One-Step-Ahead Prediction

One of the major goals of time series modeling is prediction. Figure 4.8 indicates that all of the methods are relatively unbiased. Again, the least squares method predicts about as well as any of the other methods, and the CWM was no better or worse than the MCWM. It is interesting to note that all of the methods predict better with a smaller sample size. It is unclear exactly why this might be, but it could be caused by failure to correctly classify the $t+1$ observation, as suggested by Figure 4.9.

4.2.4 Misclassification

Because knowledge of the state of the process is often valuable, another important criterion of method performance is misclassification rate. The non-iterative

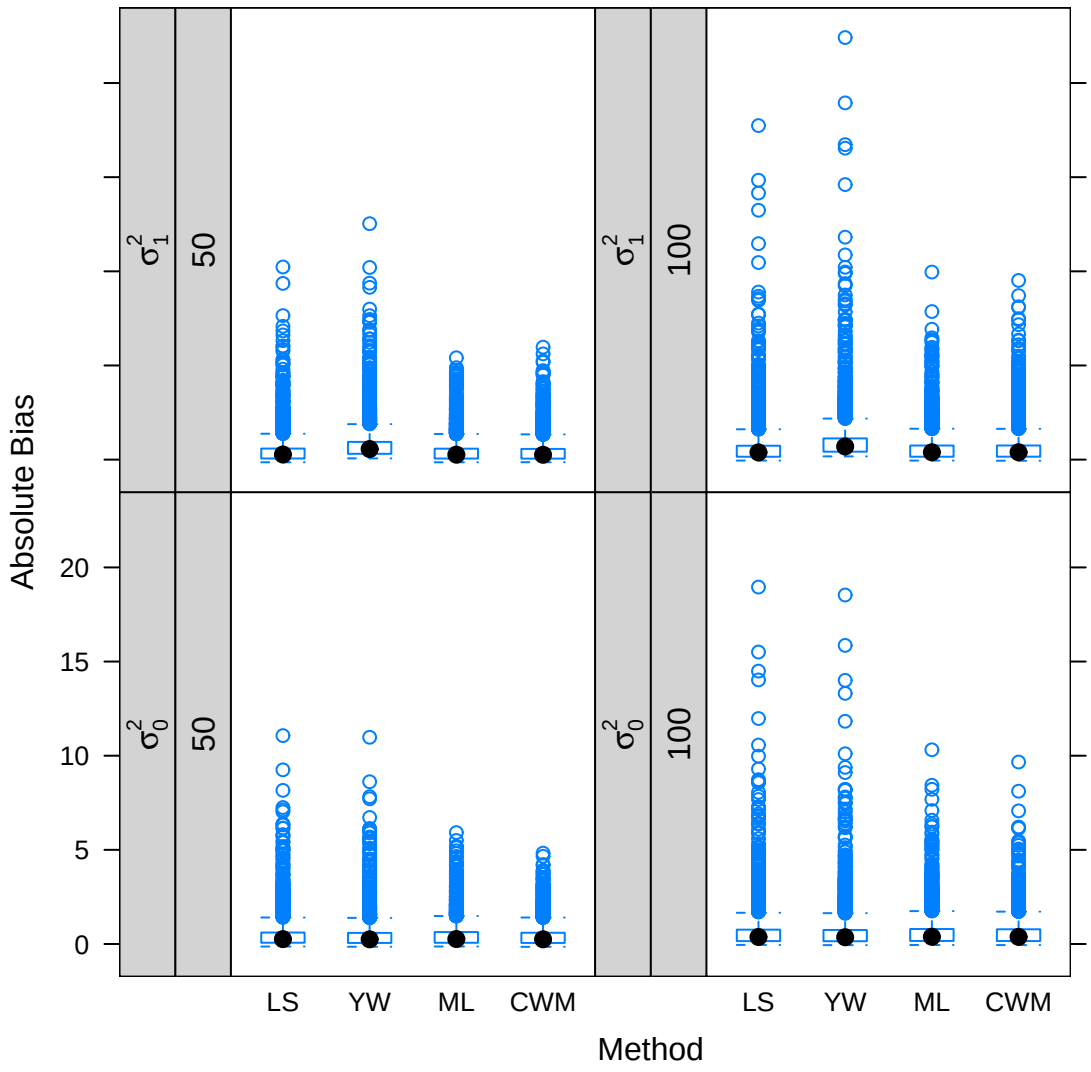

Figure 4.7: Variance Parameter Absolute Biasboxplots of the 10% trimmed mean absolute bias by sample size. Note: "0" superscript indicates a parameter for the autoregressive process in state 0, and a "1" superscript indicates a parameter for the autoregressive process in state 1.

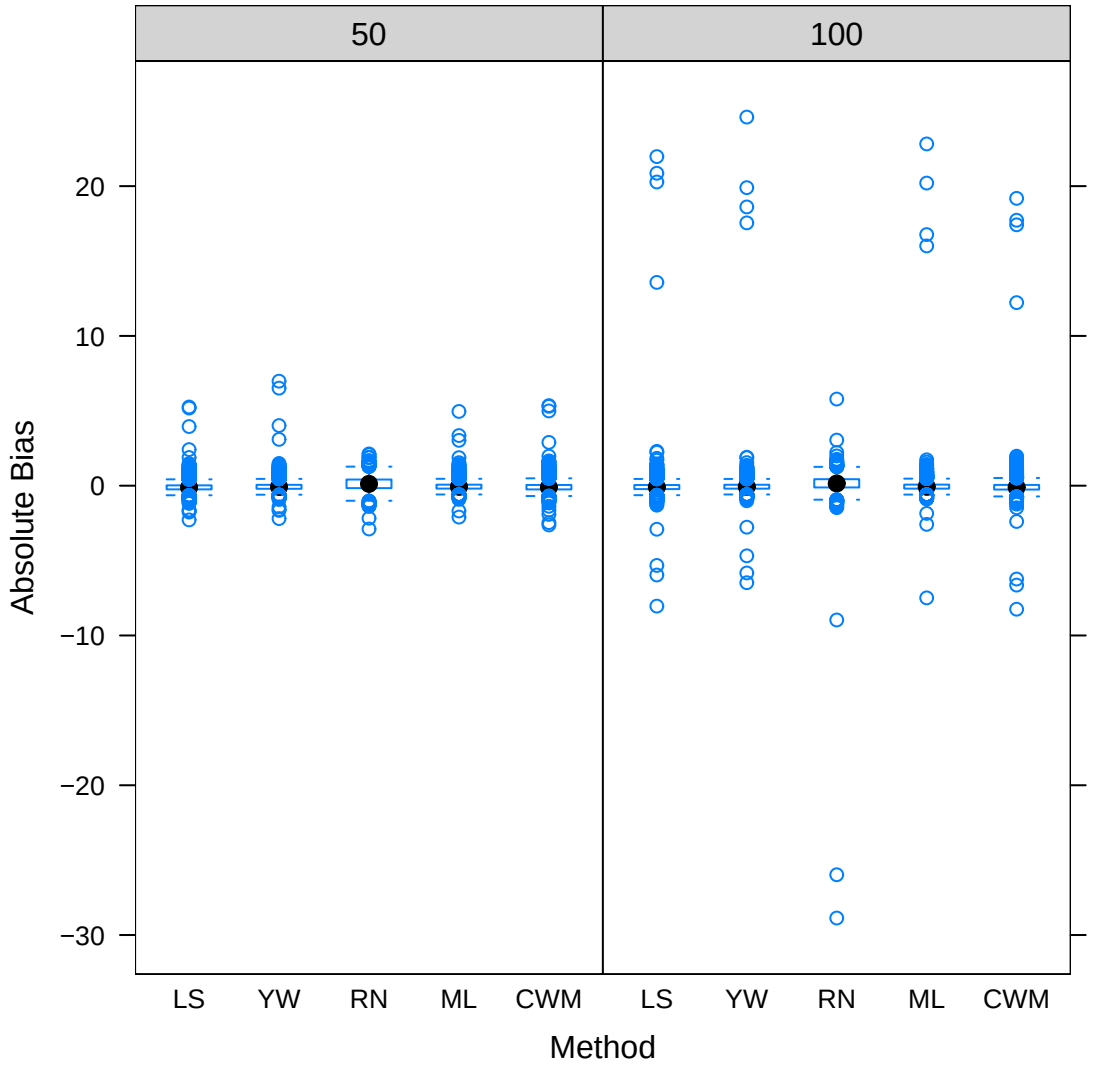

Figure 4.8: One-Step Ahead Prediction Absolute Bias—boxplots of the 10% trimmed mean absolute bias by sample size.

