10

Time Series Models: Further Topics

10.1 Seasonal ARIMA Models

Economic time series often exhibit strong seasonal variation. For example, an investor in mortgage-backed securities might be interested in predicting future housing starts, and these are usually much lower in the winter months compared to the rest of the year. Figure 10.1(a) is a time series plot of the logarithms of quarterly urban housing starts in Canada from the first quarter of 1960 to final quarter of 2001. The data are in the data set `Hstarts` in R's `Ecdat` package.

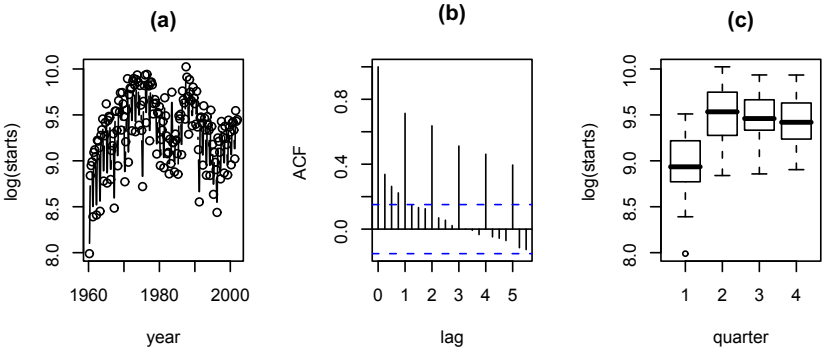

Fig. 10.1. *Logarithms of quarterly urban housing starts in Canada. (a) Time series plot. (b) ACF. (c) Boxplots by quarter.*

Figure 10.1 shows one and perhaps two types of nonstationarity: (1) There is strong seasonality, and (2) it unclear whether the seasonal subseries revert to a fixed mean and, if not, then this is a second type of nonstationarity because

D. Ruppert, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics,
DOI 10.1007/978-1-4419-7787-8_10, © Springer Science+Business Media, LLC 2011

the process is integrated. These effects can also be seen in the ACF plot in Figure 10.1(b). At lags that are a multiples of four, the autocorrelations are large, and decay slowly to zero. At other lags, the autocorrelations are smaller but also decay somewhat slowly. The boxplots in Figure 10.1(c) give us a better picture of the seasonal effects. Housing starts are much lower in the first quarter than other quarters, jump to a peak in the second quarter, and then drop off slightly in the last two quarters.

Other time series might have only seasonal nonstationarity. For example, monthly average temperatures in a city with a temperate climate will show a strong seasonal effect, but if we plot temperatures for any single month of the year, say July, we will see mean-reversion.

10.1.1 Seasonal and nonseasonal differencing

Nonseasonal differencing is the type of differencing that we have been using so far. The series Y_t is replaced by $\Delta Y_t = Y_t - Y_{t-1}$ if the differencing is first order, and so forth for higher-order differencing. Nonseasonal differencing does not remove seasonal nonstationarity and does not alone create a stationary series; see the top row of Figure 10.2.

To remove seasonal nonstationary, one uses seasonal differencing. Let s be the period. For example, $s = 4$ for quarterly data and $s = 12$ for monthly data. Define $\Delta_s = 1 - B^s$ so that $\Delta_s Y_t = Y_t - Y_{t-s}$.

Be careful to distinguish between $\Delta_s = 1 - B^s$ and $\Delta^s = (1-B)^s$. $\Delta_s = 1 - B^s$ is the first-order seasonal differencing operator while $\Delta^s = (1-B)^s$ is the sth-order nonseasonal differencing operator. For example, $\Delta_2 Y_t = Y_t - Y_{t-2}$ but $\Delta^2 Y_t = Y_t - 2Y_{t-1} + Y_{t-2}$.

The series $\Delta_s Y_t$ is called the seasonally differenced series. See the middle row of Figure 10.2 for the seasonally differenced logs of housing starts and its ACF.

One can combine seasonal and nonseasonal differencing by using, for example, for first -rder differences

$$\Delta(\Delta_s Y_t) = \Delta(Y_t - Y_{t-s}) = (Y_t - Y_{t-s}) - (Y_{t-1} - Y_{t-s-a}).$$

The order in which the seasonal and nonseasonal difference operators are applied does not matter, since one can show that

$$\Delta(\Delta_s Y_t) = \Delta_s(\Delta Y_t).$$

For a seasonal time series, seasonal differencing is necessary, but whether also to use nonseasonal differencing will depend on the particular time series. For the housing starts data, the seasonally differenced series appears stationary so only seasonal differencing is absolutely needed, but combining seasonal and nonseasonal differencing might be preferred since it results in a simpler model.

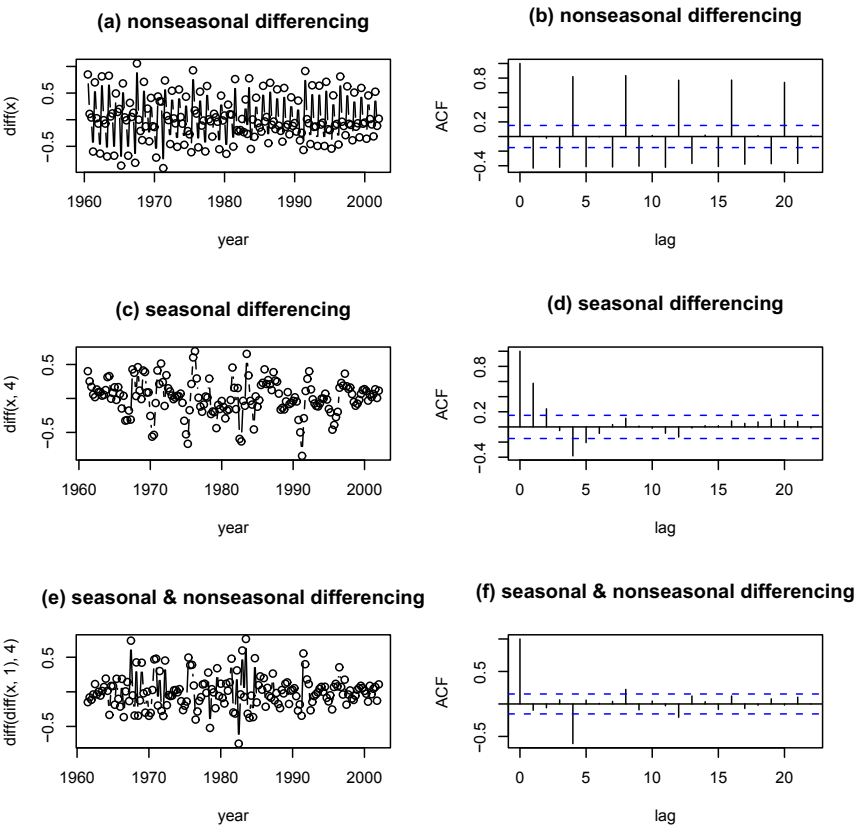

Fig. 10.2. *Time series (left column) and ACF plots (right column) of the logarithms of quarterly urban housing starts with nonseasonal differencing (top row), seasonal differencing (middle row), and both seasonal and nonseasonal differencing (bottom row). Note: In the ACF plots, lag = 1 means a lag of one year, which is four observations for quarterly data.*

10.1.2 Multiplicative ARIMA Models

One of the simplest seasonal models is the ARIMA$\{(1,1,0)\times(1,1,0)_s\}$ model, which puts together the nonseasonal ARIMA(1,1,0) model

$$(1-\phi B)(\Delta Y_t - \mu) = \epsilon_t \tag{10.1}$$

and a purely seasonal ARIMA$(1,1,0)_s$ model

$$(1-\phi^* B^s)(\Delta_s Y_t - \mu) = \epsilon_t \tag{10.2}$$

to obtain the multiplicative model

$$(1-\phi B)\,(1-\phi^* B^s)\,\{\Delta_s(\Delta Y_t)-\mu\}=\epsilon_t. \tag{10.3}$$

Model (10.2) is called "purely seasonal" and has a subscript "s" since it uses only B^s and Δ_s; it is obtained from the ARIMA(1,1,0) by replacing B and Δ by B^s and Δ_s. For a monthly time series ($s = 12$), model (10.2) gives 12 independent processes, one for Januaries, a second for Februaries, and so forth. Model (10.3) uses the components from (10.1) to tie these 12 series together.

