Journal of Statistical Software

October 2008, Volume 28, Issue 2. *http://www.jstatsoft.org/*

Improved Subset Autoregression: With R Package

A. I. McLeod
University of Western Ontario

Y. Zhang
Acadia University

Abstract

The **FitAR** R (R Development Core Team 2008) package that is available on the Comprehensive R Archive Network is described. This package provides a comprehensive approach to fitting autoregressive and subset autoregressive time series. For long time series with complicated autocorrelation behavior, such as the monthly sunspot numbers, subset autoregression may prove more feasible and/or parsimonious than using AR or ARMA models. The two principal functions in this package are `SelectModel` and `FitAR` for automatic model selection and model fitting respectively. In addition to the regular autoregressive model and the usual subset autoregressive models (Tong 1977), these functions implement a new family of models. This new family of subset autoregressive models is obtained by using the partial autocorrelations as parameters and then selecting a subset of these parameters. Further properties and results for these models are discussed in McLeod and Zhang (2006). The advantages of this approach are that not only is an efficient algorithm for exact maximum likelihood implemented but that efficient methods are derived for selecting high-order subset models that may occur in massive datasets containing long time series. A new improved extended BIC criterion, UBIC, developed by Chen and Chen (2008) is implemented for subset model selection. A complete suite of model building functions for each of the three types of autoregressive models described above are included in the package. The package includes functions for time series plots, diagnostic testing and plotting, bootstrapping, simulation, forecasting, Box-Cox analysis, spectral density estimation and other useful time series procedures. As well as methods for standard generic functions including `print`, `plot`, `predict` and others, some new generic functions and methods are supplied that make it easier to work with the output from `FitAR` for bootstrapping, simulation, spectral density estimation and Box-Cox analysis.

Keywords: Box-Cox analysis, diagnostic checks and residual autocorrelation, extended BIC and UBIC criterion for subset selection, high-order autoregression, massive datasets and long time series, monthly sunspot numbers, partial autocorrelations, spectral density estimation .

1. Introduction

The family of AR(p) models for a time series $z_t, t = 1, 2, \ldots$, may be written in operator notation, $\phi(B)(z_t - \mu) = a_t$, where $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$, μ is the mean of the series and $a_t \sim$ NID $(0, \sigma_a^2)$. It is assumed that the parameters are in the admissible region so that all roots of the equation $\phi(B) = 0$ lie outside the unit circle. The usual family of subset AR(p) models is obtained by taking a subset of the parameters $\phi_1, \ldots, \phi_p$, and may be written as $\phi(B)(z_t - \mu) = a_t$, where $\phi(B) = 1 - \phi_{i_1} B - \ldots - \phi_{i_m} B^{i_m}$. This family of subset AR models may be denoted by AR$p(i_1, \ldots, i_m)$, where $0 \le i_1 \le i_2 \ldots \le i_m \le p$[1].

To define the new family, consider the Durbin-Levinson recursion for the AR(p)

$$\phi_{j,k+1} = \phi_{j,k} - \phi_{k+1,k+1}\phi_{k+1-j,k}, \quad j = 1, \ldots, k, \tag{1}$$

where $k = 1, \ldots, p-1$ and $\zeta_{k+1} = \phi_{k+1,k+1}$ is the partial autocorrelation at lag $k+1$. This recursion can be used to define a one-to-one, onto, continuous and differentiable transformation (Barndorff-Nielsen and Schou 1973; Monahan 1984; Zhang and McLeod 2006b, Theorem 1),

$$(\phi_1, \ldots, \phi_p) \longleftrightarrow (\zeta_1, \ldots, \zeta_p), \tag{2}$$

where $\zeta_i, i = 1, \ldots, p$, are the partial autocorrelations at lags $1, \ldots, p$. The new family of subset models, denoted by AR$z(i_1, \ldots, i_m)$, is obtained by selecting $\zeta_{i_1}, \ldots, \zeta_{i_m}$ as parameters and constraining other partial autocorrelations as zero[1]. This model forms a subset of the AR(p) model that may be written as $\phi(B)(z_t - \mu) = a_t$, where the parameters $\phi_1, \ldots, \phi_p$ are reparameterized by Equation 2. For example, in the AR$z(1, 3)$, $\phi_1 = \zeta_1$, $\phi_2 = -\zeta_1\zeta_3$ and $\phi_3 = \zeta_3$, since $\zeta_2 = 0$. The AR$p(i_1, \ldots, i_m)$ and AR$z(i_1, \ldots, i_m)$ are similar but distinct models. For example, in the AR$p(1, 3)$, $\zeta_1 = \phi_1/(1 - \phi_1\phi_3 - \phi_3^2), \zeta_2 = \phi_1\phi_3/(1 - \phi_3^2)$ and $\zeta_3 = \phi_3$. The most important advantage of the new parameterization is the efficiency with which subset AR models with large p may be identified. This advantage is important with long and complex time series which are becoming available in massive datasets being collected in a variety of scientific fields. Asymptotic distributions of the parameter estimates and residual autocorrelations are derived in McLeod and Zhang (2006). The notation and models are summarized in Table 1 below.

Abbreviation	Description	Parameters	Lags
AR(p)	Non-subset AR model	$\phi_1, \ldots, \phi_p$	$1, \ldots, p$
AR$p(i_1, \ldots, i_m)$	Usual subset AR model	$\phi_{i_1}, \ldots, \phi_{i_m}$	$i_1, \ldots, i_m$
AR$z(i_1, \ldots, i_m)$	New family of subset AR models	$\zeta_{i_1}, \ldots, \zeta_{i_m}$	$i_1, \ldots, i_m$

Table 1: Summary of families of AR models.

2. Model identification

Table 2 lists the three functions useful in initial model selection.

[1]To represent the subset models, the notation ARp and ARz is used in this paper rather than AR$_\phi$ and AR$_\zeta$ used in our previous work (McLeod and Zhang 2006). This new notation is more convenient in the R help documentation.

Function	Purpose
`PacfPlot`	Plot partial autocorrelations and limits
`SelectModel`	Select best AR, ARz or ARp model
`TimeSeriesPlot`	Multi-panel or single-panel time series plot with aspect-ratio control

Table 2: Model identification functions.

2.1. Time series plots

Although time series plots can easily be produced with the built-in R function `plot` or `plot` method for '`ts`' objects, there are several advantages to using our function `TimeSeriesPlot`. As pointed out in Cleveland (1993), many stationary time series are best visualized with a relatively low aspect-ratio and it is awkward to do this with standard R graphics. The function `TimeSeriesPlot` adjusts the aspect-ratio using `par` settings. In the **lattice** package (Sarkar 2008), the function `xyplot` also allows one to control the aspect-ratio [2]. In general, Cleveland (1993) recommends a procedure, banking to 45^o. This algorithm determines the aspect-ratio so as to make the average slope 45^o. For time series this is only a rough guide since, for stationary time series, the resulting aspect-ratio will often be too close to zero, and for time series containing strong trends, it makes more sense to bank to the underlying trend. When strong trends are present, it is usually reasonable to use `aspect = 1` but the user should be prepared to experiment with what aspect-ratio is most effective.

Figure 1 and Figure 2 compare the `plot` method for '`ts`' objects and `TimeSeriesPlot` using the built-in `lynx` time series. Figure 2 with the lower aspect-ratio shows the asymmetry in the population cycles more clearly.

When plotting long time series the resolution may be improved by using a multipanel display which can be easily created using `xyplot`. Our function `TimeSeriesPlot` implements a lattice display for long time series. Figure 3 illustrates the multipanel time series display for the `Ninemile` treering time series.

2.2. Partial autocorrelation plot

The R function, `pacf`, plots the usual partial autocorrelations and their 95% confidence limits. Our function `PacfPlot`[3] is more useful for identifying ARz subset models. Given a maximum order P for the model, the estimated partial autocorrelations $\hat{\zeta}_1, \ldots, \hat{\zeta}_P$ are computed along with their estimated standard errors. The estimated standard errors are computed from the large-sample covariance matrix of $\hat{\zeta}_1, \ldots, \hat{\zeta}_P$ given in McLeod and Zhang (2006). The plot produced by `PacfPlot` shows the 95% confidence interval for each individual of $\hat{\zeta}_i, i = 1, \ldots, P$. From this plot, the nonzero subset of partial autocorrelations $\zeta_1, \ldots, \zeta_P$ for the ARz model may be selected.

[2] Zeileis and Grothendieck (2005) provide `xyplot.zoo` for plotting several types of time series objects in package **zoo**.

[3] `PacfPlot` uses the Burg algorithm as opposed to `pacf` which uses the Yule-Walker estimates. Many researchers, for example, Zhang and McLeod (2006a) and Percival and Walden (1993) have found that the Yule-Walker estimates are not always satisfactory when $p \geq 2$ while the least squares method may yield inadmissible estimates.

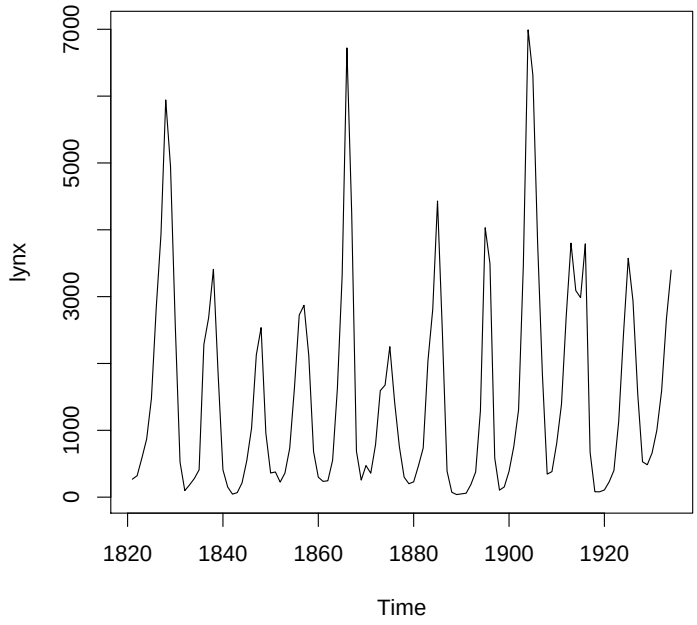

Figure 1: The `lynx` series with default aspect-ratio with `plot` method for '`ts`' objects.

