Introductory Time Series With R

Chapter · January 2009
DOI: 10.1007/978-0-387-88698-5

2 authors, including:

Andrew V Metcalfe
University of Adelaide
130 PUBLICATIONS **1,674** CITATIONS

SEE PROFILE

Introductory Time Series with R

Paul S.P. Cowpertwait

Contents

1

Introduction & Exploratory Methods

In industry and commerce, as well as in most quantitative academic subjects, many variables are measured sequentially in time. For example, in economics, interest rates and exchange rates are recorded each day. Different variables may be measured over different time intervals, for example: the amount of rainfall captured in a gauge may be recorded for each calendar month; temperatures may be taken each hour at a weather station; or gross domestic product calculated on an annual basis by the government statistics department. When a variable is measured sequentially in time over a fixed interval (or *sampling interval*) the data form a *time series*.

Observations that have been collected in the past over fixed sampling intervals form an *historical* time series. One main objective in a time series analysis of historical data is to formulate and fit an appropriate mathematical model for the series. In this book, a *statistical* approach is taken in which the historical series are treated as realisations of a sequence of *random variables*. A sequence of random variables taken over fixed sampling intervals is sometimes referred to as a discrete-time *stochastic process*. The theory of such processes is vast and may be studied without necessarily fitting any models to data. However, our focus will be more 'applied' and directed towards model fitting and data analysis, for which we will be using `R`. [1] We will also tend to rely on the context to distinguish between a random variable and a recorded observation, so that the term *time series* could refer to either an historical series or a discrete-time stochastic process.

An important feature of most historical time series is that the observations are serially dependent and correlated in time. Much of the methodology in a time series analysis is aimed at explaining this correlation and the main features in the data, using appropriate statistical models and descriptive methods. Once a good model is found and fitted to data the analyst can proceed

[1] `R` was initiated by Ihaka and Gentleman (1996) and is an open source implementation of `S` – a language for data analysis developed at the Bell Laboratories (Becker et al. (1988)).

to using the model; for example, to forecast future values for the series which may help with any long-term planning decisions that need to be made. A fitted model may also be used as a basis for statistical tests; for example, with monthly sales figures, management will probably want to know whether there is any statistical evidence that sales are increasing. Finally, a fitted statistical model also provides a concise summary of the main characteristics of a time series, thus providing a useful way of presenting the data.

1.1 Notation

In many books on statistics it is common to see random variables represented in upper case with realisations in lower case; for example, the statement $P(X = x)$ is used to mean the probability that the random variable X takes the value x. However, we will rely on context to make the distinction, and use lower case throughout for both random variables and their realisations; for example, $P(x)$ would be used to mean the probability of observing x. The notation $\{x_t : t = 1, \ldots, n\} = \{x_1, x_2, \ldots, x_n\}$ will be used to represent a time series of length n, which consists of n values measured over sampling intervals $1, 2, \ldots, n$. The notation will be abbreviated to $\{x_t\}$ when the length n of the series does not need to be specified. The 'hat' notation will be used to represent a *prediction*. For example, with the series $\{x_t : t = 1, \ldots, n\}$, $\hat{x}_t$ is the predicted value at time t corresponding to observed value x_t ($t \leq n$). If $t > n$, $\hat{x}_t$ is a *forecast* for a future value. Again, $\hat{x}_t$ could be a random variable or an actual fitted value depending on the context.

1.2 Other books on time series

In this book we will focus on applying time series methods using `R`. There are numerous good books available that provide a more detailed treatment of the theory that underpins the methods used here; some of these are now briefly mentioned.

Chatfield (1989) is a popular concise introductory text in which the mathematical details do not dominate the discussion. Alternatively, for a range of applications and further models it is worth reading Shumway and Stoffer (2000), or, for more specialised economic time series models and applications, consider Enders (1995).

The above three references have served me well, but there are plenty of other books available that are also very helpful. It is also worth cross-referencing with other texts to gain a differing perspective or a deeper insight into a particular topic. The following references could also prove helpful: Diggle (1990); Kendall and Ord (1990); Brockwell and Davis (1996); Bowerman et al. (2005).

1.3 Examples and R

It is assumed that you have R installed on your computer (version 2.0.0 or above), and it is suggested that you work through the examples making sure your output agrees with ours.[2] If you do not have R, then it can be installed free of charge from the internet site www.r-project.org. It is also recommended that you have some familiarity with the basics of R, which can be obtained by working through the first few chapters of an elementary text book on R (e.g. Dalgaard (2002)), or using the on-line "An Introduction to R", which is also available via the R help system – type help.start() at the command prompt to access this.

R has many features in common with both *functional* and *object orientated* programming languages. In particular, functions in R are treated as objects that can be manipulated or used recursively. [3] For example, the factorial function can be written recursively as:

```
> fact = function(n) if (n == 1) 1  else  n * fact(n-1)
```

In common with functional languages, assignments in R can be avoided, but are useful for clarity and convenience, and hence will be used in the examples that follow. In addition, R runs faster when 'loops' are avoided, which can often be achieved using matrix calculations instead. However, this can sometimes result in rather obscure looking code. Thus, for the sake of transparency, loops will be used in many of our examples.

The best way to learn to do a time series analysis in R is through practice and 'hands-on' experience, so we now turn to some examples, which you are invited to work through.

1.4 Examples and plots

1.4.1 A flying start: The classic air passenger data set

This first example is based on the classic airline data set used by Box and Jenkins (1970), and is helpful for illustrating several important concepts that arise in an exploratory time series analysis. The data is also available in R, making it easily accessible.

[2] Some of the output given in this book may differ slightly from yours. This is most likely to be due to editorial changes made for stylistic reasons. For conciseness, we also used options(digits=4) to set the number of digits to 4 in the computer output that appears in the book.

[3] Do not be concerned if you are unfamiliar with these 'programming' terms, as they are not really essential in understanding the material in this book. The main reason for mentioning them now is to emphasise that R can almost certainly meet your *future* statistical and programming needs, should you wish to take the study of time series further.

The number of people (in thousands) travelling per month on an airline are recorded for the period 1949–1960. It is likely that an airline company recording data of this type would be interested in making predictions of demand for the future; for example, with a view to planning the numbers of aircraft needed to meet a rising demand. The data is fairly typical of an economic series in which a future supply must be planned to match a predicted demand. Type the following commands in R, and check your results against the output shown here. To save on typing, the data are stored in a variable called AP.

```
> data(AirPassengers)
> AP = AirPassengers
> AP
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

All data in R are stored in *objects*, which have a range of *methods* available. The 'class' of an object can be found using the `class` function:

```
> class(AP)
[1] "ts"
> start(AP); end(AP); frequency(AP)
[1] 1949    1
[1] 1960   12
[1] 12
```

In this case the object is of class `ts`, which is an abbreviation for 'time series'. Time series objects have a number of *methods* available, which include the functions `start`, `end`, and `frequency`, given above. These methods can be listed using the function `methods`, but the output from this function is not always helpful. The key thing to bear in mind is that *generic* functions in R, such as `plot` or `summary`, will attempt to give the most appropriate output to any given input object; try typing `summary(AP)` now to see what happens.

As the objective in this book is to analyse time series it makes sense to put our data into objects of class `ts`. This can be achieved using a function also called `ts`, but was not necessary for the airline data which were already

stored in this form. In the next example, we shall create a `ts` object from data read directly from the internet.

One of the most important steps in a preliminary time series analysis is to plot the data, i.e. create a *time plot*. For a time series object, this is achieved with the generic plot function:

```
> plot(AP)
```

You should obtain a plot similar to Figure (1.1) below. Parameters, such as `xlab` or `ylab`, can be used in `plot` to improve the default labels.

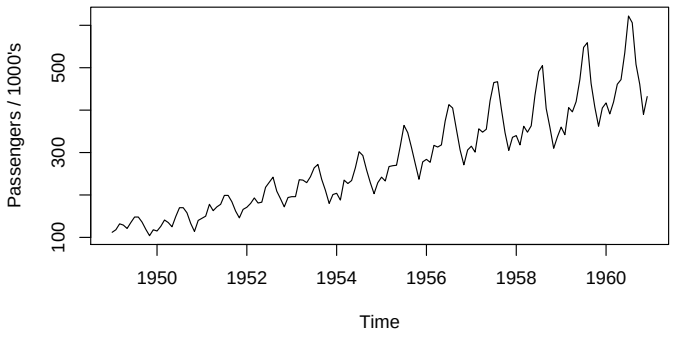


Fig. 1.1. Time plot of the air passenger data for the period 1949–1960.

There are a number of features in the time plot of the air passenger data that are common to many time series (Figure 1.1). For example, it is apparent that the number of passengers travelling on the airline is increasing with time, so there appears to be an increasing *trend*. There is also a clear *cycle* in the data which has a one-year period, i.e. there is clear *seasonal variation* in the data.

An understanding of the likely causes of the features in the plot can form a useful guide when formulating an appropriate time series model. In this case, the possible causes might include: increasing demand for air travel due to more competitive prices or greater availability of aircraft; increased demand at certain times of year, e.g. over holiday periods which might explain the seasonal variation. Alternatively, an increasing population might explain the steadily increasing trend.

In Chapter 3, time series regression models will be formulated based on underlying causes like these. However, it should be noted that many time series exhibit 'apparent' trends, which might, for example, be part of a longer cycle, or be random and subject to unpredictable change. Such random, or *stochastic*,

trends are common in economic and financial time series. A regression model would not be appropriate for a stochastic trend.

A time series plot is not only important for revealing patterns and features of the data but can also reveal potential *outliers* or *erroneous* values. For example, missing data are sometimes coded using a negative value – such values would obviously need to be handled differently in the analysis, and would certainly not want to be included as observations when fitting a model to data, such as the airline series, which can only take positive values. [4]

To get a clearer view of the trend, the seasonal effect can be 'removed' by aggregation of the data to the annual level, which can be achieved in `R` using the `aggregate` function. A summary of the values for each season can be viewed using a boxplot, with the `cycle` function to extract the seasons for each item of data. The plots can be put in a single figure using the `layout` function (Figure 1.2).

```
> plot(aggregate(AP), ylab="Annual passengers/1000's")
> boxplot(AP ~ cycle(AP), names=month.abb)

> layout(1:2)
> plot(aggregate(AP)); boxplot(AP ~ cycle(AP))
```

An increasing trend is evident in the annual series (Figure 1.2a), whilst the seasonal effects are clearly revealed in the boxplot, for which a tendency for more people to travel over the summer months of June to September can be seen (Figure 1.2b).

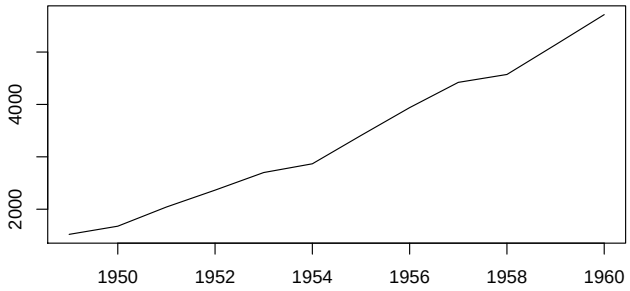
1.4.2 Example: Electricity, beer and chocolate data

This section is intended to illustrate a few important ideas and concepts related to *multiple* time series data. The monthly supply of electricity (in millions of kWh), beer and chocolate-based productivity (in tonnes) in Australia over the period January 1958 to December 1990 are available from the Australian Bureau of Statistics (ABS). [5] I have stored the three series in a single file online, which can be read as follows.

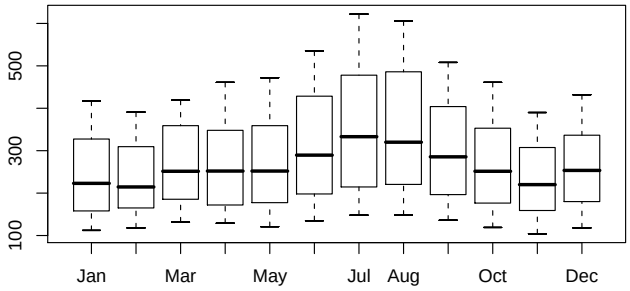
```
> www = "http://www.massey.ac.nz/~pscowper/data/cbe.dat"
> cbe = read.table(www, header=T)
```

[4] Generally speaking, missing values are suitably handled by `R`, provided they are correctly coded as 'NA'. However, if your data does contain missing values, then it is always worth checking the 'help' on the `R` function that you are using as an extra parameter or piece of coding may be required.

[5] ABS data used with permission from the Australian Bureau of Statistics: http://www.abs.gov.au.



(a) Aggregated annual series



(b) Boxplot of seasonal values

Fig. 1.2. Air passenger data for the period 1949–1960. Units on the y-axis are 1000's of people. (a) Series aggregated to the annual level; (b) Seasonal boxplots of the data.

```
> cbe[1:4,]
  choc beer elec
1 1451 96.3 1497
2 2037 84.4 1463
3 2477 91.2 1648
4 2785 81.9 1595

> class(cbe)
[1] "data.frame"
```

The 'class' is not time series but `data.frame`, which is usually the case when reading data for more than one variable in this way. The `ts` function is used

to convert the data to a time series object. To understand how `ts` works, enter the following command, and then experiment with different 'start' and 'end' times of your own – use the 'up' arrow on the keyboard to recall the last command and edit the dates or frequency.

```
> ts(1:120, start=c(1990, 1), end=c(1993, 8), freq=12)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1990   1   2   3   4   5   6   7   8   9  10  11  12
1991  13  14  15  16  17  18  19  20  21  22  23  24
1992  25  26  27  28  29  30  31  32  33  34  35  36
1993  37  38  39  40  41  42  43  44
```

Note that for the frequency of 12, the vector `1:120` gives a maximum potential of 10 years of data, but a smaller number can be selected as in the example above.

Now create time series objects for the electricity, beer and chocolate data; `plot` with `cbind` can be used to plot several series on one figure (Figure 1.3).

```
> elec.ts = ts(cbe[,3], start=1958, freq=12)
> beer.ts = ts(cbe[,2], start=1958, freq=12)
> choc.ts = ts(cbe[,1], start=1958, freq=12)

> plot(cbind(elec.ts, beer.ts, choc.ts),
  main="Chocolate, Beer, and Electricity Production: 1958-1990")
```

The plots show increasing trends in production for all three variables, probably due to increasing demand as a result of a rising population in Australia (Figure 1.3).

The three series constitute a *multiple* time series. There are many functions in `R` for handling more than one series, including `ts.intersect` to obtain the intersection of two series which overlap in time. We now illustrate the use of the intersect function and point out some potential pitfalls in analysing multiple time series. The intersection between the air passenger data and the electricity data is obtained as follows:

```
> ap.elec = ts.intersect(AP, elec.ts)
> start(ap.elec); end(ap.elec)
[1] 1958    1
[1] 1960   12

> ap.elec[1:3, ]
      AP elec.ts
[1,] 340    1497
[2,] 318    1463
[3,] 362    1648
```

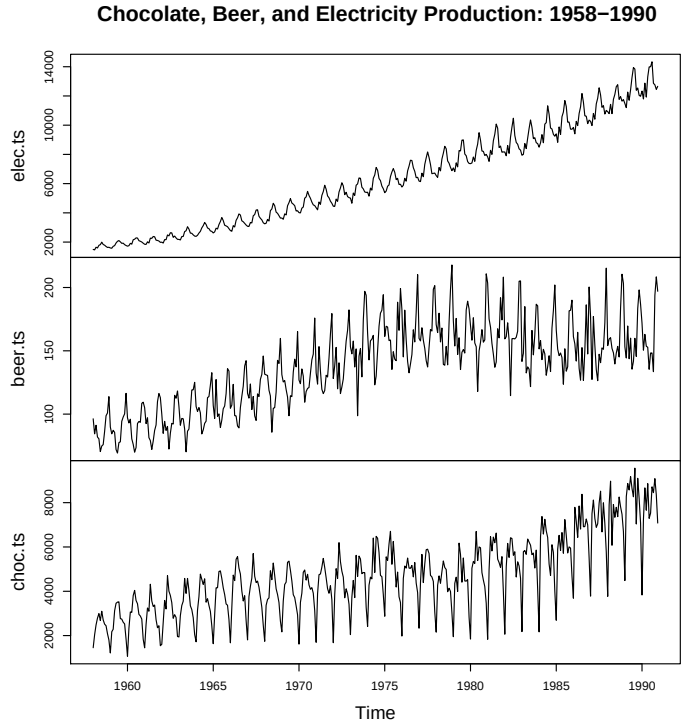


Fig. 1.3. Australian chocolate, beer and electricity production: January 1958 – December 1990.

In the code below, the data for each series is extracted and plotted: [6]

```
> ap = ap.elec[,1]; elec = ap.elec[,2]

> layout(1:2)
> plot(ap, main="", ylab="Air passengers / 1000's")
> plot(elec, main="", ylab="Electricity production / MkWh")

> plot(as.vector(ap), as.vector(elec),
   xlab="Air passengers / 1000's",
   ylab="Electricity production / MWh")
> abline(reg=lm(ap ~ elec))
```

[6] `R` is case sensitive, so lower case is used here to represent the shorter record of air passenger data.

```
> cor(ap,elec)
[1] 0.8842
```

In the `plot` function above, `as.vector` is needed to 'coerce' the `ts` objects to ordinary vectors suitable for a scatter plot.

The two time series are highly correlated, as can be seen by the plots and correlation coefficient of 0.88. Correlation will be discussed more in Section 1.5, but for the moment observe that the two time plots look similar (Figure 1.4), and that the scatter plot shows an approximate linear relationship between the two variables (Figure 1.5). However, it is important to realise that this scatter plot is *misleading*, and any regression model based on these pairs of observations would be *spurious*.[7] The plot shows an apparent relationship between two variables that have no clear causal link – the correlation exists because both data sets have an underlying trend and seasonal variation. Any series that has a trend will tend to be correlated to any other series that has a trend. For this reason, it is preferable to remove trends and seasonal effects before comparing multiple series. This is usually achieved by working with the residuals of a fitted time series model, such as the regression models of Chapter 3.

The term *non-stationary* is used to describe a time series that has underlying trends or seasonal effects. Time series models for non-stationary series are studied in Chapter **??**, whilst models suitable for stationary series are studied in Chapter **??**. In Chapter **??**, statistical tests are given that can be used to decide whether series are *stationary* before they are compared to other series.

1.4.3 Example: Quarterly exchange rate data

The trends and seasonal patterns in the previous two examples were visually obvious in the plots. In addition, reasonable explanations could be put forward for the possible causes of these features. With financial data, e.g. exchange rates, such obvious patterns are less likely to exist, and different methods of analysis are usually required. A financial series may sometimes show an obvious change which has a clear cause; for example, as a result of some catastrophe, such as a war or natural disaster. However, day-to-day changes are more difficult to predict because the underlying causes are complex and impossible to isolate. Thus, a statistical approach, in which the data are treated as random variables, is often the most suitable way of analysing a financial or economic time series.

The exchange rates, for British pounds sterling to New Zealand dollars for the period January 1991 to March 2000 are shown in Figure 1.6. The data are mean values taken over *quarterly* periods of three months, with the first quarter being January to March and the last quarter being October to December. They can be read into `R` from the internet and converted to a quarterly time series as follows:

[7] 'Spurious' regression is discussed in more detail in Chapter **??**.

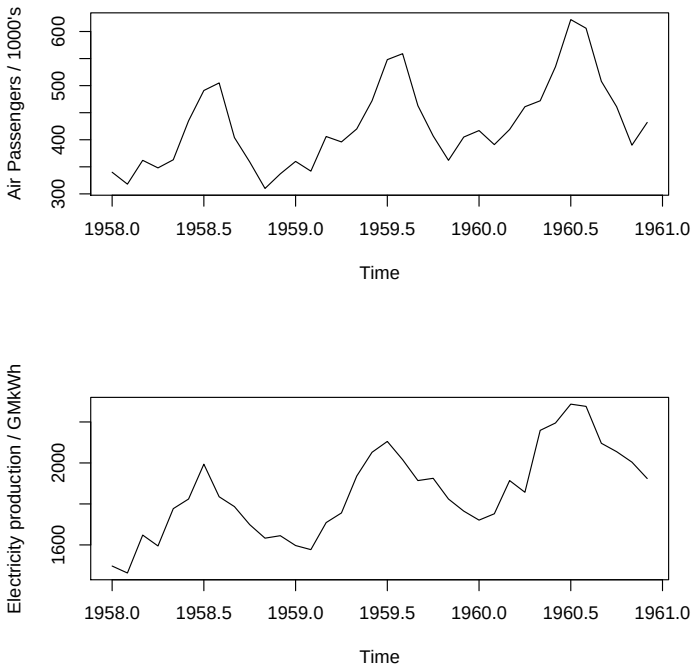


Fig. 1.4. Air Passengers and Australian Electricity Production for the period: 1958–1960. The plots look similar because both series have an increasing trend and a seasonal cycle. However, this does not imply there exists a causal relationship between the variables.