n	parameter	LS	YW	RN	ML	CWM
50	σ_0^2	0.27	0.26	46.18	0.30	0.46
100	σ_0^2	0.25	0.23	1742.73	0.27	0.46
50	σ_1^2	0.27	0.35	49.07	0.30	0.49
100	σ_1^2	0.23	0.32	1283.40	0.26	0.48

Table 4.5: Variance Parameters' Average Mean-Square Error

MCWM methods determine the state of an observation by the k-means method. The maximum likelihood methods, CWM and MCWM, determine the state of an observation by estimating the probability of an observation being in a state (see Section 3.2.2).

For the MCWM, the probability of an observation belonging to a state is a function of the previous observations from the nonlinear series and the exogenous variable. For the CWM, the probability of an observation belonging to a state for the CWM is a function of that observation only. As mentioned previously, this prevents the CWM from predicting future state membership. For the purpose of calculating the misclassification rate, the state of an observation is the state with the highest probability.

Figure 4.9 shows that all of the methods had about the same misclassification rate overall. Of course, the three non-iterative methods' success is merely the success of the k-means method. Comparing the mean misclassification rates in Table 4.6, the CWM is slightly worse than the other methods. The MCWM and the CWM maximum likelihood methods had lower misclassification rates in the upper tail than the three non-iterative methods, but the improvement does not appear to be substantial. The MCWM maximum likelihood method had a slightly lower misclassification in the upper tail than that of the CWM but, again, it was not substantially lower. Overall, the MCWM method classified the greatest number of observations correctly, but the non-iterative methods were almost equally as accurate on average. As mentioned pre-

viously, it is interesting that all of the methods had slightly higher misclassification rates in the upper tail with the larger sample size.

n	LS	YW	RN	ML	CWM
50	0.028	0.028	0.028	0.026	0.034
100	0.028	0.028	0.028	0.023	0.029

Table 4.6: Average Misclassification Rate

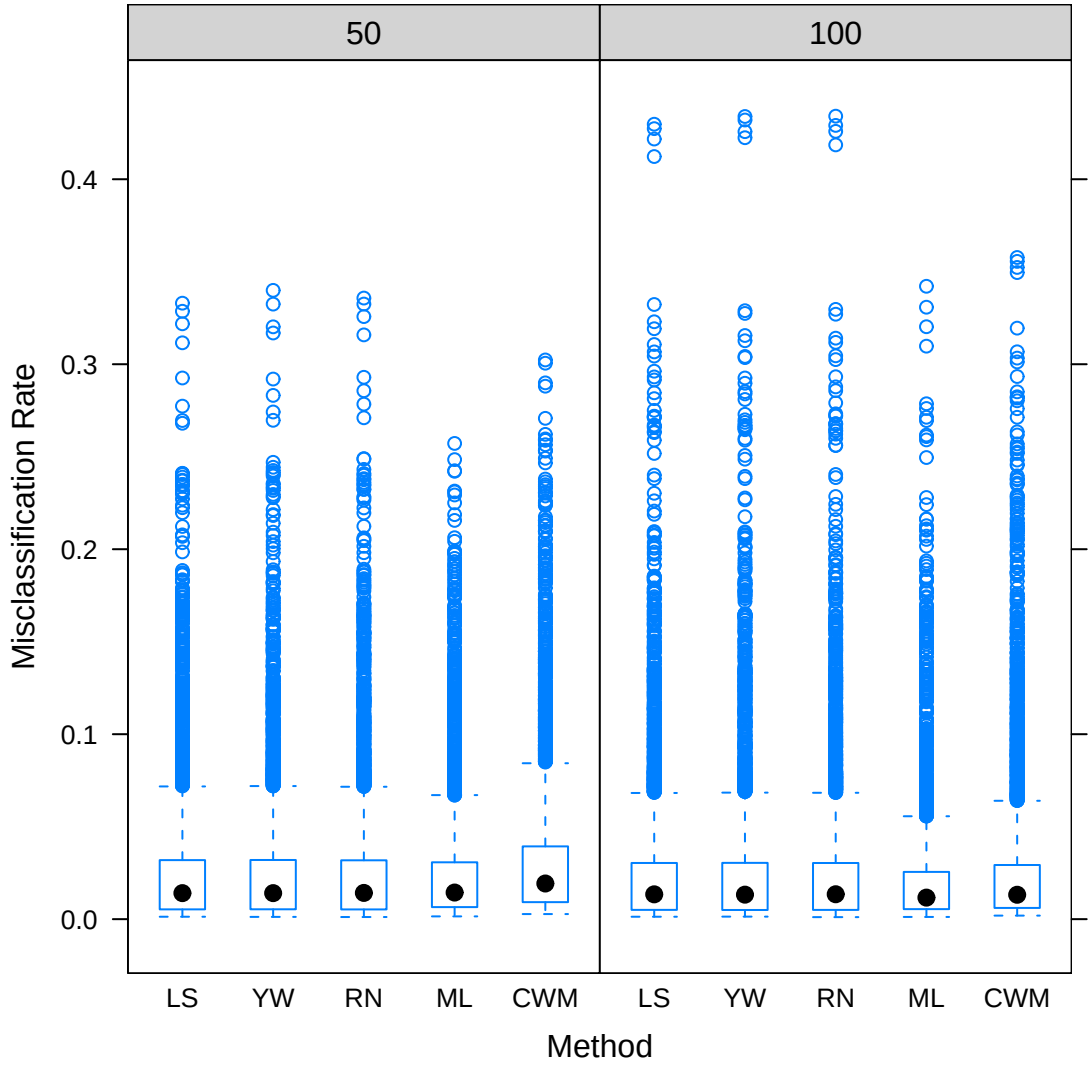

Figure 4.9: Misclassification Rate—boxplots of the misclassification rate by sample size.

5. CONCLUSIONS

In all of the previous figures the least squares method appears to perform nearly as well or better than the MCWM, CWM, root-n, and Yule-Walker methods. Along with the results of the simulation comparing time series estimators, showing that least squares estimation is a viable alternative to maximum likelihood estimators for first- and second-order models, this result indicates that attacking the problem of estimating MCWM's piecemeal with a least squares approach is perfectly acceptable. In fact, with the guarantee of only one iteration, it is much more desirable in real-time applications.

However, none of the methods of estimating the MCWM appear to perform better than the CWM. The MCWM is comparable to the CWM in modeling a non-linear process, but also allows prediction of future observations' states. Because of this additional benefit, the MCWM is recommended as a viable alternative to the CWM, especially when the state of the observations or the method in which the process changes state is of interest.

Areas of further study could include how the MCWM performs when adjusted for autocorrelation structure and probability of transition from state to state, to what extent the states can overlap before the parameters can no longer be adequately estimated, and how to estimate initial states more accurately.

