ARMA Fitting and Forecasting

6

In Chapter 5 we discussed the properties of ARMA(p,q) models, which are a broad class of models frequently used by researchers to model stationary data. We will generically use the term *ARMA model* to refer to a model which might be an AR(p), MA(q), or ARMA(p,q) model where $p > 0$ and $q > 0$. As we begin the discussion of fitting ARMA models to data, it is important to keep in mind the following points.

Key Points

1. Models are used to facilitate the understanding of underlying characteristics of times series data and assist in obtaining forecasts. The George Box quote at the beginning of Chapter 5 reminds us that we are looking for "models that are useful". ARMA models have been found to be useful for modeling stationary data.
2. Many datasets of interest do not satisfy the properties of stationarity. For example, whenever we fit a trend line or include a seasonal component in the model, we have implicitly assumed that the data are not stationary. However, as previously discussed, a final (possibly nonstationary) model will often include a stationary component that will be modeled using an ARMA model.

In this chapter we explain how to fit an ARMA(p,q) model to the data and perform analyses such as forecasting future values. Note, for example, in Example 5.6, we *assumed* a model for the sunspot data for purposes of illustration. But,

- *Where did the decision to use an AR(9) model come from?*
- *How were the parameters estimated?*
- *How are these models used to forecast future values?*

In Section 6.1 we discuss the issue of *fitting* an ARMA(p,q) model to a time series realization, and in Section 6.2 we show how to use the fitted models to forecast future values.

6.1 FITTING ARMA MODELS TO DATA

In this section we discuss the issue of fitting an ARMA(p,q) process to a given set of (apparently stationary) data. Fitting an ARM(p,q) model to a set of data involves the two steps given in the following Key Points.

DOI: 10.1201/9781003089070-6

Key Points: Fitting an ARMA(p,q) model to data involves two steps:

1. Identifying the orders p and q.
2. Estimating the parameters associated with the chosen ARMA(p,q) model.

Although we must identify p and q *before* finding the estimates of the final model, we address the two steps in the above Key Points in *reverse order*. That is, in Section 6.1.1 we consider the topic of estimating the parameters of a model for which we "know p and q". In Section 6.1.2 we discuss the identification of p and q.

Note:
The main reason for the discussion in "reverse order" is the fact that the strategies for identifying p and q involve the estimation of parameters

6.1.1 Estimating the Parameters of an ARMA(*p,q*) Model

Suppose an ARMA(p,q) model is fit to (an apparently stationary) time series realization for purposes of forecasting future values. Recall that the ARMA model has the form

$$\left(1-\phi_1 B-\cdots-\phi_p B^p\right)\left(X_t-\mu\right)=\left(1-\theta_1 B-\cdots-\theta_q B^q\right)a_t, \tag{6.1}$$

where a_t is a zero mean white noise process with finite variance σ_a^2. In this setting, to "fit" an AR$MA(p,q)$ model the following need to be estimated:

(1) p and q
(2) $\phi_1,\ldots,\phi_p$ and $\theta_1,\ldots,\theta_q$
(3) μ and σ_a^2

When we think of estimating the parameters in model (6.1) our minds immediately think of the estimation of the ϕs and θs. However, μ and σ_a^2 are also unknown parameters that must be estimated as part of a completely specified final model. As noted above, the "estimation" of p and q, the *model identification* step, is actually the first decision made in practice. We will discuss model identification in Section 6.1.2.

6.1.1.1 Maximum Likelihood Estimation of the ϕ and θ Coefficients of an ARMA Model

The basic ("gold standard") method used for estimating the parameters ϕ_1, $\phi_2,\ldots,\phi_p$ and $\theta_1,\theta_2,\ldots,\theta_q$ of an ARMA(p,q) process is *maximum likelihood*, which is widely used throughout statistical analysis. Maximum likelihood (ML) estimation involves maximizing the likelihood function

$$L=f\left(x_1,x_2\ldots,x_n\right),$$

which is the joint distribution of the time series realization. ML estimates are obtained using iterative procedures, and involve distributional assumptions concerning the white noise error, a_t. In this book we assume normal white noise with zero mean. Some problematic issues concerning ML estimation include

the facts that they can be computationally expensive and have the potential to converge to model estimates for which the roots are inside the unit circle.

Example 6.1 Estimating the parameters of the ARMA(3,1) model in Example 5.10
In Example 5.10 we considered the ARMA(3,1) model

$$\left(1-2.57B+2.50B^2-.92B^3\right)\left(X_t-30\right)=\left(1-.92B\right)a_t, \tag{6.2}$$

where $\boldsymbol{\sigma_a^2 = 1}$. The ***tswge*** code below generates the realization from this model previously examined in Example 5.10.

```
arma31=gen.arma.wge(n=150,phi=c(2.57,-2.50,.92),theta=.92,mu=30,sn=65)
```

Figures 6.1(a)–(c) are plots of the realization in R vector **arma31**, sample autocorrelations, and Parzen spectral density, shown previously in Figures 5.26(g)–(i), respectively.

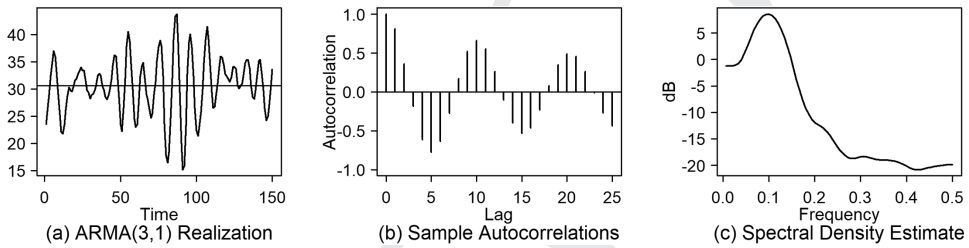

FIGURE 6.1 (a) Realization from the ARMA(3,1) model in (6.2), (b) and (c) sample autocorrelations and Parzen spectral density, respectively, of the data in (a).

In order to find the maximum likelihood estimates of the parameters of the ARMA(3,1) model fit to this realization, we use the ***tswge*** code[1]

```
est.arma.wge(arma31,p=3,q=1)
```

Among the output are the following:

```
$phi   [1] 2.4587424 -2.3304790 0.8286207
$theta [1] 0.7749636
$avar  [1] 1.035746
$xbar  [1] 30.61015
```

Thus, the final model fit by the ML estimates (to three decimal places) is

$$\left(1-2.459B+2.330B^2-.829B^3\right)\left(X_t-30.610\right)=\left(1-.775B\right)a_t, \tag{6.3}$$

with $\hat{\sigma}_a^2 = 1.036$. While the estimates of the AR and MA parameters are fairly close to the true value, they are "a little off".

Using the factor table to assess estimation performance: If the true model is known, it is natural to compare the coefficient estimates with the true values. However, we recommend that the quality of the estimates should be evaluated by comparing the associated factor tables. Table 6.1 is the factor table associated with true model (6.2) while Table 6.2 shows the factor table for the estimated model in (6.3). Comparing Tables 6.1 and 6.2 we see that both models contain a second-order AR factor associated with system frequency about

1 We note again that in this example we are presupposing that we *know* $p = 3$ and $q = 1$.

$f = .1$ with roots similarly close to the unit circle ($|r|^{-1}$ =.973 vs. 970). Both models also contain a first-order AR factor and a first-order MA factor with positive real roots in which the AR factor has a root slightly closer to the unit circle than the MA factor. Consequently, the MA factor dampens the effect of the first-order AR factor. Also, $\bar{x} = 30.610$ and $\hat{\sigma}_a^2 = 1.036$ are close to the true values of 30 and one, respectively. We address the estimation of the mean and white noise variance in the next section. Based on these observations, the basic properties of the estimated model are quite similar to those of the true model.

TABLE 6.1 Factor Table for true model in (6.2)

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.599B+.947B^2$	$.844 \pm .586i$	.973	.097
$1-.971B$		.971	.000
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.920B$	1.087	.920	.000

TABLE 6.2 Factor Table for fitted model in (6.3)

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.578B+.940B^2$	$.838 \pm .600i$	.970	.099
$1-.881B$	1.135	.881	.000
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.775B$	1.290	.775	.000

6.1.1.2 Estimating μ

As noted, after estimating the ϕ_j and θ_j coefficients, we are not finished. We still need to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}_a^2$. We estimate $\boldsymbol{\mu}$ by the sample mean $\bar{\boldsymbol{x}}$ in all of the estimation routines used in ***tswge***. In the output from the **`est.arma.wge`** function, the value for the sample mean is given in **`$xbar`**.

6.1.1.3 Estimating σ_a^2

Given a fitted model with estimates $\hat{\phi}_i, i = 1,\ldots,p$, $\hat{\theta}_i, i = 1,\ldots,q$, and $\bar{X}$, the final goal is to obtain an estimate, denoted by $\hat{\sigma}_a^2$, of the variance of the unknown white noise sequence, a_t, $t = 1,\ldots,n$. The white noise variance estimate is obtained by first estimating the a_ts which, as in regression analysis, are referred to as residuals. The residuals are denoted by $\hat{a}_t, t = 1,\ldots,n$, and $\hat{\sigma}_a^2$ is obtained as an estimate of the variance of these residuals.

Writing the fitted model in the form

$$X_t - \hat{\phi}_1 X_{t-1} - \cdots - \hat{\phi}_p X_{t-p} = \hat{a}_t - \hat{\theta}_1 \hat{a}_{t-1} - \cdots - \hat{\theta}_q \hat{a}_{t-q} + \bar{x}\left(1 - \hat{\phi}_1 - \cdots - \hat{\phi}_p\right), \tag{6.4}$$

and solving for the residuals, $\hat{a}_t$, we obtain

$$\hat{a}_t = X_t - \hat{\phi}_1 X_{t-1} - \cdots - \hat{\phi}_p X_{t-p} + \hat{\theta}_1 \hat{a}_{t-1} + \cdots + \hat{\theta}_q \hat{a}_{t-q} - \bar{x}\left(1 - \hat{\phi}_1 - \cdots - \hat{\phi}_p\right). \tag{6.5}$$

Equation 6.5 appears to provide a straightforward formula for calculating the residuals and ultimately the white noise variance estimate. However, this is not the case, and we will discuss two techniques for calculating residuals and estimating the white noise variance.

(1) Calculating Conditional Residuals
Note, for example, that

$$\hat{a}_1 = X_1 - \hat{\phi}_1 X_0 - \cdots - \hat{\phi}_p X_{1-p} + \hat{\theta}_1 \hat{a}_0 + \cdots + \hat{\theta}_q \hat{a}_{1-q} - \bar{x}\left(1 - \hat{\phi}_1 - \cdots - \hat{\phi}_p\right).$$

which cannot be calculated because $X_0, \ldots, X_{1-p}, \hat{a}_0, \ldots, \hat{a}_{1-q}$, are unknown. It follows that the first residual that can be calculated is

$$\hat{a}_{p+1} = X_p - \hat{\phi}_1 X_{p-1} - \cdots - \hat{\phi}_p X_1 + \hat{\theta}_1 \hat{a}_p + \cdots + \hat{\theta}_q \hat{a}_{p+1-q} - \bar{x}\left(1 - \hat{\phi}_1 - \cdots - \hat{\phi}_p\right).$$

Notice that even though $X_1, \ldots, X_p$ are available, the formula still involves unknown $\hat{a}_k$ values. In order to continue the calculation, we set $\hat{a}_1 = \cdots = \hat{a}_p = 0$ (their unconditional mean). Computation continues for $\hat{a}_{p+2}, \ldots, \hat{a}_n$.

For a fitted ARMA(p,q) the estimate of the white noise variance based on conditional residuals is

$$\hat{\sigma}_a^2 = \frac{1}{n-p} \sum_{t=p+1}^{n} \hat{a}_t^2. \tag{6.6}$$

Example 6.1 (Revisited)
Writing the fitted ARMA(3,1) model in (6.3) as in (6.4), we obtain

$$\begin{aligned} X_t - 2.459X_{t-1} + 2.330X_{t-2} - .829X_{t-3} &= a_t - .775a_{t-1} + 30.610\left(1 - 2.459 + 2.330 - .829\right) \\ &= \hat{a}_t - .775\hat{a}_{t-1} + 30.610\left(.042\right) \\ &= \hat{a}_t - .775\hat{a}_{t-1} + 1.286. \end{aligned} \tag{6.7}$$

In this example, Equation 6.5 for calculating residuals becomes

$$\hat{a}_t = X_t - 2.459X_{t-1} + 2.330X_{t-2} - .829X_{t-3} + .775\hat{a}_{t-1} - 1.286. \tag{6.8}$$

Note, for example

$$\hat{a}_1 = X_1 - 2.459X_0 + 2.330X_{-1} - .829X_{-2} + .775\hat{a}_0 - 1.286,$$

which cannot be calculated because $X_0, X_{-1}X_{-2}$,and $\hat{a}_0$ are unknown.

In order to avoid data values X_t with zero or negative subscripts, the calculation begins with $\hat{a}_4$. The first five values of the realization in Figure 6.1(a) are $X_1 = 23.522,\ X_2 = 26.698, X_3 = 28.877, X_4 = 32.561,$ and $X_5 = 35.089$.The calculation for $\hat{a}_4$ is

$$\hat{a}_4 = X_4 - 2.459X_3 + 2.330X_2 - .829X_1 + .775\hat{a}_3 - 1.286.$$

As noted, even though $X_1, \ldots, X_4$ are available, the formula still involves the unknown $\hat{a}_3$, so we continue calculation by setting $\hat{a}_3 = 0$. Computation proceeds as follows:

$$\hat{a}_4 = 32.561 - 2.459(28.877) + 2.330(26.698) - .829(23.522) + .775(0) - 1.286 = 2{,}973,$$
$$\hat{a}_5 = X_5 - 2.459X_4 + 2.330X_3 - .829X_2 + .775\hat{a}_4 - 1.286,$$
$$= 35.089 - 2.459(32.561) + 2.330(28.877) - .829(26.698) + .775(2.973) - 1.286$$
$$= 1.190$$

etc.

Note that the conditional estimate of the white noise variance is

$$\hat{\sigma}_a^2 = \frac{1}{n-3}\sum_{t=4}^{150}\hat{a}_t^2$$
$$= 1.095. \tag{6.9}$$

Key Points: Comments about conditional residuals:

1. Only $n - p$ residuals can be calculated.
2. The residuals $\hat{a}_p,\ldots,\hat{a}_{p+1-q}$ may be poor estimates of the true values, $a_p,\ldots,a_{p+1-q}$. The residuals $\hat{a}_{p+1},\ldots,\hat{a}_n$ are called *conditional residual estimates* because of their dependence on the starting values, $\hat{a}_p = \cdots = \hat{a}_{p+1-q} = 0$.

Because of the issues with the conditional residuals, we recommend an alternative procedure for calculating residuals that involves *backcasting*.

(2) Calculating Residuals using Backcasting

Recall that a fitted ARMA(p,q) model, such as that in (6.4), is based on an assumption that the current value of the random variable X_t is a linear combination of past values plus a linear combination of present and past noise components. In Section 6.2 we will discuss methods for forecasting future values based on a model of the form (6.4) fit to a dataset. An unusual "backward" model using the same parameter estimates is given in (6.10)

$$X_t - \hat{\phi}_1 X_{t+1} - \cdots - \hat{\phi}_p X_{t+p} = \hat{a}_t - \hat{\theta}_1\hat{a}_{t+1} - \cdots - \hat{\theta}_q\hat{a}_{t+q} + \bar{X}\left(1 - \hat{\phi}_1 - \cdots - \hat{\phi}_p\right), \tag{6.10}$$

This model has the same parameter estimates as (6.4), but the backward model specifies that the random variable, X_t, is a linear combination of *future* values plus a linear combination of *present and future* noise components. That is, this model literally "goes backward". Given a realization $x_1, x_2,\ldots,x_n$,the natural "thing to do" is to find forecasts $\hat{x}_{n+1},\hat{x}_{n+2},\ldots$and so forth as was done in Chapter 2 with exponential smoothing and Holt-Winters forecasting, and which we will do with model-based forecasting procedures throughout the remainder of this book. However, given the same realization, the backward model in (6.10) can be used to "forecast backward", that is backcast $\hat{x}_0,\hat{x}_{-1},\ldots$ and so forth.

Having fit an ARMA(p,q) model to the data, a backcast procedure for obtaining backcast residual estimates and subsequently an estimate of the white noise variance is as follows:

1. Given the data, backcast to obtain $\hat{x}_0,\hat{x}_{-1},\ldots,\hat{x}_{-K}$ (where $K = 49$ in ***tswge***).
2. Use the forward model and (6.5) to begin estimating residuals, $\hat{a}_{-49+p+1},\hat{a}_{-49+p+2},\ldots,\hat{a}_0,\hat{a}_1,\hat{a}_2,\ldots,\hat{a}_n$. Note that the starting values are set to zero as before, that is, $\hat{a}_{-49} = \hat{a}_{-49+1} = \hat{a}_{-49+p} = 0$, but the effect of the starting values has diminished by the time $t = 1$.

3. Retain $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_n$ (this is a full set of n residuals that are not very dependent on the starting values).
4. Compute $\hat{\sigma}_a^2 = \frac{1}{n}\sum_{t=1}^{n}\hat{a}_t^2$ based on the new set of n residuals.

Example 6.1 (Backcast Residuals)
The backcast procedure is computationally intensive. Estimation procedures implemented in ***tswge*** use a backcasting procedure for obtaining residuals and the estimated white noise variance. In the case of the ARMA$(3,1)$ data in Example 6.1, the backcast-based estimate of the white noise variance is $\hat{\sigma}_a^2 = 1.0356$. Recall that **`$avar 1.035746`** was one of the output values shown earlier from the command

```
est.arma.wge(arma31,p=3,q=1)
```

Key Fact: The ***tswge*** routines calculate the residuals using *backcasting*.

6.1.1.4 Alternative estimates for AR(p) models

While ML estimates are the "gold standard" for all ARMA models, the ML estimates involve (sometimes lengthy) iterative procedures. Two non-iterative estimation techniques, *Yule-Walker(YW)* and *Burg*, have been used historically to estimate the parameters of an AR(p) model, at least in part because of the computing time sometimes required for the ML estimates to converge. Although increased computing power and more efficient ML routines have somewhat alleviated many of the computation issues, we believe it is useful to familiarize readers with these alternatives for AR models. The ***tswge*** function **`est.ar.wge`** optionally calculates the ML, YW, or Burg estimates for AR(p) models.

(1) Yule-Walker Estimation for an AR Model
As the name suggests, the Yule-Walker estimates are based on the Yule-Walker equations in (5.29). By replacing the true autocorrelations in these equations with sample autocorrelations and solving the resulting $p \times p$ system

$$\begin{aligned}
\hat{\rho}_1 &= \phi_1 + \phi_1\hat{\rho}_1 + \cdots + \phi_1\hat{\rho}_{p-1} \\
\hat{\rho}_2 &= \phi_1\hat{\rho}_1 + \phi_2 + \cdots + \phi_1\hat{\rho}_{p-2} \\
&\vdots \\
\hat{\rho}_p &= \phi_1\hat{\rho}_{p-1} + \phi_1\hat{\rho}_{p-2} \cdots + \phi_1
\end{aligned} \tag{6.11}$$

for the unknown $\phi_1, \phi_2, \ldots, \phi_p$, the solutions are called the *Yule-Walker estimates*. An appealing feature of the YW estimates is that they *always* result in a *stationary* model.

QR 6.1
Yule-Walker
Estimation

(2) Burg Estimation for an AR Model
Burg estimates make use of the interesting fact that there is no preference in time direction in AR models (recall $\rho_k = \rho_{-k}$) and consequently that the forces "driving the process forward" also drive it backward.

For example, consider the realization in Figure 6.2(a) which was previously shown in Figure 5.19(d) and referred to as Realization A_real, which is from the model

$$X_t - 1.71X_{t-1} + 1.22X_{t-2} - .475X_{t-3} = a_t, \tag{6.12}$$

where $\sigma_a^2 = 1$. Figure 6.2(b) is a plot of the realization in Figure 6.2(a) in "reversed time".

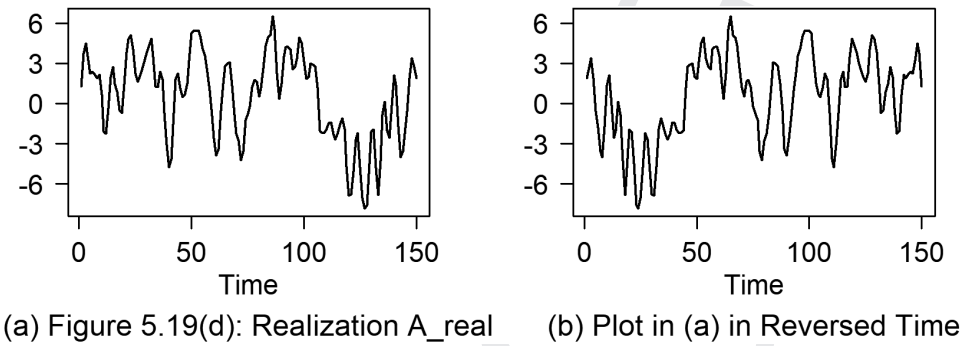

FIGURE 6.2 (a) Realization 1 in Figure 5.19(d) and (b) realization in (a) in reversed time.