```
> www =
   "http://www.massey.ac.nz/~pscowper/data/pounds_nz.dat"
> z = read.table(www, header=T)
> z[1:4,]
[1] 2.924 2.942 3.172 3.254

> z.ts = ts(z, st=1991, fr=4)
> plot(z.ts, xlab="time / years",
  ylab="Quarterly exchange rate in $NZ / pound")
```

Trends are apparent in the time series: after an initial surge prior to 1992, a negative trend leads to a minimum around 1996, which is followed by a positive trend in the second-half of the series (Figure 1.6).

The trends in this series are more 'random' in nature than those in the last two examples. Such trends have been termed *stochastic trends* to emphasise

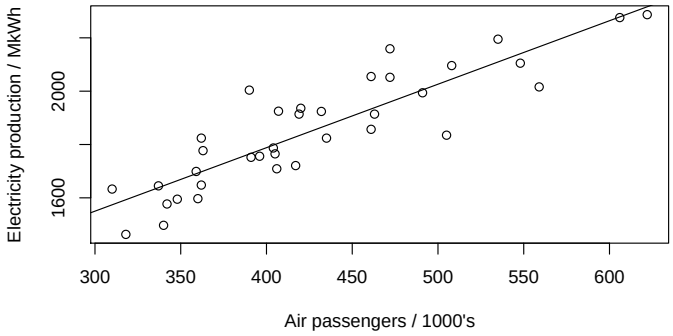


Fig. 1.5. Scatter plot of air passengers and Australian electricity production for the period: 1958–1960. The apparent linear relationship between the two variables is misleading and caused by the trends in the series.

this randomness and to distinguish them from more *deterministic* trends, like those seen in the previous two examples. A mathematical model known as a *random walk* can sometimes provide a good fit to data like this, and is fitted to this series in Section 1.7.2. Stochastic trends are common in financial series and will be studied in more detail in Chapters 2 and **??**.

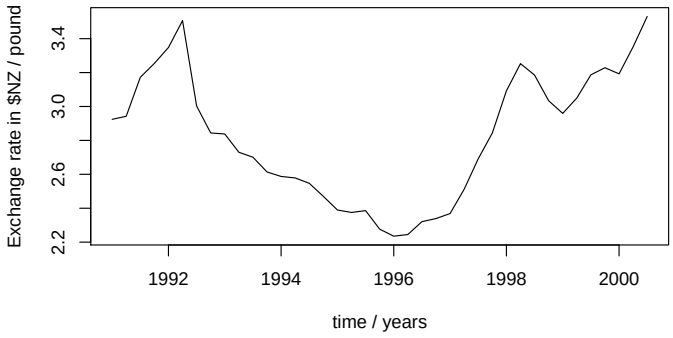


Fig. 1.6. Quarterly exchange rates for the period 1991–2000.

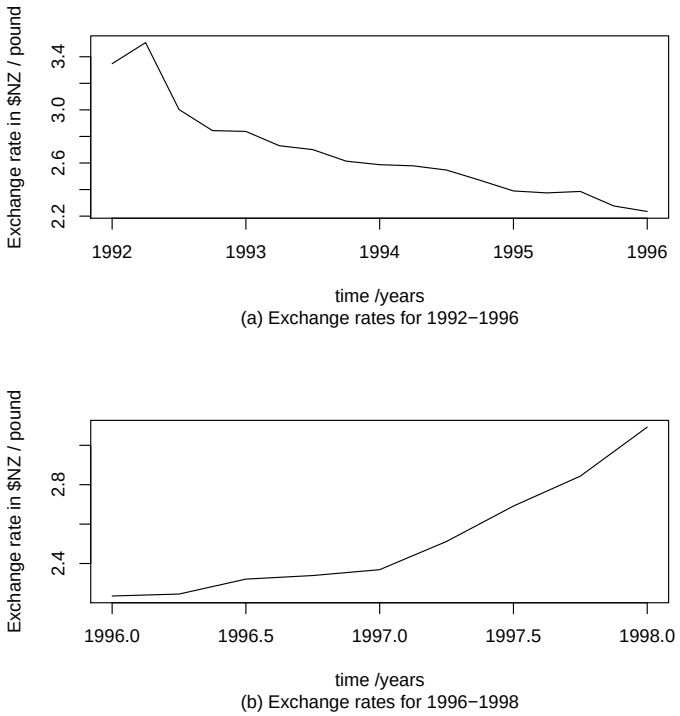


Fig. 1.7. Quarterly exchange rates for two periods. The plots indicate that, without additional information, it would be inappropriate to extrapolate the trends.

Two of the trends are clearly visible when the series is partitioned into two sub-series based on the periods 1992–1996 and 1996–1998. A dummy series is used to create the period for the intersection:

```
> dummy.ts = ts(st=c(1992, 1), end=c(1996, 1), fr=4)
> z.92_96 = ts.intersect(z.ts, dummy.ts)[,1]
> dummy.ts = ts(st=c(1996, 1), end=c(1998, 1), fr=4)
> z.96_98 = ts.intersect(z.ts, dummy.ts)[,1]

> layout(1:2)
> plot(z.92_96,
   ylab="Exchange rate in $NZ/pound", xlab="time/years")
> plot(z.96_98,
   ylab="Exchange rate in $NZ/pound", xlab="time/years")
```

Now suppose we were observing this series at the start of 1996, i.e. we had the data in Figure 1.7(a). It might have been tempting to predict a con-

tinuation of the downward trend for future years. However, this would have been a very poor prediction as Figure 1.7(b) shows the data started to follow an increasing trend. Likewise, without additional information, it would also be inadvisable to extrapolate the trend in Figure 1.7(b). This illustrates the potential pitfall of inappropriate extrapolation of stochastic trends, when underlying causes are not properly understood. In Chapter **??**, statistical tests are given to help decide whether a series has a stochastic trend to reduce the risk of making an inappropriate forecast.

1.4.4 Global temperature series

A change in the world's climate will have a major impact on the lives of many people, as global warming is likely to lead to an increase in natural hazards, such as floods and droughts. It is likely that the world economy will be severely effected as governments from around the globe try to enforce a reduction in fossil fuel usage and measures are taken to deal with any increase in natural disasters. [8]

In climate change studies (e.g. see Jones and Moberg (2003); Rayner et al. (2003)), the following global temperature series plays a central role:[9]

```
> www = "http://www.massey.ac.nz/~pscowper/data/global.dat"
> global = scan (www)
Read 1800 items
> global.ts = ts(global, st=c(1856,1), end=c(2005,12), fr=12)
> global.annual = aggregate(global.ts, FUN=mean)

> plot(global.ts); plot(global.annual)
```

As interest usually focuses on the overall trends in this series, the `aggregate` function is used above to remove the seasonal effects within each year and produce an annual series of mean temperatures for the period: 1856 to 2005 (Figure 1.8b).

The upward trend from about 1970 onwards has been used as an argument for global warming (Figure 1.9). In the code below, the monthly time intervals corresponding to the 36-year period (1970–2005) are extracted using the `time` function, and the associated observed temperature series extracted using `ts.intersect` (following the procedure used previously in Section 1.4.3). The data are plotted and a line superimposed using a regression of temperature on the new time index (Figure 1.9).

[8] For general policy documents and discussions on climate change, see the web site (and links) for the United Nations Framework Convention on Climate Change at: http://unfccc.int.

[9] The data is updated regularly and can be downloaded free of charge from the internet: http://www.cru.uea.ac.uk/cru/data/.

```
> dummy.ts = ts(st=c(1970, 1), end=c(2005, 12), fr=12)
> new.series = ts.intersect(global.ts, dummy.ts)[,1]
> new.time = time(new.series)
> plot(new.series); abline(reg=lm(new.series ~ new.time))
```

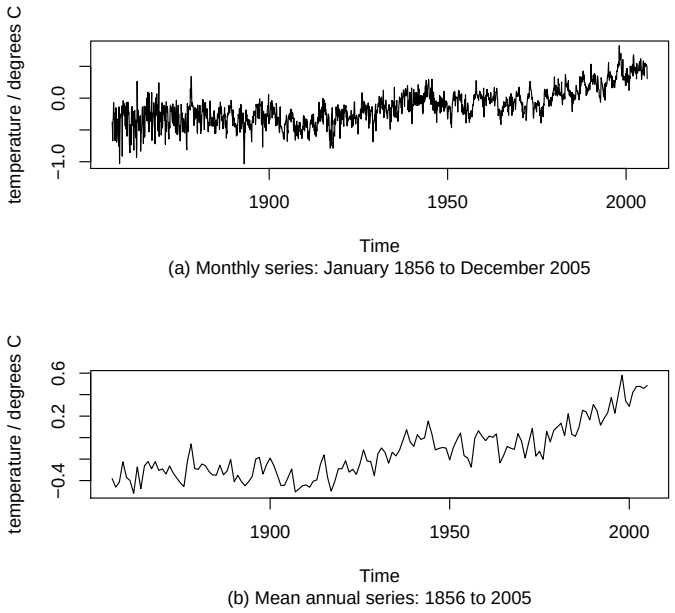


Fig. 1.8. Time plots of the global temperature series (°C).

In the previous section, we discussed a potential pitfall of inappropriate extrapolation. In climate change studies, a question of interest is whether the increasing trend in the data is induced by human activity, specifically the burning of fossil fuels and increased greenhouse gas emissions, or whether it is a more 'natural' trend, perhaps part of a longer cycle that may decrease in the future, without needing a global reduction in fossil fuel usage. It is not appropriate to use the increasing trend of fossil fuel usage to 'explain' the increase in global temperature, because, as we noted in Section 1.4.2, two unrelated time series will be correlated if they both contain a trend. However, as the general consensus among scientists is that the trend in the global temperature series is related to a global increase in greenhouse gas emissions, in this case it would be reasonable to acknowledge a causal relationship, and to expect the

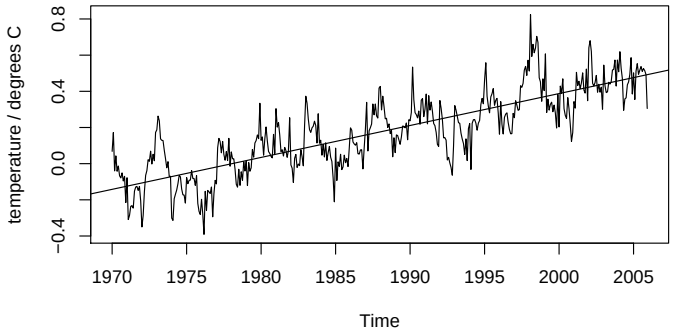


Fig. 1.9. Rising mean global temperatures: January 1970 – December 2005. According to the United Nations Framework Convention on Climate Change the mean global temperature is expected to continue to rise in the future unless greenhouse gase emissions, on a global scale, are reduced.

mean global temperature to continue to rise if greenhouse gases emissions are not reduced. [10]

1.5 The correlogram

1.5.1 Expectation

The *expectation* E of a variable (or a function of a variable) is the *mean* value, so $E(x)$ is the mean of x, denoted μ_x, and $E\{(x-\mu_x)^2\}$ is the mean of the squared deviations about μ_x, better known as the *variance* of x.[11] The variance may be generalised to the *covariance*, $\mathrm{cov}(x,y) = E\{(x-\mu_x)(y-\mu_y)\}$, to obtain a measure of *linear association* between two variables (x,y): if x and y tend to increase together, then y will tend to be greater than μ_y when x is greater than μ_x, resulting in a positive covariance; conversely, if y tends to decrease as x increases the covariance will tend to be negative. For the airline and electricity data we have:

```
> x = ap; y = elec; n = length(x)
> mean((x - mean(x))*(y - mean(y)))
```

[10] Refer to: http://unfccc.int

[11] The mean of a function $f(x)$ is a weighted average given by $\sum_{\mathtt{all}\ x} f(x)p(x)$, where the 'weights' $p(x)$ sum to unity. For discrete random variables $p(x)$ is the probability of obtaining x; conversely, if x is sampled data, $p(x)$ is the proportion of times x occurs in the sample.

```
[1] 14564
> sum((x - mean(x))*(y - mean(y)))/(n - 1)
[1] 14980
> cov(x,y)
[1] 14980
```

The correspondence between the `R` code above and the expectation definition of covariance should be noted:

$$E\left[\{x-\mu_x\}\{y-\mu_y\}\right] \rightarrow \texttt{mean((x - mean(x)*(y - mean(y)))} \qquad (1.1)$$

Given this correspondence, the more 'natural' estimate of covariance would be: `mean((x - mean(x)*(y - mean(y)))`. However, as can be seen above, the values computed using the internal function `cov` are those obtained using `sum` with a divisor of $n-1$. These are the *unbiased* estimates for the covariance, and are needed especially when the sample size n is small, as it is in this example. As n gets large, the -1 correction in the divisor becomes less important, and the more natural estimate *asymptotically* approaches the unbiased estimate. In a time series analysis, a biased estimate often has to be tolerated because the observations are serially dependent. In most cases this does not present a problem as the sample size n is usually greater than it is in the example above.

1.5.2 Autocorrelation

Correlation measures the linear association between a pair of variables (x, y), and is obtained by 'standardising' the covariance, by dividing the covariance by the standard deviations of the variables. It takes a value between -1 and $+1$, with a value of 0 indicating no *linear* association. A value of $+1$ or -1 indicates an exact linear association with the (x, y) pairs falling on a straight line of positive or negative slope respectively. The correlation between a pair of variables (x, y) is defined by:

$$\text{cor}(x,y) = E\{(x-\mu_x)(y-\mu_y)\}/\sigma_x\sigma_y = \text{cov}(x,y)/\sigma_x\sigma_y, \qquad (1.2)$$

In `R`, the correlation between a pair of variables (x, y), with data stored in vectors `x` and `y` respectively, can be evaluated using the function `cor(x,y)`.

Time series observations will tend to be serially correlated, or *autocorrelated*. For example, pairs of values taken at a time difference of lag 1 will tend to have a high correlation when there are trends in the data. The lag k autocorrelation is defined by:

$$\rho_k = \text{cor}(x_t, x_{t+k}) = E\{(x_t-\mu_x)(x_{t+k}-\mu_x)\}/\sigma_x^2, \qquad (1.3)$$

This definition follows naturally from Equation (1.2) by replacing x with x_t and y with x_{t+k}, and using the overall mean μ_x for the mean of both x_t and x_{t+k}. Note that ρ_0 is always 1.

In R the lag k autocorrelation can be obtained from the autocorrelation function `acf`. By default, the `acf` function produces a plot of ρ_k against k, which is called the *correlogram*.

```
> acf(x)
```

The autocorrelations are stored in the vector `acf(x)$acf`, with the lag k autocorrelation located in `acf(x)$acf[k+1]`. For example, the lag 1 autocorrelation for `x` is:

```
> acf(x)$acf[2]
[1] 0.7755
```

1.5.3 Interpreting the correlogram

Figure 1.10 gives the correlogram for the air passenger data over the three-year period 1958–1960, from which the following features are observed:

- The x-axis gives the lag (k) and the y-axis gives the autocorrelation (ρ_k) at each lag, i.e. the correlogram is a plot of ρ_k against k. In this example, the units on the x-axis are 'years', so lags $0, 1, 2, \ldots$ (in *months*) appear at times $0, 1/12, 2/12, \ldots$ (in *years*). Correlation is dimensionless, so there are no units for the y-axis.
- The dotted lines represent the 5% significance level for the statistical test: $\rho_k = 0$. Any correlations that fall outside these lines are 'significantly' different from zero. The lines are only a rough guide and not too much should be read into them. In particular, as they represent multiple significance tests, 5% of the values would show up as 'significant', due to sampling variation, even when their underlying values are zero. This will become more obvious when we look at simulation in Chapter 2. In the meantime, it is worth looking for significant values at *significant lags*, such as the lag that corresponds to the seasonal period.
- The lag 0 autocorrelation is always 1 and is shown on the plot. It provides an indicator of the relative values of the other autocorrelations; if a correlation turns out to be statistically significant, but is actually very small in magnitude, it may be of little or no practical consequence, and will look 'insignificant' alongside ρ_0. However, some discernment is required to decide what constitutes a 'significant' autocorrelation from a practical viewpoint. Squaring the autocorrelation can help, as this gives the percentage of variability 'explained' by a linear relationship between the variables. For example, a lag 1 autocorrelation of 0.1 implies that a linear dependency of x_t on x_{t-1} would only explain 1% of the variability between the two variables. It is a common fallacy to treat a statistically significant result as important when it has almost no practical consequence.
- The annual cycle appears in the correlogram as a cycle of the same period, with a significant positive correlation at period 1-year. This reflects a positive linear relationship between pairs of variables (x_t, x_{t+12}), separated by

12-month periods. Conversely, values separated by a period of 6 months will tend to have a negative relationship, because, for example, higher values tend to occur in the summer months followed by lower values in the winter months. A negative correlation therefore occurs at lag 6 months (or 0.5 years); Figure 1.10.

- The significant autocorrelation at lag 1 month is probably due to the increasing trend over the period of the data. Usually a trend in the data will show in the correlogram as a slow decay in the autocorrelations, due to similar values in the series occurring close together in time. As the correlogram in Figure 1.10 is based on only three years of data there is only marginal evidence of this. However, the correlogram for the full period, obtained from `acf(AP)`, exhibits a slow decay in the autocorrelations, due to the trend, as expected (Figure 1.11).

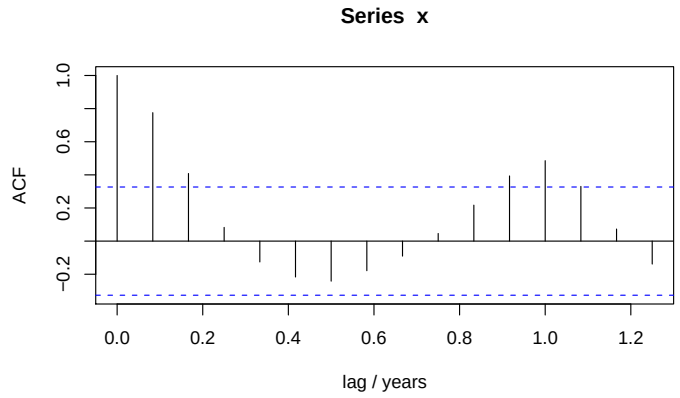


Fig. 1.10. Correlogram for the air passenger data over the period 1958–1960. The 1-year cycle in the correlogram corresponds to the seasonal period.

1.5.4 Second-order properties and stationarity

The mean and variance play an important role in the study of statistical distributions, because they summarise two key distributional properties – the 'centre' and the 'spread'. Similarly, in the study of stationary time series $\{x_t\}$, a central role is played by the *second-order properties*: the mean μ_x and the autocovariance $\gamma_k = \text{cov}(x_t, x_{t+k})$. If the mean and autocovariance do not change with time, then $\{x_t\}$ is a called a *second-order stationary* process. The implication of this is that the variance and autocorrelation of $\{x_t\}$ also do not change with time. In practice, the term 'second-order' is dropped and $\{x_t\}$

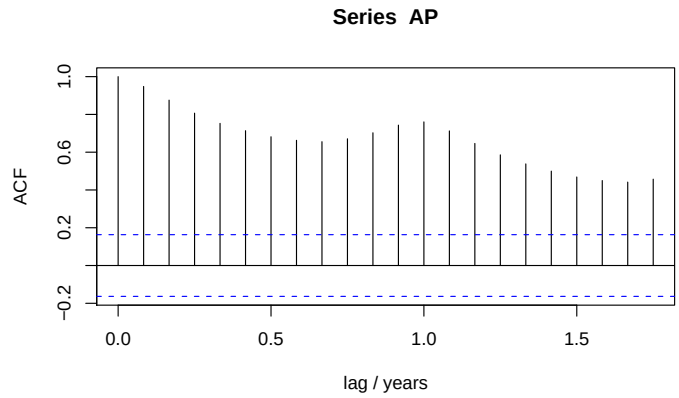


Fig. 1.11. Correlogram for the air passenger data over the period 1949–1960. The gradual decay is typical of a time series containing a trend.

is just called stationary, with the term *strictly* stationary being reserved for more rigorous conditions.

In subsequent chapters, second-order properties for several time series models are derived using the following result:

$$\text{cov}(\sum_{i=1}^{n} x_i, \sum_{j=1}^{m} y_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} \text{cov}(x_i, y_j) \tag{1.4}$$

The proof of this result is left to Exercise 2a.

1.6 Decomposition of series

1.6.1 Models

Usually a central objective in a time series analysis is to model the main features in the data. As the first two examples showed, many series are dominated by a trend and/or seasonal effects, so that a model formulation based on these two components is desirable. A simple *additive decomposition* model is given by:

$$x_t = m_t + s_t + \epsilon_t \tag{1.5}$$

where at time t: x_t is the observed series, m_t is the trend, s_t is the seasonal effect, and ϵ_t is the remainder or residual series. This model is sometimes called a *classical decomposition* model. In this section, we briefly outline two

main approaches for extracting the trend m_t and the seasonal effect s_t in (1.5), and give the main `R` functions for doing this.

The choice of method for extracting m_t and s_t is in many ways arbitrary. Classical decomposition essentially has its main value as a descriptive tool to enable the main components of a time series to be viewed prior to a more substantial statistical analysis.

If the seasonal effect tends to increase as the trend increases a multiplicative model may be more appropriate:

$$x_t = m_t \cdot s_t + \epsilon_t \tag{1.6}$$

If the residual series is also multiplicative the data can be transformed using the natural logarithm to produce an additive model for $\log(x_t)$:

$$\log(x_t) = m_t + s_t + \epsilon_t \tag{1.7}$$

Some care is required when the exponential inverse transform is applied, as the effect is usually to bias the predictions. If the residual series ϵ_t are Normally distributed with mean 0 and variance σ^2, then the predicted value at time t based on Equation (1.7) is given by:

$$\hat{x}_t = e^{m_t + s_t + \frac{1}{2}\sigma^2} \tag{1.8}$$

The bias correction factor $e^{\frac{1}{2}\sigma^2}$, which is often overlooked, is the mean of a log-normal distribution resulting from taking the logarithms of the series, i.e. $E(e^{\epsilon_t}) = e^{\frac{1}{2}\sigma^2}$. However, for data that are not Normally distributed this may be an overcorrection; in Chapter 3 we study this in more detail in the context of time series regression.

1.6.2 Estimating trends and seasonal effects

There are various ways to estimate the trend m_t at time t. One procedure is to calculate a *moving average* centred on x_t. For monthly series, the trend at time t can be estimated by the centred moving average:

$$\hat{m}_t = \frac{\frac{1}{2}x_{t-6} + x_{t-5} + \ldots + x_{t-1} + x_t + x_{t+1} + \ldots + x_{t+5} + \frac{1}{2}x_{t+6}}{12} \tag{1.9}$$

The coefficients add to 1 to ensure equal weight is given to each season. The procedure generalises naturally to any seasonal frequency.

An alternative is to fit a polynomial of order q to the series and estimate m_t from the fitted model:

$$\hat{m}_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \ldots + \alpha_q t^q \tag{1.10}$$

where the α_i are parameter estimates. For series that show obvious changes in the trend, it may be necessary to fit more than one polynomial. This practice

is acceptable when the breaks can be related to an underlying cause. For series that have a straight-line trend, $q = 1$ and the trend is estimated using the fitted line: $\alpha_0 + \alpha_1 t$. [12]

An estimate of the seasonal effect (s_t) at time t can be obtained by subtraction of $\hat{m}_t$:

$$\hat{s}_t = x_t - \hat{m}_t \tag{1.11}$$

If the seasonal effect is multiplicative, the estimate is given by division, i.e. $\hat{s}_t = x_t/\hat{m}_t$. By averaging the values for each season, a single component can be estimated. For example, for monthly data we may have n estimates $\hat{s}_i, \hat{s}_{12+i}, \ldots, \hat{s}_{12(n-1)+i}$ for each month i, which can be combined to give a single estimate of the seasonal component for each month based on the mean: $\bar{s}_i = (\hat{s}_i + \hat{s}_{12+i} + \ldots + \hat{s}_{12(n-1)+i})/n$. An alternative to using the mean, is to fit polynomials through the estimates, to allow for possible changes in the seasonal components with time.

Whichever way the seasonal effects are estimated, they can be removed from the original series to produce a *seasonally adjusted* series. If the seasonal effect is additive a seasonally adjusted series is given by $x_t - s_t$, whilst if it is multiplicative an adjusted series is obtained from x_t/s_t. In the case of monthly data, we can adjust using the mean seasonal effect: $s_t = \bar{s}_i$ when t occurs in the ith month ($i = 1, 2, \ldots, 12$).

1.6.3 Decomposition in R

In R the functions `decompose` and `stl` estimate trends and seasonal effects using methods based on moving averages and polynomials respectively. These functions can be used as descriptive methods to view the main components of a time series, before moving on to the statistical modelling methods discussed in the remaining part of this book. Nesting the functions within `plot`, using `plot(decompose())` or `plot(stl())`, produces a single figure showing the original series x_t and the decomposed series: m_t, s_t and ϵ_t. For example, with the electricity data, additive and multiplicative decomposition plots are given by the following commands; the last plot is the superposition of the seasonal effect on the trend (Figure 1.13):