BIBLIOGRAPHY

Alpargu, G. and Dutilleul, P. (2001), "Efficiency Analysis of Ten Estimation Procedures for Quantitative Linear Models with Autocorrelated Errors," *Journal of Statistical Computation and Simulation*, 69, 257–275.

Anandamohan, G. and Ram, R. (2005), "Cluster-weighted Modeling: Estimation of the Lyapunov Spectrum in Driven Systems," *Physical Review E*, 71, 016224–1–016224–6.

Azzalini, A. (1982), "Approximate Filtering of Parameter Driven Processes," *Journal of Time Series Analysis*, 3, 219–223.

Box, G. E. P., Jenkins, G. N., and Reinsel, G. C. (1994), *Time Series Analysis*, Englewood Cliffs, NJ: Prentice Hall, 3rd ed.

Brockwell, P. J. and Davis, R. A. (1983), *Time Series: Theory and Methods*, New York, NY: Wiley, 2nd ed.

Cai, Z., Fan, J., and Yao, Q. (2000), "Functional-Coefficient Regression Models for Nonlinear Time Series," *Journal of the American Statistical Association*, 95, 941–956.

Cappé, O., Moulines, E., and Rydén, T. (2005), *Inference in Hidden Markov Models*, New York, NY: Springer.

Chen, R. and Tsay, R. S. (1993), "Functional-Coefficient Autoregressive Models," *Journal of the American Statistical Association*, 88, 298–308.

Cox, D. R. (1981), "Statistical Analysis of Time Series: Some Recent Developments," *Scandinavian Journal of Statistics*, 8, 93–108.

Fan, J. and Yao, Q. (2003), *Nonlinear Time Series: Nonparametric and Parametric Methods*, New York, NY: Springer-Verlag.

Ferreira, F., Francisco, G., Machado, B., and Mrganandam, P. (2003), "Time Series Analysis for Minority Game Simulations of Financial Markets," *Physica A: Statistical Mechanics and Its Applications*, 321, 619–632.

Gershenfeld, N., Schoner, B., and Metois, E. (1999), "Cluster-weighted Modeling for Time-Series Analysis," *Nature*, 397, 329–332.

Granger, C. W. J. and Teräsvirta, T. (1993), *Modeling Nonlinear Economic Relationships*, Oxford, U.K.: Oxford University Press.

Haggan, V. and Ozaki, T. (1981), "Modeling Nonlinear Vibrations Using an Amplitude-Dependent Autoregressive Time Series Model," *Biometrika*, 68, 189–196.

Hamilton, J. D. (1989), "A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle," *Econometrica*, 57, 357–384.

Harvill, J. L. and Ray, B. K. (2005), "A Note on Multi-Step Forecasting with Functional Coefficient Autoregressive Models," *International Journal of Forecasting*, 21, 717–727.

He, Z., Pei, W., Yang, L., Hull, S. S., and Cheung, J. Y. (2002), "Modeling and Characterizing Deterministic Component of Variability by Cluster-Weighted Filtering," *Internaional Journal of Bifurcation and Chaos*, 12, 2967–2976.

Keenan, D. M. (1982), "A Time Series Analysis of Binary Data," *Journal of the American Statistical Association*, 77, 816–821.

Lehmann, E. L. (2001), *Elements of Large-Sample Theory*, New York, NY: Springer-Verlag.

Ozaki, T. (1982), "The Statistical Analysis of Perturbed Limit Cycle Processes Using Nonlinear Time Series Models," *Journal of Time Series Analysis*, 3, 29–41.

Pandit, S. M. and Wu, S.-M. (1983), *Time Series and Systems Analysis, with Applications*, New York, NY: Wiley.

R Development Core Team (2006), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Schoner, B., Cooper, C., Douglas, C., and Gershenfeld, N. (1999), "Data-Driven Modeling of Acoustical Instruments," *Journal of New Music Research*, 28, 81–89.

Shaman, P. and Stine, R. A. (1988), "The Bias of Autoregressive Coefficient Estimators," *Journal of the American Statistical Association*, 83.

Spitzer, J. J. (1979), "Small-Sample Properties of Nonlinear Least Squares and Maximum Likelihood Estimators in the Context of Autocorrelated Errors," *Journal of the American Statistical Association*, 74, 41–47.

Tong, H. (1990), *Nonlinear Time Series: A Dynamical System Approach*, New York, NY: Oxford University Press.

Wong, A. C. S. and Chan, W.-S. (2005), "Mixture Gaussian Time Series Modeling of Long-Term Market Returns," *North American Actuarial Journal*, 9, 83–94.

Zeger, S. L. (1988), "A Regression Model for Time Series Counts," *Biometrika*, 75, 621–629.

A. COMPUTER CODE

This appendix contains the R code used in the simulations, including code used to estimate the parameters of the MCWM and CWM using the various methods.