The following commands generate the plots in Figure 6.2.

```
par(mfrow=c(1,2))
a=gen.arma.wge(n=150,phi=c(1.71,-1.22,.475),sn=327,plot=FALSE)
b=rev(a)
plotts.wge(a)
plotts.wge(b)
```

In Figures 6.2(a) and (b) we see that "flipping" the time (horizontal) axis does not change the general appearance or apparent correlation structure of the realizations. The data in Figure 6.2(b) can be considered to be a realization from the "backward model" (notice the subscripts)

$$X_t - 1.71X_{t+1} + 1.22X_{t+2} - .475X_{t+3} = \delta_t, \tag{6.13}$$

where δ_t is white noise with zero mean and the same white noise variance, σ_a^2.[2] Because each of these datasets are realizations from models with the same ϕ_k values, Burg (1975) showed how to use this "additional" information to obtain improved estimates of the ϕ_ks. For more information, see Woodward et al. (2017).

QR 6.2 Least Squares Estimation

QR 6.3 Burg Estimation

2 Readers will recognize (6.13) as another example of a "backward" model analogous to the one shown in (6.10) in the context of backcasting.

Key Point: The *tswge* command **est.ar.wge** fits an AR(p) model to a set of time series data.
1. The order p is specified by the user.
2. **est.ar.wge** produces the following estimates:
   (a) ML (default) **(type=ML)** Also, **ML** is the default option.
   (b) Yule-Walker **(type=YW)**
   (c) Burg **(type=burg)**

Example 6.2 Estimating the coefficients of the AR(3) model A_real discussed in Section 5.1.3.5
Consider again the realization in Figure 6.2(a) (and Figure 5.19(d)) from the model

$$\left(1-1.71B+1.22B^2-.475B^3\right)X_t = a_t, \tag{6.14}$$

with $\sigma_a^2 = 1$,which was referred to as Model A_real. The factored form of the model is

$$\left(1-.95B\right)\left(1-.76B+.5B^2\right)X_t = a_t.$$

This model was used to show that the positive real root associated with the factor $\left(1-.95B\right)$ dominates the behavior of the process because the real root is substantially closer to the unit circle than the complex conjugate roots associated with the factor $\left(1-.76B+.5B^2\right)$ and system frequency $f_0 = .16$. The following code re-generates the data in Figure 5.19(d) and obtains the ML estimates for an AR(3) fit to the data.

```
A_real=gen.arma.wge(n=150,phi=c(1.71,-1.22,.475),sn=327)
A_real.ml=est.ar.wge(A_real,p=3,type='mle')
# The command A_real.ml=est.ar.wge(A_real,p=3) produces the same results
# since 'mle' is the default type.
```

Among the output we obtain

```
A_real.ml$phi 1.7614 -1.4010 .5605
A_real.ml$avar [1] 0.9093326
```

Consequently, the AR(3) model based on the ML estimates of the ϕs is given by

$$\left(1-1.761B+1.401B^2-.561B^3\right)X_t = a_t, \tag{6.15}$$

with $\hat{\sigma}_a^2 = .909$, which is quite close to the "true" model in (6.14). Tables 6.3 and 6.4 show factor tables for the true AR(3) model and for the fitted model using ML estimates, respectively. We see that the factors of the fitted model are very similar to those of the true model. That is, there is a 1st-order factor associated with a positive real root and a 2nd-order factor associated with system frequency .16. In both cases the positive real root is closer to the unit circle than are the complex conjugate roots associated with the 2nd-order factor.

TABLE 6.3 Factor Table for true model in (6.2)

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.95B$	1.05	.95	.00
$1-.76B+.50B^2$	$.76\pm1.20i$	.71	.16

TABLE 6.4 Factor Table for ML fit to Model A_real

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.90B$	1.12	.90	.00
$1-.87B+.63B^2$	$.69\pm1.06i$	.79	.16

Yule-Walker and Burg Estimates of AR(3) fits to Realization A_real.

The Yule-Walker and Burg estimates can be obtained using the commands:

```
A_real.yw=est.ar.wge(A_real,p=3,method='yw')
A_real.burg=est.ar.wge(A_real,p=3,method='burg')
```

The MLE, YW, and Burg estimates of ϕ_1, ϕ_2,and ϕ_3 are shown in Table 6.5 to three decimal places. It can be seen that all three estimation techniques give estimates close to the true values.

TABLE 6.5 ML, YW, and Burg estimates for AR(3) models fit to the realization in Figure 4.19(d)

	ϕ_1	ϕ_2	ϕ_3	σ_a^2
True	1.710	–1.220	.475	1.000
MLE	1.761	–1.401	.561	0.909
YW	1.757	–1.389	.554	0.910
Burg	1.767	–1.407	.564	0.909

At this point, you may be asking the following question.

Question: Since we can calculate the "gold standard" ML estimates for AR models, why do we bother discussing the other two methods?

One of the answers to the above question is that the YW and Burg estimates are used in practice. However, that's not a compelling explanation. It turns out that while the AR(3) models fit to the **A_real** dataset in Example 6.2 are quite similar for each of the estimation types, the methods have advantages and disadvantages as we will see in the following.

Example 6.3 Problems with Yule-Walker Estimates

Consider the AR(4) model

$$\left(1-2.76B+3.76B^2-2.6B^3+.89B^4\right)X_t=a_t, \tag{6.16}$$

where $\sigma_a^2=1$. The following command generates a realization from the model in (6.16) and plots the realizations, sample autocorrelations, and Parzen spectral density estimate. Figure 6.3 shows the resulting plots. The following code will produce the plots in Figure 6.3.

```
ar4=gen.arma.wge(n=100,phi=c(2.76,-3.76,2.6,-.89),sn=468)
plotts.sample.wge(ar4)
```

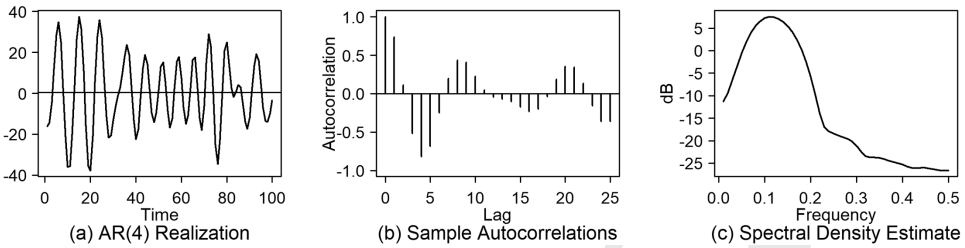

FIGURE 6.3 (a) Realization from model (6.16), (b) sample autocorrelations, and (c) Parzen spectral density estimate of the data in (a).

Although the data are based on an AR(4) model, the plots have the behavior of an AR(2) with system frequency around $f = .10$.Assuming an AR(4) model, we used the following commands to estimate the parameters.

```
ar4.ml=est.ar.wge(ar4,p=4)
ar4.yw=est.ar.wge(ar4,p=4,type= 'yw')
ar4.bg=est.ar.wge(ar4,p=4,type= 'burg')
```

Table 6.6(a) shows the true and estimated model parameters for the AR(4) model in (6.16). We see from the table that the ML and Burg estimates appear to be similar and quite good. However, the Yule-Walker estimates are surprisingly poor. To better understand the situation, and in particular, to answer the question, "What happened to the Yule-Walker estimates?" we examine (you guessed it) the *factor tables.* Table 6.6(b) shows factor tables for the true model and the three estimated models.

TABLE 6.6 Fitting an AR(4) Model to Realization from (6.16)

(a) True model parameters and ML, YW, and Burg parameter estimates for AR(4) model fits to the realization in Figure 6.3

	ϕ_1	ϕ_2	ϕ_3	σ_a^2
True	2.76	–3.76	2.60	–0.89
MLE	2.77	–3.78	2.63	–0.90
YW	1.90	–1.76	.63	–0.12
Burg	2.78	–3.81	2.65	–0.91

(b) Factor tables for model (6.16) along with ML, YW, and Burg AR(4) fits.

FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
(a) True parameter values			
$1-1.56B+.95B^2$	$.82 \pm .62i$	0.977	0.103
$1-1.20B+.93B^2$	$.65 \pm .81i$	0.966	0.143
(b) ML Estimates			
$1-1.58B+.96B^2$	$.83 \pm .60i$	0.978	0.100
$1-1.19B+.94B^2$	$.63 \pm .81i$	0.971	0.145
(c) Yule-Walker Estimates			
$1-1.38B+.91B^2$	$.76 \pm .72i$	0.954	0.121
$1-0.52B+.14B^2$	$1.90 \pm .194i$	0.368	0.126
(d) Burg Estimates			
$1-1.58B+.96B^2$	$.83 \pm .60i$	0.977	0.100
$1-1.21B+.95B^2$	$.63 \pm .81i$	0.976	0.144

In the factor table for the original model in (6.16) there are two sets of complex conjugate roots, one associated with $f_0 = .103$ and the other with $f_0 = .143$. Points to note about this model are the following:

(a) The two system frequencies are quite close together ($f_0 = .103$ and $f_0 = .143$).
(b) Both pairs of complex conjugate roots are near the unit circle.
(c) The root associated with $f_0 = .103$ is slightly closer to the unit circle than is the root associated with $f_0 = .143$.

The factor tables associated with the ML and Burg estimates do an excellent job of showing the two frequency components present in the factor table for the true model. However, the YW factor table is "another story". We see that the first factor, $1 - 1.38B + .91B^2$, is "in the ballpark" since the factor is associated with a system frequency of $f_0 = .121$ and the roots are fairly close to the unit circle although not as close as those in the original model, as well as the ML and Burg estimated models. The second factor is associated with essentially the same system frequency $f_0 = .126$, and the roots are quite far from the unit circle. In essence the YW estimates could not "separate" the two system frequencies in the model. Note also that the plots in Figure 6.3 do not separate the two system frequencies because of their close proximity to each other. The *only places* in which we saw the two separate system frequencies were in the factor tables of the true model, and the ML and Burg estimated models. If it important to detect that there are two system frequencies "close to each other" then certainly the ML and Burg fits are superior to the YW (and to information in the plots in Figure 6.3).

In this example the ML and Burg estimates performed better than the YW estimates. The following summarizes the properties of the three methods.

PROPERTIES OF THE THREE ESTIMATION TECHNIQUES

(1) *ML Estimates:*
  (a) Are generally quite good.
  (b) Can be computationally intensive. While this is not usually a serious problem with today's computer speeds, simulations or applications that require "real time" parameter estimates can be slowed down considerably when using ML estimates.
  (c) ML estimates can produce models with roots inside the unit circle (beware!)
(2) *Yule-Walker estimates*:
  (a) Are very intuitive and were historically an early technique used for estimating the parameters of an AR process.
  (b) Can be calculated rapidly.
  (c) Always produce a stationary model.
  (d) Are sometimes still used in practice (which is generally a bad idea). It is important to be aware of their shortcomings.
(3) *Burg Estimates:*
  (a) Are usually very similar to ML estimates.
  (b) Always produce a stationary model.
  (c) Can be computed rapidly.
  (d) Because of (a), (b), and (c) they are a good choice when
    - ML estimates are associated with roots inside the unit circle
    - estimating parameters in real time or in extensive simulations
    - analyzing very large daasets.

Key Points

1. We would not have understood the underlying issues with the YW estimates of the AR(4) data in Example 6.3 if we had not *factored* the models.
2. Yule-Walker and Burg estimates *are not applicable to ARMA*(p,q) *models where* $q > 0$
3. Commands **`est.arma.wge(x,p=4,q=0)`** and **`est.ar.wge`**`(p=4)` give the same (MLE) results.
4. Yule-Walker estimates can be very poor when roots are near the unit circle, or when there are repeated roots. We recommend ***against*** use of Yule-Walker estimates.

6.1.2 ARMA Model Identification

In Section 6.1.1 we discussed the issue of estimating the parameters of an ARMA(*p,q*) model fit to a set of stationary data where ***p*** and ***q*** are known. We now address the more general issue of determining *which* ARMA model best fits the data. That is, we need to "identify" the order *p* before we finalize the estimation of the parameters in the final model. The final models in (6.3) and (6.15) are based on ARMA(3,1) and AR(3) fits, respectively, because we "know" the realizations were generated from ARMA(3,1) and AR(3) models.

Key Point: It is unrealistic to assume that we *know* the orders of an ARMA model that should be fit to a real time series dataset.

6.1.2.1 Plotting the Data and Checking for White Noise

Any time we fit a model to a set of data we should first plot the data and sample autocorrelations. One thing to check in these plots is whether the data are simply white noise or whether they show obvious nonstationary structure. A quick test for white noise is based on the following facts regarding sample autocorrelations of white noise data:[3]

$$E\left[\hat{\rho}_k\right] \approx 0$$

- for $k \neq 0$
- the standard error of $\hat{\rho}_k$ is approximately $1/\sqrt{n}, k \neq 0$ so for moderately large n the $\hat{\rho}_k's$ should be small
- $\hat{\rho}_{k_1}$ and $\hat{\rho}_{k_2}$ are essentially uncorrelated when $k_1 \neq k_2$
- the $\hat{\rho}_k's$ are approximately normal

Based on the above facts, we reject $H_0 : \rho_k = 0$ vs. $H_a : \rho_k \neq 0$, at the 5% level of significance if $|\hat{\rho}_k| > 1.96\left(1/\sqrt{n}\right)$. It is common to accompany plots of sample autocorrelations that might be from white noise with (95%) limit lines at $\pm 2/\sqrt{n}$.[4] Figure 6.4(a) shows a realization of length $n = 150$ from a white noise process generated by the ***tswge*** command

3 See Bartlett (1946).

4 Although the 97.5th percentile of the normal distribution is 1.96, it is common practice to round 1.96 to 2 and state the limits as $\pm 2/\sqrt{n}$.

```
x=gen.arma.wge(n=150,phi=0,theta=0,sn=147)
```

Figure 6.4(b) shows the sample autocorrelations. Figure 6.4(a) is consistent with white noise in that there are no cyclic patterns, no wandering, and the data simply seem to be random with no correlation with other values. The sample autocorrelations in Figure 6.4(b) appear to be small (except for $\hat{\rho}_0 = 1$). Figure 6.4(c) shows the same sample autocorrelations as in Figure 6.4(b) but includes the 95% limit lines $\pm 2 / \sqrt{150} = \pm .16$. The sample autocorrelations stay within the limit lines so there is no reason to reject white noise.

Key Point: The 5% applies separately for each k, so it will not be unusual for about 5% of the sample autocorrelations to be outside the 95% limit lines even if the data are white noise.

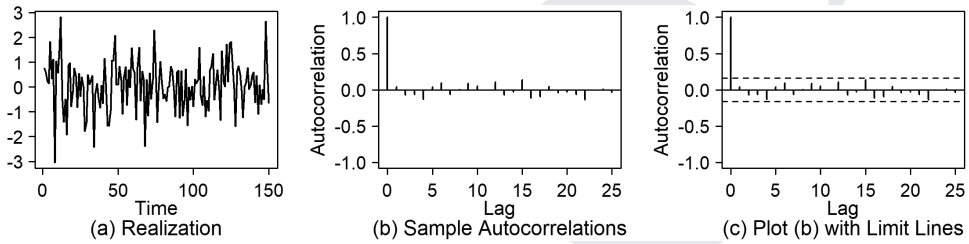

FIGURE 6.4 (a) White noise realization, (b) sample autocorrelations for the data in (a), and (c) sample autocorrelations showing the 95% limit lines.

We next take a look at Figure 6.5(b) and (c) and see that the sample autocorrelations for Figure 6.5(a) have the appearance of sample autocorrelations of white noise data. However, to decide on the basis of the sample autocorrelations alone that the corresponding data (in Figure 6.5(a)) are white noise would clearly be a *major mistake*. The time series in Figure 6.5(a) appears to be higher frequency at the beginning of the realization than it is toward the end. Techniques for analyzing data with time-varying frequencies are discussed in Chapter 13 of Woodward, et al. (2017).

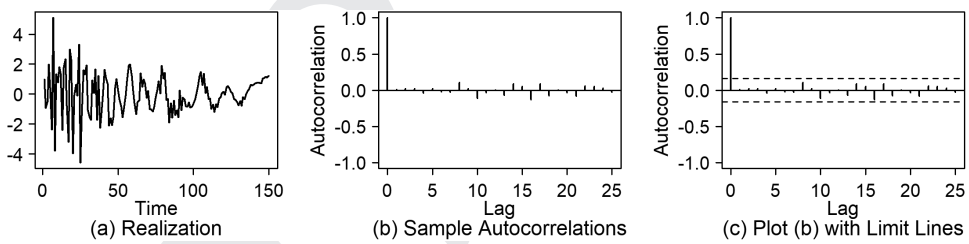

FIGURE 6.5 (a) A realization which is clearly not white noise, (b) sample autocorrelations for the data in (a), and (c) sample autocorrelations showing the 5% limit lines.

Key Point: Always plot the data!

6.1.2.2 Model Identification Types

There are two major strategies commonly used for identifying the order of an ARMA(p,q):

(1) ***AIC-type Measures:*** Akaike's Information Criterion (AIC) and its variations such as BIC and AICC are popular in a number of statistical applications for deciding among competing models

for a set of data. In the ARMA setting, AIC-type measures are used to "identify" p and q for purposes of fitting an ARMA model to a time series realization. These AIC-type techniques involve fitting ARMA models for a range of values for p and q, and selecting the optimal p and q based on the criterion being used.

(2) ***Pattern Recognition Methods****:* In their ground-breaking 1970 book, G.E.P. Box and G.M. Jenkins used plots of partial autocorrelations to identify the order of an AR process. We will define the partial autocorrelation and discuss its role in identifying the order of an AR model in Section 6.1.2.4.

Key Points

1. In this book we focus on AIC-type model identification which is introduced in Section 6.1.2.3.
2. AIC-type measures are easily automated.
3. Partial autocorrelation methods, to be discussed in Section 6.1.2.4, involve a decision regarding the order of an AR model that is based on observing graphical patterns of partial autocorrelations.
4. AIC-type procedures extend easily to ARMA model identification, while pattern recognition methods are difficult to use for ARMA modeling.

6.1.2.3 AIC-type Measures for ARMA Model Fitting

AIC is a general criterion for statistical model identification that you have probably seen before in other settings. For example, it is often used for selecting the independent variables to be used in a multiple regression model. It is also applicable for identifying the order p and q of an ARMA(p,q) model. AIC is actually one of a number of information-based criteria for model selection. Others include BIC, AICC, among others. In the following, the term "AIC" will often be used generically to represent the information-based techniques.

(1) Multiple Regression Review

Consider a hypothetical model selection problem in multiple regression in which dependent variable y is to be predicted using either the single variable x_1 or the two independent variables, x_1 and x_2. In multiple regression, the goal is to reduce the unexplained residual variability, that is, to reduce the mean square error (MSE). Suppose $\hat{y} = 6 + 3x_1$ is the best prediction equation using only x_1, and it has an unexplained variability MSE_1.If independent variable x_2 is added to the model, then there are a multitude of choices for the coefficients b_0, b_1, and b_2 in the prediction equation $\hat{y} = b_0 + b_1x_1 + b_2x_2$. Suppose the best prediction equation has unexplained variability MSE_2. Note that one of the choices of b_0, b_1, and b_2 results in the model $\hat{y} = 6 + 3x_1 + 0x_2$ (which has unexplained variability MSE_1). That is, the unexplained variability, MSE_2,of the best prediction equation using the two independent variables, must be at most MSE_1. The key point is that adding a variable to an existing equation will nearly always reduce (and will never increase) the unexplained residual variability. Consequently, the fact that the unexplained residual variability is reduced by adding a new variable is not sufficient evidence to conclude that the new variable is important to the prediction. The goal in multiple regression is to reduce the unexplained residual variability but hopefully to only use the independent variables that are *actually useful.* Clearly, the most useful model will be one with a relatively small MSE. However, because the inclusion of additional variables will never increase (and will nearly always decrease) the MSE, using the MSE for model selection will always yield the model with the most variables (whether they are truly related to the response or not). Techniques are available, for example stepwise, backward elimination, AIC, etc., for choosing which independent variables are useful. For a discussion of multiple regression, see Moore, McCabe, and Craig (2021). In time series analysis it is common to use AIC, which protects against always selecting a higher order model by adding a "penalty" (see (6.17) for each new variable added.

ARMA model identification is analogous to model selection in multiple regression. Increasing p and/or q in the ARMA setting indicates the addition of variables (and thus parameters) to the model. When fitting an ARMA(p,q) model to a set of data, a goal is to reduce the unexplained variability, as measured by $\hat{\sigma}_a^2$. Suppose, for example, we fit an ARMA(1,1) model, say $X_t - .8X_{t-1} = a_t + .5a_{t-1}$, to a realization, and associated with this fit is an estimate of the white noise variance, $\hat{\sigma}_a^2$. Then if we fit an ARMA(2,1) model to the data, one choice for the ARMA(2,1) model is $X_t - .8X_{t-1} - 0X_{t-2} = a_t + .5a_{t-1}$. Consequently, when an ARMA(2,1) model is fit to the realization, the estimated white noise variance must be at most $\hat{\sigma}_a^2$ from the ARMA(1,1) fit.