```
> plot(decompose(elec.ts))

> elec.decom = decompose(elec.ts, type="mult")
> plot(elec.decom)
> trend = elec.decom$trend; seasonal = elec.decom$seasonal

> ts.plot(cbind(trend, trend * seasonal), lty=1:2)
```

[12] Some authors refer to the value of the *slope* (α_1) as the trend, and the value at the line ($\alpha_0 + \alpha_1 t$) as the *level*.

In this example, the multiplicative model would seem more appropriate than the additive model because the variance of the original series increases with time (Figure 1.12). However, the 'random' component, which corresponds to ϵ_t, also has an increasing variance, which indicates that a log-transformation (Equation 1.7) may be more appropriate for this series (Figure 1.12). This is investigated further in Chapter 3.

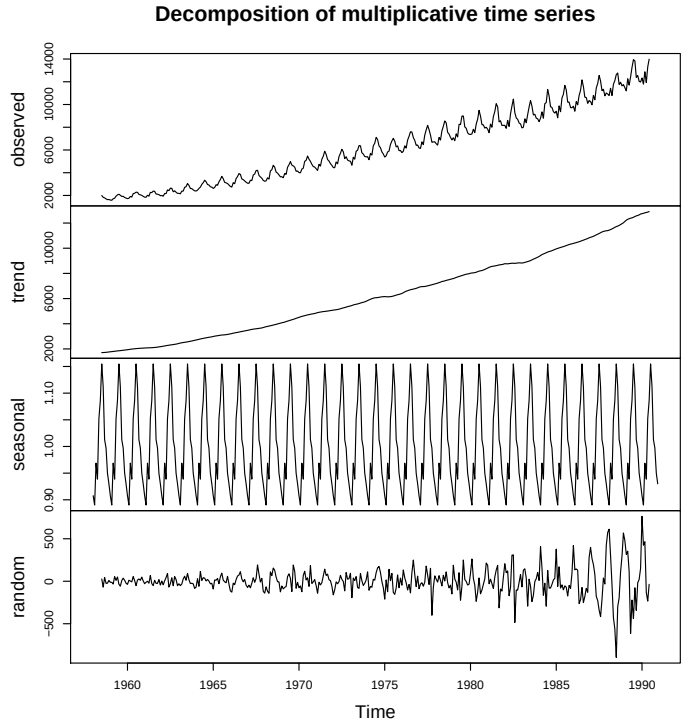


Fig. 1.12. Decomposition of the electricity production data

1.7 Forecasting

1.7.1 Exponential smoothing and the Holt-Winters method

The main objective in forecasting is to predict some future value x_{n+k} given a past history $\{x_1, x_2, \ldots, x_n\}$ of observations up to time n . One approach to this is to use the past values as dependent variables in a predictive model

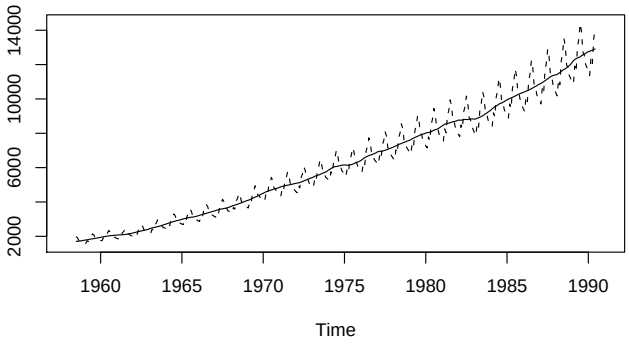


Fig. 1.13. Electricity production data: Trend with superimposed multiplicative seasonal effects

that gives more weight to the more recent observations. This approach is used in a model based on *exponential smoothing*:

$$\begin{aligned}\hat{x}_{n+1} &= \alpha x_n + \alpha(1-\alpha)x_{n-1} + \alpha(1-\alpha)^2 x_{n-2} + \ldots \\ &= \alpha x_n + (1-\alpha)\hat{x}_n \end{aligned} \tag{1.12}$$

where $-1 < \alpha < 1$, which ensures that the weights $\alpha(1-\alpha)^i$ become smaller as i increases. Note that these weights form a geometric series which sums to unity.

For any given α, the model in Equation (1.12) can be used to predict a value $\hat{x}_t$ corresponding to an observed value x_t, for $t = 1, 2, \ldots, n$. A residual error in the prediction is given by:

$$\epsilon_t = x_t - \hat{x}_t \tag{1.13}$$

A suitable estimate for α can be obtained by minimising the sum of squared errors:

$$SSE = \sum_{t=1}^{n} \epsilon_t^2 = \epsilon_1^2 + \epsilon_2^2 + \ldots + \epsilon_n^2 \tag{1.14}$$

The Holt-Winters procedure generalises Equation (1.12) to allow for a changing trend term using a parameter β and a changing seasonal term using a parameter γ. For a series $\{x_t\}$ with period p, containing a trend m_t and seasonal component s_t (c.f. Equation 1.5), the Holt-Winters procedure provides a prediction for x_{n+1} by updating the trend and seasonal effect:

$$\hat{x}_{n+1} = m_n + b_n + s_{n+1-p} \tag{1.15}$$

where b_n is the estimated *change* in trend at time n, so $m_n + b_n$ is the expected trend for time $n+1$, and s_{n+1-p} is the seasonal effect estimated using the previous value; e.g. for monthly data ($p = 12$), if time $n+1$ occurs in January then s_{n+1-12} is the seasonal effect for January in the previous year. The estimates of m_n, b_n and s_n are updated using:

$$m_n = \alpha(x_n - s_{n-p}) + (1-\alpha)(m_{n-1} + b_{n-1}) \tag{1.16}$$

$$b_n = \beta(m_n - m_{n-1}) + (1-\beta)b_{n-1} \tag{1.17}$$

$$s_n = \gamma(x_n - m_n) + (1-\gamma)s_{n-p} \tag{1.18}$$

where α, β, and γ are the model parameters, which can be estimated by minimising the sum of squared errors (1.14). When there is no change in trend or seasonal effect the model reduces to exponential smoothing, i.e. the model becomes $\hat{x}_{n+1} = m_n = \alpha x_n + (1-\alpha)\hat{x}_n$ which is Equation (1.12). Otherwise the parameters are updated for time n by taking a weighted average of the previous estimate with an appropriate estimate of the current value (at time n); for example, with monthly data ($p = 12$) the previous estimate of the seasonal effect is the value in the previous year (at time $n-12$) and the current estimate for time n is $x_n - m_n$ (c.f. Equation 1.11). These are weighted to give the estimated value at time n shown in Equation (1.18). The equations are easily modified to allow for a multiplicative seasonal effect, in most cases just by replacing subtraction with division, as discussed earlier in Section 1.6.2.

In `R`, the function `HoltWinters` can be used to estimate the parameters of the Holt-Winters model by minimisation of SSE (1.14). The exponential smoothing model can also be fitted using `HoltWinters`, with parameters `beta` and `gamma` both set to zero.

1.7.2 One-step ahead predictions for the exchange rate data

As discussed in Section 1.4.3 forecasts for the exchange rate data could be misleading. However, as the data have no obvious seasonal cycle we shall use them to illustrate the fitting of the exponential smoothing model.

```
> z.hw = HoltWinters(z.ts, beta=0, gamma=0)
> z.hw$alpha
alpha
    1
> plot(z.hw)
```

The estimate for α in Equation (1.12) is $\hat{\alpha} = 1$, so $\hat{x}_{n+1} = x_n$ and the prediction at $t+1$ is the observed value at t (Figure 1.14). The underlying model for this prediction is:

$$x_t = x_{t-1} + \epsilon_t \tag{1.19}$$

where ϵ_t are independent identically distributed residual variables with mean zero. Equation (1.19) defines the *random walk*, which often arises in the anal-

ysis of financial series. In general, exponential smoothing provides good predictions when the underlying process is a random walk or a random walk with moving average error terms.

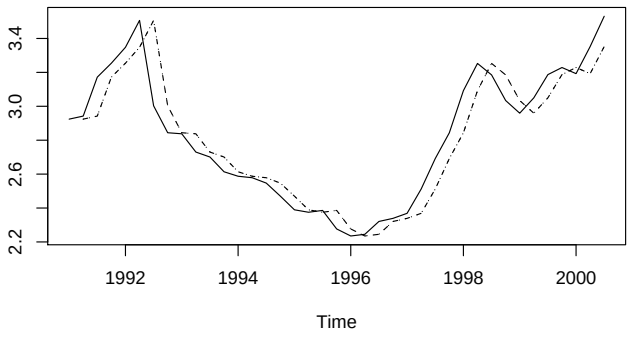


Fig. 1.14. Exchange rate data with exponentially smoothed predicted values shown as dotted points.

1.7.3 Four-year ahead forecasts for the air passenger data

The seasonal effect for the air passenger data of Section 1.4.1 appeared to increase with the trend (Figure 1.1), which suggests that a 'multiplicative' seasonal component be used in the Holt-Winters procedure. The `predict` function in `R` can be used with the fitted model to make forecasts into the future (Figure 1.15).

```
> AP.hw = HoltWinters(AP, seasonal="mult")
> plot(AP.hw)

> AP.predict = predict(AP.hw, n.ahead=4*12)
> ts.plot(AP, AP.predict, lty=1:2)
```

The estimates of the model parameters, which can be obtained from `AP.hw$alpha`, `AP.hw$beta`, and `AP.hw$gamma`, are: $\hat{\alpha} = 0.274$, $\hat{\beta} = 0.0175$ and $\hat{\gamma} = 0.877$. It should be noted that the extrapolated forecasts are based entirely on the trends in the period that the model was fitted, and would be a sensible prediction assuming these trends continue. Whilst the extrapolation in Figure 1.15 looks visually appropriate, unforeseen events could lead to completely different future values than those shown here.

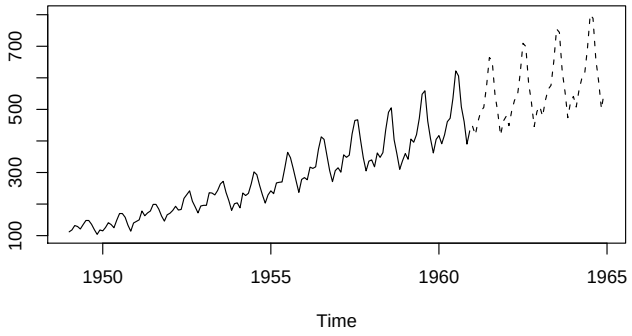


Fig. 1.15. Holt-Winters forecasts for air passenger data for 1961–1964 shown as dotted lines.

1.8 Summary of commands used in examples

`ts`	produces a time series object
`ts.plot`	produces a time plot for one or more series
`ts.intersect`	creates the intersection of one or more time series
`cycle`	returns the season for each value in a series
`time`	returns the times for each value in a series
`acf`	returns the correlogram
`decompose`	decomposes a series into the components: trend, seasonal effect and residual
`HoltWinters`	estimates the parameters of the Holt-Winters or exponential smoothing model
`predict`	forecasts future values

1.9 Exercises

1. Carry out the following exploratory time series analysis using the global temperature series from Section 1.4.4.
   a) Produce a time plot of the data. Plot the aggregated annual mean series, and a boxplot which summarises the observed values for each season, and comment on the plots.
   b) Decompose the series into the components: trend, seasonal effect and residuals, and plot the decomposed series. Produce a plot of the trend with a superimposed seasonal effect.
   c) Plot the correlogram of the residuals from question 1b. Comment on the plot, explaining any 'significant' correlations at significant lags.

d) Fit an appropriate Holt-Winters model to the monthly data. Explain why you chose that particular Holt-Winters model, and give the parameter estimates.

e) Using the fitted model, forecast values for the years 2005–2010. Add these forecasts to a time plot of the original series. Under what circumstances would these forecasts be valid? What comments of caution would you make to an economist or politician who wanted to use these forecasts to make statements about the potential impact of global warming on the world economy?

2. a) Prove Equation (1.4), using the following properties of summation, expectation, and covariance:

$$\begin{aligned} \textstyle\sum_{i=1}^{n} x_i \sum_{j=1}^{m} y_j &= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} x_i y_j \\ E(\textstyle\sum_{i=1}^{n} x_i) &= \textstyle\sum_{i=1}^{n} E(x_i) \\ cov(x, y) &= E(xy) - E(x)E(y) \end{aligned}$$

b) By taking $n = m = 1$ in Equation (1.4), derive the well-known result:

$$var(x + y) = var(x) + var(y) + 2\, cov(x, y)$$

c) Verify the result in part (b) above using `R` with `x` and `y` (`ap` and `elec` respectively) taken from Section 1.5.1.

2

Basic Models

In Chapter 1, two approaches were considered for modelling time series. The first was essentially descriptive and involved decomposing a series into a trend and seasonal components. The second involved fitting models to enable future values to be predicted from a weighted average of past terms. In both cases, a residual error series $\{\epsilon_t\}$ remained after the main features in the data were accounted for.

We begin this chapter with a study of the 'ideal' residual error series, which is called 'white noise'. After this two models are considered, each of which have white noise error terms. The first is the random walk, which is foundational to many other time series models, such as the ARIMA models of Chapter **??**. The second model, which has the random walk as a special case, is the *autoregressive* model, so-called because an observation at time t is 'regressed' on past values. Autoregressive models can account for the serial correlation in many time series, thus making them useful for a range of applications.

2.1 White noise

2.1.1 Introduction

A residual error ϵ_t is the difference between an observed value and a model predicted value at time t: $\epsilon_t = x_t - \hat{x}_t$ (Equation 1.13). As the residual errors occur in time, they form a time series: $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

In Chapter 1, we found that features of the historical series, such as the trend or seasonal variation, are reflected in the correlogram. Thus, if a model has 'explained' all the features in the data, the residual series would be serially *uncorrelated*, so that a correlogram of the residual series would exhibit no obvious patterns. This leads to the following definition.

2.1.2 Definition

A residual series $\{\epsilon_t : t = 1, 2, \ldots, n\}$ is *white noise* if the variables $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are *independent* and *identically* distributed with a mean of zero. This implies that the variables all have the same variance σ^2 and $\text{cor}(\epsilon_i, \epsilon_j) = 0$ for all $i \neq j$. If, in addition, the variables also follow a Normal distribution, i.e. $\epsilon_t \sim \text{N}(0, \sigma^2)$, the series is called *Gaussian* white noise.[1]

2.1.3 Simulation in R

A fitted time series model can be used to *simulate* data. Time series simulated using a model are sometimes called a *synthetic* series to distinguish them from an observed historical series.

For many reasons simulation is useful. For example, simulation can be used to construct confidence intervals for model parameters (*bootstrapping*) or to generate plausible future scenarios to aid decision making in management science or operations research. In R simulation is usually straightforward; most standard statistical distributions are simulated using a function that has an abbreviated name for the distribution prefixed with an 'r' (for 'random').[2] For example, `rnorm(100)` is used to simulate 100 independent standard Normal variables, which is equivalent to simulating a Gaussian white noise series of length 100 (Figure 2.1).