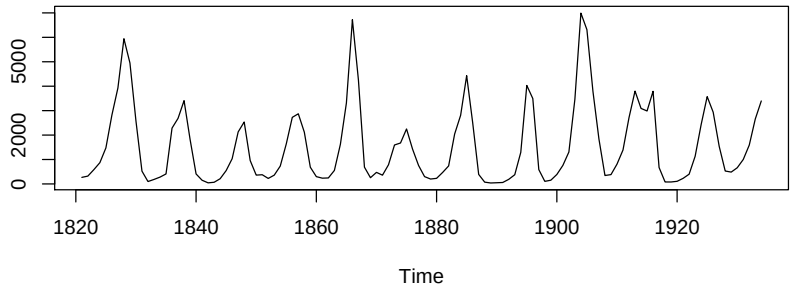

Figure 2: The aspect-ratio in `TimeSeriesPlot` has been set to 0.25. We can see more clearly the asymmetry for the `lynx` series.

The use of `PacfPlot` is illustrated in Figure 4 for the natural logarithms of the built-in `lynx` time series. This plot shows that 95% confidence intervals for $\hat{\zeta}_1$, $\hat{\zeta}_2$, $\hat{\zeta}_4$, $\hat{\zeta}_7$, $\hat{\zeta}_{10}$, and $\hat{\zeta}_{11}$ do not include zero and hence the subset partial autocorrelations at lags 1, 2, 4, 7, 10 and 11 are significantly different from zero at the 5% level. Thus the AR$z(1, 2, 4, 7, 10, 11)$ model is suggested. Since lags 4 and 10 are barely significant at 5%, an AR$z(1, 2, 7, 11)$ model may also be considered.

2.3. Automatic model selection

It is often preferable to use an automatic model selection method. We consider automatic methods for the non-subset and the subset AR models. In all model selection algorithms, we

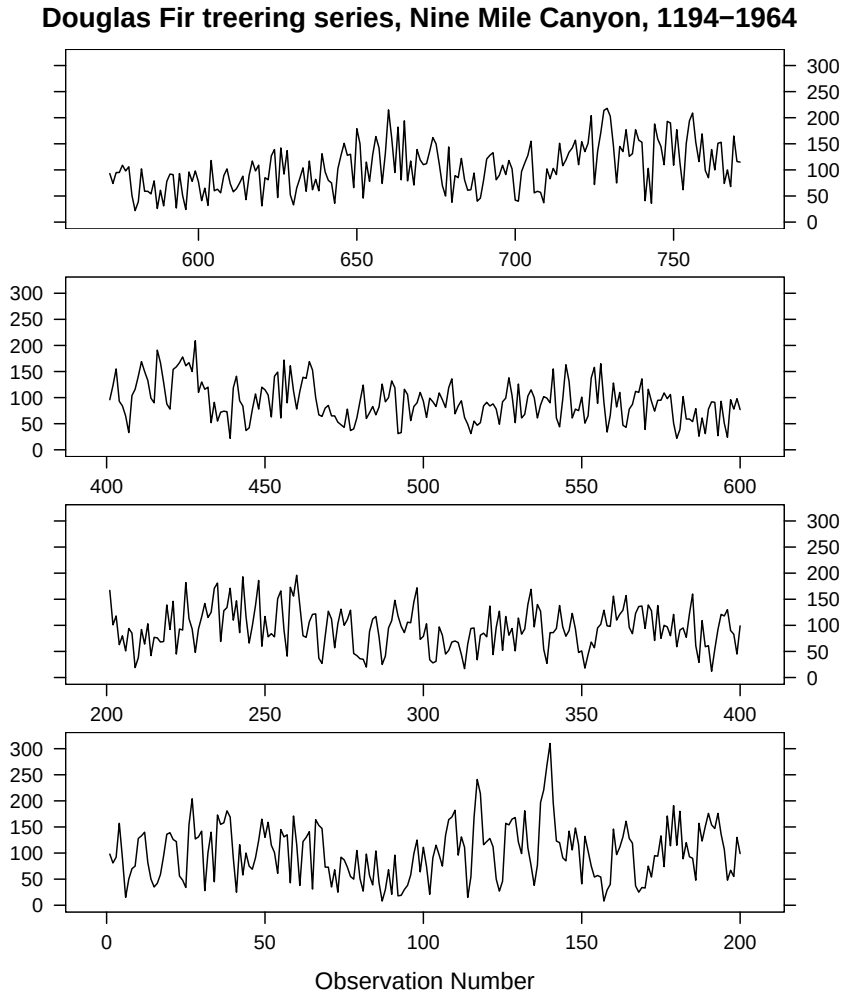

Figure 3: Trellis or lattice style graphics are useful for displaying long time series as well as for choosing a suitable aspect-ratio.

will assume that the mean parameter is included in the model and the mean is estimated by the sample mean.

Non-subset models

The approximate concentrated log-likelihood for an AR(p) model may be written,

$$\mathcal{L} = -n \log \left(\prod_{k=1}^{p} (1 - \hat{\zeta}_k^2) \right) \tag{3}$$

where n is the series length. When $p = 0$, $\mathcal{L} = 0$. The AIC model selection criterion,

$$\text{AIC} = -2\mathcal{L} + 2p \tag{4}$$

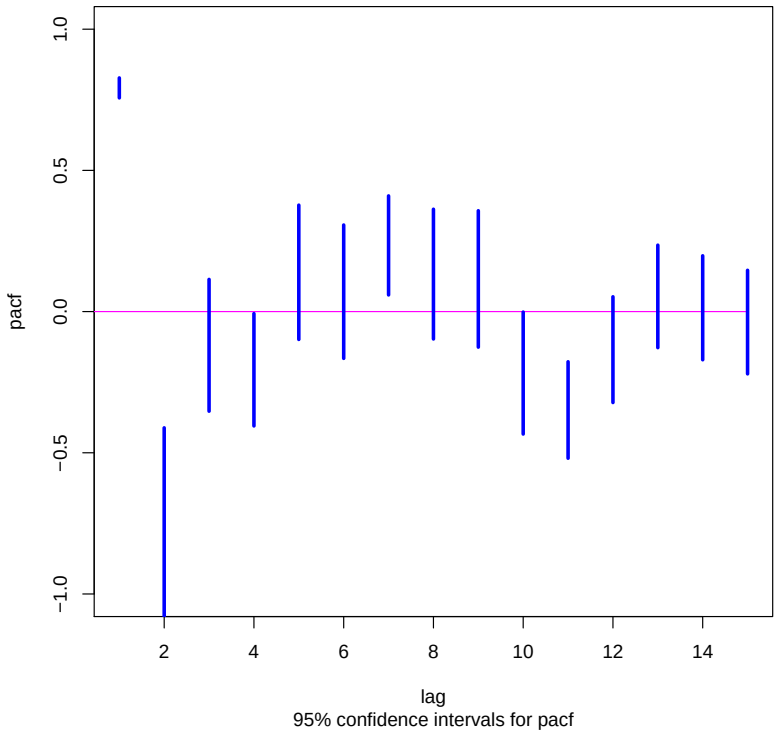

Figure 4: Partial autocorrelation plot for identifying an ARz model to the `lynx` time series.

is evaluated for $p = 0, \ldots, P$, where P is the maximum order[4]. Similarly for the BIC,

$$\text{BIC} = -2\mathcal{L} + p \log n. \tag{5}$$

Initially κ_0 candidate models are selected using the AIC/BIC evaluated using the approximate likelihood in Equation 3. These models are then refit using exact likelihood method described in Section 3 and new AIC/BIC values are computed. From these models the best $\kappa_1 < \kappa_0$ are output as the final selection. As noted by Duong (1984) and Choi (1992), it is often useful to consider not just the best model but also close runner-ups. So in this case we would choose $\kappa_1 > 1$. In the function `SelectModel`, κ_0 and κ_1 correspond to the arguments `Candidates` and `Best` which have default settings, `Candidates = 5` and `Best = 3`.

The example below illustrates why this two-step procedure is needed. We consider the high-order AR models fit by Percival and Walden (1993, Chapter 10) to the Willamette river data. Using the Akaike FPE, which is asymptotically equivalent to the AIC (Nishii 1984), Percival and Walden (1993, Section 10.15) find that the best model is an AR(38) which agrees with our function `SelectModel`:

```
R> SelectModel(log(Willamette), lag.max = 150, Criterion = "AIC")

   p       AIC      AICx
1 38 -2474.058 -485.9111
```

[4] Akaike (1970, p.216) recommends $P = n/10$. This is just an empirical rule-of-thumb. In practice, while P must be taken large enough to obtain an adequate fit, it is important not to take P too large in order to avoid over-fitting. This is our recommendation for subset models as well.

```
2 27 -2473.823 -484.9443
3 39 -2472.308 -485.0843
```

The AIC-column is based on the exact log-likelihood given in Equation 8 and the AICx-column is based on the approximate likelihood in Equation 3. In this case both methods agreed that the best model was with $p = 38$ but the exact method selected $p = 27$ as second best whereas the approximate method would have selected $p = 39$.

The default order selection criterion in `SelectModel` is BIC since in forecasting applications this often provides the most suitable model (Koehler and Murphree 1988; Granger and Jeon 2004; Hipel and McLeod 1994). For spectral density estimation using non-subset models, many authors prefer the AIC (Choi 1992; Percival and Walden 1993; Hipel and McLeod 1994).

Subset models

In the case of subset model selection of high-order AR, the model space can become very large. When the model space is large like this, the usual AIC/BIC criteria may select over-parameterized models (Chen and Chen 2008; Broman and Speed 2002). Chen and Chen (2008) have developed the UBIC criterion for this large model space problem. Therefore, for subset model selection, we have implemented UBIC as the default in `SelectModel` [5]. For comparison, the usual AIC/BIC criteria are also implemented. The computation of these criteria is outlined next.

The approximate concentrated log-likelihood function for an AR$z(i_1, \ldots, i_m)$ evaluated at $\hat{\zeta}_1, \ldots, \hat{\zeta}_m$ may be written (McLeod and Zhang 2006, Equation 15),

$$\mathcal{L} = -n \log \left(\prod_{k \in \{i_1, \ldots, i_m\}} (1 - \hat{\zeta}_k^2) \right) \tag{6}$$

where n is the series length. The UBIC criterion (Chen and Chen 2008) may be written,

$$\text{UBIC} = -2\mathcal{L} + m \log n + 2 \log \binom{P}{m}, \tag{7}$$

where P is the maximum possible lag. Any value of P such that $P \geq i_m$ produces an equivalent ordering of the UBIC.

