13

Time Series Models: Further Topics

13.1 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 13.1a 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.

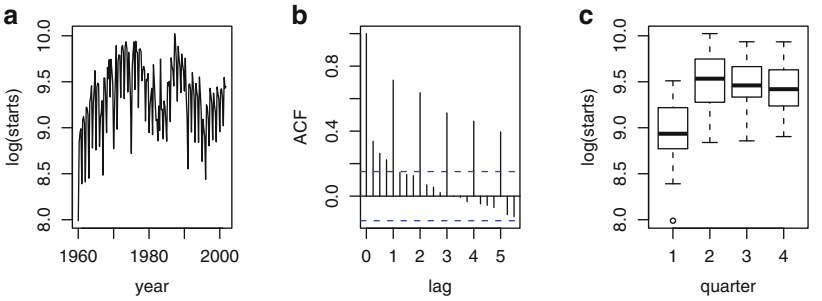

Fig. 13.1. *Logarithms of quarterly urban housing starts in Canada: (a) time series plot; (b) sample ACF; (c) boxplots by quarter.*

Figure 13.1 shows one and perhaps two types of nonstationarity: (1) There is strong seasonality, and (2) it is unclear whether the seasonal sub-series revert to a fixed mean and, if not, then this is a second type of nonstationarity because the process is integrated. These effects can also be seen in the ACF plot in Fig. 13.1b. At lags that are a multiples of four, the autocorrelations

D. Ruppert, D.S. Matteson, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics,
DOI 10.1007/978-1-4939-2614-5_13

are large, and decay slowly to zero. At other lags, the autocorrelations are smaller but also decay somewhat slowly. The boxplots in Fig. 13.1c 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.

13.1.1 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 Fig. 13.2.

To remove seasonal nonstationarity, 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 $\Delta^s = (1-B)^s$. Note that $\Delta_s = 1 - B^s$ is the first-order seasonal differencing operator and $\Delta^s = (1-B)^s$ is the sth-order nonseasonal differencing operator. For example, $\Delta_2 Y_t = Y_t - Y_{t-2}$ but $\Delta^2 Y_t = \Delta(\Delta Y_t) = Y_t - 2Y_{t-1} + Y_{t-2}$.

The series $\Delta_s Y_t$ is called the seasonally differenced series. See the middle row of Fig. 13.2 for the seasonally differenced logarithm of housing starts and its Sample ACF.

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

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

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.

13.1.2 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

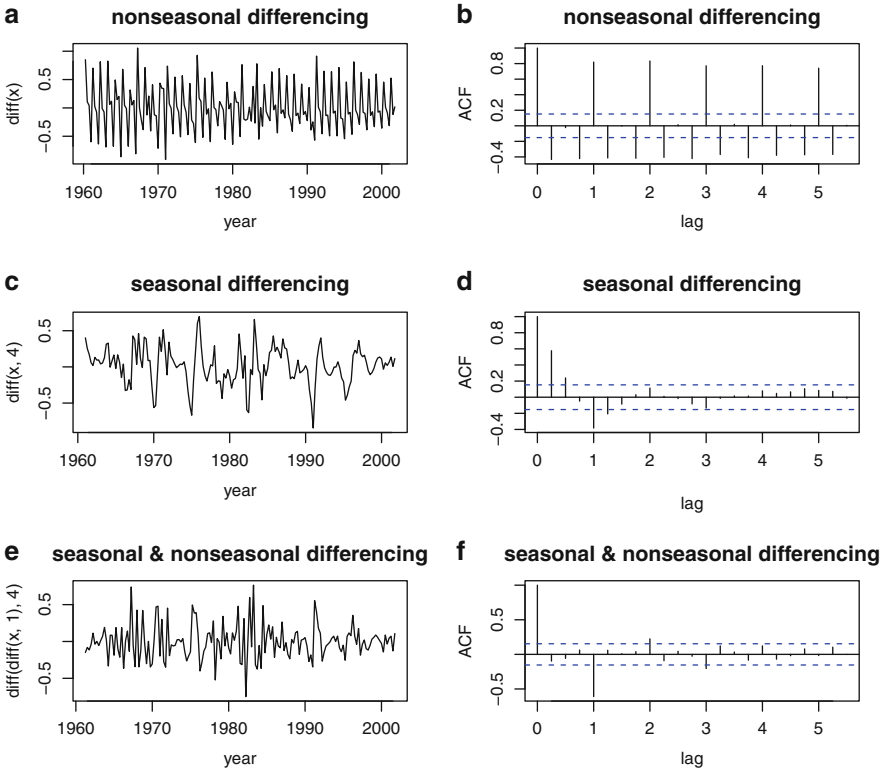

Fig. 13.2. *Time series (left column) and sample ACF plots (right column) of the logarithms of quarterly urban housing starts with nonseasonal differencing (top row), seasonal differencing (middle row), and both seasonal and nonseasonal differencing (bottom row). Note: in the sample ACF plots, lag = 1 means a lag of one year, which is four observations for quarterly data.*

$$(1 - \phi B)(\Delta Y_t - \mu) = \epsilon_t \tag{13.1}$$

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

$$(1 - \phi^* B^s)(\Delta_s Y_t - \mu) = \epsilon_t \tag{13.2}$$

to obtain the multiplicative model

$$(1 - \phi B)\,(1 - \phi^* B^s)\,\{\Delta(\Delta_s Y_t) - \mu\} = \epsilon_t. \tag{13.3}$$

Model (13.2) 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 (13.2) gives 12 independent processes, one for Januaries, a second for Februaries, and so forth. Model (13.3) uses the components from (13.1) to tie these 12 series together.

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

$$(1-\phi_1 B-\cdots-\phi_p B^p)\{1-\phi_1^* B^s-\cdots-\phi_{p_s}^*(B^s)^{p_s}\}\{\Delta^d(\Delta_s^{d_s}Y_t)-\mu\}$$
$$=(1+\theta_1 B+\ldots+\theta_q B^q)\{1+\theta_1^* B^s+\ldots+\theta_{q_s}^*(B^s)^{q_s}\}\epsilon_t. \quad (13.4)$$

