ARIMA Models

Lesson 3.2

Introduction

ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting, and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

Stationarity

A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time.

In general, a stationary time series will have no predictable patterns in the long-term. For example, a time series with cyclic behaviour (but with no trend or seasonality) is stationary. This is because the cycles are not of a fixed length, so before we observe the series we cannot be sure where the peaks and troughs of the cycles will be. Time plots will show the series to be roughly horizontal (although some cyclic behaviour is possible), with constant variance.

Which of the following plots do you think are stationary?

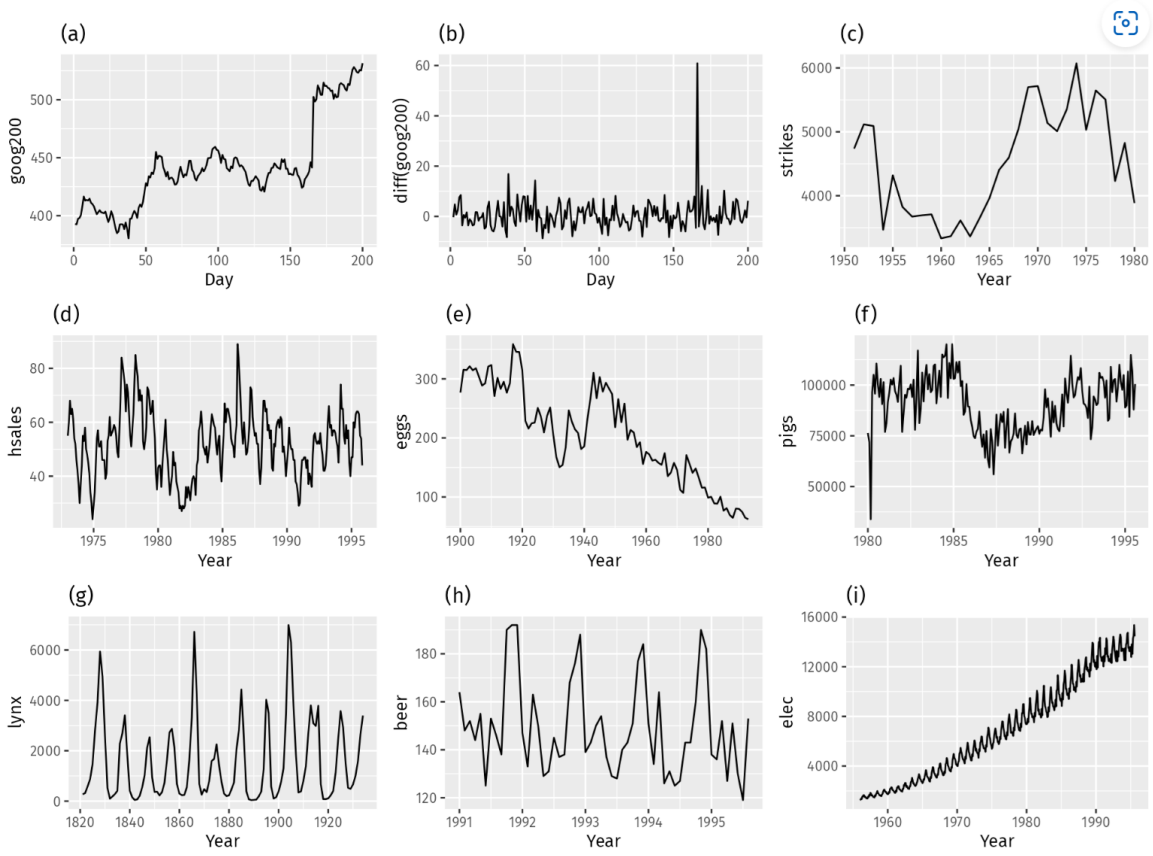

Figure 1: Which of these series are stationary?

Observe that the Google stock price was non-stationary in panel (a), but the daily changes were stationary in panel (b). This shows one way to make a non-stationary time series stationary — compute the differences between consecutive observations. This is known as differencing.

