Forecasting Princples and Practice 2 (8-10)

Koji Mizumura

October 12, 2018(Beg) - (End)

Table of Contents

This document is html\_notebook output (not html\_document).

# Resources

[Forecasting: princples and practice](https://otexts.org/fpp2/)  
[Slides](https://robjhyndman.com/seminars/uwa2017/)  
[Github](https://github.com/nealxun/ForecastingPrinciplePractices)

From the Github, we can install the package fpp2.

# ARIMA models

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 smooothing models are based on the trend and seasonality in the data. ARIMA models aim to describe the autocorrelations in the data.

Before we introduce ARIMA models, we must first discuss the concept of **stationairity** and differencing time series.

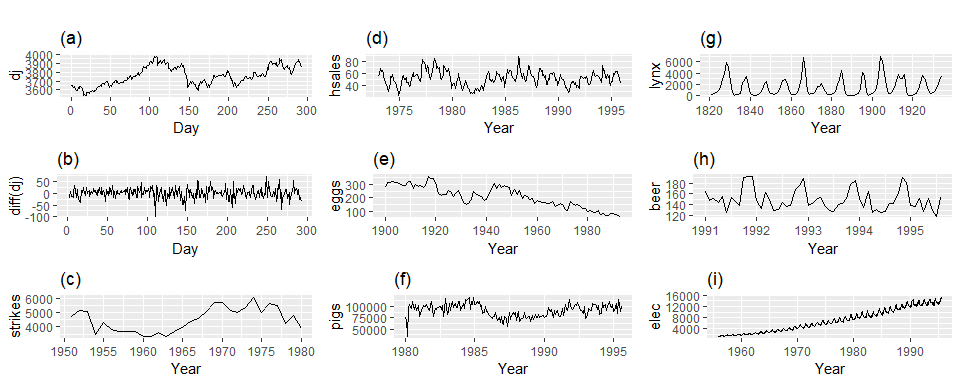
## Stationarity and differencing

A stationary time series is one whose properties do not depend on the time at which the series is observed.

More precisely, if is a *stationary* time series, then for all , the distribution of does not depend on . 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.

Some cases can be confusing — 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.

In general, a stationary time series will have no predictable patterns in the long-term. Time plots will show the series to be roughly horizontal (although some cyclic behaviour is possible), with constant variance.



Which of these series are stationary? (a) Dow Jones index on 292 consecutive days; (b) Daily change in the Dow Jones index on 292 consecutive days; (c) Annual number of strikes in the US; (d) Monthly sales of new one-family houses sold in the US; (e) Annual price of a dozen eggs in the US (constant dollars); (f) Monthly total of pigs slaughtered in Victoria, Australia; (g) Annual total of lynx trapped in the McKenzie River district of north-west Canada; (h) Monthly Australian beer production; (i) Monthly Australian electricity production.

Consider the nine series plotted in Figure @ref(fig:stationary). Which of these do you think are stationary?

Obvious seasonality rules out series (d), (h) and (i). Trend rules out series (a), (c), (e), (f) and (i). Increasing variance also rules out (i). That leaves only (b) and (g) as stationary series.

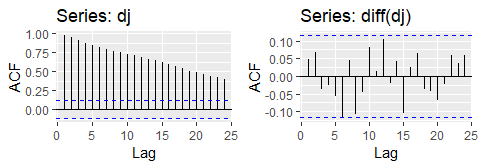
At first glance, the strong cycles in series (g) might appear to make it non-stationary. But these cycles are aperiodic — they are caused when the lynx population becomes too large for the available feed, so that they stop breeding and the population falls to very low numbers, then the regeneration of their food sources allows the population to grow again, and so on. In the long-term, the timing of these cycles is not predictable. Hence the series is stationary.

### Differencing

In Figure @ref(fig:stationary), note that the Dow Jones index was non-stationary in panel (a), but the daily changes were stationary in panel (b). This shows one way to make a 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.

As well as looking at the time plot of the data, the ACF plot is also useful for identifying non-stationary time series. 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 is often large and positive.



The ACF of the Dow-Jones index (left) and of the daily changes in the Dow-Jones index (right).

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

##   
## Box-Ljung test  
##   
## data: diff(dj)  
## X-squared = 14.461, df = 10, p-value = 0.153

The ACF of the differenced Dow-Jones index looks just like that of a white noise series. There is only one autocorrelation lying just outside the 95% limits, and the Ljung-Box statistic has a *p*-value of 0.153 (for ). This suggests that the *daily change* in the Dow-Jones index is essentially a random amount which is uncorrelated with that of previous days.

### Random walk model

The differenced series is the *change* between consecutive observations in the original series, and can be written as

The differenced series will have only values, since it is not possible to calculate a difference for the first observation.

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

Random walk models are very 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. Thus, the random walk model underpins naïve forecasts.

A closely related model allows the differences to have a non-zero mean. Then

The value of is the average of the changes between consecutive observations. If is positive, then the average change is an increase in the value of . Thus, will tend to drift upwards. However, if is negative, will tend to drift downwards.

This is the model behind the drift method discussed in Section @ref(sec-2-methods).

### 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:

In this case, will have 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

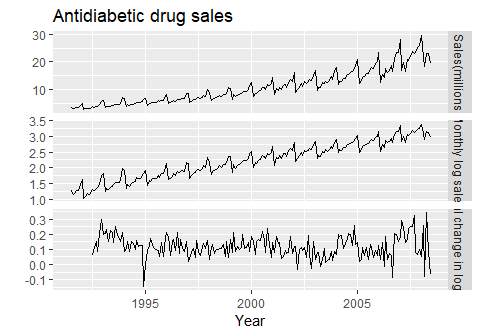
A seasonal difference is the difference between an observation and the corresponding observation from the previous year. So

where the number of seasons. These are also called “lag- differences”, as we subtract the observation after a lag of periods.

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

Forecasts from this model are equal to the last observation from the relevant season. That is, this model gives seasonal naïve forecasts.

cbind("Sales(millions"=a10,  
 "Monthly log sales"=log(a10),  
 "Annual change in log sales"=diff(log(a10),12)) %>%   
 autoplot(facets=TRUE)+  
 labs(x="Year",y="")+  
 ggtitle("Antidiabetic drug sales")



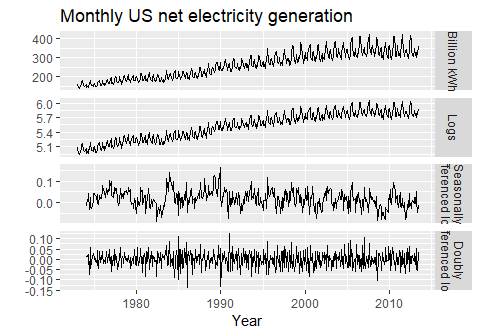
Logs and seasonal differences of the A10 (antidiabetic) sales data. The logarithms stabilize the variance, while the seasonal differences remove the sesonality and trend

Figure @ref(fig:a10diff) shows the seasonal differences of the logarithm of the monthly scripts for A10 (antidiabetic) drugs sold in Australia. The transformation and differencing have made the series look relatively stationary.

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 do both a **seasonal difference** and a **first difference** to obtain **stationary data**, as is shown in Figure @ref(fig:usmelec). Here, the data are first transformed using logarithms (second panel), then seasonal differences are calculated (third panel). The data still seem a little non-stationary, and so a further lot of first differences are computed (bottom panel).

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")



Top panel: US net electricity generation (billion kWh). Other panels show the same data after transforming and differencing.

There is a degree of subjectivity in selecting which differences to apply. The seasonally differenced data in Figure @ref(fig:a10diff) do not show substantially different behaviour from the seasonally differenced data in Figure @ref(fig:usmelec). 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 will be discussed later, but there are always some choices to be made in the modeling process, and different analysts may make different choices.