Key Point: What is needed is a strategy for increasing the order p of the AR model *only if the increase in order is sufficiently helpful.*

As mentioned above, AIC imposes a penalty for adding terms to the model. Theoretically, AIC involves the maximized likelihood. We will use an approximation to AIC that is easy to implement and makes intuitive sense. We define AIC in (6.18) and the strategy is to select the orders p and q such that

$$AIC = \ln(\hat{\sigma}_a^2) + 2(p+q+1)/n \tag{6.17}$$

is minimized. AIC is implemented by selecting the maximum orders we want to allow for p and q, call them P and Q, respectively. Then we fit all models with $0 \le p \le P$ and $0 \le q \le Q$, and choose the order combinations for which AIC is minimized. For simplicity, let $P = 2$ and $Q = 1$ in which case we would fit the models ARMA(0,0), ARMA(0,1), ARMA(1,0), ARMA(1,1), ARMA(2,0), and ARMA(2,1), and select the p and q for which AIC is minimized.[5]

Key Point: The AIC procedure for ARMA model identification is basically to reduce the natural log of the estimated white noise variance subject to a penalty for increasing the number of parameters.

BIC and AICC are variations of AIC that are based on modifications of the penalty. Specifically, we define BIC and AICC as

$$BIC = \ln(\hat{\sigma}_a^2) + (p+q+1)\frac{\ln(n)}{n} \tag{6.18}$$

and

$$AICC = \ln(\hat{\sigma}_a^2) + (n+p+q+1)/(n-p-q-3). \tag{6.19}$$

AIC tends to select larger orders for larger realization lengths and thus tends to select orders p and q that are too large, for long realizations. BIC adjusts for this tendency by imposing a stronger penalty. AICC tends to serve as a compromise between AIC and BIC.

5 Note that the ARMA(0,0) model is white noise, ARMA(0,1) is an MA(1), ARMA(1,0) is an AR(1), and ARMA(2,0) is an AR(2).

Key Points

1. AIC and its variations are designed to select a *stationary* model. We will discuss nonstationary models in Chapters 7 and 8.
2. When AIC and its alternatives search for $0 \le p \le P$ and $0 \le q \le Q$, one of the options is $p = q = 0$, that is, white noise. Thus, AIC-type methods provide another check for white noise.
3. AIC, BIC, and AICC may produce different "competing" models. The wise analyst does not simply select the order that, for example, AIC selects, but instead selects from among the reasonable models.
4. We will introduce additional strategies and methods for choosing from among these "competing models", not the least of which is *domain knowledge*.

*(2) AIC-type Methods for ARMA Model Identification in **tswge***

The basic ***tswge*** functions for AIC-type ARMA model identification are **`aic.wge`** and **`aic5.wge`.** Given a realization in vector **`x`**, then to fit an ARMA(p,q) model to this realization where p is allowed to range between 0 and 10, and q ranges from 0 to 4, we use the command

```
aic.wge(x,p=0:10,q=0:4,type= 'aic').
```

To restrict our choice to AR models, we could use

```
aic.wge(x,p=0:10,q=0:0,type= 'aic')
```

Note that **`type= 'aic'`** is the default and could have been omitted. Other values for type are **`type= 'bic'`** and **`type='aicc'.`**

QR 6.4 ARMA Model Identification

Key Point: ***tswge*** function, **`aic5.wge`**, is very useful for identifying alternative ARMA model choices. While the command **`aic.wge`** selects a single model (based on the criterion selected), **`aic5.wge`** shows the top five models as selected by the chosen selection criterion.

Example 6.4 Using AIC-type Methods to Select the Order of the Data in `arma31` Which was Generated from the ARMA(3,1) Model in (6.2)

Recall that Figure 6.1(a) shows a realization of length $n = 150$ from the ARMA(3,1) model in (6.2)

$$\left(1 - 2.57B + 2.50B^2 - .92B^3\right)X_t = \left(1 - .92B\right)a_t,$$

with $\sigma_a^2 = 1$.

The following code generates the realization in Figure 6.1(a).

```
arma31=gen.arma.wge(n=150,phi=c(2.57,-2.50,.92),theta=.92,mu=30,sn=65)
```

Clearly from the data and sample autocorrelations shown in Figures 6.1(a) and (b) respectively, it can be seen that the data in Figure 6.1(a) are not from a white noise model. Assuming we do not know the order of the model that generated the data, we use AIC to select the orders p and q from among the options, $p = 0,1,\ldots,12$ and $q = 0,\ldots,4$ using the ***tswge*** command

```
a=aic.wge(arma31,p=0:12, q=0:4).
```

Among the output are

```
a$p 3
a$q 1
```

which indicates that AIC has correctly chosen the ARMA(3,1) model.[6] Also included in the output are the parameter estimates

```
a$phi 2.4587424 -2.3304790 0.8286207
a$theta 0.7749636
a$xbar 30.61015
a$vara 1.035746
```

which were previously given in Example 6.1 as output from **`est.arma.wge`**. By default the criterion used in the above **`aic.wge`** command was AIC. To identify the orders using BIC and AICC we use the commands

```
b=aic.wge(arma31,p=0:12,q=0:4,type= 'bic')
cc=aic.wge(arma31,p=0:12,q=0:4,type= 'aicc')
```

Among the output from these commands are

```
b$p 2
b$q 0
cc$p 3
cc$q 1
```

Thus, AICC also selects an ARMA(3,1) model while BIC chooses an AR(2).

Key Point: While **`aic.wge`** selects the "best" model associated with the chosen criterion, we recommend using the function **`aic5.wge`** which outputs the top five choices for a given criterion.

For example, the following commands were executed, and the output is shown in Table 6.7.

```
aic5.wge(arma31,p=0:12,q=0:4)
aic5.wge(arma31,p=0:12,q=0:4,type= 'aicc')
aic5.wge(arma31,p=0:12,q=0:4,type= 'bic')
```

6 Notice the delay when issuing the AIC command with $p = 0:12$ and $q = 0:4$. In this case, **`est.arma.wge`** had to find ML estimates of 65 models. In earlier days this might have taken several hours (or overnight) to compute. Recall that Yule-Walker and Burg estimates are only designed to estimate the parameters of an AR(p) model, so because the range of values includes cases in which $q \neq 0$, we cannot use YW or Burg estimates to "speed things up". As mentioned previously, analysts sometimes choose to simplify things by finding the "best" AR model. Choosing the best AR model is the topic of Section 6.1.2.4.

TABLE 6.7 Output Showing the Top 5 AIC, AICC, and BIC Choices for the `arma31` Dataset

p	*q*	*AIC*	*p*	*q*	*AICC*	*p*	*q*	*BIC*
3	1	0.1017890	3	1	1.119038	2	0	0.1789293
4	0	0.1056942	4	0	1.122944	3	0	0.1899721
5	1	0.1059744	3	0	1.125800	2	1	0.1948918
2	2	0.1095763	5	1	1.126116	3	1	0.2021435
3	0	0.1096885	2	2	1.126826	4	0	0.2060487

This table shows, as was already noted, that AIC and AICC selected an ARMA(3,1) as the top choice, and BIC selected an AR(2). The table also shows that an ARMA(3,1) was the fourth choice for BIC. Also, an AR(2) model was not one of the top five choices for either AIC or AICC. At this point an obvious question of interest is how the ARMA(3,1) and AR(2) fits compare. The models under consideration are

(a) ARMA(3,1) (This fitted model is already shown in (6.3) and will be repeated here.)

$$\left(1-2.459B+2.330B^2-.829B^3\right)\left(X_t-30.610\right)=\left(1-.775B\right)a_t,$$

with $\hat{\sigma}_a^2=1.036$.

(b) AR(2)

$$\left(1-1.591B+.938B^2\right)\left(X_t-30.610\right)=a_t,$$

where $\hat{\sigma}_a^2=1.082$.

Comparing the coefficients does not provide information about the characteristics of the two models. Consequently, we examine the factor tables. Table 6.8 shows the factor table for the ARMA(3,1) fitted model (already shown in Table 6.2).

TABLE 6.8 Factor Tables for the ARMA(3,1) and AR(2) Fits to the `arma31` Data

(a) Factor Table for ARMA(3,1) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.578B+.940B^2$	$.838\pm.600i$	.970	.099
$1-.881B$	1.135	.881	.000
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.775B$	1.290	.775	.000

(b) Factor Table for AR(2) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.591B+.938B^2$	$.848\pm.589i$	.968	.097

The factor tables show that the AR(2) model has essentially the same 2nd-order factor as the ARMA(3,1) model. Recalling that in the discussion preceding Table 6.1 we noted that the 1st-order AR and MA factors almost canceled each other, we conclude that the AR(2) fit to the data is a logical choice for BIC.

Example 6.5 Using AIC-type Methods to Fit ARMA Models to the Log-Lynx Data
In Example 5.3 we examined the log(10) of the classical Canadian lynx dataset that exhibits an interesting 10-year cycle. Tong (1977) fit an AR(11) to these data. As we examine a model fit using AIC, we will include an AR(11) as one of the possible model orders. We issue the ***tswge*** commands:

```
data(lynx)
llynx=log10(lynx)
aic5.wge(llynx,p=0:12,q=0:4)
aic5.wge(llynx,p=0:12,q=0:4,type='aicc')
aic5.wge(llynx,p=0:12,q=0:4,type='bic')
```

The results are shown in Table 6.9. AIC and AICC both select an AR(12) as the top pick with AR(11) as third and second choice for AIC and AICC, respectively. However, BIC, which often selects a model with fewer parameters, chooses an ARMA(2,3), ARMA(3,3), and AR(2) as the top three choices. We will examine the factor tables for the AR(12), ARMA(2,3), and because of its simplicity, the AR(2) model.

TABLE 6.9 Output Showing the Top 5 AIC, AICC, and BIC Choices for the Log Lynx Data

p	*q*	*AIC*	*p*	*q*	*AICC*	*p*	*q*	*BIC*
12	0	–3.128705	12	0	–2.073947	2	3	–2.904979
11	1	–3.121965	11	0	–2.072181	3	3	–2.875738
11	0	–3.121655	11	1	–2.067207	2	0	–2.855197
11	2	–3.115068	11	2	–2.054560	4	3	–2.850560
12	1	–3.112605	12	1	–2.052097	2	1	–2.837161

The three models under consideration are:

(a) AR(12)

$$\left(1-1.116B+.514B^2-.288B^3+.313B^4-.161B^5+.165B^6-.076B^7+.070B^8-.170B^9\right.$$
$$\left.-.138B^{10}+.191B^{11}+.135B^{12}\right)\left(X_t-2.904\right) \tag{6.20}$$

with $\hat{\sigma}_a^2=.035$.

(b) ARMA(2,3)

$$\left(1-1.555B+.953B^2\right)\left(X_t-2.903\right)=\left(1-.453B-.149B^2+.563B^3\right)a_t. \tag{6.21}$$

with $\hat{\sigma}_a^2=.043$.

(c) AR(2)

$$\left(1-1.377B+.740B^2\right)\left(X_t-2.903\right)=a_t. \tag{6.22}$$

with $\hat{\sigma}_a^2=.051$.

If only comparing the coefficients of the three models, it is difficult to see how they are similar to each other and how they are different. In order to understand the models better, we examine the factor tables. The factor tables for the three models are given in Table 6.10.

TABLE 6.10 Factor Tables for AR(12), ARMA(2,3), and AR(2) Models Fit to Log Lynx Data

(a) Factor Table for AR(12) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.578B+0.977B^2$	$0.81\pm0.61i$	0.988	0.103
$1-0.592B+0.877B^2$	0.34 ± 1.01	0.937	0.199
$1-1.825B+0.877B^2$	$1.04\pm0.24i$	0.936	0.036
$1+0.499B+0.780B^2$	$-0.32\pm1.09i$	0.883	0.296
$1+1.245B+0.660B^2$	$-0.94\pm0.79i$	0.813	0.389
$1+1.135B+0.348B^2$	$-1.63\pm0.46i$	0.590	0.456

(b) Factor Table for ARMA(2,3) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.555B+.953B^2$	$.82\pm.62i$	.976	.103
MA FACTOR			
$1-1.203B+.752B^2$	$.80\pm.83i$	.867	.128
$1+.749B$	-1.34	.749	.500

(c) Factor Table for AR(2) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.378B+0.740B^2$	$0.93\pm0.69i$	0.860	0.102

From the factor tables we see that the dominant factor in all three models is a 2nd-order factor associated with a frequency of about $f = 0.10$. However, the roots associated with that factor are much closer to the unit circle in the ARMA(2,3) and AR(12) models than they are for the AR(2) model.

Example 6.6 The Effect of Increasing Realization Length

Recall the ARMA(3,1) model in (6.2) from which the realization **arma31** of length $n = 150$ was generated. To illustrate the effect of increasing realization length on the performance of AIC, AICC, and BIC, we extend the realization in **arma31** to $n = 500$ using the following command:

```
arma31.500=gen.arma.wge(n=500,phi=c(2.57,-2.50,.92),mu=30,sn=65)
```

We then use AIC, AICC, and BIC to select the model orders for the longer series:

```
aic5.wge(arma31.500,p=0:12, q=0:4)
aic5.wge(arma31.500,p=0:12, q=0:4,type= 'bic')
aic5.wge(arma31.500,p=0:12, q=0:4,type= 'aicc')
```

The output (not shown here) reveals that AIC and AICC select an ARMA(7,1) model while BIC picks an AR(4). These models are shown in (6.23) and (6.24), respectively. The ARMA(7,1) model is

$$\begin{aligned}&\left(1-2.521B+2.315B^2-.598B^3-.182B^4-.157B^5+.273B^6-.111B^7\right)\left(X_t-30.217\right)\\&=\left(1-.870B\right)a_t,\end{aligned}\tag{6.23}$$

where $\hat{\sigma}_a^2 = 0.987$. (It is not surprising that the higher order fit has a lower white noise variance estimate.)

The AR(4) model is given by

$$\left(1-1.671B+.898B^2+.196B^3-.164B^4\right)\left(X_t-30.217\right)=a_t, \tag{6.24}$$

where $\hat{\sigma}_a^2 = 1.013$.

Recall that the true mean for the ARMA(3,1) model (6.2) is $\mu = 30$.The sample mean for the dataset **arma31** (of length $n = 150$) is $\bar{x} = 30.610$ while the sample mean for **arma31.500** (of length $n = 500$) is $\bar{x} = 30.217$. That is, as the realization length increased, the sample mean moved closer to the true mean. This behavior fits our understanding of the effect of increasing sample size. However, it is disconcerting that AIC and AICC correctly selected an ARMA(3,1) model for the 150-point dataset but picked an ARMA(7,1) model for the longer realization, **arma31.500**. This illustrates the earlier comment that increasing the realization length is often accompanied by larger model orders selected by AIC. Note that for **arma31.500**, BIC selected an AR(4) model which has the same number of parameters as the original ARMA(3,1) model. We note that the **aic5.wge** output for the **arma31.500** dataset did not include an ARMA(3,1) as one of the top choices by any of the three selection criteria.

We next examine the new models selected by AIC and BIC for the dataset **arma31.500**. Comparing the coefficients in (6.23) and (6.24) to those of the true model (6.2) or the fitted ARMA(3,1) model in (6.3) provides no information concerning the relationship. So, we examine factor tables. Table 6.11(a) is the factor table for the fitted ARMA(3,1) model based on **arma31** previously shown in Table 6.2. As noted earlier the ARMA(3,1) model is dominated by a 2nd-order factor, $1-1.578B+.940B^2$, with roots fairly close to the unit circle and associated system frequency about $f_0 = .1$. The model also includes first-order AR and MA factors with positive real roots for which the root of the AR factor is closer to the unit circle and the MA factor tends to dampen its effect, leaving an overall effect of a weak first-order behavior.

The factor table shown in Table 6.11(b) for the ARMA(7,1) model fit to **arma31.500** includes a second-order factor, $1-1.594B+.937B^2$,very similar to the one in the ARMA(3,1) fit and the true model. The ARMA(7,1) model fit to **arma31.500** contains 1st-order AR and MA factors similar to those in the ARMA(3,1) model fit to **arma31**. Additionally, the ARMA(7,1) model contains two 2nd-order factors whose roots are not close to the unit circle. Knowing that the true model is an ARMA(3,1), these two 2nd-order factors in the ARMA(7,1) model seem to be superfluous.

TABLE 6.11 Factor Table for ARMA(3,1) fit to arma3,1 and the ARMA(7,1) and AR(4) Fits to arma31.500

(a) Factor Table for ARMA(3,1) Model Fit to ARMA31

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.578B+.940B^2$	$.838\pm.600i$	.970	.099
$1-.881B$	1.135	.881	.000
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.775B$	1.290	.775	.000

(b) Factor Table for ARMA(7,1) Model Fit to arma31.500

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.594B+.937B^2$	$.85\pm.59i$	.968	.096
$1-.949B$	1.05	.949	.000
$1-.876B+.370B^2$	$1.19\pm1.14i$	.608	.122
$1-.898B+.337B^2$	$-1.33\pm1.09i$	.581	.391
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.870B$	1.150	.870	.000

GUIDELINES FOR SELECTING MAXIMUM ORDERS P AND Q IN AN AIC-TYPE SEARCH

You may be wondering how to choose the maximum orders P and Q when using **`aic.arma.wge`** and **`aic5.arma.wge`**. Admittedly there are no hard-and-fast rules but the following are considerations:

(1) Use larger orders P and Q than you think you will need.
(2) If the order selected, p is equal to P or q is equal to Q, then enlarge the search.
(3) Let known characteristics of the data help you decide.
   - For example, if you have monthly data, then we recommend choosing P to be larger than 12 to take into consideration any 12^{th}-order monthly behavior. This suggestion will be clearer after discussing seasonal models in Chapter 7.

An Additional Suggestion:

Compare models selected by AIC, BIC, and AICC. If there are major discrepancies in the identified model orders, compare models using factor tables to determine whether the larger order model contains useful information that was missing from the simpler model.

Key Points

1. AIC and AICC have a tendency to select models with more parameters as the realization length increases.
2. Because we know that the true model in Example 6.6 was an ARMA(3,1), then we understand that the fitted ARMA(7,1) model had superfluous factors.
   - This "knowledge" will not be available to us in practical situations.
3. Although the true model was an ARMA(3,1), fitted models selected by AIC, AICC, or BIC will in all likelihood have characteristics similar to those of the true model.
4. In a practical situation with real data and no "true model" it is likely that more than one model can successfully be used for analysis of the data.

6.1.2.4 The Special Case of AR Model Identification

In Section 6.1.1.4 we noted that alternative estimates, Burg and Yule-Walker, were available for estimating the parameters of an AR model. These estimation procedures can also be used in conjunction with AIC-type model identification when you are willing to restrict your search to AR models only. Additionally, as mentioned earlier, the partial autocorrelation function provides a pattern recognition method for AR model identification popularized by Box and Jenkins.

(1) AIC-type AR Model Identification

As previously discussed, the AR model is a widely used model for time series analysis. As noted in Chapter 5, the ARMA model has the advantage of often producing a model with fewer parameters than would be present in a purely AR model fit. As also mentioned, some analysts prefer to deal strictly with AR models.[7]

For AR model identification using AIC-type methods, Equations 6.17−6.19 are modified by setting $q = 0$. The *tswge* functions **`aic.ar.wge`** and **`aic5.ar.wge`** perform AR model identification, and in

7 The functions **`aic.wge`** and **`aic5.wge`** can be used to restrict the search for AR models by setting Q equal to zero. However, when using **`aic.wge`** and **`aic5.wge`,** the parameter estimation is restricted to maximum likelihood.

the AR case the identification can be based on ML, Burg, or YW estimates. The functions **aic.ar.wge** and **aic5.ar.wge** use the white noise variance estimate, $\hat{\sigma}_a^2$, in (6.17)–(6.19) associated with the estimation method selected. In the function calls

```
aic.ar.wge(x,p=0:P,type=aic mode, method=estimation method)
aic5.ar.wge(x,p=0:P,type=aic mode, method=estimation method)
```

the parameter options are **type** = **'aic'**, **'bic'**, or **'aicc'** and **method** = **'mle'**, **'burg'**, or **'yw'**. Note that **type** = **'aic'** and **method='mle'** are defaults.

Example 6.7 AR Model Identification for the A_real realization from the AR(3) Model in (6.14)
Let **A_real** be the realization of length $n = 150$ previously generated from AR(3) model (6.5) using the command

```
A_real=gen.arma.wge(n=150,phi=c(1.71,-1.22,.475),sn=327)
```

The data, sample autocorrelations, and Parzen spectral estimate are shown in Figure 6.6. From the plots in Figure 6.6, it is clear that the data are not white noise.