Transformations such as logarithms can help to stabilize the variance of a time series. Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

Another way to determine stationarity of a time series is to generate the ACF plot. For a stationary time series, the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly. Also, for non-stationary data, the value of
r_1 is often large and positive.

```
library(fpp2)
library(tidyverse)
library(patchwork)
```

```
g1 <- ggAcf(goog200)
g2 <- ggAcf(diff(goog200))
g1 + g2
```

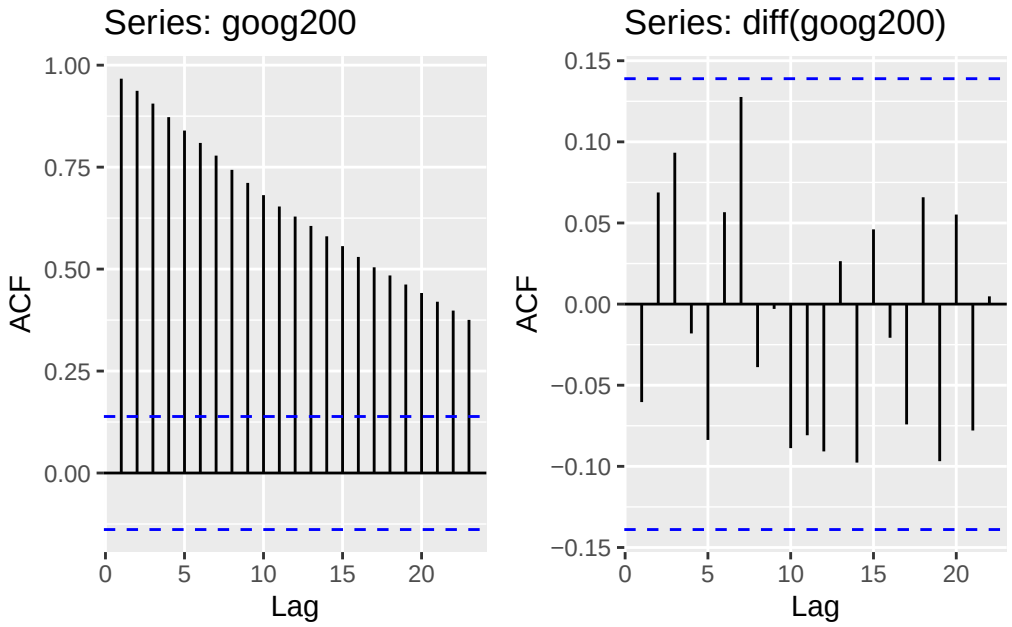

Figure 2: The ACF plots of the original Google stock price and the differnced series

The ACF of the differenced Google stock price looks just like that of a white noise series. There are no autocorrelations lying outside the 95% limits. To formally test if there is no more significant autocorrelations, we use the Ljung-Box test.

```
Box.test(diff(goog200), lag=10, type="Ljung-Box")
```

```
    Box-Ljung test

data:  diff(goog200)
X-squared = 11.031, df = 10, p-value = 0.3551
```

The above result suggests that the daily change in the Google stock price is essentially a random amount which is uncorrelated with that of previous days.

Random walk model

The differenced series which is the change between consecutive observations in the original series is written as

$$y_t^{'} = y_t - y_{t-1},\ t = 1, 2, \cdots, T$$

The differenced series will have only $T-1$ values, since it is not possible to calculate a difference $y_t^{'}$ for the first observation.

When the differenced series is white noise, the model for the original series can be written as

$$y_t - y_{t-1} = \epsilon_t$$

where ϵ_t denotes white noise. Rearranging this leads to the **random wal** model

$$y_t = y_{t-1} + \epsilon_t$$

Random walk models are widely used for non-stationary data, particularly financial and economic data. Random walks typically have:

- long periods of apparent trends up or down
- sudden and unpredictable changes in direction.

The forecasts from a random walk model are equal to the last observation, as future movements are unpredictable, and are equally likely to be up or down.