If denotes a seasonally differenced series, then the twice-differenced series is

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 different unit root tests are available, which are based on different assumptions and may lead to conflicting answers.

#### ADF test

One of the most popular tests is the *Augmented Dickey-Fuller (ADF) test*. For this test, the following regression model is estimated:

where denotes the first-differenced series, , and is the number of lags to include in the regression (often set to be about 3). If the original series, , needs differencing, then the coefficient should be approximately zero. If is already stationary, then . The usual hypothesis tests for regression coefficients do not work when the data are non-stationary, but the test can be carried out using the following R function from the tseries package.

library(tseries)  
adf.test(x, alternative="stationary")

In R, the default value of is set to , where is the length of the time series and means the largest integer not greater than .

The **null-hypothesis** for an ADF test is that the data are **non-stationary**. Thus, large p-values are indicative of non-stationarity, and small p-values suggest stationarity. Using the usual 5% threshold, differencing is required if the p-value is greater than 0.05.

#### KPSS test

Another popular unit root test is the *Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test*. This reverses the hypotheses, so the null hypothesis is that the data are stationary. In this case, small p-values (e.g., less than 0.05) suggest that differencing is required. The kpss.test is also from the tseries package.

library(tseries)  
kpss.test(x)

or ur.kpss function from the urca package.

library(urca)

## Warning: package 'urca' was built under R version 3.5.1

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

he 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(), which uses the measure of seasonal strength introduced in Section 6.7 to determine the appropriate number of seasonal differences required. No seasonal differences are suggested if , 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.

### Biblography

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? Journal of Econometrics, 54(1-3), 159–178. <https://doi.org/10.1016/0304-4076(92)90104-Y>

## Backshift notation

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

(Some references use for “lag” instead of for “backshift”.) In other words, , operating on , has the effect of **shifting the data back one period**. Two applications of to **shifts the data back two periods**:

For monthly data, if we wish to consider “the same month last year,” the notation is = .

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

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

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

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

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

the same result we obtained earlier.

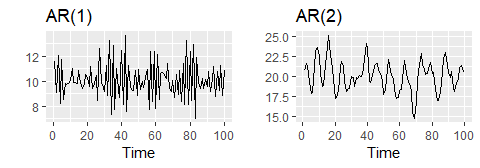
## Autoregressive models

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

Thus, an autoregressive model of order can be written as

where is white noise. This is like a multiple regression but with *lagged values* of as predictors. We refer to this as an **AR() model**.

Autoregressive models are remarkably flexible at handling a wide range of different time series patterns. The two series in Figure 8.5 show series from an and model. Changin the parameters results in different time series patterns. The variance of the error term will only change the scale of the series, not the patterns.



Two examples of data from autoregressive models with different parameters. Left: AR(1) with . Right: AR(2) with . In both cases, is normally distributed white noise with mean zero and variance one.

For an AR(1) moel: - when , is equivalent to white noise; - when and , is equivalent to a random walk; - when and , is equivalent to a random walk with drift; - when , tends to oscillate between positive and negative values;

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

* For an AR(1) model: .
* For an AR(2) model: , ,

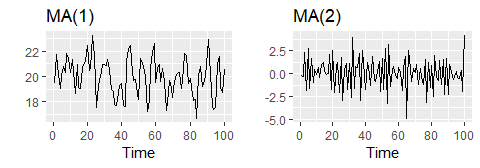
When , the restrictions are much more complicated. R takes care of these restrictions when estimating a model.

## 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.

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

Notice that each value of 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* we discussed in Chapter @ref(ch-decomposition). A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.



Two examples of data from moving average models with different parameters. Left: MA(1) with . Right: MA(2) with . In both cases, is normally distributed white noise with mean zero and variance one.

Figure @ref(fig:maq) shows some data from an MA(1) model and an MA(2) model. Changing the parameters results in different time series patterns. As with autoregressive models, the variance of the error term will only change the scale of the series, not the patterns.

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

Provided , the value of will get smaller as gets larger. So eventually we obtain

an MA() process.

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() process as an AR() process.

**Invertible models** are not simply introduced to enable us to convert from MA models to AR models. They also have some mathematical properties that make them easier to use in practice.

The **invertibility** constraints are similar to the stationarity constraints.

More complicated conditions hold for . Again, R will take care of these constraints when estimating the models.

For example, consider the MA(1) process, . In its AR() representation, the most recent error can be written as a linear function of current and past observations.

$$
\[\varepsilon\_t = \sum\_{j=0}^\infty (-\theta)^j y\_{t-j}.] When (|\theta| > 1)
$$

, the weights increase as lag increases, so the more distant the observation the greater their influence on the current error. When , the weights are constant in size, and the distant observations have the same influence as the recent observations. As nether of these situations make much sense, we require , so most recent observations have higher weight han observations from the more distant past. Thus, the process is invertible when .

The invertibility constrains for other models are similar to the stationary constraints. - For an MA(1) model: () - For an MA(2) model: ()

More complicated conditions hold for q. Again, R will take care of these constrains when estimating the model.

## Non-seasonal ARIMA models

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average model (in this context, “integration” is the reverse of differencing). The full model can be written as

where is the differenced series (it may have been differenced more than once). The “predictors” on the right hand side include both lagged values of and lagged errors. We call this an **ARIMA() model**, where

|  |  |
| --- | --- |
|  | order of the autoregressive part; |
|  | degree of first differencing involved; |
|  | order of the moving average part. |

The same stationarity and invertibility conditions that are used for autoregressive and moving average models also apply to this ARIMA model.

Many of the models we have already discussed are special cases of the ARIMA model, as shown in the following table.

|  |  |
| --- | --- |
| White noise | ARIMA(0,0,0) |
| Random walk | ARIMA(0,1,0) with no constant |
| Random walk with drift | ARIMA(0,1,0) with a constant |
| Autoregression | ARIMA(,0,0) |
| Moving average | ARIMA(0,0,) |

Once we start combining components in this way to form more complicated models, it is much easier to work with the backshift notation. Then equation @ref(eq:8-arima) can be written as

R uses a slightly different parameterization:

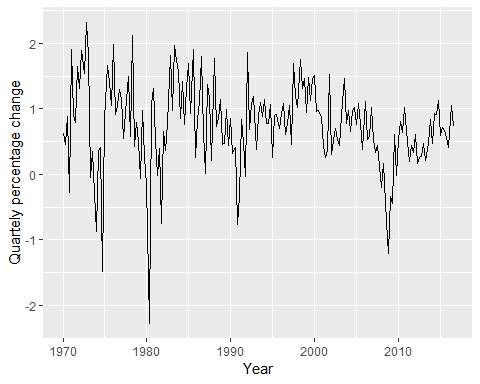
where and is the mean of . To convert to the form given by @ref(eq:arimaB), set .

Selecting appropriate values for , and can be difficult. However, the auto.arima() function in R will do it for you automatically. Later in this chapter, we will learn how the function works, and some methods for choosing these values yourself.

### US consumption expenditure

Figure 8.7 shows quarterly percentage change in US consumption expenditure. Although it is a quarterly serieis, there does not apparent to be a seasonal pattern, so we will fit a non-seasonal ARIMA model.

autoplot(uschange[,"Consumption"])+  
 xlab("Year")+ylab("Quartely percentage change")



The following R code was used to select a model automatically.