A.1 Simulation Code

```
# Function for generating simulated data sets
source('sim_nltime.R')
# Function for fitting MCWM model with EM algorithm
source('nltime_em.R')
# Function for fitting CWM model with EM algoritm
source('nltime_clust.R')
# Function for fitting MCWM model with least squares and weighted
# least squares
source('nltime_ls.R')
# Function for adjusting least squares estimates with Yule-Walker
# equations
source('nltime_yw.R')
# Function for adjusting least squares estimates with Root-n
# adjustment
source('nltime_rootn.R')
# Function for adjusting time series for shock
source('nltime_adjust.R')
# Function for obtaining predictions from non-linear time series model
source('nltime_predict.R')
# Wrapper function for other model fitting functions
source('nltime.R')
# Miscellaneous functions primarily used for this simulation
# specifically
source('misc_func.R')

# Create matrices and vectors of values to be used in the
# simulation for all parameters
beta0<-matrix(c(1.15,1.7,1.15,1.7,1.73,2.55),2)
beta1<-matrix(c(.58,.85,.58,.85,.87,1.28),2)
phi0 <- -3
phi1 <- 3

cand.phi <- c(-0.9,-0.5,-0.2,0.2,0.5,0.9)
cand.phi <- as.matrix(cand.phi)

cand.phi2<-as.matrix(expand.grid(phi1=cand.phi,phi2=cand.phi))
cand.phi2<-
```

```
cand.phi2[apply(matrix(apply(matrix(apply(cbind(1,-cand.phi2),
1,polyroot),2),2,Mod),2),2,min)>1,]

cand.phi<-as.matrix(rbind(cbind(cand.phi,NA),cand.phi2))

n <- c(50,100)

method <- c('ls','rootn','yw','em','clust')
method.f <- factor(method,levels=c('ls','rootn','yw','em','clust'))
method.f<-as.numeric(method.f)

nsim<-1000

# Combine all combinations of parameters into a data frame of all
# combinations of indexes
sim.mat<-expand.grid(beta0=1:nrow(beta0),beta1=1:nrow(beta1),
phi0=1:nrow(cand.phi),phi1=1:nrow(cand.phi),n=1:length(n))

# Create connections for all of the output vectors

out.connection<-file('nltimedata_temp.dat','w')

# Main Simulation #
for (i in 1:nrow(sim.mat))
{
# Determine order of the series in both states
  ar0.order<-sum(!is.na(cand.phi[sim.mat[i,'phi0'],]))
  ar1.order<-sum(!is.na(cand.phi[sim.mat[i,'phi1'],]))
# Create a temporary variable telling the data simulator to create an
# extra observation for use in determining prediction bias
  temp.n<-n[sim.mat[i,'n']]+1

# Repeat calculations for each settings nsim times
  for (j in 1:nsim)
  {
# Create list of 5 data objects each simulated under the same
# conditions, one for each estimation method
  out<-replicate(length(method),sim.nltime(beta0[sim.mat[i,'beta0'],],
   beta1[sim.mat[i,'beta1'],],c(phi0,cand.phi[sim.mat[i,'phi0'],]),
   c(phi1,cand.phi[sim.mat[i,'phi1'],]),1,1,n[sim.mat[i,'n']]+1),
   simplify=FALSE)

# Use kmeans to determine (initial) groups, assuming first or first
# and second observations are known
  grp<-lapply(out,grp.create)
```

```
# Adjust data for transition shock
  new.data<-mapply(data.adjust,out,grp,SIMPLIFY=FALSE)
# Fit each method to one set of data
  out.nl<-mapply(nl.fit,method,new.data,out,grp,SIMPLIFY=FALSE)
# Perform one-step-ahead prediction for each data set
  out.pred<-mapply(nl.predict,new.data,out,out.nl,method!="clust",
   SIMPLIFY=FALSE)
# Extract all parameter estimates in preparation for output
  tmp<-unlist(out.nl)
  tmp<-tmp[grep('estimates',names(tmp))]
# Create vectors of bias for each parameter and prediction
# and misclassification rate
  beta00.bias<-c(tmp[grep('beta00',names(tmp))],NA)-
   beta0[sim.mat[i,'beta0'],][1]
  beta01.bias<-c(tmp[grep('beta01',names(tmp))],NA)-
   beta0[sim.mat[i,'beta0'],][2]
  beta02.bias<-c(tmp[grep('beta02',names(tmp))],NA)-
   beta0[sim.mat[i,'beta0'],][3]
  beta10.bias<-c(tmp[grep('beta10',names(tmp))],NA)-
   beta1[sim.mat[i,'beta1'],][1]
  beta11.bias<-c(tmp[grep('beta11',names(tmp))],NA)-
   beta1[sim.mat[i,'beta1'],][2]
  beta12.bias<-c(tmp[grep('beta12',names(tmp))],NA)-
   beta1[sim.mat[i,'beta1'],][3]
  phi00.bias<-tmp[grep('phi00',names(tmp))]-phi0
  phi01.bias<-tmp[grep('phi01',names(tmp))]-
   cand.phi[sim.mat[i,'phi0'],][1]
  phi02.bias<-ifelse(rep(ar0.order,5)==2,
   tmp[grep('phi02',names(tmp))],rep(NA,5))-
   cand.phi[sim.mat[i,'phi0'],][2]
  phi10.bias<-tmp[grep('phi10',names(tmp))]-phi1
  phi11.bias<-tmp[grep('phi11',names(tmp))]-
   cand.phi[sim.mat[i,'phi1'],][1]
  phi12.bias<-ifelse(rep(ar1.order,5)==2,
   tmp[grep('phi02',names(tmp))],rep(NA,5))-
   cand.phi[sim.mat[i,'phi1'],][2]
  sigma0.bias<-tmp[grep('sigma0',names(tmp))]-1
  sigma1.bias<-tmp[grep('sigma1',names(tmp))]-1

  predbias<-mapply(nl.predbias,out.pred,out)
  misclass<-mapply(nl.misclass,out.nl,out)

# Create vector indexing what set of settings the data were simulated
# under
  setting<-rep(i,5)
```

```
  cat('setting = ',i,'iteration = ',j,'\n')

# Write each vector to a binary file

output<-cbind(setting,method.f,beta00.bias,beta01.bias,beta02.bias,
beta10.bias,beta11.bias,beta12.bias,phi00.bias,phi01.bias,phi02.bias,
phi10.bias,phi11.bias,phi12.bias,sigma0.bias,sigma1.bias,predbias,
misclass)

write(t(output),out.connection,18,TRUE)

  }
}

# Close the connection for each file

close(out.connection)
```

A.2 Data Simulation Code

```
sim.nltime<-function(beta0,beta1,phi0,phi1,sigma0,sigma1,n)
{

phi0<-phi0[!is.na(phi0)]
phi1<-phi1[!is.na(phi1)]

p0<-length(phi0)-1
p1<-length(phi1)-1

x<-rep(NA,n+max(p0,p1))
k <- rep(NA,n+max(p0,p1))
y <- rnorm(n+max(p0,p1))

p<- .5
k[1] <- rbinom(1,1,p)

for (i in 1:(n-1+max(p0,p1))){
if (k[i]==0){
mu<-ifelse(i<=p0,0,embed(x[(i-p0):(i-1)],p0)%*%phi0[-1])
x[i]<-rnorm(1,mu,sqrt(sigma0))
p<-exp(beta0[1]+beta0[2]*x[i]+beta0[3]*y[i])/
(1+exp(beta0[1]+beta0[2]*x[i]+beta0[3]*y[i]))
k[i+1]<-rbinom(1,1,p)
} else {
mu<-ifelse(i<=p1,0,embed(x[(i-p1):(i-1)],p1)%*%phi1[-1])
```

```
x[i]<-rnorm(1,mu,sqrt(sigma1))
p<-exp(beta1[1]+beta1[2]*x[i]+beta0[3]*y[i])/
(1+exp(beta1[1]+beta1[2]*x[i]+beta0[3]*y[i]))
k[i+1]<-rbinom(1,1,p)
}
}

x[n+max(p0,p1)]<-ifelse(k[n+max(p0,p1)]==0,
rnorm(1,embed(x[(i-p0):(i-1)],p0)%*%phi0[-1],sqrt(sigma0)),
rnorm(1,embed(x[(i-p1):(i-1)],p1)%*%phi1[-1],sqrt(sigma1)))

x<-x[-1:-max(p0,p1)]
y<-y[-1:-max(p0,p1)]
k<-k[-1:-max(p0,p1)]
x[k==0]<-x[k==0]+phi0[1]
x[k==1]<-x[k==1]+phi1[1]

return(list(x=x,y=y,k=k))
}
```

