24

Time Series Analysis

Time series data are vectors of numbers, typically regularly spaced in time. Yearly counts of animals, daily prices of shares, monthly means of temperature, and minute-by-minute details of blood pressure are all examples of time series, but they are measured on different time scales. Sometimes the interest is in the time series itself (e.g. whether or not it is cyclic, or how well the data fit a particular theoretical model), and sometimes the time series is incidental to a designed experiment (e.g. repeated measures). We cover each of these cases in turn.

The three key concepts in time series analysis are

- trend,
- serial dependence, and
- stationarity.

Most time series analyses assume that the data are untrended. If they do show a consistent upward or downward trend, then they can be detrended before analysis (e.g. by differencing). Serial dependence arises because the values of adjacent members of a time series may well be correlated. Stationarity is a technical concept, but it can be thought of simply as meaning that the time series has the same properties wherever you start looking at it (e.g. white noise is a sequence of mutually independent random variables each with mean zero and variance $\sigma^2 > 0$).

24.1 Nicholson's blowflies

The Australian ecologist, A.J. Nicholson, reared blowfly larvae on pieces of liver in laboratory cultures that his technicians kept running continuously for almost 7 years (361 weeks, to be exact). The time series for numbers of adult flies looks like this:

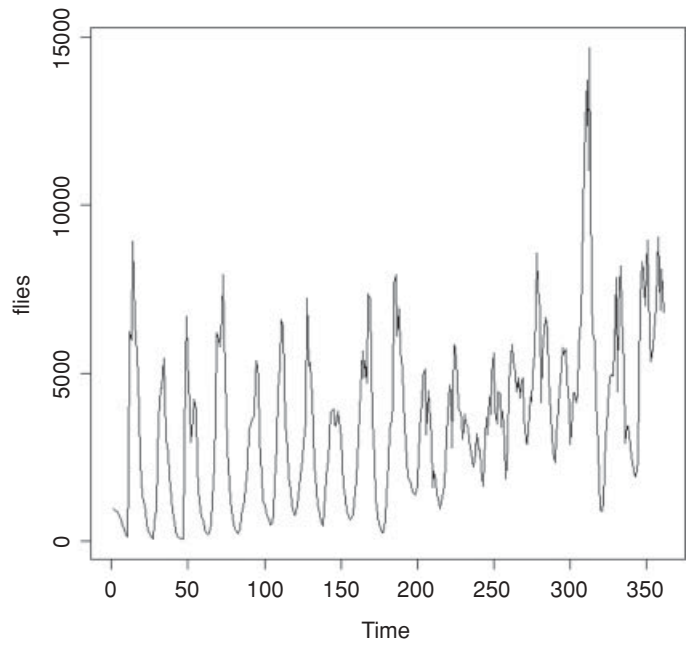
```
blowfly <- read.table("c:\\temp\\blowfly.txt",header=T)
attach(blowfly)
names(blowfly)
```

```
[1] "flies"
```

The R Book, Second Edition. Michael J. Crawley.

First, make the flies variable into a time series object and plot it:

```
flies <- ts(flies)
plot(flies)
```



This classic time series has two clear features:

- For the first 200 weeks the system exhibits beautifully regular cycles.
- After week 200 things change (perhaps a genetic mutation had arisen); the cycles become much less clear-cut, and the population begins a pronounced upward trend.

There are two important ideas to understand in time series analysis: **autocorrelation** and **partial autocorrelation**. The first describes how this week's population is related to last week's population. This is the autocorrelation at lag 1. The second describes the relationship between this week's population and the population at lag t once we have controlled for the correlations between all of the successive weeks between this week and week t. This should become clear if we draw the scatterplots from which the first four autocorrelation terms are calculated (lag 1 to lag 4).

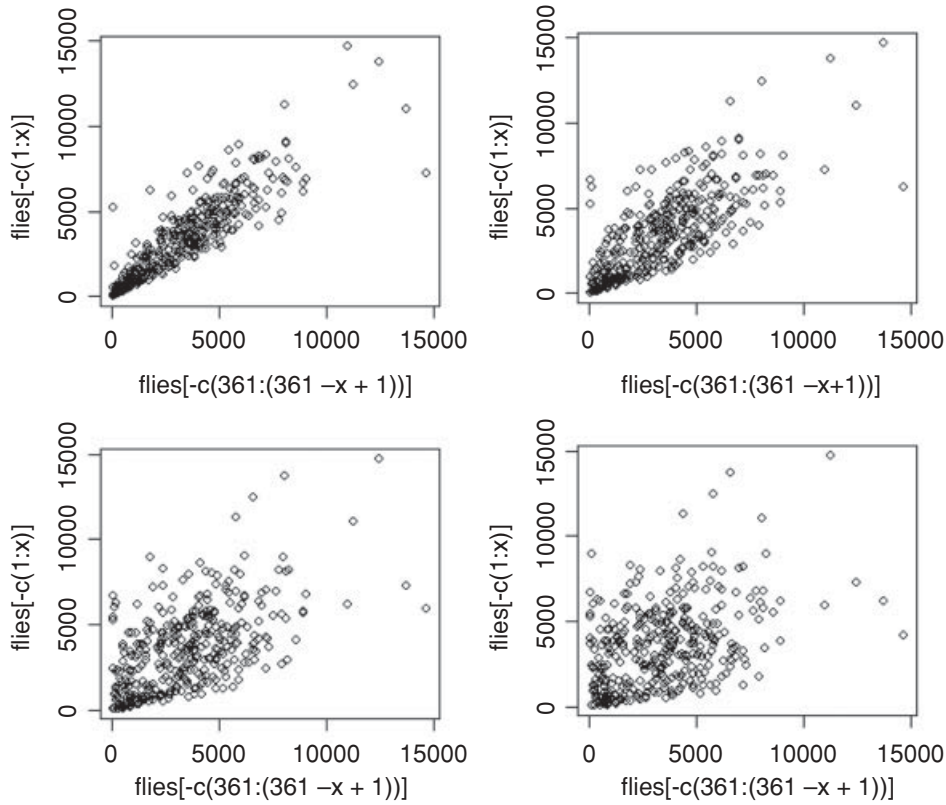
There is a snag, however. The vector of flies at lag 1 is shorter (by one) than the original vector because the first element of the lagged vector is the second element of flies. The coordinates of the first data point to be drawn on the scatterplot are (`flies[1],flies[2]`) and the coordinates of the last plot that can be drawn are (`flies[360], flies[361]`) because the original vector is 361 element long:

```
length(flies)
```

```
[1] 361
```

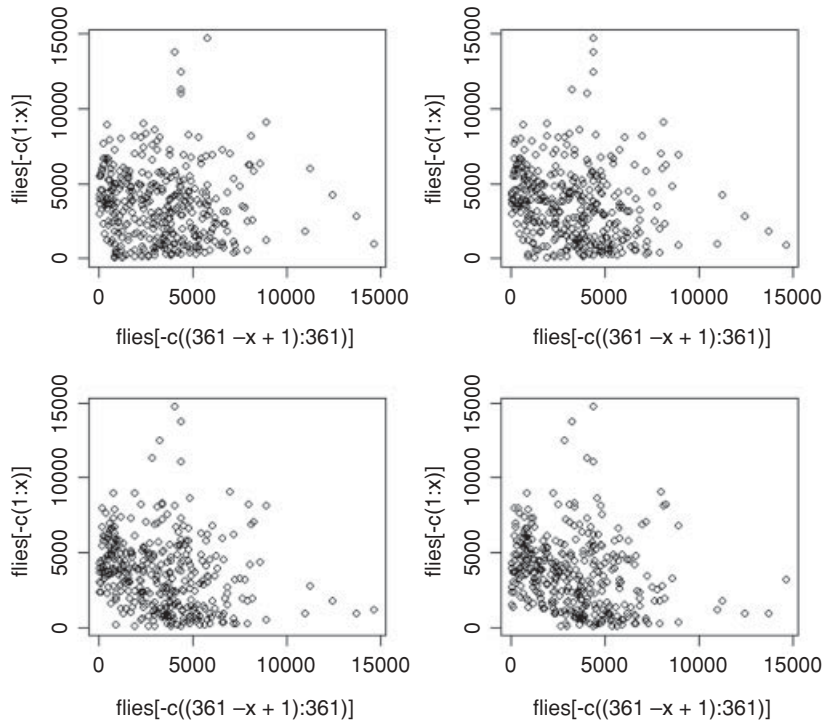
Thus, the lengths of the vectors that can be plotted go down by one for every increase in the lag of one. We can produce the four plots for lags 1 to 4 in a function like this:

```
par(mfrow=c(2,2))
sapply(1:4, function(x) plot(flies[-c(361: (361-x+1))], flies[-c(1:x)] ) )
```