The usual AIC/BIC criteria defined in Equation 4 and Equation 5 may also be used but these criteria are not recommended for the subset problem. For the model order selection problem in the non-subset case, the UBIC reduces to the BIC.

The UBIC or AIC/BIC subset can be found simply by ordering the parameters in ascending order of magnitude, $|\hat{\zeta}_{i_1}| \leq |\hat{\zeta}_{i_2}| \leq \ldots \leq |\hat{\zeta}_{i_P}|$, and then evaluating successively $\mathcal{L}$ starting with $\mathcal{L} = 0$ for the null model containing only the mean parameter[6]. As in the non-subset model selection algorithm, a second pass is made after an initial number of candidate models are found, in which each model is refit using exact maximum likelihood. For the `log(lynx)` data, we find using the UBIC,

[5] In the non-subset case, our problem is model order selection. In this case, the model space is not large and the UBIC criterion reduces to the usual BIC.

[6] As in `PacfPlot`, the Burg estimates are used for the partial autocorrelations.

```
R> SelectModel(log(lynx), lag.max = 15, ARModel = "ARz", Best = 1)

[1]  1  2  7 10 11
```

and this agrees well with the model suggested by the `PacfPlot`. To summarize the procedure used in this example, the best 5 models were found using the approximate likelihood to estimate the UBIC. Then for each of these 5 models, the exact likelihood was used to determine the UBIC for each model. Then the best model, the AR$z(1, 2, 7, 10, 11)$ was found. For this data, the model space is not very large and consequently, it may be shown that the BIC selects the AR$z(1, 2, 7, 10, 11)$.

Model identification for subset ARp models is supported using the R subset regression package **leaps** (Lumley and Miller 2004). Using this package the best ARp subset model is identified using a two-step procedure. In the first step, the `leaps` function is used to find the AR$p(i_1, \ldots, i_m)$ subsets which when fit using least squares have minimum residual variance for each i_m, $m = 0, 1, \ldots, P$ parameters. For the next step, all P models are refit using our least squares AR fitting algorithm, `GetFitARpLS`, and the exact log-likelihood is determined using `LoglikelihoodAR`. Based on these log-likelihoods, the UBIC/BIC/ AIC is re-evaluated and the best model is selected.

As shown below, the model selected by Tong (1977) for the `lynx` series, AR$p(1, 2, 4, 10, 11)$, is selected using `SelectModel`,

```
R> SelectModel(log(lynx), lag.max = 15, ARModel = "ARp", Criterion = "BIC",
+ Best = 1))

[1]  1  2  4 10 11
```

however, the UBIC selects the more parsimonious AR$p(1, 2, 9, 12)$. When `Best > 1`, the output from `SelectModel` is a list and is defined as a '`SelectModel`' S3 class object. This output may be graphically viewed using the `plot` function. For the `lynx` dataset, Figure 5 compares the 3 best ARp and ARz subset models selected using the UBIC and BIC. In Figure 5, the left vertical axis shows the scale in terms of BIC while the right one uses relative plausibility, defined by $\exp\{(BIC_1 - BIC_0)/2\}$, where BIC_0 and BIC_1 denote the BIC values for the best model and the alternative respectively. Likelihood and plausibility for time series models were introduced by Akaike (1978) and Akaike (1979). As with the relative likelihood (Fisher 1959; Royall 1997; Sprott 2000), alternative models with relative plausibility less than 1% may be said to be implausible. As shown in Figure 5, the BIC and UBIC select the same best three ARz models although for the ARp a more parsimonious model is selected using the UBIC than with the BIC.

The best subset for ARz models is easily found even when P is large since only sorting is needed. On the other hand, for the ARp best subset selection, the **leaps** package (Lumley and Miller 2004), available on the Comprehensive R Archive Network, is used and this package implements an exponential-time branch-and-bound algorithm. The function used, `leaps`, is interfaced to Fortran for maximum efficiency. But still it is limited to a maximum order of around $P = 50$ or so due to the rapidly increasing computing time needed as P increases. Thus it is possible to fit much higher order subset models using the ARz model. This is illustrated with the Zurich monthly sunspots $1749 - 1983$ available in R as a built-in dataset, `sunspots`.

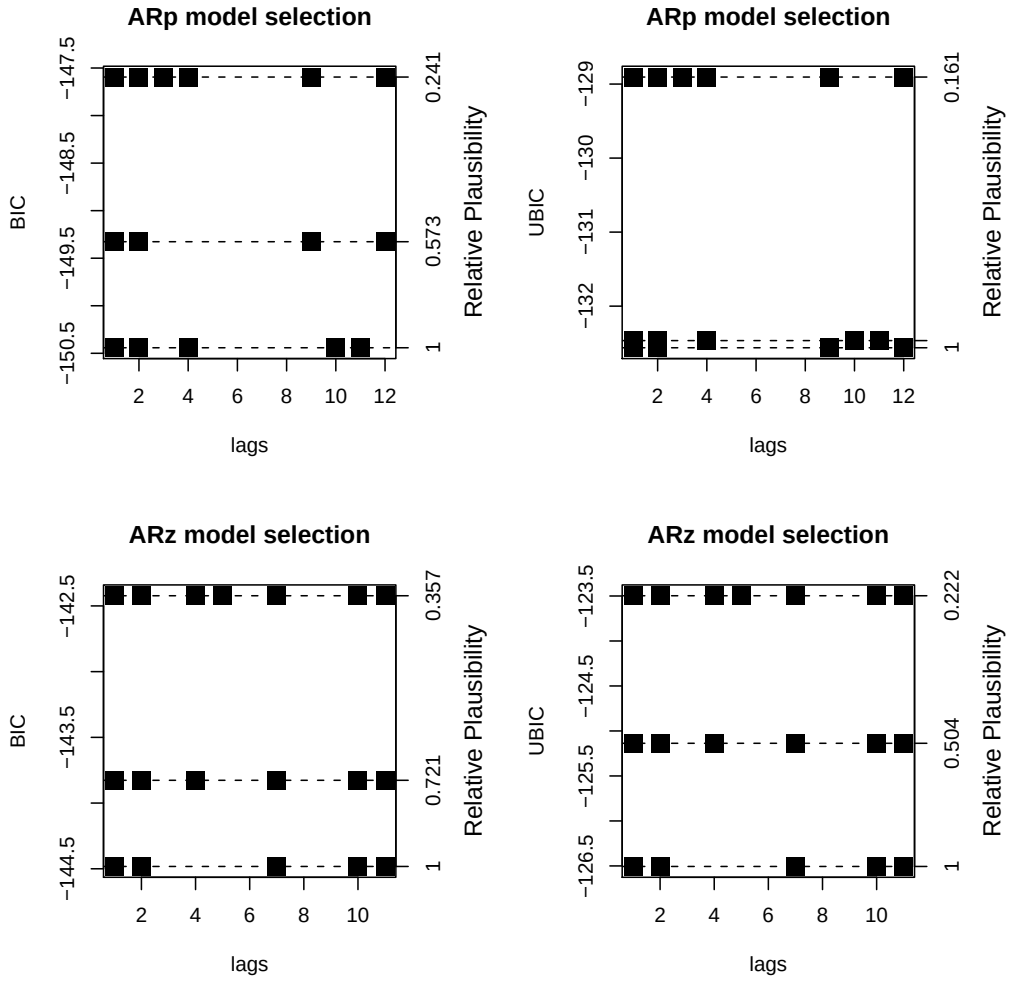

Figure 5: Plot of the output from `SelectModel` with `Best = 3` for ARp and ARz models using BIC and UBIC for subset selection for the logged `lynx` series.

Taking a maximum model order of $P = 200$ and using a square-root transformation, the best non-subset AR and subset ARz models were determined. In the case of AR model selection the AR(27) and AR(21) models were selected using the AIC and BIC criteria. For the subset selection, AIC, BIC and UBIC were used and the number of parameters not including the mean in the selected models were 55, 18 and 8 respectively for AIC, BIC and UBIC. Figure 6 compares the spectral density functions for the estimated models. The AIC subset model spectral density was very jagged and is not shown in Figure 6[7].

3. Estimation

The log-likelihood function given n time series observations $z_1, \ldots, z_n$ from a covariance sta-

[7] See Example 3 in the documentation for `sdfplot` for an R script to estimate the spectral density function for the monthly sunspot series by using the AIC criterion to fit an ARz model.