A.3 MCWM: Maximum Likelihood Method

```
nltime.em<-function(x,x2,k,param=NULL,ar0.order,ar1.order,
reltol=sqrt(.Machine$double.eps))
{

source('lik_em.R')

if(is.null(param)){
param<-nltime.ls(x,x2,k,ar0.order,ar1.order)$estimates
}

max.order<-max(ar0.order,ar1.order)
n<-length(x)
x2<-x2[-(n-max.order+1):-n]
X<-embed(x,max.order+1)
y<-X[,1]
X0<-cbind(1,X[,2:(ar0.order+1)])
X1<-cbind(1,X[,2:(ar1.order+1)])
X.b<-cbind(1,X[,2],x2)

beta0<-param[1:3]
beta1<-param[4:6]
phi0<-param[1:(ar0.order+1)+6]
phi1<-param[1:(ar1.order+1)+(ar0.order+7)]
```

```
sigma0.hat<-param[length(param)-1]
sigma1.hat<-param[length(param)]

old.lik<- -100000

k.hat<-rep(NA,nrow(X)+max.order)

k.hat[1:max.order]<-k[1:max.order]

for (i in 1:100){

for (j in 1:nrow(X))
{
cp0 <- lik.func(X0[j,]%*%phi0,X1[j,]%*%phi1,X.b[j,]%*%beta0,
X.b[j,]%*%beta1,sigma0.hat,sigma1.hat,y[j],0,k.hat[j])
cp1 <- lik.func(X0[j,]%*%phi0,X1[j,]%*%phi1,X.b[j,]%*%beta0,
X.b[j,]%*%beta1,sigma0.hat,sigma1.hat,y[j],1,k.hat[j])
deny <-cp0+cp1
new.cp1 <- cp1/deny
k.hat[j+max.order] <- new.cp1
}

W1<-diag(k.hat[-1:-max.order])
W0<-diag(1-k.hat[-1:-max.order])
phi0<-solve(t(X0)%*%W0%*%X0)%*%t(X0)%*%W0%*%y
phi1<-solve(t(X1)%*%W1%*%X1)%*%t(X1)%*%W1%*%y

pred0<-X0%*%phi0
pred1<-X1%*%phi1

res0<-y-pred0
res1<-y-pred1

sigma0.hat<-sum(diag(W0)*res0^2)/sum(diag(W0))
sigma1.hat<-sum(diag(W1)*res1^2)/sum(diag(W1))

W1.b<-diag(k.hat[(1:max.order+nrow(X))*-1])
W0.b<-diag(1-k.hat[(1:max.order+nrow(X))*-1])
theta0<-exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))
theta0[sapply(theta0,identical,1)]<-
theta0[sapply(theta0,identical,1)]-
sqrt(.Machine$double.eps)
theta1<-exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))
theta1[sapply(theta1,identical,1)]<-
```

```
theta1[sapply(theta1,identical,1)]-
sqrt(.Machine$double.eps)
logit.theta0<-X.b%*%beta0
logit.theta1<-X.b%*%beta1
Y0<-matrix(logit.theta0+(diag(W1)-theta0)/theta0/(1-theta0))
Y1<-matrix((logit.theta1+(diag(W1)-theta1)/theta1/(1-theta1)))
W0.b<-diag(c(theta0*(1-theta0)))%*%W0.b
W1.b<-diag(c(theta1*(1-theta1)))%*%W1.b
beta0<-solve(t(X.b)%*%W0.b%*%X.b)%*%t(X.b)%*%W0.b%*%Y0
beta1<-solve(t(X.b)%*%W1.b%*%X.b)%*%t(X.b)%*%W1.b%*%Y1

param<-c(beta0,beta1,phi0,phi1,sigma0.hat,sigma1.hat)
names(param)<-c(paste(rep(c('beta0','beta1'),each=3),0:2,sep='',
paste('phi0',0:ar0.order,sep=''),
paste('phi1',0:ar1.order,sep=''),'sigma0','sigma1')

lik<-sum(log(lik.func(X0%*%phi0,X1%*%phi1,X.b%*%beta0,X.b%*%beta1,
sigma0.hat,sigma1.hat,y,diag(W1),diag(W1.b))))

if (lik - old.lik < reltol *(abs(lik) + reltol))
{
break
}

old.lik<-lik
}
return(list(estimates=param,group=round(k.hat),likelihood=lik))
}
```

A.3.1 MCWM: ML—lik.func

```
lik.func<-function(mu0,mu1,log.odd0,log.odd1,sigma0,sigma1,y,kt,ktm1)
{
pi0<-exp(log.odd0)/(1+exp(log.odd0))
pi1<-exp(log.odd1)/(1+exp(log.odd1))

((sigma0)^-0.5*exp(-(y-mu0)^2/2/sigma0))^(1-kt)*
((sigma1)^-0.5*exp(-(y-mu1)^2/2/sigma1))^kt*
(pi0^kt*(1-pi0)^(1-kt))^(1-ktm1)*(pi1^kt*(1-pi1)^(1-kt))^ktm1

}
```

A.4 CWM: Maximum Likelihood Method

```
nltime.clust<-function(x,x2=NULL,k,param=NULL,ar0.order=1,ar1.order=1,
```

```
  reltol=sqrt(.Machine$double.eps))
{

source('log_lik_clust.R'
x2<-NULL
n<-length(x)
p.k <- mean(k)
X<-embed(x,max(ar0.order,ar1.order)+1)
y<-X[,1]
X0<-cbind(1,X[,2:(ar0.order+1)])
X1<-cbind(1,X[,2:(ar1.order+1)])

if(is.null(param)){
param<-nltime.ls(x,NULL,k,ar0.order,ar1.order)$estimates
param<-param[-1:-4]
}

phi0<-param[1:(ar0.order+1)]
phi1<-param[(ar0.order+2):(ar0.order+ar1.order+2)]
pred0<-X0%*%phi0
pred1<-X1%*%phi1

sigma0<-param[length(param)-1]
sigma1<-param[length(param)]

old.lik<- -100000

for (i in 1:100){

cp0.k <- dnorm(y,pred0,sqrt(sigma0))*(1-p.k)
cp1.k <- dnorm(y,pred1,sqrt(sigma1))*p.k
deny <- cp0.k+cp1.k
cp0.k <- ifelse(is.infinite(cp0.k) & is.infinite(deny),1,cp0.k/deny)

cp1.k <- 1-cp0.k

p.k <- mean(cp1.k)

W0<-diag(cp0.k)
W1<-diag(cp1.k)
phi0<-solve(t(X0)%*%W0%*%X0)%*%t(X0)%*%W0%*%y
phi1<-solve(t(X1)%*%W1%*%X1)%*%t(X1)%*%W1%*%y

pred0<-X0%*%phi0
pred1<-X1%*%phi1
```

```
res0<-y-pred0
res1<-y-pred1

sigma0<-sum(cp0.k*res0^2)/sum(cp0.k)
sigma1<-sum(cp1.k*res1^2)/sum(cp1.k)

param<-c(phi0,phi1,sigma0,sigma1)
names(param)<-c(paste('phi0',0:ar0.order,sep=''),
paste('phi1',0:ar1.order,sep=''),'sigma0','sigma1')

lik<-clust.log.lik(phi0,phi1,sigma0,sigma1,X0,X1,y,p.k)

if (lik - old.lik < reltol*(abs(lik) + reltol)) break

old.lik<-lik
}
return(list(estimates=param,
group=round(c(k[1:max(ar0.order,ar1.order)],
cp1.k)),likelihood=lik,p.k=p.k))
}
```

A.4.1 CWM: ML—clust.log.lik

```
clust.log.lik <- function(phi0.hat,phi1.hat,sigma0.hat,sigma1.hat,
X0,X1,y,p){

mu0<-X0%*%phi0.hat
mu1<-X1%*%phi1.hat

sum(log((1-p)*dnorm(y,mu0,sqrt(sigma0.hat))+
p*dnorm(y,mu1,sqrt(sigma1.hat))))
}
```