The correlation is very strong at lag 1, but notice how the variance increases with population size: small populations this week are invariably correlated with small populations next week, but large populations this week may be associated with large or small populations next week. The striking pattern here is the way that the correlation fades away as the size of the lag increases. Because the population is cyclic, the correlation goes to zero, then becomes weakly negative and then becomes strongly negative. This occurs at lags that are half the cycle length. Looking back at the time series, the cycles look to be about 20 weeks in length. So let us repeat the exercise by producing scatterplots at lags of 7, 8, 9 and 10 weeks:

```
sapply(7:10, function(x) plot(flies[-c((361-x+1):361)], flies[-c(1:x)] ) )
par(mfrow=c(1,1))
```



The negative correlation at lag 10 gradually emerges from the fog of no correlation at lag 7.

More formally, the autocorrelation function $\rho(k)$ at lag k is

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)},$$

where $\gamma(k)$ is the autocovariance function at lag k of a stationary random function $\{Y(t)\}$ given by

$$\gamma(k) = \text{cov}\{Y(t), Y(t-k)\}.$$

The most important properties of the autocorrelation coefficient are as follows:

- They are symmetric backwards and forwards, so $\rho(k) = \rho(-k)$.
- The limits are $-1 \leq \rho(k) \leq 1$.
- When $Y(t)$ and $Y(t-k)$ are independent, then $\rho(k) = 0$.
- The converse of this is not true, so that $\rho(k) = 0$ does not imply that Y(t) and Y($t-k$) are independent (look at the scatterplot for $k = 7$ in the scatterplots above).

A first-order autoregressive process is written as

$$Y_t = \alpha Y_{t-1} + Z_t.$$

This says that this week's population is α times last week's population plus a random term Z_t. The randomness is **white noise**; the values of Z are **serially independent**, they have a **mean of zero**, and they have **finite variance** σ^2.

In a stationary times series $-1 < \alpha < 1$. In general, then, the autocorrelation function of $\{Y(t)\}$ is

$$\rho_k = \alpha^k, \qquad k = 0, 1, 2, \ldots.$$

Partial autocorrelation is the relationship between this week's population and the population at lag t when we have controlled for the correlations between all of the successive weeks between this week and week t. That is to say, the partial autocorrelation is the correlation between $Y(t)$ and $Y(t + k)$ after regression of $Y(t)$ on $Y(t + 1)$, $Y(t + 2)$, $Y(t + 3)$, $\ldots$, $Y(t + k - 1)$. It is obtained by solving the Yule–Walker equation

$$\rho_k = \sum_1^p \alpha_i \rho_{k-i}, \qquad k > 0,$$

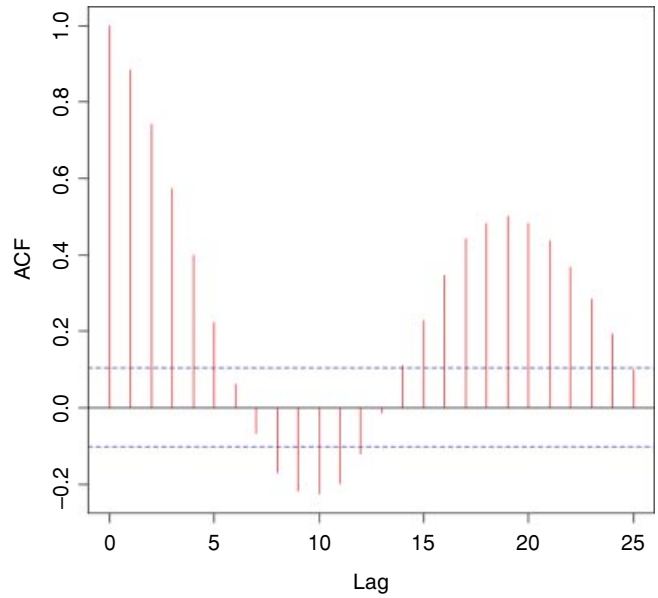
with the ρ replaced by r (correlation coefficients estimated from the data). Suppose we want the partial autocorrelation between time 1 and time 3. To calculate this, we need the three ordinary correlation coefficients r_{12}, r_{13} and r_{23}. The partial $r_{13,2}$ is then

$$r_{13,2} = \frac{r_{13} - r_{12}r_{23}}{\sqrt{(1 - r_{12}^2)(1 - r_{23}^2)}}.$$

For more on partial correlation coefficients, see p. 375.

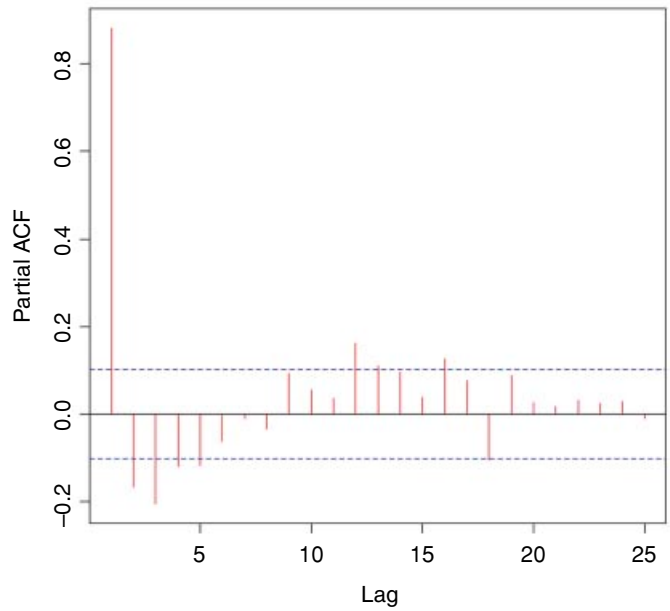
Let us look at the correlation structure of the blowfly data. The R function for calculating autocorrelations and partial autocorrelations is `acf` (the 'autocorrelation function'). First, we produce the autocorrelation plot to look for evidence of cyclic behaviour:

```
acf(flies,main="",col="red")
```



You will not see more convincing evidence of cycles than this. The blowflies exhibit highly significant, regular cycles with a period of 19 weeks. The blue dashed lines indicate the threshold values for significant correlation. What kind of time lags are involved in the generation of these cycles? We use partial autocorrelation (`type="p"`) to find this out:

```
acf(flies,type="p",main="",col="red")
```



The significant density-dependent effects are manifest at lags of 2 and 3 weeks, with other, marginally significant negative effects at lags of 4 and 5 weeks. These lags reflect the duration of the larval and pupal period (1 and 2 periods, respectively). The cycles are clearly caused by overcompensating density dependence, resulting from intraspecific competition between the larvae for food (what Nicholson christened 'scramble competition'). There is a curious positive feedback at a lag of 12 weeks (12–16 weeks, in fact). Perhaps you can think of a possible cause for this?