The ARIMA$\{(p,d,q)\times(p_s,d_s,q_s)_s\}$ process is

$$\begin{aligned}&(1-\phi_1 B-\cdots-\phi_p B^p)\,\{1-\phi_1^* B^s-\cdots-\phi_{p_s}^*\,(B^s)^{p_s}\}\,\{\Delta^d(\Delta_s^{d_s}Y_t)-\mu\}\\&=(1+\theta_1 B+\ldots+\theta_q B^q)\,\{1+\theta_1^* B^s+\ldots+\theta_{q_s}\,(B^s)^{q_s}\}\,\epsilon_t.\end{aligned} \tag{10.4}$$

This process multiplies together the AR components, the MA components, and the differencing components of two processes: the nonseasonal ARIMA (p,d,q) process

$$(1-\phi_1 B-\cdots-\phi_p B^p)\,\{(\Delta^d Y_t)-\mu\}=(1+\theta_1 B+\ldots+\theta_q B^q)\,\epsilon_t$$

and the seasonal ARIMA$(p_s,d_s,q_s)_s$ process

$$\{1-\phi_1^* B^s-\cdots-\phi_{p_s}^*\,(B^s)^{p_s}\}\,\{(\Delta_s^{d_s}Y_t)-\mu\}=\{1+\theta_1^* B^s+\ldots+\theta_{q_s}\,(B^s)^{q_s}\}\,\epsilon_t.$$

Example 10.1. ARIMA$\{(1,1,1)\times(0,1,1)_4\}$ model for housing starts

We return to the housing starts data. The first question is whether to difference only seasonally, or both seasonally and nonseasonally. The seasonally differenced quarterly series in the middle row of Figure 10.2 is possibly stationary, so perhaps seasonal differencing is sufficient. However, the ACF of the seasonally and nonseasonally differenced series in the bottom row has a simpler ACF than the data that are only seasonally differenced. By differencing both ways, we should be able find a more parsimonious ARMA model.

Two models with seasonal and nonseasonal differencing were tried, ARIMA $\{(1,1,1)\times(1,1,1)_4\}$ and ARIMA $\{(1,1,1)\times(0,1,1)_4\}$. Both provided good fits and had residuals that passed the Ljung–Box test. The second of the two models was selected, because it has one fewer parameter than the first, though the other model would have been a reasonable choice. The results from fitting the chosen model are

```
Call:
arima(x = hst, order = c(1, 1, 1), seasonal
= list(order = c(0, 1, 1), period = 4))

Coefficients:
         ar1      ma1     sma1
```

```
       0.675  -0.890  -0.822
s.e.   0.142   0.105   0.051

sigma^2 estimated as 0.0261: log-likelihood = 62.9,
   aic = -118
```

Thus, the fitted model is

$$(1 - 0.675\,B)Y_t^* = (1 - 0.890\,B)(1 - 0.822\,B_4)\,\epsilon_t$$

where $Y_t^* = \Delta(\Delta_4 Y_t)$ and ϵ_t is white noise.

Figure 10.3 shows forecasts from this model for the four years following the end of the time series.

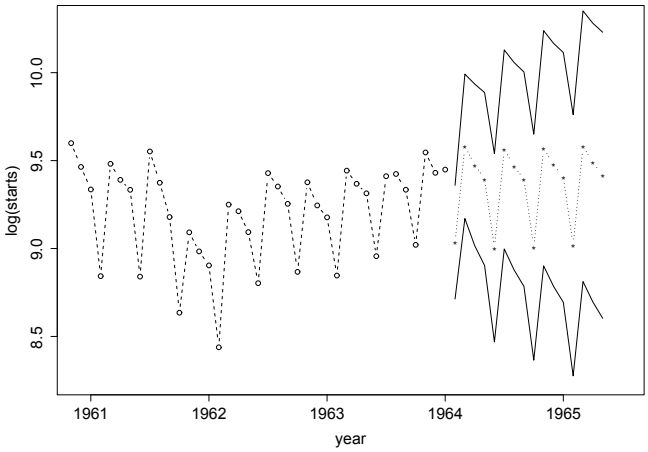

Fig. 10.3. *Forecasting logarithms of quarterly urban housing starts using the ARIMA*$\{(1,1,1) \times (0,1,1)_4\}$ *model. The dashed line connects the data, the dotted line connects the forecasts, and the solid lines are the forecast limits.*

When the size of the seasonal oscillations increases, as with the air passenger data in Figure 9.2, some type of preprocessing is needed before differencing. Often, taking logarithms stabilizes the size of the oscillations. This can be seen in Figure 10.4. Box, Jenkins, and Reinsol (2008) obtain a parsimonious fit to the log passengers with an ARIMA$(0,1,1) \times (0,1,1)_{12}$ model.

For the housing starts series, the data come as logarithms in the `Ecdat` package. If they had come untransformed, then we would have needed to apply some type of transformation.

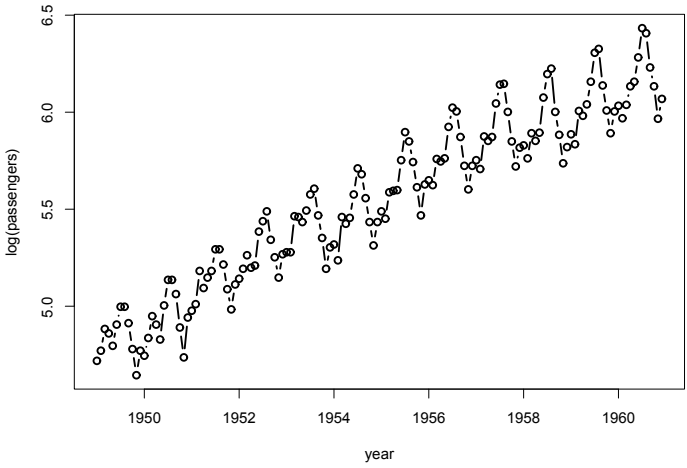

Fig. 10.4. *Time series plot of the logarithms of the monthly totals of air passengers (in thousands).*

10.2 Box–Cox Transformation for Time Series

As just discussed, it is often desirable to transform a time series to stabilize the size of the variability, both seasonal and random. Although a transformation can be selected by trial-and-error, another possibility is automatic selection by maximum likelihood estimation using the model

$$\begin{aligned}(\Delta^d Y_t^{(\alpha)} - \mu) = \phi_1(\Delta^d Y_{t-1}^{(\alpha)} - \mu) + \cdots + \phi_p(\Delta^d Y_{t-p}^{(\alpha)} - \mu) \\ + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q},\end{aligned} \tag{10.5}$$

where $\epsilon_1, \epsilon_2, \ldots$ is Gaussian white noise. Model (10.5) states that after a Box–Cox transformation, Y_t follows an ARIMA model with Gaussian noise that has a constant variance. The transformation parameter α is considered unknown and is estimated by maximum likelihood along with the AR and MA parameters and the noise variance. For notational simplicity, (10.5) uses a nonseasonal model, but a seasonal ARIMA model could just as easily have been used.

Example 10.2. Selecting a transformation for the housing starts

Figure 10.5 show the profile likelihood for α for the housing starts series (not the logarithms). The ARIMA model was ARIMA$\{(1,1,1) \times (1,1,1)_4\}$. The figure was created by the `BoxCox.Arima` function in R's `FitAR` package. This function denotes the transformation parameter by λ. The MLE of α

is 0.34 and the 95% confidence interval is roughly from 0.15 to 0.55. Thus, the log transformation ($\alpha = 0$) is somewhat outside the confidence interval, but the square-root transformation is in the interval. Nonetheless, the log transformation worked satisfactorily in Example 10.1 and might be retained.

Without further analysis, it is not clear why $\alpha = 0.34$ achieves a better fit than the log transformation. Better fit could mean that the ARIMA model fits better, that the noise variability is more nearly constant, that the noise is closer to being Gaussian, or some combination of these effects. It would be interesting to compare forecasts using the log and square-root transformations to see in what ways, if any, the square-root transformation outperforms the log transformation for forecasting. The forecasts would need to be back-transformed to the original scale in order for them to be comparable. One might use the final year as test data to see how well housing starts in that year are forecast.