A.5 MCWM: Least Squares Method

```
nltime.ls<-function(x,x2,k,ar0.order=1,ar1.order=1)
{
# All parameters are estimate from the adjusted X values

n<-length(x)
x2<-x2[-(n-max(ar0.order,ar1.order)+1):-n]
X<-embed(x,max(ar0.order,ar1.order)+1)
y<-X[,1]
X0<-cbind(1,X[,2:(ar0.order+1)])
```

```
X1<-cbind(1,X[,2:(ar1.order+1)])
X.b<-cbind(1,X[,2],x2)

param <- rep(NA,10)

# Estimate beta parameters for probabability of jumping

W1.b <- diag(k[-(n-max(ar0.order,ar1.order)+1):-n])
W0.b <- diag(1-k[-(n-max(ar0.order,ar1.order)+1):-n])
beta0<-solve(t(X.b)%*%W0.b%*%X.b)%*%t(X.b)%*%W0.b%*%
k[-1:-max(ar0.order,ar1.order)]
beta1<-solve(t(X.b)%*%W1.b%*%X.b)%*%t(X.b)%*%W1.b%*%
k[-1:-max(ar0.order,ar1.order)]

theta0<-exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))
theta1<-exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))
logit.theta0<-X.b%*%beta0
logit.theta1<-X.b%*%beta1
Y0<-matrix(logit.theta0+(k[-1:-max(ar0.order,ar1.order)]-theta0)/
theta0/(1-theta0))
Y1<-matrix((logit.theta1+(k[-1:-max(ar0.order,ar1.order)]-theta1)/
theta1/(1-theta1)))
W0.b<-diag(c(theta0*(1-theta0)))%*%
diag(1-k[-(n-max(ar0.order,ar1.order)+1):-n])
W1.b<-diag(c(theta1*(1-theta1)))%*%
diag(k[-(n-max(ar0.order,ar1.order)+1):-n])
beta0<-solve(t(X.b)%*%W0.b%*%X.b)%*%t(X.b)%*%W0.b%*%Y0
beta1<-solve(t(X.b)%*%W1.b%*%X.b)%*%t(X.b)%*%W1.b%*%Y1

# Estimate phi parameters for local time series
W1<-diag(k[-1:-max(ar0.order,ar1.order)])
W0<-diag(1-k[-1:-max(ar0.order,ar1.order)])
phi0<-solve(t(X0)%*%W0%*%X0)%*%t(X0)%*%W0%*%y
phi1<-solve(t(X1)%*%W1%*%X1)%*%t(X1)%*%W1%*%y

pred0<-X0%*%phi0
pred1<-X1%*%phi1

res0<-y-pred0
res1<-y-pred1

# Estimate variance for local time series

sigma0.hat<-sum(diag(W0)*res0^2)/sum(diag(W0))
sigma1.hat<-sum(diag(W1)*res1^2)/sum(diag(W1))
```

```
param<-c(beta0,beta1,phi0,phi1,sigma0.hat,sigma1.hat)
name.length<-ifelse(identical(x2,NULL),2,3)
names(param)<-c(paste(rep(c('beta0','beta1'),each=name.length),
0:(name.length-1),sep=''),paste('phi0',0:ar0.order,sep=''),
paste('phi1',0:ar1.order,sep=''),'sigma0','sigma1')

return(list(estimates=param,group=k))
}
```

A.6 MCWM: Yule-Walker Method

```
nltime.yw<-function(x,x2,k,ar0.order=1,ar1.order=1)
{
# All parameters are estimate from the adjusted X values

n<-length(x)
x2<-x2[-(n-max(ar0.order,ar1.order)+1):-n]
X<-embed(x,max(ar0.order,ar1.order)+1)
y<-X[,1]
X0<-cbind(1,X[,2:(ar0.order+1)])
X1<-cbind(1,X[,2:(ar1.order+1)])
X.b<-cbind(1,X[,2],x2)

param <- rep(NA,10)

# Estimate beta parameters for probabability of jumping

W1.b <- diag(k[-(n-max(ar0.order,ar1.order)+1):-n])
W0.b <- diag(1-k[-(n-max(ar0.order,ar1.order)+1):-n])
beta0<-solve(t(X.b)%*%W0.b%*%X.b)%*%t(X.b)%*%W0.b%*%
k[-1:-max(ar0.order,ar1.order)]
beta1<-solve(t(X.b)%*%W1.b%*%X.b)%*%t(X.b)%*%W1.b%*%
k[-1:-max(ar0.order,ar1.order)]

theta0<-exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))
theta1<-exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))
logit.theta0<-X.b%*%beta0
logit.theta1<-X.b%*%beta1
Y0<-matrix(logit.theta0+(k[-1:-max(ar0.order,ar1.order)]-theta0)/
theta0/(1-theta0))
Y1<-matrix((logit.theta1+(k[-1:-max(ar0.order,ar1.order)]-theta1)/
theta1/(1-theta1)))
W0.b<-diag(c(theta0*(1-theta0)))%*%
diag(1-k[-(n-max(ar0.order,ar1.order)+1):-n])
```

```
W1.b<-diag(c(theta1*(1-theta1)))%*%
diag(k[-(n-max(ar0.order,ar1.order)+1):-n])
beta0<-solve(t(X.b)%*%W0.b%*%X.b)%*%t(X.b)%*%W0.b%*%Y0
beta1<-solve(t(X.b)%*%W1.b%*%X.b)%*%t(X.b)%*%W1.b%*%Y1

# Estimate phi parameters for local time series
acov.mat<-matrix(NA,2,max(ar0.order,ar1.order)+1)
mean.vec<-unique(ave(x,k)[order(k)])
for (i in 1:2){
for (h in 0:max(ar0.order,ar1.order)){
acov.mat[i,h+1]<-sum((x[(h+1):n][k[1:(n-h)]==(i-1)]-mean.vec[i])*
(x[1:(n-h)][k[1:(n-h)]==(i-1)]-mean.vec[i]))/
sum(k[1:(n-h)]==(i-1))
}
}
acor.mat<-sweep(acov.mat,1,acov.mat[,1],'/')

if (ar0.order==1) phi0 <- acor.mat[1,2]
else phi0 <- c(acor.mat[1,2]*(1-acor.mat[1,3]),
  acor.mat[1,3]-acor.mat[1,2]^2)/(1-acor.mat[1,2]^2)

if (ar1.order==1) phi1 <- acor.mat[2,2]
else phi1 <- c(acor.mat[2,2]*(1-acor.mat[2,3]),
acor.mat[2,3]-acor.mat[2,2]^2)/(1-acor.mat[2,2]^2)

sigma0<-acov.mat[1,1]*(1-matrix(acor.mat[1,1:ar0.order+1],nrow=1)%*%
matrix(phi0))
sigma1<-acov.mat[2,1]*(1-matrix(acor.mat[2,1:ar1.order+1],nrow=1)%*%
matrix(phi1))

phi0<-c(mean.vec[1],phi0)
phi1<-c(mean.vec[2],phi1)

param<-c(beta0,beta1,phi0,phi1,sigma0,sigma1)
names(param)<-c(paste(rep(c('beta0','beta1'),each=3),0:2,sep=''),
paste('phi0',0:ar0.order,sep=''),paste('phi1',0:ar1.order,sep=''),
'sigma0','sigma1')

return(list(estimates=param,group=k))
}
```

A.7 MCWM: Root-n Method

```
nltime.rootn<-function(x,x2,k,param=NULL,ar0.order,ar1.order,
  reltol=sqrt(.Machine$double.eps))
```

```
{
source('information3.R')
source('score4.R')
source('lik_em.R')

if(is.null(param)){
param<-nltime.ls(x,x2,k,ar0.order,ar1.order)$estimates
}

max.order<-max(ar0.order,ar1.order)
n<-length(x)
x2<-x2[-(n-max.order+1):-n]
X<-embed(x,max.order+1)
y<-X[,1]
X0<-cbind(1,X[,2:(ar0.order+1)])
X1<-cbind(1,X[,2:(ar1.order+1)])
X.b<-cbind(1,X[,2],x2)

beta0<-param[1:3]
beta1<-param[4:6]
phi0<-param[1:(ar0.order+1)+6]
phi1<-param[1:(ar1.order+1)+(ar0.order+7)]
sigma0<-param[length(param)-1]
sigma1<-param[length(param)]

lik<-sum(log(lik.func(X0%*%phi0,X1%*%phi1,X.b%*%beta0,X.b%*%beta1,
sigma0,sigma1,y,k[-1:-max.order],k[-(n-max.order+1):-n])))
s<-score(beta0,beta1,phi0,phi1,sigma0,sigma1,X.b,X0,X1,y,
k[-1:-max.order],k[-(n-max.order+1):-n])
I<-information(beta0,beta1,phi0,phi1,sigma0,sigma1,X.b,X0,X1,y,
k[-1:-max.order],k[-(n-max.order+1):-n])
new.param<-param+s%*%solve(I)

colnames(new.param)<-
c(paste(rep(c('beta0','beta1'),each=3),0:2,sep=''),
paste('phi0',0:ar0.order,sep=''),paste('phi1',0:ar1.order,sep=''),
'sigma0','sigma1')

return(list(estimates=new.param[1,],group=k))
}
```