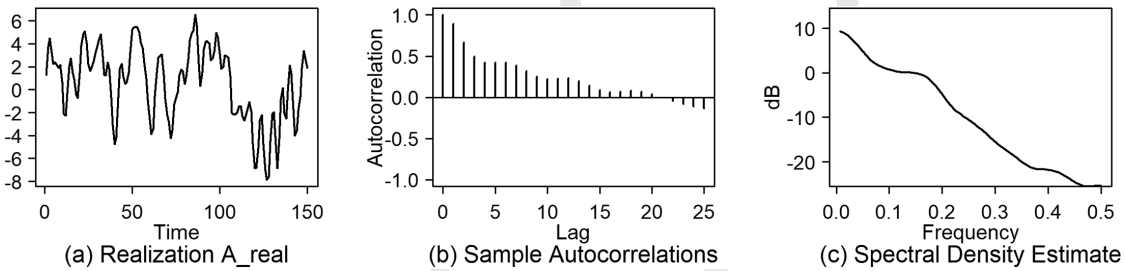

FIGURE 6.6 (a) Realization A_real, (b) and (c) sample autocorrelations and Parzen spectral estimate, respectively, of data in (a).

Assuming we do not know the order of the model that generated the data, we select the order p from among the options, $p = 0,1,\ldots,12$ using the *tswge* command **aic.ar.wge(x,p=0:12).** Among the output is

```
$p 3
```

That is, **aic.ar.wge** has "correctly" chosen an AR(3) model. By default the criterion type is AIC and the estimation method is MLE. The output (not shown here) also includes the ML estimates of the parameters of the AR(3) model (already shown in Table 6.5). Additional output, shown below, includes the value of the minimized AIC, the estimates $\hat{\phi}_j, j = 1,2,$ and 3, the sample average, and the white noise variance.

```
$type "aic"
$value -0.041711
$p 3
$phi 1.7613987 -1.4010223 0.5605187
$xbar 0.5514184
$vara 0.9093326
```

The command

```
aic5.ar.wge(A_real,p=0:12,type= 'bic',method='burg')
```

yields the following output:

```
Five Smallest Values of bic
Method= burg
```

```
p  bic
3  0.03836811
4  0.06384600
5  0.09125547
6  0.12209338
7  0.15453973
```

Interestingly the top orders were $p = 3,4,5,6$, and 7, respectively. In Problem 6.4 you are asked to identify the order of the AR model using each of the nine type/method combinations.

Example 6.8 Fitting an AR Model to the Sunspot Data

In Example 5.6 we fit an AR(9) model to the sunspot data in **`sunspot2.0`**. These data are annual averages from 1700 through 2020 using the sunspot2.0 method of counting. Using the commands

```
data(sunspot2.0)
ss=aic.ar.wge(sunspot2.0,p=0:12,q=0:4)
ss
```

we see that indeed AIC selects an AR(9) model. Output from the **`aic.ar.wge`** command is given below:

```
ss$type "aic"
ss$method "mle"
ss$value 6.361889
ss$p 9
ss$phi 1.165113596 -0.407848045 -0.133850375 0.104019738 -0.068796255
0.007175265 0.023075536 -0.050308201 0.222699470
ss$xbar 78.51651
ss$vara 544.3455
```

Examination of the estimates in **`$phi`** shows that the model in (5.36) is the fitted ML model rounded to two decimal places. Using

```
aic5.ar.wge(sunspot2.0,p=0:12)
```

we obtain the following output:

```
Five Smallest Values of aic
Method= mle

P   aic
9   6.361889
10  6.367760
11  6.373801
12  6.380013
8   6.407677
```

It can be seen that $p = 10$ is a close second choice (there is very little difference in the AIC values). See Problem 6.2.

Try This:

Let p range from 0 to 20 and continue to use the type/method combination AIC/MLE on the dataset **`sunspot2.0`**. Before issuing the following command be ready to see how long it takes for the command to run on your computer.

```
aic5.ar.wge(sunspot2.0,p=0:20)
```

It takes my computer over 10 seconds (yours is probably faster). The top five orders are 9, 18, 10, 19, and 11. Unless the model with $p = 18$ is useful in capturing details in the model missed

by the AR(9) model, most analysts would prefer the model with 9 coefficients to the one with 18 because it is a much simpler model.

Now modify the above command to compute Burg estimates. That is, issue the following command and again check to see how long it takes to run.

```
aic5.ar.wge(sunspot2.0,p=0:20,method='burg')
```

On my computer the results were almost instantaneous. This is an example of the fact that ML estimates can be quite slow to converge. We have seen this dramatically in Examples 6.4 and 6.6. (Did you run the code?) Unfortunately, in the ARMA(p,q) case where q is allowed to take on a value greater than zero, we are "stuck with" ML estimates and convergence may take a long time.

If speed is an issue, then restricting the search to AR models using Burg estimation may be preferable.[8] The top five orders using Burg estimates are 9, 18, 10, 19, and 20–very nearly the same results.

Now, let's see what orders BIC selects. Issue the following command:

```
aic5.ar.wge(sunspot2.0,p=0:20,type='bic')
```

This command computes the "slow" ML estimates, but notice that the top five orders are 9, 10, 8, 11, 12. That is, the BIC tends to select orders that are smaller than AIC (because there is a greater penalty for increasing the order p). Using BIC as above but this time with Burg estimates yields the same order selections.

Conclusions:

(1) The AR(9) model is a winner with AIC and BIC using MLE and Burg estimates. AR(9) also "wins" using AICC. Yule-Walker estimates also select AR(9) as the top model.
(2) Box and Jenkins (1970) select an AR(2) model, but we will show in Section 9.3 that the AR(9) is the better model.

(2) Pattern Recognition Methods for AR Model Identification Using the Partial Autocorrelation Function

Box, Jenkins, and Reinsel (2008) use the partial autocorrelations as their key tool for identifying the order p of an autoregressive model. There are two definitions of the partial autocorrelation function. We give them both in Definition 6.1.

Definition 6.1: Partial Autocorrelations

Ley X_t be a stationary process with autocorrelations $\rho_j, j = 0,1,\ldots$.

(a) The partial autocorrelation at lag k,denoted ϕ_{kk}, is defined to be the correlation between X_t and X_{t+k} conditional on "knowing" the intervening variables X_{t+1}, X_{t+2}, and X_{t+k-1}.[9]
(b) Consider the following Yule-Walker equations where ϕ_{kj} denotes the jth coefficient associated with the kth order Yule-Walker equations.

8 In earlier days of the microcomputer, computing speeds were much slower. Consequently, when using the MLE along with AIC (or its variants) this computation would have taken much longer. In such a case, the first author used to tell his students that they might want to go get a cup of coffee while the AIC routine was running (or in the case of an ARMA setting as in Example 6.6 – check back tomorrow).

9 Recall that the coefficient β_k in the multiple regression equation, $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + e_t$ can be interpreted as the partial correlation between Y and X_k after controlling for $X_1,\ldots$, and $X_{(k-1)}$. Analogously, the coefficient φ_k in the autoregressive equation $X_t = \beta + \varphi_1 X_{(t-1)} + \varphi_2 X_{(t-2)} + \ldots + \varphi_k X_{(t-k)} + a_t$ can be interpreted as the partial correlation between X_t and $X_{(t+k)}$after controlling for $X_{(t+1)},\ldots$, and $X_{(t+k-1)}$.

$k = 1$

$\rho_1 = \phi_{11}$

$k = 2$

$\rho_1 = \phi_{21} + \phi_{22}\rho_1$

$\rho_2 = \phi_{21}\rho_1 + \phi_{22}$

General k

$$\rho_1 = \phi_{k1} + \phi_{k2}\rho_1 + \cdots + \phi_{kk}\rho_{k-1}$$
$$\rho_2 = \phi_{k1}\rho_1 + \phi_{k2} + \cdots + \phi_{kk}\rho_{k-2}$$
$$\vdots$$
$$\rho_k = \phi_{k1}\rho_{k-1} + \phi_{k2}\rho_{k-2} + \cdots + \phi_{kk}$$

The partial autocorrelation function is defined to be ϕ_{kk}, $k = 1, 2....$

Definition 6.1(a) shows that ϕ_{kk} is actually an autocorrelation for each k, so that $|\phi_{kk}| < 1$ for each k. However, Definition 6.1(b) is the definition that is useful in AR model identification. Definition 6.1(b) is straightforward, but may be confusing at first glance. Recall the Yule-Walker equations given in (5.29) and repeated in (6.25):

$$\rho_1 = \phi_1 + \phi_2\rho_1 + \cdots + \phi_p\rho_{p-1}$$
$$\rho_2 = \phi_1\rho_1 + \phi_2 + \cdots + \phi_p\rho_{p-2}$$
$$\vdots$$
$$\rho_p = \phi_1\rho_{p-1} + \phi_2\rho_{p-2} + \cdots + \phi_p. \tag{6.25}$$

The equations in (6.25) are for the case in which the model *is* an AR(p). The ϕ_js, $j = 1, \ldots, p$ are the *actual* coefficients if the model is an AR(p). In Definition 6.1(b) we assume that we know the autocorrelations but not the order p.

- Start by *assuming* $p = 1$, in which case (6.25) says $\rho_1 = \phi_1$. (This is only true if $p = 1$.) To keep things straight, we use the notation $\rho_1 = \phi_{11}$ where the first subscript denotes the AR order assumed and the second designates that ϕ_{11} is the first (and only coefficient) in AR(1) case. (To be clear, $\phi_{11} = \phi_1$ if the model really is an AR(1).)
- *Assuming* the model *is* an AR(2) then

  $\rho_1 = \phi_1 \quad + \phi_2\rho_1$

  $\rho_2 = \phi_1\rho_1 + \phi_2,$

  Again, this is true only if the model is an AR(2). If we only *assume* the model *is* an AR(2), then we write these equations as

  $\rho_1 = \phi_{21} \quad + \phi_{22}\rho_1$

  $\rho_2 = \phi_{21}\rho_1 + \phi_{22},$

  That is, the coefficient ϕ_{21} is the first coefficient in the AR(2) case while ϕ_{22} is the corresponding second coefficient under the assumption of an AR(2). (If the model is an AR(2) then $\phi_{22} = \phi_2$.)

  - Continue this process up to some value P that we believe is at least as large as the true order.
  - and in each case keep the last coefficient, that is $\phi_{11}, \phi_{22}, \ldots$.

Key Point: ϕ_{kk} is the last (kth) coefficient assuming X_t is an AR(k) process.

Consider an AR(p) model with white noise variance σ_a^2 and without loss of generality, zero mean. Then, the AR(p) model, $X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = a_t$, is fully described by $\phi_1, \ldots, \phi_p$. If we decided to express this model as an AR$(p+1)$, then the model would be

$$X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} - 0X_{t-(p+1)} = a_t.$$

That is, the lagged variable $X_{t-(p+1)}$ is not involved in the model, and said another way, $\phi_{p+1} = 0$. Using the same logic, $\phi_{kk} = 0$ if $k > p$.

(3) PACF Model Identification Strategy
Consecutively fit AR(1), AR(2), …, AR(P) models to the data (where P is a maximum order similar to that used in the AIC-type methods). For the kth fitted model, retain and plot $\hat{\phi}_{kk}$, $k = 1,2,\ldots,P$ where $\hat{\phi}_{kk}$ is the last (kth) coefficient of the fitted AR(k) model. If the model is an AR(p), then it should follow that $\hat{\phi}_{p+1,p+1}$, $\hat{\phi}_{p+2,p+2}, \ldots$ will be near zero compared to $\hat{\phi}_{pp}$.

The ***tswge*** function **`pacf.wge`** computes and plots the partial autocorrelations.

Example 6.7 (continued)
We return to the AR(3) dataset **`A_real`** in Example 6.7. A realization from this model, sample autocorrelations, and Parzen spectral density are shown in Figure 6.6. The following code generates the realization **`A_real`** and calls the ***tswge*** **`pacf.wge`** function.

```
A_real=gen.arma.wge(n=150,phi=c(1.71,-1.22,.475),sn=327)
pacf.wge(A_real)
```

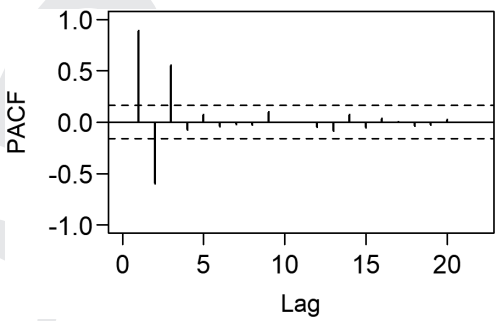

FIGURE 6.7 Base R pacf output for AR(3) data in Figure 6.5(a).

Figure 6.7 shows that $\hat{\phi}_{33}$ is a little larger than .5 and for $k \geq 4$, the sample partial autocorrelations are near zero. The sample partial autocorrelations, $\hat{\phi}_{kk}$, for $k > p$ are asymptotically normal and approximately independent with zero mean and variance $1/n$..[10] The dashed lines in the plot are at $\pm 2(1/\sqrt{n})$, and these limits are often used as a guide for testing $\phi_{kk} = 0$. The plot in Figure 6.7 strongly suggests an AR(3). See problem 6.13 for practice using the partial autocorrelations

10 See Quenouille (1949).

Key Points

1. The 95% limit lines for $\hat{\phi}_{kk}$ for $k > p$ where p is the actual order of the AR model are similar to those for sample autocorrelations of white noise data. That is, the 95% applies separately for each $k(>p)$ so it will not be surprising for about 5% of the sample partial autocorrelations (for $k > p$) to fall outside the limit lines.
2. The estimated partial autocorrelations, $\hat{\phi}_{kk}$, calculated using Cramer's rule are the estimates of the kth autoregressive coefficient using the Yule-Walker method.
   - Base R function **`pacf`** uses Yule-Walker based partial autocorrelations and will be subject to the problems associated with Yule-Walker estimates such as those mentioned in Example 6.3.
   - tswge function **`pacf.wge`** provdes the ability to base the calculation of the partial autocorrelations on ML and Burg estimates as well as the traditional YW estimates.

6.2 FORECASTING USING AN ARMA(*P*,*Q*) MODEL

One of the main purposes for fitting a time series model is to forecast (or predict) future behavior of the time series given a finite realization of its past. We may want to forecast sales for the next quarter based on sales for the past few years, or we may want to address the more challenging problem of forecasting sales for the next four quarters. Based on historical data we could forecast monthly home sales in a particular city for the next year. Other examples include prices of oil, gross national product, Dow Jones Index, and of course, a topic of current interest and concern is to forecast the global temperature for the next few decades. In chapters that follow we will discuss forecasting time series data based on models that include trends, seasonal components, multiple explanatory variables, and so forth.

In Section 6.2.1 we focus on the problem of forecasting the future values of data that satisfy the conditions of stationarity. ARMA-based forecasting was popularized by G.E.P. Box and G.M. Jenkins in their classic 1970 book. These forecasts are based on the underlying assumption that the future is guided by its correlation to the past.

6.2.1 ARMA Forecasting Setting, Notation, and Strategy

It is assumed that a realization has been observed from the time series X_t, for times $t = 1, 2, \ldots, t_0$. The goal is to forecast a future value (or values), say X_{t_0+1} or $X_{t_0+1}, \ldots, X_{t_0+4}$.

Note that the typical case is $t_0 = n$, that is, $X_{t_0} = X_n$ is the last value in the observed realization. In some cases we may, for example, want to "forecast" the last four values in the realization (which are known values) to assess the performance of our forecasts. In this case, $t_0 = n - 4$.

6.2.1.1 Strategy and Notation

$\hat{X}_{t_0}(\ell)$ is the forecast of $X_{t_0+\ell}$ given data up to time t_0

- t_0 is the *forecast origin*
- ℓ is the *lead time* or *horizon* that is, the number of units (*steps ahead*) that we want to forecast

For example, suppose we observe a realization from the time series X_t for times $t = 1,2,\ldots,t_0 = 10$, and we want to forecast $X_{12}\left(= X_{t_0+2}\right)$.

Then $\hat{X}_{10}(2)$ is the forecast of X_{12}. Also,

$\hat{X}_{10}(3)$ is the forecast of X_{13}
$\hat{X}_{10}(7)$ is the forecast of X_{17}
$\hat{X}_{10}(\ell)$ is the forecast of $X_{10+\ell}$.

6.2.1.2 Forecasting $X_{t_0+\ell}$ for $\ell \le 0$

$$\hat{X}_{t_0}(\ell) = X_{t_0+\ell} \text{ if } \ell \le 0. \tag{6.26}$$

Note that equation (6.26) says that $\hat{X}_{10}(0) = x_{10}$, which has already been observed. Also,

$\hat{X}_{10}(-1) = x_9$
$\hat{X}_{10}(-2) = x_8$
$\vdots$

all of which have already been observed.

Key Point: Equation 6.26 is trivial and you may ask, "Why would we want to 'forecast' something we have already observed?" Good question. The fact is that the iterative equations for computing $\hat{X}_{t_0}(\ell)$ for $\ell > 0$ involve "forecasts" $\hat{X}_{t_0}(m)$ for negative values of m. As defined in (6.26) these "forecasts" are actually values we have already observed. See Example 6.9.

6.2.1.3 Forecasting $a_{t_0+\ell}$ for $\ell > 0$

$$\hat{a}_{t_0}(\ell) = 0 \text{ if } \ell > 0. \tag{6.27}$$

Note that $\hat{a}_{10}(1)$ is the forecast of a_{11} given data to time $t = 10$. Because a_t is random white noise with zero mean, and as of time $t = 10$, a_{11} is unknown and uncorrelated with past values of a_t, it makes intuitive sense to define $\hat{a}_{10}(1) = 0$. So, in general the forecast formula in (6.27) makes sense for all $\ell > 0$.

6.2.2 Forecasting Using an AR(p) Model

We first consider the simplest AR(p)case, that is, forecasting based on an AR(1).

6.2.2.1 Forecasting Using an AR(1) Model

Suppose X_t satisfies an AR(1) model given by $X_t = \phi_1 X_{t-1} + (1-\phi_1)\mu + a_t$, and that we have observed values for $x_1, x_2, \ldots, x_{10}\left(= x_{t_0}\right)$. As mentioned, we will estimate ϕ_1 with the ML estimate, $\hat{\phi}_1$, and the mean

μ by $\bar{x}$. Using the fitted model, the formula for (unobserved) X_{11} is $X_{11} = \hat{\phi}_1 x_{10} + (1-\phi_1)\bar{x} + a_{11}$. Because X_{11} and a_{11} have not been observed, we replace all quantities with their forecasts as of time $t = 10$ yielding

$$\hat{X}_{10}(1) = \hat{\phi}_1 \hat{X}_{10}(0) + \bar{x}(1-\hat{\phi}_1) + \hat{a}_{10}(1).$$

Obviously, by (6.26), $\hat{X}_{10}(0) = x_{10}$ and by (6.27), $\hat{a}_{10}(1) = 0$. This leads to the general formula for calculating forecasts using an AR(1) model.

The ***basic formula for forecasting from a fitted AR(1) model*** is

$$\hat{X}_{t_0}(\ell) = \hat{\phi}_1 \hat{X}_{t_0}(\ell-1) + \bar{x}(1-\hat{\phi}_1). \tag{6.28}$$

The formula in (6.28) is used recursively. That is, to calculate $\hat{X}_{t_0}(\ell)$ we iteratively calculate

$$\hat{X}_{t_0}(1),\ \hat{X}_{t_0}(2),\ \ldots,\ \hat{X}_{t_0}(\ell-1),\ \hat{\boldsymbol{X}}_{\boldsymbol{t_0}}(\ell).$$

($\hat{X}_{t_0}(\ell)$ is in bold face to emphasize that it was the forecast that was desired. The other forecasts were involved in the iterative procedure used to find $\hat{X}_{t_0}(\ell)$. For example, to forecast X_{12} based on data to time $t = 10$. the forecast formula in (6.28) says that

$$\hat{X}_{10}(2) = \hat{\phi}_1 \hat{X}_{10}(1) + \bar{x}(1-\hat{\phi}_1),$$

but we must first calculate $\hat{X}_{10}(1)$ which is given by

$$\hat{X}_{10}(1) = \hat{\phi}_1 \hat{X}_{10}(0) + \bar{x}(1-\hat{\phi}_1).$$

Based on (6.26), $\hat{X}_{10}(\ell) = X_{t_0+\ell}$ if $\ell \le 0$, so it follows that $\hat{X}_{10}(0) = x_{10}$, a *known value*. So, to calculate $\hat{X}_{10}(12)$ we recursively calculate

$$\hat{X}_{10}(1) = \hat{\phi}_1 x_{10} + \bar{x}(1-\hat{\phi}_1)$$
$$\hat{\boldsymbol{X}}_{\mathbf{10}}(\mathbf{2}) = \hat{\phi}_1 \hat{X}_{10}(1) + \bar{x}(1-\hat{\phi}_1).$$

QR 6.5 AR(1) Forecasting

Example 6.9 AR(1) Forecasting Example

Consider the following ***tswge*** code:

```
x=c(14.0,14.8,14.0,14.0,13.8,12.4,12.9,11.9,12.0,10.9,10.5,9.2,10.7,11.3,
9.2,8.6,8.2,7.3,7.9,8.8)
aic.x=aic.wge(x,p=0:4,q=0:0) # aic.x$p is the value of p selected by AIC
# AIC picks p=1
est.x=est.ar.wge(x,aic.x$p)
```

```
# the MLE of phi(1)=.92 (est.x$phi)
# white noise variance estimate is .82 (est.x$avar)
mean(x)
# the estimated mean is 11.12
```

After running the above code we see that AIC picks an AR(1) and the fitted AR(1) model is

$$X_t = 0.92X_{t-1} + .89 + a_t, \tag{6.29}$$

where $\hat{\sigma}_a^2 = 0.82$, because $\bar{x}(1-\hat{\phi}_1) = 11.12(1-.92) = .89$. The dataset **x** is plotted in Figure 6.8(a) along with a horizontal line at the mean, $\bar{x} = 11.12$. We see the typical wandering behavior associated with an AR(1) model with positive real root. That is, if the value x_{t_1} is below the mean, then observations in the neighborhood of t_1 will also tend to be below the mean (and similarly for values above the mean).