□

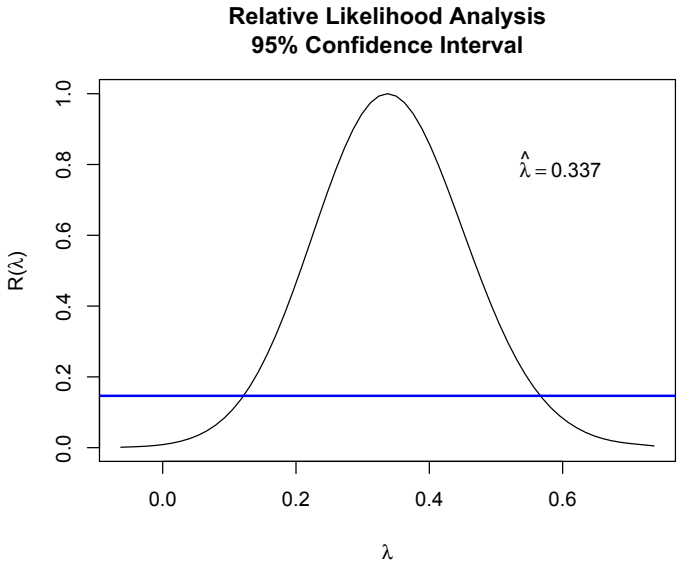

Fig. 10.5. *Profile likelihood for α (called λ in the legend) in the housing start example. Values of λ with $R(\lambda)$ (the profile likelihood) above the horizontal line are in the 95% confidence limit.*

Data transformations can stabilize some types of variation in time series, but not all types. For example, in Figure 9.2 the seasonal oscillations in

the numbers of air passengers increase as the series itself increases, and we can see in Figure 10.4 that a log transformation stabilizes these oscillations. In contrast, the S&P 500 returns in Figure 4.1 exhibit periods of low and high volatility even though the returns maintain a mean near 0. Transformations cannot remove this type of volatility clustering. Instead, the changes of volatility should be modeled by a GARCH process; this topic is pursued in Chapter 18.

10.3 Multivariate Time Series

Suppose that for each t, $\boldsymbol{Y}_t = (Y_{1,t}, \ldots, Y_{d,t})$ is a d-dimensional random vector representing quantities that were measured at time t, e.g., returns on d equities. Then $\boldsymbol{Y}_1, \boldsymbol{Y}_2 \ldots$ is called a d-dimensional *multivariate time series*.

The definition of stationarity for multivariate time series is the same as given before for univariate time series. A multivariate time series said to be *stationary* if for every n and m, $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ and $\boldsymbol{Y}_{1+m}, \ldots, \boldsymbol{Y}_{n+m}$ have the same distributions.

10.3.1 The cross-correlation function

Suppose that Y_j and $Y_{j'}$ are the two component series of a stationary multivariate time series. The *cross-correlation function* (CCF) between Y_j and $Y_{j'}$ is defined as

$$\rho_{Y_j,Y_{j'}}(k) = \mathrm{Corr}\{Y_j(t), Y_{j'}(t-k)\} \tag{10.6}$$

and is the correlation between Y_j at a time t and $Y_{j'}$ at k time units earlier. As with autocorrelation, k is called the *lag*. However, unlike the ACF, the CCF is not symmetric in the lag variable k, that is, $\rho_{Y_j,Y_{j'}}(k) \neq \rho_{Y_j,Y_{j'}}(-k)$. Instead, as a direct consequence of definition (10.6), we have that $\rho_{Y_j,Y_{j'}}(k) = \rho_{Y_{j'},Y_j}(-k)$.

The CCF can be defined for multivariate time series that are not stationary but only weakly stationary. A multivariate time series $\boldsymbol{Y}_1, \ldots$ is said to be weakly stationary if the mean and covariance matrix of $\boldsymbol{Y}_t$ do not depend on t and if the right-hand side of (10.6) is independent of t for all j, j', and k.

Cross-correlations can suggest how the component series might be influencing each other or might be influenced by a common factor. Like all correlations, cross-correlations only show statistical association, not causation, but causal relationship might be deduced from other knowledge.

Example 10.3. Cross-correlation between changes in CPI (Consumer Price Index) and IP (industrial production)

The cross-correlation function between changes in CPI and changes in IP is plotted in Figure 10.6, which was created by the `ccf` function in `R`. The

largest absolute cross-correlations are at positive lags and these correlations are negative. This means that an above-average (below-average) change in CPI predicts a future change in IP that is below (above) average. As just emphasized, correlation does not imply causation, so we cannot say that changes in CPI cause opposite changes in future IP, but the two series behave as if this were happening. Correlation does imply predictive ability. Therefore, if we observe an above-average change in CPI, then we should predict future changes in IP that will be below average. In practice, we should use the currently observed changes in both CPI and IP, not just CPI, to predict future changes in IP. We will discuss prediction using two or more related time series in Section 10.3.4.

□

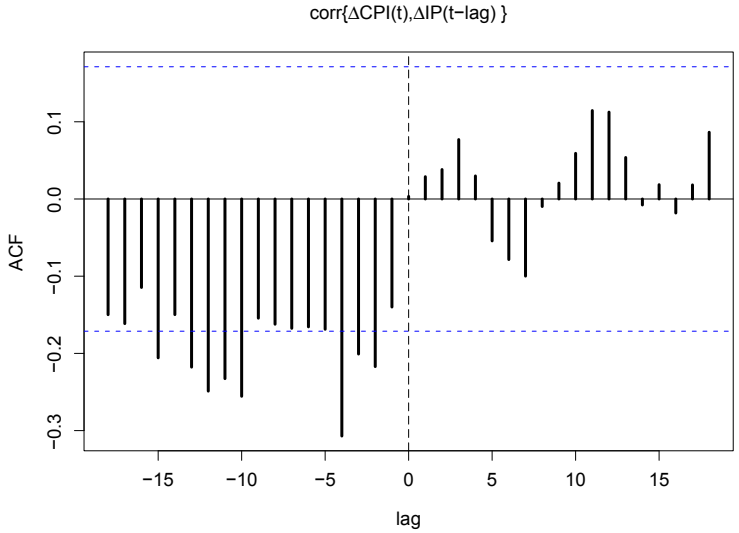

Fig. 10.6. *CCF for ΔCPI and ΔIP. Note the negative correlation at negative lags, that is, between the CPI and future values of IP.*

10.3.2 Multivariate White Noise

A d-dimensional multivariate time series $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots$ is a weak WN$(\mu, \boldsymbol{\Sigma})$ process if

1. $E(\boldsymbol{\epsilon}_t) = \boldsymbol{\mu}$ for all t,
2. COV$(\boldsymbol{\epsilon}_t) = \boldsymbol{\Sigma}$ for all t, and

3. for all $t \neq t'$, all components of $\boldsymbol{\epsilon}_t$ are uncorrelated with all components of $\boldsymbol{\epsilon}_{t'}$.

Notice that if $\boldsymbol{\Sigma}$ is not diagonal, then there is cross-correlation between the components of $\boldsymbol{\epsilon}_t$ because $\text{Corr}(\epsilon_{j,t}, \epsilon_{j',t}) = \Sigma_{j,j'}$; in other words, there may be nonzero *contemporaneous* correlations. However, for all $1 \leq j, j' \leq d$, $\text{Corr}(\epsilon_{j,t}, \epsilon_{j',t'}) = 0$ if $t \neq t'$.

Furthermore, $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots$ is an i.i.d. WN$(\mu, \boldsymbol{\Sigma})$ process if, in addition to conditions 1–3, $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots$ are independent and identically distributed. If $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots$ are also multivariate normally distributed, then they are a Gaussian WN$(\mu, \boldsymbol{\Sigma})$ process.

10.3.3 Multivariate ARMA processes

A d-dimensional multivariate time series $\boldsymbol{Y}_1, \ldots$ is a multivariate ARMA(p, q) process with mean $\boldsymbol{\mu}$ if for $p \times p$ matrices $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p$ and $\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_q$,

$$\boldsymbol{Y}_t - \boldsymbol{\mu} = \boldsymbol{\Phi}_1(\boldsymbol{Y}_{t-1} - \boldsymbol{\mu}) + \cdots + \boldsymbol{\Phi}_p(\boldsymbol{Y}_{t-p} - \boldsymbol{\mu}) + \boldsymbol{\Theta}_1\boldsymbol{\epsilon}_{t-1} + \cdots + \boldsymbol{\Theta}_q\boldsymbol{\epsilon}_{t-q} + \boldsymbol{\epsilon}_t, \tag{10.7}$$

where $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n$ is a multivariate WN$(0, \boldsymbol{\Sigma})$ process. Multivariate AR processes (the case $q = 0$) are also called vector AR or VAR processes and are widely used in practice.