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 13.1. 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 Fig. 13.2 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 below.

```
data(Hstarts, package="Ecdat")
x = ts(Hstarts[,1], start=1960, frequency=4)
fit2 = arima(x, c(1,1,1), seasonal = list(order = c(0,1,1),
  period = 4))
fit2
```

```
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 with mean zero and variance 0.0261. Figure 13.3 shows forecasts from this model for the four years following the end of the time series. □

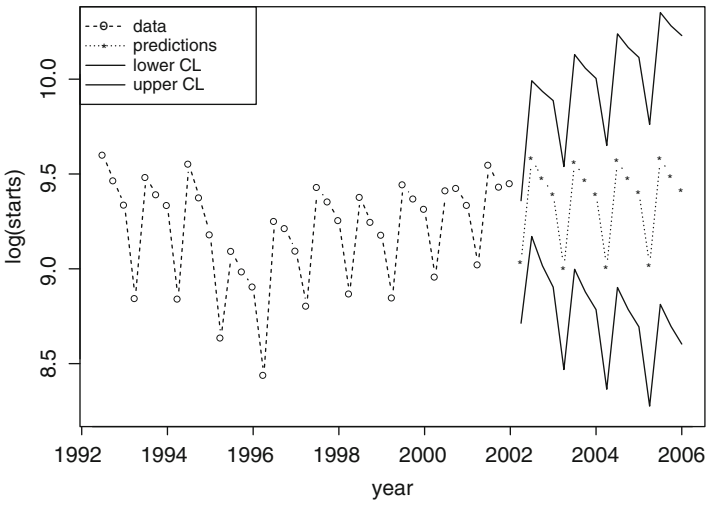

Fig. 13.3. *Forecasting logarithms of quarterly urban housing starts using the ARIMA*$\{(1,1,1) \times (0,1,1)_4\}$ *model. The dashed line connects the data, the dotted line connects the forecasts, and the solid lines are the forecast limits.*

When the size of the seasonal oscillations increases, as with the air passenger data in Fig. 12.2, some type of preprocessing is needed before differencing. Often, taking logarithms stabilizes the size of the oscillations. This can be seen in Fig. 13.4. Box, Jenkins, and Reinsel (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.

13.2 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

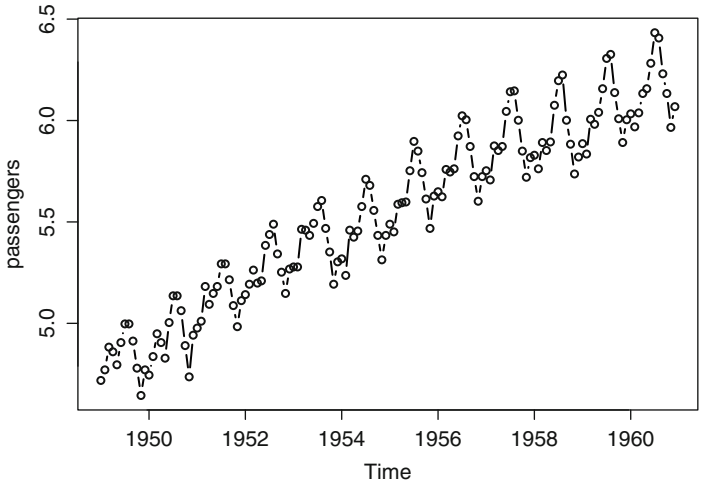

Fig. 13.4. *Time series plot of the logarithms of the monthly total international airline passengers (in thousands).*

$$(\Delta^d Y_t^{(\alpha)} - \mu) = \phi_1(\Delta^d Y_{t-1}^{(\alpha)} - \mu) + \cdots + \phi_p(\Delta^d Y_{t-p}^{(\alpha)} - \mu) + \epsilon_t + \theta_1\epsilon_{t-1} + \cdots + \theta_q\epsilon_{t-q}, \tag{13.5}$$

where $\epsilon_1, \epsilon_2, \ldots$ is Gaussian white noise. Model (13.5) 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, (13.5) uses a nonseasonal model, but a seasonal ARIMA model could just as easily have been used.

Example 13.2. Selecting a transformation for the housing starts

Figure 13.5 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 13.1 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. □

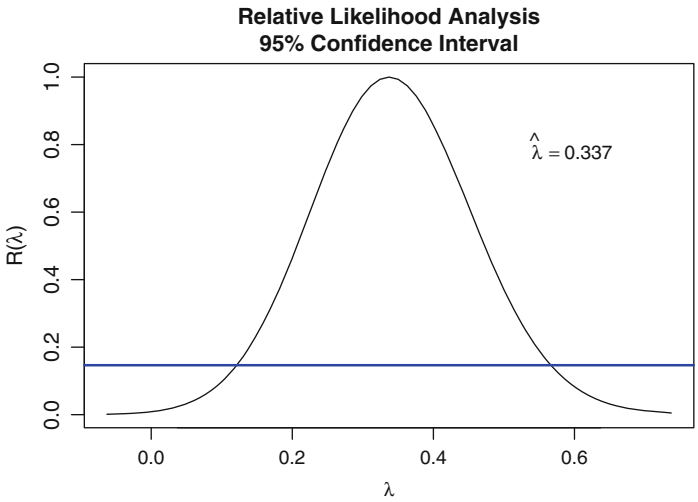

Fig. 13.5. *Profile likelihood for α (called λ in the legend) in the housing start example. Values of λ with $R(\lambda)$ (the profile likelihood) above the horizontal line are in the 95 % confidence limit.*

Data transformations can stabilize some types of variation in time series, but not all types. For example, in Fig. 12.2 the seasonal oscillations in the numbers of air passengers increase as the series itself increases, and we can see in Fig. 13.4 that a log transformation stabilizes these oscillations. In contrast, the S&P 500 returns in Fig. 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, these changes of volatility could be modeled by a GARCH process; this topic is pursued in Chap. 14.

13.3 Time Series and Regression

In a multiple linear regression model (9.1) the errors ϵ_i are assumed to be mutually independent. However, if the data $\{(\boldsymbol{X}_i, Y_i),\ i = 1, \ldots, n\}$ are time series, then it is likely that the errors are correlated, a problem we will call *residual correlation*; see Sect. 13.3.1.

Residual correlation causes standard errors and confidence intervals (which incorrectly assume uncorrelated noise) to be incorrect. In particular, the coverage probability of confidence intervals can be much lower than the nominal value. A solution to this problem is to adjust or correct the estimated covariance matrix of the coefficient estimates; see Sect. 13.3.2. An alternative solution is to model the noise as an ARMA process, assuming that the residuals are stationary; see Sect. 13.3.3.

13.3.1 Residual Correlation and Spurious Regressions

In the extreme case where the residuals are an integrated process, the least-squares estimator is inconsistent, meaning that it will not converge to the true parameter as the sample size converges to ∞. If an $I(1)$ process is regressed on another $I(1)$ process and the two processes are independent (so that the regression coefficient is 0), it is quite possible to obtain a highly significant result, that is, to strongly reject the true null hypothesis that the regression coefficient is 0. This is called a *spurious regression*. The problem, of course, is that the test is based on the incorrect assumption of independent error. The residuals from the The problem of correlated noise can be detected by looking at the sample ACF of the residuals. Sometimes the presence of residual correlation is obvious. In other cases, it is not so clear and a statistical test is desirable. The Durbin–Watson test can be used to test the null hypothesis of no residual autocorrelation. More precisely, the null hypothesis of the Durbin–Watson test is that the first p autocorrelation coefficients are all 0, where p can be selected by the user. The p-value for a Durbin–Watson test is not trivial to compute, and different implementations use different computational methods. In the `R` function `durbinWatsonTest()` in the `car` package, p is called `max.lag` and has a default value of 1. The p-value is computed by `durbinWatsonTest()` using bootstrapping. The `lmtest` package of `R` has another function, `dwtest()`, that computes the Durbin–Watson test, but only with $p = 1$. The function `dwtest()` uses either a normal approximation (default) or an exact algorithm to calculate the p-value.

Example 13.3. Residual plots for weekly interest changes

Using the interest rate data from Chap. 9, Fig. 13.6 contains residual plots for the regression of `aaa_dif` on `cm10_dif` and `cm30_dif`. The normal plot in panel (a) shows heavy tails. A t-distribution was fit to the residuals, and the estimated degrees of freedom was 2.99, again indicating heavy tails. Panel (b) shows a QQ plot of the residuals and the quantiles of the fitted t-distribution with a 45° reference line. There is excellent agreement between the data and the t-distribution.

Panel (c) is a plot of the ACF of the residuals. There is some evidence of autocorrelation. The Durbin–Watson test was performed three times with

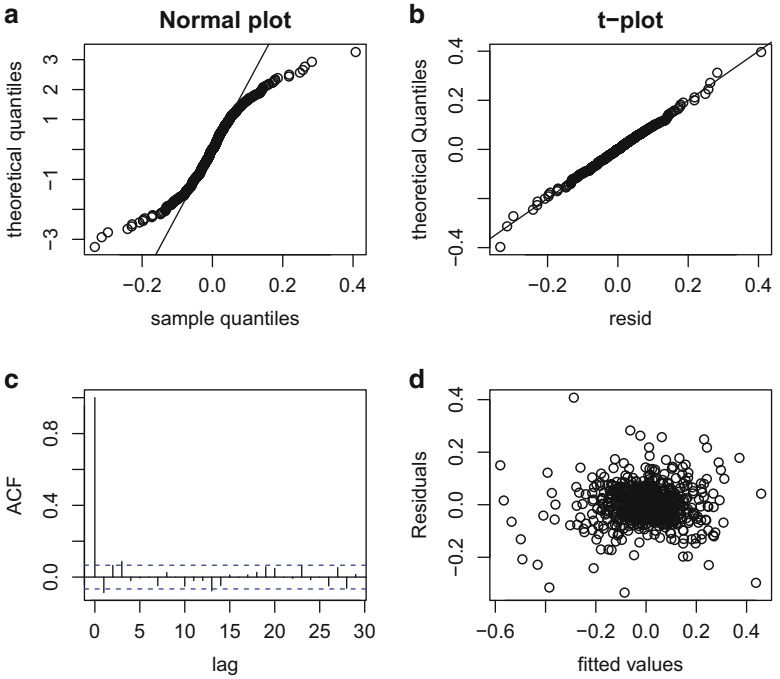

Fig. 13.6. *Residual plots for the regression of* `aaa_dif` *on* `cm10_dif` *and* `cm30_dif`.

R's `durbinWatsonTest()` using `max.lag` =1 and gave p-values of 0.006, 0.004, and 0.012. This shows the substantial random variation due to bootstrapping with the default of $B = 1000$ resamples. Using a larger number of resamples will compute the p-value with more accuracy. For example, when the number of resamples was increased to 10,000, three p-values were 0.0112, 0.0096, and 0.0106. Using `dwtest()`, the approximate p-value was 0.01089 and the exact p-value could not be computed. Despite some uncertainty about the p-value, it is clear that the p-value is small, so there is at least some residual autocorrelation.

To further investigate autocorrelation, ARMA models were fit to the residuals using the `auto.arima()` function in R to automatically select the order. Using BIC, the selected model is ARIMA(0,0,0), that is, white noise. Using AIC, the selected model is ARIMA(0,0,3) with estimates:

```
6 auto.arima(resid, ic="aic")

  Series: resid
  ARIMA(0,0,3) with zero mean

  Coefficients:
            ma1     ma2     ma3
        -0.0857  0.0770  0.0888
```

```
s.e.   0.0336  0.0338  0.0342

sigma^2 estimated as 0.004075:  log likelihood=1172.54
AIC=-2337.09   AICc=-2337.04   BIC=-2317.97
```

Several of the coefficients are large relative to their standard errors. There is evidence of some autocorrelation, but not a great deal and the BIC-selected model does not have any autocorrelation. The sample size is 880, so there are enough data to detect small autocorrelations. The autocorrelation that was found seems of little practical significance and could perhaps be ignored; see Sect. 13.3.2 for further investigation. The plot of residuals versus fitted values in panel (d) shows no sign of heteroskedasticity. □

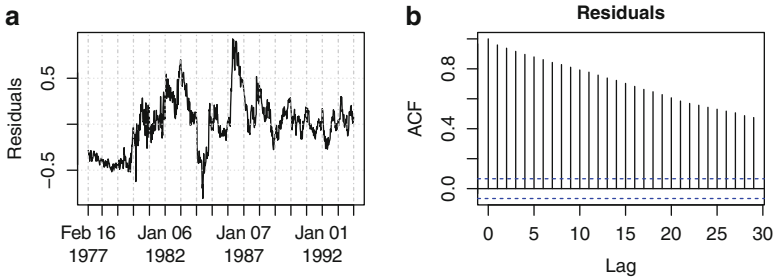

Fig. 13.7. *Time series plot and ACF plot of residuals when* `aaa` *is regressed on* `cm10` *and* `cm30`*. The plots indicate that the residuals are nonstationary.*

Example 13.4. Residual plots for weekly interest rates without differencing

The reader may have noticed that differenced time series have been used in the examples. There is a good reason for this. Many, if not most, financial time series are nonstationary or, at least, have very high and long-term autocorrelation. When one nonstationary series is regressed upon another, it happens frequently that the residuals are nonstationary. This is a substantial violation of the assumption of uncorrelated noise and can lead to serious problems. An estimator is said to be consistent if it converges to the true value of the parameter as the sample size increases to ∞. The least-squares estimator is not consistent when the errors are an integrated process.

As an example, we regressed `aaa` on `cm10` and `cm30`. These are the weekly time series of AAA, 10-year Treasury, and 30-year Treasury interest rates, which, when differenced, gave us `aaa_dif`, `cm10_dif`, and `cm30_dif` used in previous examples. Figure 13.7 contains time series and ACF plots of the residuals. The residuals are very highly correlated and perhaps are nonstationary. Unit root tests provide more evidence that the residuals are nonstationary. The p-values of augmented Dickey–Fuller tests are on one side of 0.05 or the

other, depending on the order. With the default lag order in the `adf.test()` function from the `tseries` package in `R`, the p-value is 0.12, so one would not reject the null hypothesis of nonstationarity at level 0.05 or even level 0.1. The `kpss.test()` function does reject the null hypothesis of stationarity.

Let us compare the estimates from regression using the original series with the estimates from the differenced series. First, what should we expect when we make this comparison? Suppose that X_t and Y_t are time series following the regression model

$$Y_t = \alpha + \beta_0 t + \beta_1 X_i + \epsilon_t. \tag{13.6}$$

Note the linear time trend $\beta_0 t$. Then, upon differencing, we have

$$\Delta Y_t = \beta_0 + \beta_1 \Delta X_i + \Delta \epsilon_t, \tag{13.7}$$

so the original intercept α is removed, and the time trend's slope β_0 in (13.6) becomes an intercept in (13.7). The time trend could be omitted in (13.6) if the intercept in (13.7) is not significant, as happens in this example. The slope β_1 in (13.6) remains unchanged in (13.7). However, if ϵ_t is $I(1)$, then the regression of Y_t on X_t will not provide a consistent estimate of β_1, but the regression of ΔY_t on ΔX_i will consistently estimate β_1, so the estimates from the two regressions could be very different. This is what happens with this example.

The results from regression with the original series without the time trend are