Second-order differencing

Occasionally the differenced data will not appear to be stationary and it may be necessary to difference the data a second time to obtain a stationary series:

$$\begin{aligned} y_t^{"} &= y_t^{'} - y_{t-1}^{'} \\ &= [y_t - y_{t-1}] - [y_{t-1} - y_{t-2}] \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned}$$

In this case, $y_t^{"}$ will have $T-2$ values. Then, we would model the "change in the changes" of the original data. In practice, it is almost never necessary to go beyond second-order differences.

Seasonal differencing

A seasonal difference is the difference between an observation and the previous observation from the same season. So

$$y_t^{'} = y_t - y_{t-m}$$

where $m =$ the number of seasons. These are also called *lag-m* differences, as we subtract the observation after a lag of m periods.

If seasonally differenced data appear to be white noise, then an appropriate model for the original data is

$$y_t = y_{t-m} + \epsilon_t$$

Forecasts from this model are equal to the last observation from the relevant season.

The following code chunk illustrates the use transformation and differencing to achieve stationarity. The transformation and differencing have made the series look relatively stationary.

```
cbind("Sales ($million)" = a10,
      "Monthly log sales" = log(a10),
      "Annual change in log sales" = diff(log(a10),12)) %>%
  autoplot(facets=TRUE) +
    xlab("Year") + ylab("") +
    ggtitle("Antidiabetic drug sales")
```

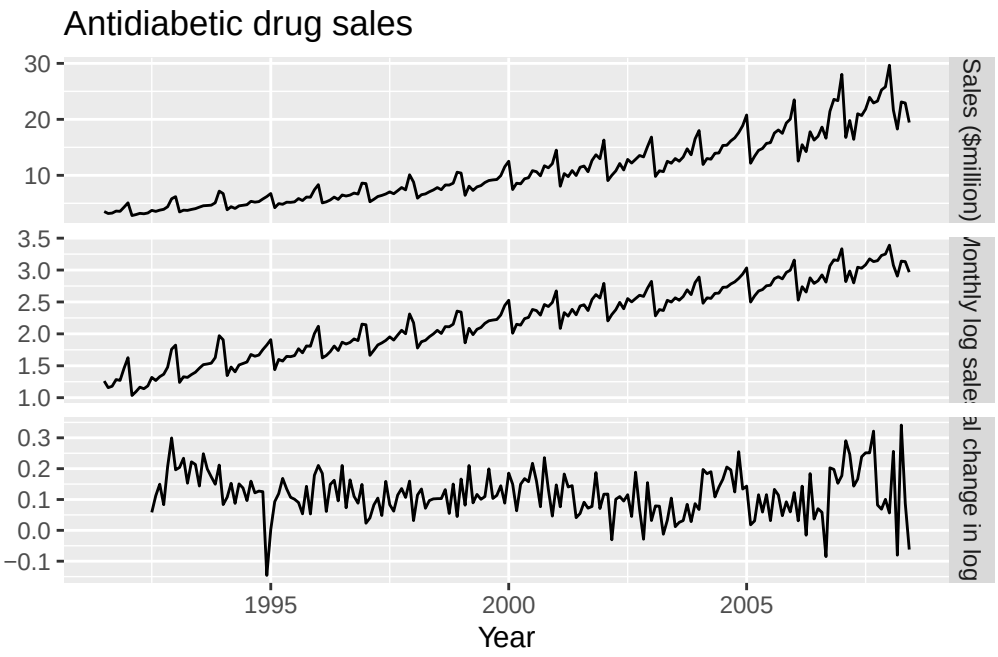

Figure 3: Logs and seasonal differences of A10 (antidiabetic) sales data

To distinguish seasonal differences from ordinary differences, we sometimes refer to ordinary differences as "first differences", meaning differences at lag 1.