We should investigate the behaviour of the second half of the time series separately. Let us say it is from week 201 onwards:

```
second <- flies[201:361]
```

Now test for a linear trend in mean fly numbers against day number, from 1 to `length(second)`:

```
summary(lm(second~I(1:length(second))))
```

Note the use of `I` in the model formula (for 'as is') to tell R that the colon we have used is to generate a sequence of x values for the regression (and not an interaction term as it would otherwise have assumed).

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         2827.531    336.661   8.399 2.37e-14 ***
I(1:length(second))   21.945      3.605   6.087 8.29e-09 ***
```

```
Residual standard error: 2126 on 159 degrees of freedom

Multiple R-squared: 0.189,      Adjusted R-squared: 0.1839
F-statistic: 37.05 on 1 and 159 DF,  p-value: 8.289e-09
```

This shows that there is a highly significant upward trend of about 22 extra flies on average each week in the second half of time series. We can detrend the data by subtracting the fitted values from the linear regression of `second` on day number:

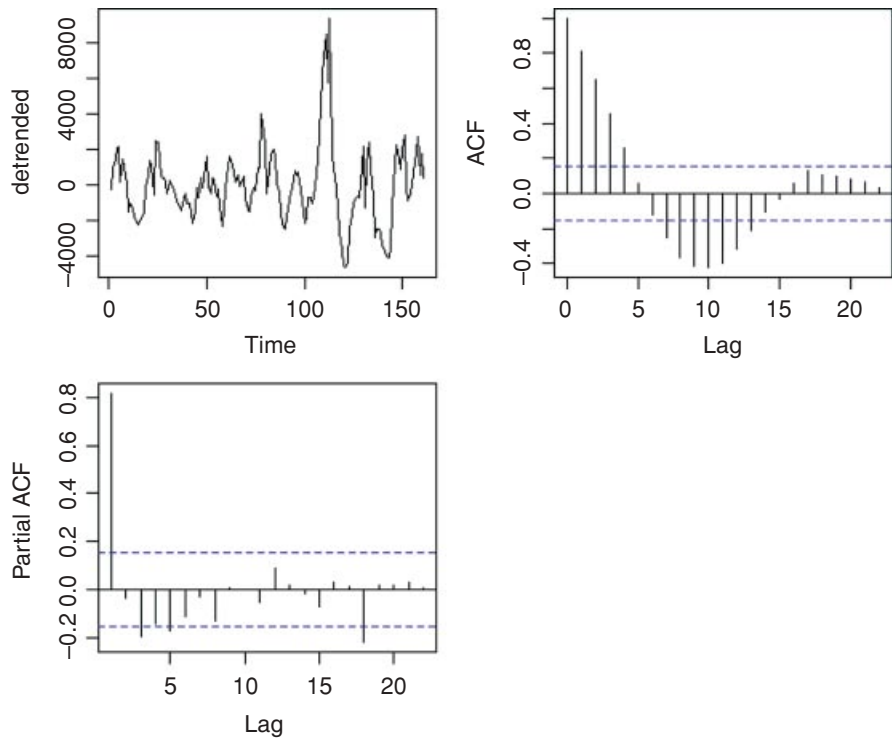
```
detrended <- second - predict(lm(second~I(1:length(second))))
par(mfrow=c(2,2))
ts.plot(detrended)
```

There are still cycles there, but they are weaker and less regular. We repeat the correlation analysis on the detrended data:

```
acf(detrended,main="")
```

These look more like damped oscillations than repeated cycles. What about the partials?

```
acf(detrended,type="p",main="")
par(mfrow=c(1,1))
```



There are still significant negative partial autocorrelations at lags 3 and 5, but now there is a curious extra negative partial at lag 18. It looks, therefore, as if the main features of the ecology are the same (scramble

competition for food between the larvae, leading to negative partials at 3 and 5 weeks after 1 and 2 generation lags), but population size is drifting upwards and the cycles are showing a tendency to dampen out.

24.2 Moving average

The simplest way of seeing pattern in time series data is to plot the moving average. A useful summary statistic is the three-point moving average:

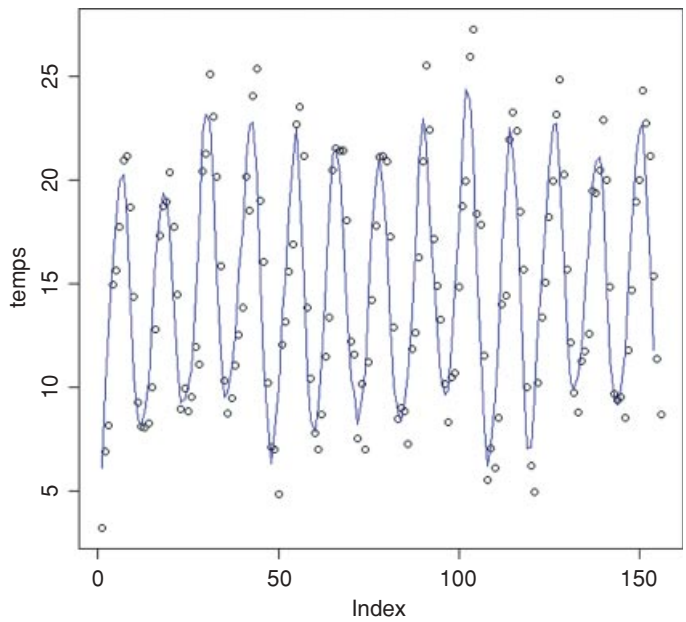
$$y_i' = \frac{y_{i-1} + y_i + y_{i+1}}{3}.$$

The function `ma3` will compute the three-point moving average for any input vector x:

```
ma3 <- function (x) {
      y <- numeric(length(x)-2)
for (i in 2:(length(x)-1)) {
      y[i] <- (x[i-1]+x[i]+x[i+1])/3
}
y }
```

A time series of mean monthly temperatures will illustrate the use of the moving average:

```
temperature <- read.table("c:\\temp\\temp.txt",header=T)
attach(temperature)
tm <- ma3(temps)
plot(temps)
lines(tm[2:158],col="blue")
```



The seasonal pattern of temperature change over the 13 years of the data is clear. Note that a moving average can never capture the maxima or minima of a series (because they are averaged away). Note also that the three-point moving average is undefined for the first and last points in the series.

24.3 Seasonal data

Many time series applications involve data that exhibit seasonal cycles. The commonest applications involve weather data. Here are daily maximum and minimum temperatures from Silwood Park in south-east England over a 19-year period:

```
weather <- read.table("c:\\temp\\SilwoodWeather.txt",header=T)
attach(weather)
names(weather)

[1] "upper" "lower" "rain" "month" "yr"

plot(upper,type="l")
```

The seasonal pattern of temperature change is clear, but there is no clear trend (e.g. warming, see p. 791). Note that the x axis is labelled by the day number of the time series ('Index').

We start by modelling the seasonal component. The simplest models for cycles are scaled so that a complete annual cycle is of length 1.0 (rather than 365 days). Our series consists of 6940 days over a 19-year span, so we write:

```
length(upper)