```
Call:
lm(formula = aaa ~ cm10 + cm30)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9803     0.0700   14.00  < 2e-16 ***
cm10          0.3183     0.0445    7.15  1.9e-12 ***
cm30          0.6504     0.0498   13.05  < 2e-16 ***
```

The results with the differenced series are

```
Call:
lm(formula = aaa_dif ~ cm10_dif + cm30_dif)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.38e-05   2.18e-03   -0.04     0.97
cm10_dif     3.60e-01   4.45e-02    8.09  2.0e-15 ***
cm30_dif     2.97e-01   4.98e-02    5.96  3.7e-09 ***
```

The estimated slopes for `cm10` and `cm10_dif`, 0.3183 and 0.360, are somewhat similar. However, the estimated slopes for `cm30` and `cm30_dif`, 0.650 and 0.297, are quite dissimilar relative to their standard errors. This is to

be expected if the estimators using the undifferenced series are not consistent; also, their standard errors are not valid because they are based on the assumption of uncorrelated noise. In the analysis with the differenced data, the p-value for the intercept is 0.97, so we can accept the null hypothesis that the intercept is zero; this justifies the omission of the time trend when using the undifferenced series. □

Example 13.5. Simulated independent AR processes

To illustrate further the problems caused by regressing nonstationary series, or even stationary series with high correlation, we simulated two independent AR process, both of length 200 with $\phi = 0.99$.

```
set.seed(997711)
n = 200
x = arima.sim(list(order=c(1,0,0),ar=.99),n=n)
y = arima.sim(list(order=c(1,0,0),ar=.99),n=n)
fit1 = lm(y~x)
fit5 = lm(diff(y)~diff(x))
```

These processes are stationary but near the borderline of being nonstationary. After simulating these processes, one process was regressed on the other. We repeated this three more times. Since the processes are independent, the true slope is 0. In each case, the estimated slope was far from the true value of 0 and was statistically significant according to the (incorrect) p-value. The results are below.

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -8.40      0.269     -31  1.9e-78
x               0.48      0.036      13  1.6e-29

             Estimate Std. Error t value Pr(>|t|)
(Intercept)     5.96      0.328    18.2  4.9e-44
x              -0.43      0.088    -4.8  2.6e-06

             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -5.154      0.213   -24.2  4.5e-61
x              0.095      0.031     3.1  2.3e-03

             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.51      0.312    -1.6  1.1e-01
x              -0.53      0.079    -6.7  2.3e-10
```

Notice how the estimated intercepts and slope randomly vary between the four simulations. The standard errors and p-values are based on the invalid assumption of independent errors and are erroneous and very misleading, a problem that is called *spurious regression*. Fortunately, the violation of the independence assumption would be easy to detect by plotting the residuals.

We also regressed the differenced series and obtained completely different results:

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.082      0.069    1.18     0.24
diff(x)        -0.023      0.068   -0.34     0.73

             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.027      0.064   -0.41     0.68
diff(x)        -0.021      0.063   -0.33     0.74

             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.015      0.071   -0.21     0.83
diff(x)        -0.022      0.076   -0.29     0.77

             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.025      0.077   -0.32     0.75
diff(x)         0.022      0.078    0.28     0.78
```

Notice that now the estimated slopes are all near the true value of 0. All the p-values are large and lead one to the correct conclusion that the true slope is 0.

When the noise process is stationary, an alternative to differencing is to use an ARMA model for the noise process; see Sect. 13.3.3. □

13.3.2 Heteroscedasticity and Autocorrelation Consistent (HAC) Standard Errors

We now consider the effect of correlated noise and heteroskedasticity on standard errors and confidence intervals in multiple linear regression models. If $\mathrm{COV}(\boldsymbol{\epsilon}) \neq \sigma_\epsilon^2 \boldsymbol{I}$ but rather $\mathrm{COV}(\boldsymbol{\epsilon}) = \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ for some matrix $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$, then

$$\begin{aligned} \mathrm{COV}(\widehat{\boldsymbol{\beta}}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) &= (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\mathrm{COV}(\boldsymbol{Y}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1} \\ &= (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}. \end{aligned} \tag{13.8}$$

This result lets us see the effect of correlation or nonconstant variance among $\epsilon_1, \ldots, \epsilon_n$.

Example 13.6. Regression with AR(1) errors

Suppose that $\epsilon_1, \ldots, \epsilon_n$ is a stationary AR(1) process so that $\epsilon_t = \phi\epsilon_{t-1} + u_t$, where $|\phi| < 1$ and $u_1, u_2, \ldots$ is weak WN$(0, \sigma_u^2)$. Then

$$\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} = \sigma_\epsilon^2 \begin{pmatrix} 1 & \phi & \phi^2 & \cdots & \phi^{n-1} \\ \phi & 1 & \phi & \cdots & \phi^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi^{n-1} & \phi^{n-2} & \phi^{n-3} & \cdots & 1 \end{pmatrix}. \tag{13.9}$$

As an example, suppose that $n = 21$, $X_1, \ldots, X_n$ are equally spaced between -10 and 10, and $\sigma_\epsilon^2 = 1$. Substituting (13.9) into (13.8) gives the covariance matrix of the estimator $(\widehat{\beta}_0, \widehat{\beta}_1)$, and taking the square roots of the diagonal elements gives the standard errors. This was done with $\phi = -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75$.

Figure 13.8 plots the ratios of standard errors for the independent case ($\phi = 0$) to the standard errors for the true value of ϕ. These ratios are the factors by which the standard errors are miscalculated if we assume that $\phi = 0$, but it is not. Notice that negative values of ϕ result in a conservative (too large) standard error, but positive values of ϕ give a standard error that is too small. In the case of $\phi = 0.75$, assuming independence gives standard errors that are only about half as large as they should be. As discussed in Sect. 13.3.3, this problem can be fixed by assuming (correctly) that the noise process is AR(1). □

As discussed in Sect. 13.3.1, if the errors in a regression model are an integrated process, such as a random walk, the least-squares estimator is inconsistent. However, when the dependence between the errors is not too strong, there are mild conditions under which the least-squares estimator is consistent, meaning that it will converge to the true parameter as the sample size converges to ∞. For the latter case there are methods available to estimate consistent standard errors for the coefficient estimates. Two simple and widely used approaches are the heteroskedasticity consistent (HC) and heteroskedasticity and autocorrelation consistent (HAC) estimators.

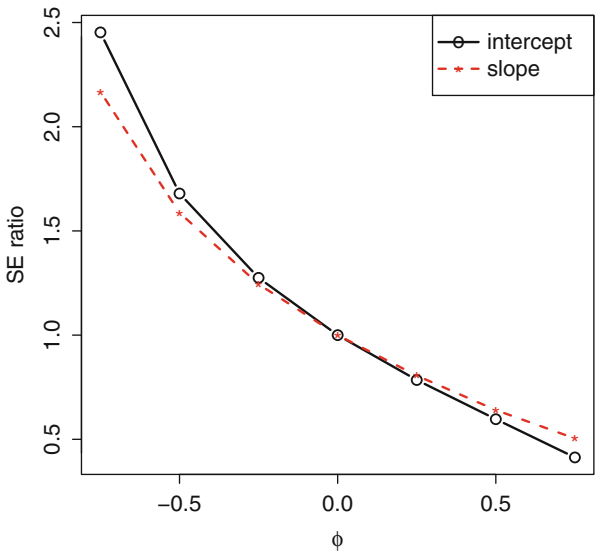

Fig. 13.8. *Factor by which the standard error is changed when ϕ deviates from 0 for intercept (solid) and slope (dashed).*

Let $\hat{\epsilon}_i$ denote the OLS residuals, let $\widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\epsilon}}} = \text{diag}\{\hat{\epsilon}_1^2, \ldots, \hat{\epsilon}_n^2\}$ denote a diagonal matrix of the squared residuals, and let $\widehat{\boldsymbol{C}}_{HC} = \boldsymbol{X}^\mathsf{T}\widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\epsilon}}}\boldsymbol{X}$. Then a heteroskedasticity consistent (HC) estimator (see White, 1980) of the covariance matrix for the coefficient estimates is

$$\widehat{\text{COV}}_{HC}(\widehat{\boldsymbol{\beta}}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\widehat{\boldsymbol{C}}_{HC}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}. \tag{13.10}$$

The corresponding HC standard errors are defined as the square roots of the diagonal entries of (13.10).

A heteroskedasticity and autocorrelation consistent (HAC) estimator (see Newey and West, 1987) of the covariance matrix for the coefficient estimates is similarly defined as

$$\widehat{\text{COV}}_{HAC}(\widehat{\boldsymbol{\beta}}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\widehat{\boldsymbol{C}}_{HAC}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}, \tag{13.11}$$

in which

$$\widehat{\boldsymbol{C}}_{HAC} = \widehat{\boldsymbol{C}}_{HC} + \sum_{\ell=1}^{L} w_\ell \sum_{i=\ell+1}^{n} \left(\boldsymbol{X}_i\hat{\epsilon}_i\hat{\epsilon}_{i-\ell}\boldsymbol{X}_{i-\ell}^\mathsf{T} + \boldsymbol{X}_{i-\ell}\hat{\epsilon}_{i-\ell}\hat{\epsilon}_i\boldsymbol{X}_i^\mathsf{T}\right), \tag{13.12}$$

where $w_\ell = 1 - \ell/(L+1)$ denotes the Bartlett weight function, although other weight functions w_ℓ can also be used. The corresponding HAC standard errors are defined as the square root of the diagonal entries of (13.11).

Example 13.7. HC and HAC estimates for regression of weekly interest changes

In Sect. 13.3.1 a regression of `aaa_dif` on `cm10_dif` and `cm30_dif` produced residuals that exhibited minor autocorrelation; AIC suggested an MA(3) model for the residuals while BIC selected ARIMA(0,0,0), i.e., white noise. We now consider whether ignoring the small autocorrelations has a practical impact on inference. The previous regression results are obtained from the following `R` commands.