(fit <- auto.arima(uschange[,"Consumption"],seasonal=FALSE))

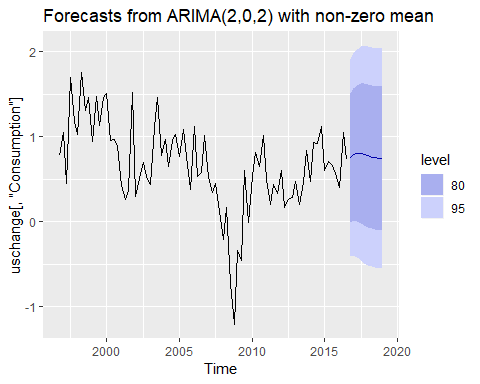
## Series: uschange[, "Consumption"]   
## ARIMA(2,0,2) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ma1 ma2 mean  
## 1.3908 -0.5813 -1.1800 0.5584 0.7463  
## s.e. 0.2553 0.2078 0.2381 0.1403 0.0845  
##   
## sigma^2 estimated as 0.3511: log likelihood=-165.14  
## AIC=342.28 AICc=342.75 BIC=361.67

This is ARIMA(2,0,2) model:

, where c=0.746\*(1-1.391+0.581)=0.142 and is white noise with a standard deviation of .

Forecast from the model are shown in Figure 8.8.

fit %>% forecast(h=10) %>% autoplot(include=80)



### Understanding ARIMA models

The auto.arima() function is useful, but anything can be a little dangerous, and it is worth understanding something of the behaviour of the models even whem yo urely on an automatic procedure to choose the model for you.

The constant c has an important effect on the long-term forecasts obtained from these models. - If and , the long-term forecasts will go to zero. - If and , the long-term forecasts will go to a non-zero constant. - If and , the long-term forecasts will follow a straight line. - If and , the long-term forecasts will go to the mean of the data. - If and , the long-term forecasts will follow a straight line. - If and , the long-term forecasts will follow a quadratic trend.

The value of also has an effect on the prediction intervals - the higher the vale of , the more rapidly the prediction intervals increase in size. For , tlhe long-term forecast standard deviation will got to the standard deviation of the historical data, so the prediction inteval will all be essentially the same.

This behaviour is seen in Figure @ref(fig:usconsumptionf) where and . In this figure, the prediction intervals are almost the same for the last few forecast horizons, and the point forecasts are equal to the mean of the data.

The value of is important if the data show cycles. To obtain cyclic forecasts, it is necessary to have , along with some additional conditions on the parameters. For an AR(2) model, cyclic behaviour occurs if . In that case, the average period of the cycles is [[1]](#footnote-50)

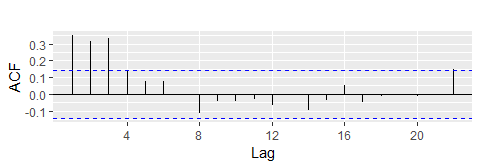
### ACF and PACF plots It is usually not possible to tell, simply from a time plot, what values of and are appropriate for the data. However, it is sometimes possible to use the ACF plot, and the closely related PACF plot, to determine appropriate values for and .

Recall that an ACF plot shows the autocorrelations which measure the relationship between and for different values of . Now if and are correlated, then and must also be correlated. However, then and might be correlated, simply because they are both connected to , rather than because of any new information contained in that could be used in forecasting .

To overcome this problem, we can use **partial autocorrelations**. These measure the relationship between and after removing the effects of lags . So the first partial autocorrelation is identical to the first autocorrelation, because there is nothing between them to remove. Each partial autocorrelation can be estimated as the last coefficient in an autoregressive model. Specifically, , the th partial autocorrelation coefficient, is equal to the estimate of in an AR() model. In practice, there are more efficient algorithms for computing than fitting all of these autoregressions, but they give the same results.

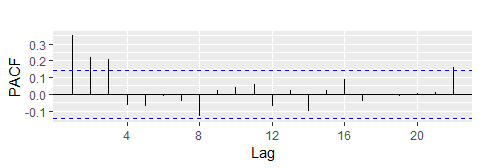
Figures @ref(fig:usconsumptionacf) and @ref(fig:usconsumptionpacf) shows the ACF and PACF plots for the US consumption data shown in Figure @ref(fig:usconsumption). The partial autocorrelations have the same critical values of as for ordinary autocorrelations, and these are typically shown on the plot as in Figure @ref(fig:usconsumptionacf).

ggAcf(uschange[,"Consumption"],main="")



ACF quarterly percentage change in US consumption. A convenient way to produce a time plot, ACF plot and PACE plot in one command is to use the ggtsdiplay function in R

ggPacf(uschange[,"Consumption"],main="")



PACF of quarterly percentage change in US consumption.

If the data are from an ARIMA(, , 0) or ARIMA(0,,) model, then the ACF and PACF plots can be helpful in determining the value of or . If and are both positive, then the plots do not help in finding suitable values of and .

The data may follow an ARIMA(,, 0) model if the ACF and PACF plots of the differenced data show the following patterns:

* the ACF is exponentially decaying or sinusoidal;
* there is a significant spike at lag in the PACF, but none beyond lag .

The data may follow an ARIMA(0,,) model if the ACF and PACF plots of the differenced data show the following patterns:

* the PACF is exponentially decaying or sinusoidal;
* there is a significant spike at lag in the ACF, but none beyond lag .

In Figure @ref(fig:usconsumptionacf), we see that there are three spikes in the ACF, followed by an almost significant spike at lag 4. In the PACF, there are three significant spikes, and then no significant spikes thereafter (apart from one just outside the bounds at lag 22). We can ignore one significant spike in each plot if it is just outside the limits, and not in the first few lags. After all, the probability of a spike being significant by chance is about one in twenty, and we are plotting 22 spikes in each plot. The pattern in the first three spikes is what we would expect from an ARIMA(3,0,0), as the PACF tends to decrease. So in this case, the ACF and PACF lead us to think an **ARIMA(3,0,0)** model might be appropriate.

(fit2 <- Arima(uschange[,"Consumption"], order=c(3,0,0)))

## Series: uschange[, "Consumption"]   
## ARIMA(3,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 mean  
## 0.2274 0.1604 0.2027 0.7449  
## s.e. 0.0713 0.0723 0.0712 0.1029  
##   
## sigma^2 estimated as 0.3494: log likelihood=-165.17  
## AIC=340.34 AICc=340.67 BIC=356.5

This model is actually slightly better than the model identified by auto.arima() (with an AICc value of 350.67 compared with 342.75), The auto.arima() function did not find this model because it does not consider all possible models in its search. You can make it work harder by using the arguments stepwise=FALSE and approximation=FALSE.

(fit3 <- auto.arima(uschange[,"Consumption"],seasonal=FALSE,  
 stepwise = FALSE, approximation = FALSE))

## Series: uschange[, "Consumption"]   
## ARIMA(3,0,0) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ar3 mean  
## 0.2274 0.1604 0.2027 0.7449  
## s.e. 0.0713 0.0723 0.0712 0.1029  
##   
## sigma^2 estimated as 0.3494: log likelihood=-165.17  
## AIC=340.34 AICc=340.67 BIC=356.5

We also use the argument seasonal=FALSE to prevent it searching for seasonal ARIMA models; we will consider these models in Section 8.9.

This time, auto.arima() has found the same model that we guessed from the ACF and PACF plots. The forecasts from this ARIMA(3,0,0) model are almost identical to those shown in Figure 8.8 for the ARIMA(2,0,2) model, so we do not produce the plot here.

## Estimation and order selection

## Maximum likelihood estimation

Once the model order has been identified (i.e., the values of , and ), we need to estimate the parameters , , . When R estimates the ARIMA model, it uses *maximum likelihood estimation* (MLE). This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed. For ARIMA models, MLE is very similar to the *least squares* estimates that would be obtained by minimizing

For the regression models considered in Chapter @ref(ch-regression), MLE gives exactly the same parameter estimates as least squares estimation.) Note that ARIMA models are much more complicated to estimate than regression models, and different software will give slightly different answers as they use different methods of estimation, and different optimization algorithms.