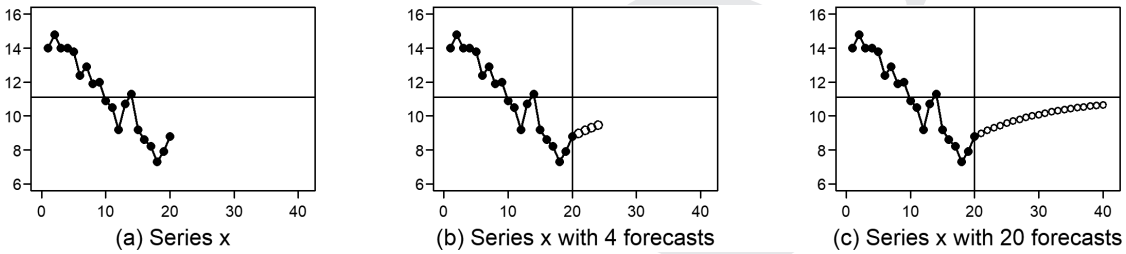

FIGURE 6.8 (a) Realization x for which the fitted model is (6.29), (b) Realization in (a) along with forecasts for t = 21,22, 23, and 24 calculated by hand, and (c) realization in (a) along with forecasts for t = 21,22,...,30 calculated using `fore.arma.wge`.

For purposes of calculating forecasts using the formula $\hat{X}_{t_0}(\ell) = \hat{\phi}_1\hat{X}_{t_0}(\ell-1) + \bar{x}(1-\hat{\phi}_1)$, we note that $\hat{\phi}_1 = .92$ and $x_{20} = 8.8$. The forecast function for this dataset is

$$\hat{X}_{20}(\ell) = .92\hat{X}_{20}(\ell-1) + .89. \tag{6.30}$$

The forecasts of X_{21} through X_{24} are given by

$$\begin{aligned}
\hat{X}_{20}(1) &= .92\hat{X}_{20}(0) + .89 = .92(8.8) + .89 = 8.99\\
\hat{X}_{20}(2) &= .92\hat{X}_{20}(1) + .89 = .92(8.99) + .89 = 9.16\\
\hat{X}_{20}(3) &= .92(9.16) + .89 = 9.32\\
\hat{X}_{20}(4) &= .92(9.32) + .89 = 9.46\\
&\vdots
\end{aligned}$$

The realization in **x** and forecasts $\hat{X}_{20}(1), \hat{X}_{20}(2), \hat{X}_{20}(3)$, and $\hat{X}_{20}(4)$ calculated above are shown in Figure 6.8(b). It is important to note that because **x** is from a stationary time series, there is an "attraction" to the mean, in this case $\bar{x} = 11.12$. Thus, it is not surprising that because $x_{20} = 8.8$ is below the mean, forecasts trend upward toward the mean. Forecasts shown in Figure 6.8(c) were obtained using the ***tswge*** command

```
fore.x=fore.arma.wge(x,est.x$phi,n.ahead=20,limits=FALSE)
```

where **x** is the dataset in Figure 6.8(a), **`est.x$phi`** are the ML estimates of the AR(1) fit to the data, **`n.ahead=20`** instructs the function to calculate and plot the forecasts up to 20 steps ahead, that is.

$\hat{X}_{20}(\ell), \ell = 1,\ldots,20,$ shown in Figure 6.8(c). Also, **`limits=FALSE`** indicates that we want to omit the prediction limits from the plot (these will be discussed in Section 6.2.5). In Figure 6.8(c) it is clear that the forecasts continue trending upward toward the mean, $\overline{X} = 11.12$. The values for $\hat{X}_{20}(\ell), \ell = 1,\ldots,4$ are given as the first four values in **`fore.x$f`** which are

```
fore.x$f 8.980427 9.146823 9.300277 9.441798
```

Note that these vary slightly from the calculations above because the coefficient estimates are rounded to two decimal places.

QR 6.6 AR(1) Forecasting Real Data

The AR(0) Model: Consider the case of normal noise, a_t. The "AR(0)" model is given by

$$X_t - \mu = a_t, \tag{6.31}$$

where $a_t \sim N(0, \sigma_a^2)$. If $X_t, t = 1,\ldots,n$ satisfies this model, then $X_1, X_2,\ldots,X_n$ are actually a random sample from a Normal(μ, σ_a^2) distribution, and there is no dependence on time. Consequently, the best estimate of X_{n+1} is $\overline{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$. That is, if $t_0 = n$, then $\hat{X}_{t_0}(1) = \overline{x}$. Likewise, because the data are uncorrelated (and not dependent on time), then $\hat{X}_{t_0}(\ell) = \overline{x}$ for any $\ell > 0$. Of course, these are "unexciting" forecasts.

Key Point: Earlier we said that an AR(1) is the simplest of the AR(p) models. Technically, white noise is the simplest of the AR(p) models. Actually, white noise, that is, an AR(0) (and an ARMA(0,0)) is the "simplest model".

QR 6.7 AR(2) Forecasting

Example 6.10 AR(2) forecasting example
Consider the following ***tswge*** code.

```
y=c(40.3,36.6,40.1,42.4,40.7,38.5,39.3,42.0,41.5,39.3,37.8,40.4,43.5,41.5,37.
4,37.8,40.6,43.2,40.7,38.71,40.9,40.7,40.2,39.5,39.4)
# y is a time series of length n=25
aic.y=aic.wge(y,p=0:10,q=0:0)
# AIC picks p=2 so we estimate the parameters
est.y=est.ar.wge(y,aic.y$p)
# MLE in $phi: 0.30 -0.90
# white noise variance estimate is $avar=.57
# sample mean is $xbar=40.12
```

The output from the above code shows that AIC selects an AR(2) and the fitted AR(2) model is

$$\left(1-0.3B+0.9B^2\right)\left(X_t-40.12\right)=a_t, \tag{6.32}$$

where $\hat{\sigma}_a^2 = 0.57$.The one-line factor table is shown in Table 6.12.

TABLE 6.12 Factor Table for Model: $\left(1-0.30B+0.90B^2\right)\left(X_t-40.12\right)=a_t$.

FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-0.3B+.9B^2$	$.17\pm.1.04i$	.95	.22

The factor table shows a single second-order factor whose roots are fairly close to the unit circle ($\lvert r\rvert^{-1} = .95$) and are associated with system frequency $f = .22$ (that is, a cycle length of $1/.22 = 4.5$). The data shown in Figure 6.9(a) have a cyclic behavior with cycle length of about four or five time points. Also, $\bar{x} = 40.12$ is plotted as a horizontal line on the plot.

Key Point: We could have used **`aic5.wge`** to examine other models, but because the focus of this example is the forecast function, we simply use the model selected by AIC. Although we recommend checking alternatives to the model automatically selected by AIC, we will at times simply use the AIC choice for simplicity of discussion.

The model in (6.32) can be written as

$$X_t = .3X_{t-1} - .9X_{t-2} + 64.19 + a_t \tag{6.33}$$

because $\bar{x}(1-\hat{\phi}_1-\hat{\phi}_2) = 40.12\left(1-.3+.9\right) = 64.19$. Using (6.33) we have the forecast function

$$\hat{X}_{25}\left(\ell\right) = .30\hat{X}_{25}\left(\ell-1\right) - .90\hat{X}_{25}\left(\ell-2\right) + 64.19, \tag{6.34}$$

and because $x_{24} = 39.5$ and $x_{25} = 39.4$, the forecasts of X_{26} through X_{29} are given by

$$\hat{X}_{25}\left(1\right) = .30\hat{X}_{25}\left(0\right) - .90\hat{X}_{25}\left(-1\right) + 64.19 = .30\left(39.4\right) - .90\left(39.5\right) + 64.19 = 40.46$$
$$\hat{X}_{25}\left(2\right) = .30\hat{X}_{25}\left(1\right) - .90\hat{X}_{25}\left(0\right) + 64.19 = .30\left(40.46\right) - .90\left(39.4\right) + 64.19 = 40.87$$
$$\hat{X}_{25}\left(3\right) = .30\left(40.87\right) - .90\left(40.46\right) + 64.19 = 40.04$$
$$\hat{X}_{25}\left(4\right) = .30\left(40.04\right) - .90\left(40.87\right) + 64.19 = 39.42$$

Forecasts $\hat{X}_{25}\left(\ell\right), \ell = 1,\ldots,4$ calculated above are shown in Figure 6.9(b), while the forecasts $\hat{X}_{25}\left(\ell\right), \ell = 1,\ldots,20$ shown in Figure 6.9(c) were obtained using the ***tswge*** command

```
fore.y=fore.arma.wge(y,phi=est.y$phi,n.ahead=20,limits=FALSE)
```

where **`y`** is the dataset specified in the above code and plotted in Figure 6.9(a). The parameters of the fitted model in **`est.y$phi`** are the ML estimates of the AR(2) fit to the data, and **`n.ahead=20`** instructs the function to calculate and plot the forecasts up to 20 steps ahead. The use of **`limits=FALSE`** indicates that we want to omit the prediction limits from the plot (these will be discussed in Section 6.2.5). Note that the 20 forecasts are in vector **`fore.y$f`**. In Figure 6.9(c) it is clear that the forecasts have a damped

cyclic behavior around the mean, $\overline{X} = 40.12$. and with a cycle length of about 4 to 5. The values for $\hat{X}_{25}(\ell), \ell = 1,\ldots,4$ are given as the first four values in

```
fore.x$f: 40.45693 40.86736 40.04601 39.42927
```

which vary slightly from the calculations above due to the fact that we kept only two decimal places in the coefficient estimates.

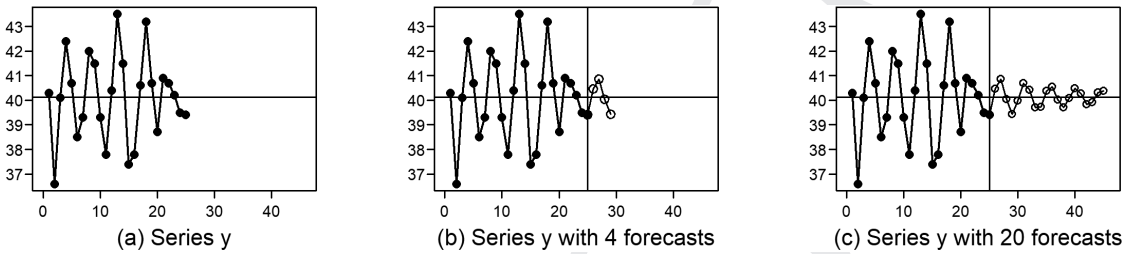

FIGURE 6.9 (a) Realization y for which the fitted model is (6.34), (b) Realization in (a) along with forecasts for $t = 26,27,28$, and 29 calculated by hand above, and (c) realization in (a) along with forecasts for $t = 26,27,\ldots,35$ calculated using fore.arma.wge.

6.2.3 Basic Formula for Forecasting Using an ARMA(p,q) Model

Equation (6.35) gives the general formula for calculating forecasts from a fitted ARMA(p,q) model.

$$\hat{X}_{t_0}(\ell) = \sum_{j=1}^{p} \hat{\phi}_j \hat{X}_{t_0}(\ell - j) - \sum_{j=\ell}^{q} \hat{\theta}_j \hat{a}_{t_0+\ell-j} + \overline{x}\left[1 - \sum_{j=1}^{p} \hat{\phi}_j\right]. \tag{6.35}$$

Note that for the AR(p) case these forecasts simplify to

$$\hat{X}_{t_0}(\ell) = \sum_{j=1}^{p} \hat{\phi}_j \hat{X}_{t_0}(\ell - j) + \overline{x}\left[1 - \sum_{j=1}^{p} \hat{\phi}_j\right]. \tag{6.36}$$

Thus we see that ARMA(p,q) forecasts for $q > 0$ depend on white noise estimates, while white noise estimates are not involved in AR(p) forecasts. Obtaining the white noise estimates is computationally intensive, and the ***tswge*** forecasting routines use the backcasting procedure. (See Woodward et al. (2017)).

Key Point: AR forecasts using (6.36) are simply the special case in which $-\sum_{j=\ell}^{q} \hat{\theta}_j \hat{a}_{t_0+\ell-j} = 0$

QR 6.8 ARMA Forecasting Formula

Example 6.11 Forecasts from an ARMA(2,1) Model
Consider the ARMA(2,1) model

$$\left(1-1.6B+.9B^2\right)(X_t-20)=\left(1-.9B\right)a_t, \tag{6.37}$$

where $\sigma_a^2=1$. Table 6.13 shows that the AR part of the model is associated with a system frequency of $f_0=.09$ (or periods of about 11). The MA part indicates the "removal" of frequency behavior at around $f=0$. Figure 6.10(a) shows a realization of length $n=100$, generated using the command

```
arma21=gen.arma.wge(n=100,phi=c(1.6,-.9),theta=.9,mu=20,sn=5789)
```

TABLE 6.13 Factor Table for ARMA(2,1) Model

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.60B+90B^2$	$.89\pm.57i$	.95	.09
MA-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-.90B$	1.11	.90	.00

The realization shows a cyclic behavior with cycle length about 11 (1/.09). The command

```
aic5.wge(arma21,p=0:12,q=0:4)
```

returns an ARMA(2,1) as the top choice with an ARMA(3,1) being the second choice. The command

```
est21=est.arma.wge(arma21,p=2,q=1)
```

returns the ML fitted model

$$\left(1-1.583B+.899B^2\right)(X_t-20.038)=\left(1-.929B\right)a_t, \tag{6.38}$$

with $\hat{\sigma}_a^2=.84$. Using (6.35) the forecast function is

$$\hat{X}_{100}(\ell)=\sum_{j=1}^{2}\hat{\phi}_j\hat{X}_{100}(\ell-j)-\sum_{j=\ell}^{1}\hat{\theta}_j\hat{a}_{100+\ell-j}+\overline{x}\left[1-\sum_{j=1}^{2}\hat{\phi}_j\right]. \tag{6.39}$$

Notice that the term

$$-\sum_{j=\ell}^{1}\hat{\theta}_j\hat{a}_{100+\ell-j}$$

is $-\hat{\theta}_1\hat{a}_{100}$ when $j=1$ and the sum is null for $\ell>1$ because the lower limit (ℓ) is greater than the upper limit (1). So,

$$\hat{X}_{100}(1)=1.583\hat{X}_{100}(0)-.899\hat{X}_{100}(-1)-.929\hat{a}_{100}(100)+20.038\left(1-1.583+.929\right) \tag{6.40}$$

The term $20.038\left(1-1.583+.899\right)=20.038\left(.316\right)=6.332$, and by typing **arma21**, 100 data values of the realization are displayed. Based on (6.40), in order to compute $\hat{X}_{100}(1)$, the values for $\hat{X}_{100}(0)=x_{100}=19.046$ and $\hat{X}_{100}(-1)=x_{99}=20.596$ are needed. Thus, (6.40) becomes

$$\hat{X}_{100}(1) = 1.583(19.046) - .899(20.596) - .929\hat{a}_{100}(100) + 6.332. \tag{6.41}$$

However, we are still not in a position to calculate the forecast because we need the value $\hat{a}_{100}(100)$. The residuals are included in the output of **`est.arma.wge`** in the output vector **`est21$res`**. To see the 100 residuals, calculated using backcasting, type **`est21$res`**. From this listing of the residuals, it can be seen that $\hat{a}_{100}(100) = -.029$. Consequently,

$$\hat{X}_{100}(1) = 1.583(19.046) - .899(20.596) - .929(-.029) + 6.332 = 17.993.$$

The forecasts for $\hat{X}_{100}(2)$ and $\hat{X}_{100}(3)$, which do not include the moving average term in (6.41), are given by

$$\begin{aligned}\hat{X}_{100}(2) &= 1.583\hat{X}_{100}(1) - .899\hat{X}_{100}(0) + 6.332\\ &= 1.583(17.993) - .899(19.046) + 6.332 = 17.693\\ \hat{X}_{100}(3) &= 1.583\hat{X}_{100}(2) - .899\hat{X}_{100}(1) + 6.332\\ &= 1.583(17.693) - .899(17.993) + 6.332 = 18.164\end{aligned}$$

It is obvious that the forecasts calculated above, which required considerable effort, are best calculated using computer code. After running the ***tswge*** code above to calculate realization **`arma21`** and estimate the parameters, we can calculate the forecasts using

```
fore.21=fore.arma.wge(arma21,phi=est21$phi,theta=est21$theta,n.ahead=30,
limits=FALSE)
```

The first three forecasts, shown under **`fore.21$f,`** are given by **`17.99311, 17.696328,`** and **`18.16562`**. These match closely to those calculated above with slight differences due to rounding. Figure 6.11(b) shows the realization of length $n = 100$ along with the forecasts for the next 30 time points. The forecasts have the appearance of a damping sinusoid centered about the line $\bar{x} = 20.038$. The system frequency associated with $(1 - 1.583B + .899B^2)$ is $f_0 = 0.093$ which is consistent with the fact that the cycle length for the data and the forecasts is about $1/.093 = 10.75$.

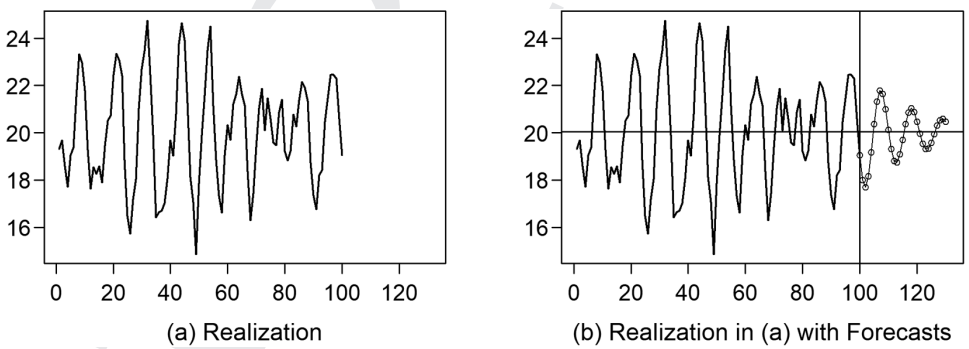

FIGURE 6.10 Generated data and forecasts using a fitted ARMA(2,1) model for the next 30 time periods calculated using `fore.arma.wge`.

Example 6.12 Sunspot Numbers with Forecasts

In Example 5.6 we used an AR(9) model for the sunspot data in file **`sunspot2.0`** that contains sunspot numbers based on the new counting procedure for the years 1700 through 2020. Example 6.8 showed that AIC selects an AR(9) model for these data. The ML estimates are shown in (5.36) and again in the output vector **`ss$phi`** shown in Example 6.8. Table 5.6 shows the factor table for this fitted model. These

sunspot numbers are plotted in Figure 6.11 along with forecasts for the 30 years following 2020. There you can see that the forecasts follow a damped sinusoidal pattern with a period of about 10–11 years. This is to be expected based on the factor table in Table 6.5. The horizontal line is at the mean 78.97 and the vertical line is at year 2020. The forecasts were obtained using the ***tswge*** code

```
data(sunspot2.0)
aic.ss=aic.wge(sunspot2.0,p=0:10,q=0:4)
est.ss=est.ar.wge(sunspot2.0,p=aic.ss$p) # aic selects p=9
fore.ss=fore.arma.wge(sunspot2.0,phi=est.ss$phi,n.ahead=30,limits=FALSE)
```

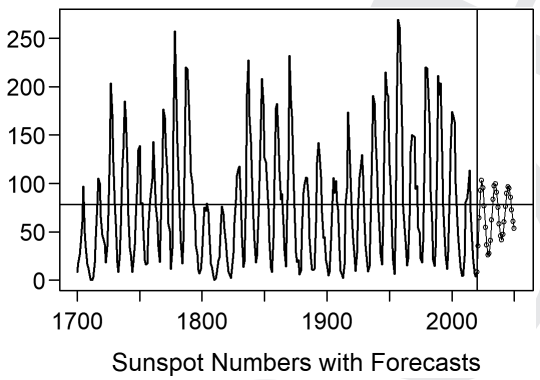

FIGURE 6.11 Sunspot numbers 1700–2020 with forecasts for the 30 years following 2020 calculated using fore.arma.wge based on the AR(9) fit in Examples 5.6 and 6.8.