```
> set.seed(1)
> e = rnorm(100)
> plot(e, type="l")
```

Simulation experiments can easily be repeated using the 'up' arrow on the keyboard. For this reason, it is sometimes preferable to put all the commands on one line, separated by ';', or to nest the functions; for example a plot of a white noise series is given by: `plot(rnorm(100), type="l")`.

In the above code, the function `set.seed` is used to provide a starting point (or *seed*) in the simulations, thus ensuring that the simulations can be reproduced. If this function is left out a different set of simulated data are obtained, although the underlying statistical properties remain unchanged. To see this, re-run the plot above a few times with and without `set.seed(1)`.

To illustrate by simulation how samples may differ from their underlying populations, consider the following histogram of a Gaussian white noise series:

```
> x = seq(-3,3, len=1000)
> hist(rnorm(100), prob=T); points(x, dnorm(x), type="l")
```

[1] The term 'white noise' originated in engineering applications where it was used to refer to series that contained *all* frequencies in *equal* proportions, analogous to white light. The term *purely random* is sometimes used for white noise series.

[2] Other prefixes are also available to calculate properties for standard distributions, e.g. the prefix 'd' is used to calculate the probability (density) function. See the R help for more details.

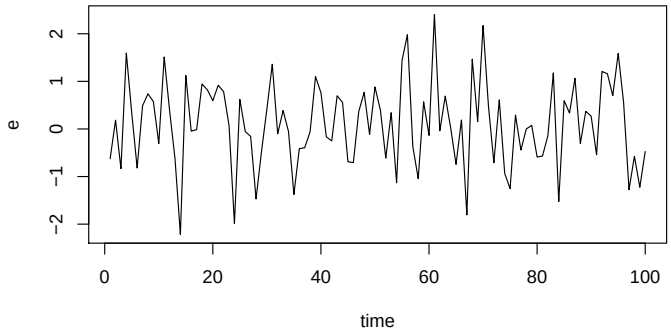


Fig. 2.1. Time plot of simulated Gaussian white noise series

Repetitions of the last command will show a range of different *sample* distributions that arise when the underlying distribution is Normal. Distributions that show a 'significant' departure from the plotted curve have arisen due to sampling variation.

2.1.4 Second-order properties and the correlogram

The second-order properties of a white noise series $\{\epsilon_t\}$ are an immediate consequence of the definition in Section 2.1.2. However, as they are needed so often in the derivation of the second-order properties for more complex models, we explicitly state them here:

$$\left.\begin{aligned} \mu_\epsilon &= 0 \\ \gamma_k = \mathrm{cov}(\epsilon_t, \epsilon_{t+k}) &= \begin{cases} \sigma^2 & \texttt{if} \quad k = 0 \\ 0 & \texttt{if} \quad k \neq 0 \end{cases} \end{aligned}\right\} \tag{2.1}$$

The autocorrelation function follows as:

$$\rho_k = \begin{cases} 1 & \texttt{if} \quad k = 0 \\ 0 & \texttt{if} \quad k \neq 0 \end{cases} \tag{2.2}$$

Simulated white noise data will not have autocorrelations that are *exactly* zero (when $k \neq 0$), because of sampling variation. In particular, for a simulated white noise series it is expected that 5% of the autocorrelations will be significantly different from zero at the 5% significance level, shown as dotted lines on the correlogram. Try repeating the following command to view a range of

correlograms that could arise from an underlying white noise series. An example plot, with one 'significant' autocorrelation, occurring at lag 7, is shown in Figure 2.2.

```
> acf(rnorm(100))
```

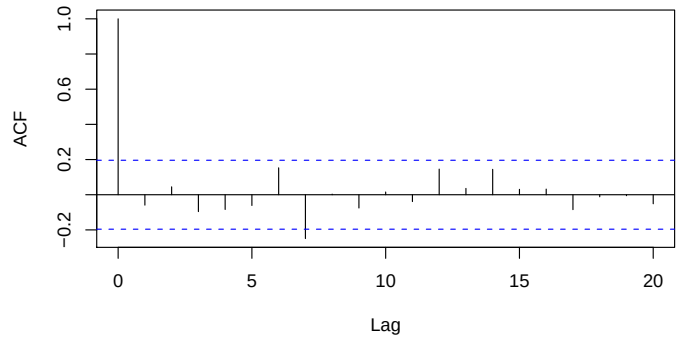


Fig. 2.2. Correlogram of a simulated white noise series. The underlying autocorrelations are all zero (except at lag 0); the statistically significant value at lag 7 is due to sampling variation.

2.1.5 Fitting a white noise model

A white noise series usually arises as a residual series after fitting an appropriate time series model – the correlogram usually provides sufficient evidence to support the conjecture that the residuals are white noise.

The only parameter for a white noise series is the variance σ^2, which is usually estimated by the residual variance, adjusted by degrees of freedom, given in the computer output of the fitted model. In the trivial case, where your analysis begins on data that are already approximately white noise, then only σ^2 needs to be estimated, which is easily achieved using the `var` function.

2.2 Random walks

2.2.1 Introduction

In Chapter 1, the exchange rate data were examined and found to exhibit 'stochastic' trends. In addition, a random walk was found to provide the best

fit to this data, when minimising the sum of squared errors from an exponential smoothing model. In general, a random walk often provides a good fit to data with stochastic trends, although even better fits are usually obtained from more general model formulations, such as the ARIMA models of Chapter **??**.

2.2.2 Definition

Let $\{x_t\}$ be a time series. Then $\{x_t\}$ is a random walk if:

$$x_t = x_{t-1} + \epsilon_t \tag{2.3}$$

where $\{\epsilon_t\}$ is a white noise series. Substituting $x_{t-1} = x_{t-2} + \epsilon_{t-1}$ in Equation (2.3), and then substituting for x_{t-2}, followed by x_{t-3} and so on (a process known as 'back substitution') gives:

$$x_t = \epsilon_t + \epsilon_{t-1} + \epsilon_{t-2} + \ldots \tag{2.4}$$

In practice, the above series will not be infinite but will start at some time $t = 1$. Hence,

$$x_t = \epsilon_1 + \epsilon_2 + \ldots + \epsilon_t \tag{2.5}$$

'Back substitution' is used to define more complex time series models and also to derive second-order properties. The procedure occurs so frequently in the study of time series models that the following definition is needed.

2.2.3 The backward shift operator

The *backward shift* operator $\mathbf{B}$ is defined by:

$$\mathbf{B}x_t = x_{t-1} \tag{2.6}$$

For obvious reasons, the backward shift operator is sometimes called the 'lag operator'. By repeatedly applying $\mathbf{B}$, it follows that:

$$\mathbf{B}^n x_t = x_{t-n} \tag{2.7}$$

Using $\mathbf{B}$, Equation (2.3) can be rewritten:

$$x_t = \mathbf{B}x_t + \epsilon_t \Rightarrow (1 - \mathbf{B})x_t = \epsilon_t \Rightarrow x_t = (1 - \mathbf{B})^{-1}\epsilon_t$$

$$\Rightarrow x_t = (1 + \mathbf{B} + \mathbf{B}^2 + \ldots)\epsilon_t \Rightarrow x_t = \epsilon_t + \epsilon_{t-1} + \epsilon_{t-2} + \ldots$$

and Equation (2.4) is recovered.

2.2.4 Random walk: Second-order properties

The second-order properties of a random walk follow as:

$$\left.\begin{aligned} \mu_x &= 0 \\ \gamma_k(t) &= \mathrm{cov}(x_t, x_{t+k}) = t\sigma^2 \end{aligned}\right\} \tag{2.8}$$

The covariance is a function of time, so the process is non-stationary. The lag k autocorrelation follows from (2.8) as:

$$\rho_k(t) = \frac{\mathrm{cov}(x_t, x_{t+k})}{\sqrt{\mathrm{var}(x_t)\mathrm{var}(x_{t+k})}} = \frac{t\sigma^2}{\sqrt{t\sigma^2(t+k)\sigma^2}} = \frac{1}{\sqrt{1+k/t}} \tag{2.9}$$

so that for large t, $\rho_1 \approx 1$, and, in general, $\rho_k < \rho_{k-1}$, so that for large t with $k \ll t$, $\rho_k \approx 1$. Hence, the correlogram for a random walk is characterised by positive autocorrelations that decay very slowly down from unity. This is demonstrated by simulation in Section 2.2.7.

2.2.5 Derivation of second-order properties

Equation (2.5) is a finite sum of white noise terms, each with zero mean and variance σ^2. Hence, the mean of x_t is zero (Equation 2.8). The autocovariance in (2.8) can be derived using Equation (1.4) as follows:

$$\gamma_k = \mathrm{cov}(x_t, x_{t+k}) = \mathrm{cov}(\sum_{i=1}^{t} \epsilon_i, \sum_{j=1}^{t+k} \epsilon_j) = \sum_{i \neq j} \mathrm{cov}(\epsilon_i, \epsilon_j) = t\sigma^2$$

2.2.6 The difference operator

Differencing adjacent terms of a series can transform a non-stationary series to a stationary series. For example, if the series $\{x_t\}$ is a random walk it is non-stationary. However, from (2.3), the first-order differences of $\{x_t\}$ produce the stationary white noise series $\{\epsilon_t\}$ given by: $x_t - x_{t-1} = \epsilon_t$. Hence, differencing turns out to be a useful 'filtering' procedure in the study of non-stationary time series. The difference operator ∇ is defined by:

$$\nabla x_t = x_t - x_{t-1} \tag{2.10}$$

Note that $\nabla x_t = (1 - \mathbf{B})x_t$, so that ∇ can be expressed in terms of the backward shift operator $\mathbf{B}$. In general, higher-order differencing can be expressed as:

$$\nabla^n = (1 - \mathbf{B})^n \tag{2.11}$$

The proof of the last result is left to Exercise 4.

2.2.7 Simulation

It is often helpful to study a time series model by simulation. This enables the main features of the model to be 'observed' in plots, so that when historical data exhibit similar features the model may be selected as a potential candidate. The following commands can be used to simulate random walk data for `x`.

```
> x = e = rnorm(1000)
> for (t in 2:1000) x[t] = x[t-1] + e[t]

> plot(x, type="l")
```

The first command above places a white noise series into `e`, and uses this series to initialise `x`. The 'for' loop then generates the random walk using Equation (2.3) – the correspondence between the `R` code above and (2.3) should be noted. The series is plotted and shown in Figure 2.3.

A correlogram of the series is obtained from `acf(x)`, and is shown in Figure 2.4 – a gradual decay in the correlations is evident in the figure, thus supporting the theoretical results in Section 2.2.4.

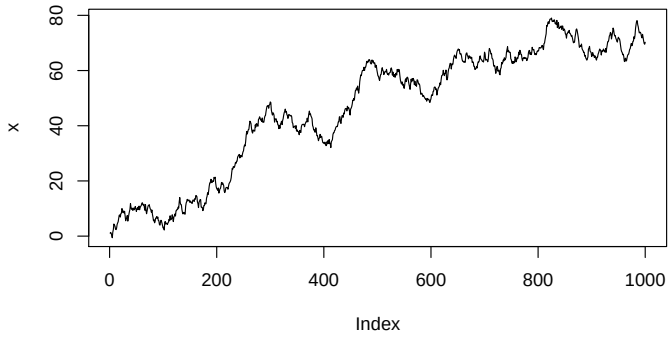


Fig. 2.3. Time plot of a simulated random walk. The series exhibits an increasing trend. However, this is purely 'stochastic' and due to the high serial correlation.

Throughout this book we will often fit models to data that we have simulated and attempt to recover the underlying model parameters. At first sight this might seem odd given that the parameters are used to simulate the data so that we already know at the outset the values the parameters should take. However, the procedure is useful for a number of reasons. In particular, to be able to simulate data using a model requires that the model formulation

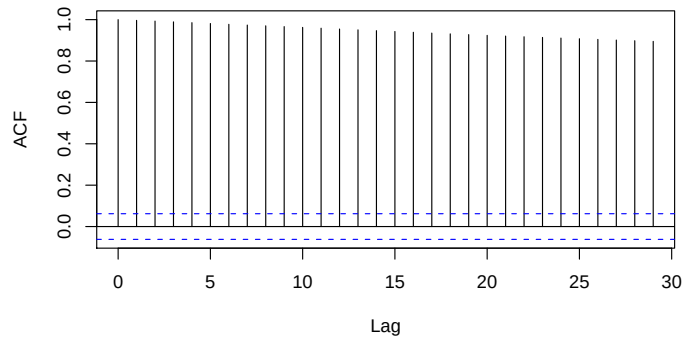


Fig. 2.4. The correlogram for the simulated random walk. A gradual decay from a high serial correlation is a notable feature of a random walk series.

is correctly understood. On the other hand, if the model is understood but incorrectly implemented, then the parameter estimates from the fitted model may deviate significantly from the underlying model values used in the simulation. Simulation can therefore help ensure that the model is both correctly understood and correctly implemented.

2.2.8 Fitted models and diagnostic plots

Simulated random walk series

The first order differences of a random walk are a white noise series, so the correlogram of the series of differences can be used to assess whether a given series is a random walk.

```
> acf(diff(x))
```

As can be seen in Figure 2.5, there are no obvious patterns in the correlogram, with only one marginally significant value at lag 2. This 'significant' value can be ignored because it is small in magnitude and about 5% of the values are expected to be statistically significant even when the underlying values are zero (Section 1.5.3). Thus, as expected, there is good evidence that the simulated series in `x` follows a random walk.

Exchange rate series

The correlogram of the first-order differences of the exchange rate data from Section 1.4.3 can be obtained from `acf(diff(z.ts))` and are shown in Figure 2.6.

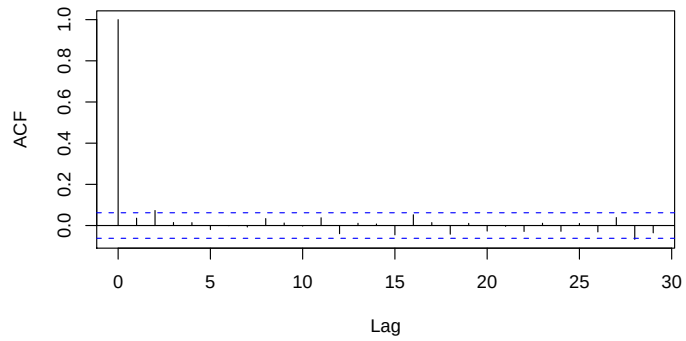


Fig. 2.5. Correlogram of differenced series. If a series follows a random walk the differenced series will be white noise.

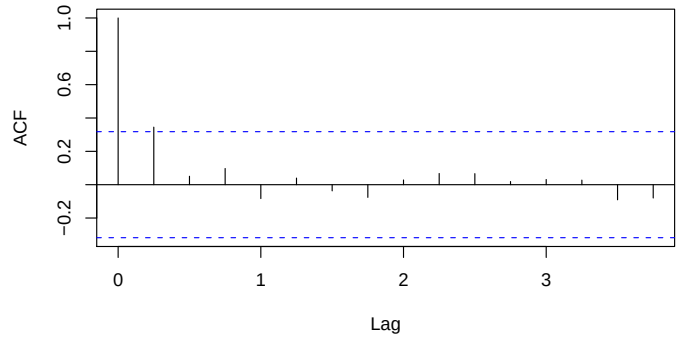


Fig. 2.6. Correlogram of first-order differences of the exchange rate series (UK pounds to NZ dollars: 1991-2000). The significant value at lag 1 indicates that an extension of the random walk model is needed for this series.

A significant value occurs at lag 1, suggesting that a more complex model may be needed, although the lack of any other significant values in the correlogram does suggest that the random walk provides a good approximation for the series (Figure 2.6). An additional parameter can be added to the random walk using the Holt-Winters procedure, allowing the parameter β to be non-zero but still forcing the seasonal term γ to be zero:

```
> z.hw = HoltWinters(z.ts, gamma=0)
> acf(resid(z.hw))
```

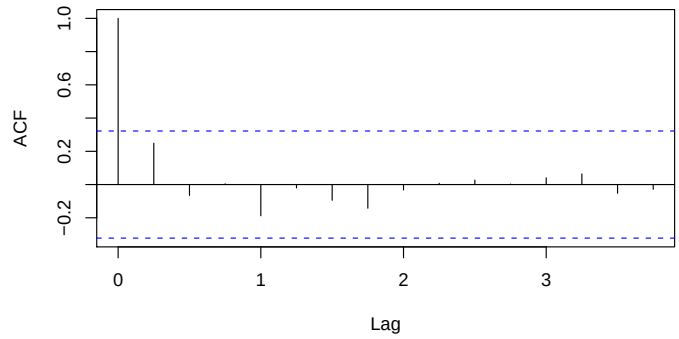


Fig. 2.7. The correlogram of the residuals from the fitted Holt-Winters model for the exchange rate series (UK pounds to NZ dollars: 1991-2000). There are no significant correlations in the residual series, so the model provides a good approximation to the exchange rate data.

Figure 2.7 shows the correlogram of the residuals from the fitted Holt-Winters model. This correlogram suggests that the residual series is white noise, implying that the fitted model provides a good fit to the data (Figure 2.7). Using Equation (1.15), with the parameter estimates obtained from `z.hw$alpha` and `z.hw$beta`, the fitted model can be expressed as:

$$\left.\begin{aligned} x_t &= x_{t-1} + b_{t-1} + \epsilon_t \\ b_{t-1} &= 0.167(x_{t-1} - x_{t-2}) + 0.833 b_{t-2} \end{aligned}\right\} \tag{2.12}$$

where $\{\epsilon_t\}$ is white noise with zero mean.

After some algebra, Equations (2.12) can be expressed as one equation in terms of the backward shift operator:

$$(1 - 0.167\mathbf{B} + 0.167\mathbf{B}^2)(1 - \mathbf{B})x_t = \epsilon_t \tag{2.13}$$

Equation (2.13) is a special case – the *integrated autoregressive* model – within the important class of models known as ARIMA models (Chapter **??**). The proof of (2.13) is left to Exercise 5.

2.3 Autoregressive models

2.3.1 Definition

The series $\{x_t\}$ is an autoregressive process of order p, abbreviated to AR(p), if:

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \ldots + \alpha_p x_{t-p} + \epsilon_t \tag{2.14}$$

where $\{\epsilon_t\}$ is white noise, and the α_i are the model parameters with $\alpha_p \neq 0$ for an order p process. Equation (2.14) can be expressed in terms of the backward shift operator:

$$\left.\begin{aligned}(1 - \alpha_1\mathbf{B} - \alpha_2\mathbf{B}^2 - \ldots - \alpha_p\mathbf{B}^p)x_t = \epsilon_t \\ \Rightarrow \theta_p(\mathbf{B})x_t = \epsilon_t\end{aligned}\right\} \tag{2.15}$$

where $\theta_p(\mathbf{B})$ is a polynomial of order p. The following points should be noted:

(a) The random walk is the special case AR(1) with $\alpha_1 = 1$, $\alpha_i = 0$ for all $i = 2, 3, \ldots, p$ (c.f. Equation 2.3).
(b) The exponential smoothing model is the special case $\alpha_i = \alpha(1-\alpha)^i$ for $i = 1, 2, \ldots$ and $p \to \infty$. (c.f. Equation 1.12).
(c) The model is a regression of x_t on past terms from the same series; hence, the use of the term 'autoregressive'.
(d) A prediction (or forecast) at time t is given by:

$$\hat{x}_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \ldots + \alpha_p x_{t-p} \tag{2.16}$$

(e) The model parameters can be estimated by minimising the sum of squared errors (Equation 1.14).

Stationary and non-stationary AR processes

The roots of the polynomial θ_p in Equation (2.15) must *all* exceed unity in *absolute* value for the process to be *stationary*. This result is essentially a generalisation of the result for the random walk, i.e. the random walk has $\theta = 1 - \mathbf{B}$ with root $\mathbf{B} = 1$ and is *non-stationary*. The following two examples illustrate the procedure for determining whether an AR process is stationary or non-stationary:

1. The AR(2) model $x_t = x_{t-1} - \frac{1}{4}x_{t-2} + \epsilon_t$ is stationary. The proof of this result is obtained by first expressing the model in terms of the backward shift operator: $\frac{1}{4}(\mathbf{B}^2 - 4\mathbf{B} + 4)x_t = \epsilon_t$, i.e. $\frac{1}{4}(\mathbf{B} - 2)^2 x_t = \epsilon_t$. The roots of the polynomial are given by solving $\theta(\mathbf{B}) = \frac{1}{4}(\mathbf{B} - 2)^2 = 0$, and are therefore obtained as: $\mathbf{B} = 2$. As the roots are greater than unity this AR(2) model is stationary.

2. The model $x_t = \frac{1}{2}x_{t-1} + \frac{1}{2}x_{t-2} + \epsilon_t$ is non-stationary, because one of the roots is unity. To prove this, first express the model in terms of the backward shift operator: $-\frac{1}{2}(\mathbf{B}^2+\mathbf{B}-2)x_t = \epsilon_t$, i.e. $-\frac{1}{2}(\mathbf{B}-1)(\mathbf{B}+2)x_t = \epsilon_t$. The polynomial $\theta(\mathbf{B}) = -\frac{1}{2}(\mathbf{B}-1)(\mathbf{B}+2)$ has roots $\mathbf{B} = 1, -2$. As there is a *unit root* ($\mathbf{B} = 1$) the model is *non-stationary*. Note: the other root ($\mathbf{B} = -2$) exceeds unity in *absolute* value, so only the presence of the unit root makes this process non-stationary.

2.3.2 Second-order properties of an AR(1) model

From (2.14), the AR(1) process is given by:

$$x_t = \alpha x_{t-1} + \epsilon_t \tag{2.17}$$

where $\{\epsilon_t\}$ is a white noise series with mean zero and variance σ^2. It can be shown (Section 2.3.3) that the second-order properties follow as:

$$\left.\begin{aligned} \mu_x &= 0 \\ \gamma_k &= \alpha^k\sigma^2/(1-\alpha^2) \end{aligned}\right\} \tag{2.18}$$

2.3.3 Derivation of second-order properties for an AR(1) process

Using $\mathbf{B}$, an AR(1) process can be written:

$$\left.\begin{aligned} (1-\alpha\mathbf{B})x_t &= \epsilon_t \\ \Rightarrow x_t &= (1-\alpha\mathbf{B})^{-1}\epsilon_t \\ &= \epsilon_t + \alpha\epsilon_{t-1} + \alpha^2\epsilon_{t-2} + \ldots = \textstyle\sum_{i=0}^{\infty}\alpha^i\epsilon_{t-i} \end{aligned}\right\} \tag{2.19}$$

Hence, the mean is given by

$$E(x_t) = E(\sum_{i=0}^{\infty}\alpha^i\epsilon_{t-i}) = \sum_{i=0}^{\infty}\alpha^i E(\epsilon_{t-i}) = 0$$

and, for $|\alpha| < 1$, the autocovariance follows as:

$$\begin{aligned} \gamma_k = \mathrm{cov}(x_t, x_{t+k}) &= \mathrm{cov}(\textstyle\sum_{i=0}^{\infty}\alpha^i\epsilon_{t-i}, \sum_{j=0}^{\infty}\alpha^j\epsilon_{t+k-j}) \\ &= \textstyle\sum_{j=k+i}\alpha^i\alpha^j\mathrm{cov}(\epsilon_{t-i}, \epsilon_{t+k-j}) \\ &= \alpha^k\sigma^2\textstyle\sum_{i=0}^{\infty}\alpha^{2i} = \alpha^k\sigma^2/(1-\alpha^2) \end{aligned}$$

using Equations (1.4) and (2.1).

2.3.4 Correlogram of an AR(1) process

From (2.18), the autocorrelation function follows as:

$$\rho_k = \alpha^k \quad (k \geq 0) \tag{2.20}$$

where $|\alpha| < 1$. Thus, the correlogram decays to zero, more rapidly for small α. The following example gives two correlograms for positive and negative values of α respectively (Figure 2.8):