In practice, R will report the value of the *log likelihood* of the data; that is, the logarithm of the probability of the observed data coming from the estimated model. For given values of , and , R will try to maximize the log likelihood when finding parameter estimates.

### Information criteria

Akaike’s Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. It can be written as

where is the likelihood of the data, if and if . Note that the last term in parentheses is the number of parameters in the model (including , the variance of the residuals).

For ARIMA models, the corrected AIC can be written as

and the Bayesian Information Criterion can be written as

Good models are obtained by minimizing the AIC, AICc or BIC. Our preference is to use the AICc.

It is important to note that these information criteria tend not to be good guides to selectingn the appropriate order of differencing () of a model, but only for selecting the values of and . This is because the differencing changes the data on which the likelihood is computed, making the AIC values between models with different orders of differencing not comparable. So we need to use some other approach to choose , and then we can use the AICc to select and .

## ARIMA modelling in R

### How does auto.arima work?

The auto.arima() function in R uses a variation of the Hyndman-Khandakar algorithm [@HK08], which combines unit root tests, minimization of the AICc and MLE to obtain an ARIMA model. The algorithm follows these steps.

\*\* Hyndman-Khandakar algorithm for automatic ARIMA modelling\*\*

1. The number of differences is determined using repeated KPSS tests.
2. The values of and are then chosen by minimizing the AICc after differencing the data times. Rather than considering every possible combination of and , the algorithm uses a stepwise search to traverse the model space.
   1. Four initial models are fitted:
      * ARIMA,
      * ARIMA,
      * ARIMA,
      * ARIMA. A constant is included unless . If , an additional model
      * ARIMA without a constant is also fitted.
   2. The best model (with the smallest AICc value) fitted in step (a) is set to be the “current model”.
   3. Variations on the current model are considered:
      * vary and/or from the current model by ;
      * include/exclude from the current model. The best model considered so far (either the current model or one of these variations) becomes the new current model.
   4. Repeat Step 2(c) until no lower AICc can be found.

ARIMA(p,d,q) model, where - p= order of the autoregressive part; - d= degree of first differencing involved; - q= order of the moving average part.

The argument to auto.arima() provide for many variations on the algorithm. What is described here is the default behavior.

The default procedure uses some approximations to speed up the search. These approximations can be avoided with the argument approximation=FALSE. It is possible that the minimum AICc model will not be found due to these approximations, or because of the use of a stepwise procedure. A much larger set of models will be searched if the argment stepwise=FALSE is used. See the help file for a full description of the arguments.

### Choosing your own model

If you want to choose the model yourself, use the Arima() function in R. There is another function arima() in R which also fits an ARIMA model. However, it does not allow for the constant unless , and it does not return everything required for other functions in the forecast package to work. Finally, it does not allow the estimated model to be applied to new data (which is useful for checking forecast accuracy). Consequently, it is recommended that Arima() be used instead.

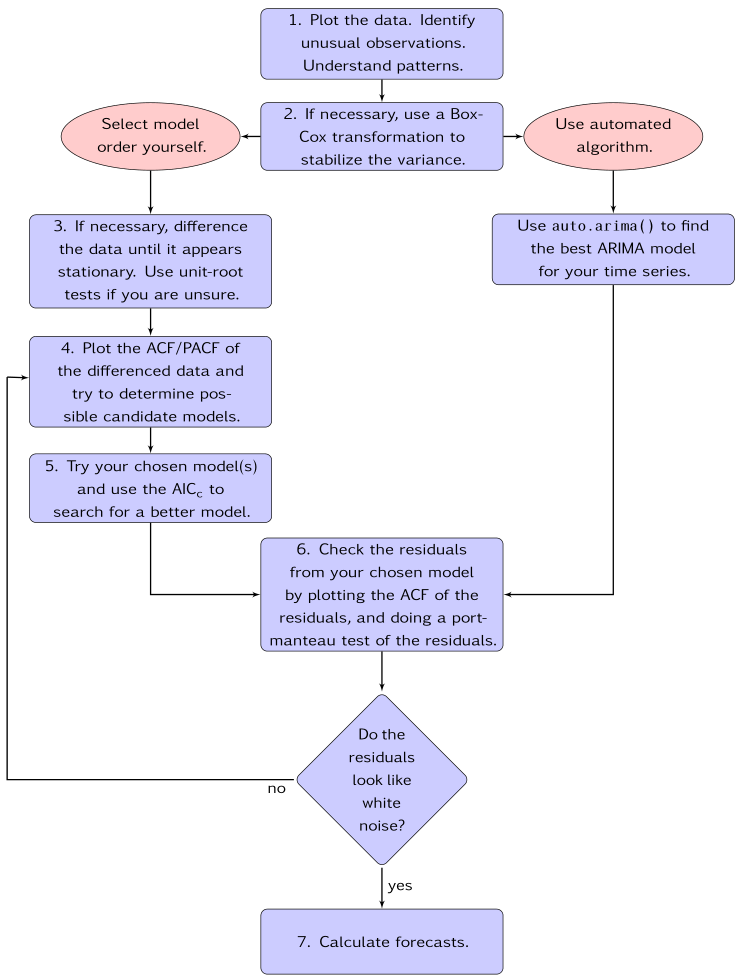
### Modeling procedure

When fitting an ARIMA model to a set of (non-seasonal) time series data, the following procedure provides a useful general approach.

1. Plot the data and identify any usual observations
2. If necessary, transform the data (using the Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary, take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an ARIMA() or ARIMA() model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

The automated algorithm only takes care of step 3–5. So even if you use it, you will still need to take care of the other steps yourself.

The process is summarized as follows.

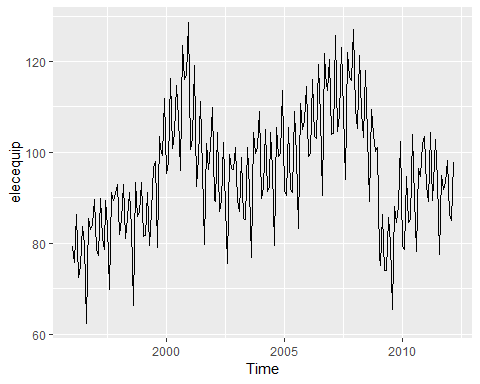


General process for forecasting using an ARIMA model.

### Example: Seasonal adjusted electrical equipment orders

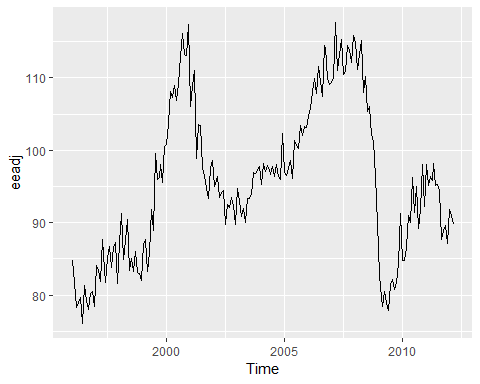
We will apply this procedure to the seasonally adjusted electrical equipment orders data shown in Figure @ref(fig:ee1).

autoplot(elecequip)



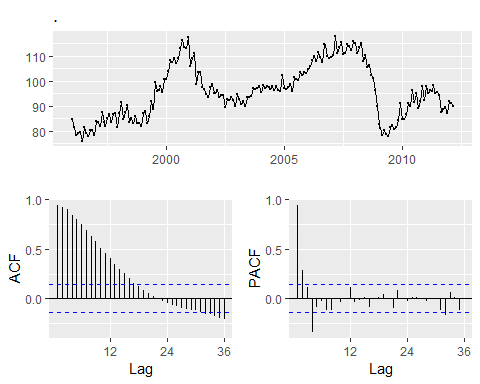
Seasonally adjusted electrical equipment orders index in the Euro area.