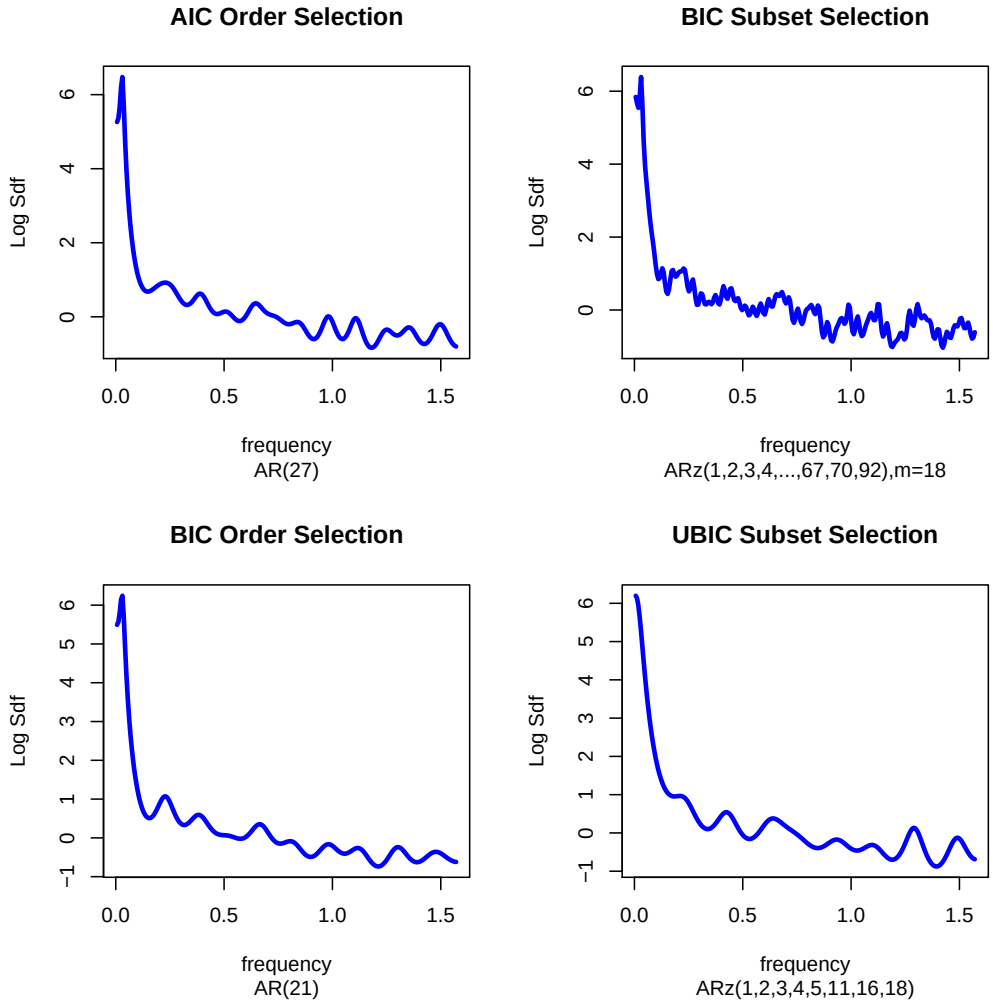

Figure 6: Logged spectral density functions of the non-subset AR and subset ARz models fitted to the square-root of the `sunspots` series.

tionary Gaussian time series with mean μ and autocovariance function $\gamma_k, k = 0, 1, \ldots$, may be determined from the multivariate normal distribution. Letting $M_n = \Gamma_n/\sigma_a^2$ where Γ_n is the covariance matrix of n successive observations and maximizing the log-likelihood function over σ_a^2, the concentrated log-likelihood may be written, after dropping constants,

$$L(\phi) = -\frac{n}{2}\log(S(\phi)/n) - \frac{1}{2}\log(g_n), \tag{8}$$

where $S(\phi) = z' M_n^{-1} z$, $g_n = \det(M_n)$ and $\hat{\sigma}_a^2 = S(\phi)/n$ where $\phi = (\phi_1, \ldots, \phi_p)$. Note that $\mathcal{L}$ given in Equation 6 is an affine large-sample approximation to the exact log-likelihood given in Equation 8.

Assuming that the mean is known, the exact log-likelihood function may be computed in $O(1)$ flops independent of n in repeated function evaluations after an initial setup computation is performed (McLeod and Zhang 2006, Section 2.1). This is done by conditioning on the sufficient statistics, $D_{i,j} = z_i z_j + \cdots + z_{n-j} z_{n-i}$, where we have assumed a mean-zero time

series. The R function `optim` is then used to maximize the log-likelihood function. Initial starting values are obtained using Burg estimates. This exact maximum likelihood estimation algorithm is implemented in our functions `FitAR` and `GetARFit`. The function `GetARFit` may be preferred for bootstrapping applications while `FitAR` is intended as the main function for model fitting when diagnostic checking and forecasting are important.

`FitAR` also computes the covariance matrix of the estimates, residuals, fitted values and some diagnostic checks. The function `FitAR` returns a S3 class object '`FitAR`' and methods functions `print.FitAR`, `summary.FitAR`, `coef.FitAR`, `residual.FitAR`, and `plot.FitAR` are supplied. In addition some new generic functions `Boot` and `sdfplot` are defined and methods are given for class '`FitAR`', '`ts`' and other objects of interest. More methods are supplied for these functions in McLeod, Yu, and Krougly (2007).

`FitAR` and `getFitAR` use several other functions which may be of interest in some applications and are described briefly in Appendix A. In Appendix B, we discuss the perils of fitting the ARp model using exact MLE. This provides another reason for preferring the ARz model.

3.1. Exact MLE for mean

Assuming that $(\phi_1, \ldots, \phi_p)$ are known, the exact MLE for mean μ is given by

$$\hat{\mu} = \frac{1_n' \Gamma_n^{-1} z}{1_n' \Gamma_n^{-1} 1_n}, \tag{9}$$

where 1_n denotes the n dimensional column vector with all entries equal to 1, $z = (z_1, \ldots, z_n)$ and Γ_n^{-1} denotes the inverse of the covariance matrix of n successive observations. Since $\hat{\mu}$ does not depend on σ_a^2, we may assume without loss of generality that $\sigma_a^2 = 1$. Direct evaluation of Equation 9 using the exact inverse matrix derived by Siddiqui (1958) would require $O(n^2)$ flops. A more efficient algorithm is given in McLeod and Zhang (2008) for evaluating Equation 9 in $O(n)$ flops and this is implemented in our R function `GetARMeanMLE`. As a check for the correctness of `GetARMeanMLE`, in the example section of the help documentation for `GetARMeanMLE`, we provide a R script which compares the results obtained by computing the MLE of the mean μ directly and using `GetARMeanMLE`.

When $(\phi_1, \ldots, \phi_p)$ is unknown, the iterative algorithm given in McLeod and Zhang (2008, Section 2.2) may be used to obtain the simultaneous joint MLEs of μ and the other model parameters. This iterative algorithm is implemented in `FitAR` with the option `MeanMLEQ = TRUE`. In the documentation for `FitAR`, an example is given to demonstrate the agreement with results obtained by the built-in function `arima`.

Some simulations were carried out to compare the timings for `GetFitAR`, `FitAR` and the built-in function `ar` for fitting nonsubset AR$(p), p = 1, 2$ and $p = 20, 40$. The main purpose of these timings is just to indicate the relative computer times for the three different algorithms. As far as accuracy is concerned, there is no meaningful difference in accuracy between `FitAR` and `ar` when the latter function converges. As noted below, `ar` does not always work and sometimes it fails completely. It seems reasonable to recommend that users avoid the exact MLE option in `ar`. Between `GetFitAR` and `FitAR` there are some slight differences in the estimates but overall there does not seem to be any notable difference in accuracy for the models in this experiment.

For each model, the admissible parameters were randomly selected by randomly selecting partial autocorrelations uniformly from -1 to 1 and then reparameterizing through Equation 2.

n	GetFitAR	FitAR	ar	GetFitAR	FitAR	ar
		AR(1)			AR(2)	
50	0.01	0.02	0.02	0.02	0.08	0.03
100	0.01	0.03	0.03	0.02	0.08	0.03
200	0.02	0.05	0.04	0.03	0.10	0.04
500	0.04	0.09	0.06	0.05	0.14	0.07
1000	0.07	0.15	0.09	0.09	0.21	0.11
		AR(20)			AR(40)	
1000	0.14	5.52	2.04	0.24	34.66	47.60
2000	0.20	5.52	1.98	0.34	22.99	51.60
5000	0.37	5.89	2.60	0.60	24.09	44.85

Table 3: Timings for `GetFitAR`, `FitAR` and `ar`. For each n and each p, 100 series were simulated and the models estimated using the functions `GetFitAR`, `FitAR` and `ar`. The average time over all 100 simulations is reported for each parameter combination. These timings are illustrative of the relative performance of the functions but are *highly system dependent*. A 3.6 GHz Pentium PC with R Version 2.70 running Windows XP was used.

Various lengths of series were used from $n = 50, 100, 200, 500, 1000$. The timings do not vary much once n and p are fixed but the timing for a single fitting is too short to get an accurate value for low order models, so 100 simulations were done for each of these parameter settings. The average CPU times required for simulations and fitting are given below in Table 3. During fitting, `ar` reported one error in `optim` which is shown below:

```
Error in optim(init[mask], arma0f, method = "BFGS", hessian = TRUE,
   control = optim.control): non-finite finite-difference value [3]
```

This error was trapped using the built-in R function `try`. Many warning messages about convergence were also generated by `ar`. On the other hand `GetFitAR` or `FitAR` worked correctly in all cases without generating any messages. The experiment was repeated, this time with $p = 20, 40$ and $n = 1000, 2000, 5000$. Once again `ar` generated error and warning messages. With `GetFitAR`, the sample mean was used but for `FitAR` the iterative algorithm was used to compute the exact MLE for the mean as well as the other parameters. With `ar` the option `method = "mle"` was used. All timings reported in this paper are for a Pentium 4 processor at 3 GHz PC running Windows XP with 2 GB RAM. The CPU times in Table 3 show that `FitAR` is much faster than `ar` for $p = 40$. If the sample mean is used then `GetFitAR` is always very fast and for most AR models that seem to occur in practice, there is not much difference between using the sample mean and the exact MLE.