As an example, a bivariate AR(1) process can be written as

$$\begin{pmatrix} Y_{1,t} - \mu_1 \\ Y_{2,t} - \mu_2 \end{pmatrix} = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} \\ \phi_{2,1} & \phi_{2,2} \end{pmatrix} \begin{pmatrix} Y_{1,t-1} - \mu_1 \\ Y_{2,t-1} - \mu_2 \end{pmatrix} + \begin{pmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \end{pmatrix},$$

where

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}_1 = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} \\ \phi_{2,1} & \phi_{2,2} \end{pmatrix}.$$

Therefore,

$$Y_{1,t} = \mu_1 + \phi_{1,1}(Y_{1,t-1} - \mu_1) + \phi_{1,2}(Y_{2,t-1} - \mu_2) + \epsilon_{1,t}$$

and

$$Y_{2,t} = \mu_2 + \phi_{2,1}(Y_{1,t-1} - \mu_1) + \phi_{2,2}(Y_{2,t-1} - \mu_2) + \epsilon_{2,t},$$

so that $\phi_{i,j}$ is the amount of "influence" of $Y_{j,t-1}$ on $Y_{i,t}$. Similarly, for a bivariate AR(p) process, $\phi^k_{i,j}$ (the i,jth component of $\boldsymbol{\Phi}^k$) is the influence of $Y_{j,t-k}$ on $Y_{i,t}$, $k = 1, \ldots, p$.

For a d-dimensional AR(1), it follows from (10.7) with $p = 1$ and $\boldsymbol{\Phi} = \boldsymbol{\Phi}_1$ that

$$E(\boldsymbol{Y}_t | \boldsymbol{Y}_{t-1}) = \boldsymbol{\mu} + \boldsymbol{\Phi}(\boldsymbol{Y}_{t-1} - \boldsymbol{\mu}). \tag{10.8}$$

How does $E(\boldsymbol{Y}_t)$ depend on the more distant past, say on $\boldsymbol{Y}_{t-2}$? To answer this question, we can generalize (10.8). To keep notation simple, assume that the mean has been subtracted from $\boldsymbol{Y}_t$ so that $\boldsymbol{\mu} = 0$. Then

$$\boldsymbol{Y}_t = \boldsymbol{\Phi}\boldsymbol{Y}_{t-1} + \boldsymbol{\epsilon}_t = \boldsymbol{\Phi}\{\boldsymbol{\Phi}\boldsymbol{Y}_{t-1} + \boldsymbol{\epsilon}_{t-1}\} + \boldsymbol{\epsilon}_t$$

and, because $E(\boldsymbol{\epsilon}_{t-1}|\boldsymbol{Y}_{t-2}) = 0$ and $E(\boldsymbol{\epsilon}_t|\boldsymbol{Y}_{t-2}) = 0$,

$$E(\boldsymbol{Y}_t|\boldsymbol{Y}_{t-2}) = \boldsymbol{\Phi}^2\boldsymbol{Y}_{t-2}.$$

By similar calculations,

$$E(\boldsymbol{Y}_t|\boldsymbol{Y}_{t-k}) = \boldsymbol{\Phi}^k\boldsymbol{Y}_{t-k}, \text{ for all } k > 0. \tag{10.9}$$

It can be shown using (10.9), that the mean will explode if any of the eigenvectors of $\boldsymbol{\Phi}$ are greater than 1 in magnitude. In fact, an AR(1) process is stationary if and only if all of the eigenvalues of $\boldsymbol{\Phi}$ are less than 1 in absolute value. The `eigen` function in `R` can be used to find the eigenvalues.

Example 10.4. A bivariate AR model for ΔCPI and ΔIP

This example uses the CPI and IP data sets discussed in earlier examples. Bivariate AR processes were fit to (Δ CPI, Δ IP) using `R`'s function `ar`. AIC as a function of p is shown below. The two best-fitting models are AR(1) and AR(5), with the latter being slightly better by AIC. Although BIC is not part of `ar`'s output, it can be calculated easily since BIC = AIC + $\{\log(n) - 2\}p$. Because $\{\log(n) - 2\} = 2.9$ in this example, it is clear that BIC is much smaller for the AR(1) model than for the AR(5) model. For this reason and because the AR(1) model is so much simpler to analyze, we will use the AR(1) model.

```
p              0      1      2      3      4
AIC       127.99   0.17   1.29   5.05   3.40

          5      6      7      8      9     10
       0.00   6.87   9.33  10.83  13.19  14.11
```

The results of fitting the AR(1) model are

$$\widehat{\boldsymbol{\Phi}} = \begin{pmatrix} 0.767 & 0.0112 \\ -0.330 & 0.3014 \end{pmatrix}$$

and

$$\widehat{\boldsymbol{\Sigma}} = \begin{pmatrix} 5.68\text{e}-06 & 3.33\text{e}-06 \\ 3.33\text{e}-06 & 6.73\text{e}-05 \end{pmatrix}. \tag{10.10}$$

`ar` does not estimate μ, but μ can be estimated by the sample mean, which is is (0.00173, 0.00591).

It is useful to look at the two off-diagonals of $\widehat{\boldsymbol{\Phi}}$. Since $\Phi_{1,2} = 0.01 \approx 0$, $Y_{2,t-1}$ (lagged IP) has little influence on $Y_{1,t}$ (CPI), and since $\Phi_{2,1} = -0.330$, $Y_{1,t-1}$ (lagged CPI) has a substantial negative effect on $Y_{2,t}$ (IP). It should

be emphasized that "effect" means statistical association, not necessarily causation. This agrees with what we found when looking at the CCF for these series in Example 10.3.

How does IP depend on CPI further back in time? To answer this question we look at the (1,2) elements of the following powers of $\boldsymbol{\Phi}$:

$$\widehat{\boldsymbol{\Phi}}^2 = \begin{pmatrix} 0.58 & 0.012 \\ -0.35 & 0.087 \end{pmatrix}, \quad \widehat{\boldsymbol{\Phi}}^3 = \begin{pmatrix} 0.44 & 0.010 \\ -0.30 & 0.022 \end{pmatrix},$$

$$\widehat{\boldsymbol{\Phi}}^4 = \begin{pmatrix} 0.34 & 0.0081 \\ -0.24 & 0.0034 \end{pmatrix}, \quad \text{and} \quad \widehat{\boldsymbol{\Phi}}^5 = \begin{pmatrix} 0.26 & 0.0062 \\ -0.18 & -0.0017 \end{pmatrix}.$$

What is interesting here is that the (1,2) elements, that is, -0.35, -0.30, -0.24, and -0.18, decay to zero slowly, much like the CCF. This helps explain why the AR(1) model fits the data well. This behavior where the cross-correlations are all negative and decay only slowly to zero is quite different from the behavior of the ACF of a univariate AR(1) process. For the later, the correlations either are all positive or else alternate in sign, and in either case, unless the lag-1 correlation is nearly equal to 1, the correlations decay rapidly to 0.

In contrast to these negative correlations between ΔCPI and future ΔIP, it follows from (10.10) that the white noise series has a positive, albeit small, correlation of $3.33/\sqrt{(5.68)(67.3)} = 0.17$. The white noise series represents unpredictable changes in the ΔCPI and ΔIP series, so we see that the unpredictable changes have positive correlation. In contrast, the negative correlations between ΔCPI and future ΔIP concern predictable changes.

Figure 10.7 shows the ACF of the ΔCPI and ΔIP residuals and the CCF of these residuals. There is little auto- or cross-correlation in the residuals at nonzero lags, indicating that the AR(1) has a satisfactory fit.

Figure 10.7 was produced by the `acf` function in `R`. When applied to a multivariate time series, `acf` creates a matrix of plots. The univariate ACFs are on the main diagonal, the ccf's at positive lags are above the main diagonal, and the CCFs at negative values of lag below the main diagonal.