A.7.1 MCWM: Root-n—information

```
information <- function(beta0,beta1,phi0,phi1,sigma0,sigma1,X.b,X0,
X1,y,kt,ktm1){
```

```
  #I11
  i11<-sum((1-ktm1)*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))^2)

  #I22
  i22<-sum((1-ktm1)*X.b[,2]^2*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))^2)

  #I33
  i33<-sum((1-ktm1)*X.b[,3]^2*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))^2)

  #I44
  i44<-sum(ktm1*exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))^2)

  #I55
  i55<-sum(ktm1*X.b[,2]^2*exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))^2)

  #I66
  i66<-sum(ktm1*X.b[,3]^2*exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))^2)

  #I77
  i77<-sum(1-kt)/sigma0

  #I88
  i88<-sum((1-kt)*X0[,2]^2)/sigma0

  #I99
if(ncol(X0)<3) i99<-NULL else i99<-sum((1-kt)*X0[,3]^2)/sigma0

  #I1010
  i1010<-sum(kt)/sigma1

  #I1111
  i1111<-sum(kt*X1[,2]^2)/sigma1

  #I1212
if(ncol(X1)<3) i1212<-NULL else i1212<-sum(kt*X1[,3]^2)/sigma1

  #I1313
  i1313<-sum(-(1-kt)/2/sigma0^2+(y-X0%*%phi0)^2/sigma0^3)

  #I1414
  i1414<-sum(-kt/2/sigma1^2+(y-X1%*%phi1)^2/sigma1^3)

  #I12
  i12<-sum((1-ktm1)*X.b[,2]*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))^2)
```

```
#I13
i13<-sum((1-ktm1)*X.b[,3]*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0))^2)

#I14-I112
i14<-i15<-i16<-i17<-i18<-0
if(ncol(X0)<3) i19<-NULL else i19<-0
i110<-i111<-0
if(ncol(X1)<3) i112<-NULL else i112<-0
i113<-i114<-0

#I23
i23<-sum((1-ktm1)*X.b[,2]*X.b[,3]*exp(X.b%*%beta0)/
 (1+exp(X.b%*%beta0))^2)

#I24-I214
i24<-i25<-i26<-i27<-i28<-0
if(ncol(X0)<3) i29<-NULL else i29<-0
i210<-i211<-0
if(ncol(X1)<3) i212<-NULL else i212<-0
i213<-i214<-0

#I34-I314
i34<-i35<-i36<-i37<-i38<-0
if(ncol(X0)<3) i39<-NULL else i39<-0
i310<-i311<-0
if(ncol(X1)<3) i312<-NULL else i312<-0
i313<-i314<-0

#I45
i45<-sum(ktm1*X.b[,2]*exp(X.b%*%beta1)/(1+exp(X.b%*%beta1))^2)

#I45-I414
i45<-i46<-i47<-i48<-0
if(ncol(X0)<3) i49<-NULL else i49<-0
i410<-i411<-0
if(ncol(X1)<3) i412<-NULL else i412<-0
i413<-i414<-0

#I56
i56<-sum(ktm1*X.b[,2]*X.b[,3]*exp(X.b%*%beta1)/
 (1+exp(X.b%*%beta1))^2)

#I57-I514
i57<-i58<-0
```

```
if(ncol(X0)<3) i59<-NULL else i59<-0
i510<-i511<-0
if(ncol(X1)<3) i512<-NULL else i512<-0
i513<-i514<-0

#I67-I614
i67<-i68<-0
if(ncol(X0)<3) i69<-NULL else i69<-0
i610<-i611<-0
if(ncol(X1)<3) i612<-NULL else i612<-0
i613<-i614<-0

#I78
i78<-sum((1-kt)*X0[,2])/sigma0

#I79
if(ncol(X0)<3) i79<-NULL else i79<-sum((1-kt)*X0[,3])/sigma0

#I710-I712
i710<-i711<-0
if(ncol(X1)<3) i712<-NULL else i712<-0

#I713
i713<-sum((1-kt)*(y-X0%*%phi0))/sigma0^2

#I714
i714<-0

#I89
if(ncol(X0)<3) i89<-NULL else i89<-sum((1-kt)*X0[,3]*X0[,2])/sigma0

#I810-I812
i810<-i811<-0
if(ncol(X1)<3) i812<-NULL else i812<-0

#I813
i813<-sum((1-kt)*X0[,2]*(y-X0%*%phi0))/sigma0^2

#I814
i814<-0

#I910
if(ncol(X0)<3) i910<-NULL else i910<-0

#I911
```

```
if(ncol(X0)<3) i911<-NULL else i911<-0

#I912
if(ncol(X0)<3 | ncol(X1)<3) i912<-NULL else i912<-0

#I913
if(ncol(X0)<3) i913<-NULL else i913<-sum((1-kt)*X0[,3]*
 (y-X0%*%phi0))/sigma0^2

#I914
if(ncol(X0)<3) i914<-NULL else i914<-0

#I1011
i1011<-sum(kt*X1[,2])/sigma1

#I1012
if(ncol(X1)<3) i1012<-NULL else i1012<-sum(kt*X1[,3])/sigma1

#I1013
i1013<-0

#I1014
i1014<-sum(kt*(y-X1%*%phi1))/sigma1^2

#I1112
if(ncol(X1)<3) i1112<-NULL else i1112<-sum(kt*X1[,2]*X1[,3])/sigma1

#I1113
i1113<-0

#I1114
i1114<-sum(kt*X1[,2]*(y-X1%*%phi1))/sigma1^2

#I1213
if(ncol(X1)<3) i1213<-NULL else i1213<-0

#I1214
if(ncol(X1)<3) i1214<-NULL else i1214<-sum(kt*X1[,3]*(y-X1%*%phi1))/
 sigma1^2

#I1314
i1314<-0
out<-diag(c(i11,i22,i33,i44,i55,i66,i77,i88,i99,i1010,i1111,i1212,
 i1313,i1414))
out[upper.tri(out)]<-c(i12,i13,i23,i14,i24,i34,i15,i25,i35,i45,i16,
```

```
   i26,i36,i46,i56,i17,i27,i37,i47,i57,i67,i18,i28,i38,i48,i58,i68,
   i78,i19,i29,i39,i49,i59,i69,i79,i89,i110,i210,i310,i410,i510,i610,
   i710,i810,i910,i111,i211,i311,i411,i511,i611,i711,i811,i911,i1011,
   i112,i212,i312,i412,i512,i612,i712,i812,i912,i1012,i1112,i113,i213,
   i313,i413,i513,i613,i713,i813,i913,i1013,i1113,i1213,i114,i214,
   i314,i414,i514,i614,i714,i814,i914,i1014,i1114,i1214,i1314)
  out[lower.tri(out)]<-t(out)[lower.tri(out)]

  return(out)
}
```