Example 6.13 Canadian Lynx Data Revisited

In Example 6.5 we examined the log(10) of the classical Canadian lynx dataset that exhibits an interesting 10-year cycle. In that example we decided that AR(12), ARMA(2,3), and an AR(2) were models worth pursuing further. Examining the factor tables in Table 6.10 we noted that the dominant factor in all three models is a 2nd-order factor associated with a frequency of $f = 0.10$. However, the roots associated with that factor are much closer to the unit circle in the ARMA(2,3) and AR(12) models than they are for the AR(2) model.

Figure 6.12 shows forecasts for 30 years following 1934, and we see that the ARMA(2,3) and AR(12) models detect the 10-year period and predict this pattern is to continue while the AR(2) model forecasts quickly damp to the mean. It is interesting to note the similarity between the ARMA(2,3) and AR(12) forecasts. While the ARMA(2,3) and AR(12) produce "more appealing-looking" forecasts than the AR(2) model, this does not indicate they are better forecasts. The assessment of forecast performance will be discussed in Section 6.2.6.

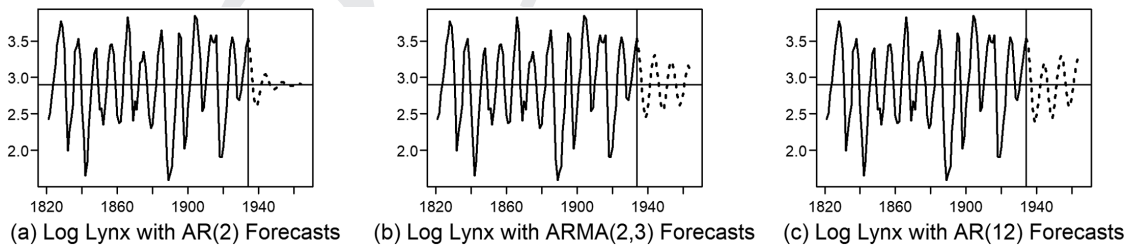

FIGURE 6.12 Log Lynx data showing (a) forecasts using AR(2) fit, (b) forecasts using ARMA(2,3) fit, and (c) forecasts using the AR(12) fit.

6.2.4 Eventual Forecast Functions

In the forecasting examples using AR and ARMA models (which are stationary models) it should be noted that in all cases, the forecasts eventually tend toward the realization mean as the number of steps ahead, ℓ, increases. In Figure 6.13 we replot forecasts from the AR(1) and AR(2) models shown previously in Figure 6.8(c) and Figure 6.9(c). In both cases the forecasts trend toward the sample mean illustrated by the horizontal line, $\bar{x} = 11.12$ and 40.12 in Figures 6.13(a) and 6.13(b), respectively. For stationary AR and ARMA models, the autocorrelations, ρ_k, damp to zero as $k \to \infty$. Thus, the correlations between observed values for X_t up to time t_0 and $X_{t_0+\ell}$ decrease as ℓ increases. Because the observed values give very little information about $X_{t_0+\ell}$ when ℓ is large, it makes sense to simply forecast $\bar{x}$. In both cases in Figure 6.13, the forecasts trend toward the sample mean shown as a horizontal line on the plots.

Key Fact: The tendency for the forecasts to eventually tend to the sample mean can be "disappointing" at first glance. Novice forecasters may think that they can "do better" without the use of a model. However, if the data are stationary, then it would be a mistake to make "bold" forecasts into the "distant" future because it is essentially uncorrelated with observed data. See Section 6.2.6 Assessing Forecast Performance..

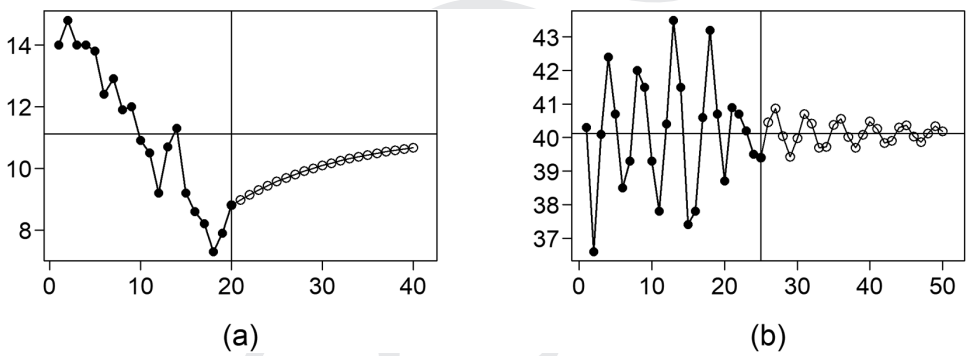

FIGURE 6.13 Forecasts previously shown in Figure 6.8(c) and 6.9(c).

6.2.5 Probability Limits for ARMA Forecasts

We have found how to find forecasts of $X_{t_0+\ell}$ based on data to time t_0 using ARMA models. In addition to finding a forecast for $X_{t_0+\ell}$, that is, $\hat{X}_{t_0}(\ell)$, the knowledgeable forecaster will also want to assess the uncertainty associated with the prediction. As an example, we may want to construct limits such that there is a 95% chance that the actual future value of $X_{t_0+\ell}$ falls within these limits. In this section we will learn how to construct such limits.

The difference, $e_{t_0}(\ell) = X_{t_0+\ell} - \hat{X}_{t_0}(\ell)$, is called the *forecast error*. That is, it measures the amount by which the forecast of $X_{t_0+\ell}$ differs from the actual value. The following summarizes the properties of the forecast error.

Key Point: In the following we assume *normal white noise*, a_t, with zero mean and finite variance σ_a^2.

6.2.5.1 Facts about Forecast Errors

First recall that a stationary ARMA model can be expressed in GLP form, $X_t - \mu = \sum_{k=0}^{\infty} \psi_k a_{t-k}$. It follows that $X_{t_0+\ell}$ satisfies $X_{t_0+\ell} - \mu = \sum_{k=0}^{\infty} \psi_k a_{t_0+\ell-k}$. With this as a foundation, consider the following facts about the forecast error, $e_{t_0}(\ell)$:

(1) $e_{t_0}(\ell) = \sum_{k=0}^{\ell-1} \psi_k a_{t_0+\ell-k}$

(2) $e_{t_0}(\ell)$ is normally distributed (it is a finite linear combination of normal random variables) with

(a) mean $E\left[e_{t_0}(\ell)\right] = 0$ ($E[a_t = 0]$ for all t)

(b) variance $Var\left[e_{t_0}(\ell)\right] = Var\left[\sum_{k=0}^{\ell-1} \psi_k a_{t_0+\ell-k}\right]$

$$= \sigma_a^2 \sum_{k=0}^{\ell-1} \psi_k^2.$$

$(1-\alpha)\times 100\%$ Prediction Limits

Using the above facts, it follows that

$$\frac{e_{t_0}(\ell) - 0}{\sigma_a \sqrt{\sum_{k=0}^{\ell-1} \psi_k^2}} \sim N(0,1). \tag{6.42}$$

That is,

$$\frac{X_{t_0+\ell} - \hat{X}_{t_0}(\ell)}{\sigma_a \sqrt{\sum_{k=0}^{\ell-1} \psi_k^2}} \sim \mathrm{N}(0,1),$$

from which it follows that

$$\mathrm{prob}\left\{-z_{1-\alpha/2} \le \frac{X_{t_0+\ell} - \widehat{X_{t_0}(\ell)}}{\sigma_a \sqrt{\sum_{k=0}^{\ell-1} \psi_k^2}} \le z_{1-\alpha/2}\right\} \sim \mathrm{N}(0,1), \tag{6.43}$$

where z_β is the $\beta\times 100\%$ percentile of the standard normal distribution. For example, $z_{1-.05/2} = z_{.975} = 1.96$. The theoretical $(1-\alpha)\times 100\%$ prediction interval for $X_{t_0+\ell}$ is

$$\hat{X}_{t_0}(\ell) - z_{1-\alpha/2}\sigma_a \left\{\sum_{k=0}^{\ell-1} \psi_k^2\right\}^{\frac{1}{2}} \le X_{t_0+\ell} \le \hat{X}_{t_0}(\ell) + z_{1-\alpha/2}\sigma_a \left\{\sum_{k=0}^{\ell-1} \psi_k^2\right\}^{\frac{1}{2}} = 1-\alpha,$$

or

$$\hat{X}_{t_0}(\ell) \pm z_{1-\alpha/2}\,\sigma_a \left\{\sum_{k=0}^{\ell-1} \psi_k^2\right\}^{\frac{1}{2}}. \tag{6.44}$$

In practice we will not know σ_a so the white noise variance, σ_a^2, is estimated by $\hat{\sigma}_a^2$ based on the backcasting procedure. Also, the notation ψ_k denotes the ψ-weights of the "true" model, so we denote the ψ-weights of the fitted model by $\hat{\psi}_k$. Consequently, in practice, the $(1-\alpha)\times 100\%$ prediction interval for $X_{t_0+\ell}$ is

$$\hat{X}_{t_0}(\ell) \pm z_{1-\alpha/2}\,\hat{\sigma}_a \left\{\sum_{k=0}^{\ell-1}\hat{\psi}_k^2\right\}^{\frac{1}{2}}, \tag{6.45}$$

and, for example, a 95% prediction interval is

$$\hat{X}_{t_0}(\ell) \pm 1.96\,\hat{\sigma}_a \left\{\sum_{k=0}^{\ell-1}\hat{\psi}_k^2\right\}^{\frac{1}{2}}. \tag{6.46}$$

QR 6.9 ARMA Forecast Limits

Key Points

1. The $(1-\alpha)\times 100\%$ prediction limits apply to individual forecasts. If 95% prediction limits are used to obtain forecasts $\hat{X}_{t_0}(\ell), \ell = 1,\ldots,m$, we should not be surprised to find that about 5% of the actual values fall outside the prediction limits.
2. As we go from ℓ step-ahead to $\ell+1$ step-ahead forecasts, the forecast limits expand because $\hat{\psi}_\ell^2$ is added to the sum of squared ψ-weights (unless $\hat{\psi}_\ell^2 = 0$).

Example 6.14 (revisited)
Figure 6.14 shows a realization of length $n = 100$ to which we fit the ARMA(2,1) $(1-1.583B+.899B^2)(X_t - 20.038) = (1-.929B)a_t$, where $\hat{\sigma}_a^2 = .84$. Figure 6.14 also shows the forecasts for 30 steps beyond the end of the dataset. If we replace the call to **`fore.arma.wge`** in Example 6.12 with the following,

```
fore.21=fore.arma.wge(arma21,phi=c(1.583,-.899),theta=.929,n.ahead=30,
limits=TRUE,alpha=.05)
```

then we obtain the forecasts along with the 95% prediction limits shown in Figure 6.14.

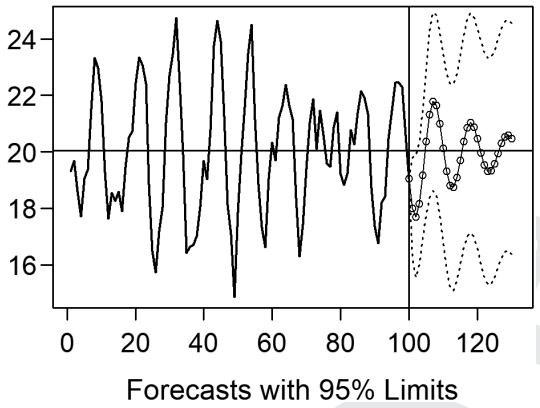

FIGURE 6.14 Data and forecasts in Figure 6.11(b) with 95% forecast limits.

The quantity

$$z_{1-\alpha/2}\,\hat{\sigma}_a\left\{\sum_{k=0}^{\ell-1}\hat{\psi}_k^2\right\}^{\frac{1}{2}} \tag{6.47}$$

is referred to as the half-width (or margin of error) of the prediction interval. As mentioned previously the half-widths increase as ℓ increases. Table 6.14 shows the forecasts and half-widths for the 95% prediction limits plotted in Figure 6.14 for a few values of ℓ. Recall that the ψ-weights can be found using ***tswge*** function **`psi.weights.wge`.** For the fitted ARMA(2,1) model we use the command

```
psi.weights.wge(phi=c(1.583,-.899),theta=.929,lag.max=4)
```

and find that $\hat{\psi}_k$, $k = 1,\ldots,4$ are given by $0.614, 0.048 - 0.477$, and -0.780, respectively. Of course, $\hat{\psi}_0 = 1$. In Example 6.12 we found that $\hat{\sigma}_a = .843$. Given the above information, the half-width, for $\ell = 1$, is given by

$$\hat{\sigma}_a\sqrt{1} = 1.96\sqrt{.843}\sqrt{1} = 1.96(.918) = 1.800,$$

for $\ell = 2$, is given by

$$1.96\,\hat{\sigma}_a\sqrt{1+\hat{\psi}_1^2} = 1.96\sqrt{.843}\sqrt{1+.654^2} = 1.96(.918)(1.1949) = 2.150,$$

for $\ell = 3$, is

$$1.96\hat{\sigma}_a\sqrt{\left(1+\hat{\psi}_1^2+\hat{\psi}_2^2\right)} = 1.96(.918)\sqrt{1+.654^2+.136^2}$$

$$= 1.96(.918)(1.2026) = 2.164.$$

and for $\ell = 4$, is

$$1.96\,\hat{\sigma}_a\sqrt{\left(1+\hat{\psi}_1^2+\hat{\psi}_2^2+\hat{\psi}_3^2\right)} = 1.96(.918)\sqrt{1+.654^2+.136^2+.372^2}$$

$$= 1.96(.9165)(1.2589) = 2.265.$$

Table 6.14 shows forecasts and half-widths for $\ell = 1,\ldots,4$. Notice that for the calculation of the half-width for $\ell = 4$, we did not require $\hat{\psi}_4$ because the sum of squared ψ-weights only goes to $\ell - 1$. Based on the table we see that the forecast for X_{104} is 20.712, and the 95% forecast limits for X_{104} are $\hat{X}_{100}(4) \pm 2.908$. We expect X_{104} to fall somewhere in the interval 19.177±2.262, that is, between $19.182 - 2.265 = 16.917$ and $19.182 + 2.265 = 21.447$.

Also, Table 6.14 shows forecasts and half-widths for 10, 20, and 50 steps ahead using $\alpha = .05$. Note the following. Theorem 5.4 states that the variance of the GLP $X_t = \sum_{k=0}^{\infty} \psi_k a_{t-k}$ is given by $\sigma_X^2 = \sigma_a^2 \sum_{k=0}^{\infty} \psi_k^2$, which is finite for stationary processes. Consequently, the sum in (6.47) does not increase without bound, and in fact

$$\hat{\sigma}_a \left\{ \sum_{k=0}^{\ell-1} \hat{\psi}_k^2 \right\}^{\frac{1}{2}} \rightarrow \hat{\sigma}_a \left\{ \sum_{k=0}^{\infty} \hat{\psi}_k^2 \right\}^{\frac{1}{2}} = \hat{\sigma}_X, \tag{6.48}$$

So that the half-width in (6.47) converges to $1.96\hat{\sigma}_X$.

For the fitted model $(1 - 1.583B + .899B^2)(X_t - 20.038) = (1 - .929B)a_t$, where $\hat{\sigma}_a^2 = .843$ and using ***tswge*** command

```
v=true.arma.aut.wge(phi=c(1.583,-.899),theta=.929,vara=.8426)
```

the process variance of the fitted model, $\hat{\sigma}_X^2$, is 4.5242 (`v$acv[1]`) so that $\hat{\sigma}_x = 2.1270$. The right-hand side of (6.48) is $1.96\sqrt{4.5242} = 4.169$. Notice that the half-widths in Table 6.14 seem to converge to 4.169, and the forecasts converge to the sample mean, $\bar{x} = 20.038$ as expected.

TABLE 6.14 Forecasts and 95% Prediction Interval Half-Widths for the ARMA(2,1) Realization plotted in Figure 6.14.

ℓ	*FORECAST*	*HALF-WIDTH*
1	17.993	1.800
2	17.693	2.150
3	18.164	2.164
4	19.182	2.265
⋮		
10	20.133	3.313
20	20.480	3.894
50	20.212	4.160
100	20.028	4.169
500	20.038	4.169

Example 6.14 Sunspot Forecasts with 95% Prediction Limits

In Example 6.11 we used the following code to find forecasts for the sunspot data using the AR(9) model in (5.36).

```
data(sunspot2.0)
est.ss=est.ar.wge(sunspot2.0,p=9)
```

In that example we found and plotted forecasts for the next 30 years. We modify the call to

```
forecast.arma.wge
```

so that the plot includes prediction limits.

```
fore.ss=fore.arma.wge(sunspot2.0,phi=est.ss$phi,n.ahead=30,limits=TRUE,alpha=.05)
```

Figures 6.14(a) and (b) show the forecasts with 95% limits. Notice that the forecast limits are getting a little wider for larger ℓ but the increase is not dramatic.

Key Point: Note that some of the lower limits for the sunspot forecasts are negative. Since the sunspot number cannot be negative, any negative lower limit should be converted to 0.

6.2.5.2 Lack of Symmetry

Although the sunspot data have been analyzed through the years using ARMA models, it clearly is not data from an ARMA model. The troughs are all very similar but there is much variation in peak heights, resulting in the asymmetric appearance in Figure 6.15. However, AR and ARMA models fit to the sunspot data, such as the AR(9) model we have examined, do not have realizations with this type of asymmetric appearance. Tong (1990) and others have investigated the use of nonlinear models to model these data. Recall that we encountered a similar problem with the lynx dataset and solved the problem by taking logs. However, the annual sunspot number for 1810 is zero so we cannot take logs unless we, for example, add one to each sunspot number. Traditionally, the raw sunspot numbers have been analyzed despite their obvious lack of symmetry.

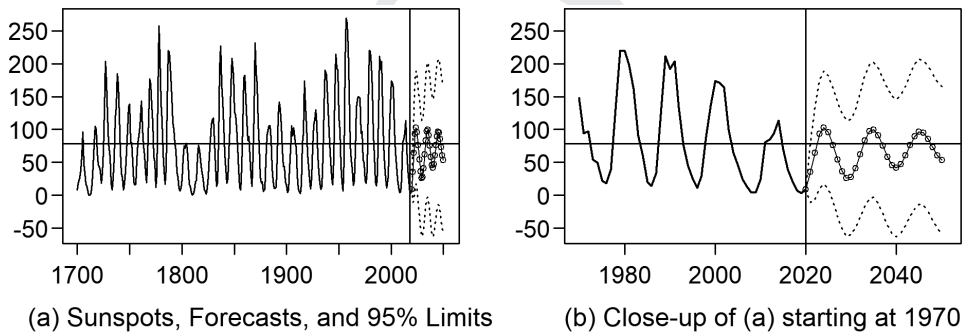

FIGURE 6.15 (a) `sunspot2.0` with 30 forecasts shown in Figure 6.10 along with 95% forecast limits (b) "close-up" forecasts and limits showing only sunspot data from 1970.

6.2.6 Assessing Forecast Performance

6.2.6.1 Obtaining "Forecasts" of Known Values

The forecast limits are useful for understanding the reliability of forecasts, that is, for assessing how precise we believe they are. However, the acid test of forecast performance is to compare forecasts with reality. One way to do this is to measure how well "actual values" can be predicted.

6.2.6.2 How "Good" Are the Forecasts?

For example, given a realization of length 100, you may want to determine how well your model predicts the last 20 values using $t_0 = 80$ as the forecast origin. In order to develop forecast models based on their performance in forecasting known values, we need tools for assessing "how good" the forecasts are. There are many ways to assess the quality, but the two methods we will discuss are the following:

(a) *Check actual values with the forecast limits*. If your model is appropriate and you have obtained, for example, 95% forecast limits, then approximately 95% of the actual values should fall within the forecast limits. If this condition is not satisfied and substantially fewer than 95% of actual values fall within the limits, then this is evidence that your model is not appropriate.

(b) *Quantify the difference between forecasts and actual values*. A useful measure of the quality of the forecasts is the root mean square error. That is, if $x_{t_0+1},\ldots,x_{t_0+k}$ are the actual values and $f_{t_0+1},\ldots,f_{t_0+k}$ are the forecasts based on the forecast model at origin t_0, then the *forecast error* at time t_{0+j}, for $1 \le j \le k$ is the "forecast value minus actual value", or $f_{t_0+j} - x_{t_0+j}$. A measure of how close the actual values and forecasts are to each other *as a group* is the root mean square error

$$RMSE = \sqrt{\frac{1}{k}\sum_{t=t_0+1}^{t_0+k}\left(f_t - x_t\right)^2}. \tag{6.49}$$

The MSE is the average of the squared forecast errors (removing the effect of the signs). The RMSE is the square root of this quantity which reverts back to the units of the problem. The RMSE is a measure of the "typical amount by which forecasts differ from actual values", that is, it measures the typical size of the forecast errors. Use of the RMSE to assess forecast results was introduced in Section 2.2.4.