[1] 6940

index <- 1:6940
6940/19

[1] 365.2632

time <- index/365.2632
```

The equation for the seasonal cycle is:

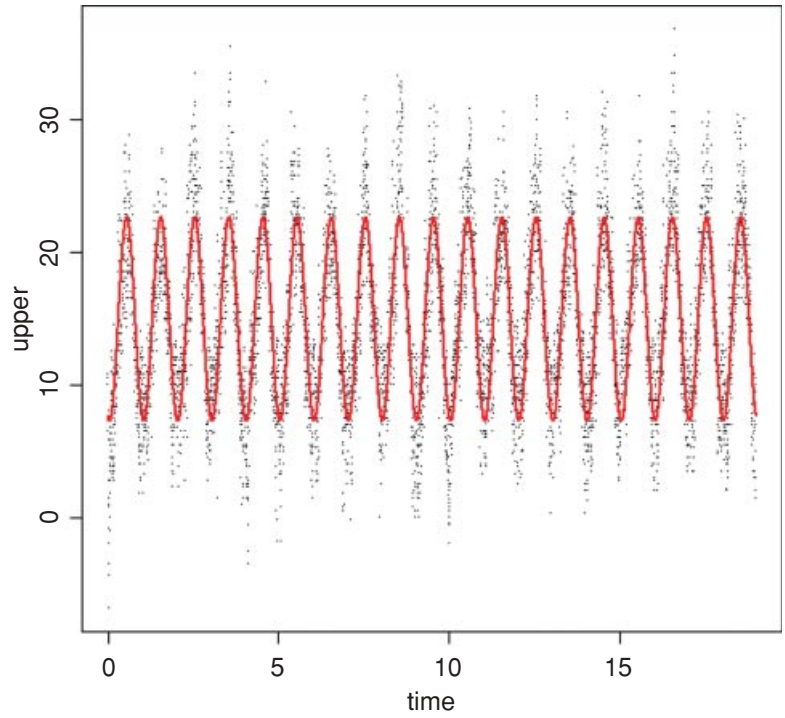
$$y = \alpha + \beta \sin(2\pi t) + \gamma \cos(2\pi t) + \varepsilon.$$

This is a linear model, so we can estimate its three parameters very simply:

```
model <- lm(upper~sin(time*2*pi)+cos(time*2*pi))
```

To investigate the fit of this model we need to plot the scattergraph using very small symbols (otherwise the fitted line will be completely obscured). The smallest useful plotting symbol is the dot ".":

```
plot(time, upper, pch=".")
lines(time, predict(model),col="red",lwd=2)
```

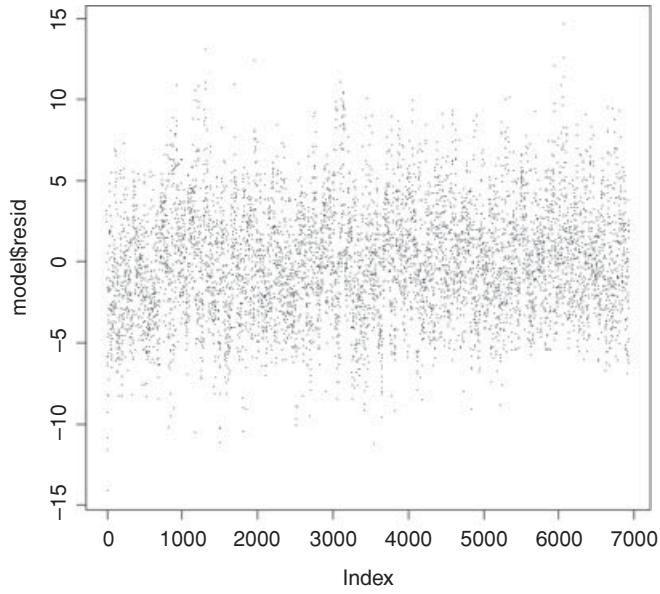


The three parameters of the model are all highly significant:

```
summary(model)

Call:
lm(formula = upper ~ sin(time * 2 * pi) + cos(time * 2 * pi))

Residuals:
     Min       1Q   Median       3Q      Max
-14.1336  -2.4220  -0.1233   2.2162  14.6456

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       14.95647    0.04088  365.86   <2e-16 ***
sin(time * 2 * pi) -2.53883    0.05781  -43.91   <2e-16 ***
cos(time * 2 * pi) -7.24017    0.05781 -125.23   <2e-16 ***

Residual standard error: 3.406 on 6937 degrees of freedom
Multiple R-squared: 0.7174,     Adjusted R-squared: 0.7173
F-statistic:  8806 on 2 and 6937 DF,  p-value: < 2.2e-16
```

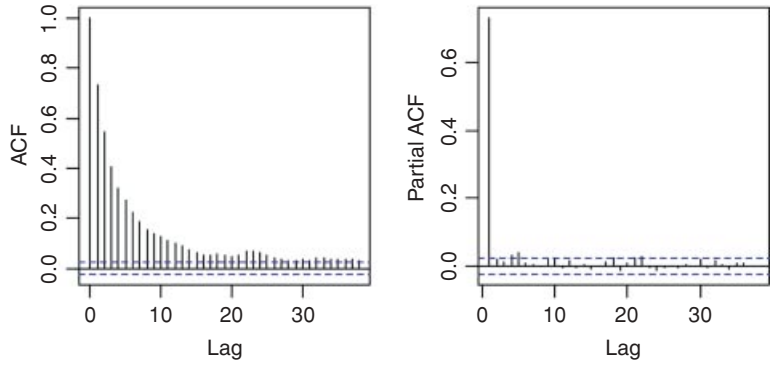
We can investigate the residuals to look for patterns (e.g. trends in the mean, or autocorrelation structure). Remember that the residuals are stored as part of the model object:

```
plot(model$resid,pch=".")
```



There looks to be some periodicity in the residuals, but no obvious trends. To look for serial correlation in the residuals, we use the `acf` function like this:

```
windows(7,4)
par(mfrow=c(1,2))
acf(model$resid,main="")
acf(model$resid,type="p",main="")
```

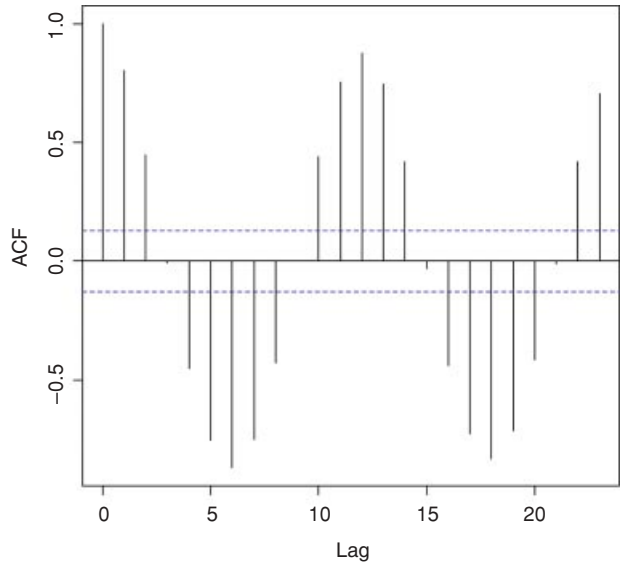


There is very strong serial correlation in the residuals, and this drops off roughly exponentially with increasing lag (left-hand graph). The partial autocorrelation at lag 1 is very large (0.7317), but the correlations at higher lags are much smaller. This suggests that an AR(1) model (autoregressive model with order 1) might be appropriate. This is the statistical justification behind the old joke about the weather forecaster who was asked what tomorrow's weather would be. 'Like today's', he said.

24.3.1 Pattern in the monthly means

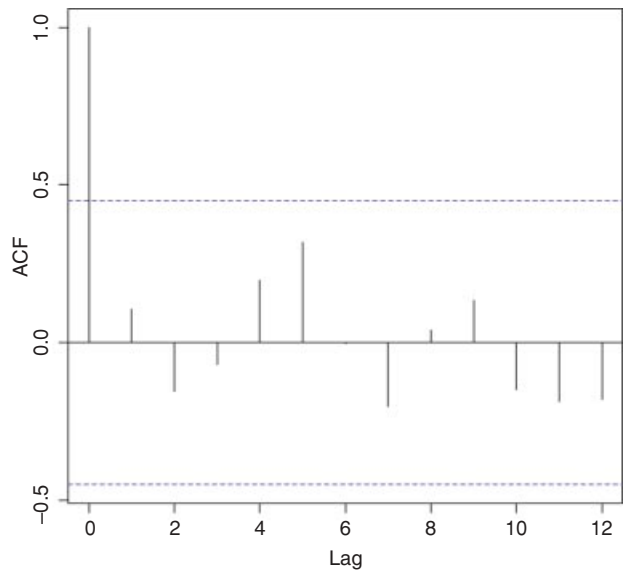
The monthly average upper temperatures show a beautiful seasonal pattern when analysed by `acf`:

```
temp <- ts(as.vector(tapply(upper,list(month,yr),mean)))
windows(7,7)
acf(temp,main="")
```



There is a perfect cycle with period 12 (as expected). What about patterns across years?