```
> rho = function(k, alpha) alpha^k
> layout(1:2)
> plot(0:10, rho(0:10,  0.7), type="b")
> plot(0:10, rho(0:10, -0.7), type="b")
```

Try experimenting using other values for α. For example, use a small value of α to observe a more rapid decay to zero in the correlogram.

2.3.5 Partial autocorrelation

From (2.20) the autocorrelations are non-zero for all lags even though in the underlying model x_t only depends on the previous value x_{t-1} (Equation 2.17). The *partial autocorrelation* at lag k is the correlation that results after removing the effect of any correlations due to previous terms. For example, the partial autocorrelation of an AR(1) process will be zero for all lags greater than 1. In general, the partial autocorrelation at lag k is the kth coefficient of a fitted AR(k) model; if the underlying process is AR(p), then the coefficients α_k will be zero for all $k > p$. Thus, an AR(p) process has a correlogram of partial autocorrelations that 'cuts-off' to zero after lag p. Hence, a plot of the partial autocorrelations can be useful when determining the order of an underlying AR process. In `R`, the function `pacf` can be used to calculate the partial autocorrelations of a time series and produce a plot of the partial autocorrelations against lag (the 'partial-correlogram').

2.3.6 Simulation

An AR(1) process can be simulated in `R` as follows:

```
> set.seed(1)
> x = e = rnorm(100)
> for (t in 2:100) x[t] = 0.7*x[t-1] + e[t]

> plot(x); acf(x); pacf(x)
```

The resulting plots of the simulated data are shown in Figure 2.9, and give one possible realisation of the model. The partial-correlogram has no significant correlations, except the value at lag 1 as expected (Figure 2.9c). The difference

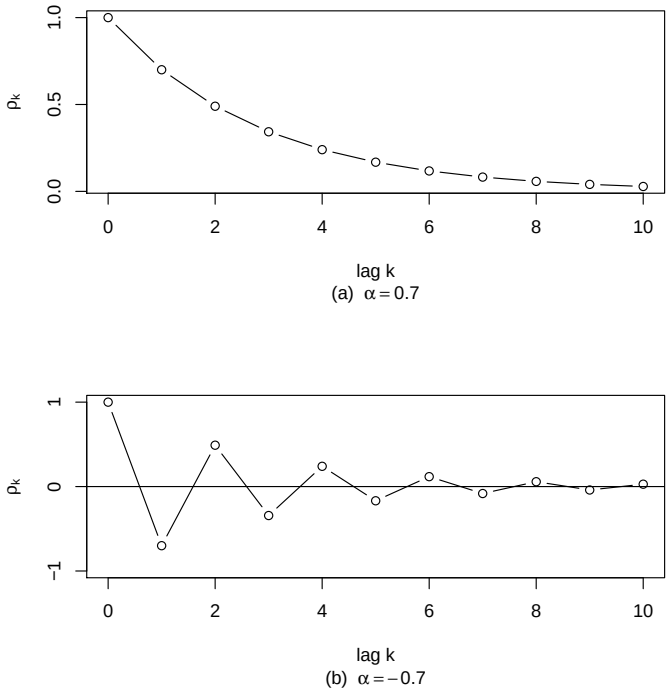


Fig. 2.8. Example correlograms for two autoregressive models: (a) $x_t = 0.7x_{t-1}+\epsilon_t$; (b) $x_t = -0.7x_{t-1} + \epsilon_t$

between the correlogram of the underlying model (Figure 2.8a) and the sample correlogram of the simulated series (Figure 2.9b) shows discrepancies that have arisen due to sampling variation. Try repeating the above commands several times to obtain a range of possible sample correlograms for an AR(1) process with underlying parameter $\alpha = 0.7$.

2.3.7 Fitted models

Model fitted to simulated series

An AR(p) model can be fitted to data in `R` using the `ar` function. In the code below the model is fitted to the simulated series of the last section, and an upper bound for an approximate 95% confidence for the underlying parameter is given:

```
> x.ar = ar(x, method="mle")
> x.ar$order; x.ar$ar
```

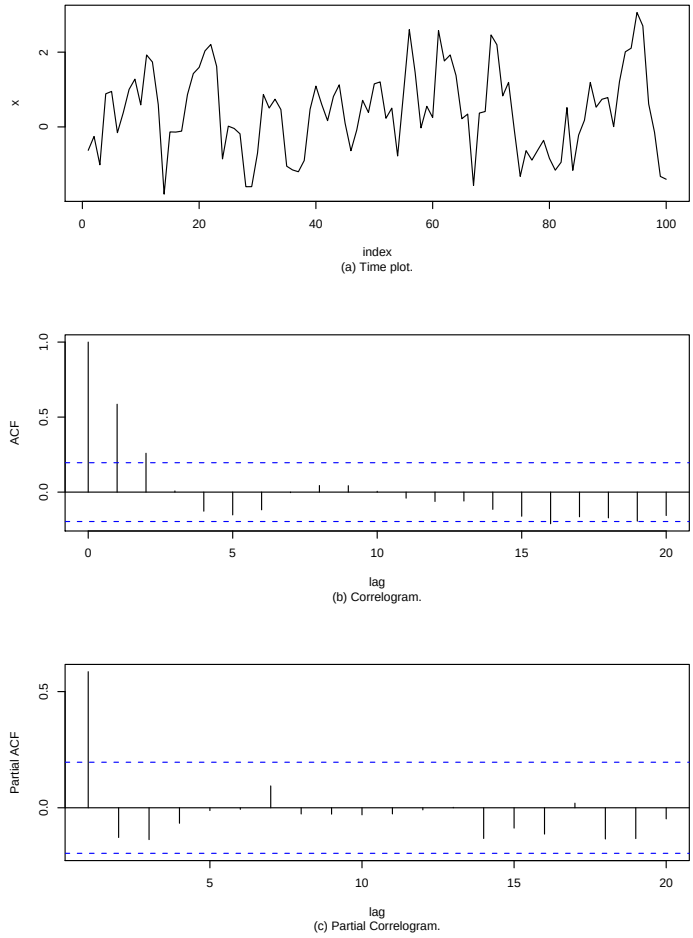


Fig. 2.9. A simulated AR(1) process: $x_t = 0.7x_{t-1} + \epsilon_t$. Note that in the partial-correlogram (c) only the first lag is significant, which is usually the case when the underlying process is AR(1).

```
[1] 1
[1] 0.601

> x.ar$ar + 2 * sqrt(x.ar$asy.var)
       [,1]
[1,] 0.7615
```

The method "mle" used in the fitting procedure above is based on maximising the likelihood function (the probability of obtaining the data given the

model) with respect to the unknown parameters. The order p of the process is chosen using the Akaike Information Criteria (AIC; Akaike (1974)), which penalises models with too many parameters:

$$\text{AIC} = -2 \times \text{log-likelihood} + 2 \times \text{number of parameters} \tag{2.21}$$

In the function `ar`, the model with the smallest AIC is selected as the best fitting AR model. Note that in the above code the correct order ($p = 1$) of the underlying process is recovered. The parameter estimate for the fitted AR(1) model is: $\hat{\alpha} = 0.60$. Whilst this is smaller than the underlying model value of $\alpha = 0.7$, the upper bound for the approximate 95% confidence interval (0.76) does contains the value of the model parameter as expected, giving us no reason to doubt the implementation of the model.

Exchange rate series: Fitted AR model

With the fitted model below the upper bound of the confidence interval indicates that there would not be sufficient evidence to reject the hypothesis $\alpha = 1$, thus supporting the earlier conclusion that a random walk provides a good description of this series:

```
> z.ar = ar(z.ts)
> mean(z.ts)
[1] 2.823
> z.ar$order; z.ar$ar
[1] 1
[1] 0.8903

> z.ar$ar + 2 * sqrt(z.ar$asy.var)
     [,1]
[1,] 1.04

> acf(z.ar$res[-1])
```

In the above code, a "-1" is used in the vector of residuals to remove the first item from the residual series. (For a fitted AR(1) model the first item has no predicted value, because there is no observation at $t = 0$; in general, the first p values will be 'not available' (NA) in the residual series of a fitted AR(p) model.)

By default, the mean is subtracted before the parameters are estimated, so a predicted value $\hat{z}_t$ at time t based on the above output is given by:

$$\hat{z}_t = 2.8 + 0.89(z_{t-1} - 2.8) \tag{2.22}$$

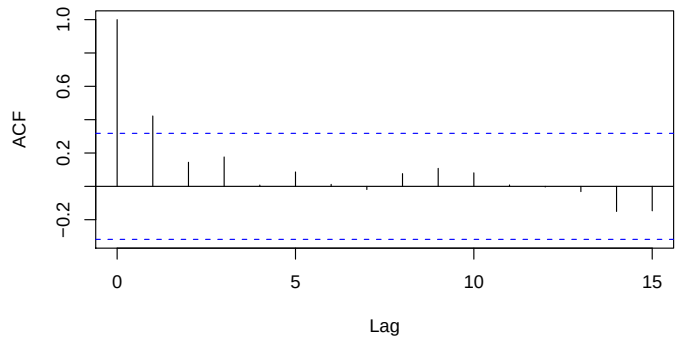


Fig. 2.10. The correlogram of residual series for the AR(1) model fitted to the exchange rate data.

2.3.8 Global temperature series: Fitted AR model

The global temperature series was introduced in Section 1.4.4, where it was apparent that the data exhibited an increasing trend after 1970, which may be due to the 'greenhouse effect'. It is of interest here to see how well this trend can be modelled without the use of deterministic functions, as this would give an indication of whether the trend could be considered 'random'.

Consider the following AR model fitted to the mean annual temperature series:

```
> global.ar = ar(aggregate(global.ts, FUN=mean), method="mle")
> mean(aggregate(global.ts, FUN=mean))
[1] -0.1383
> global.ar$order; global.ar$ar
[1] 4
[1] 0.58762 0.01260 0.11117 0.26764

> acf(global.ar$res[-(1:global.ar$order)], lag=50)
```

Based on the above output a predicted mean annual temperature $\hat{x}_t$ at time t is given by:

$$\begin{aligned}\hat{x}_t = &-0.14 + 0.59(x_{t-1} + 0.14) + 0.013(x_{t-2} + 0.14) \\ &+0.11(x_{t-3} + 0.14) + 0.27(x_{t-4} + 0.14)\end{aligned} \tag{2.23}$$

The correlogram of the residuals has only one (marginally) significant value at lag 27, so the underlying residual series could be white noise (Figure 2.11). Thus the fitted AR(4) model (2.23) provides a good fit to the data. As the

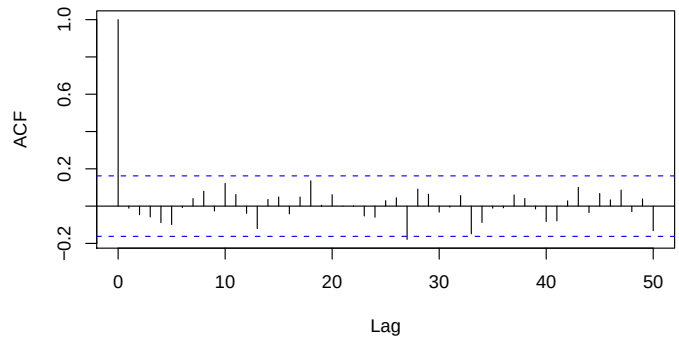


Fig. 2.11. The correlogram of the residual series for the AR(4) model fitted to the annual global temperature series. The correlogram is approximately white noise so that, in the absence of further information, a simple stochastic model can 'explain' the correlation and trends in the series.

AR model has no deterministic trend component, the trends in the data can be explained by serial correlation and random variation, implying that it is possible that these trends are stochastic (or could arise from a purely stochastic process). Of course, this does not imply that there is no underlying reason for the trends – if a valid scientific explanation is known, such as a link with the increased use of fossil fuels, then this information would clearly need to be included in any future forecasts of the series.

2.4 Summary of R commands

`set.seed`	sets a seed for the random number generator enabling a simulation to be reproduced
`rnorm`	simulates a white noise series
`diff`	creates a series of first-order differences
`ar`	gets the best fitting AR(p) model
`pacf`	extracts partial autocorrelations and partial-correlogram

2.5 Exercises

1. a) Simulate a time series of length 1000 for the following model, giving appropriate `R` code and placing the simulated data in a vector `x`:

$$x_t = \frac{5}{6}x_{t-1} - \frac{1}{6}x_{t-2} + \epsilon_t \tag{2.24}$$

b) Plot the correlogram and partial-correlogram for the simulated data. Comment on the plots.
c) Fit an AR model to the data in `x` giving the parameter estimates and order of the fitted AR process.
d) Construct 95% confidence intervals for the parameter estimates of the fitted model. Do the model parameters fall within the confidence intervals? Explain your results.
e) Is the model in Equation (2.24) stationary or non-stationary? Justify your answer.
f) Plot the correlogram of the residuals of the fitted model. Comment on the plot.

2. a) Show that the series $\{x_t\}$ given by $x_t = \frac{3}{2}x_{t-1} - \frac{1}{2}x_{t-2} + \epsilon_t$ is non-stationary.
b) Write down the model for $\{y_t\}$, where $y_t = \nabla x_t$. Show that $\{y_t\}$ is stationary.
c) Simulate a series of 1000 values for $\{x_t\}$, placing the simulated data in `x`, and use these simulated values to produce a series of 999 values for $\{y_t\}$, placing this series in the vector `y`.
d) Fit an AR model to `y`. Give the fitted model parameter estimates and a 95% confidence interval for the underlying model parameters based on these estimates. Compare the confidence intervals to the parameters used to simulate the data and explain the results.
e) Plot the correlogram of the residuals of the fitted model and comment.

3. a) Re-fit the AR(4) of Section 1.4.4 to the annual mean global temperature series, and using the fitted model create a series of predicted values from $t = 2$ to the last value in the series (using Equation 2.23). Create a residual series from the difference between the predicted value and the observed value and verify that within machine accuracy your residual series is identical to the series extracted from the fitted model in `R`.
b) Plot a correlogram and partial-correlogram for the mean annual temperature series. Comment on the plots.
c) Use the `predict` function in `R` to forecast 100 years of future values for the annual global temperature series using the fitted AR(4) model (2.23) of Section 1.4.4.
d) Create a time plot of the mean annual temperature series and add the 100-year forecasts to the plot (use a different colour or symbol for the forecasts).
e) Add a line representing the overall mean global temperature. Comment on the final plot and on any potential inadequacies in the fitted model.

4. Prove Equation (2.11) by mathematical induction as follows. (i) First, show that if (2.11) holds for $n = k$ then it also holds for $n = k + 1$. (ii) Next, show that (2.11) holds for the case $n = 2$, and hence (from (i)) holds for all n.

5. Prove Equation (2.13). [Hint: Express the two equations in (2.12) in terms of the backward shift operator, then substitute for b_n.]

3

Time Series Regression

Recall from Chapters 1 and 2, that trends in time series can be classified into two main types: *stochastic* and *deterministic*. A trend is stochastic when the underlying cause is not understood, and can only be attributed to high serial correlation with random error. Trends of this type, which are common in financial series, can be simulated in `R` using models such as the random walk or autoregressive process (Chapter 2). Other trends are deterministic, so-called because there is a greater understanding of the underlying cause of the trend. For example, a deterministic increasing trend in the data may be related to an increasing population, or a regular cycle may be related to a known seasonal frequency. Deterministic trends and seasonal variation can be modelled using regression, which is the main focus in this chapter.

Time series regression usually differs from a standard regression analysis because the residuals of the fitted model also form a time series and therefore tend to be serially correlated. If this correlation is positive and unaccounted for, the standard errors of the parameter estimates will be less than their true value, so the estimates will appear more accurate than they should, which can result in incorrect 'significant' values appearing in standard regression output. In this chapter, we use generalised least squares to allow for autocorrelation in the residual series and to obtain improved estimates of the standard errors of the fitted model parameters.

We begin the chapter by looking at linear models for trends, and then introduce regression models that account for seasonal variation using indicator and harmonic variables. The logarithmic transformation, which is often used to stabilise the variance, is also considered along with an appropriate inverse transform and bias correction factor needed for making forecasts. An introduction to non-linear models is also included in this chapter.

3.1 Linear models

3.1.1 Definition

A model for a time series $\{x_t : t = 1, \ldots n\}$ is *linear* if it can be expressed as:

$$x_t = \alpha_0 + \alpha_1 y_{t,1} + \alpha_2 y_{t,2} + \ldots + \alpha_p y_{t,p} + z_t \tag{3.1}$$

where $y_{t,i}$ is the value of the i-th explanatory variable at time t ($i = 1, \ldots, p$; $t = 1, \ldots, n$), z_t is the residual error at time t, and $\alpha_0, \alpha_1, \ldots, \alpha_p$ are model parameters, which can be estimated by least squares. Note that the residual errors form a time series $\{z_t\}$ which does not have to be Gaussian white noise. An example, of a linear model is the p-th order polynomial function of t:

$$x_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2 \ldots + \alpha_p t^p + z_t \tag{3.2}$$

This model is linear because the explanatory variables can be written: $y_{t,i} = t^i$ ($i = 1, \ldots, p$). The term 'linear' is therefore a reference to the summation of model parameters, each multiplied by an explanatory variable.

A simple special case of a linear model is the straight line model obtained by putting $p = 1$ in Equation (3.2): $x_t = \alpha_0 + \alpha_1 t + z_t$. In this case, the value of the line at time t is the trend m_t. For the more general polynomial, the trend at time t is the value of the underlying polynomial evaluated at t, so in (3.2) the trend is: $m_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2 \ldots + \alpha_p t^p$.

Many non-linear models can be transformed to linear models. For example, the model $x_t = e^{\alpha_0 + \alpha_1 t + z_t}$ can be transformed by taking natural logarithms to obtain a linear model for the transformed series $\{x'_t\}$ given by:

$$x'_t = \log\, x_t = \alpha_0 + \alpha_1 t + z_t \tag{3.3}$$

In (3.3), standard least squares regression could then be used to fit a linear model (i.e. estimate the parameters α_0 and α_1) and make predictions for x'_t. To make predictions for x_t the inverse transform needs to be applied to x'_t, which in this example is $e^{x'_t}$. However, this usually has the effect of biasing the predictions as the least squared residual errors are also transformed within the non-linear function for x_t (Equation 3.3). In Section 3.6.3 suitable factors are calculated to correct this bias.

Most natural processes that generate time series are probably non-linear. In addition, many non-linear functions cannot be transformed to linear functions through a simple operation such as taking logs. For example, the model $x_t = e^{\alpha_0 + \alpha_1 t} + z_t$ cannot be transformed to a linear model by taking logs, because the error term is added to a non-linear function of t. However, Taylor's Theorem in Calculus shows that a large class of non-linear functions can be written as linear functions expressed as a sum of polynomial terms. Hence, a linear model will often provide a good approximation even when the underlying process is non-linear. Consequently, linear models play a central role in the study of time series.

3.1.2 Stationarity and differencing

Linear models for time series are non-stationary when they include functions of time. However, differencing can often transform a non-stationary series with a deterministic trend to a stationary series. For example, if the time series $\{x_t\}$ is given by the straight line function: $x_t = \alpha_0 + \alpha_1 t + z_t$, then the first-order differences are given by:

$$\nabla x_t = x_t - x_{t-1} = z_t - z_{t-1} + \alpha_1 \tag{3.4}$$

The series $\{\nabla x_t\}$ is stationary because it is not a function of t (assuming the error series $\{z_t\}$ is stationary). In Section 2.2.6 we found that first-order differencing can transform a non-stationary series with a stochastic trend (the random walk) to a stationary series. Thus, differencing turns out to be a very useful general procedure as it can 'remove' *both* stochastic and deterministic trends. If the underlying trend is a polynomial of order p, then p-th order differencing will be required to remove it.

3.1.3 Simulation

We have already seen that simulation is a useful tool when studying time series (Section 2.1.3). In time series regression, it is common for the residual error series ($\{z_t\}$ in Equation 3.1) to be autocorrelated. In the code below, a time series that has an underlying straight line trend ($50 + 3t$) and AR(1) residual errors is simulated and plotted:

```
> set.seed(1)
> z = e = rnorm(100, sd=20)
> for (t in 2:100) z[t] = 0.8*z[t-1] + e[t]

> t = 1:100
> x = 50 + 3*t + z

> plot(x, xlab="time", type="l")
```

The model for the code above can be expressed as: $x_t = 50 + 3t + z_t$, where $\{z_t\}$ is the AR(1) process $z_t = 0.8z_{t-1} + \epsilon_t$ and $\{\epsilon_t\}$ is Gaussian white noise with $\sigma = 20$. A time plot of a realisation of $\{x_t\}$ is given in Figure 3.1.

3.1.4 Fitted models

Model fitted to simulated data

Linear models are usually fitted by minimising the sum of squared errors, as defined in Equation (1.14), which can be achieved in `R` using the function `lm`. In the code below, a linear model is fitted to the simulated series of the previous section, and an approximate 95% confidence interval for the estimated parameters extracted:

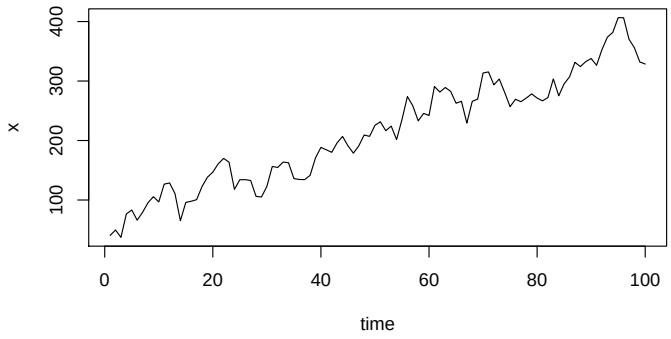


Fig. 3.1. Time plot of a simulated time series with a straight line trend and AR(1) residual errors

```
> x.lm = lm(x ~ t)

> coef(x.lm)
(Intercept)           t
      58.55        3.06

> confint(x.lm)
            2.5 % 97.5 %
(Intercept) 48.87  68.24
t            2.90   3.23
```

The above confidence intervals are likely to be slightly too narrow due to the autocorrelation in the residual series.

The function `summary` can be used to obtain a standard regression output – to observe this, type `summary(x.lm)` after entering the commands above. Specific information can also be extracted using `summary`. For example, the coefficients and their standard errors with respective t-tests, the residual standard deviation, and R^2 can all be extracted as follows:

```
> summary(x.lm)$coef
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    58.55     4.8801    12.0 6.06e-21
t               3.06     0.0839    36.5 7.27e-59

> summary(x.lm)$sigma
[1] 24.2
> summary(x.lm)$r.sq
[1] 0.932
```

It is appropriate after fitting a regression model to consider various diagnostic plots. In the case of time series regression, important diagnostic plots include the correlogram of the residuals and the partial-correlogram:

```
> acf(resid(x.lm)); pacf(resid(x.lm))
```

As expected, on viewing the correlograms we find that the residual time series is autocorrelated (Figure 3.2). In Figure 3.3, only the lag 1 partial autocorrelation is significant, which implies that the residual series follows an AR(1) process. Again this should not be surprising, given that we used an AR(1) process to simulate this series.

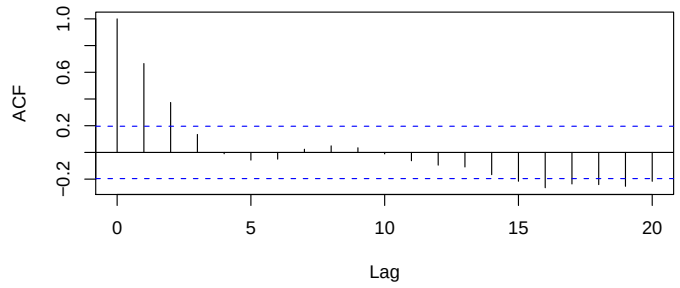


Fig. 3.2. The residual correlogram for the fitted straight line model.

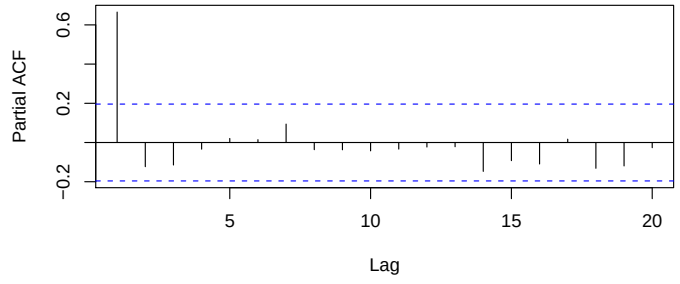


Fig. 3.3. The residual partial-correlogram for the fitted straight line model. The significant value at lag 1 indicates the residuals follow an AR(1) process, as expected.

Model fitted to the temperature series: 1970–2005

In Section 1.4.4 we extracted global temperature data for the more recent period (1970–2005) for which there appears to be an increasing trend. In this section, we would like to establish whether there is supporting statistical evidence for the apparent trend over this period, which can be assessed by considering an approximate 95% confidence interval for the slope as follows:

```
> temp.lm = lm(temp ~ time(temp))
> confint(temp.lm)
                 2.5 %   97.5 %
(Intercept) -37.2100 -32.6308
time(temp)    0.0165   0.0188
```

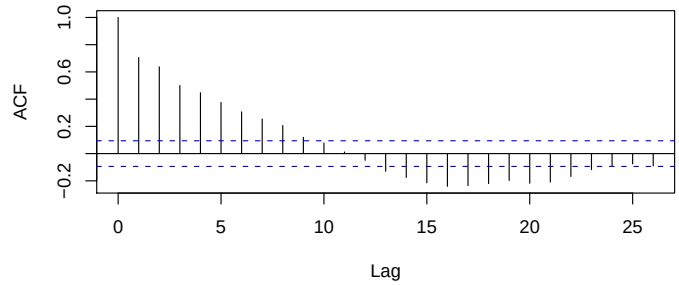


Fig. 3.4. Residual correlogram for the regression model fitted to the global temperature series: 1970–2005.

The confidence interval for the 'time' variable above does not contain zero – the bounds are both positive. Hence, based on this interval, there is statistical evidence that the slope is positive, i.e. there is statistical evidence of an increasing trend in global temperatures for the period 1970–2005. However, this deduction may be premature because the correlogram indicates that the residuals are autocorrelated (Figure 3.4):

```
> acf(resid(lm(temp ~ time(temp))))
```

If the autocorrelation in the residual series is not accounted for, then the standard errors used to create confidence intervals, such as those above, or any t-ratios, will be under-estimated. A mathematical explanation follows.

3.1.5 Autocorrelation and the estimation of sample statistics

To illustrate the effect of autocorrelation in estimation, the sample mean will be used as it is straightforward to analyse and used in the calculation of other statistical properties.

Suppose $\{x_t : t = 1, \ldots, n\}$ is a time series of *independent* random variables with mean $E(x_t) = \mu$ and variance $\text{var}(x_t) = \sigma^2$. Then it is well-known that the sample mean $\bar{x} = \sum_{t=1}^{n} x_t/n$ has $E(\bar{x}) = \mu$ and $\text{var}(\bar{x}) = \sigma^2/n$. Now let $\{x_t : t = 1, \ldots, n\}$ be a stationary time series with $E(x_t) = \mu$, $\text{var}(x_t) = \sigma^2$, and autocorrelation function $\text{cor}(x_t, x_{t+k}) = \rho_k$. Then, the variance of the sample mean is given by:

$$\text{var}(\bar{x}) = \frac{\sigma^2}{n}\left\{1 + 2\sum_{k=1}^{n-1}(1 - k/n)\rho_k\right\} \tag{3.5}$$

In Equation (3.5) the variance σ^2/n for an independent random sample arises as the special case when $\rho_k = 0$ for all $k > 0$. If $\rho_k > 0$ then $\text{var}(\bar{x}) > \sigma^2/n$, so that the estimate for μ is less accurate for a positively autocorrelated series than for a purely random series of the same length. Conversely, if $\rho_k < 0$ then the variance of the estimate may actually be smaller than the variance obtained from a random sample of the same size. This latter result is caused by negative autocorrelation, which produces a tendency for a value above the mean to be followed by a value below the mean, thus providing a more 'efficient' estimate for the overall mean level.

Equation (3.5) can be proved using Equation (1.4) and the properties of variance as follows:

$$\begin{aligned}
\text{var}(\bar{x}) &= \text{var}\{(x_1 + x_2 + \cdots + x_n)/n^2\} = \text{var}(x_1 + x_2 + \cdots + x_n)/n^2 \\
&= n^{-2}\text{cov}(\textstyle\sum_{i=1}^{n} x_i, \sum_{j=1}^{n} x_j) \quad = n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n}\text{cov}(x_i, x_j) \\
&= n^{-2}\big\{ \quad \gamma_0 \quad + \gamma_1 \quad + \cdots + \gamma_{n-2} + \gamma_{n-1} \\
&\qquad + \gamma_1 \quad + \gamma_0 \quad + \cdots + \gamma_{n-3} + \gamma_{n-2} \\
&\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
&\qquad + \gamma_{n-2} + \gamma_{n-3} + \cdots + \gamma_2 \quad + \gamma_1 \\
&\qquad + \gamma_{n-1} + \gamma_{n-2} + \cdots + \gamma_1 \quad + \gamma_0 \quad \big\} \\
&= n^{-2}\big\{ \; n\gamma_0 + 2\textstyle\sum_{k=1}^{n-1}(n-k)\,\gamma_k \; \big\}
\end{aligned}$$

Equation (3.5) follows after substituting $\gamma_0 = \sigma^2$ and $\rho_k = \gamma_k/\sigma^2$ in the last line above.

3.1.6 Generalised least squares

We have seen that in time series regression it is common and expected that the residual series will be autocorrelated. For a positive serial correlation in the residual series, this implies that the standard errors of the estimated regression parameters are likely to be underestimated (Equation 3.5), and should therefore be corrected.

A fitting procedure known as *generalised least squares* (GLS) can be used to provide better estimates of the standard errors of the regression parameters to account for autocorrelation in the residuals. The procedure is essentially based on maximising the likelihood function, conditional on the autocorrelation, and is implemented in the `R` function: `gls` (located in the `nlme` library). We illustrate the use of `gls` in the following example.

GLS fit to simulated series

In Section 3.1.3, we simulated a time series that had an underlying straight-line trend and a residual series that followed an AR(1) process (with a lag 1 autocorrelation of 0.8). A linear model can be fitted, allowing for an AR(1) autocorrelated residual series, using the `gls` function:

```
> library(nlme)

> x.gls = gls(x ~ t, cor = corAR1(0.8))

> coef(x.gls)
(Intercept)           t
      58.23        3.04
> summary(x.gls)$sigma
[1] 25.6
> confint(x.gls)
            2.5 % 97.5 %
(Intercept) 34.86  81.60
t            2.65   3.44
```

In the above output, the estimate of the residual standard deviation exceeds that obtained in Section 3.1.4, which results in standard errors of the parameter estimates being larger than those obtained from OLS using `lm` and wider confidence intervals for the model parameters.

A slight discrepancy exists between the estimated parameters obtained from GLS compared with those obtained from OLS. For example, the slope is estimated as 3.06 using `lm` but 3.04 using `gls`. Where discrepancies in the model parameter estimates exist, the *ordinary* least squares estimates should be used in preference to the GLS estimates because they tend to be computationally more robust. The primary use of GLS is therefore in the

estimation of the standard errors and adjusting the confidence intervals for the parameters.

For an historical series the lag 1 autocorrelation would need to be first estimated from the correlogram of the residuals of a fitted linear model. In the next example, we illustrate this procedure using the global temperature series.

Confidence interval for the trend in the temperature series

To calculate an approximate 95% confidence interval for the trend in the global temperature series (1970–2005), OLS is first used to estimate the trend and GLS is then used to estimate the standard error, accounting for the autocorrelation in the residual series (Figure 3.4). In the `gls` function, we approximate the residual series as an AR(1) process with a lag 1 autocorrelation of 0.7 read from Figure 3.4:

```
> temp.gls = gls(temp ~ time(temp), cor = corAR1(0.7))

> confint(temp.gls)
                2.5 %   97.5 %
(Intercept) -39.8057 -28.4966
time(temp)    0.0144   0.0201
```

Although the above confidence interval is now wider than it was in Section 3.1.4, zero is still not contained in the interval for the slope. This implies, as before, that the slope is statistically significant and that there is statistical evidence of an increasing trend in the global temperatures over the period 1970–2005.

3.2 Linear models with seasonal variables

3.2.1 Introduction

From the previous chapters, it is clear that time series records often extend over many years. For example, monthly global temperatures were available for the 150-year period: 1856–2005. If the sampling interval is smaller than a year, e.g. if the data are measured over monthly time intervals, then it is likely that seasonal variation will be present in the series.

In this section, we consider two regression methods suitable for seasonally varying time series. The first is based on using indicator (or 'dummy') variables, and provides a method of measuring a constant additive seasonal factor in each time interval. The second is based on using sine and cosine functions of time to account for seasonal changes. The latter approach is particularly useful and parameter efficient when the seasonal effect varies 'smoothly'.

3.2.2 Additive seasonal indicator variables

Suppose a time series contains s seasons. For example, $s = 12$ for time series measured over each calendar month, $s = 4$ for quarterly data, and $s = 2$ for series measured over six-month intervals (corresponding to two seasons: 'summer' and 'winter'). A seasonal indicator model for a time series $\{x_t : t = 1, \ldots, n\}$, with seasonal effect s_t and trend m_t, can be written as:

$$x_t = m_t + s_t + z_t \tag{3.6}$$

where $s_t = \beta_i$ when t falls in the i-th season ($t = 1, \ldots, n; i = 1, \ldots, s$), and $\{z_t\}$ is the residual error series, which may be autocorrelated. This model takes the same form as the additive classical decomposition model (Equation 1.5), but differs in that the components m_t and s_t are formulated with model parameters that need to be estimated.

In (3.6), m_t will be taken to be the trend without a constant term, so that m_t could be a polynomial of order p with parameters: $\alpha_1, \ldots, \alpha_p$ (refer back to Equation 3.2 for the definition of a polynomial trend). Instead, the constant for the model will effectively be included within the seasonal terms s_t. Equation (3.6) is therefore equivalent to a polynomial trend in which the constant term depends on the season, so that the s seasonal parameters $(\beta_1, \ldots, \beta_s)$ correspond to s possible constants in the polynomial equation (3.2). Equation (3.6) can therefore be re-written as:

$$x_t = m_t + \beta_{1+(t-1) \bmod s} + z_t \tag{3.7}$$

For example, with a monthly time series $\{x_t\}$, beginning with $t = 1$ at January, a seasonal indicator model with a straight line trend is given by:

$$x_t = \alpha_1 t + s_t + z_t = \begin{cases} \alpha_1 t + \beta_1 + z_t & t = 1, 13, \ldots \\ \alpha_1 t + \beta_2 + z_t & t = 2, 14, \ldots \\ \vdots & \\ \alpha_1 t + \beta_{12} + z_t & t = 12, 24, \ldots \end{cases} \tag{3.8}$$

In `R`, the parameters for a seasonal indicator model can be estimated by OLS by treating the seasonal term s_t as a 'factor' in the model expression within `lm`. The `factor` function can be used with `cycle` to create the seasonal indicator explanatory variables for the regression model. The procedure is fairly straightforward, as the following example illustrates.

3.2.3 Example: Seasonal model for the temperature series

A regression model with a straight line trend and additive seasonal indices can be fitted to the temperature series (1970–2005) as follows:

```
> s = factor(cycle(temp))
> t = time(temp)
> temp.lm = lm(temp ~ 0 + t + s)
> coef(temp.lm)
       t       s1       s2       s3       s4       s5       s6
  0.0177 -34.9973 -34.9880 -35.0100 -35.0123 -35.0337 -35.0251
      s7       s8       s9      s10      s11      s12
-35.0269 -35.0248 -35.0383 -35.0525 -35.0656 -35.0487
```

A zero is used within the R-formula above to ensure that the straight line model does not include a constant term, and results in all twelve seasonal indices being listed in the above output. If this zero is left out, one of the seasonal terms terms would be dropped during the estimation procedure and an estimate for an overall constant would appear in the output. Both approaches actually give equivalent results and it is more a matter of personal preference which you use. The parameters can also be estimated by GLS by replacing `lm` with `gls` in the above code.

To forecast a future value, we can apply the above fitted model to a new series of times. For example, in the code below, a 2-year ahead future prediction for the temperature series is obtained by creating a new set of times using the `seq` function beginning at the year 2006, and then applying the fitted model function to these times. In the last line of the code below, the forecasts for the next six months in the series are extracted.