A.7.2 MCWM: Root-n—score

```
score <- function(beta0,beta1,phi0,phi1,sigma0,sigma1,X.b,X0,
X1,y,kt,ktm1){

# S1
s1<-sum((1-ktm1)*kt-(1-ktm1)*exp(X.b%*%beta0)/(1+exp(X.b%*%beta0)))

# S2
s2<-sum((1-ktm1)*kt*X.b[,2]-(1-ktm1)*exp(X.b%*%beta0)*X.b[,2]/
(1+exp(X.b%*%beta0)))

# S3
s3<-sum((1-ktm1)*kt*X.b[,3]-(1-ktm1)*exp(X.b%*%beta0)*X.b[,3]/
(1+exp(X.b%*%beta0)))

# S4
s4<-sum(ktm1*kt-ktm1*exp(X.b%*%beta1)/(1+exp(X.b%*%beta1)))

# S5
s5<-sum(ktm1*kt*X.b[,2]-ktm1*exp(X.b%*%beta1)*X.b[,2]/
(1+exp(X.b%*%beta1)))

# S6
s6<-sum(ktm1*kt*X.b[,3]-ktm1*exp(X.b%*%beta1)*X.b[,3]/
(1+exp(X.b%*%beta1)))

# S7
s7<-sum((1-kt)*(y-X0%*%phi0)/sigma0)

# S8
s8<-sum(X0[,2]*(1-kt)*(y-X0%*%phi0)/sigma0)

# S9
```

```
s9<-ifelse(ncol(X0)>2,sum(X0[,3]*(1-kt)*(y-X0%*%phi0)/sigma0),NA)

# S10
s10<-sum(kt*(y-X1%*%phi1)/sigma1)

# S11
s11<-sum(X1[,2]*kt*(y-X1%*%phi1)/sigma1)

# S12
s12<-ifelse(ncol(X1)>2,sum(X1[,3]*kt*(y-X1%*%phi1)/sigma1),NA)

# S13
s13<- sum((1-kt)*(-2*sigma0+(y-X0%*%phi0)^2)/(2*sigma0)^2)

# S14
s14<- sum((1-kt)*(-2*sigma1+(y-X1%*%phi1)^2)/(2*sigma1)^2)

s<-c(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14)
return(s[!is.na(s)])
}
```

A.7.3 MCWM: Root-n—lik.func

See Appendix A.3.1.

A.8 Shock Adjustment

```
nltime.adjust<-function(x,k,ar0.order,ar1.order)
{
n<-length(x)
m<-rep(NA,n)
m[1]<-1

for (i in 2:n)
{
m[i]<-ifelse(k[i-1]==k[i],m[i-1]+1,1)
}

mean0 <- mean(x[k==0])
mean1 <- mean(x[k==1])
x0<-x[-n]-mean0
x1<-x[-n]-mean1
y0<-x[-1]-mean0
y1<-x[-1]-mean1
w0<-diag(1-k[-1])
```

```
w1<-diag(k[-1])
delta0 <- solve(t(x0)%*%w0%*%x0)%*%t(x0)%*%w0%*%y0
delta1 <- solve(t(x1)%*%w1%*%x1)%*%t(x1)%*%w1%*%y1

shock <- rep(0,length(x))
shock[k==0] <- mean0 - mean1
shock[k==1] <- mean1 - mean0
shock[cumsum(k!=k[1])==0] <- 0
adj.x <- rep(NA,length(x))
adj.x[k==0] <- x[k==0] - delta0^m[k==0]*shock[k==0]
adj.x[k==1] <- x[k==1] - delta1^m[k==1]*shock[k==1]

shock<-unique(shock[order(k) & shock!=0])

return(list(adj.x=adj.x,delta=c(delta0,delta1),shock=shock))
}
```

A.9 Prediction

```
predict.nltime<-function(x,y,grp,param,ar0.order,ar1.order,delta=NULL,
shock=NULL,modified=T,prob1=NULL){

  n<-length(x)

  if(!is.numeric(param)) return(list(x=NA,grp=NA))

  if(modified){
    beta0<-param[1:3]
    beta1<-param[4:6]
    phi0<-param[1:(ar0.order+1)+6]
    phi1<-param[1:(ar1.order+1)+(ar0.order+7)]

    sigma0.hat<-param[length(param)-1]
    sigma1.hat<-param[length(param)]

    if(grp[n]==0) {
     prob1<-exp(beta0%*%c(1,x[n],y[n]))/
     (1+exp(beta0%*%c(1,x[n],y[n])))
    }else {
     prob1<-exp(beta1%*%c(1,x[n],y[n]))/
     (1+exp(beta1%*%c(1,x[n],y[n])))
    }

    new.grp<-prob1[1,1]
```

```
  }else{
    phi0<-param[1:(ar0.order+1)]
    phi1<-param[(ar0.order+2):(ar0.order+ar1.order+2)]

    sigma0.hat<-param[length(param)-1]
    sigma1.hat<-param[length(param)]
    new.grp<-prob1
 }

  new.x<-c(1,x[-1:-(n-length(phi0)+1)])%*%phi0*(1-new.grp)+
    c(1,x[-1:-(n-length(phi1)+1)])%*%phi1*new.grp

  new.grp<-round(new.grp)

  if(!is.null(delta)){
    m<-n-max(which(grp!=new.grp))+1
    new.x<-new.x[1,1]+delta[new.grp+1]^m*shock[new.grp+1]
  }

  return(list(x=new.x,grp=new.grp))
}
```

A.10 Estimation Function Wrapper Function

```
nltime<-function(my.method,...)
{
    switch(my.method,
      ls = nltime.ls(...),
      rootn = nltime.rootn(...),
      yw = nltime.yw(...),
      clust = nltime.clust(...),
      em = nltime.em(...)
    )
}
```

A.11 Miscellaneous Simulation Specific Functions

```
# Chooses the group centers based on the first point and the second
# if AR(2). If AR(1) the first point and another random point are
# used as group centers.
grp.create <- function(y)
{
  tmp<-ifelse(identical(y$k[2],y$k[1]),sample(y$x[c(-1,-temp.n)],1),
   y$x[2])
  tmp<-c(y$x[1],tmp)
```

```
  out<-kmeans(y$x[-temp.n],tmp)$cluster-1
# This line ensures that the estimation procedure does not perform
# poorly by merely mislabeling the states.
  if(mean(out==y$k[-temp.n])<0.5) out<-abs(out-1)
  return(out)
}

# Function for applying the shock adjustment to each element
# of the list.

data.adjust <- function(x,y)
{
tryCatch(nltime.adjust(x$x[-temp.n],y,ar0.order,ar1.order),
error = function(e) list(adj.x = rep(NA,length(x$x[-temp.n])),
delta=c(NA,NA),shock=c(NA,NA)))
}

# Function for fitting the various model fitting methods to each
# element of the list
# Checks for non-convergence and returns NA's for all values if error
nl.fit <- function(method,x,y,z){

  my.error.func<-function(e)
  {
    grp<-rep(NA,length(z))
    if(identical(method,'clust')){
      param<-rep(NA,ar0.order+ar1.order+4)
      names(param)<-c(paste('phi0',0:ar0.order,sep=''),
       paste('phi1',0:ar1.order,sep=''),
       'sigma0','sigma1')
      list(estimates=param,group=grp)
    } else {
      param<-rep(NA,ar0.order+ar1.order+10)
      names(param)<-
       c(paste(rep(c('beta0','beta1'),each=3),0:2,sep=''),
       paste('phi0',0:ar0.order,sep=''),
       paste('phi1',0:ar1.order,sep=''),'sigma0','sigma1')
      list(estimates=param,group=grp)
    }
  }

    tryCatch(nltime(method,x=x$adj.x,x2=y$y[-temp.n],k=z,
     ar0.order=ar0.order,ar1.order=ar1.order),error = my.error.func)
}
```

```
# Function for obtaining one-step predictions for all elements of the
# list
nl.predict <- function(x,y,z,modified=TRUE)
{
  predict.nltime(x$adj.x,y$y[-temp.n],z$group,z$estimates,ar0.order,
  ar1.order,x$delta,x$shock,modified=modified,prob1=mean(z$group))
}

# Function for calculating one-step-ahead prediction bias
# especially for elements of list
nl.predbias <- function(x,y) x$x-y$x[temp.n]

# Function for calculating misclassifiaction rates especially
# for elements of list
nl.misclass <- function(x,y)
{
mean(x$group[c(-1:-max(ar0.order,ar1.order))]!=
y$k[c(-1:-max(ar0.order,ar1.order),-temp.n)])
}
```