```
ytemp <- ts(as.vector(tapply(upper,yr,mean)))
acf(ytemp,main="")
```



Nothing! The pattern you may (or may not) see depends upon the scale at which you look for it. As for spatial patterns (Chapter 26), so it is with temporal patterns. There is strong pattern between days within months (tomorrow will be like today). There is very strong pattern from month to month within years (January is cold, July is warm). But there is no pattern at all from year to year (there may be progressive global warming, but it is not apparent within this recent time series (see below), and there is absolutely no evidence for untrended serial correlation).

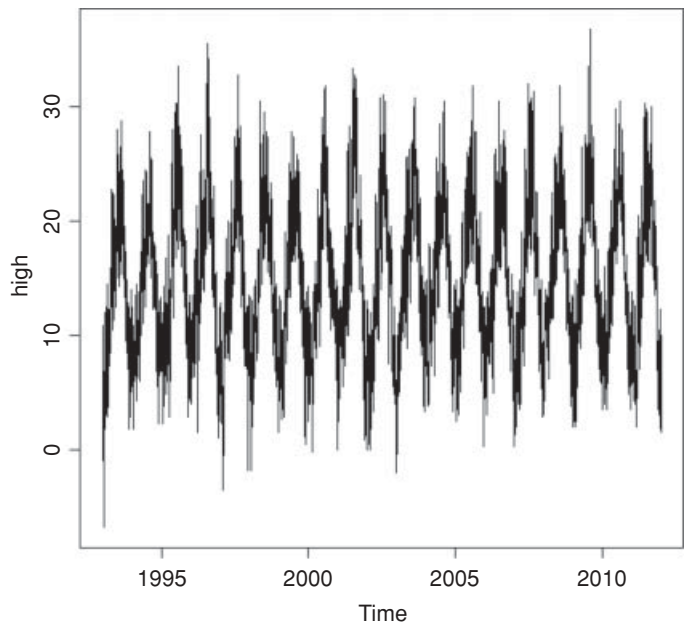
24.4 Built-in time series functions

The analysis is simpler, and the graphics are better labelled, if we convert the temperature data into a regular time series object using `ts`. We need to specify the first date (January 1993) as `start=c(1993,1)`, and the number of data points per year as `frequency=365`.

```
high <- ts(upper,start=c(1993,1),frequency=365)
```

Now use `plot` to see a plot of the time series, correctly labelled by years:

```
plot(high)
```



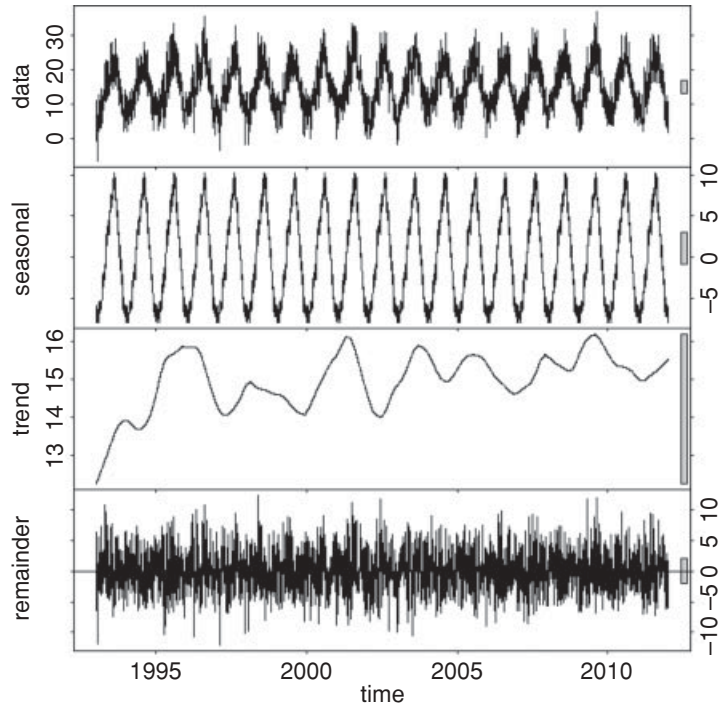
24.5 Decompositions

It is useful to be able to turn a time series into components. The function `stl` (with a lower-case letter L, not numeral one) performs seasonal decomposition of a time series into seasonal, trend and irregular components using `loess`. First, we make a time series object, specifying the start date and the frequency (as in Section 24.4), then use `stl` to decompose the series:

```
up <- stl(high,"periodic")
```

The `plot` function produces the data series, the seasonal component, the trend and the residuals in a single frame:

```
plot(up)
```



The remainder component is the residuals from the seasonal plus trend fit. The bars at the right-hand side are of equal heights (in user coordinates).

24.6 Testing for a trend in the time series

It is important to know whether these data provide any evidence for global warming. The trend part of the figure indicates a fluctuating increase, but is it significant? The mean temperature in the last 9 years was 0.71°C higher than in the first 10 years:

```
ys <- factor(1+(yr>2002))
tapply(upper,ys,mean)
```

```
       1        2
14.62056  15.32978
```

We cannot test for a trend with linear regression because of the massive temporal pseudoreplication. Suppose we tried this:

```
model1 <- lm(upper~index+sin(time*2*pi)+cos(time*2*pi))
summary(model1)
```

```
Coefficients:
                      Estimate Std. Error  t value Pr(>|t|)
(Intercept)          1.433e+01  8.136e-02  176.113   <2e-16 ***
index                1.807e-04  2.031e-05    8.896   <2e-16 ***
sin(time * 2 * pi) -2.518e+00  5.754e-02  -43.758   <2e-16 ***
cos(time * 2 * pi) -7.240e+00  5.749e-02 -125.939   <2e-16 ***

Residual standard error: 3.387 on 6936 degrees of freedom
Multiple R-squared: 0.7206,     Adjusted R-squared: 0.7205
F-statistic:  5963 on 3 and 6936 DF,  p-value: < 2.2e-16
```

It would suggest (wrongly, as we shall see) that the warming was highly significant (index p value less than 2×10^{-16} for a slope of 0.000 180 7 degrees of warming per day, leading to a predicted increase in mean temperature of 1.254°C over the 6940 days of the time series).

Since there is so much temporal pseudoreplication we should use a mixed model (`lmer`, p. 695), and because we intend to compare two models with different fixed effects we use the method of maximum likelihood (`REML=FALSE`). The explanatory variable for any trend is `index`, and we fit the model with and without this variable, allowing for different intercepts for the different years as a random effect:

```
model2 <-
 lmer(upper~index+sin(time*2*pi)+cos(time*2*pi)+(1 | factor(yr)),REML=FALSE)
model3 <-
 lmer(upper~sin(time*2*pi)+cos(time*2*pi)+(1 | factor(yr)),REML=FALSE)
anova(model2,model3)
```