```
dat = read.table(file="WeekInt.txt", header=T)
attach(dat)
cm10_dif = diff(cm10)
aaa_dif = diff(aaa)
cm30_dif = diff(cm30)
fit = lm(aaa_dif ~ cm10_dif + cm30_dif)
round(summary(fit)$coef, 4)
```

The HC and HAC covariance matrix estimates can be computed using the `NeweyWest()` function from the `R` package `sandwich`. The first argument is a fitted model object, in this case `fit`. In both cases we set `prewhite = F`. For the HAC estimate, the argument `lag` corresponds to the maximal lag L used in the Bartlett weight function above. If no value is specified, one is selected automatically via the `bwNeweyWest()` function (see the help file for more information). For the HC estimate we specify `lag = 0`. The HC estimate and HAC estimate with $L = 3$ are shown below.

```
library(sandwich)
options(digits=2)
NeweyWest(fit, lag = 0, prewhite = F)
```

```
             (Intercept) cm10_dif cm30_dif
(Intercept)      4.7e-06  7.3e-06 -1.1e-05
cm10_dif         7.3e-06  6.3e-03 -6.2e-03
cm30_dif        -1.1e-05 -6.2e-03  6.7e-03
```

```
NeweyWest(fit, lag = 3, prewhite = F)
```

```
             (Intercept) cm10_dif cm30_dif
(Intercept)      4.6e-06 -0.00003  2.6e-05
cm10_dif        -3.0e-05  0.00666 -6.6e-03
cm30_dif         2.6e-05 -0.00662  7.0e-03
```

The OLS regression results, as well as the HC and HAC estimated standard errors, and their corresponding t values are summarized in Table 13.1. Recall that the HC and HAC standard error estimates are computed as the square roots of the diagonal entries of the covariance matrix estimates.

```
sqrt(diag(NeweyWest(fit, lag = 0, prewhite = F)))
sqrt(diag(NeweyWest(fit, lag = 3, prewhite = F)))
```

The corresponding t values are the OLS coefficient estimates divided by their standard error estimates.

```
coef(fit)/sqrt(diag(NeweyWest(fit, lag = 0, prewhite = F)))
coef(fit)/sqrt(diag(NeweyWest(fit, lag = 3, prewhite = F)))
```

Table 13.1. *Regression estimates of* `aaa_dif` *on* `cm10_dif` *and* `cm30_dif`*: the OLS estimates, estimated standard errors, and t values are shown on the left; the estimated HC standard errors and corresponding t values are shown in the middle; and the estimated HAC standard errors with* $L = 3$ *and corresponding t values are shown on the right.*

	OLS			HC		$\text{HAC}_{L=3}$	
Coefficient	Estimate	Std. Err.	t value	Std. Err.	t value	Std. Err.	t value
(Intercept)	−0.0001	0.0022	−0.043	0.0022	−0.043	0.0021	−0.044
cm10_dif	0.3602	0.0445	8.091	0.0791	4.553	0.0816	4.415
cm30_dif	0.2968	0.0498	5.956	0.0816	3.637	0.0836	3.551

From Table 13.1 we see that the HC and HAC estimates produced similar results. The estimated standard error for the intercept are stable, while the estimated HC and HAC standard errors for the `cm10_dif` and `cm30_dif` coefficients are about twice as large as the OLS estimates of the standard errors, and as a result, the corresponding t values are about half as large in magnitude. In this case, however, the `cm10_dif` and `cm30_dif` coefficients remain statistically significant, with both estimates over three standard errors above zero. The minor serial correlation (and heteroskedasticity) in the OLS residuals does not appear to have a practical impact on inference in this example. □

13.3.3 Linear Regression with ARMA Errors

When residual analysis shows that the residuals are correlated, then one of the key assumptions of the linear model does not hold, and tests and confidence intervals based on this assumption are invalid and cannot be trusted. Fortunately, there is a solution to this problem: replace the assumption of independent noise by the weaker assumption that the noise process is stationary but possibly correlated. One could, for example, assume that the noise is an ARMA process. This is the strategy we will discuss in this section; this approach is referred to as an ARMAX model, in which the X indicates the inclusion of exogenous regression variables.

The linear regression model with ARMA errors combines the linear regression model (9.1) and the ARMA model (12.26) for the noise, so that

$$Y_t = \beta_0 + \beta_1 X_{t,1} + \cdots + \beta_p X_{t,p} + \epsilon_t, \tag{13.13}$$

where

$$(1 - \phi_1 B - \cdots - \phi_p B^p)\,\epsilon_t = (1 + \theta_1 B + \cdots + \theta_q B^q)\,u_t, \tag{13.14}$$

and $u_1, \ldots, u_n$ is white noise.

Example 13.8. Demand for ice cream

This example uses the data set `Icecream` in R's `Ecdat` package. The data are four-weekly observations from March 18, 1951, to July 11, 1953 on four variables, `cons` = U.S. consumption of ice cream per head in pints; `income` = average family income per week (in U.S. Dollars); `price` = price of ice cream (per pint); and `temp` = average temperature (in Fahrenheit). There is a total of 30 observations. Since there are 13 four-week periods per year, there are slightly over two years of data.

First, a linear model was fit with `cons` as the response and `income`, `price`, and `temp` as the predictor variables. One can see that `income` and `temp` are significant, especially `temp` (not surprisingly).

```
Call:
lm(formula = cons ~ income + price + temp, data = Icecream)

Residuals:
     Min       1Q   Median       3Q      Max
-0.06530 -0.01187  0.00274  0.01595  0.07899

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.197315   0.270216    0.73    0.472
income       0.003308   0.001171    2.82    0.009 **
price       -1.044414   0.834357   -1.25    0.222
```

```
temp         0.003458   0.000446    7.76  3.1e-08 ***
---

Residual standard error: 0.0368 on 26 degrees of freedom
Multiple R-squared: 0.719,      Adjusted R-squared: 0.687
F-statistic: 22.2 on 3 and 26 DF,  p-value: 2.45e-07
```

A Durbin–Watson test has a very small p-value, so we can reject the null hypothesis that the noise is uncorrelated.

```
options(digits=3)
library("car")
durbinWatsonTest(fit_ic_lm)
```

```
 lag Autocorrelation D-W Statistic p-value
   1            0.33          1.02       0
 Alternative hypothesis: rho != 0
```

Next, the linear regression model with AR(1) errors was fit and the AR(1) coefficient was over three times its standard error, indicating statistical significance. This was done using R's `arima()` function, which specifies the regression model with the `xreg` argument. It is interesting to note that the coefficient of `income` is now nearly equal to 0 and no longer significant. The effect of `temp` is similar to that of the linear model fit, though its standard error is now larger.

```
Series: cons
ARIMA(1,0,0) with non-zero mean

Coefficients:
        ar1  intercept  income   price   temp
      0.732      0.538   0.000  -1.086  0.003
s.e.  0.237      0.325   0.003   0.734  0.001

sigma^2 estimated as 0.00091:  log likelihood=62.1
AIC=-112   AICc=-109   BIC=-104
```

Finally, the linear regression model with MA(1) errors was fit and the MA(1) coefficient was also over three times its standard error, again indicating statistical significance. The model with AR(1) errors has a slightly better (smaller) AIC and BIC values than the model with MA(1), but there is not much of a difference between the models in terms of AIC or BIC. However, the two models imply rather different types of noise autocorrelation. The MA(1) model has no correlation beyond lag 1. The AR(1) model with coefficient 0.732 has autocorrelation persisting much longer. For example, the autocorrelation is $0.732^2 = 0.536$ at lag 2, $0.732^3 = 0.392$ at lag 3, and still $0.732^4 = 0.287$ at lag 4.

```
Series: cons
ARIMA(0,0,1) with non-zero mean
```

```
Coefficients:
        ma1  intercept  income   price   temp
      0.503      0.332   0.003  -1.398  0.003
s.e.  0.160      0.270   0.001   0.798  0.001