□

10.3.4 Prediction Using Multivariate AR Models

Forecasting with multivariate AR processes is much like forecasting with univariate AR processes. Given a multivariate AR(p) time series $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, the forecast of $\boldsymbol{Y}_{n+1}$ is

$$\widehat{\boldsymbol{Y}}_{n+1} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1(\boldsymbol{Y}_n - \widehat{\boldsymbol{\mu}}) + \cdots + \widehat{\boldsymbol{\Phi}}_p(\boldsymbol{Y}_{n+1-p} - \widehat{\boldsymbol{\mu}}),$$

the forecast of $\boldsymbol{Y}_{n+2}$ is

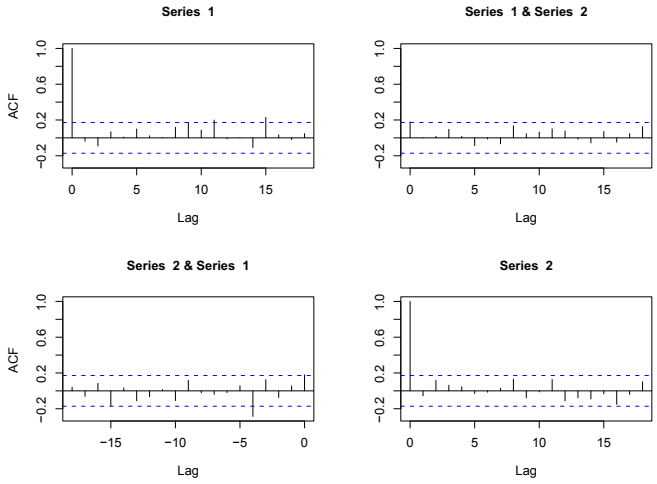

Fig. 10.7. *The ACF and CCF for the residuals when fitting a bivariate AR(1) model to (Δ CPI, Δ IP). Top left: The ACF of ΔCPI residuals. Top right: The CCF of ΔCPI and ΔIP residuals with positive values of lag. Bottom left: The CCF of ΔCPI and ΔIP residuals with negative values of lag. Bottom right: The ACF of ΔIP residuals.*

$$\widehat{\boldsymbol{Y}}_{n+2} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1(\widehat{\boldsymbol{Y}}_{n+1} - \widehat{\boldsymbol{\mu}}) + \cdots + \widehat{\boldsymbol{\Phi}}_p(\boldsymbol{Y}_{n+2-p} - \widehat{\boldsymbol{\mu}}),$$

and so forth, so that for all k,

$$\widehat{\boldsymbol{Y}}_{n+k} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1(\widehat{\boldsymbol{Y}}_{n+k-1} - \widehat{\boldsymbol{\mu}}) + \cdots + \widehat{\boldsymbol{\Phi}}_p(\widehat{\boldsymbol{Y}}_{n+k-p} - \widehat{\boldsymbol{\mu}}), \tag{10.11}$$

where we use the convention that $\widehat{\boldsymbol{Y}}_t = \boldsymbol{Y}_t$ if $t \leq n$. For an AR(1) model, repeated application of (10.11) shows that

$$\widehat{\boldsymbol{Y}}_{n+k} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1^k(\boldsymbol{Y}_n - \widehat{\boldsymbol{\mu}}). \tag{10.12}$$

Example 10.5. Using a bivariate AR(1) model to predict CPI and IP

The ΔCPI and ΔIP series were forecast using (10.12) with estimates found in Example 10.4. Figure 10.8 shows forecasts up to 10 months ahead for both CPI and IP. Figure 10.9 show forecast limits computed by simulation using the techniques described in Section 9.12.2 generalized to a multivariate time series.

□

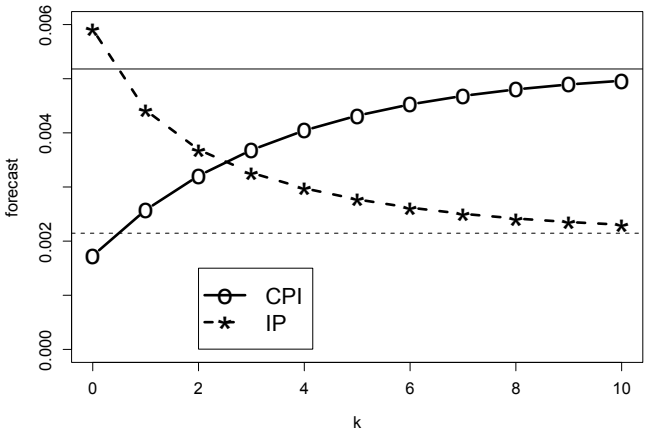

Fig. 10.8. *Forecasts of changes in CPI (solid) and changes in IP (dashed) using a bivariate AR(1) model. The number of time units ahead is k. At $k = 0$, the last observed values of the time series are plotted. The two horizontal lines are at the means of the series, and the forecasts will asymptote to these lines as $k \to \infty$.*

10.4 Long-Memory Processes

10.4.1 The Need for Long-Memory Stationary Models

In Chapter 9, ARMA processes were used to model stationary time series. Stationary ARMA processes have only short memories in that their autocorrelation functions decay to zero exponentially fast. That is, there exist a $D > 0$ and $r < 1$ such that

$$\rho(k) < D|r|^k$$

for all k. In contrast, many financial time series appear to have long memory since their ACFs decay at a (slow) polynomial rather than a (fast) exponential rate, that is,

$$\rho(k) \sim Dk^{-\alpha}$$

for some D and $\alpha > 0$. A polynomial rate of decay is sometimes called a hyperbolic rate. In this section, we will introduce the fractional ARIMA models, which include stationary processes with long memory.

10.4.2 Fractional Differencing

The most widely used models for stationary, long-memory processes use fractional differencing. For integer values of d we have

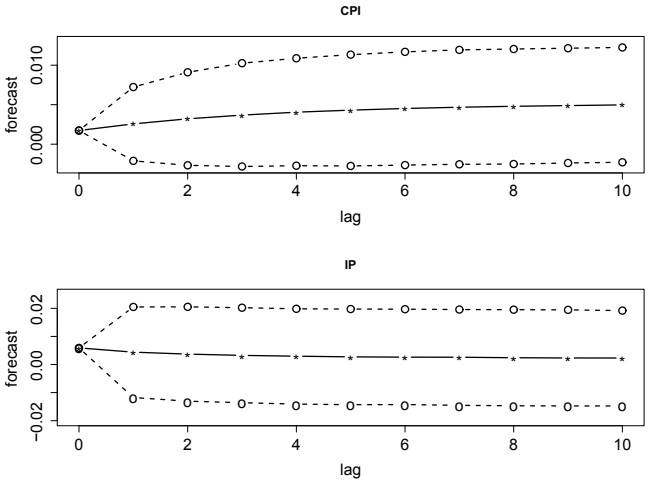

Fig. 10.9. *Forecast limits (dashed) for changes in CPI and IP computed by simulation and forecasts (solid). At lag = 0, the last observed changes are plotted so the widths of the forecast intervals are zero.*

$$\Delta^d = (1-B)^d = \sum_{k=0}^{d} \binom{d}{k} (-B)^k. \tag{10.13}$$

In this subsection, the definition of Δ^d will be extended to noninteger values of d. The only restriction on d will be that $d > -1$.

Let $\Gamma(t) = \int_0^\infty x^{t-1}e^{-x}dx$, for any $t > 0$, be the gamma function previously defined by (5.13). Integration by parts shows that

$$\Gamma(t) = (t-1)\Gamma(t-1) \tag{10.14}$$

and simple integration shows that $\Gamma(1) = 1$. It follows that for any integer t, we have $\Gamma(t+1) = t!$. Therefore, the definition of $t!$ can be extended to all $t > 0$ if $t!$ is defined as $\Gamma(t+1)$ whenever $t > 0$. Moreover, (10.14) allows the definition of $\Gamma(t)$ to be extended to all t except nonnegative integers. For example, $\Gamma(1/2) = -(1/2)\,\Gamma(-1/2)$, so we can define $\Gamma(-1/2)$ as $-2\,\Gamma(1/2)$. However, this device does not work if t is 0 or a negative integer. For example, $\Gamma(1) = 0\Gamma(0)$ does not give us a way to define $\Gamma(0)$. In summary, $\Gamma(t)$ can be defined for all real t except $0, -1, -2, \ldots$ and therefore $t!$ can be defined for all real values of t except the negative integers.

We can now define

$$\binom{d}{k} = \frac{d!}{k!(d-k)!} \tag{10.15}$$

for any d except negative integers and any integer $k \geq 0$, except if d is an integer and $k > d$, in which case $d-k$ is a negative integer and $(d-k)!$ is not

defined. In the latter case, we define $\binom{d}{k}$ to be 0, so $\binom{d}{k}$ is defined for all d except negative integers and for all integer $k \geq 0$. Only values of d greater than -1 are needed for modeling long-memory processes, so we will restrict attention to this case.