```
Data:
Models:
model3: upper ~ sin(time * 2 * pi) + cos(time * 2 * pi) + (1 | factor(yr))
model2: upper ~ index + sin(time * 2 * pi) + cos(time * 2 * pi) + (1 |
model2:     factor(yr))
       Df   AIC   BIC logLik Chisq Chi Df Pr(>Chisq)
model3  5 36452 36486 -18221
model2  6 36458 36499 -18223     0      1          1
```

Clearly, the trend is non-significant (chi-squared = 0, $p = 1$). If you are prepared to ignore all the variation (from day to day and from month to month), then you can get rid of the pseudoreplication by averaging and test for trend in the yearly mean values: these show a significant trend if the first year (1993) is included, but not if it is omitted:

```
means <- as.vector(tapply(upper,yr,mean))
model <- lm(means~I(1:19))
summary(model)
```

```
Coefficients:
             Estimate  Std. Error   t value   Pr(> | t |)
(Intercept)  14.27105     0.32220    44.293        <2e-16 ***
I(1:19)       0.06858     0.02826     2.427        0.0266 *
```

```
model <- lm(means[-1]~I(1:18))
summary(model)
```

```
Coefficients:
             Estimate   Std. Error   t value   Pr(> | t |)
(Intercept)  14.59826      0.30901    47.243        <2e-16 ***
I(1:18)       0.04761      0.02855     1.668         0.115
```

Obviously, you need to be circumspect when interpreting trends in time series.

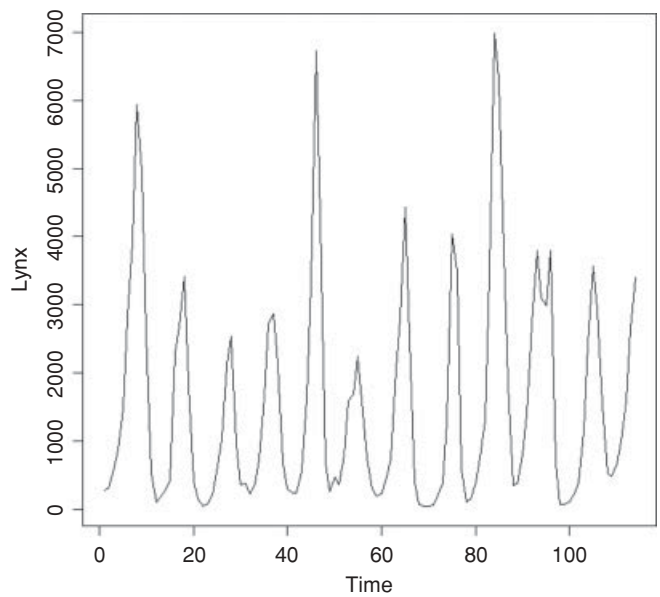
24.7 Spectral analysis

There is an alternative approach to time series analysis, which is based on the analysis of **frequencies** rather than fluctuations of numbers. Frequency is the reciprocal of cycle period. Ten-year cycles would have a frequency 0.1 per year. Here are the famous Canadian lynx data:

```
numbers <- read.table("c:\\temp\\lynx.txt",header=T)
attach(numbers)
names(numbers)
```

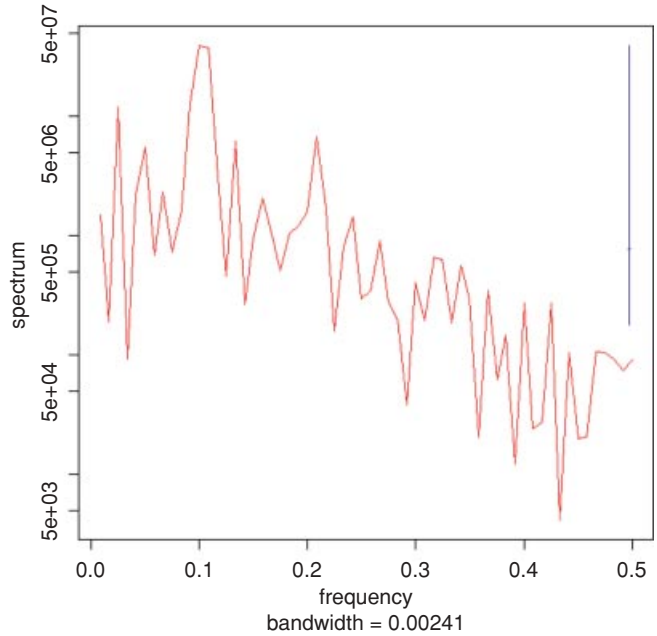
```
[1] "Lynx"
```

```
plot.ts(Lynx)
```



The fundamental tool of spectral analysis is the **periodogram**. This is based on the squared correlation between the time series and sine/cosine waves of frequency ω, and conveys exactly the same information as the autocovariance function. It may (or may not) make the information easier to interpret. Using the function is straightforward; we employ the `spectrum` function like this:

```
spectrum(Lynx,main="",col="red")
```



The plot is on a log scale, in units of decibels, and the subtitle on the x axis shows the bandwidth, while the 95% confidence interval in decibels is shown by the vertical blue bar in the top right-hand corner. The figure is interpreted as showing strong cycles with a frequency of about 0.1, where the maximum value of spectrum occurs. That is to say, it indicates cycles with a period of $1/0.1 = 10$ years. There is a hint of longer period cycles (the local peak at frequency 0.033 would produce cycles of length $1/0.033 = 30$ years) but no real suggestion of any shorter-term cycles.

24.8 Multiple time series

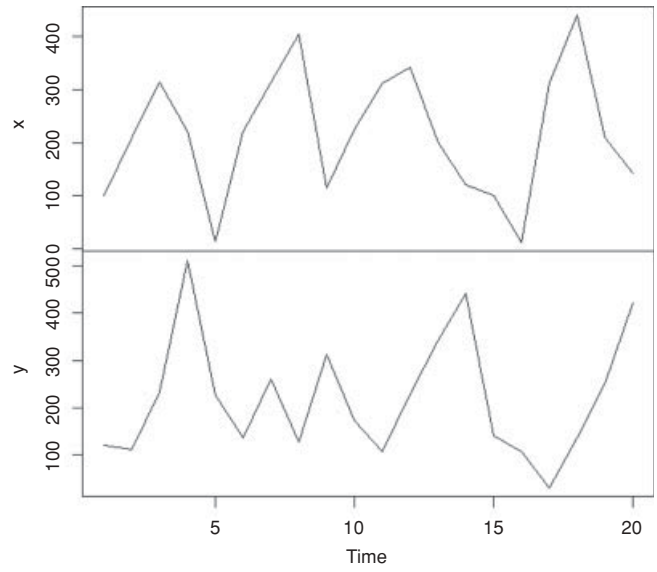
When we have two or more time series measured over the same period, the question naturally arises as to whether or not the ups and downs of the different series are correlated. It may be that we suspect that change in one of the variables causes changes in the other (e.g. changes in the number of predators may cause changes in the number of prey, because more predators means more prey eaten). We need to be careful, of course, because it will not always be obvious which way round the causal relationship might work (e.g. predator numbers may go up because prey numbers are higher; ecologists call this a numerical response). Suppose we have the following sets of counts:

```
twoseries <- read.table("c:\\temp\\twoseries.txt",header=T)
attach(twoseries)
names(twoseries)
```