For example: Given a model and a dataset of length 100, a measure of the RMSE associated with forecasting the last 20 data values using the model is

$$RMSE = \sqrt{\frac{1}{20}\sum_{t=81}^{100}\left(f_t - x_t\right)^2}. \tag{6.49}$$

Note: A measure related to the RMSE is the mean absolute deviation (MAD), defined by

$$MAD = \frac{1}{k}\sum_{t=t_0+1}^{t_0+k}\left|f_t - x_t\right|.$$

Although forecast performance will probably be *your gold standard* for assessing model appropriateness in practice, in Chapter 9 we will discuss other methods to judge how well your model fits the data.

6.2.6.3 Some Strategies for Using RMSE to Measure Forecast Performance

(1) Suppose we want to forecast, for example, the last 20 data values, by "ignoring" the last 20 values. We could develop the forecast function based on the first $n-20$ data values, and then forecast values $n-19, n-18,\ldots,n$. Note that if this procedure is repeated for various forecast origins then it requires refitting a new model for *each* new forecast origin using all data up to and including each forecast origin. For example, to forecast the last 20 data values, AIC may choose an ARMA(2,1) model fit to the first $n-20$ data values. However, if forecasting the last 40 data values, AIC may select an AR(4) model fit to the first $n-40$ data values.

(2) A simplification of (1) is to use AIC to choose a model *order*, say AR(4) for the entire dataset, and then refit models for each forecast origin, t_0, using an AR(4) model fit to the realization $x_1, x_2,\ldots,x_{t_0}$.

(3) A further simplification is to fit a model using the entire dataset, and then use the associated forecast function to forecast from each forecast origin of interest.

Key Point: All of these strategies are designed to help answer the fundamental question at hand. That is, ***"How well will the model forecast future values when they become available?"***

The following rather lengthy example illustrates the ideas discussed above.

Example 6.15 Forecasting the sunspot data

In Example 6.8 we fit an AR(9) model to the data in file **`sunspot2.0`** which contains sunspot data from 1700–2020 using the new numbering system (Clette et al., 2016). Figures 6.15(a)–(d) show forecasts with forecast origins, $t_0 = 2010, 2000, 1990,$ and 1980, respectively. We have used strategy (2) above and we fit an AR(9) model to the available data up to and including the forecast origin. Notice that 2010=1700+310, …, 1980=1700+280. Because **`sunspot2.0`** is sequenced on the integers 1 through 319, the four models can be obtained using the ***tswge*** commands

```
data(sunspot2.0)
a=est.arma.wge(sunspot2.0[1:311],p=9)
b=est.arma.wge(sunspot2.0[1:301],p=9)
c=est.arma.wge(sunspot2.0[1:291],p=9)
d=est.arma.wge(sunspot2.0[1:281],p=9)
```

We leave it to the reader to check, but the commands above produce four very similar models and factor tables. From all forecast origins it is seen that the forecasts have a damping sinusoidal appearance with a period length of about 10.5 years.

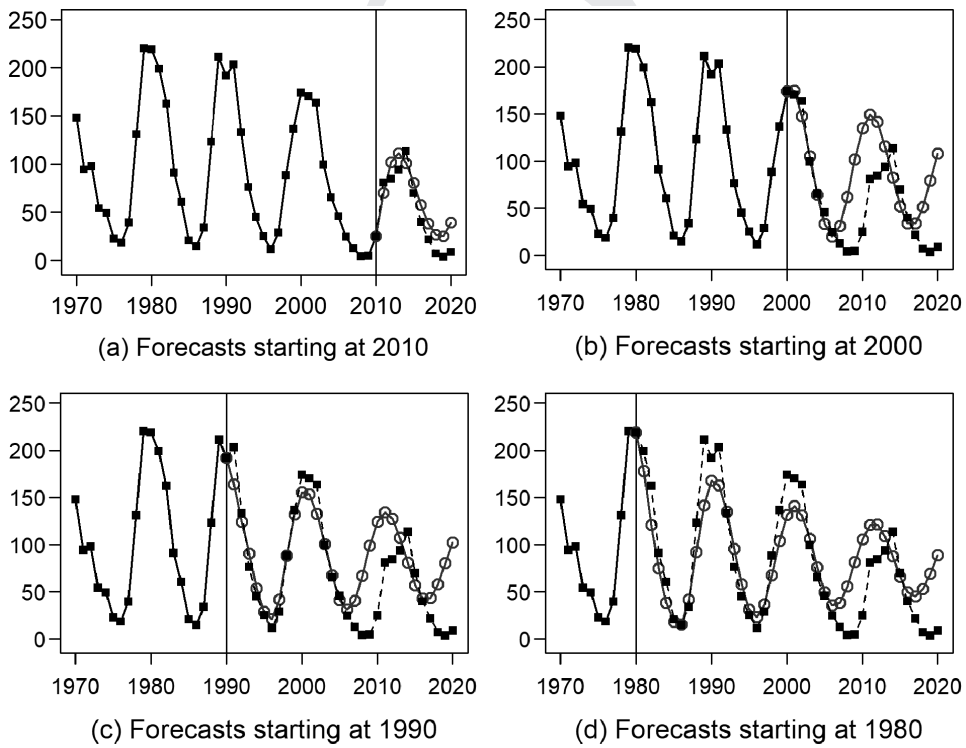

FIGURE 6.16 Comparing forecasts (open circles) with actual values (solid squares) for forecast origins (a) 2010, (b) 2000, (c) 1990, and (d) 1980 using the sunspot data in `sunspot2.0`.

Figure 6.16 illustrates an important issue related to the sunspot data that was discussed in Chapter 1. That is, while on the average, cycle lengths tend to be about 11 years, in reality each actual sunspot cycle is not of exactly the same length. For example, the cycle starting at the peak at 1980 is of length close to 11 years, while the cycle beginning at 2000 is of length about 12 years. Forecasts above starting at origin 2010 do an excellent job of tracking the actual behavior while forecasts from 2000 get "off cycle" at about 2007 and as a whole are disappointing. Forecasts from origin 1990 are quite good for the years 1991–2006, but again get off cycle at about 2007. Finally, forecasts from 1980 stay "in cycle" for the 26 years from 1981–2006. However, these forecasts tend to underestimate the cycle peaks. In the following we will quantify the forecast performance described above.

(1) Do the Actual Values Tend to Stay with the Prediction Limits?

We first evaluate the performance of these forecasts by determining whether the actual values stay within the prediction limits. That is, for the 95% limits, we should be concerned if fewer than 95% of the actual values stay within the limits for a given forecast origin. Figure 6.17(a)–(d) shows the forecasts in Figure 6.16 along with 95% prediction limits. It can be seen that in all cases, the actual values are within the prediction limits. Note that for forecasts with forecast origin 1980, the actual value is close to the upper limit at 1989. However, in this case the actual value is 211.1 and the upper limit is 231.85. Again note that many of the lower limits were negative, and in these cases it would have been appropriate to convert negative lower limits to zero. Based on the forecast limits we see no reason for concern.

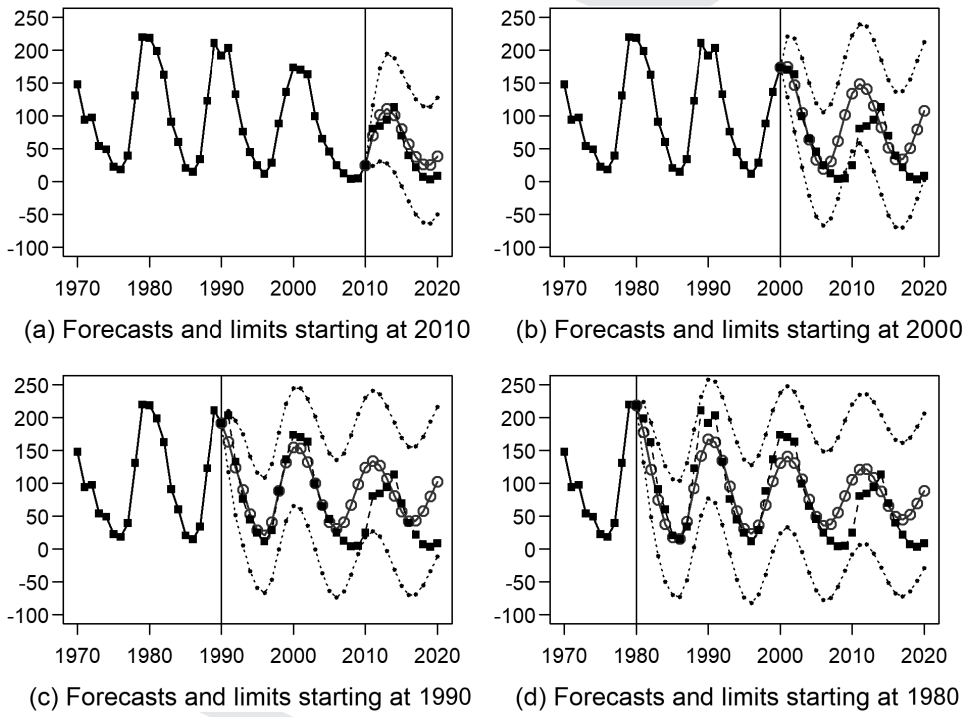

FIGURE 6.17 Forecasts in Figure 6.16 showing associated 95% prediction limits.

(2) Quantifying Forecast Performance

The fact that actual values stay within the 95% limits is not necessarily an indication that the forecasts are "good" or the best we can do. For example, the forecasts for 2007–2020 beginning at forecast origins 1980, 1990, and 2000 were "off cycle". This is still a cause for concern even though the forecasts stay within the

limits. Table 6.15 shows the actual values and associated forecasts for years 2011–2020 based on forecast origin 2010. The last column shows the forecast errors. Recall that the forecasts in Figures 6.15 and 6.16 were using AR(9) models fit to data up to and including the forecast origin.

TABLE 6.15 Actual Values and Forecasts from Forecast Origin 2010

FORECAST YEAR	*ACTUAL*	*FORECAST*	*FORECAST ERROR*
2011	80.8	70.3	–10.5
2012	84.5	102.0	17.5
2013	94.0	111.2	17.2
2014	113.3	101.0	12.3
2015	69.8	80.5	10.7
2016	39.8	57.5	17.7
2017	21.7	38.1	16.4
2018	7.0	26.5	19.5

In order to quantify the "typical size of the forecast errors ignoring signs" from the four forecast origins, we calculate the associated forecast RMSEs. For the above data the RMSE is 15.6 and the MAD is 15.2, and it is clear that both measures give a "typical" size of the forecast error.

In Table 6.16 we show the RMSEs associated with forecasts from forecast origins 2010, 2000, 1990, and 1980. The RMSE for origin 2010 is 15.6 as mentioned above.

TABLE 6.16 RMSE for Forecasts in Figures 6.15 and 6.16

FORECAST ORIGIN	*RMSE*
2010	15.6
2000	45.6
1990	36.2
1980	32.3

As observed earlier, the forecasts as a group from origin 2010 were quite good and the RMSE is the smallest of those tabled above. The "disappointing" forecasts from origin 2000 resulted in the largest RMSE, 45.6. Forecasts from origins 1990 and 1980 were satisfactory for early lags before getting off cycle at 2007, and their RMSEs are smaller than those for origin 2000.

As mentioned, the RMSE values give us a measurement of the size of the forecast errors and are useful for informing us concerning whether the forecasts are "doing well enough for our purposes". Another use of the RMSE is to compare forecasts from two or more competing models. In our case AIC measures were consistent in selecting an AR(9) model for forecasting from the origins used in the above analyses. Box, Jenkins, and Reinsel (2008) analyze a shorter sunspot dataset based on the previous counting measure. They chose an AR(2) model based on the partial autocorrelations.

(a) Using an AR(2) Model for Forecasting

In Examples 6.12 and 6.14 we examined forecasts from the AR(9) model fit to the **`sunspot2.0`** dataset. At this point we will consider using AR(2) fits for forecasting. In Figure 6.15 we showed forecasts for the next 30 years obtained using the AR(9) model along with 95% prediction limits. Figure 6.18 is the analogous figure based on an AR(2) fit.

The following code fits an AR(2) model to the **`sunspot2.0`** data.

```
data(sunspot2.0)
est.ss=est.ar.wge(sunspot2.0,p=2)
```

The resulting model is

$$\left(1-1.38B+.69B^2\right)\left(X_t-78.97\right)=a_t, \tag{6.50}$$

with $\hat{\sigma}_a^2 = 659.39$. The single row factor table is given in Table 6.17 where it can be seen that there is a single factor, and, not surprisingly, an associated system frequency (.094) that corresponds to a cycle length of about 10.5 years. However, the associated roots are not as close to the unit circle as they are in the AR(9) model.

TABLE 6.17 Factor Table AR(2) Fit to sunspot2.0 Data

AR-FACTOR	*ROOTS*	$\lvert r\rvert^{-1}$	f_0
$1-1.38B+.69.0B^2$	$1.0\pm.67i$	.83	.094

We use the following code to find the forecasts and prediction limits using an AR(2) model.

```
fore.ss=fore.arma.wge(sunspot2.0,phi=est.ss$phi,n.ahead=30,limits=TRUE)
```

The primary difference between forecasts from the AR(9) and AR(2) models is that the AR(9) model picks up the 10.5-year cycle and seemingly does a better job forecasting it to continue into the future. On the other hand, after about 10 steps ahead, the AR(2) model essentially forecasts the series mean.

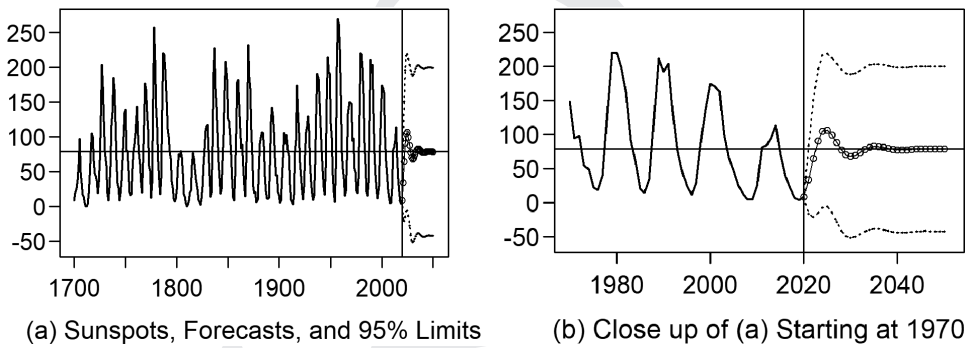

FIGURE 6.18 (a) `sunspot2.0` with 30 forecasts from AR(2) model along with 95% forecast limits (b) "close-up" forecasts and limits showing only sunspot data from 1970.

Figure 6.19 shows the AR(9) forecasts in Figures 6.15 and 6.16 as open circles, AR(2) forecasts with "×," and actual values with solid squares. In these figures we see that again, the AR(2) forecasts are rather "boring" in that they tend quickly to the mean and seem to minimize the prominent 10−11 year cycle. Table 6.18 shows the RMSE for the two models based on four forecast origins. Surprisingly, at forecast origin 2000, the RMSE for the AR(2) forecasts is smaller than for the AR(9). The lesson is that the AR(9) forecasts "boldly" predicted a cyclic behavior that was off cycle, suggesting that in this particular case, it was better to be very conservative. Even though the AR(9) forecasts also got "off-cycle" at 2007 for forecast origins 1990 and 1980, the fact that the AR(9) forecasts tracked well until 2007 caused the AR(9) forecasts to be preferable.

Key Points

1. As discussed in Chapter 1, the approximate 11-year cycle in the sunspot data is not well understood. For example, scientists do not know when a future cycle may be shorter or longer than is typical. In the presence of this type of cyclic behavior, it may be a mistake to be too confident about predictions based on a certain cycle length to continue. Notice that even the AR(9) forecasts become more "conservative" at longer cycle lengths (and will eventually tend to the mean) providing protection from "overconfidence".
2. We mentioned in Chapter 1 that some cycles, such as monthly temperature patterns at a certain location, are explainable. In this case the cycle length is based on the Earth's rotation around the sun. In such cases, these are referred to as "cyclic data" with fixed cycle length. If the data relate to the calendar year, then they are referred to as "seasonal" data. For fixed cycle length data, forecasts can be made more "boldly" into the future. In fact, this type of cyclic behavior may be indicative of the fact that a signal-plus-noise model is appropriate. See Chapter 8.

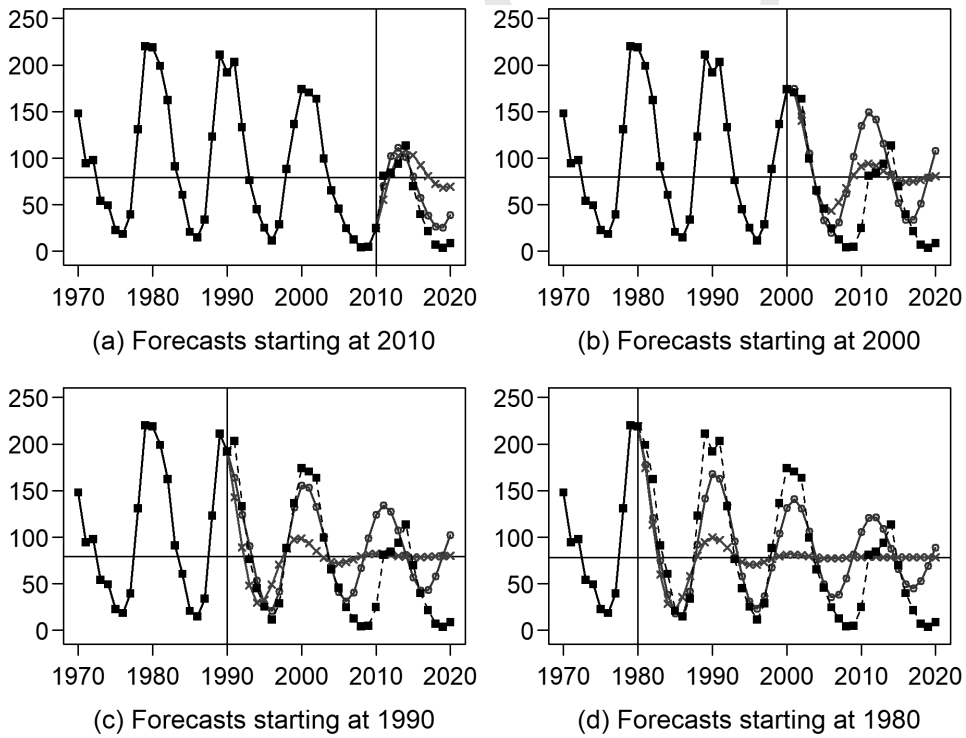

FIGURE 6.19 Comparing AR(9) forecasts (open circles) and AR(2) forecasts (×)with actual values (solid squares) for forecast origins (a) 2010, (b) 2000, (c) 1990, and (d) 1980 using the sunspot data in `sunspot2.0`.

TABLE 6.18 RMSE for Forecasts Based on AR(9) and AR(2) Fits

FORECAST ORIGIN	*RMSE AR(9)*	*RMSE AR(2)*
2010	18.3	45.0
2000	51.4	43.7
1990	41.4	49.6
1980	35.5	55.3

(3) A Rolling Window RMSE Approach

In the previous example, the AR(9) with a forecast origin of 2010 (and time horizon of 10) was favored over the AR(2) model based on the lower RMSE. It is important to note that this assessment is based on only the last 10 observations of the **`sunspot2.0`** series. What if this model were evaluated on a different 10-year window of sunspot data? Would the AR(9) still be preferred? To investigate this question, we consider the performance of each of these models during a 10-year window near the *beginning* of the series.

Question: Could we evaluate the models on the first 10 years of recorded sunspot data, that is, 1700–1709?

Not quite, right? To forecast sunspots for the year 1700, the AR(2) model would require knowing the sunspot data for 1698 and 1699, and the AR(9) would require the data from 1691–1699. However, these data are unavailable in **`sunspot2.0`**. Consequently, given this particular dataset, the first year that the AR(2) model can forecast is 1702, while the first year that the AR(9) model can forecast is 1709. Furthermore, the *backcasting* procedure used to obtain better estimates of the residuals requires one additional observation to calculate forecasts from these models. For this reason, the earliest 10-year window for which these models can provide forecasts is 1703–1712 and 1710–1719 for the AR(2) and AR(9), respectively. For the purpose of comparison, consider the AR(2) and AR(9) forecasts of sunspots from 1710–1719.

Figure 6.x (a) and (b) below displays the performance of these forecasts with their corresponding RMSEs.