```
> new.t = seq(2006, len=2*12, by=1/12)

> alpha = coef(temp.lm)[1]
> beta  = rep(coef(temp.lm)[2:13], 2)

> (alpha * new.t + beta) [1:6]
   s1    s2    s3    s4    s5    s6
0.524 0.535 0.514 0.514 0.494 0.504
```

Alternatively, the `predict` function can be used to forecast values, provided the new data is correctly labelled within a `data.frame`, i.e. the names in the `data.frame` correspond to those used in the fitted model:

```
> new.dat  = data.frame(t = new.t, s=factor(rep(1:12, 2)))
> predict(temp.lm, new.dat)[1:6]
    1     2     3     4     5     6
0.524 0.535 0.514 0.514 0.494 0.504
```

As expected, both the approaches above give the same answer for each forecast, so in practice either (or both) can be used. The `predict` function is a generic function that can be applied to a range of different types of fitted time series models, so it is important to learn how to use `predict` correctly. We will revisit this function in Section 3.6.

3.3 Harmonic seasonal models

In the previous section, one parameter estimate is used per season. However, seasonal effects often vary smoothly over a year, so that it may be more parameter efficient to use a smooth function instead of separate indices for each season.

Sine and cosine functions can be used to build smooth variation into a seasonal model. A harmonic sine wave with frequency f (or period $1/f$), amplitude A and phase shift ϕ can be expressed as:

$$A\sin(2\pi ft + \phi) = \alpha_s \sin(2\pi ft) + \alpha_c \cos(2\pi ft) \tag{3.9}$$

where $\alpha_s = A\cos(\phi)$ and $\alpha_c = A\sin(\phi)$. The expression on the right-hand-side of Equation (3.9) is linear in the parameters α_s and α_c, whilst the left-hand-side is non-linear because the parameter ϕ is within the sine function. Hence, the expression on the right-hand-side is preferred in the formulation of a seasonal regression model, so that OLS can be used to estimate the parameters. For a time series $\{x_t\}$ with s seasons there are $[s/2]$ possible cycles. [1] A harmonic seasonal model for $\{x_t\}$ is defined by:

$$x_t = m_t + \sum_{i=1}^{[s/2]} \{s_i \sin(2\pi it/s) + c_i \cos(2\pi it/s)\} + z_t \tag{3.10}$$

where m_t is the trend which includes a parameter for the constant term, and s_i and c_i are unknown parameters. The trend may take a polynomial form as in Equation (3.2). It will not be possible to detect a higher frequency than $1/2$ (which is called the Nyquist frequency), unless data is available over smaller sampling intervals. Also, when s is an even number, the value of the sine at frequency $1/2$ (when $i = s/2$ in the summation term shown in Equation 3.10) will be zero for all values of t, and so the term can be left out of the model. Hence, with a constant term included, the maximum number of parameters in the harmonic model equals that of the seasonal indicator variable model (Equation 3.6).

At first sight it may seem strange that the harmonic model has cycles of a higher frequency than the seasonal frequency of $1/s$. However, the addition of further harmonics has the effect of perturbing the underlying wave to make it less regular than a standard sine wave of period s. This usually still gives a dominant seasonal pattern of period s, but with a more realistic underlying shape. For example, suppose data are taken at monthly intervals, then the second plot given below might be a more realistic underlying seasonal pattern than the first plot as it perturbs the standard sine wave by adding another two harmonic terms of frequencies 2/12 and 4/12 (Figure 3.5):

[1] The notation [ ] represents the integer part of the expression within. In most practical cases s is even and so [ ] can be omitted. However, for some 'seasons' s may be an odd number, making the notation necessary. For example, if the 'seasons' are the days of the week, there would be $[7/2] = 3$ possible cycles.

```
> t = seq(1,12, len=1000)
> plot (t, sin(2*pi*t/12), type="l")
> plot (t, sin(2*pi*t/12) + 0.2*sin(2*pi*2*t/12) +
  0.1*sin(2*pi*4*t/12) + 0.1*cos(2*pi*4*t/12), type="l")
```

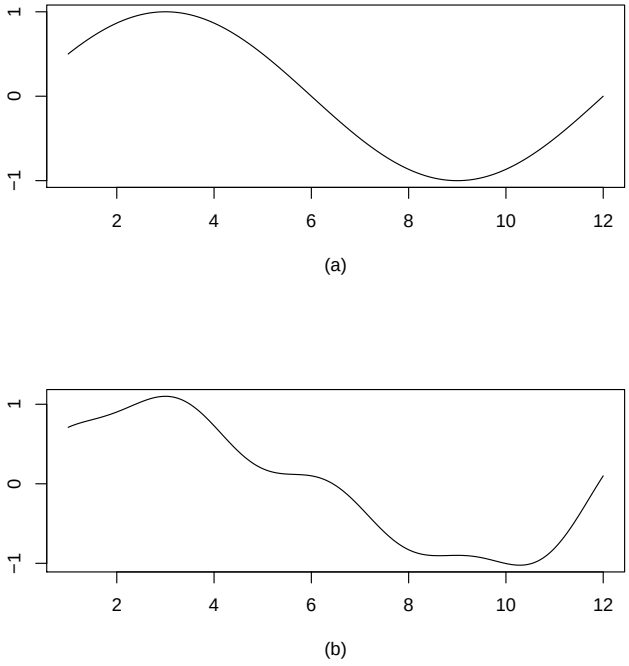


Fig. 3.5. Two possible underlying seasonal patterns for monthly series based on the harmonic model (Equation 3.10). Plot (a) above is the first harmonic evaluated over a year, which would usually be too regular for the seasonal variation in most practical applications. Plot (b) shows the same wave with a further two harmonics added, and illustrates just one of many ways in which an underlying sine wave can be perturbed to produce a less regular, but still dominant, seasonal pattern of period 12 months.

3.3.1 Simulation

It is straightforward to simulate a time series based on the harmonic model in Equation (3.10). For example, suppose the underlying model is:

$$x_t = 0.1 + 0.005t + 0.001t^2 + \sin(2\pi t/12) + \\ 0.2\sin(4\pi t/12) + 0.1\sin(8\pi t/12) + 0.1\cos(8\pi t/12) + z_t \tag{3.11}$$

where $\{z_t\}$ is Gaussian white noise with standard deviation 0.5. This model has the same seasonal harmonic components as the model represented in Figure (3.5b) but also contains an underlying quadratic trend. In the code below a monthly series of length 10 years is simulated and plotted (Figure 3.6).

```
> set.seed(1)
> t = 1:(10*12)
> z = rnorm(10*12, sd=0.5)
> trend    = 0.1 + 0.005*t + 0.001*t^2
> seasonal = sin(2*pi*t/12) + 0.2*sin(2*pi*2*t/12) +
              0.1*sin(2*pi*4*t/12) + 0.1*cos(2*pi*4*t/12)
> x = trend + seasonal + z
> plot(x, type="l")
```

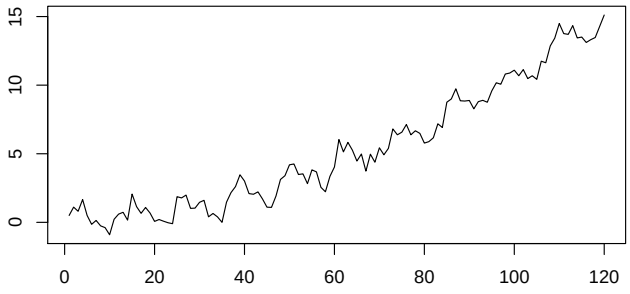


Fig. 3.6. Ten years of simulated data for the model in Equation (3.11).

3.3.2 Fitted models

Fit to simulated series

One of the most straightforward ways to fit a harmonic time series model is to place the harmonic variables in a matrix. In this section we illustrate the procedure using the simulated series from the previous section:

```
> s = c = matrix(nr=length(t), nc=6)
```

```
> for (i in 1:6)
  { c[,i] = cos(2*pi*i*t/12)
    s[,i] = sin(2*pi*i*t/12) }

> s = s[,-6]
```

In the last expression above, the 6-th harmonic term is removed for the sine function as this is always zero. In most cases the order of the harmonics and polynomial trend will be unknown. However, the harmonic coefficients are known to be independent which means that any harmonic coefficients that are not statistically 'significant' can be dropped. It is largely a subjective decision on the part of the statistician to decide what constitutes a 'significant' value – a t-ratio of magnitude 2 is a common choice, and corresponds to an approximate 5% significance level. This information, along with the estimated parameters, can be extracted using the `summary` function. In the code below, a harmonic seasonal model with quadratic trend is fitted to the simulated data – the 'inhibit' function `I` is used to ensure t^2 is treated as an explanatory variable:

```
> x.lm1 = lm(x ~ t + I(t^2) + s + c)

> summary(x.lm1)$coef
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.155756   1.25e-01   1.243 2.16e-01
t            0.005410   4.77e-03   1.134 2.60e-01
I(t^2)       0.000995   3.82e-05  26.038 7.29e-48
s1           0.901722   5.81e-02  15.508 5.04e-29
s2           0.199566   5.80e-02   3.440 8.33e-04
s3          -0.042624   5.80e-02  -0.735 4.64e-01
s4           0.062144   5.80e-02   1.072 2.86e-01
s5           0.046820   5.80e-02   0.808 4.21e-01
c1           0.019021   5.80e-02   0.328 7.44e-01
c2          -0.028115   5.80e-02  -0.485 6.29e-01
c3           0.012703   5.80e-02   0.219 8.27e-01
c4           0.014766   5.80e-02   0.255 7.99e-01
c5          -0.064209   5.80e-02  -1.107 2.71e-01
c6          -0.023818   4.10e-02  -0.581 5.63e-01
```

In the above output, there are 3 'significant' coefficients (at the 5% level), which are used in the model below:

```
> x.lm2 = lm(x ~ I(t^2) + s[,1] + s[,2])

> confint(x.lm2)
               2.5 %  97.5 %
(Intercept) 0.16040 0.40041
I(t^2)      0.00102 0.00105
```

```
s[, 1]       0.78726 1.01315
s[, 2]       0.08616 0.31157
```

The last command gives a confidence interval for each model parameter. As no zero's are contained in the intervals, this confirms that all the coefficients in the fitted model are significant. The estimated coefficients of the best fitting model are given by:

```
> coef(x.lm2)
(Intercept)      I(t^2)      s[, 1]      s[, 2]
    0.28040     0.00104     0.90021     0.19886
```

The above coefficients give the following model equation for predictions at time t:

$$\hat{x}_t = 0.280 + 0.00104t^2 + 0.900\sin(2\pi t/12) + 0.199\sin(4\pi t/12) \qquad (3.12)$$

AIC is an appropriate statistic for comparing models. For the two fitted models above we have:

```
> AIC(x.lm1); AIC(x.lm2)
[1] 164
[1] 150
```

As expected the last model has the smallest AIC and therefore provides the best fit to the data. Due to sampling variation, the best fitting model is not the 'true' underlying model used to simulate the data, which is easily verified by calculating the AIC of the known underlying model (3.11):

```
> AIC(lm(x ~ t + I(t^2) + s[,1] + s[,2] + s[,4] + c[,4]))
[1] 153
```

The function `step` can be used to automate the selection of the best fitting model by AIC, as illustrated in the next example.

Harmonic model fitted to the temperature series: 1970–2005

In the code below, a harmonic model with quadratic trend is fitted to the temperature series (1970–2005) from Section 3.1.4. The units for the 'time' variable are 'years', so the divisor of 12 is not needed when creating the harmonic variables. To reduce computational error in the OLS procedure (due to large numbers occurring in the sum of squares), the 'time' variable is standardized by subtracting the mean and then dividing by the standard deviation:

```
> s = c = matrix(nr=length(temp), nc=6)
> for (i in 1:6)
  { c[,i] = cos(2*pi*i*time(temp))
    s[,i] = sin(2*pi*i*time(temp)) }
> s = s[,-6]
```

```
> t = (time(temp) - mean(time(temp)))/sd(time(temp))
> mean(time(temp)); sd(time(temp))
[1] 1988
[1] 10.4

> temp.lm1 = lm(temp ~  t + I(t^2) + s[,1] + c[,1]
                   + s[,2] + c[,2]  + s[,3] + c[,3]
                   + s[,4] + c[,4]  + s[,5] + c[,5] + c[,6])
```

The `step` function tends to produce a large amount of output. You can allow the output to appear on your screen or redirect it using `capture.output` as shown below: [2]

```
> output = capture.output(temp.step <- step(temp.lm1))

> summary(temp.step)$coef
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)   0.1750    0.00604   28.98 5.48e-103
t             0.1841    0.00605   30.44 3.78e-109
s[, 1]        0.0204    0.00854    2.39  1.73e-02
s[, 2]        0.0162    0.00854    1.89  5.92e-02
```

Three coefficients appear in the final model, with the sine component of the second harmonic being only possibly significant. AIC can be used to test whether it is better to leave the second harmonic out of the equation and can also be used to compare the best fitting harmonic model with the fitted seasonal indicator model of the previous section (`temp.lm`):

```
> temp.lm2 = lm(temp ~ t + s[,1] + s[,2])

> confint(temp.lm2)
                2.5 % 97.5 %
(Intercept)  0.163143 0.1869
t            0.172210 0.1960
s[, 1]       0.003629 0.0372
s[, 2]      -0.000632 0.0329

> temp.lm3 = lm(temp ~ t + s[,1])

> AIC(temp.lm); AIC(temp.lm1); AIC(temp.lm2); AIC(temp.lm3)
[1] -547
[1] -547
[1] -561
[1] -560
```

[2] Within functions: use `<-` to make assignments and `=` for parameter values.

Based on the above output, and in agreement with the results from the `step` function, the best fitting model is `temp.lm2`.

To check the adequacy of the fitted model it is appropriate to create a time plot and correlogram of the residuals, because the residuals form a time series (Figure 3.7). The time plot is used to detect patterns in the series. For example, if a higher ordered polynomial is required, this would show up as a curve in the time plot. The correlograms should reveal any significant autocorrelation in the residuals, which would then need to be accounted for.

```
> plot(time(temp), resid(temp.lm2), type="l"); abline(0,0)
> acf (resid(temp.lm2)); pacf(resid(temp.lm2))
```

In Figure 3.7(a) there is no discernible curve in the series, which implies that a straight line is an adequate description of the trend over the period (1970–2005). The tendency for the series to persist above or below the x-axis in Figure 3.7(a) implies that the residuals are positively autocorrelated. This is verified in the correlogram of the residuals, which shows a clear positive autocorrelation at lags 1–10 (Figure 3.7b).

The correlogram in Figure 3.7(b) is similar to that expected of an AR(p) process (Section 2.3.4). The partial-correlogram has significant partial autocorrelations at lags 1 and 2, which implies that an AR(2) model should provide a good fit to the residual series (Figure 3.7c).

In the code below, we use the `ar` function to select the best fitting AR(p) model, which turns out to be an AR(2) model in agreement with the partial-correlogram (Figure 3.7c). The first 2 terms are missing from the residual series and so are removed using `-(1:2)` in the vector of residuals.

```
> res.ar = ar(resid(temp.lm2), method="mle")
> res.ar$ar
[1] 0.494 0.307
> sd(res.ar$res[-(1:2)])
[1] 0.0837
> acf(res.ar$res[-(1:2)])
```

The correlogram of the residuals of the fitted AR(2) model is given in Figure 3.8, from which it is clear that the residuals of the fitted AR(2) model are approximately white noise. Thus, the final form of the fitted model (Equation 3.13 below) provides a good description of the temperature series over the period 1970–2005:

$$x_t = 0.175 + \frac{0.184(t - 1988)}{10.4} + 0.0204\sin(2\pi t) + 0.0162\sin(4\pi t) + z_t \quad (3.13)$$

where t is 'time' (measured in units of 'years'), and the residual series $\{z_t\}$ follows an AR(2) process given by:

$$z_t = 0.494z_{t-1} + 0.307z_{t-2} + \epsilon_t \quad (3.14)$$

where $\{\epsilon_t\}$ is white noise with mean zero and standard deviation 0.0837.

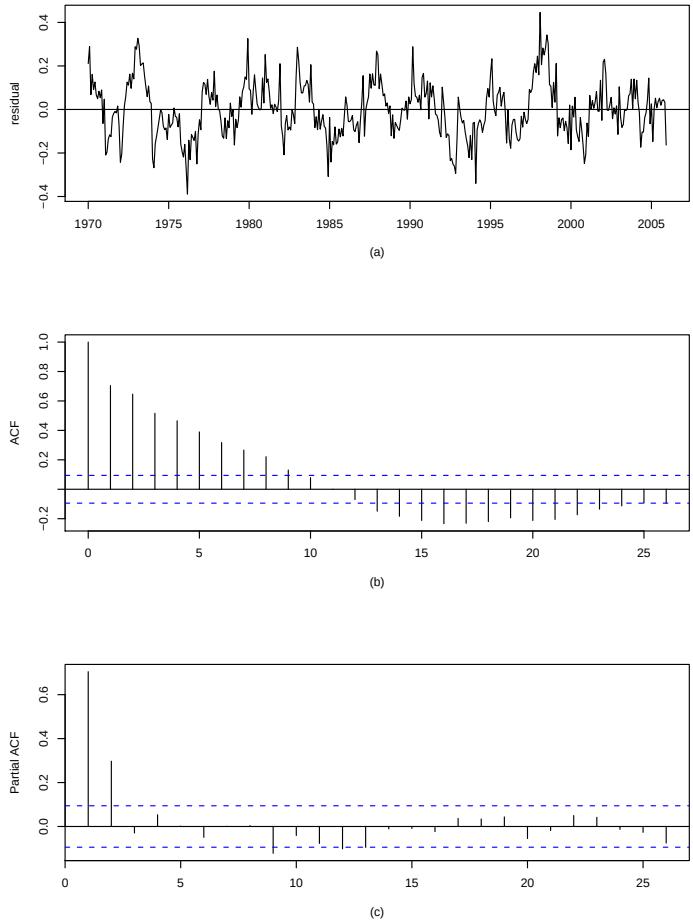


Fig. 3.7. Residual diagnostic plots for the harmonic model fitted to the temperature series (1970–2005). (a) The residuals plotted against time (in years). (b) The correlogram of the residuals (time units are months). (c) The partial-correlogram.

3.4 Logarithm transformations

3.4.1 Introduction

Recall from Sections 1.6 and 3.1 that the natural logarithm (base e) can be used to transform a model with multiplicative components to a model with additive components. For example, if $\{x_t\}$ is a time series given by:

$$x_t = m'_t \; s'_t \; z'_t \tag{3.15}$$

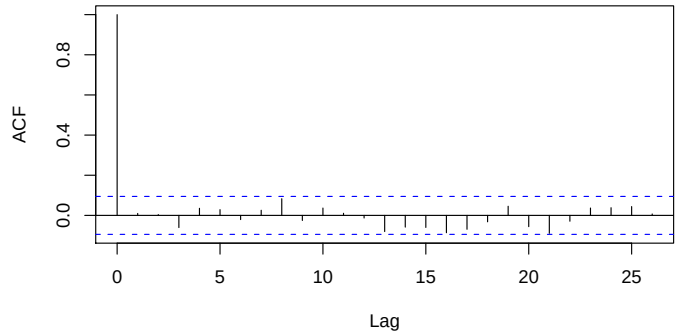


Fig. 3.8. Correlogram of the residuals of the AR(2) model fitted to the residuals of the harmonic model for the temperature series.

where m'_t is the trend, s'_t is the seasonal effect, and z'_t is the residual error, then the series $\{y_t\}$, given by

$$y_t = \log x_t = \log m'_t + \log s'_t + \log z'_t = m_t + s_t + z_t \tag{3.16}$$

has additive components. Thus, if m_t and s_t are linear functions, the parameters in (3.16) can be estimated by OLS.

In Equation (3.16), logs can be taken only if all values in the series $\{x_t\}$ are positive, i.e. $x_t > 0$ for all t. Conversely, a log-transformation can be seen as an appropriate model formulation if the series can only take positive values, because the anti-log function ensures predicted and simulated values for $\{x_t\}$ are positive.

3.4.2 An example using the air passenger series

Consider the air passenger series from Section 1.4.1. Time plots of the original series and the natural logarithm of the series can be obtained as follows and are shown in Figure 3.9.

```
> data(AirPassengers); AP = AirPassengers
> plot(AP); plot(log(AP))
```

A log-transformation is appropriate for the airline series for two main reasons. First, in Figure 3.9(a) the variance can be seen increasing as t increases, whilst after the logarithm is taken the variance is approximately constant over the period of the record (Figure 3.9b). Second, the number of people using the airline is always a positive value (Figure 3.9a); a log-transformation would ensure any predictions or simulations are also positive.

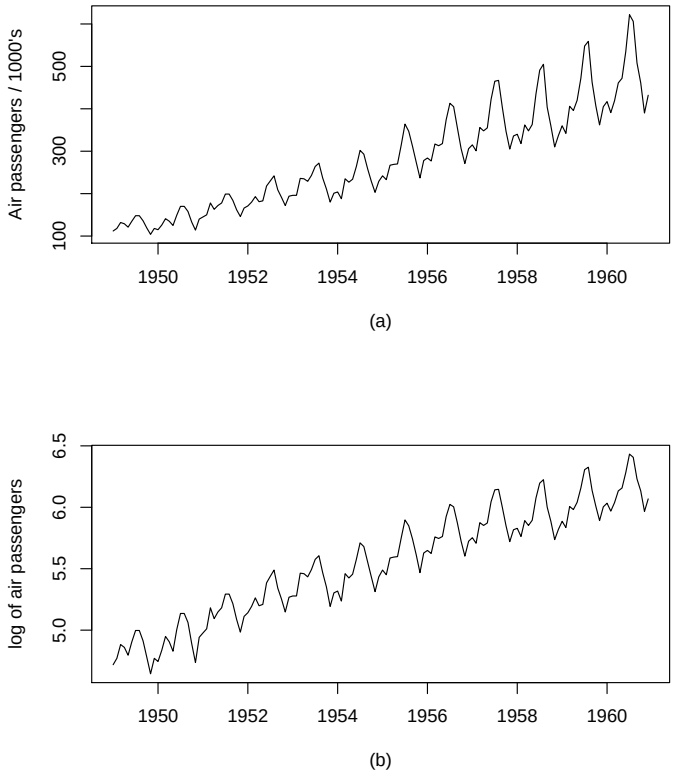


Fig. 3.9. Time plots of: (a) the airline series (1949–1960) and (b) the natural logarithm of the airline series.

In the code below, 'stepwise' regression is used to fit a harmonic model with a polynomial trend to the logarithm of the air passenger series. (As before, the 'time' variable is standardised to reduce computational errors due to large numbers appearing in the sum of squares.)