3.2. Box-Cox analysis

Box and Cox (1964) derived a maximum likelihood method for estimating the transformation parameter λ in the family of transformations,

$$z_t^{(\lambda)} = \begin{cases} (z_t^\lambda - 1)/\lambda & \lambda \neq 0 \\ \log(z_t) & \lambda = 0 \end{cases}$$

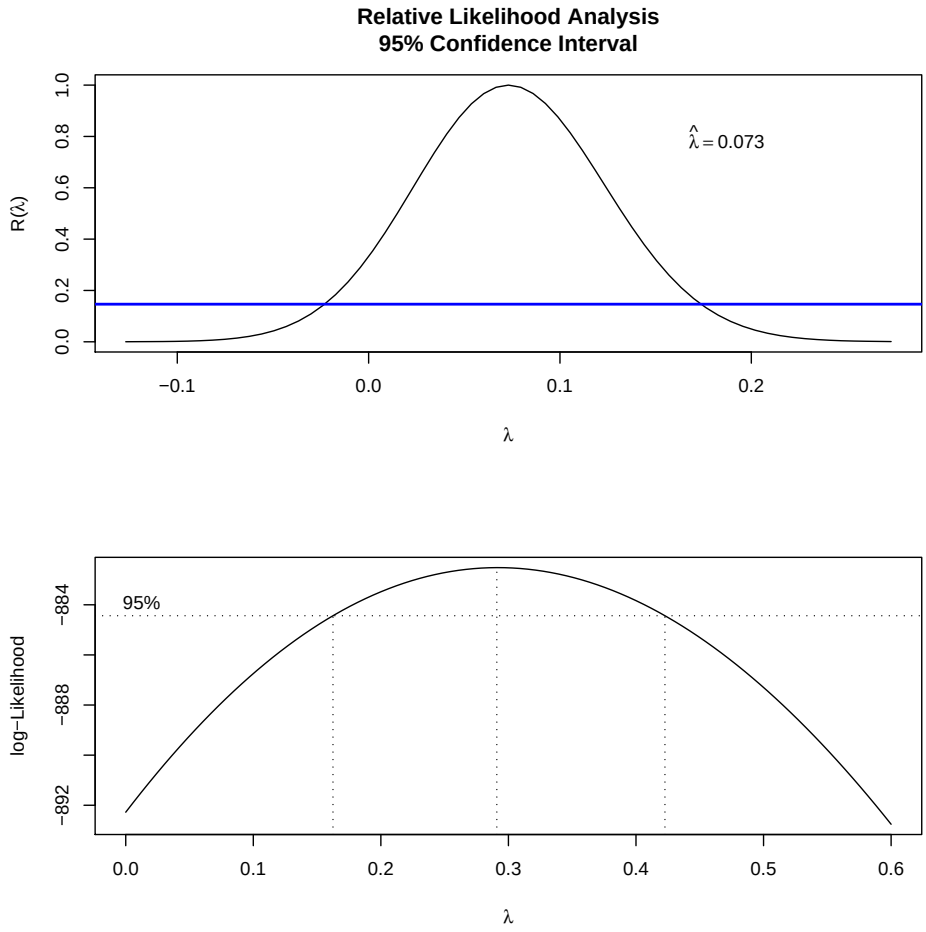

Figure 7: Top graph shows output from `BoxCox` method for a '`FitAR`' object and bottom graph from `boxcox` for fitting AR$p(1, 2, 4, 10, 11)$ to `lynx` series.

for data $z_t, t = 1, \ldots, n$, from a normal linear regression and ANOVA. This method was adapted to ARIMA time series in Hipel and McLeod (1977, Equation 10, p.571). For autoregressive models, the concentrated loglikelihood function can be written,

$$L(\phi, \lambda) = -\frac{n}{2}\log(S(\phi)/n) - \frac{1}{2}\log(g_n) + (1-\lambda)\sum_{t=1}^{n}\log(z_t). \tag{10}$$

For fixed λ we can maximize over ϕ to obtain $L(\lambda)$. The function $L(\lambda)$ may be maximized numerically with the `optimize` function to obtain the MLE, $\hat{\lambda}$. It is convenient to plot the relative likelihood function $R(\lambda) = L(\lambda)/L(\hat{\lambda})$. The relative likelihood, $R(\lambda)$, itself provides a quantification of the relative plausibility of various values of λ (Sprott 2000, Section 2.4 and Section 4.5). It may be shown that a 95% confidence interval based on the likelihood-ratio test corresponds to $R(\lambda) \geq 0.1465$. A horizonal line is drawn to indicate a 95% confidence interval for λ.

The function `BoxCox` is implemented as a generic function with methods for classes '`FitAR`', '`Arima`', '`ts`' and '`numeric`'. `BoxCox` method for '`FitAR`' objects parses its input to construct

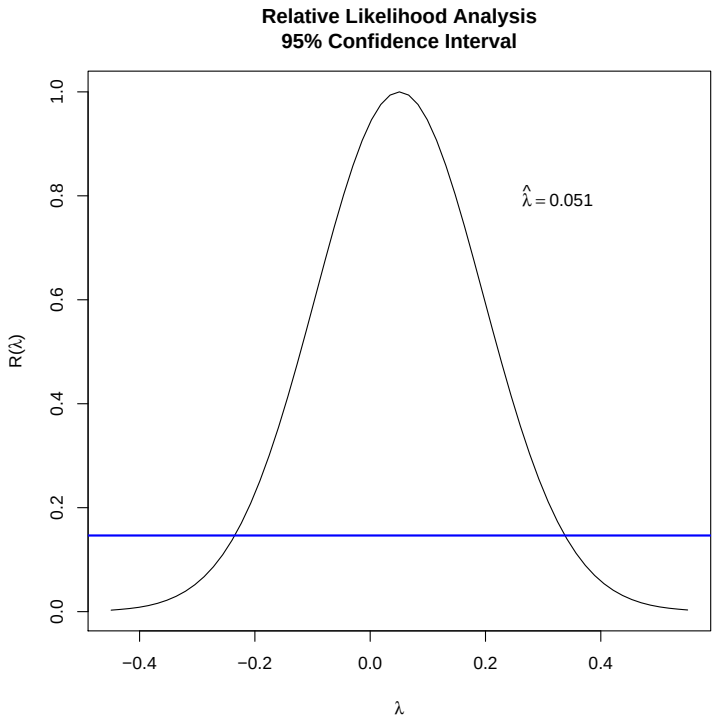

Figure 8: Box-Cox analysis produced by `BoxCox(AirPassengers)`.

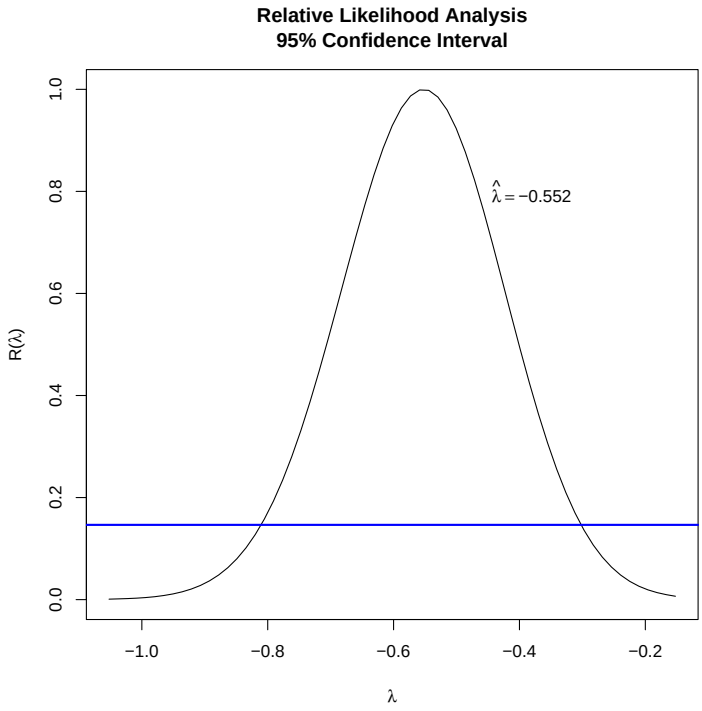

Figure 9: Box-Cox analysis produced by `BoxCox(rivers)`.

the likelihood function in Equation 10 and then the MLE is determined using `optimize`. A plot of the likelihood function is determined to show all values of λ with plausibility greater than 1% as well as a horizontal line indicating the 95% confidence interval. In Figure 7, we compare the Box-Cox analysis for the `lynx` time series using our function and the `boxcox` function in the R package **MASS** (Venables and Ripley 2002). For the `lynx` time series we fit the AR$p(1, 2, 4, 10, 11)$ model by least-squares using `FitARp`. In this case, the output from `FitARp` contains the design matrix for the regression as well as the column for the dependent variable. These results were then used with the **MASS** `boxcox` function to obtain the bottom plot in Figure 7. The difference in the plots is substantial. The top plot using `BoxCox` method for the '`FitAR`' object shows that a log transformation is reasonable choice whereas the bottom plot strongly suggests otherwise and favors a transformation such as a cube-root. The reason for the difference is that in `BoxCox` method for '`FitAR`' objects the log-likelihood in Equation 10 is used whereas in `boxcox` the approximate log-likelihood corresponding to linear regression is used.

`BoxCox` method for '`ts`' and '`numeric`' objects enable one to do a Box-Cox analysis for arbitrary time series as well as simple random samples. With `BoxCox` method for a '`ts`' object, the untransformed data is fit by a high-order AR(p) model which is then used for Box-Cox analysis. In practice, this procedure often works well even for non-stationary time series as is shown in Figure 8 for the R dataset `AirlinePassengers`. An example of using `BoxCox` method for '`numeric`' objects for the R built-in `rivers` data set is shown in Figure 9.

The `BoxCox` method for '`Arima`' objects implements a Box-Cox analysis for time series fit with the R `arima` function. An illustrative example is provided by the time series of annual production of tobacco in the U.S. for the period 1871-1984 (See Figure 10). Note the increase in variability. Since this variability is related to level, a model of the Box-Cox transformed series provides a much simpler alternative to the more complex ARIMA-GARCH approach (Wei 2005, p.379). Wei (2005, p.120) also suggested an ARIMA$(0, 1, 1)$ to the logarithms of this series. However Box-Cox analysis indicates that a square-root transformation works much better than logarithms (Figure 11).

4. Diagnostic checks

A methods function is supplied so that the generic function `plot` can be used for a variety of diagnostic plots for '`FitAR`' class objects. The default is with option `terse = TRUE` which produces a plot of the p-values of the Ljung-Box portmanteau test (Ljung and Box 1978) and a plot of the residual autocorrelations with simultaneous 95% limits (Hosking and Ravishanker 1993). Both these plots appear as a panel display. These are usually the most important diagnostics. The default diagnostic plots shown in Figure 12 are produced by:

```
R> ans <- FitARp(log(lynx), c(1, 2, 4, 10, 11))
R> plot(ans)
```

When `terse = FALSE`, after the plots shown in Figure 12, seven more plots are produced in the order as follows:

- Normal probability plot and the p-value for an ominbus normality test of the Jarque-Bera test (Jarque and Bera 1987). This is useful in detecting outliers or thick tails.