sigma^2 estimated as 0.000957:  log likelihood=61.6
AIC=-111   AICc=-107   BIC=-103
```

Interestingly, the estimated effect of `income` is larger and significant, much like its effect as estimated by the linear model with independent errors but unlike the result for the linear model with AR(1) errors.

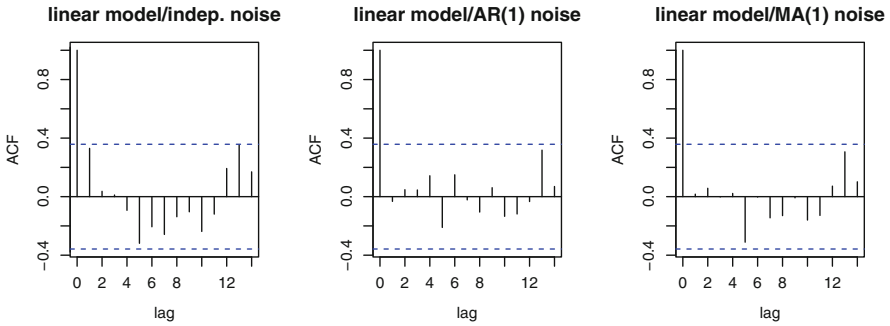

Fig. 13.9. *Ice cream consumption example. Residual ACF plots for the linear model with independent noise, the linear model with AR(1) noise, and the linear model with MA(1) noise.*

The ACFs of the residuals from the linear model and from the linear models with AR(1) and MA(1) errors are shown in Fig. 13.9. The residuals from the linear model estimate $\epsilon_1, \ldots, \epsilon_n$ in (13.13), and show some autocorrelation. The residuals from the linear models with either AR(1) or MA(1) errors estimate $u_1, \ldots, u_n$ in (13.14), and show little autocorrelation. One concludes that the linear model with either AR(1) or MA(1) errors fits well and either an AR(1) or MA(1) term is needed.

Why is the effect of `income` larger and significant if the noise is assumed to be either independent or MA(1) but smaller and insignificant if the noise is AR(1)? To attempt an answer, time series plots of the four variables were examined. The plots are shown in Fig. 13.10. The strong seasonal trend in temp is obvious and `cons` follows this trend. There is a slightly increasing trend in `cons`, which appears to have two possible explanations. The trend might be explained by the increasing trend in `income`. However, with the strong residual autocorrelation implied by the AR(1) model, the trend in `cons` could also be explained by noise autocorrelation. One problem here is that we have a small sample size, only 30 observations. With more data it might be possible to separate the effects on ice cream consumption of income and noise autocorrelation.

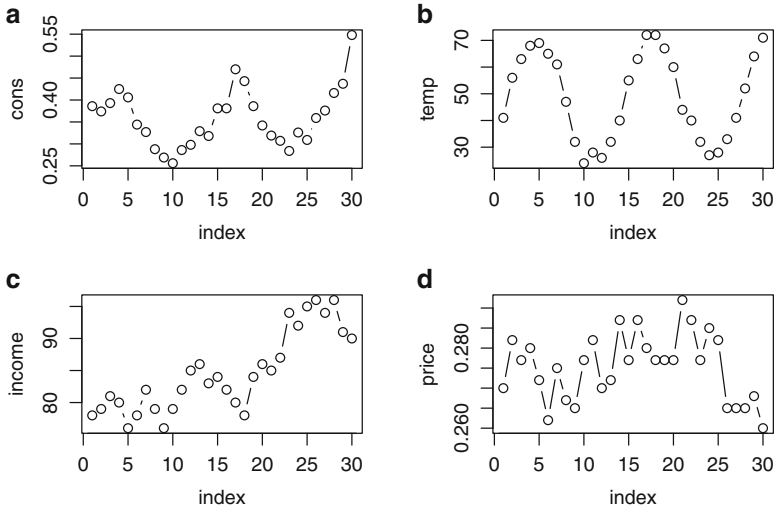

Fig. 13.10. *Time series plots for the ice cream consumption example and the variables used to predict consumption.*

In summary, there is a strong seasonal component to ice cream consumption, with consumption increasing, as would be expected, with warmer temperatures. Ice cream consumption does not depend much, if at all, on `price`, though it should be noted that `price` has not varied much in this study; see Fig. 13.10. Greater variation in `price` might cause `cons` to depend more on `price`. Finally, it is uncertain whether ice cream consumption increases with family income. □

13.4 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 is 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.

13.4.1 The Cross-Correlation Function

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

$$\rho_{Y_j,Y_i}(h) = \text{Corr}\{Y_j(t), Y_i(t-h)\} \tag{13.15}$$

and is the correlation between Y_j at a time t and Y_i at h time units earlier. As with autocorrelation, h is called the *lag*. However, unlike the ACF, the CCF is not symmetric in the lag variable h, that is, $\rho_{Y_j,Y_i}(h) \neq \rho_{Y_j,Y_i}(-h)$. Instead, as a direct consequence of definition (13.15), we have that $\rho_{Y_j,Y_i}(h) = \rho_{Y_i,Y_j}(-h)$.

The CCF can be defined for multivariate time series that are not stationary, but only weakly stationary. A multivariate time series $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots$ is said to be weakly stationary if the mean and covariance matrix of $\boldsymbol{Y}_t$ are finite and do not depend on t, and if the right-hand side of (13.15) is independent of t for all j, i, and h.

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 a causal relationship might be deduced from other knowledge.

Example 13.9. Cross-correlation between changes in CPI (consumer price index) and IP (industrial production)

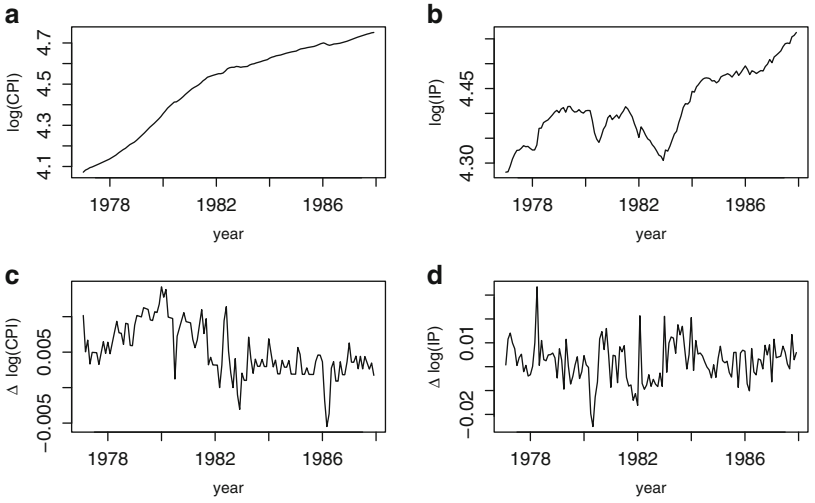

Fig. 13.11. *(**a**) Time series plot of log(CPI) (**b**) Time series plot of log(IP) (**c**) Time series plot of changes in log(CPI) (**d**) Time series plot of changes in log(IP).*

Time series plots for the logarithm of CPI (*cpi*), the logarithm of IP (*ip*), and changes in *cpi* and *ip*, are shown in Fig. 13.11 panels (a)–(d), respectively. The cross-correlation function between changes in the logarithm of CPI (Δcpi) and changes in the logarithm of IP (Δip) is shown in Fig. 13.12. It was created by the `ccf()` function in `R`.

```
CPI.dat = read.csv("CPI.dat.csv")
CPI_diff1 = diff(log(as.matrix(CPI.dat$CPI)[769:900,])) # 1977--1987
```

```
IP.dat = read.csv("IP.dat.csv")
IP_diff1 = diff(log(as.matrix(IP.dat$IP)[697:828,]))    # 1977--1987
ccf(CPI_diff1, IP_diff1)
```

The largest absolute cross-correlations are at negative 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 Sect. 13.4.5. □

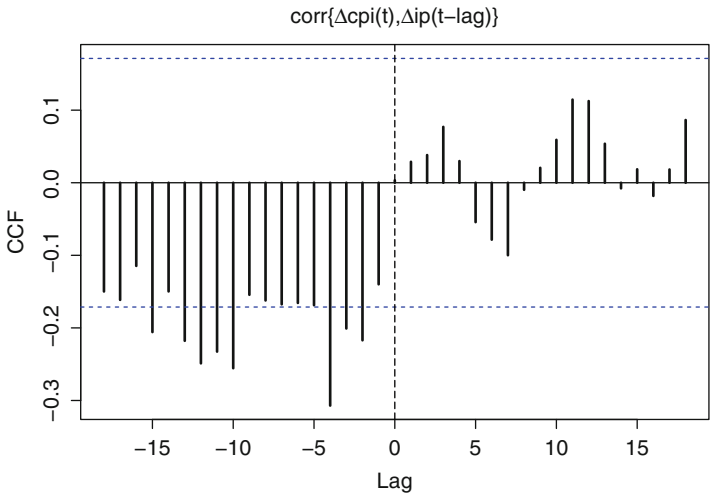

Fig. 13.12. *Sample CCF for Δcpi and Δip. Note the negative correlation at negative lags, that is, between the cpi and future values of ip.*

13.4.2 Multivariate White Noise

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

1. $E(\boldsymbol{Y}_t) = \boldsymbol{\mu}$ (constant and finite) for all t;
2. $\text{COV}(\boldsymbol{Y}_t) = \boldsymbol{\Sigma}$ (constant and finite) for all t; and
3. for all $t \neq s$, all components of $\boldsymbol{Y}_t$ are uncorrelated with all components of $\boldsymbol{Y}_s$.

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

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

13.4.3 Multivariate ACF Plots and the Multivariate Ljung-Box Test

The ACF for multivariate time series includes the d marginal ACFs for each univariate series $\{\rho_{Y_i}(h) : i = 1, \ldots, d\}$, and the $d(d-1)/2$ CCFs for all unordered pairs of the univariate series $\{\rho_{Y_j,Y_i}(h) : 1 \leq j < i \leq d\}$. It is sufficient to only consider the unordered pairs because $\rho_{Y_j,Y_i}(h) = \rho_{Y_i,Y_j}(-h)$.

In `R`, if two (or more) univariate time series with matching time indices are stored as $n \times 1$ vectors, the `cbind()` function can by used to make an $n \times 2$ matrix consisting of the joint time series. The `acf()` function may be applied to such multivariate time series. The sample ACF for $(\Delta cpi, \Delta ip)'$ is shown in Fig. 13.13, and generated by the following commands in `R`.

```
CPI_IP = cbind(CPI_diff1, IP_diff1)
acf(CPI_IP)
```

The marginal sample ACF for Δcpi and Δip are shown in the first and second diagonal panels, respectively. Both show significant serial correlation, but there is much more persistence in the first. The sample CCF for Δcpi and Δip has been split between the top right and bottom left panels by positive and negative lags, respectively. Notice that combining the off-diagonal panels in Fig. 13.13 reproduce the CCF shown in Fig. 13.12.

Each of the panels in Fig. 13.13 include *test bounds* to test the null hypothesis that an individual autocorrelation or lagged cross-correlation coefficient is 0. As in the univariate case, the usual level of the test is 0.05, and one can expect to see about 1 out of 20 sample correlations outside the test bounds simply by chance. Also, as in the univariate case, a simultaneous test is available.

Let $\boldsymbol{\rho}(h)$ denote the $d \times d$ lag-h cross-correlation matrix for a d-dimensional multivariate time series. The null hypothesis of the multivariate Ljung–Box test is $\text{H}_0 : \boldsymbol{\rho}(1) = \boldsymbol{\rho}(2) = \cdots = \boldsymbol{\rho}(K) = \mathbf{0}$ for some K, say $K = 5$ or 10. If the multivariate Ljung–Box test rejects, then we conclude that one or more of $\boldsymbol{\rho}(1), \cdots, \boldsymbol{\rho}(K)$ is nonzero. If, in fact, the lagged cross-correlation 1 to K are all zero, then there is only a 1 in 20 chance of falsely concluding that they are not all zero, assuming a level 0.05 test. In contrast, if the lagged cross-correlation are tested one at time, then there is a much higher chance of concluding that one or more is nonzero.

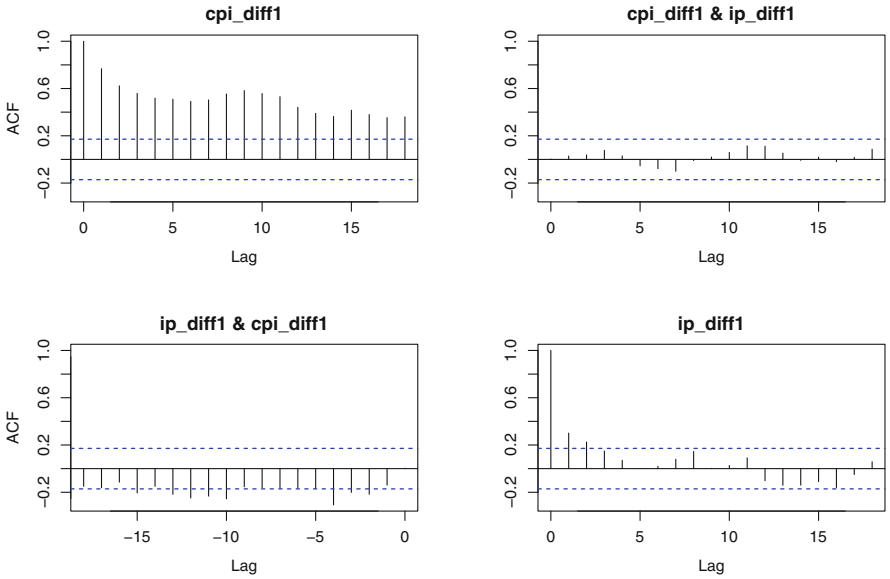

Fig. 13.13. *Sample ACF for* $(\Delta cpi, \Delta ip)'$*. The marginal sample ACF for* Δcpi *and* Δip *are shown in the first and second diagonal panels, respectively; the sample CCF for* Δcpi *and* Δip *has been split between the top right and bottom left panels by positive and negative lags, respectively.*

The following commands will conduct the multivariate Ljung–Box test in `R` for the bivariate series $(\Delta cpi, \Delta ip)'$.