The function $f(x) = (1-x)^d$ has an infinite Taylor series expansion

$$(1-x)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-x)^k. \tag{10.16}$$

Since $\binom{d}{k} = 0$ if $k > 0$ and $d > -1$ is integer, when d is an integer we have

$$(1-x)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-x)^k = \sum_{k=0}^{d} \binom{d}{k} (-x)^k. \tag{10.17}$$

The right-hand side of (10.17) is the usual finite binomial expansion for d a nonnegative integer, so (10.16) extends the binomial expansion to all $d > -1$. Since $(1-x)^d$ is defined for all $d > -1$, we can define $\Delta^d = (1-B)^d$ for any $d > -1$. In summary, if $d > -1$, then

$$\Delta^d Y_t = \sum_{k=0}^{\infty} \binom{d}{k} (-1)^k Y_{t-k}. \tag{10.18}$$

10.4.3 FARIMA Processes

Y_t is a fractional ARIMA(p, d, q) process, also called an ARFIMA or FARIMA (p, d, q) process, if $\Delta^d Y_t$ is an ARMA(p, q) process. We say that Y_t is a fractionally integrated process of order d or, simply, $I(d)$ process. This is, of course, the previous definition of an ARIMA process extended to noninteger values of d. Usually, $d \geq 0$, with $d = 0$ being the ordinary ARMA case, but d could be negative. If $-1/2 < d < 1/2$, then the process is stationary. If $0 < d < 1/2$, then it is a long-memory stationary processes.

If $d > \frac{1}{2}$, then Y_t can be differenced an integer number of times to become a stationary process, though perhaps with long-memory. For example, if $\frac{1}{2} < d < 1\frac{1}{2}$, then ΔY_t is fractionally integrated of order $d - 1 \in (-\frac{1}{2}, \frac{1}{2})$ and ΔY_t. has long-memory if $1 < d < 1\frac{1}{2}$ so that $d - 1 \in (0, \frac{1}{2})$.

Figure 10.10 shows time series plots and sample ACFs for simulated FARIMA$(0, d, 0)$ processes with $n = 2500$ and $d = -0.35$, 0.35, and 0.7. The last case is nonstationary. The `R` function `simARMA0` in the `longmemo` package was used to simulate the stationary series. For the case $d = 0.7$, `simARMA0` was used to simulate an FARIMA$(0, -0.3, 0)$ series and this was integrated to create a FARIMA$(0, d, 0)$ with $d = -0.3 + 1 = 0.7$. As explained in Section

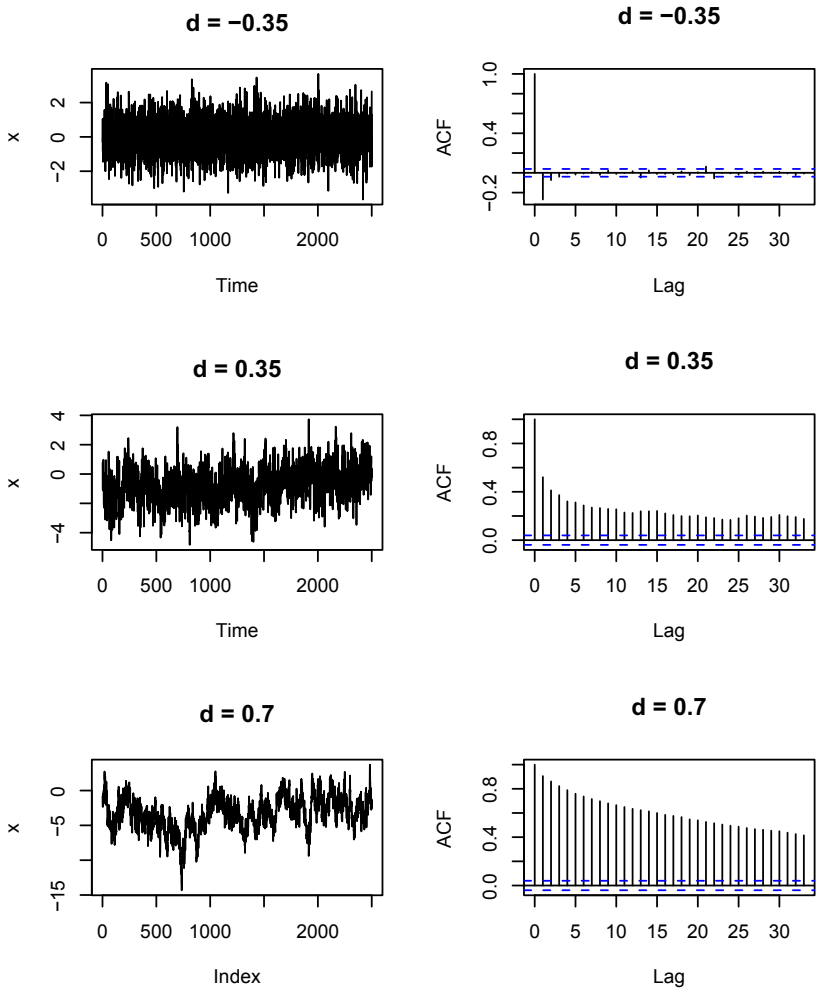

Fig. 10.10. *Time series plots (left) and sample ACFs (right) for simulated FARIMA*$(0, d, 0)$*. The top series is stationary with short-term memory. The middle series is stationary with long-term memory. The bottom series is nonstationary.*

9.9, integration is implemented by taking partial sums, and this was done with R's function `cumsum`.

The FARIMA$(0, 0.35, 0)$ process has a sample ACF with drops below 0.5 almost immediately but then persists well beyond 30 lags. This behavior is typical of stationary processes with long memory. A short-memory stationary process would not have autocorrelations persisting that long, and a nonsta-

tionary processes would not have a sample ACF that dropped below 0.5 so quickly.

Note that the case $d = -0.35$ in Figure 10.10 has an ACF with a negative lag-1 autocorrelation and little additional autocorrelation. This type of ACF is often found when a time series is differenced once. After differencing, an MA term is needed to accommodate the negative lag-1 autocorrelated. A more parsimonious model can sometimes be used if the differencing is fractional. For example, consider the third series in Figure 10.10. If it is differenced once, then a series with $d = -0.3$ is the result. However, if it is differenced with $d = 0.7$, then white noise is the result. This can be seen in the ACF plots in Figure 10.11.

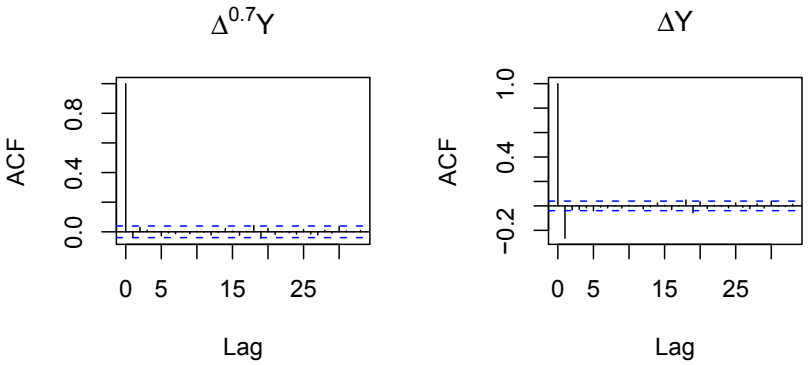

Fig. 10.11. *ACF plots for the simulated FARIMA*$(0, 0.7, 0)$ *series in Figure* 10.10 *after differencing using* $d = 0.7$ *and* 1.

Example 10.6. Inflation rates—FARIMA modeling

This example used the inflation rates that have been studied already in Chapter 9. From the analysis in that chapter it was unclear whether to model the series as $I(0)$ or $I(1)$. Maybe it would be better to have a compromise between these alternatives. Now, with the new tool of fractional integration, we can try differencing with d between 0 and 1. There is some reason to believe that fractional differencing is suitable for this example, since the ACF plot in Figure 9.3 is similar to that of the $d = 0.35$ plot in Figure 10.10.

The function `fracdiff` in `R`'s `fracdiff` package will fit a FARIMA (p, d, q) process. The values of p, d, and q must be input; I am not aware of any `R` function that will chose p, d, and q automatically in the way this can be done for an ARIMA process (that is, with d restricted to be an integer) using

`auto.arima`. First, a trial value of d was chosen by using `fracdiff` with $p = q = 0$, the default values. The estimate was $\widehat{d} = 0.378$. Then, the inflation rates were fractionally differenced using this value of d and `auto.arima` was applied to the fractionally differenced series. The result was that BIC selected $p = q = d = 0$. The value $d = 0$ means that no further differencing is applied to the already fractionally differenced series. Fractional differencing was done with the `diffseries` function in R's `fracdiff` package.