```
> s = c = matrix(nr=length(AP), nc=6)

> for (i in 1:6)
  {  s[,i] = sin(2*pi*i*time(AP))
     c[,i] = cos(2*pi*i*time(AP)) }
> s = s[,-6]

> t = (time(AP) - mean(time(AP)))/sd(time(AP))
```

```
> mean(time(AP)); sd(time(AP))
[1] 1955
[1] 3.48

> AP.lm = lm(log(AP) ~ t + I(t^2)
              + s[,1] + c[,1] + s[,2] + c[,2] + s[,3] + c[,3]
              + s[,4] + c[,4] + s[,5] + c[,5] + c[,6])

> output = capture.output(AP.step <- step(AP.lm))

> summary(AP.step)$coef
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)  5.57929    0.00602  927.32 1.32e-253
t            0.41997    0.00404  104.01 1.59e-128
I(t^2)      -0.03738    0.00452   -8.28  1.22e-13
s[, 1]       0.02807    0.00568    4.94  2.33e-06
c[, 1]      -0.14719    0.00567  -25.94  1.12e-53
s[, 2]       0.05906    0.00567   10.41  6.87e-19
c[, 2]       0.05680    0.00567   10.01  6.70e-18
s[, 3]      -0.02731    0.00567   -4.81  3.97e-06
c[, 3]      -0.00871    0.00567   -1.54  1.27e-01
s[, 4]      -0.03200    0.00567   -5.64  9.85e-08
c[, 4]       0.01111    0.00567    1.96  5.22e-02
s[, 5]      -0.02127    0.00567   -3.75  2.64e-04

> acf(resid(AP.step))
```

The correlogram shows that the residuals are positively autocorrelated (Figure 3.10), whilst the partial-correlogram has one clear significant value at lag 1 implying that the residual series is approximately AR(1). Thus, in the code below, an AR(1) model is fitted to the residual series, and the residuals of the fitted AR(1) model examined for any remaining autocorrelation:

```
> AP.ar = ar(resid(AP.step), order=1, method="mle")
> acf(AP.ar$res[-1])
```

The correlogram of the residuals of the fitted AR(1) model is approximately white noise (Figure 3.11). However, there is slight evidence that the seasonal variation may not have been fully accounted for, because a significant value remains at lag 12 which corresponds to the seasonal period in the data (Figure 3.11). The harmonic seasonal terms used in the fitted model are essentially deterministic components, because they are functions of time. However, seasonal effects can be stochastic, rather than deterministic, in a similar way to trends. We will examine this further in Chapter **??**, where models with stochastic trends and stochastic seasonal terms will be investigated in more detail.

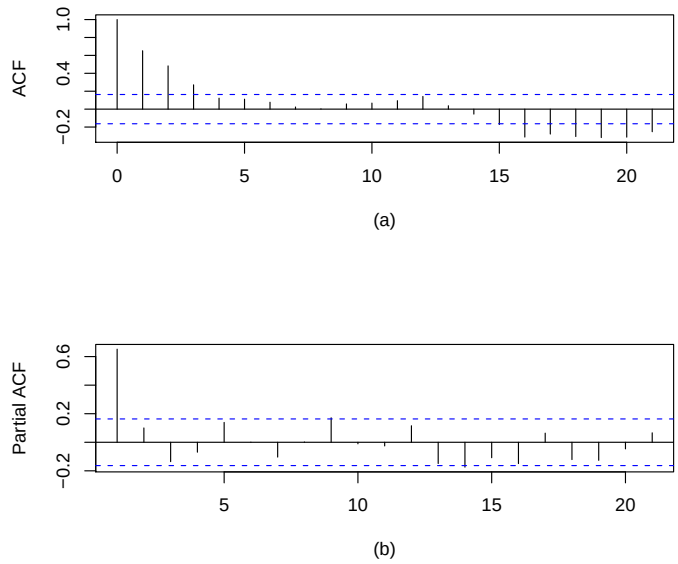


Fig. 3.10. (a) The correlogram and (b) the partial-correlogram of the residual series of the regression model fitted to the log-transformed airline series.

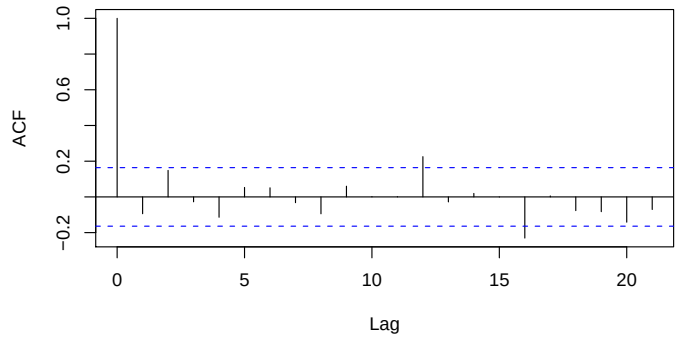


Fig. 3.11. Correlogram of the residuals from the fitted AR(1) model. The AR(1) model was fitted to the residuals from the regression model of the log-transformed airline series.

3.5 Non-linear models

3.5.1 Introduction

For the reasons given in Section 3.1, linear models are applicable to a wide range of time series. However, for some time series it may be more appropriate to fit a non-linear model directly rather than take logs or use a linear polynomial approximation. For example, if a series is known to derive from a known non-linear process, perhaps based on an underlying known deterministic law in science, then it would be better to use this information in the model formulation, and fit a non-linear model directly to the data. In `R`, a non-linear model can be fitted by least squares using the function `nls`.

In the last section, we found that using the natural logarithm of a series could help stabilise the variance. However, using logs can present difficulties when a series contains negative values, because the log of a negative value is undefined. One way round this problem is to add a constant to all the terms in the series, so if $\{x_t\}$ is a series containing (some) negative values, then adding c_0 such that $c_0 > \max\{-x_t\}$ and then taking logs produces a transformed series $\{\log(c_0 + x_t)\}$ that is defined for all t. A linear model, e.g. a straight line trend, could then be fitted to produce the following model for $\{x_t\}$:

$$x_t = -c_0 + e^{\alpha_0 + \alpha_1 t + z_t} \tag{3.17}$$

where α_0 and α_1 are model parameters, and $\{z_t\}$ is a residual series, which may be autocorrelated.

The main difficulty with the approach leading to Equation (3.17) is that c_0 should really be estimated like any other parameter in the model, whilst in practice a user will often arbitrarily choose a value that satisfies the constraint ($c_0 > \max\{-x_t\}$). If there is a reason to expect a model similar to that in (3.17), but there is no evidence for multiplicative residual terms, then the constant c_0 should be estimated with the other model parameters using non-linear least squares, i.e. the following model should be fitted:

$$x_t = -c_0 + e^{\alpha_0 + \alpha_1 t} + z_t \tag{3.18}$$

3.5.2 Example of a simulated and fitted non-linear series

As non-linear models are generally fitted when the underlying non-linear function is known, we will simulate a non-linear series based on Equation (3.18) with $c_0 = 0$, and compare parameters estimated using `nls` with those of the known underlying function.

Below, a non-linear series with AR(1) residuals is simulated and plotted (Figure 3.12):

```
> set.seed(1); e = rnorm(100, sd=10); z = rep(0,100)
> for (t in 2:100) z[t] = 0.7*z[t-1] + e[t]
```

```
> t = 1:100;
> f = function(x) exp(1 + 0.05*t)

> x = f(t) + z
> plot(x); abline(0,0)
```

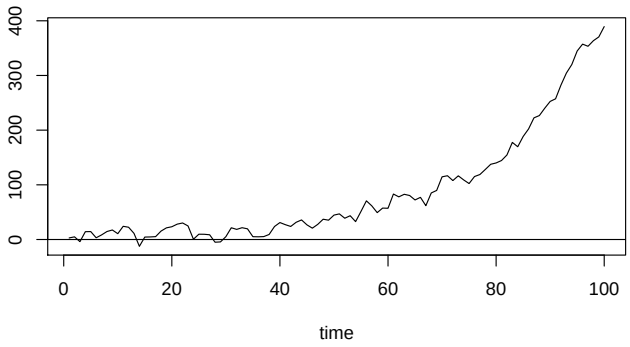


Fig. 3.12. Plot of a non-linear series containing negative values.

The series plotted in Figure 3.12, has an apparent increasing exponential trend but also contains negative values so that a direct log-transformation cannot be used, and a non-linear model is needed. In R, a non-linear model is fitted by specifying a formula with the parameters and their starting values contained in a `list`:

```
> x.nls = nls(x ~ exp(alp0 + alp1 * t),
                    start = list (alp0 = 0.1, alp1 = 0.5))

> summary(x.nls)$parameters
     Estimate Std. Error t value Pr(>|t|)
alp0   1.1764   0.074295    15.8 9.20e-29
alp1   0.0483   0.000819    59.0 2.35e-78
```

The estimates for α_0 and α_1 are close to the underlying values that were used to simulate the data, although the standard errors of these estimates are likely to be under-estimated because of the autocorrelation in the residuals. [3]

[3] The generalised least squares function `gnls` can be used to fit non-linear models with autocorrelated residuals. However, in practice, computational difficulties often arise when using this function.

3.6 Forecasting

3.6.1 Introduction

A forecast is a prediction into the future. In the context of time series regression, a forecast involves extrapolating a fitted model into the future by evaluating the model function for a new series of times. The main problem with this approach is that the trends present in the fitted series may change in the future. It is better, therefore, to think of a forecast from a regression model as an expected value conditional on past trends continuing into the future.

3.6.2 Prediction in R

The generic function for making predictions in `R` is `predict`. The function essentially takes a fitted model and new data as parameters. The key to using this function with a regression model is to ensure that the new data is properly defined and labelled in a `data.frame`.

In the code below, we use the fitted 'stepwise' regression model of Section 3.4.2 to forecast the number of air passengers travelling for the 10-year period that follows the record (Figure 3.13). The forecast is given by applying the exponential function (anti-log) to `predict` because the regression model was fitted to the logarithm of the series:

```
> new.t = time(ts(start=1961, end=c(1970,12), fr=12))
> t = (new.t - mean(time(AP)))/sd(time(AP))

> s = c = matrix(nr=length(new.t), nc=6)
> for (i in 1:6)
  { c[,i] = cos(2*pi*i*new.t)
    s[,i] = sin(2*pi*i*new.t) }
> s = s[,-6]

> new.dat = data.frame(t=as.vector(t), s=s, c=c)
> AP.pred.ts = exp(ts(predict(AP.step, new.dat), st=1961, fr=12))

> ts.plot(log(AP), log(AP.pred.ts), lty=1:2)
> ts.plot(AP, AP.pred.ts, lty=1:2)
```

3.6.3 Inverse transform and bias correction

Log-Normal residual errors

The predictions in Figure 3.13(b) were obtained by applying the anti-log to the predicted values obtained from the log-regression model. However, the process

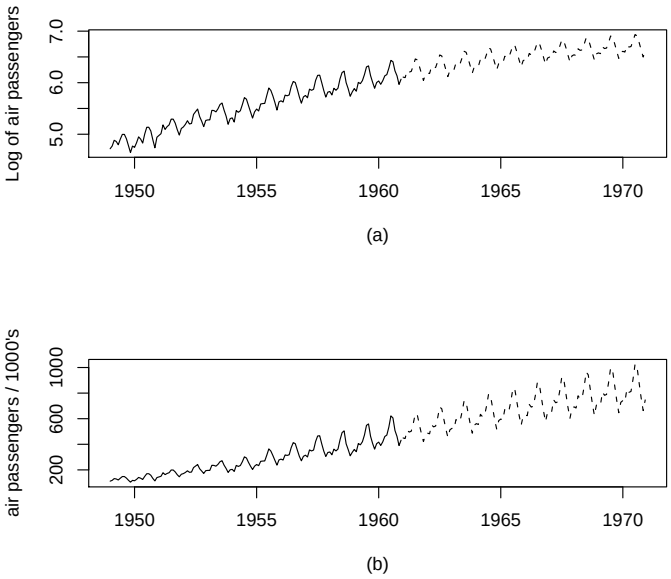


Fig. 3.13. Air Passengers (1949–1960; solid line) and forecasts (1961–1970; dotted lines): (a) Logarithm and predicted values; (b) Original series and anti-log of the predicted values.

of using a transformation, such as the logarithm, and then applying an inverse transformation introduces a bias in the predictions. If the regression model closely fits the data this bias will be small (as shown in the next example for the airline predictions).

The bias arises as a result of applying the inverse transform to a residual series. For example, if the residuals are Gaussian white noise $\{\epsilon_t\}$, with mean zero and standard deviation σ, then the distribution of the inverse-transform (the anti-log) of the residuals is log-Normal with mean $e^{\frac{1}{2}\sigma^2}$. This can be verified theoretically, or empirically by simulation as in the code below:

```
> set.seed(1); sigma = 3; z = rnorm(10000, sd=sigma)
> mean(z); mean(exp(z))
[1] -0.0196
[1] 73
> exp(sigma^2/2)
[1] 90
> t.test(exp(z), mu=exp(sigma^2/2))$p.value
[1] 0.185
```

The t-test above, shows there is no significant difference between the mean of the anti-log of the residuals and the expected mean from a log-normal distribution. Hence, for a Gaussian white noise residual series a correction factor of $e^{\frac{1}{2}\sigma^2}$ should be applied to the predictions. The importance of this correction factor really depends on the value of σ^2. If σ^2 is very small, the correction factor will hardly change the predictions at all, and so could be neglected without major concern, especially as errors from other sources are likely to be significantly greater.

Empirical correction factor

The $e^{\frac{1}{2}\sigma^2}$ correction factor can be used when the residual series of the fitted log-regression model is Gaussian white noise. In general, however, the distribution of the residuals is often 'skewed', in which case a correction factor can be determined empirically using the mean of the anti-log of the residual series. In this approach, adjusted predicted values $\{\hat{x}'_t\}$ can be obtained from:

$$\hat{x}'_t = e^{\hat{\log} x_t} \sum_{t=1}^{n} e^{z_t}/n \tag{3.19}$$

where $\{\hat{\log} x_t : t = 1, \ldots, n\}$ is the predicted series given by the fitted log-regression model, and $\{z_t\}$ is the residual series from this fitted model.

The following example illustrates the procedure for calculating the correction factors.

3.6.4 Example using the air passenger data

For the airline series the forecasts can be adjusted by multiplying the predictions by $e^{\frac{1}{2}\sigma^2}$, where σ is the standard deviation of the residuals, or using an empirical correction factor as follows:

```
> summary(AP.lm2)$r.sq
[1] 0.989
> sigma = summary(AP.lm2)$sigma

> lognorm.correction.factor = exp((1/2)*sigma^2)
> empirical.correction.factor = mean(exp(resid(AP.lm2)))

> lognorm.correction.factor
[1] 1.001171
> empirical.correction.factor
[1] 1.001080

> AP.pred.ts = AP.pred.ts * empirical.correction.factor
```

The adjusted forecasts in `AP.pred.ts` above allow for the bias in taking the anti-log of the predictions. However, the small σ (and $R^2 = 0.99$) results in a small correction factor (of the order 0.1%), which is probably negligible compared to other sources of errors that exist in the forecasts. Whilst in this example the correction factor is small, there is no reason why it will be small in general. Consequently, in practice when using log-regression, the correction factor should always be calculated and applied to the forecasts.

3.7 Summary of R commands

`lm`	fits a linear (regression) model
`coef`	extracts the parameter estimates from a fitted model
`confint`	returns a (95%) confidence interval for the parameters of a fitted model
`summary`	gets basic summary information
`gls`	fits a linear model using generalised least squares (allowing for autocorrelated residuals)
`factor`	returns variables in the form of 'factors' or indicator variables
`predict`	returns predictions (or forecasts) from a fitted model
`nls`	fits a non-linear model by least squares

3.8 Exercises

1. a) Produce a time plot for $\{x_t : t = 1, \ldots, 100\}$ where $x_t = 70 + 2t - 3t^2 + z_t$, $\{z_t\}$ is the AR(1) process $z_t = 0.5z_{t-1} + \epsilon_t$, and $\{\epsilon_t\}$ is white noise with standard deviation 25.
   b) Fit a quadratic trend to the series $\{x_t\}$. Give the coefficients of the fitted model.
   c) Find a 95% confidence interval for the parameters of the quadratic model and comment.
   d) Plot the correlogram of the residuals and comment.
   e) Re-fit the model using GLS. Give the standard errors of the parameters estimates, and comment.

2. The standard errors of the parameter estimates of a fitted regression model are likely to be under-estimated if there is positive serial correlation in the data. This implies that explanatory variables may appear as 'significant' when they should not. Use GLS to check the significance of the variables of the fitted model from Section 3.4.2. Use an appropriate estimate of the lag 1 autocorrelation within `gls`.

3. This question is based on the electricity production series (1958–90).
   a) Give two reasons why a log-transformation may be appropriate for the electricity series.
   b) Fit a seasonal indicator model with a quadratic trend to the (natural) logarithm of the series. Use stepwise regression to select the best model based on AIC.
   c) Fit a harmonic model with a quadratic trend to the logarithm of the series. Use stepwise regression to select the best model based on AIC.
   d) Plot the correlogram and partial-correlogram of the residuals from the best fitting model and comment on the plots.
   e) Fit an AR model to the residuals of the best fitting model. Give the order of the best fitting AR model and the estimated model parameters.
   f) Plot the correlogram of the residuals of the AR model and comment.
   g) Write down in full the equation of the best fitting model.
   h) Use the best fitting model to forecast electricity production for the years 1991–2000, making sure you have corrected for any bias due to taking logs.

4. Suppose a sample of size n follows an AR(1) process with lag 1 autocorrelation: $\rho_1 = \alpha$. Use Equation (3.5) to find the variance of the sample mean.

References

Akaike, H. (1974). A new look at statistical model indentification. *IEEE Transactions on Automatic Control*, 19:716–722.

Becker, R., Chambers, J., and Wilks, A. (1988). *The NEW S Language*. New York: Chapman & Hall.

Bowerman, B., O'Connell, R., and Koehler, A. (2005). *Forecasting, Time Series, and Regression*. Belmont: Thomson Brooks/Cole, 4 edition.

Box, G. and Jenkins, G. (1970). *Time Series Analysis, Forecasting and control*. San Francisco: Holden-Day.

Brockwell, P. and Davis, R. (1996). *Introduction to Time Series and Forecasting*. New York: Springer-Verlag.

Chatfield, C. (1989). *The Analysis of Time Series: An Introduction*. London: Chapman & Hall, 4 edition.

Dalgaard, P. (2002). *Introductory Statistics with R*. New York: Springer-Verlag.

Diggle, P. (1990). *Time Series: A Biostatistical Introduction*. Oxford: Oxford University Press.

Enders, W. (1995). *Applied Econometric Time Series*. New York: John Wiley & Sons.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Graphical and Computational Statistics*, 5:299–314.

Jones, P. and Moberg, A. (2003). Hemispheric and large-scale surface air temperature variations: An extensive revision and an update to 2001. *Journal of Climate*, 16:206–223.

Kendall, M. and Ord, J. (1990). *Time Series*. London: Edward Arnold, 3 edition.

Rayner, N., Parker, D., Horton, E., Folland, C., Alexander, L., Rowell, D., Kent, E., A., and Kaplan (2003). Globally complete analyses of sea surface temperature, sea ice and night marine air temperature, 1871-2000. *Journal of Geophysical Research*, 108:4407.

Shumway, R. and Stoffer, D. (2000). *Time Series Analysis and Its Applications*. New York: Springer-Verlag.