```
source("SDAFE2.R")
mLjungBox(CPI_IP, lag = 10)
```

```
      K   Q(K) d.f. p-value
1 10 532.48   40       0
```

The multivariate Ljung–Box test statistic was 532.48, and the approximate p-value was 0, confirming that there is significant serial correlation in the first $K = 10$ lags.

13.4.4 Multivariate ARMA Processes

A d-dimensional multivariate time series $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots$ is a multivariate ARMA (p, q) process with mean $\boldsymbol{\mu}$ if for $d \times d$ matrices $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p$ and $\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_q$,

$$\boldsymbol{Y}_t - \boldsymbol{\mu} = \boldsymbol{\Phi}_1(\boldsymbol{Y}_{t-1} - \boldsymbol{\mu}) + \cdots + \boldsymbol{\Phi}_p(\boldsymbol{Y}_{t-p} - \boldsymbol{\mu}) + \boldsymbol{\epsilon}_t + \boldsymbol{\Theta}_1\boldsymbol{\epsilon}_{t-1} + \cdots + \boldsymbol{\Theta}_q\boldsymbol{\epsilon}_{t-q}, \tag{13.16}$$

where $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n$ is a multivariate weak WN$(\boldsymbol{0}, \boldsymbol{\Sigma})$ 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, $\phi_{i,j}^k$ (the (i,j)th 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 (13.16) with $p = 1$ and $\boldsymbol{\Phi} = \boldsymbol{\Phi}_1$ that

$$E(\boldsymbol{Y}_t | \boldsymbol{Y}_{t-1}, \ldots, \boldsymbol{Y}_1) = E(\boldsymbol{Y}_t | \boldsymbol{Y}_{t-1}) = \boldsymbol{\mu} + \boldsymbol{\Phi}(\boldsymbol{Y}_{t-1} - \boldsymbol{\mu}). \tag{13.17}$$

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 (13.17). To keep notation simple, assume that the mean has been subtracted from $\boldsymbol{Y}_t$ so that $\boldsymbol{\mu} = \mathbf{0}$. Then

$$\boldsymbol{Y}_t = \boldsymbol{\Phi}\boldsymbol{Y}_{t-1} + \boldsymbol{\epsilon}_t = \boldsymbol{\Phi}\{\boldsymbol{\Phi}\boldsymbol{Y}_{t-2} + \boldsymbol{\epsilon}_{t-1}\} + \boldsymbol{\epsilon}_t$$

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

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

By similar calculations,

$$E(\boldsymbol{Y}_t | \boldsymbol{Y}_{t-k}) = \boldsymbol{\Phi}^k \boldsymbol{Y}_{t-k}, \text{ for all } k > 0. \tag{13.18}$$

It can be shown using (13.18), 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 13.10. A bivariate AR model for Δcpi and Δip

This example uses the CPI and IP data sets discussed in earlier examples (*cpi* and *ip* denote the log transformed series). Bivariate AR processes were fit to $(\Delta cpi, \Delta ip)'$ 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.

```
CPI_IP = cbind(CPI_diff1,IP_diff1)
arFit = ar(CPI_IP,order.max=10)
options(digits=2)
arFit$aic
```

```
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 commands and results for fitting the bivariate AR(1) model are

```
arFit1 = ar(CPI_IP, order.max = 1) ; arFit1
```

with

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

and

$$\widehat{\boldsymbol{\Sigma}} = \begin{pmatrix} 5.68\text{e}-06 & 3.33\text{e}-06 \\ 3.33\text{e}-06 & 6.73\text{e}-05 \end{pmatrix}. \tag{13.19}$$

The function `ar()` does not estimate $\boldsymbol{\mu}$, but $\boldsymbol{\mu}$ can be estimated by the sample mean, which is $(0.0052, 0.0021)'$.

```
colMeans(CPI_IP)
```

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*), given the other variables in the model. 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 13.9.

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 $\boldsymbol{\Phi}$:

```
bPhi = arFit1$ar[,,] ; bPhi
bPhi2 = bPhi %*% bPhi ; bPhi2
bPhi3 = bPhi2 %*% bPhi ; bPhi3
bPhi4 = bPhi3 %*% bPhi ; bPhi4
bPhi5 = bPhi4 %*% bPhi ; bPhi5
```

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

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

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 latter, 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 $\Delta\, cpi$ and future $\Delta\, ip$, it follows from (13.19) 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 $\Delta\, cpi$ and $\Delta\, ip$ series, so we see that the unpredictable changes have positive correlation. In contrast, the negative correlations between $\Delta\, cpi$ and future $\Delta\, ip$ concern predictable changes.

Figure 13.14 shows the ACF of the $\Delta\, cpi$ and $\Delta\, ip$ 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 13.14 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 CCFs at positive lags are above the main diagonal, and the CCFs at negative values of lag are below the main diagonal. □

13.4.5 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

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

and so forth, so that for all h,

$$\widehat{\boldsymbol{Y}}_{n+h} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1(\widehat{\boldsymbol{Y}}_{n+h-1} - \widehat{\boldsymbol{\mu}}) + \cdots + \widehat{\boldsymbol{\Phi}}_p(\widehat{\boldsymbol{Y}}_{n+h-p} - \widehat{\boldsymbol{\mu}}), \tag{13.20}$$

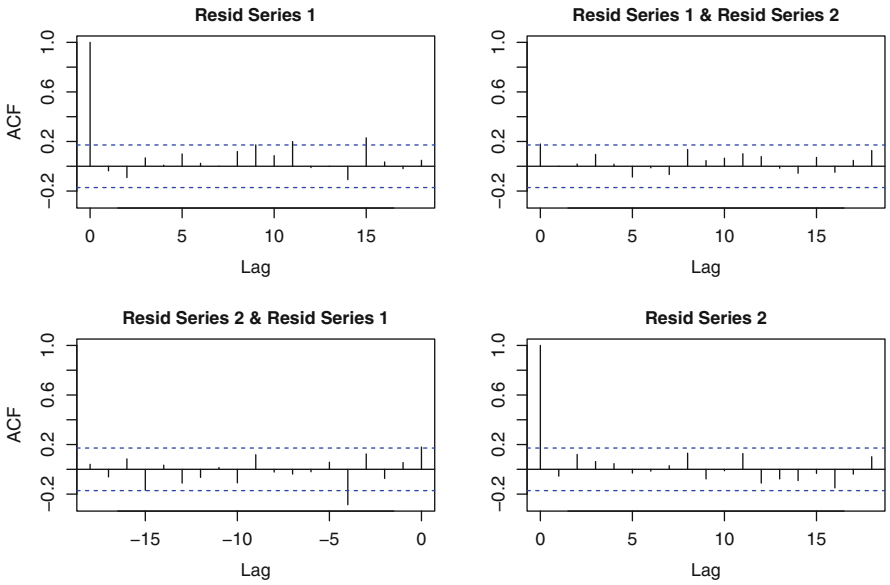

Fig. 13.14. *The ACF and CCF for the residuals when fitting a bivariate AR(1) model to* $(\Delta cpi, \Delta ip)'$. *Top left: The ACF of* Δcpi *residuals. Top right: The CCF of* Δcpi *and* Δip *residuals with positive values of lag. Bottom left: The CCF of* Δcpi *and* Δip *residuals with negative values of lag. Bottom right: The ACF of* Δip *residuals.*

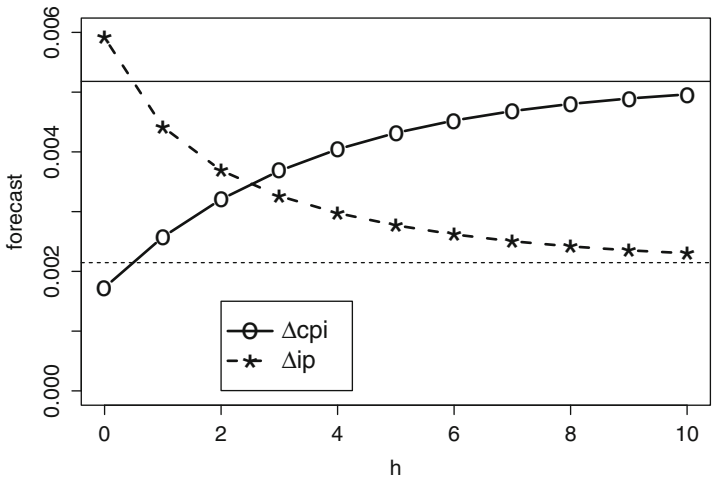

Fig. 13.15. *Forecasts of* Δcpi *(solid) and* Δip *(dashed) using a bivariate AR(1) model. The number of time units ahead is* h. *At* $h = 0$, *the last observed values of the time series are plotted. The two horizontal lines are at the means of the series, and the forecasts will asymptote to these lines as* $h \to \infty$ *since this model is stationary.*