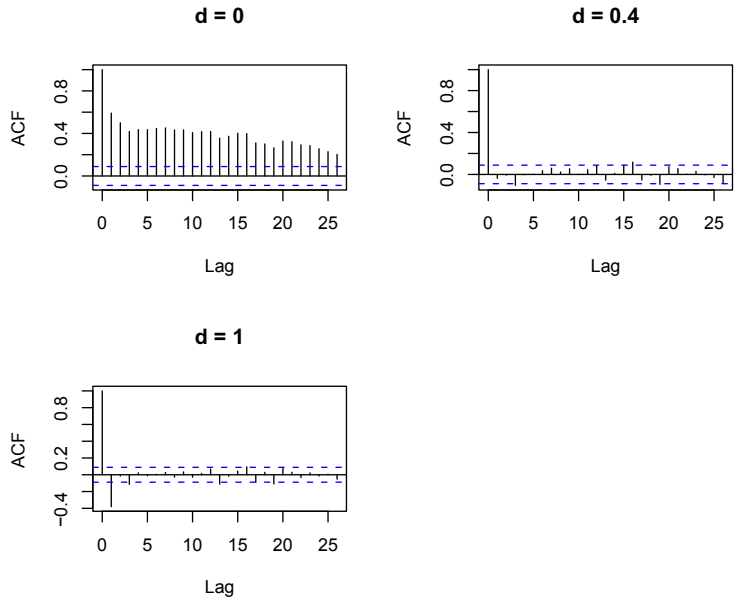

Fig. 10.12. *ACF plots for the inflation rates series with differencing using* $d = 0$, 0.4*, and* 1.

Figure 10.12 has ACF plots of the original series and the series differenced with $d = 0$, 0.4 (from rounding 0.378), and 1. The first series has a slowly decaying ACF typical of a long-memory process, the second series looks like white noise, and the third series has negative autocorrelation at lag-1 which indicates overdifferencing.

The conclusion is that a white noise process seems to be a suitable model for the fractionally differenced series and the original series can be model as FARIMA(0,0.378,0), or, perhaps, more simply as FARIMA(0,0.4,0).

Differencing a stationary process creates another stationary process, but the differenced process often has more complex autocorrelation structure compared to the original process. Therefore, one should not *overdifference* a time

series. However, if d is restricted to integer values, then often, as in this example, overdifferencing cannot be avoided.

□

10.5 Bootstrapping Time Series

The resampling methods introduced in Chapter 6 are designed for i.i.d. univariate data but are easily extended to multivariate data. As discussed in Section 7.11, if $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ is a sample of vectors, then one resamples the $\boldsymbol{Y}_i$ themselves, not their components, to maintain the covariance structure of the data in the resamples.

It is not immediately obvious whether one can resample a time series $Y_1, Y_2, \ldots, Y_n$. A time series is essentially a sample of size 1 from a stochastic process. Resampling a sample of size 1 in the usual way is a futile exercise—each resample is the original sample, so one learns nothing by resampling. Therefore, resampling of a time series requires new ideas.

Model-based resampling is easily adapted to time series. The resamples are obtained by simulating the time series model. For example, if the model is ARIMA$(p, 1, q)$, then the resamples start with simulated samples of an ARMA(p, q) model with MLEs (from the differenced series) of the autoregressive and moving average coefficients and the noise variance. The resamples are the sequences of partial sums of the simulated ARMA(p, q) process.

Model-free resampling of a time series is accomplished by *block resampling*, also called the *block bootstrap*, which can be implemented using the `tsboot` function in `R`'s boot package. The idea is to break the time series into roughly equal-length blocks of consecutive observations, to resample the blocks with replacment, and then to paste the blocks together. For example, if the time series is of length 200 and one uses 10 blocks of length 20, then the blocks are the first 20 observations, the next 20, and so forth. A possible resample is the fourth block (observations 61 to 80), then the last block (observations 181 to 200), then the second block (observations 21 to 40), then the fourth block again, and so on until there are 10 blocks in the resample.

A major issue is how best to select the block length. The correlations in the original sample are preserved only within blocks, so a large block size is desirable. However, the number of possible resamples depends on the number of blocks, so a large number of blocks is also desirable. Obviously, there must be a tradeoff between the block size and the number of blocks. A full discussion of block bootstraping is beyond the scope of this book, but see Section 10.6 for further reading.

10.6 Bibliographic Notes

Beran (1994) is a standard reference for long-memory processes, and Beran (1992) is a good introduction to this topic. Most of the time series textbooks listed in Section 9.15 discuss seasonal ARIMA models. Enders (2004) has a section of bootstrapping time series and a chapter on multivariate time series. Reinsel (2003) is an in-depth treatment of multivariate time series; see also Hamilton (1994) for this topic. Transfer function models are another method for analyzing multivariate time series; see Box, Jenkins, and Reinsel (2008). Davison and Hinkley (1997) discuss both model-based and block resampling of time series and other types of dependent data. Lahiri (2003) provides an advanced and comprehensive account of block resampling. Bühlmann (2002) is a review article about bootstrapping time series.

10.7 References

Beran, J. (1992) Statistical methods for data with long-range dependence. *Statistical Science*, **7**, 404–427.

Beran, J. (1994) *Statistics for Long-Memory Processes*, Chapman & Hall, Boca Raton, FL.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008) *Times Series Analysis: Forecasting and Control*, 4th ed., Wiley, Hoboken, NJ.

Bühlmann, P. (2002) Bootstraps for time series. *Statistical Science*, **17**, 52–72.

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Applications*, Cambridge University Press, Cambridge.

Enders, W. (2004) *Applied Econometric Time Series*, 2nd ed., Wiley, New York.

Hamilton, J. D. (1994) *Time Series Analysis*, Princeton University Press, Princeton, NJ.

Lahiri, S. N. (2003) *Resampling Methods for Dependent Data*, Springer, New York.

Reinsel, G. C. (2003) *Elements of Multivariate Time Series Analysis*, 2nd ed., Springer, New York.

10.8 R Lab

10.8.1 Seasonal ARIMA Models

This section uses seasonally nonadjusted quarterly data on income and consumption in the UK. Run the following code to load the data and plot the variable `consumption`.

```
library("Ecdat")
data(IncomeUK)
consumption = IncomeUK[,2]
plot(consumption)
```

Problem 1 *Describe the behavior of* `consumption`. *What types of differencing, seasonal, nonseasonal, or both, would you recommend? Do you recommend fitting a seasonal ARIMA model to the data with or without a log transformation? Consider also using ACF plots to help answer these questions.*

Problem 2 *Regardless of your answers to Problem 1, find an ARIMA model that provides a good fit to* `log(consumption)`. *What order model did you select? (Give the orders of the nonseasonal and seasonal components.)*

Problem 3 *Check the ACF of the residuals from the model you selected in Problem 2. Do you see any residual autocorrelation?*

Problem 4 *Apply* `auto.arima` *to* `log(consumption)` *using BIC. What model is selected?*

Problem 5 *Forecast* `log(consumption)` *for the next eight quarters using the models you found in Problems 2 and 4. Plot the two sets of forecasts in side-by-side plots with the same limits on the x- and y-axes. Describe any differences between the two sets of forecasts.*

Note: To predict an `arima` object (an object returned by the `arima` function), use the `predict` function. To learn how the predict function works on an `arima` object, use `?predict.Arima`. To forecast an object returned by `auto.arima`, use the `forecast` function in the `forecast` package. For example, the following code will forecast eight quarters ahead using the object returned by `auto.arima` and then plot the forecasts.

```
fitAutoArima = auto.arima(logConsumption,ic="bic")
foreAutoArima = forecast(fitAutoArima,h=8)
plot(foreAutoArima,xlim=c(1985.5,1987.5),ylim=c(10.86,11))
```

10.8.2 VAR Models

This section uses data on the 91-day Treasury bill, the real GDP, and the inflation rate. Run the following `R` code to read the data, find the best-fitting multivariate AR to changes in the three series, and check the residual correlations.

```
data(Tbrate,package="Ecdat")
#  r = the 91-day Treasury bill rate
#  y = the log of real GDP
#  pi = the inflation rate
del_dat = diff(Tbrate)
var1 = ar(del_dat,order.max=4,aic=T)
var1
acf(var1$resid[-1,])
```

Problem 6 *For this problem, use the notation of equation (10.7) with* $q = 0$.

(a) What is p *and what are the estimates* $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p$*?*
(b) What is the estimated covariance matrix of $\boldsymbol{\epsilon}_t$*?*
(c) If the model fits adequately, then there should be no residual auto- or cross-correlation. Do you believe that the model does fit adequately?