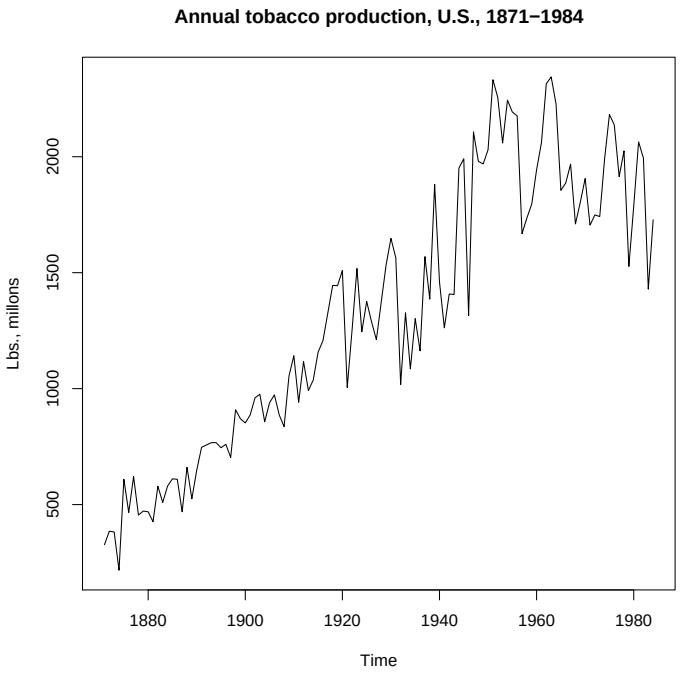

Figure 10: Time series plot for annual tobacco production series.

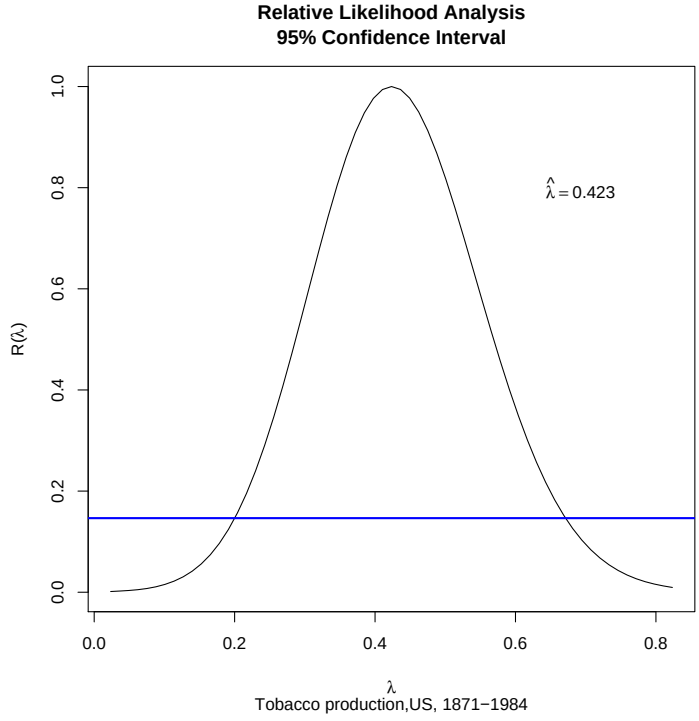

Figure 11: Box-Cox analysis for annual tobacco production series.

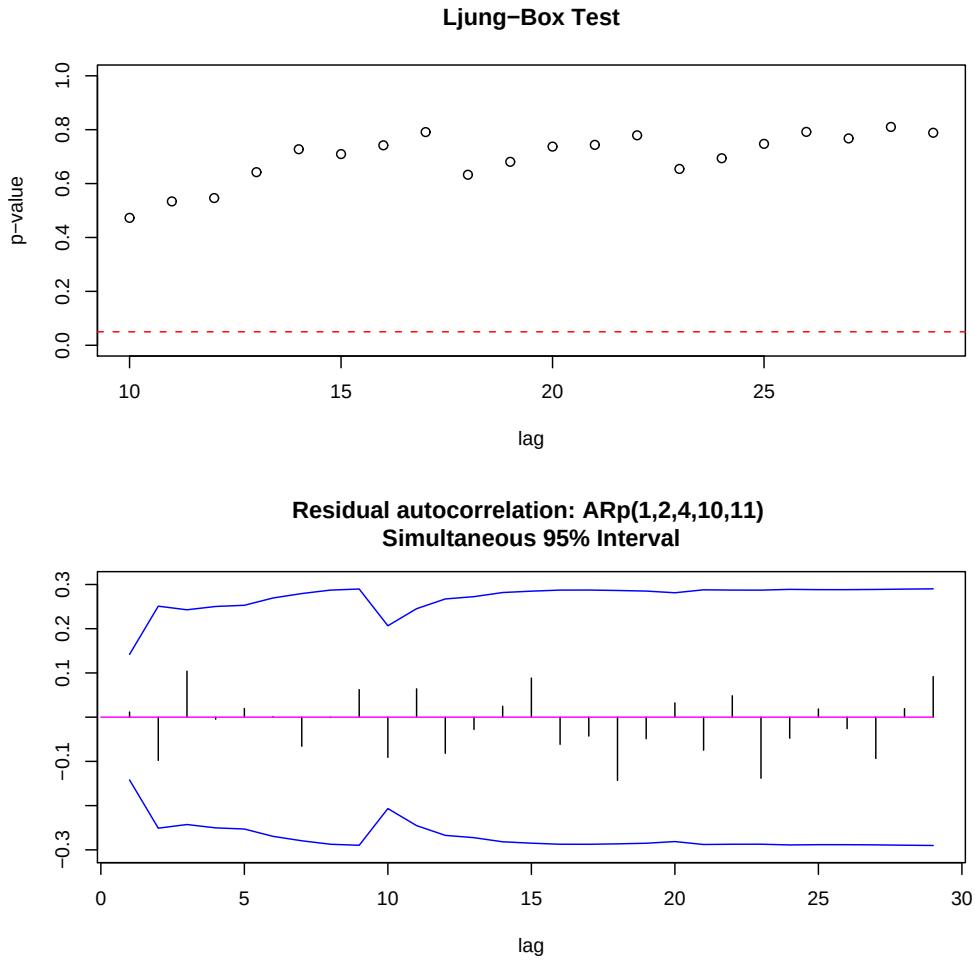

Figure 12: Basic diagnostic plots for AR$p(1, 2, 4, 10, 11)$ model fit to the logarithm of the `lynx` series.

Thick tails are a characteristic of ARCH/GARCH models (Tsay 2005, Chapter 3) so this test may suggest that some form of conditional heteroscedasticity is present.

- Box-and-whisker plot of the residuals. This is useful in detecting skewness. Skewness may indicate a power transformation. On the other hand, if the residuals are symmetric but outliers are present, some form of conditional heteroscedasticity may be present. Various types of ARCH/GARCH models may be used for modeling conditional heteroscedasticity.

- Plots of the time series and a bootstrap version of the time series. This gives some idea of whether or not important features in the time series appear in the simulated model.

- Monotone spread plot (Cleveland 1979, 1993). This type of residual diagnostic plot is useful for detecting situations where the variance depends on the level or mean. In general Cleveland (1993) suggested plotting the square root of the absolute residual vs. the fitted value and then visualizing the relationship using a suitable loess smooth. The

R function `lowess` with `f = 1` is used. For time series models, the fitted value represents a conditional mean. If the model is adequate, the loess curve should be approximately horizontal. If it is monotonic up or down, a power transformation could be used to remove this dependence.

- Residual-fit spread (RFS) plot (Cleveland 1993). This plot is useful for visualizing how much of the variation in the data is explained by model and so whether the model may be useful for forecasting. Alternatively, the coefficient of determination, R^2, defined as the ratio of the variance of the one-step forecast divided by the series variance may be used. For the `lynx` and AR$p(1, 2, 7, 10, 11)$, $R^2 = 85\%$, as shown below,

```
R> z<-log(lynx)
R> var(fitted(FitARp(z,c(1, 2, 7, 10, 11))))/var(z)

[1] 0.8536185
```

The RFS plot is shown in Figure 13.

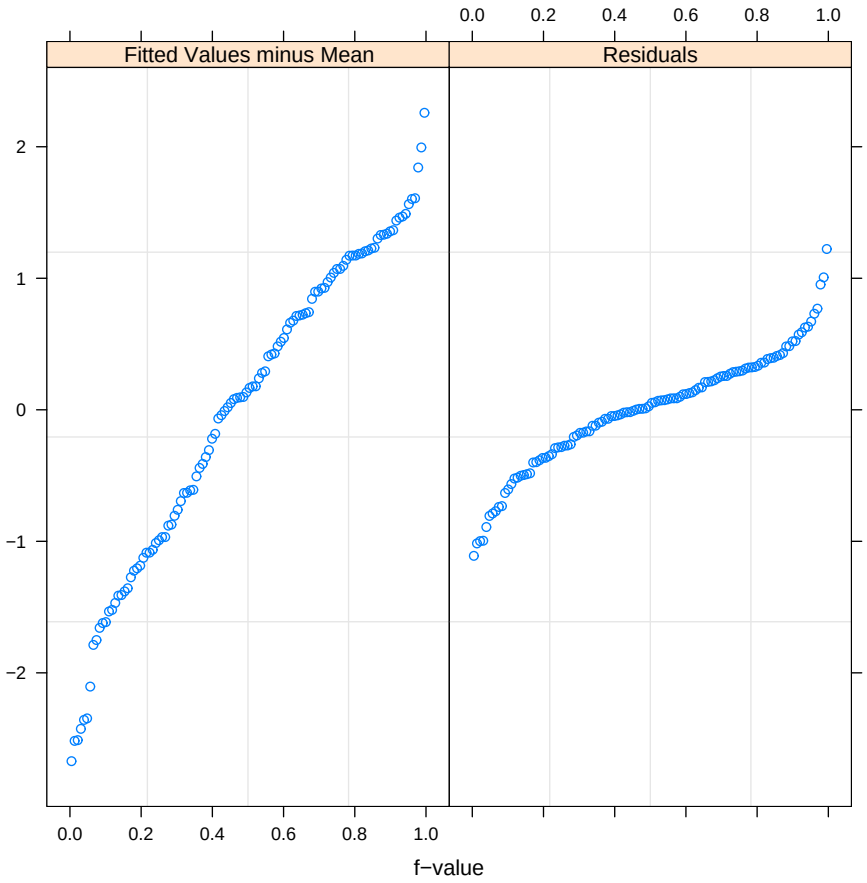

Figure 13: RFS plot for AR$p(1, 2, 4, 10, 11)$ model fit to the logarithm of the `lynx` series. For this model, $R^2 = 85\%$.