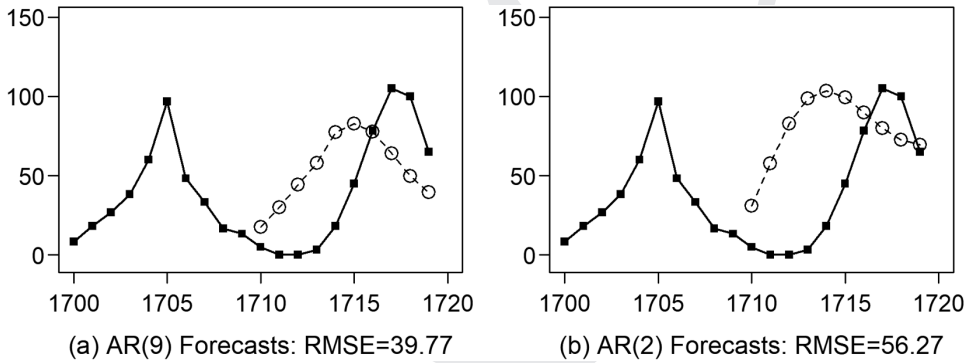

FIGURE 6.20 Horizon 10 forecasts from 1710–1719 using (a) an AR(9) and (b) an AR(2) Model

The AR(2) forecasts produce an RMSE of 56.27 compared to an RMSE of 39.77 for the AR(9) forecasts. As was the case for the forecast origin beginning in 2010, the AR(9) again outperforms the AR(2) with a forecast origin beginning in 1710.

A natural question is, "What about a third 10-year window?" which would then lead to, "What about a fourth 10-year window?", and so on. Table 6.19 displays the two 10-year windows we have already assessed (in italics), along with three additional randomly selected 10-year windows from the series with their corresponding RMSEs (ordered chronologically):

TABLE 6.19 AR(9) and AR(2) RMSEs for a horizon of 10 for five different "windows" in the sunpot2.0 series.

WINDOW	*RMSE AR(9)*	*RMSE AR(2)*
1710 – 1719	*39.77*	*56.27*
1790 – 1799	86.97	54.73
1845 – 1854	32.11	54.33
1986 – 1995	35.05	68.97
2011 – 2020	*18.3*[1]	*45.01*

[1] Note that the RMSEs for the last window in Table 6.19 for the AR(9) and AR(2) are calculated using the mean of the **`sunspot2.0`** data only up to year 2010 to match the output

TABLE 6.19 (*continued*)

seen in Table 6.18. The other RMSEs are calculated using the mean of the full data so that forecasts will approach the same sample mean. This doesn't make a big difference for the window starting at 2011; (values using the full mean are 17.65 and 44.53 respectively). However, it has the potential to make a very impactful difference as the data before the current window becomes sparse. For example, the sample mean for the first window in Table 6.19 would have been calculated using only 10 of the 321 values (1700–1709) in the **`sunspot2.0`** series.

We note that the AR(9) had a lower RMSE in all but one of the windows in Table 6.19. The obvious question is, "What if the models were compared only using the window from 1790–1799?" This table shows that the choice of model, with respect to the RMSE, is largely dependent on the choice of the window.

To address the dependency on the window, one might choose to use the average of the five windows for each model to obtain a more stable assessment. This would yield an average RMSE for the AR(9) model of 42.44 compared to 55.86 for the AR(2) model. The AR(9) model would again be favored. However, this time the comparison would be using information from all five windows thus reducing the impact on any single window.

The natural progression of this idea is to then include as many windows as the series will allow. For the AR(9) and the **`sunspot2.0`** data, this would entail starting with the first possible window (1710−1719) and calculating the RMSE and then shifting ("rolling") the window ahead one year and calculating the RMSE for the second window (1711−1720). This process of rolling the window ahead one year and calculating the RMSE continues until the RMSE is calculated on the final window (2011–2020). This is the strategy employed by the "*Rolling Window RMSE*". The rolling window approach will result in the comparison of 302 windows (and thus 302 RMSEs) using the AR(9) model on the **`sunspot2.0`** data. Using ***tswge*** function roll.win.rmse.wge, the rolling window RMSE (rwRMSE) is calculated by taking the average of these 302 RMSEs which is found to be 37.67. A tabular illustration of this process for the AR(9) is displayed in the left-hand side of Table 6.20.

TABLE 6.20 Illustration of the Windows in the Rolling Window RMSE Calculation for a Horizon of 10 for the AR(9) and AR(2) models. The Average of the RMSEs is in the Last Row and is the "Rolling Window RMSE" (rwRMSE)

AR(9)			*AR(2)*		
WINDOW #	*WINDOW*	*RMSE*	*WINDOW #*	*WINDOW*	*RMSE*
1	1710 – 1719	39.77	1	1703 – 1719	56.27
2	1711 – 1720	36.28	2	1704 – 1720	45.52
3	1712 – 1721	30.81	3	1705 – 1721	35.68
	…	…		…	…
166	1875 – 1884	50.52	166	1875 – 1884	33.88
167	1876 – 1885	51.85	167	1876 – 1885	27.84
168	1877 – 1886	53.23	168	1877 – 1886	32.12
	…	…		…	…
300	2009 – 2018	35.15	306	2009 – 2018	31.78
301	2010 – 2019	19.28	307	2010 – 2019	39.37
302	2011 – 2020	17.65[1]	308	2011 – 2020	44.53
	rwRMSE	**37.67**			**48.81**

[1] Differs from value in Table 6.19 as per the discussion in footnote 11.

Note in the right half of Table 6.20, an analogous process is implemented for the AR(2) in which, because of the smaller order ($p=2$), there are now 308 possible windows across the series ranging from starting points of 1703 to 2011. The rwRMSE for the AR(2) is 48.61.

It can again be seen that at different windows in the series, the models may trade in achieving the smallest RMSE. This dependence is now *eliminated* by averaging across *all* possible windows. The AR(9) is ultimately found to be favored with respect to this measure because the AR(9) rwRMSE of 37.67 is smaller than the 48.81 produced by the AR(2).

The function **`roll.win.rmse.wge`** in ***tswge*** will calculate the rwRMSE given the data, the model, and the horizon. In the code and output below, the AR(9) model is first estimated for the **`sunspot2.0`** series as it was in Example 6.12 and the subsequent model is then used in the call to **`roll.win.rmse.wge.`**

```
est9 = est.arma.wge(sunspot2.0, p = 9, factor = FALSE)
roll.win.rmse.wge(sunspot2.0, phis = est9$phi, h = 10)
"Please Hold For a Moment, TSWGE is processing the Rolling Window RMSE with 302
windows."
"The Summary Statistics for the Rolling Window RMSE Are:"
Min. 1st Qu. Median  Mean 3rd Qu.  Max.
8.399 23.274 35.040 37.666 47.226 96.858
"The Rolling Window RMSE is: 37.666"
```

Examining the output above, we see that the first line shows that 302 windows were used in the calculation of the rwRMSE which in the last line is shown to be 37.67 (as reported in Table 6.20). Since the rwRMSE in this case is calculated from 302 RMSEs, it may also be useful to know the spread of the RMSEs for the individual windows. The Tukey five-number summary is provided by **`roll.win.rmse.wge`** to reflect the spread and distribution of these RMSEs. The **`roll.win.rmse.wge`** function also outputs a histogram of the RMSEs shown in Table 6.20. Calculating the rwRMSE for the AR(2) is left as an exercise (Problem 6.11).

QR 6.10 Forecast Assessment Methods

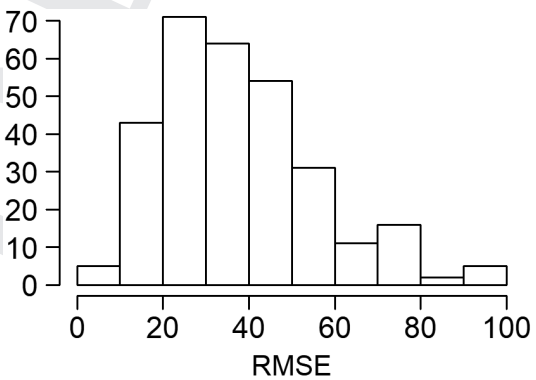

FIGURE 6.21 Histogram of RMSEs for AR(9) fits to all 302 horizon 10 windows.

6.3 CONCLUDING REMARKS

In the previous chapter, the ARMA(*p,q*) model was defined and various properties of AR(*p*), MA(*q*), and ARMA(*p,q*) models were established. It was shown that, for example, realizations from AR(1) models exhibit noticeably different behavior than realizations from AR(2) models. Therefore, given the importance of the order of a time series model, a key objective of this chapter was to learn how to identify the order *p* and *q* of an ARMA(p,q) model.

Three long-standing, classical methods to estimate the optimal *p* and *q* were presented: MLE, Burg, and Yule-Walker. Each of the three methods has various strengths and weaknesses, but ***tswge*** can easily calculate all three so the analyst can have as much as information as possible to make a decision regarding model order. Parsimony is to be considered, so the question is whether higher orders of p and q are worth the amount of observed gain in model performance.

Finally, the methodology of calculating forecasts from ARMA(*p,q*) models was discussed in detail and examples were given to illustrate the process. The assessment of forecasts can be measured by root mean square error (RMSE) or the rolling window RMSE.

APPENDIX 6A

TSWGE FUNCTIONS

(a) **`aic.wge(x,p=pmin:P,q=qmin:Q,type='aic')`** is a function that finds the best ARMA(*p,q*) model using ML estimation within the *p* and *q* ranges specified in the call statement. The function can base the model identification on AIC, AICC, or BIC.

`x` is a vector containing the realization to be analyzed.
`p=pmin:P` specifies the range of values for the autoregressive order *p* to be used in the search for the best model. (default is **`p=0:5`** and it is common for **`pmin`** to be 0)
`type` specifies the identification criterion type to be used. Choices are type= **`'aic','bic'`**, or **`'aicc'`**. (default is **`'aic'`**)

(b) **`aic.ar.wge(x,p=pmin:P,type='aic',method='mle')`** is a function that finds the best AR(*p*) model within the *p* range specified in the call statement. The function can base the model identification on AIC, AICC, or BIC and can be used with estimation methods MLE, Burg, and Yule-Walker.

`x` is a vector containing the realization to be analyzed.
`p=pmin:P` specifies the range of values for the autoregressive order *p* to be used in the search for the best model. (default is **`p=0:5`** and it is common for **`pmin`** to be 0)
`type` specifies the identification criterion type to be used. Choices are type= **`'aic','bic'`**, or **`'aicc'`**. (default is **`'aic'`**)
`method` specifies the estimation method used. Choices are method= **`'mle','burg'`**, or **`'yw'`**. (default is **`'mle'`**)

(c) **`aic5.wge(x,p=pmin:pmax,q=qmin:qmax,type,method)`** finds the five best AR(*p*) models within the **`p`** range in the call statement using the selection criterion specified by **`type`** and estimation criterion indicated by **`method`**.

x is a vector containing the realization to be analyzed.
p=pmin:P specifies the range of values for the autoregressive order p to be used in the search for the best model. (default is **p=0:5** and it is common for **pmin** to be 0)
type specifies the identification criterion type to be used. Choices are type= **'aic','bic',** or **'aicc'**. (default is **'aic'**)

Note: The program outputs the model orders of the top five models (and associated criterion values) within the range of p and q specified in the function statement.

(d) **aic5.ar.wge(x,p=pmin:P,type='aic',method='mle')** finds the five best AR(p) models within the **p** range in the call statement using the selection criterion specified by **type** and estimation criterion indicated by **method**.

x is a vector containing the realization to be analyzed.
p=pmin:P specifies the range of values for the autoregressive order p to be used in the search for the best model. (default is **p=0:5** and it is common for **pmin** to be 0)
type specifies the identification criterion type to be used. Choices are type= **'aic','bic'**, or **'aicc'**. (default is **'aic'**)
method specifies the estimation method used. Choices are type= **'mle','burg'**, or **'yw'**. (default is **'mle'**)

Note: The program outputs the model orders of the top five models (and associated criterion values) within the range of p specified in the function statement.

(e) **est.arma.wge(x, p=0,q=0, factor=TRUE,method='mle')** obtains parameter estimates of a stationary ARMA(p,q) that is fit to the data in the vector **x**.
x is a vector containing the realization to be analyzed,
p is the order of the AR part
q is the order of the MA part
factor is a logical variable, and if **factor=TRUE** (default) a factor table is printed for the estimated model.

(f) **est.ar.wge(x, p=2,factor=TRUE,method='mle')** obtains parameter estimates of a stationary autoregressive model of order **p** that is fit to the data in the vector **x**.
x is a vector containing the realization to be
p is the order of the AR model (default is **p=2)**
factor is a logical variable, and if **factor=TRUE** (default) a factor table is printed for the estimated model.
method specifies the method used to estimate the parameters: either **'4'** (default), **'burg'**, or **'yw'**

(g) f**ore.arma.wge(x,phi,theta,n.ahead,lastn,plot,alpha,limits)** forecasts **n.ahead** steps ahead for stationary ARMA models (of which AR models are a special case with **theta=0**). The forecasts and forecast limits can optionally be plotted. You can calculate and plot forecasts **n.ahead** steps beyond the end of the series, or you can forecast the last n**.ahead** data values (if **lastn=TRUE**).

Note: The components of phi and/or theta can be obtained by first using **mult.wge** if the models are given in factored form.

Note: The ARMA model can be a given/known model, but the most common usage will be to fit an ARMA(p,q) model to the data and base the forecasts on the fitted model.

x is a vector containing the time series realization $x_t, t = 1,\ldots,n$

phi is a vector of autoregressive coefficients (default=0)
theta is a vector of moving average coefficients (default=0)
n.ahead specifies the number of steps ahead you want to forecast (default=2)
lastn is a logical variable (default=**FALSE**).

If **lastn=FALSE** (default) then the forecasts are for $x(n+1), x(n+2), \ldots, x(n+n.ahead)$ where n is the length of the realization.
If **lastn=TRUE**, then the program forecasts the last **n.ahead** values in the realization.
plot is a logical variable. (default=**TRUE**)
If **plot=TRUE** then the forecasts will be plotted along with the realization.
If **plot=FALSE** then no plots are output.
alpha specifies the significance levels of the prediction limits. (default=.05)
limits is a logical variable. (default=**TRUE**)
If **limits=TRUE** then the (1-**alpha**)×100% prediction limits will be plotted along with the forecasts.

(h) **pacf.wge(x, lag.max=20,plot=TRUE,method)** calculates and optionally plots the estimated partial autocorrelations
x is a vector containing the realization to be
lag.max is the maximum lag at which the PACF is calcul;ated. (default is **p=20)**
method specifies the method used to estimate the parameters: either **'mle'**, **'burg'**, or **'yw'**
plot is a logical variable. (default=**TRUE**)
If **plot=TRUE** then the PACFs will be plotted along with 95% limit lines.
If **plot=FALSE** then no plots are output but the PACFs are contained in the output vector **$pacf**.

(i) **roll.win.rmse.wge(series,horizon=1,s=0,d=0,phi=0,theta=0)** is a function that creates as many "windows" as is possible with the data vector **series** and calculates an RMSE for each window. The resulting "rolling window RMSE" is the average of the individual RMSEs from each window.

series The data
horizon The number of steps ahead to be forecasted.
s Order of the seasonal difference, default=1
d Order of the difference
phi Vector of AR coefficients
theta Vector of MA coefficients
Output
rwRMSE The average of the individual RMSEs of each window
numwindows The number of windows
horizon The number of observations ahead to be forecasted.
s Order of the seasonal difference, default=1
d Order of the difference
phi Vector of AR coefficients
theta Vector of MA coefficients
RMSEs An array of the calculated RMSEs for each window.

PROBLEMS

6.1 (a) Factor the Yule-Walker and the Burg estimates fit to the dataset **A** shown in Figure 5.19(a) and compare them with the factors of those of the true model.

(b) Generate realization from the ML, Burg, and YW models *fit* to dataset **A**. Plot the realizations, sample autocorrelations, and Parzen spectral density estimates for the three realizations. Compare and contrast these with the plots in Figures 5.19(a)−(c) for the original data generated from the "true model".

6.2 The second choice of a model for the **sunspot2.0** data using **aic5.ar.wge** is an AR(10) model. Use factor tables to compare the AR(9) and AR(10) models and explain which model you would choose and why.

6.3 Generate a realization using the command

```
x=gen.arma.wge(n=150,phi=c(2.6,-3.34,2.46,-.9024),sn=3233)
```

Use aic.ar.wge(x,p=0:10,type= 'a',method= 'b') to identify the order of the selected model letting a=aic, bic, and **aicc** and **b=mle**, burg, and **yw**.

(a) Use all nine combinations of "type" and "method".

(b) If an AR(4) model is not selected, examine the factor table of the estimated model and compare it with the factor table for the model from which the realization is generated, **phi=c(2.6,-3.34,2.46,-**

6.4 Consider the dataset in Example 5.9 for which the Yule-Walker estimates are poor.

```
ar3=gen.arma.wge(n=100,phi=c(.9,-.9,.81),sn=11)
```

Use **aic5.ar.wge** using the Burg and Yule-Walker methods with p ranging from 0 to 10.

(a) What is the top model selected by Burg?

(b) What is the top model selected by Yule-Walker?

(c) Factor each of the models obtained in (a) and (b) and compare the factors with those of the actual AR(4) model.

6.5 Use **aic5.ar.wge** using AIC and ML estimates to find the top model for the following ***tswge*** datasets:

(a) **dfw.month**

(b) **dfw.yr**

(c) **wtcrude2020**

6.6 Use Base R function **pacf** to identify the order of the AR model you would fit to the datasets in Problem 6.5. Additionally, use pacf on the:

(d) **x500=gen.arma.wge(n=500,phi=c(.13,1.4414,-.0326,-.8865),sn=9310)**

(e) **ar3** in Problem 6.6

6.7 Calculate the data **logss**=log(**sunspot2.0**+10)

(a) Use **plotts.sample.wge** on **sunspot2.0** and **logss**. Compare and contrast the results. What has been the effect of taking the logarithm?

(b) Why didn't we calculate log(**sunspot2.0**)

(c) Use AIC and BIC to select model orders for the **logss** data letting **p=0:10** and **q=0:4**. How do these model orders compare with AIC and BIC fits to the raw sunspot2.0 data?

(d) Find ML estimates for the models selected by AIC for the raw sunspot and **logss** data.

(e) Use the ML model fit to **logss** to forecast the last 10 and 20 years of **logss** data. Plot the forecasts.

(f) Find the RMSE and MAD for the forecasts in (e).

6.8 Generate the following realization:
Consider the model: $\left(1-B+.9B^2\right)\left(1-.95B\right)\left(1+.9B\right)\left(X_t-100\right)=a_t$ where $\sigma_a^2=1$.

(a) Use **mult.wge** (see Appendix 5A) to find the coefficients of the resulting 4th order model.

(b) Generate a realization of length $n=200$ from this model.

(c) Use **aic.ar.wge** or **aic5.ar.wge** (and using AIC and Burg estimates) to fit a model to your realization.
Did AIC select $p=4$?

Factor your model and discuss how its factors compare with those of the true model.

(d) Use your fitted model to
   (i) Forecast 20 time periods beyond the end of the realization and show the 80% prediction limits.
   (ii) Forecast the last 20 time periods of the data and find the RMSE and MD for these forecasts.

6.9 Show that the commands below produce four similar models and factor tables.

```
data(sunspot2.0)
par(mfrow=c(2,2),mar=c(4,2,1,1))
a=est.ar.wge(sunspot2.0[1:311],p=9)
b=est.ar.wge(sunspot2.0[1:301],p=9)
c=est.ar.wge(sunspot2.0[1:291],p=9)
d=est.ar.wge(sunspot2.0[1:281],p=9)
```

6.10 In Example 6.12 and in the Rolling Window RMSE example, it was found that the ARMA(2,1) had a lower AIC and lower rolling window RMSE than the ARMA(3,1). Aic5.wge ranked these first and second respectively.

(a) Use aic5.wge() on the data from Example 6.12 to identify the top five models picked by AIC.th
(b) Find the rolling window RMSE for a horizon of 30 for each model.
(c) Do the models with the lowest AIC have the lowest rolling window RMSE?
(d) It turns out that the model with the 3rd ranked AIC has the lowest rolling window RMSE of the five models. What are your thoughts? How can this be?
(e) Finally, instead of generating 100 from the model in Example 6.12, generate 10000 observations from that model with the same seed (a much larger series with more information). Rerun aic5.wge to re-id the best p and q given the data. Compare the five models on AIC and rolling window RMSE. Do the models with the lowest AIC also have the lowest rolling window RMSE (again with a horizon of 30)?
(f) Conduct (d) again with the BIC.
(g) Now try all of the above analysis after generating 100,000 from the same model that was in Example 6.12.

6.11 Verify the rwRMSE shown in Table 6.20 for the AR(2) model.

6.12 In Table 6.16 it was shown that the AR(9) achieved a lower RMSE for all but one of the tested forecast origins. Calculate a rolling window RMSE for each horizon and compare to the results of Table

6.13 Use **`pacf.wge`** to select the "best" AR model for the following datasets:

(a) sunspot2.0
(b) log(10) lynx
(c) Example 6.3 data vector ar4
(d) DFW.month
(e) wtcrude2020

For each dataset plot the PACFs and "assess model order" using ML, Burg, and YW estimates.