```
[1] "x" "y"
```

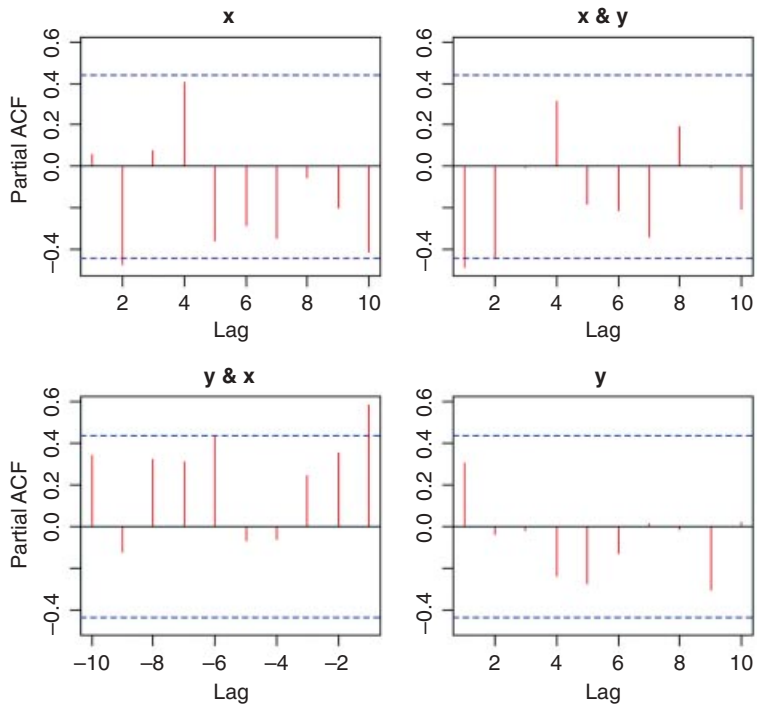
We start by inspecting the two time series one above the other:

```
plot.ts(cbind(x,y),main="")
```



There is some evidence of periodicity (at least in x) and it looks as if y lags behind x by roughly 2 periods (sometimes 1). Now let us carry out straightforward analyses on each time series separately and the cross-correlation between the two series:

```
par(mfrow=c(2,2))
acf(cbind(x,y),type="p",col="red")
```



As we suspected, the evidence for periodicity is stronger in x than in y: the partial autocorrelation is significant and negative at lag 2 for x, but not for y. The interesting point is the cross-correlation between x and y which is significant at lags 1 and 2 (top right). Positive changes in x are associated with negative changes in y and vice versa.

24.9 Simulated time series

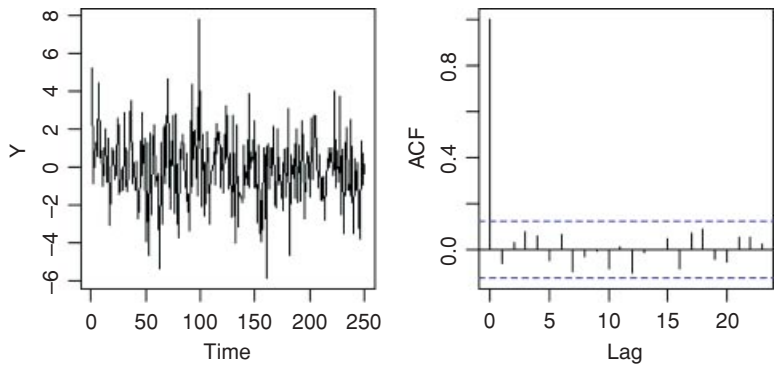
To see how the correlation structure of an AR(1) depends on the value of α, we can simulate the process over, say, 250 time periods using different values of α. We generate the white noise Z_t using the random number generator `rnorm(n,0,s)` which gives n random numbers with a mean of 0 and a standard deviation of s. To simulate the time series we evaluate

$$Y_t = \alpha Y_{t-1} + Z_t,$$

multiplying last year's population by α then adding the relevant random number from Z_t.

We begin with the special case of $\alpha = 0$ so that $Y_t = Z_t$ and the process is pure white noise:

```
Y <- rnorm(250,0,2)
windows(7,4)
par(mfrow=c(1,2))
plot.ts(Y)
acf(Y,main="")
```



The time series is bound to be stationary because each value of Z is independent of the value before it. The correlation at lag 0 is 1 (of course), but there is absolutely no hint of any correlations at higher lags.

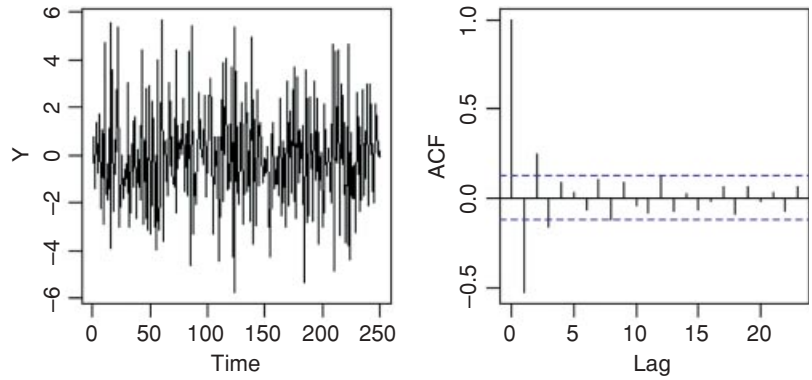
To generate the time series for non-zero values of α we need to use recursion: this year's population is last year's population times α plus the white noise. We begin with a negative value of $\alpha = -0.5$. First we generate all the noise values (by definition, these *do not* depend on population size):

```
Z <- rnorm(250,0,2)
```

Now the initial population at time 0 is set to 0 (remember that the population is stationary, so we can think of the Y values as departures from the long-term mean population size). This means that $Y_1 = Z_1$. Thus, Y_2 will be whatever Y_1 was, times –0.5, plus Z_2. And so on.

```
Y <- numeric(250)
Y[1] <- Z[1]
```

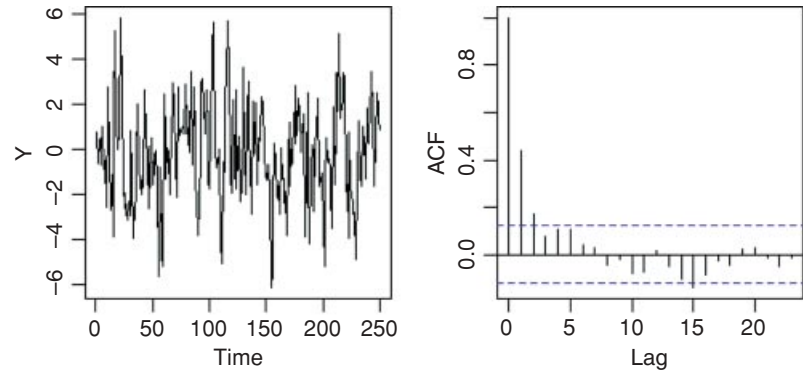
```
for (i in 2:250) Y[i] <-  -0.5*Y[i-1]+Z[i]
plot.ts(Y)
acf(Y,main="")
```



The time series shows rapid return to equilibrium following random departures from it. There is a highly significant negative autocorrelation at lag 1, significant positive autocorrelation at lag 2 and so on, with the size of the correlation gradually damping away.

Let us simulate a time series with a positive value of, say, $\alpha = 0.5$:

```
Z <- rnorm(250,0,2)
Y[1] <- Z[1]
for (i in 2:250) Y[i] <-  0.5*Y[i-1]+Z[i]
plot.ts(Y)
acf(Y, main="")
```

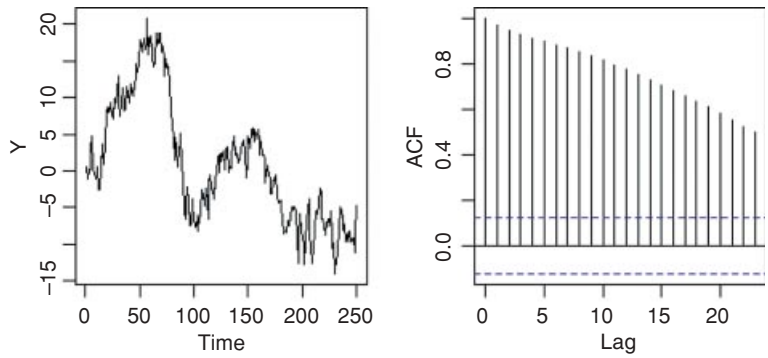


Now the time series plot looks very different, with protracted periods spent drifting away from the long-term average. The autocorrelation plot shows significant positive correlations for the first three lags.

Finally, we look at the special case of $\alpha = 1$. This means that the time series is a classic **random walk**, given by

$$Y_t = \mathrm{Y}_{t-1} + Z_t.$$