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 (13.20) shows that

$$\widehat{\boldsymbol{Y}}_{n+h} = \widehat{\boldsymbol{\mu}} + \widehat{\boldsymbol{\Phi}}_1^h(\boldsymbol{Y}_n - \widehat{\boldsymbol{\mu}}). \tag{13.21}$$

Example 13.11. Using a bivariate AR(1) model to predict CPI and IP

The ΔCPI and ΔIP series were forecast using (13.21) with estimates found in Example 13.10. Figure 13.15 shows forecasts up to 10 months ahead for both CPI and IP. Figure 13.16 shows forecast limits computed by simulation using the techniques described in Sect. 12.12.2 generalized to a multivariate time series. □

13.5 Long-Memory Processes

13.5.1 The Need for Long-Memory Stationary Models

In Chap. 12, 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

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

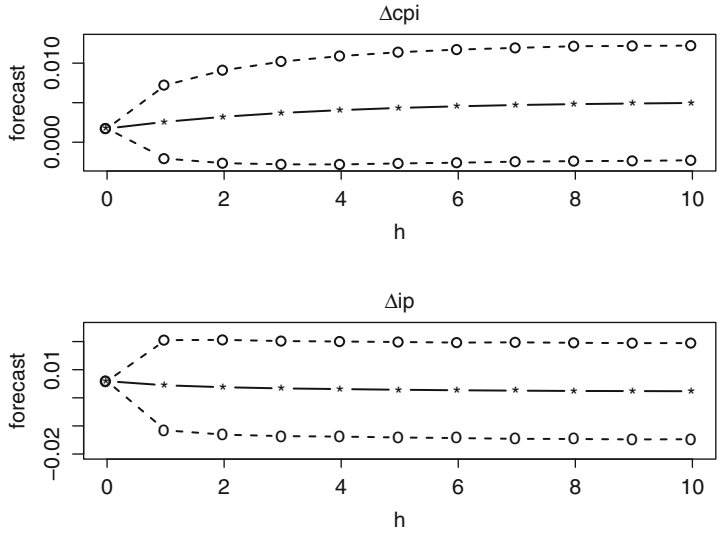

Fig. 13.16. *Forecast limits (dashed) for Δ cpi and Δ ip computed by simulation, and forecasts (solid). At $h = 0$, the last observed changes are plotted so the widths of the forecast intervals are zero.*

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

13.5.2 Fractional Differencing

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

$$\Delta^d = (1-B)^d = \sum_{k=0}^{d} \binom{d}{k} (-B)^k. \tag{13.22}$$

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

We define

$$\binom{d}{k} = \frac{d(d-1)\cdots(d-k+1)}{k!} \tag{13.23}$$

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

$$(1-x)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-x)^k. \tag{13.24}$$

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

$$(1-x)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-x)^k = \sum_{k=0}^{d} \binom{d}{k} (-x)^k. \tag{13.25}$$

The right-hand side of (13.25) is the usual finite binomial expansion for d a nonnegative integer, so (13.24) 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

$$\Delta^d Y_t = \sum_{k=0}^{\infty} \binom{d}{k} (-1)^k Y_{t-k}. \tag{13.26}$$

13.5.3 FARIMA Processes

A process 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 process.

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 13.17 shows time series plots and sample ACFs for simulated FARIMA$(0, d, 0)$ processes with $n = 2{,}500$ 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 a 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 Sect. 12.9, 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 which 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 nonstationary processes would not have a sample ACF that dropped below 0.5 so quickly.

Note that the case $d = -0.35$ in Fig. 13.17 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 Fig. 13.17. 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 Fig. 13.18.

Example 13.12. Inflation rates—FARIMA modeling

This example uses the inflation rates that have been studied already in Chap. 12. From the analysis in that chapter it was unclear whether to model the series as $I(0)$ or $I(1)$. Perhaps it would be better to have a compromise

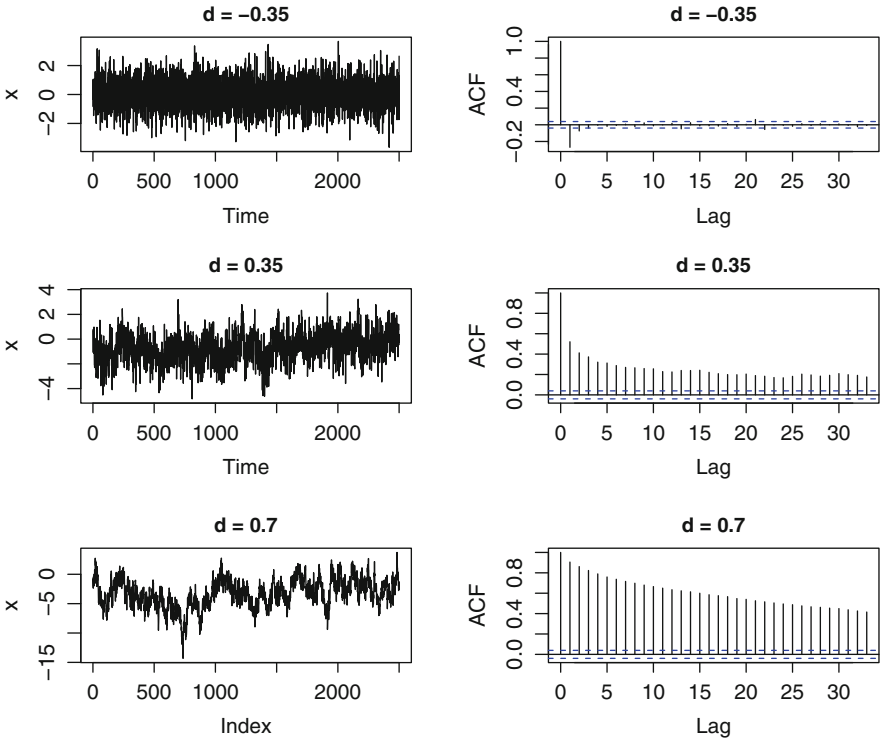

Fig. 13.17. *Time series plots (left) and sample ACFs (right) for simulated* $FARIMA(0, d, 0)$*: the top series is stationary with short-term memory; the middle series is stationary with long-term memory; the bottom series is nonstationary.*

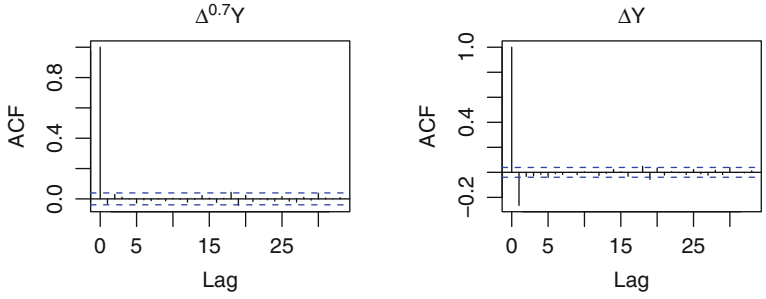

Fig. 13.18. *Sample ACF plots for the simulated* $FARIMA(0, 0.7, 0)$ *series in Figure 13.17 after differencing using* $d = 0.7$ *and* 1.

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 Fig. 12.3 is similar to that of the $d = 0.35$ plot in Fig. 13.17.

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; we are 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.

Figure 13.19 has sample 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.

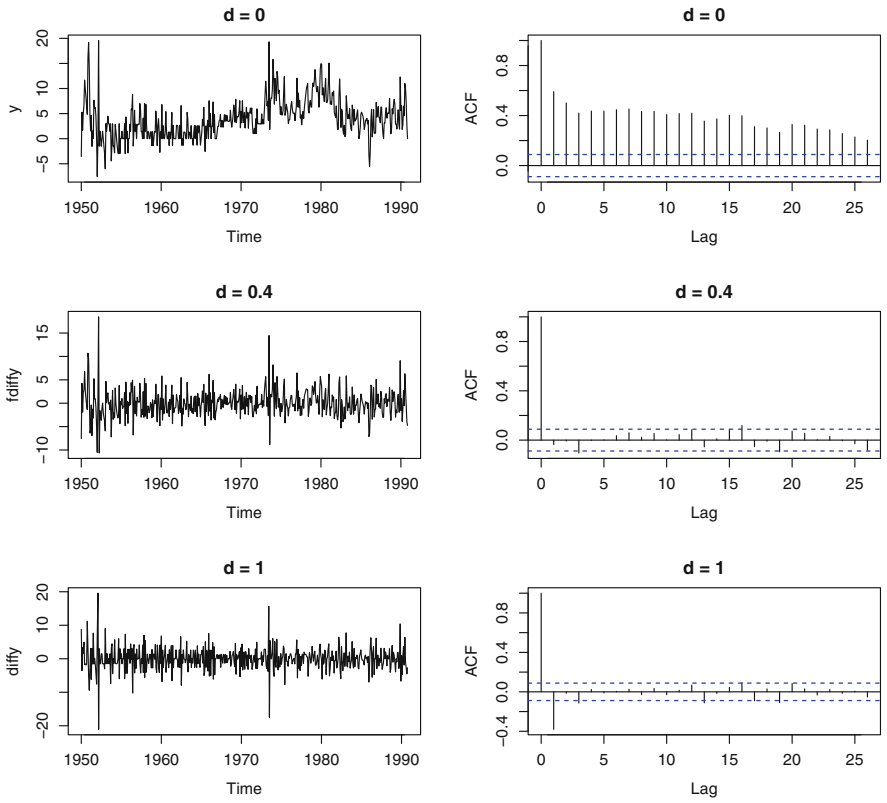

Fig. 13.19. *Time series plots (left) and sample ACF plots (right) for the inflation rates series with differencing using* $d = 0$, 0.4, *and* 1.

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 a more complex autocorrelation structure than 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. □

13.6 Bootstrapping Time Series

The resampling methods introduced in Chap. 6 are designed for i.i.d. univariate data but are easily extended to multivariate data. As discussed in Sect. 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 $Y_1, \ldots, Y_n$. 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 replacement, 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 bootstrapping is beyond the scope of this book, but see Sect. 13.7 for further reading.

13.7 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 the "References" section discuss seasonal ARIMA models. For more details on HC and HAC covariance matrix estimators and the R package `sandwich` see Zeileis (2004). Enders (2004) has a section on 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.