Sometimes it is necessary to take both a seasonal difference and a first difference to obtain stationary data. The following code chunk illustrates this idea.

```
cbind("Billion kWh" = usmelec,
      "Logs" = log(usmelec),
      "Seasonally\n differenced logs" =
        diff(log(usmelec),12),
      "Doubly\n differenced logs" =
        diff(diff(log(usmelec),12),1)) %>%
  autoplot(facets=TRUE) +
    xlab("Year") + ylab("") +
    ggtitle("Monthly US net electricity generation")
```

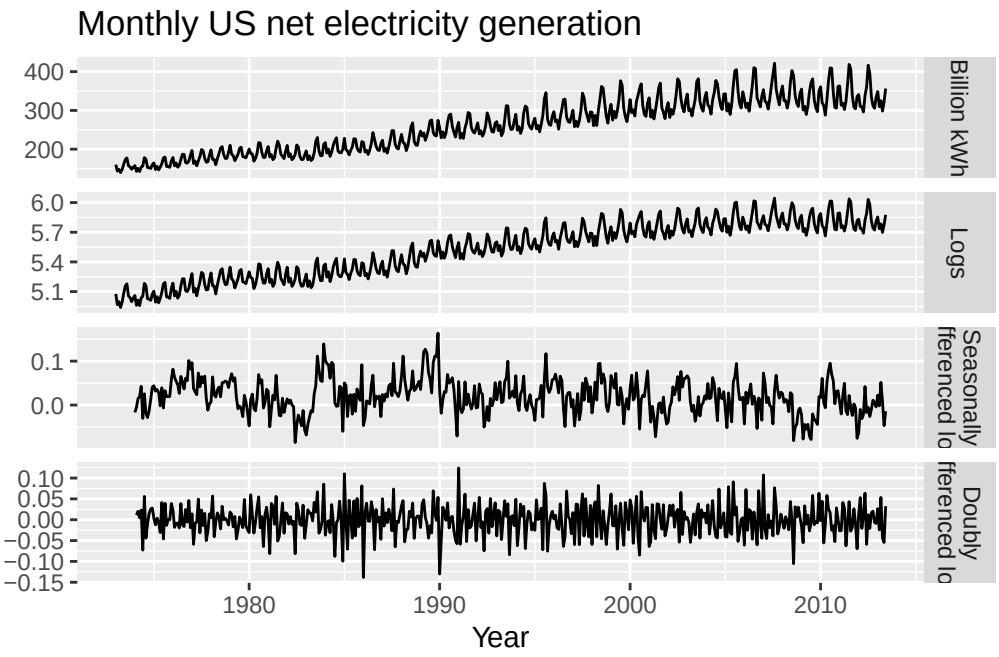

Figure 4: US net electricity generation (billion kWh)

Here, the data are first transformed using logarithms (second panel), then seasonal differences are calculated (third panel). The data still seem somewhat non-stationary, and so a further lot of first differences are computed (bottom panel).

There is a degree of subjectivity in selecting which differences to apply. The seasonally differenced data in the Figure3 do not show substantially different behaviour from the seasonally differenced data in Figure 4. In the latter case, we could have decided to stop with the seasonally differenced data, and not done an extra round of differencing. In the former case, we could have decided that the data were not sufficiently stationary and taken an extra round of differencing. Some formal tests for differencing are discussed below, but there are always some choices to be made in the modelling process, and different analysts may make different choices.

If $y_t^{'} = y_t - y_{t-m}$ denotes the seasonally differenced series, then the twice-differenced series is

$$\begin{aligned} y_t^{"} &= y_t^{'} - y_{t-1}^{'} \\ &= [y_t - y_{t-m}] - [y_{t-1} - y_{t-m-1}] \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1} \end{aligned}$$

When both seasonal and first differences are applied, it makes no difference which is done first—the result will be the same. However, if the data have a strong seasonal pattern, we recommend that seasonal differencing be done first, because the resulting series will sometimes

be stationary and there will be no need for a further first difference. If first differencing is done first, there will still be seasonality present.