```
Z <- rnorm(250,0,2)
Y[1] <- Z[1]
for (i in 2:250) Y[i] <-  Y[i-1]+Z[i]
plot.ts(Y)
acf(Y, main="")
```



The time series wanders about and strays far away from the long-term average. The `acf` plot shows positive correlations dying away very slowly, and still highly significant at lags of more than 20. Of course, if you do another realization of the process, the time series will look very different, but the autocorrelations will be similar.

24.10 Time series models

Time series models come in three kinds (Box and Jenkins, 1976):

- moving average (MA) models where

$$X_t = \sum_{j=0}^{q} \beta_j \varepsilon_{t-j};$$

- autoregressive (AR) models where

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \varepsilon_t;$$

- autoregressive moving average (ARMA) models where

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{j=0}^{q} \beta_j \varepsilon_{t-j}.$$

A moving average of order q averages the random variation over the last q time periods. An autoregressive model of order p computes X_t as a function of the last p values of X, so, for a second-order process, we would use

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \varepsilon_t.$$

Typically, we would use the partial autocorrelation plot (above) to determine the order. So, for the lynx data (p. 800) we would use order 2 or 4, depending on taste. Other things being equal, parsimony suggests the use of order 2. The fundamental difference is that a set of random components (ε_{t-j}) influences the current value of a MA process, whereas only the current random effect (ε_t) affects an AR process. Both kinds of effects are at work in an ARMA processes. Ecological models of population dynamics are typically AR models. For instance,

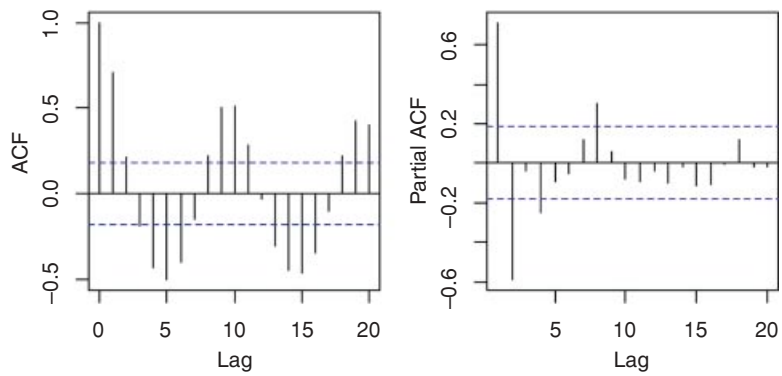
$$N_t = \lambda N_{t-1}$$

(the discrete-time version of exponential growth ($\lambda > 1$) or decay ($\lambda < 1$)) looks just like an first order AR process with the random effects missing. This is somewhat misleading, however, since time series are supposed to be stationary, which would imply a long-term average value of $\lambda = 1$. But, in the absence of density dependence (as here), this is impossible. The α of the AR model is *not* the λ of the population model.

Models are fitted using the `arima` function, and their performances are compared using the AIC (see p. 415). The most important component of the model is `order`. This is a vector of length 3 specifying the order of the autoregressive operators, the number of differences, and the order of moving average operators. Thus `order=c(1,3,2)` is based on a first-order autoregressive process, three differences, and a second-order moving average. The Canadian lynx data are used as an example of `arima` in time series modelling.

Records of the number of skins of predators (lynx) and prey (snowshoe hares) returned by trappers were collected over many years by the Hudson's Bay Company. The lynx numbers are shown on p. 800 and exhibit a clear 10-year cycle. We begin by plotting the autocorrelation and partial autocorrelation functions:

```
windows(7,4)
par(mfrow=c(1,2))
acf(Lynx,main="")
acf(Lynx,type="p",main="")
```



The population is very clearly cyclic, with a period of 10 years. The dynamics appear to be driven by strong, negative density dependence (a partial autocorrelation of –0.588) at lag 2. There are other significant partials at lag 1 and lag 8 (positive) and lag 4 (negative). Of course you cannot infer the mechanism by observing the

dynamics, but the lags associated with significant negative and positive feedbacks are extremely interesting and highly suggestive. The main prey species of the lynx is the snowshoe hare and the negative feedback at lag 2 may reflect the timescale of this predator–prey interaction. The hares are known to cause medium-term induced reductions in the quality of their food plants as a result of heavy browsing pressure when the hares are at high density, and this could map through to lynx populations with lag 4.

The `order` vector specifies the non-seasonal part of the ARIMA model: the three components (p, d, q) are the AR order, the degree of differencing, and the MA order. We start by investigating the effects of AR order with no differencing and no moving average terms, comparing models on the basis of the AIC:

```
model10 <- arima(Lynx,order=c(1,0,0))
model20 <- arima(Lynx,order=c(2,0,0))
model30 <- arima(Lynx,order=c(3,0,0))
model40 <- arima(Lynx,order=c(4,0,0))
model50 <- arima(Lynx,order=c(5,0,0))
model60 <- arima(Lynx,order=c(6,0,0))
AIC(model10,model20,model30,model40,model50,model60)
```

```
        df      AIC
model10  3 1926.991
model20  4 1878.032
model30  5 1879.957
model40  6 1874.222
model50  7 1875.276
model60  8 1876.858
```

On the basis of AR alone, it appears that order 4 is best (AIC = 1874.222). What about MA?

```
model01 <- arima(Lynx,order=c(0,0,1))
model02 <- arima(Lynx,order=c(0,0,2))
model03 <- arima(Lynx,order=c(0,0,3))
model04 <- arima(Lynx,order=c(0,0,4))
model05 <- arima(Lynx,order=c(0,0,5))
model06 <- arima(Lynx,order=c(0,0,6))
AIC(model01,model02,model03,model04,model05,model06)
```

```
        df      AIC
model01  3 1917.947
model02  4 1890.061
model03  5 1887.770
model04  6 1888.279
model05  7 1885.698
model06  8 1885.230
```

The AIC values are generally higher than given by the AR models. Perhaps there is a combination of AR and MA terms that is better than either on their own?

```
model40 <- arima(Lynx,order=c(4,0,0))
model41 <- arima(Lynx,order=c(4,0,1))
model42 <- arima(Lynx,order=c(4,0,2))
model43 <- arima(Lynx,order=c(4,0,3))
AIC(model40,model41,model42,model43)
```

```
         df      AIC
model40   6 1874.222
model41   7 1875.351
model42   8 1862.435
model43   9 1880.432
```

Evidently there is no need for a moving average term (`model40` is best). What about the degree of differencing?

```
model400 <- arima(Lynx,order=c(4,0,0))
model401 <- arima(Lynx,order=c(4,1,0))
model402 <- arima(Lynx,order=c(4,2,0))
model403 <- arima(Lynx,order=c(4,3,0))
AIC(model400,model401,model402,model403)
```

```
         df      AIC
model400  6 1874.222
model401  5 1890.961
model402  5 1917.882
model403  5 1946.143
```

The model with no differencing performs best. The lowest AIC is 1874.222, which suggests that a model with an AR lag of 4, no differencing and no moving average terms is best. This implies that a rather complex ecological model is required which takes account of both the significant partial correlations at lags of 2 and 4 years, and not just the 2-year lag (i.e. plant–herbivore effects may be necessary to explain the dynamics, in addition to predator–prey effects).