13.8 R Lab

13.8.1 Seasonal ARIMA Models

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

```
library("Ecdat")
library("forecast")
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. Which 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.

```
logConsumption = log(consumption)
fitAutoArima = auto.arima(logConsumption, ic="bic")
foreAutoArima = forecast(fitAutoArima, h=8)
plot(foreAutoArima, xlim=c(1985.5,1987.5), ylim=c(10.7,11.2))
```

13.8.2 Regression with HAC Standard Errors

Run the following commands in `R` to compute the OLS estimates of the regression of the differenced one-month T-bill rates, `tb1_diff`, on the differenced three-month T-bill rates, `tb3_diff`.

```
data(Mishkin, package="Ecdat")
tb1_dif = diff(as.vector(Mishkin[,3]))
tb3_dif = diff(as.vector(Mishkin[,4]))
fit = lm(tb1_dif ~ tb3_dif )
round(summary(fit)$coef, 4)
acf(fit$resid)
```

Problem 6 *Is there evidence of significant autocorrelation among the residuals? Why?*

Now run the following commands to compute the HC standard error estimates and their associated t values.

```
library(sandwich)
sqrt(diag(NeweyWest(fit, lag = 0, prewhite = F)))
coef(fit)/sqrt(diag(NeweyWest(fit, lag = 0, prewhite = F)))
```

Problem 7 *How do these t values compare to the t values from the OLS fit? Does the HC adjustment change the conclusions of the hypothesis tests?*

Problem 8 *Run the commands again, but with* `lag` *equal to 1,2, and 3 to obtain the corresponding HAC t values. How do the t values vary with* `lag`*?*

13.8.3 Regression with ARMA Noise

This section uses the USMacroG data set used earlier in Sect. 9.11.1. In the earlier analysis, we did not investigate residual correlation, but now we will. The model will be the regression of changes in unemp = unemployment rate on changes in government = real government expenditures and changes in invest = real investment by the private sector. Run the following R code to read the data, compute differences, and then fit a linear regression model with AR(1) errors.

```
library(AER)
data("USMacroG")
MacroDiff = as.data.frame(apply(USMacroG, 2, diff))
attach(MacroDiff)
fit1 = arima(unemp, order=c(1,0,0), xreg=cbind(invest, government))
```

Problem 9 *Fit a linear regression model using* lm(), *which assumes uncorrelated errors. Compare the two models by AIC and residual ACF plots. Which model fits better?*

Problem 10 *What are the values of BIC for the model with uncorrelated errors and for the model with AR(1) errors? Does the conclusion in Problem 9 about which model fits better change if one uses BIC instead of AIC?*

Problem 11 *Does the model with AR(2) noise or the model with ARMA(1,1) noise offer a better fit than the model with AR(1) noise?*

13.8.4 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.

```
TbGdpPi = read.csv("TbGdpPi.csv", header=TRUE)
#  r = the 91-day treasury bill rate
#  y = the log of real GDP
#  pi = the inflation rate
TbGdpPi = ts(TbGdpPi, start = 1955, freq = 4)
del_dat = diff(TbGdpPi)
var1 = ar(del_dat, order.max=4, aic=T)
var1
acf(na.omit(var1$resid))
```

Problem 12 *For this problem, use the notation of Eq. (13.16) 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 13 *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?*

```
tail(TbGdpPi, n = 4)
```

```
          r   y    pi
[233,] 0.07 9.7  1.38
[234,] 0.04 9.7  0.31
[235,] 0.02 9.7  0.28
[236,] 0.07 9.7 -0.47
```

Now fit a VAR(1) using the following commands.

```
var1 = ar(del_dat, order.max=1)
```

Suppose we observe changes in `r`, `y`, and `pi` that are each 10 % above the mean changes:

```
yn = var1$x.mean * 1.1 ; yn
```

Problem 14 *Compute the* h*-step forecasts for* $h = 1, 2,$ *and* 5 *using* `yn` *as the most recent observation. How do these forecasts compare to the mean* `var1$x.mean`*? For each* h*, compute ratios between the forecasts and the mean. How do these values compare to the starting value,* `yn/var1$x.mean = 1.1`*? Are they closer to or farther from* `1.0 = var1$x.mean/var1$x.mean`*? What does this suggest?*

Using the fitted VAR(1) from above, examine the estimate of $\hat{\boldsymbol{\Phi}}$:

```
Phi_hat = var1$ar[1,,] ; Phi_hat
```

Problem 15 *What do the elements of* `Phi_hat` *suggest about the relationships among the changes in* `r`, `y`, *and* `pi`*?*

A VAR(1) process is stationary provided that the eigenvalues of $\boldsymbol{\Phi}$ are less than one in magnitude. Compute the eigenvalues of $\hat{\boldsymbol{\Phi}}$:

```
eigen.values = eigen(Phi_hat)$values
abs(eigen.values)
```

Problem 16 *Is the estimated process stationary? How does this result relate to the forecast calculations in Problem 14 above?*

The dataset MacroVars.csv contains three US macroeconomic indicators from Quarter 1 of 1959 to Quarter 4 of 1997: Real Gross Domestic Product (a measure of economic activity), Consumer Price Index (a measure of inflation), and Federal Funds Rate (a proxy for monetary policy). Each series has been transformed to stationary based on the procedures suggested by Stock and Watson (2005).

```
MacroVars = read.csv("MacroVars.csv", head=TRUE)
```

Problem 17 *Fit a VAR(p) model using the* ar() *function in* R *using AIC (the default) to select lag order.*

Problem 18
By modifying the output of the ar() *function as discussed in Example 13.10, use BIC to select the lag order. Comment on any differences.*

13.8.5 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 19 *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 20 *Do you see any short- or long-term autocorrelation in the fractionally differenced series?*

Problem 21 *Fit an ARIMA model to the fractionally differenced series using* auto.arima(). *Compare the models selected using AIC and BIC.*

13.8.6 Model-Based Bootstrapping of an ARIMA Process

This exercise 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(2,1,0). 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(2,1,0) model, run the following commands.

```
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] = 0.2825*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 22 *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 = 1000
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 23 *Find the biases, standard deviations, and MSEs of the estimators of the two coefficients.*

13.9 Exercises

1. Figure 13.20 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?
2. Figure 13.21 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?

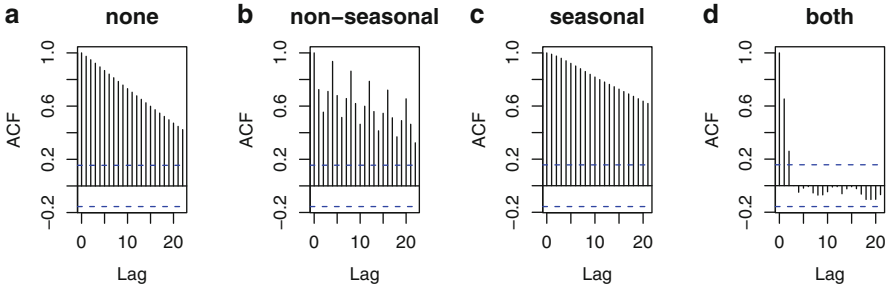

Fig. 13.20. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

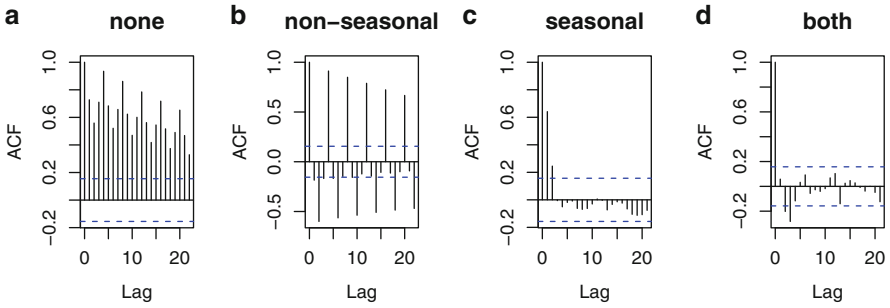

Fig. 13.21. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

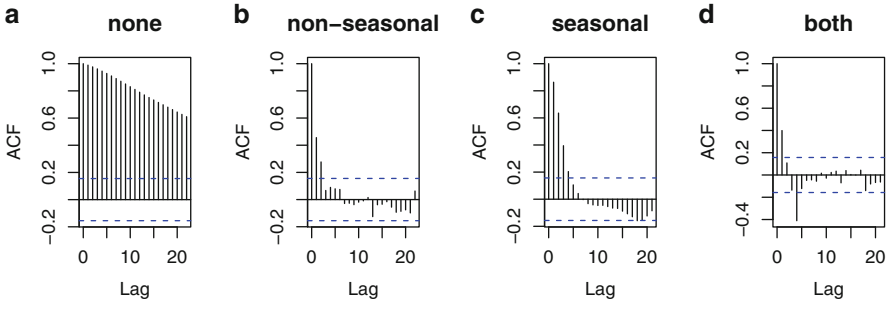

Fig. 13.22. *ACF plots of quarterly data with no differencing, nonseasonal differencing, seasonal differencing, and both seasonal and nonseasonal differencing.*

3. Figure 13.22 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 13.10, a bivariate AR(1) model was fit to $(\Delta cpi, \Delta ip)'$ and

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

The mean of $(\Delta cpi, \Delta ip)'$ is $(0.0052, 0.0021)'$ and the last observation of $(\Delta cpi, \Delta ip)'$ is $(0.0017, 0.0059)'$. Forecast the next two values of Δip. (The forecasts are shown in Fig. 13.15, 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 your 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 `TbGdpPi.csv` data set. In Sect. 12.15.1, 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?

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.

Newey, W. and West, K. (1987) A simple, positive semidefinite, heteroscedasticity and autocorrelation consistent covariance matrix. *Econometrica*, **55**, 703–708.

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

Stock, J. H. and Watson, M. W. (2005). *An empirical comparison of methods for forecasting using many predictors,* manuscript http://www4.ncsu.edu/~arhall/beb_4.pdf

White, H. (1980) A heteroscedasticity consistent covariance matrix estimator and a direct test for heteroscedasticity. *Econometrica*, **48**, 827–838.

Zeileis, A. (2004) Econometric computing with HC and HAC covariance matrix estimators. *Journal of Statistical Software*, **11**(10), 1–17.