It is important that if differencing is used, the differences are interpretable. First differences are the change between one observation and the next. Seasonal differences are the change between one year to the next. Other lags are unlikely to make much interpretable sense and should be avoided.

Unit root tests

One way to determine more objectively whether differencing is required is to use a unit root test. These are statistical hypothesis tests of stationarity that are designed for determining whether differencing is required.

A number of unit root tests are available, which are based on different assumptions and may lead to conflicting answers. In our analysis, we use the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test (Kwiatkowski, Phillips, Schmidt, & Shin, 1992). In this test, the null hypothesis is that the data are stationary, and we look for evidence that the null hypothesis is false. Consequently, small p-values (e.g., less than 0.05) suggest that differencing is required. The test can be computed using the *ur.kpss()* function from the **urca** package.

For example, let us apply it to the Google stock price data.

```
library(urca)
goog %>%
  ur.kpss() %>%
  summary()
```

```
######################
# KPSS Unit Root Test #
######################

Test is of type: mu with 7 lags.

Value of test-statistic is: 10.7223

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

The test statistic is much bigger than the 1% critical value, indicating that the null hypothesis is rejected. That is, the data are not stationary. We can difference the data, and apply the test again.

```
goog %>%
  diff() %>%
  ur.kpss() %>%
  summary()
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 7 lags.

Value of test-statistic is: 0.0324

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

This time, the test statistic is tiny, and well within the range we would expect for stationary data. So we can conclude that the differenced data are stationary.

This process of using a sequence of KPSS tests to determine the appropriate number of first differences is carried out by the function *ndiffs()*.

```
ndiffs(goog)
```

```
[1] 1
```

As we saw from the KPSS tests above, one difference is required to make the **goog** data stationary.

A similar function for determining whether seasonal differencing is required is *nsdiffs()* to determine the appropriate number of seasonal differences required. No seasonal differences are suggested if $F_S < 0.64$, otherwise one seasonal difference is suggested.

We can apply *nsdiffs()* to the logged US monthly electricity data.

```
usmelec %>%
  log() %>%
  nsdiffs()
```

```
[1] 1
```

```
usmelec %>%
  log() %>%
  diff(lag=12) %>%
  ndiffs()
```

```
[1] 1
```

Because *nsdiffs()* returns 1 (indicating one seasonal difference is required), we apply the *ndiffs()* function to the seasonally differenced data. These functions suggest we should do both a seasonal difference and a first difference.

Backshift operator

The backward shift operator B is a useful notational device when working with time series lags:

$$By_t = y_{t-1}$$

Some references use L for "lag" instead of B for "backshift". In other words,
B, operating on y_t has the effect of shifting the data back one period. Two applications of B to y_t shifts the data back two periods:

$$B(BY_t) = B^2 y_t = y_{t-2}$$

For monthly data, if we wish to consider "the same month last year," the notation is

$$B^{12} y_t = y_{t-12}$$

The backward shift operator is convenient for describing the process of differencing. A first difference can be written as

$$y_t^{'} = y_t - y_{t-1} = y_t - By_t = (1-B)y_t$$

Note that a first difference is represented by $(1-B)$. Similarly, if second-order differences have to be computed, then:

$$y_t^{"} = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1-B)^2 y_t$$

In general, a dth-order difference can be written as

$$(1-B)^d y_t$$

Backshift notation is particularly useful when combining differences, as the operator can be treated using ordinary algebraic rules. In particular, terms involving
B can be multiplied together.

For example, a seasonal difference followed by a first difference can be written as

$$\begin{aligned}(1-B)(1-B^m)y_t &= (1-B-B^m+B^{m+1})y_t \\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m+1}\end{aligned}$$

the same result obtained earlier.

Autoregressive models

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself.

Thus, an autoregressive model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p yt - p + \epsilon_t$$

where ϵ_t is white noise. This is like a multiple regression but with lagged values of y_t as predictors. We refer to this as an $AR(p)$ model, an autoregressive model of order p.