elecequip %>% stl(s.window="periodic") %>% seasadj() -> eeadj  
autoplot(eeadj)



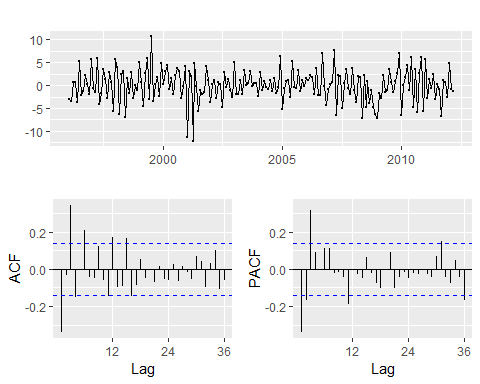
Seasonally adjusted electrical equipment orders index in the Euro area.

eeadj %>% ggtsdisplay()



Seasonally adjusted electrical equipment orders index in the Euro area.

eeadj %>% diff() %>% ggtsdisplay(main="")



Seasonally adjusted electrical equipment orders index in the Euro area.

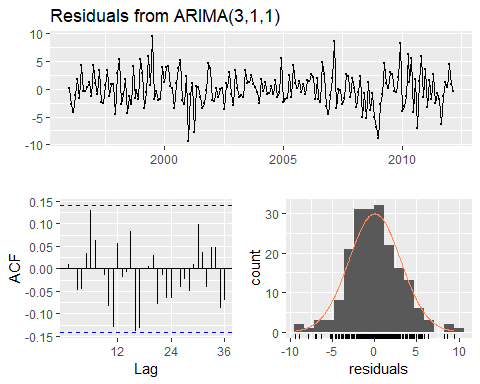
1. The time plot show some sudden changes, particularly the big drop in 2008/2009. These changes are due to the global economic environment. Otherwise, there is nothing usual about the time plot and there appear to be no need to do any data adjustments.
2. There is no evidence of **changing variance**, so we will not do a Box-Cox transformation.
3. The data are clearly non-stationary, as the sereies wanders up and fown for long periods. Consequently, we will take a first difference of the data. The differenced data are shown in Figure @ref(fig:ee2).These look stationary, and so we will not consider further differences.
4. The PACF shown in Figure @ref(fig:ee2) is suggestive of an **AR(3) model**. So an initial candidate model is an ARIMA(3,1,0). There are no other obvious candidate models.
5. We fit an ARIMA(3,1,0) model along with variations including ARIMA(4,1,0), ARIMA(2,1,0), ARIMA(3,1,1), etc. Of these, the ARIMA(3,1,1) has a slightly smaller AICc value.

fit <- Arima(eeadj, order=c(3,1,1))  
summary(fit)

## Series: eeadj   
## ARIMA(3,1,1)   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1  
## 0.0044 0.0916 0.3698 -0.3921  
## s.e. 0.2201 0.0984 0.0669 0.2426  
##   
## sigma^2 estimated as 9.577: log likelihood=-492.69  
## AIC=995.38 AICc=995.7 BIC=1011.72  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.03288179 3.054718 2.357169 -0.00647009 2.481603 0.28844  
## ACF1  
## Training set 0.008980578

1. The ACF plot of the residuals from the ARIMA(3.1,1) model shows that all correlations are within the threshold limits, indicating that the residuals are behaving like white noise. A portmanteu test returns a large p-value, also suggesting that the residuals are white noise.

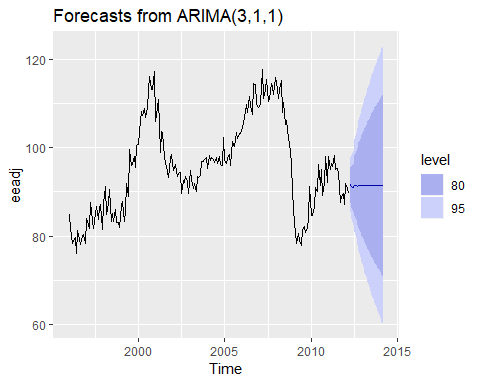
checkresiduals(fit)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(3,1,1)  
## Q\* = 24.034, df = 20, p-value = 0.2409  
##   
## Model df: 4. Total lags used: 24

7．The ACF plot of the residuals from the **ARIMA(3,1,1)** model shows that all correlations are within the threshold limits, indicating that the residuals are behaving like white noise. A port

autoplot(forecast(fit))



Forecasts for the seasonally adjusted electrical orders index.

IF we had used the automated algorithm instead, we would have obtained an ARIMA(3,1,0) model using the default settings, but the ARIMA

### Understanding constants in R

A non-seasonal ARIMA model can be written as

or equivalently as

Thus, the inclusion of a constant in a non-stationary ARIMA model is equivalent to inducing a polynomial trend of order in the forecast function. (If the constant is omitted, the forecast function includes a polynomial trend of order .) When , we have the special case that is the mean of .

By default, the Arima() function sets when and provides an estimate of when . It will be close to the sample mean of the time series, but usually not identical to it as the sample mean is not the maximum likelihood estimate when .

The argument include.mean only has an effect when and is TRUE by default. Setting include.mean=FALSE will force .

The argument include.drift allows when . For , no constant is allowed as a quadratic or higher order trend is particularly dangerous when forecasting. The parameter is called the “drift” in the R output when .

There is also an argument include.constant which, if TRUE, will set include.mean=TRUE if and include.drift=TRUE when . If include.constant=FALSE, both include.mean and include.drift will be set to FALSE. If include.constant is used, the values of include.mean=TRUE and include.drift=TRUE are ignored.

The auto.arima() function automates the inclusion of a constant. By default, for or , a constant will be included if it improves the AICc value; for the constant is always omitted. If allowdrift=FALSE is specified, then the constant is only allowed when .

### Plotting the characteristic roots

*(This is a more advanced section and can be skipped if desired.)*

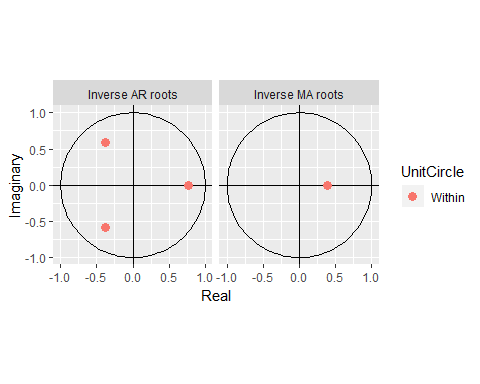
We can re-write equation @ref(eq:c) as

where is a th order polynomial in and is a th order polynomial in .

The stationarity conditions for the model are that the complex roots of lie outside the unit circle, and the invertibility conditions are that the complex roots of lie outside the unit circle. So we can see whether the model is close to invertibility or stationarity by a plot of the roots in relation to the complex unit circle.

It is easier to plot the inverse roots instead, as they should all lie *within* the unit circle. This is easily done in R. For the ARIMA(3,1,1) model fitted to the seasonally adjusted electrical equipment index, we obtain the following plot.

autoplot(fit)



The three red dots in the left hand plot correspond to the roots of the polynomials , while the red dot in the right hand plot corresponds to the root of . They are all inside the unit circle, as we would expect because R ensures the fitted model is both stationary and invertible. Any roots close to the unit circle may be numerically unstable, and the corresponding model will not be good for forecasting.

The Arima function will never return a model with inverse roots outside the unit circle. The auto.arima function is even stricter and will not select a model with roots close to the unit circle either.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27(1), 1–22. <https://doi.org/10.18637/jss.v027.i03>

## Forecasting

### Point forecasts

Although we have calculated forecasts from the ARIMA models in our examples, we have not yet explained how they are obtained. Point forecasts can be calculated using the following three steps.