- Plots of the observed autocorrelation function, the fitted theoretical autocorrelation function and the sample autocorrelation of a parametric bootstrap. In many cases a good model can be expected to match the main features of the sample autocorrelations, particularly at the lower lags. However, caution is necessary since it is well known that the sample autocorrelations may be highly autocorrelated with large variances (Box, Jenkins, and Reinsel 1994, Section 2.1.6). In some cases, such as highly autocorrelated fractional noise, the sample autocorrelations may have very large biases (Newbold and Agiakloglou 1993). In this situation, even when the model has been correctly specified and fit, the sample autocorrelations may not resemble very well the theoretical autocorrelations. This is quite a surprising, but in hindsight, not an unreasonable result. To the degree that fractional noise may be approximated by a high-order AR model, this problem can be expected to occur with some AR models as well. In summary, we can say that it may happen that even for a correctly fit model, the sample and theoretical autocorrelations might not match very well and there may be more similarity in the sample autocorrelations of the data and the bootstrap version.

- Plot of the fitted spectral density function. This plot provides a frequency domain summary of the model.

Because of the large number of graphics windows, the default behavior of `plot` is to delete all graphics windows before producing the plots. This default can be bypassed by using the optional argument `clearGraphics = FALSE`. Some of the diagnostic checks are also available as separate functions as shown in Table 4. For more details, please see the online documentation.

5. Other useful time series functions

We give a brief overview of the functions shown in Table 5. For more details and examples, please see the online R documentation.

5.1. `AcfPlot`: Correlation plots

The function `AcfPlot` is useful for plotting other types of correlation functions, for example, the residual autocorrelations or the inverse autocorrelations. Cleveland (1971) used the inverse autocorrelations to select an AR$p(1, 2, 7)$ model for the Series A time series. The example given in the online help documentation for `AcfPlot` shows how to compute and plot the inverse autocorrelations.

5.2. `AR1Est`: Exact MLE for AR(1)

The function `AR1Est` evaluates the exact MLE for a mean-zero AR(1) model using a closed form solution (Zhang 2002). This function is useful in bootstrapping and Monte-Carlo since it is fast, accurate and reliable.

5.3. `ARSdf` and `PlotARSdf`: Autoregressive spectral density

The Fast Fourier Transform is used to evaluate the AR spectral density function,

Function	Purpose
`Boot`	Generic bootstrap function
`Boot.FitAR`	Simulate a fitted AR
`Boot.ts`	Parametric time series bootstrap
`LjungBoxTest`	Ljung-Box test for randomness
`LBQPlot`	Plot Ljung-Box test p-value vs lag
`RacfPlot`	Residual autocorrelation plot
`JarqueBeraTest`	Jarque-Bera normality test

Table 4: Model diagnostic checking functions.

Function	Purpose
`AcfPlot`	Basic ACF plotting
`AR1Est`	Exact MLE for AR(1)
`ARSdf`	Autoregressive spectral density function
`ARToMA`	Coefficients in infinite moving average expansion
`ARToPacf`	Reparametrize AR coefficients in terms of PACF
`cts`	Concantenate time series
`BackcastResidualsAR`	Innovation residuals in AR
`InformationMatrixAR`	Information matrix for AR(p)
`InformationMatrixARp`	Fisher information matrix subset case, ARp
`InformationMatrixARz`	Fisher information matrix subset case, ARz
`InvertibleQ`	Test if invertible or stationary-casual
`PacfDL`	Partial autocorrelations via Durbin-Levinson
`PacfToAR`	Transform from PACF parameters to AR coefficients
`PlotARSdf`	Plot AR or ARMA spectral density
`sdfplot`	Autoregressive spectral density estimation
`sdfplot.FitAR`	Autoregressive spectral density estimation for '`FitAR`'
`sdfplot.Arima`	Spectral density of fitted ARIMA model
`sdfplot.ar`	Spectral density of fitted ARIMA model
`sdfplot.ts`	Autoregressive spectral density estimation for '`ts`' object
`sdfplot.numeric`	Autoregressive spectral density estimation for '`numeric`'
`SimulateGaussianAR`	Autoregression simulation
`Readts`	Input a time series
`TacvfAR`	Theoretical autocovariance function of AR
`TacvfMA`	Theoretical autocovariances for moving average process
`VarianceRacfAR`	Covariance matrix residual autocorrelations for AR
`VarianceRacfARp`	Covariance matrix residual autocorrelations for ARp
`VarianceRacfARz`	Covariance matrix residual autocorrelations for ARz

Table 5: Other useful time series functions.

$$f(\lambda) = \frac{1}{2\pi} \frac{\sigma_a^2}{|\phi(e^{-2\pi i\lambda})|^2} \tag{11}$$

at a large number of equally spaced frequencies in the range $(0, \pi)$. By default $2^8 = 256$ equally spaced frequencies, $\lambda_j = 0.5\pi j/256, j = 1, \ldots, 256$. The function `ARSdf` produces the vector output $(f(\lambda_1), \ldots, f(\lambda_{256}))$ taking $\sigma_a^2 = 1$ in Equation 11.

5.4. `ARToMA`: Moving-average approximation

This function is used in computing the variances of the residual autocorrelations and in constructing plots of the residual autocorrelations. This type of computation arises also in other computations such as in the confidence limits for forecasts from AR models.

5.5. `ARToPacf` and `PacfToAR`: Reparameterization

These functions are central to working with ARz models. For many purposes it is necessary to convert from one parameterization to the other.

5.6. `BackcastResidualsAR`: Compute residuals

The innovation residuals for a fitted AR(p) may be computed recursively from

$$\hat{a}_t = (z_t - \hat{\mu}) - \phi_1(z_{t-1} - \hat{\mu}) - \ldots - \phi_p(z_{t-p} - \hat{\mu}), \tag{12}$$

for $t = p + 1, \ldots, n$. For $t = 1, \ldots, p$, the conditional expected value of a_t may be computed by backforecasting the z_t for $t = 0, -1, -\ldots, Q$ for Q large enough so that the backforecast value of z_t is approximately μ for all $t \leq Q$. Then the required residuals may be computed directly using Equation 12. This backcasting approach is described in detail in Box *et al.* (1994, Section 6.4.3).

5.7. Concatenation of time series

The function `cts` allows values to be easily concatenated to an existing time series object. See documentation for more details.

5.8. Information matrix

The large-sample Fisher information per observation can be obtained for AR, ARp and ARz models. This is used in `FitAR` to obtain the estimated standard errors of the estimated parameters. Standard errors can also be obtained by using the observed Fisher information that is obtained by numerically differentiating the log-likelihood function. Our preference is to use the theoretical information matrix since it is well-known that numerical differentiation may be unreliable (Fröberg 1969, Chapter 9, p.192). We have found for simulations and bootstrapping applications this unreliability does occasionally arise resulting in negative variances and other difficulties. Standard errors may also be estimated by bootstrapping (Box and Luceño 1997). Table 6 below compares the bootstrap estimates with the large-sample ones. The script to generate these results is included in the online documentation for `Boot` method

Estimate	Bootstrap	Large-sample
$\hat{\zeta}_1$	0.024	0.018
$\hat{\zeta}_2$	0.095	0.086
$\hat{\zeta}_4$	0.069	0.063
$\hat{\zeta}_7$	0.112	0.099
$\hat{\zeta}_{10}$	0.113	0.089
$\hat{\zeta}_{11}$	0.091	0.088

Table 6: Comparison of bootstrap and large-sample estimates of the standard deviations of the parameter estimates in an AR$z(1, 2, 4, 7, 10, 11)$ fitted to the logged `lynx` time series using 100 bootstrap replications.

for '`FitAR`' class objects. It took about 18 seconds for the 100 bootstrap replications used in Table 6.

5.9. `PacfDL`: Partial autocorrelations

`PacfDL` implements the Durbin-Levinsion algorithm to compute the partial autocorrelations and the optimal linear predictor given a sequence of autocovariances. This function is not used in our package but we include it since it can be useful in many applications. For example, Hipel and McLeod (1977) and Hipel and McLeod (1994) pointed out that the inverse partial autocorrelations may be useful in identifying MA(q) time series models. After computing the inverse autocorrelations we can then directly compute the inverse partial autocorrelations using `PacfDL`. An example of this is given in the online documentation.

5.10. Prediction

`FitAR` produces a '`FitAR`' class object and a `predict` method is implemented for this object. This function uses `TrenchForecast` in **ltsa** (McLeod *et al.* 2007). See documentation for illustrative examples of its usage.

5.11. `Readts`: Time series input

The `Readts` function is another function which is not used directly in our package but that we have found very helpful. Many time series data are usefully stored in ASCII files with titles and comments that serve as documentation. These ASCII datasets may then be input to other software although sometimes some editing is necessary to remove the documentation, as for example, if the R function `scan` is used. Our `Readts` function is especially convenient since it can input ASCII data files containing documentation and title information. In interactive mode, `Readts` can be helpful in setting up the necessary parameters for a '`ts`' object. When `Readts` is used, a '`ts`' object is created with attribute `title`. This attribute, if present, is used to put a title on plots created by `TimeSeriesPlot`.

5.12. `SimulateGaussianAR`: Autoregressive simulation

The built-in R `arima` method for '`sim`' objects provides for simulation of AR and more general

ARIMA models with non-Gaussian innovations. It simulates by using the model equation directly with arbitrary starting values. Strictly speaking this process is not stationary but after a burn-in period, it may closely approximate a stationary process. To avoid the problem of choosing a burn-in period, which may not be adequate or which may be more than adequate and hence computationally inefficient, we can use an exact technique for Gaussian time series suggested by McLeod (1975) for the ARMA case. In the AR(p) case, we simply generate p initial time series values $z_1, \ldots, z_p$ using the appropriate multivariate normal distribution. Then $z_t, t = p+1, \ldots, n$ may be calculated directly from the model equation.

5.13. `TacvfAR` and `TacvfMA`: Theoretical autocovariance functions

The built-in function `ARMAacf` is quite complicated and uses an interface to C. Our function `TacvfAR` is based on the exact algorithm given by McLeod (1975) and is much simpler. It is very easy to translate our `TacvfAR` function into other programming environments such as MATLAB.