Autoregressive models are remarkably flexible at handling a wide range of different time series patterns. The series below show series from an $AR(1)$ model and an $AR(2)$ model. Changing the parameters $\phi_1, \phi_2, \cdots, \phi_p$ results in different time series patterns. The variance of the error term ϵ_t will only change the scale of the series, not the patterns.

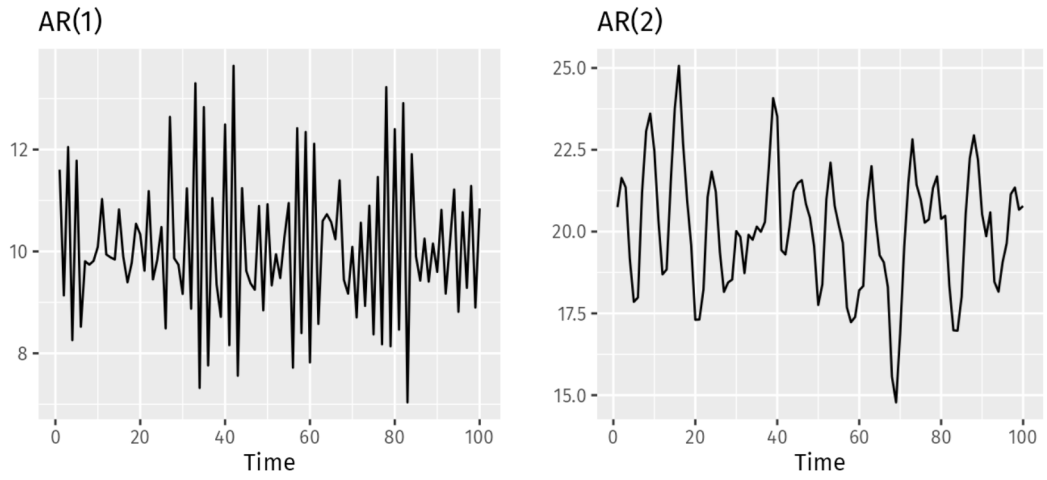

Figure 5: Series from AR(1) and AR(2) models

In the above figures, the $AR(1)$ model is $y_t = 18 - 0.8y_{t-1} + \epsilon_t$ and the $AR(2)$ model is $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \epsilon_t$ with $\epsilon_t \sim N(), 1)$.

For an $AR(1)$ model:

- when $\phi_1 = 0$, y_t is equivalent to white noise
- when $\phi_1 = 1$ and $c = 0$, then y_t is equivalent to a rabdom walk
- when $\phi_1 = 1$ and $c \neq 0$, then y_t is equivalent to a random walk with a drift
- when $\phi_1 < 0$, y_t tends to oscillate aroud the mean

We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required.

- For an $AR(1)$ model: $-1 < \phi_1 < 1$
- For an $AR(2)$ model: $-1 < \phi_2 < 1, \phi_1 + \phi_2 < 1, \phi_2 - \phi_1 < 1$

When $p \geq 3$, the restrictions are much more complicated. R takes care of these restrictions when estimating a model.

Moving average models

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

where ϵ_t is white noise. We refer to this as an $MA(q)$ model, a moving average model of order q. Of course, we do not observe the values of ϵ_t, so it is not really a regression in the usual sense.

Notice that each value of y_t can be thought of as a weighted moving average of the past few forecast errors. However, moving average models should not be confused with the moving average smoothing discussed in the previous lessons. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