1. Expand the ARIMA equation so that is on the left hand side and all other terms are on the right
2. Rewrite the equation by replacing with
3. On the right hand side of the equation, replace future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals.

Beginning with , these steps are repeated for until all forecasts have been caluculated. The procedure is most easily understood via an example. We will illustrate using the model fitted in the previous section. The model can be written as follows.

The procedure is most easily understood via an example. We will illustrate it using the ARIMA(3,1,1) model fitted in the previous section. The model can be written as follows:

where , , and . Then we expand the left hand side to obtain

and applying the backshift operator gives

Finally, we move all terms other than to the right hand side:

This completes the first step. While the equation now looks like an ARIMA(4,0,1), it is still the same ARIMA(3,1,1) model we started with. It cannot be considered an ARIMA(4,0,1) because the coefficients do not satisfy the stationarity conditions.

For the second step, we replace with in @ref(eq:arima301f):

Assuming we have observations up to time , all values on the right hand side are known except for , which we replace with zero, and , which we replace with the last observed residual :

A forecast of is obtained by replacing with in @ref(eq:arima301f) . All values on the right hand side will be known at time except which we replace with , and and , both of which we replace with zero:

The process continues in this manner for all future time periods. In this way, any number of point forecasts can be obtained.

### Prediction intervals

The calculation of ARIMA prediction intervals is more difficult, and the details are largely beyond the scope of this book. We will only give some simple examples.

The first prediction interval is easy to calculate. If is the standard deviation of the residuals, then a 95% prediction interval is given by . This result is true for all ARIMA models regardless of their parameters and orders.

Multi-step prediction intervals for ARIMA(0,0,) models are relatively easy to calculate. We can write the model as

Then, the estimated forecast variance can be written as

and a 95% prediction interval is given by .

In Section @ref(sec:mamodels), we showed that an AR(1) model can be written as an MA() model. Using this equivalence, the above result for MA() models can also be used to obtain prediction intervals for AR(1) models.

More general results, and other special cases of multi-step prediction intervals for an ARIMA(,,) model, are given in more advanced textbooks such as @BDbook16.

The prediction intervals for ARIMA models are based on assumptions that the residuals are uncorrelated and normally distributed. If either of these assumptions does not hold, then the prediction intervals may be incorrect. For this reason, always plot the ACF and histogram of the residuals to check the assumptions before producing prediction intervals.

In general, prediction intervals from ARIMA models increase as the forecast horizon increases. For stationary models (i.e., with ) they will converge, so that prediction intervals for long horizons are all essentially the same. For , the prediction intervals will continue to grow into the future.

As with most prediction interval calculations, ARIMA-based intervals tend to be too narrow. This occurs because only the variation in the errors has been accounted for. There is also variation in the parameter estimates, and in the model order, that has not been included in the calculation. In addition, the calculation assumes that the historical patterns that have been modelled will continue into the forecast period.

### Bibliography

Bibliography Brockwell, P. J., & Davis, R. A. (2016). Introduction to time series and forecasting (3rd ed). New York, USA: Springer. [Amazon]

## Seasonal ARIMA models

So far, we have restricted our attention to non-seasonal data and non-seasonal ARIMA models. However, ARIMA models are also capable of modelling a wide range of seasonal data.

A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

|  |  |  |
| --- | --- | --- |
| ARIMA |  |  |
|  |  |  |
|  | Non-seasonal part | Seasonal part of |
|  | of the model | of the model |

where number of observations per year. We use uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model.

The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but involve backshifts of the seasonal period. For example, an ARIMA(1,1,1)(1,1,1) model (without a constant) is for quarterly data (), and can be written as

The additional seasonal terms are simply multiplied by the non-seasonal terms. ### ACF/PACF The seasonal part of an AR or MA model will be seen in the seasonal lags of the PACF and ACF. For example, an ARIMA(0,0,0)(0,0,1) $\_{12$ model will show:

* a spike at lag 12 in the ACF but no other significant spikes;
* exponential decay in the seasonal lags of the PACF (i.e., at lags 12, 24, 36, …).

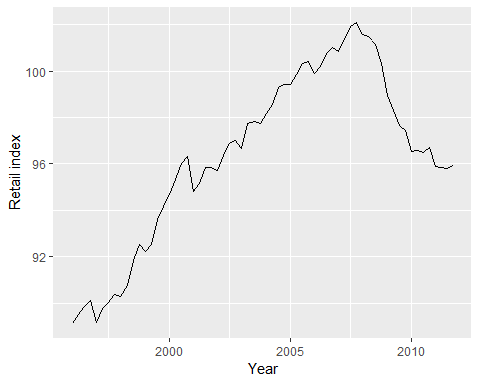
In considering the appropriate seasonal orders for a seasonal ARIMA model, restrict attention to the seasonal lags. and

The modelling procedure is almost the same as for non-seasonal data, except that we need to select seasonal AR and MA terms as well as the non-seasonal components of the model. The process is best illustrated via examples.

### Example; European quarterly retail trade

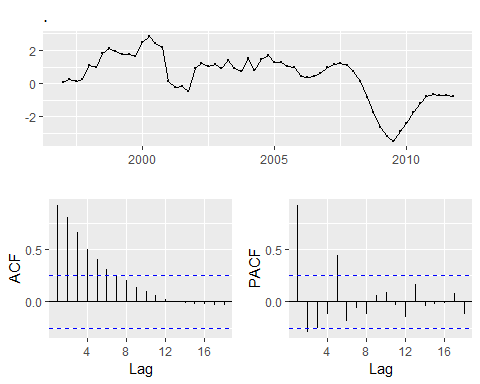
We will describe the seasonal ARIMA modelling procedure using quarterly European retail trade data from 1996 to 2011. The data are plotted in Figure 8.17.

autoplot(euretail)+ylab("Retail index")+xlab("Year")

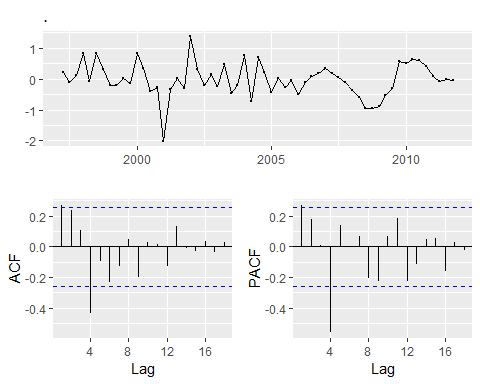


The data are clearly non-stationary, with some seasonality, so we will first take a seasonal difference. The seasonally differenced data are shown in Figure 8.18. These also appear to be non-stationary, so we take an additional first difference, shown in Figure 8.19.

euretail %>% diff(lag=4) %>% ggtsdisplay()

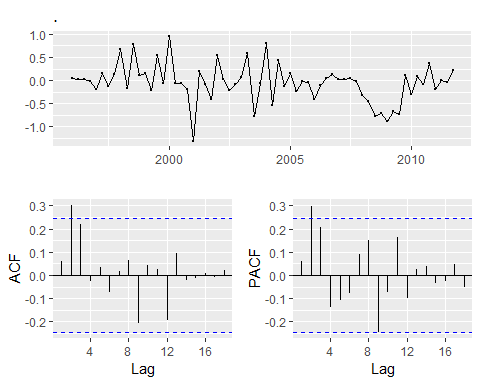


euretail %>% diff(lag=4) %>% diff() %>% ggtsdisplay()