The `TacvfMA` is used by `GetMeanMLEAR`. `TacfMA` uses an efficient vectorized approach to compute the theoretical autocorrelations for a moving-average process. The R code is very simple and easy to understand. For an MA(q), $z_t = a_t - \theta_1 a_{t-1} - \ldots - \theta_q a_{t-q}$, the non-zero autocovariances can be written as the product of a square matrix of order $q+1$ times a column vector. In R, this matrix multiplication is most efficiently computed using `crossprod` (Venables and Ripley 2002, Section 3.9).

6. Conclusion

A complete suite of functions is described for selecting, identifying and fitting autoregressive and subset autoregressive models. Two families of subset autoregressive models are fully supported. The subset autoregressive models may provide a parsimonious alternative to the more complex ARMA models. In particular, the ARz family of models is suitable for modeling complex long time series with high-order lags. For such time series, it is difficult to select and estimate the parameters when using standard ARMA models. The ARz with the UBIC model selection criterion provides a practical and perhaps better alternative to the ARMA model for such series. Our package **FitAR** is available from the Comprehensive R Archive Network at http://CRAN.R-project.org/package=FitAR.

R scripts to generate all figures and simulations reported in Table 3 and Table 6 are available in the examples in the help for FitAR-package. Please see,

```
R> help("FitAR-package")
```

Acknowledgments

Both authors were supported by NSERC Discovery Grants. A. I. McLeod also thanks Jiahua Chen for helpful discussions about EBIC and Duncan Murdoch for advice about R. The authors would like to thank two referees for helpful comments and Achim Zeileis for numerous technical suggestions which improved the presentation of this paper.

References

Akaike H (1970). "Statistical Predictor Identification." *Annals of the Institute of Statistical Mathematics*, **22**, 203–217.

Akaike H (1978). "On the Likelihood of a Time Series Model." *The Statistician*, **27**, 217–235.

Akaike H (1979). "A Bayesian Extension of the Minimum AIC Procedure of Autoregressive Model Fitting." *Biometrika*, **66**, 237–242.

Barndorff-Nielsen O, Schou G (1973). "On the Parametrization of Autoregressive Models by Partial Autocorrelations." *Journal of Multivariate Analysis*, **3**, 408–419.

Box GEP, Cox DR (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society B*, **26**, 211–246.

Box GEP, Jenkins GM, Reinsel GC (1994). *Time Series Analysis: Forecasting and Control*. 3rd edition. Holden-Day, San Francisco.

Box GEP, Luceño A (1997). *Statistical Control by Monitoring and Feedback Adjustment*. John Wiley & Sons, New York.

Broman KW, Speed TP (2002). "A Model Selection Approach for the Identification of Quantitative Trait Loci in Experimental Crosses." *Journal of the Royal Statistical Society B*, **64**, 641–656.

Chen J, Chen Z (2008). "Extended Bayesian Information Criteria for Model Selection with Large Model Space." *Biometrika*, **95**, 759–771.

Choi B (1992). *ARMA Model Identification*. Springer-Verlag, New York.

Cleveland WS (1971). "The Inverse Autocorrelations of a Time Series and Their Applications." *Technometrics*, **14**, 277–298.

Cleveland WS (1979). "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of the American Statistical Association*, **74**, 829–836.

Cleveland WS (1993). *Visualizing Data*. Hobart Press, New Jersey.

Duong QP (1984). "On the Choice of the Order of Autoregressive Models: A Ranking and Selection Approach." *Journal of Time Series Analysis*, **5**, 145–157.

Fisher RA (1959). *Statistical Methods and Scientific Inference*. 2nd edition. Hafner, New York.

Fröberg CE (1969). *Introduction to Numerical Analysis*. Addison-Wesley, Reading.

Granger C, Jeon Y (2004). "Forecasting Performance of Information Criteria with Many Macro Series." *Journal of Applied Statistics*, **31**, 1227–1240.

Hipel KW, McLeod AI (1977). "Advances in Box-Jenkins Modelling. Part 1, Model Construction." *Water Resources Research*, **13**, 567–575.

Hipel KW, McLeod AI (1994). *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier, Amsterdam. Electronic reprint available, http://www.stats.uwo.ca/faculty/aim/1994Book/.

Hosking JRM, Ravishanker N (1993). "Approximate Simultaneous Significance Intervals for Residual Autocorrelations of Autoregressive-moving Average Time Series Models." *Journal of Time Series Analysis*, **14**, 19–26.

Jarque CM, Bera AK (1987). "A Test for Normality of Observations and Regression Residuals." *International Statistical Review*, **55**, 163–172.

Koehler AB, Murphree ES (1988). "A Comparison of the Akaike and Schwarz Criteria for Selecting Model Order." *Applied Statistics*, **37**, 187–195.

Ljung GM, Box GEP (1978). "On a Measure of Lack of Fit in Time Series Models." *Biometrika*, **65**, 297–303.

Lumley T, Miller A (2004). ***leaps**: Regression Subset Selection*. R package version 2.7, URL http://CRAN.R-project.org/package=leaps.

McLeod AI (1975). "Derivation of the Theoretical Autocorrelation Function of Autoregressive-moving Average Time Series." *Applied Statistics*, **24**, 255–256.

McLeod AI, Yu H, Krougly ZL (2007). "Linear Time Series Modeling with R Package." *Journal of Statistical Software*, **23**(5). URL http://www.jstatsoft.org/v23/i05/.

McLeod AI, Zhang Y (2006). "Partial Autocorrelation Parameterization for Subset Autoregression." *Journal of Time Series Analysis*, **27**, 599–612.

McLeod AI, Zhang Y (2008). "Faster ARMA Maximum Likelihood Estimation." *Computational Statistics and Data Analysis*, **52**, 2166–2176.

Monahan JF (1984). "A Note on Enforcing Stationarity in Autoregressive Moving Average Models." *Biometrika*, **71**, 403–404.

Newbold P, Agiakloglou C (1993). "Bias in the Sample Autocorrelations of Fractional Noise." *Biometrika*, **80**, 698–702.

Nishii R (1984). "Asymptotic Properties of Criteria for Selection of Variables in Multiple Regression." *The Annals of Statistics*, **12**, 758–765.

Percival DB, Walden AT (1993). *Spectral Analysis For Physical Applications*. Cambridge University Press, Cambridge.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Royall R (1997). *Statistical Evidence: A Likelihood Paradigm*. Chapman and Hall, New York.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Siddiqui MM (1958). "On the Inversion of the Sample Covariance Matrix in a Stationary Autoregressive Process." *Annals of Mathematical Statistics*, **29**, 585–588.

Sprott DA (2000). *Statistical Inference in Science*. Springer, New York.

Tong H (1977). "Some Comments on the Canadian Lynx Data." *Journal of the Royal Statistical Society A*, **140**, 432–436.

Tsay RS (2005). *Analysis of Financial Time Series*. 2nd edition. John Wiley & Sons, New York.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer, New York.

Wei WW (2005). *Time Series Analysis: Univariate and Multivariate Methods*. 2nd edition. Addison-Wesley.

Zeileis A, Grothendieck G (2005). "**zoo**: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software*, **14**(6), 1–27. URL http://www.jstatsoft.org/v14/i06/.

Zhang Y (2002). *Topics in Autoregression*. Ph.D. thesis, University of Western Ontario, London, Ontario, Canada.

Zhang Y, McLeod AI (2006a). "Computer Algebra Derivation of the Bias of Linear Estimators of Autoregressive Models." *Journal of Time Series Analysis*, **27**, 157–165.

Zhang Y, McLeod AI (2006b). "Fitting MA(q) Models in the Closed Invertible Region." *Statistics and Probablity Letters*, **76**, 1331–1334.

A. FitAR implementation details

The function FitAR can be used for fitting any other three types of models: AR, ARp or ARz. For some applications such as bootstrapping, it may be more convenient to work with the function GetFitAR since it only does the basic fitting. For completeness all functions used by FitAR are listed in Table 7.

The function FitAR invokes FitARz or FitARp. By default, the function FitARp invokes GetFitARpLS for least-squares estimation of the ARp. As discussed in the next section exact MLE for ARp is not reliable although the option is available.

Function	Purpose
GetFitAR	Invoked by FitAR
FitARz	Invoked by FitAR for ARz models
FitARp	Invoked by FitAR for ARp models
GetFitARz	Invoked by GetFitAR for ARz models
GetFitARpMLE	Optional function for exact MLE in ARp
GetFitARpLS	Invoked by FitAR for ARp models
GetARMeanMLE	Invoked by FitAR for exact MLE of mean
LoglikelihoodAR	Exact log-likelihood for AR model

Table 7: Estimation functions.

B. Exact MLE in ARp models

Optionally, FitARp can invoke GetFitARpMLE to attempt exact MLE estimation of the ARp. GetFitARpMLE is based on the built-in function arima which uses a penalty function approach in the subset case. But with this approach, the objective function is discontinuous outside the stationary region and this may cause errors as is illustrated in the following example. In this example, using R Version 2.6, we attempt to fit an AR$p(1, 2, 9, 12)$ to the logged lynx series but arima fails due to the penalty function approach.

```
R> z <- log(lynx)
R> p <- c(1, 2, 9, 12)
R> P <- max(p)
R> ind <- rep(0, P+1)
R> ind[p] <- NA
R> ind[P+1] <- NA
R> arima(z, order = c(P, 0, 0), fixed = ind, transform.pars = FALSE)

Error in optim(init[mask], armafn, method = "BFGS", hessian = TRUE, :
  non-finite finite-difference value [3]
In addition: Warning message:
In log(s2) : NaNs produced
```

Probably a work around using a different optimization algorithm might be found to work in this case but the general problem of using a discontinuous objective function remains and could possibly cause problems in other cases.

In conclusion, the lack of a reliable exact MLE algorithm for the ARp subset models is another reason for preferring the ARz subset models.

Affiliation:

A. I. McLeod
Department of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario N6A 5B9, Canada
E-mail: aimcleod@uwo.ca
URL: http://www.stats.uwo.ca/faculty/aim/

Y. Zhang
Department of Mathematics and Statistics
Acadia University
Wolfville, Nova Scotia B4P 2R6, Canada
E-mail: zhang@acadiau.ca
URL: http://ace.acadiau.ca/MATH/zhang/homepage.htm

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 28, Issue 2 *Submitted:* 2007-08-26
October 2008 *Accepted:* 2008-07-03