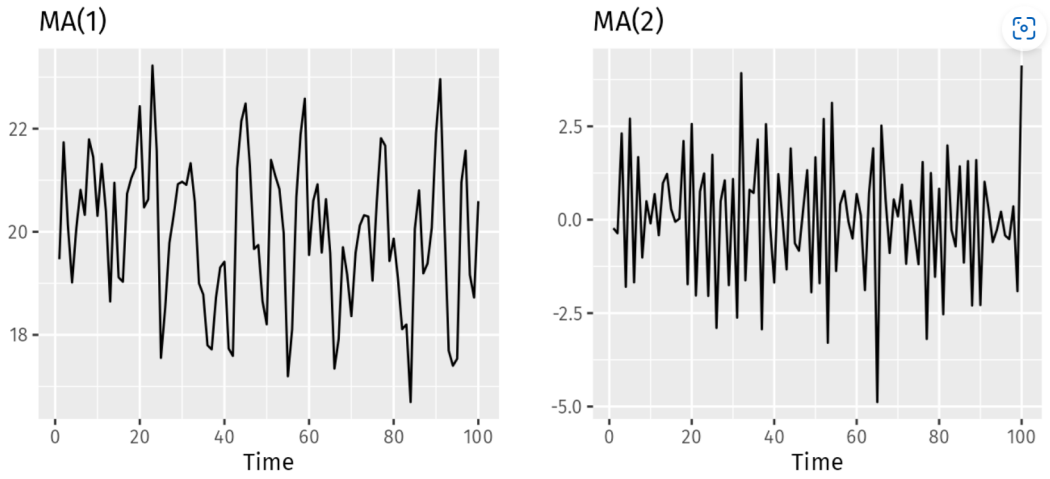

Figure 6: Series from MA(1) and MA(2) models

The equation of the $MA(1)$ model is $y_t = 20 + \epsilon_t + 0.8\epsilon_{t-1}$ and the $MA(2)$ model is $y_t = \epsilon_t - \epsilon_{t-1} + 0.8\epsilon_{t-2}$. In both cases, $\epsilon_t \sim N(0, 1)$.

Figure 6 shows some data from an MA(1) model and an MA(2) model. Changing the parameters $\theta_1, \theta_2, \cdots, \theta_q$ results in different time series patterns. As with autoregressive models, the variance of the error term ϵ_t will only change the scale of the series, not the patterns.

It is possible to write any stationary $AR(p)$ model as an $MA(\infty)$ model. For example, using repeated substitution, we can demonstrate this for an AR(1) model:

$$\begin{aligned}
y_t &= \phi_1 y_{t-1} + \epsilon_t \\
&= \phi_1(\phi_1 y_{t-2} + \epsilon_{t-1}) + \epsilon_t \\
&= \phi_1^2 y_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t \\
&= \phi_1^3 y_{t-3} + \phi_1^2 \epsilon_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t \\
&\text{etc.}
\end{aligned}$$

Provided $-1 < \phi_1 < 1$, the value ϕ_1^k will will get smaller as k gets larger. So eventually we obtain

$$y_t = \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_1^2 \epsilon_{t-2} + \phi_1^3 \epsilon_{t-3} + \cdots$$

an $MA(\infty)$ model.

The reverse result holds if we impose some constraints on the MA parameters. Then the MA model is called invertible. That is, we can write any invertible $MA(q)$ process as an $AR(\infty)$ process. Invertible models are not simply introduced to enable us to convert from MA models to AR models. They also have some desirable mathematical properties.

For example, consider the $MA(1)$ process, $y_t = \epsilon_t + \theta_1 \epsilon_{t-1}$. Its $AR(\infty)$ representation, the most recent error can be written as a linear function of current and past observations:

$$\epsilon_t = \sum_{j=0}^{\infty} (-\theta)^j y_{t-j}$$

where $|\theta| > 1$, the weights increase as lags increase, so the more distant the observations the greater their influence on the current error. When $|\theta| = 1$, the weights are constant in size, and the distant observations have the same influence as the recent observations. As neither of these situations make much sense, we require $|\theta| < 1$, so the most recent observations have higher weight than observations from the more distant past. Thus, the process is invertible when $|\theta| < 1$.

The invertibility constraints for other models are similar to the stationarity constraints.

- For an $MA(1)$ model: $-1 < \theta_1 < 1$
- For an $MA(2)$ model: $-1 < \theta_2 < 1, \theta_2 + \theta_1 > -1, \theta_1 - \theta_2 < 1$.

More complicated conditions hold for $q \geq 3$. Again, R will take care of these constraints when estimating the models.