Our aim now is to find an approproate ARIMA model based on the ACF and PACF shown in Figure 8.19. The significant spike at lag 1 in the ACF suggests a non-seasonal MA(1) component, and the significant spike at lag4 in the ACF suggests a seasonal Ma(1) component. Consequently, we begin with an ARIMA(0,1,1)(0,1,1) model, indicating a first and seasonal difference, and non-seasonal and seasonal MA(!) components. The residuals for the fitted model are shown in Figure 8.20 (By analogous logi applied to the PCAF, we could also have started with an model)

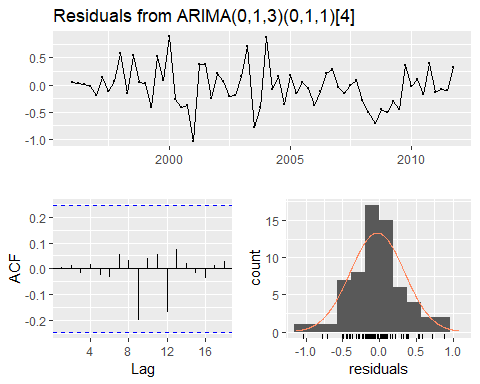
euretail %>%   
 Arima(order=c(0,1,1),seasonal=c(0,1,1)) %>%   
 residuals %>%   
 ggtsdisplay



Residuals from the fitted ARIMA(0,1,3)(0,1,1)model for the European retail trade index data

Both the ACF and PACF show significant spikes at lag 2, and almost significant spikes at lag 3, indicating that some additional non-seasonal terms need to be included in the model. the AICc of the ARIMA(0,1,2)(0,1,1) model is 74.36, while that for the ARIMA(0,1,3)(0,1,1) model is 68.53. We tried other models with AR terms as well, but none that gave a smaller AICc value. Consequently, we choose the ARIMA(0,1,3)(0,1,1) model. Its residuals are plotted in Figure @ref(fig:euretail5). All the spikes are now within the significance limits, so the residuals appear to be white noise. The **Ljung-Box test** also shows that the residuals have no remaining autocorrelations.

fit3 <- arima(euretail, order=c(0,1,3), seasonal=c(0,1,1))  
checkresiduals(fit3)



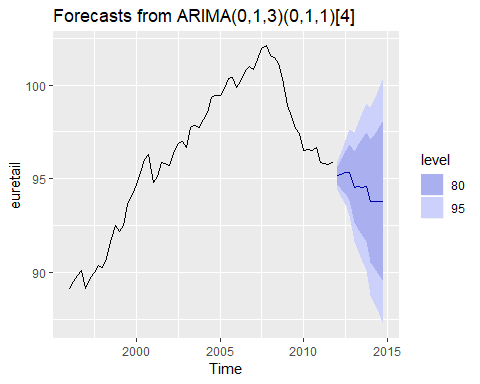
Residuals from the fitted ARIMA(0,1,3)(0,1,1) model for the European retail trade index data.

##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,3)(0,1,1)[4]  
## Q\* = 0.51128, df = 4, p-value = 0.9724  
##   
## Model df: 4. Total lags used: 8

Thus, we now have aseasonal ARIMA model that passes the required checks and is ready for forecasting. Forecasts from the model for the next three years are shown in Figure @ref(fig:euretail6).

The forecasts follow the recent trend in the data, because of the double differencing. The large and rapidly increasing prediction intervals show that the retail trade index could start increasing or decreasing at any time — while the point forecasts trend downwards, the prediction intervals allow for the data to trend upwards during the forecast period.

fit3 %>% forecast(h=12) %>% autoplot()



Forecasts of the European retail trade index data using the ARIMA(0,1,3)(0,1,1) model. 80% and 95% prediction intervals are shown.

We could have used auto.arima() to do most of this work for us. It would have given the following result.

auto.arima(euretail)

## Series: euretail   
## ARIMA(1,1,2)(0,1,1)[4]   
##   
## Coefficients:  
## ar1 ma1 ma2 sma1  
## 0.7362 -0.4663 0.2163 -0.8433  
## s.e. 0.2243 0.1990 0.2101 0.1876  
##   
## sigma^2 estimated as 0.1587: log likelihood=-29.62  
## AIC=69.24 AICc=70.38 BIC=79.63

Notice that it has selected a different model (with a larger AICc value). auto.arima() takes some short-cuts in order to speed up the computation, and will not always give the best model. The short-cuts can be turned off, and then it will sometimes return a different model.

auto.arima(euretail, stepwise = FALSE, approximation = FALSE)

## Series: euretail   
## ARIMA(0,1,3)(0,1,1)[4]   
##   
## Coefficients:  
## ma1 ma2 ma3 sma1  
## 0.2630 0.3694 0.4200 -0.6636  
## s.e. 0.1237 0.1255 0.1294 0.1545  
##   
## sigma^2 estimated as 0.156: log likelihood=-28.63  
## AIC=67.26 AICc=68.39 BIC=77.65

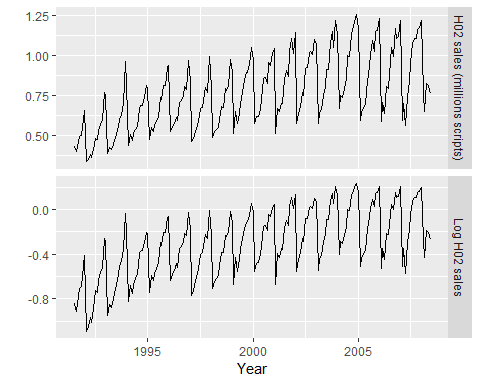
This time, it returned the same model we had identified.

The auto.arima() function uses nsdiffs() to determine (the number of seasonal differences to use) and ndiffs() (the number of ordinary differences to use). The selection of the other model parameters () are all determined by minimizing the AICc, as with non-seasonal ARIMA models.

### Example: Corticosteroid drug sales in Australia

Our second example is more difficult. We will try to forecast monthly corticorsteriod sales in Australia. These are known as H02drugs under the Anatomical Therapeutic Chemical Classification scheme.

lh02 <- log(h02)  
cbind("H02 sales (millions scripts)"=h02,  
 "Log H02 sales"=lh02) %>%   
 autoplot(facets=T)+xlab("Year")+ylab("")



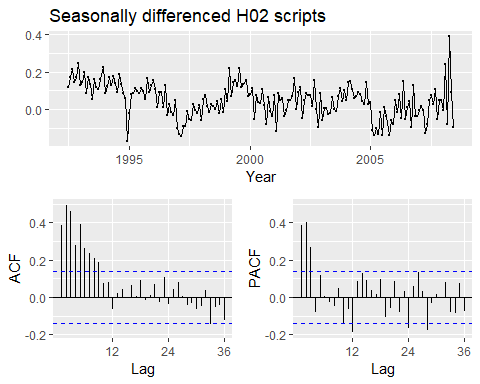
Cortecosteroid drug sales in Australia (in millions of scripts per month). Logged data shown in bottom panel.

Data from July 1991 to June 2009 are plotted in Figure @ref(fig:h02). There is a small increase in the variance with the level, so we take logarithms to stabilize the variance.

The data are strongly seasonal and obviously non-stationary, so seasonal differencing will be used. The seasonally differenced data are shown in Figure @ref(fig:h02b). It is not clear at this point whether we should do another difference or not. We decide not to, but the choice is not obvious.

The last few observations appear to be different (more variable) from the earlier data. This may be due to the fact that data are sometimes revised when earlier sales are reported late.

lh02 %>% diff(lag=12) %>%   
 ggtsdisplay(xlab="Year",main="Seasonally differenced H02 scripts")



Seasonall differenced cortecosteroid drug sales in Australia (in millions of scripts per month)

In the plots of the seasonally differenced data, there are spikes in the PACF at lags 12 and 24, but nothing at seasonal lags in the ACF. This may be suggestive of a seasonal AR(2) term. In the non-seasonal lags, there are three significant spikes in the PACF, suggesting a possible AR(3) term. The pattern in the ACF is not indicative of any simple model.