Problem 7 *The last three changes in* `r`*,* `y`*, and* `pi` *are given next. What are the predicted values of the next set of changes in these series?*

```
     r          y    pi
 -1.41 -0.019420  2.31
 -0.48  0.015147 -1.01
  0.66  0.003303  0.31
```

10.8.3 Long-Memory Processes

This section uses changes in the square root of the Consumer Price Index. The following code creates this time series.

```
data(Mishkin,package="Ecdat")
cpi = as.vector(Mishkin[,5])
DiffSqrtCpi = diff(sqrt(cpi))
```

Problem 8 *Plot* `DiffSqrtCpi` *and its ACF. Do you see any signs of long memory? If so, describe them.*

Run the following code to estimate the amount of fractional differencing, fractionally difference `DiffSqrtCpi` appropriately, and check the ACF of the fractionally differenced series.

```
library("fracdiff")
fit.frac = fracdiff(DiffSqrtCpi,nar=0,nma=0)
fit.frac$d
fdiff = diffseries(DiffSqrtCpi,fit.frac$d)
acf(fdiff)
```

Problem 9 *Do you see any short- or long-term autocorrelation in the fractionally differenced series?*

Problem 10 *Fit an ARIMA model to the fractionally differenced series using* `auto.arima`*. Compare the models selected using AIC and BIC.*

10.8.4 Model-Based Bootstrapping of an ARIMA Process

This example uses the price of frozen orange juice. Run the following code to fit an ARIMA model.

```
library(AER)
library(forecast)
data("FrozenJuice")
price = FrozenJuice[,1]
plot(price)
auto.arima(price,ic="bic")
```

The output from `auto.arima`, which is needed for model-based bootstrapping, is

```
Series: price
ARIMA(2,1,0)

Coefficients:
         ar1     ar2
      0.2825  0.0570
s.e.  0.0407  0.0408

sigma^2 estimated as 9.989:  log likelihood = -1570.11
AIC = 3146.23   AICc = 3146.27   BIC = 3159.47
```

Next, we will use the model-based bootstrap to investigate how well BIC selects the "correct" model, which is ARIMA(2,0,0). Since we will be looking at the output of each fitted model, only a small number of resamples will be used. Despite the small number of resamples, we will get some sense of how well BIC works in this context. To simulate 10 model-based resamples from the ARIMA(2,0,0) model, run

```
n=length(price)
sink("priceBootstrap.txt")
set.seed(1998852)
for (iter in 1:10)
{
eps = rnorm(n+20)
```

```
y = rep(0,n+20)
for (t in 3:(n+20))
{
y[t] = .2825 *y[t-1] + 0.0570*y[t-2] + eps[t] }
y = y[101:n+20]
y = cumsum(y)
y = ts(y,frequency=12)
fit=auto.arima(y,d=1,D=0,ic="bic")
print(fit)
}
sink()
```

The results will be sent to the file `priceBootstrap.txt`. The first two values of `y` are independent and are used to initialize the process. A burn-in period of 20 is used to remove the effect of initialization. Note the use of `cumsum` to integrate the simulated AR(2) process and the use of `ts` to convert a vector to a monthly time series.

Problem 11 *How often is the "correct" AR(2) model selected?*

Now we will perform a bootstrap where the correct model AR(2) is known and study the accuracy of the estimators. Since the correct model is known, it can be fit by `arima`. The estimates will be stored in a matrix called `estimates`. In contrast to earlier when model-selection was investigated by resampling, now a large number of bootstrap samples can be used, since `arima` is fast and only the estimates are stored. Run the following:

```
set.seed(1998852)
niter=250
estimates=matrix(0,nrow=niter,ncol=2)
for (iter in 1:niter)
{
eps = rnorm(n+20)
y = rep(0,n+20)
for (t in 3:(n+20))
{
y[t] = .2825 *y[t-1] + 0.0570*y[t-2] + eps[t] }
y = y[101:n+20]
y = cumsum(y)
y = ts(y,frequency=12)
fit=arima(y,order=c(2,1,0))
estimates[iter,]=fit$coef
}
```

Problem 12 *Find the biases, standard deviations, and MSEs of the estimators of the two coefficients.*

10.9 Exercises

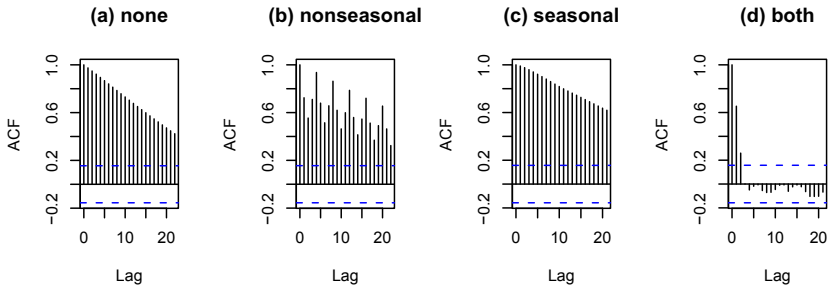

Fig. 10.13. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

1. Figure 10.13 contains ACF plots of 40 years of quarterly data, with all possible combinations of first-order seasonal and nonseasonal differencing. Which combination do you recommend in order to achieve stationarity?

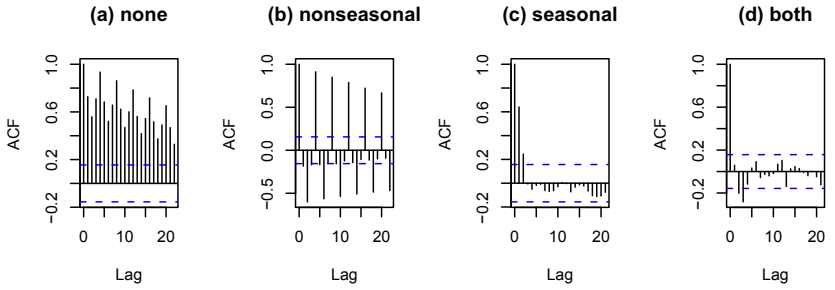

Fig. 10.14. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

2. Figure 10.14 contains ACF plots of 40 years of quarterly data, with all possible combinations of first-order seasonal and nonseasonal differencing. Which combination do you recommend in order to achieve stationarity?

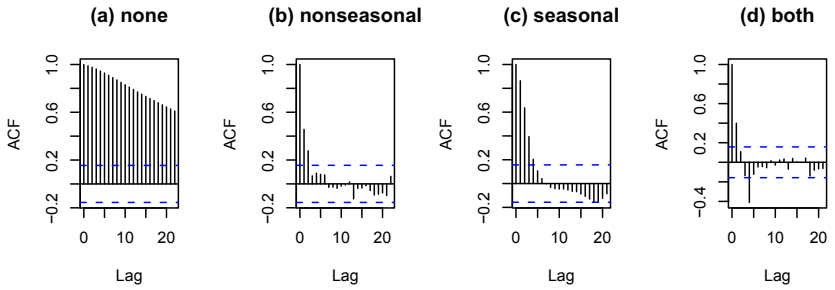

Fig. 10.15. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

3. Figure 10.15 contains ACF plots of 40 years of quarterly data, with all possible combinations of first-order seasonal and nonseasonal differencing. Which combination do you recommend in order to achieve stationarity?
4. In example 10.4, a bivariate AR(1) model was fit to (ΔCPI, ΔIP) and

$$\widehat{\boldsymbol{\Phi}} = \begin{pmatrix} 0.767 & 0.0112 \\ -0.330 & 0.3014 \end{pmatrix}.$$

   The mean of (ΔCPI, ΔIP) is (0.00518, 0.00215) and the last observation of (Δ CPI, Δ IP) is (0.00173, 0.00591). Forecast the next two values of Δ IP. (The forecasts are shown in Figure 10.8, but you should compute numerical values.)
5. Fit an ARIMA model to `income`, which is in the first column of the `IncomeUK` data set in the `Ecdat` package. Explain why you selected the model you did. Does you model exhibit any residual correlation?
6. (a) Find an ARIMA model that provides a good fit to the variable `unemp` in the `USMacroG` data set in the `AER` package.
   (b) Now perform a small model-based bootstrap to see how well `auto.arima` can select the true model. To do this, simulate eight data sets from the ARIMA model selected in part (a) of this problem. Apply `auto.arima` with BIC to each of these data sets. How often is the "correct" amount of differencing selected, that is, d and D are correctly selected? How often is the "correct" model selected? "Correct" means in agreement with the simulation model. "Correct model" means both the correct amount of differencing and the correct orders for all the seasonal and nonseasonal AR and MA components.
7. This exercise uses the `Tbrate` data set in the `Ecdat` package. In Section 9.16.1, nonseasonal models were fit. Now use `auto.arima` to find a seasonal model. Which seasonal model is selected by AIC and by BIC? Do you feel that a seasonal model is needed, or is a nonseasonal model sufficient?