Consequently, this initial analysis suggests that a possible model for these data is an ARIMA(3,0,0)(2,1,0). We fit this model, along with some variations on it, and compute the AICc values shown in the following table.

|  |  |
| --- | --- |
| Model | AICc |
| ARIMA(3,0,1)(0,1,2) | -485.48 |
| ARIMA(3,0,1)(1,1,1) | -484.25 |
| ARIMA(3,0,1)(0,1,1) | -483.67 |
| ARIMA(3,0,1)(2,1,0) | -476.31 |
| ARIMA(3,0,0)(2,1,0) | -475.12 |
| ARIMA(3,0,2)(2,1,0) | -474.88 |
| ARIMA(3,0,1)(1,1,0) | -463.40 |

Of these models, the best is the ARIMA(3,0,1)(0,1,2) model (i.e., it has the smallest AICc value).

(fit <- Arima(h02, order=c(3,0,1), seasonal=c(0,1,2), lambda=0))

## Series: h02   
## ARIMA(3,0,1)(0,1,2)[12]   
## Box Cox transformation: lambda= 0   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 sma1 sma2  
## -0.1603 0.5481 0.5678 0.3827 -0.5222 -0.1768  
## s.e. 0.1636 0.0878 0.0942 0.1895 0.0861 0.0872  
##   
## sigma^2 estimated as 0.004278: log likelihood=250.04  
## AIC=-486.08 AICc=-485.48 BIC=-463.28

The residuals from this model are shown in Figure @ref(fig:h02res). There are a few significant spikes in the ACF, and the model fails the **Ljung-Box test**. The model can still be used for forecasting, but the prediction intervals may not be accurate due to the correlated residuals.

Next, we will try using the automatic ARIMA algorithm. Running auto.arima() with all arguments left at their default values led to an ARIMA(2,1,3)(0,1,1) model. However, the model still fails the Ljung-Box test. Sometimes it is just not possible to find a model that passes all of the tests.

#### Test set evaluation:

We will compare some of the models fitted so far using a test set consisting of the last two years of data. Thus, we fit the models using data from July 1991 to June 2006, and forecast the script sales for July 2006 – June 2008. The results are summarised in the following table.

|  |  |
| --- | --- |
| Model | RMSE |
| ARIMA(3,0,1)(0,1,2) | 0.0622 |
| ARIMA(3,0,1)(1,1,1) | 0.0630 |
| ARIMA(2,1,4)(0,1,1) | 0.0632 |
| ARIMA(2,1,3)(0,1,1) | 0.0634 |
| ARIMA(3,0,3)(0,1,1) | 0.0640 |
| ARIMA(2,1,5)(0,1,1) | 0.0640 |
| ARIMA(3,0,1)(0,1,1) | 0.0644 |
| ARIMA(3,0,2)(0,1,1) | 0.0644 |
| ARIMA(3,0,2)(2,1,0) | 0.0645 |
| ARIMA(3,0,1)(2,1,0) | 0.0646 |
| ARIMA(4,0,2)(0,1,1) | 0.0648 |
| ARIMA(4,0,3)(0,1,1) | 0.0648 |
| ARIMA(3,0,0)(2,1,0) | 0.0661 |
| ARIMA(3,0,1)(1,1,0) | 0.0679 |

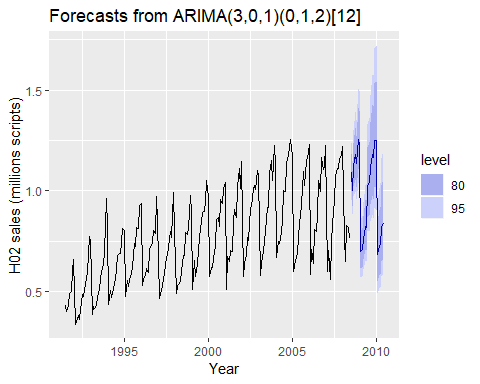
The models chosen manually and with auto.arima() are both in the top four models based on their RMSE values.

When models are compared using AIcc values, it is important that all models have the same orders of differencing. However, when comparing models using a test set, it does not matter how the forecasts were produced — the comparisons are always valid. Consequently, in the table above, we can include some models with only seasonal differencing and some models with both first and seasonal differencing, while in the earlier table containing AICc values, we only compared models with seasonal differencing but no first differencing.

None of the models is considered here pass all of the residual tests.In practice, we would normally use the best model we could find, even if it did not pass all of the tests.

Forecasts from the RIMA(3,0,1)(0,1,2) model (which has the lowest RMSE value on the test set, and the best AICc value amongst models with only seasonal differencing) are shown in Figure @ref(fig:h02f).

h02 %>%   
 Arima(order=c(3,0,1), seasonal=c(0,1,2), lambda = 0) %>%   
 forecast() %>%   
 autoplot()+  
 ylab("H02 sales (millions scripts)")+xlab("Year")



Forecasts from the ARIMA(3,0,1)(0,1,2) model applied to the H02 monthly script sales data.

## ARIMA vs ETS

It is commonly held myth that ARIMA models are more general than **exponential smoothing**. While linear exponential smoothing models are all *special cases* of ARIMA models, the non-linear exponential smoothing models have no equivalent ARIMA couterparts.

On the other hand, there are also many ARIMA models that have no exponential smoothing counterparts. In particular, all ETS models are non-stationary, while some ARIMA models are stationary.

The following table gives the equivalence relationships for the two classes of models.

|  |  |  |
| --- | --- | --- |
| **ETS model** | **ARIMA model** | **Parameters** |
| ETS(A,N,N) | ARIMA(0,1,1) |  |
| ETS(A,A,N) | ARIMA(0,2,2) |  |
|  |  |  |
| ETS(A,A,N) | ARIMA(1,1,2) |  |
|  |  |  |
|  |  |  |
| ETS(A,N,A) | ARIMA(0,0,)(0,1,0) |  |
| ETS(A,A,A) | ARIMA(0,1,)(0,1,0) |  |
| ETS(A,A,A) | ARIMA(1,0,)(0,1,0) |  |

For the seasonal models, the ARIMA parameters have a large number of restrictions.

The AICc is useful for selecting between models in the same class. For example, we can use it to select an ARIMA model between candiate ARIMA models[[2]](#footnote-90) or an ETS model between candidate ETS models. However, it cannot be used to compare between ETS and ARIMA models because they are in different model classes, and the likelihood is computed in different ways. The examples below demonstrate selecting between these classes of models.

### Example: Comparing auto.arima() and ets() on non-seasonal data

We can use time series cross-validation to compare an ARIMA model and an ETS model. The code below provides functions that return forecast objects from auto.arima() and ets() respectively.

fets <- function(X,h){  
 forecast(ets(x), h = h)  
}  
  
farima <- function(x, h) {  
 forecast(auto.arima(x), h=h)  
}

The returned objects can then be passed into tsCV. Let’s consider ARIMA models and ETS models for the air data as introduced in Section @ref(sec-7-trendmethods) where, air <- window(ausair, start=1990).

head(air)

## Time Series:  
## Start = 1990   
## End = 1995   
## Frequency = 1   
## [1] 17.5534 21.8601 23.8866 26.9293 26.8885 28.8314

# Compute CV errors for ETS as e1  
e1 <- tsCV(air, fets, h=1)  
# Compute CV errors for ARIMA as e2  
e2 <- tsCV(air, farima, h=1)  
# Find MSE of each model class  
mean(e1^2, na.rm=TRUE)

## [1] NaN

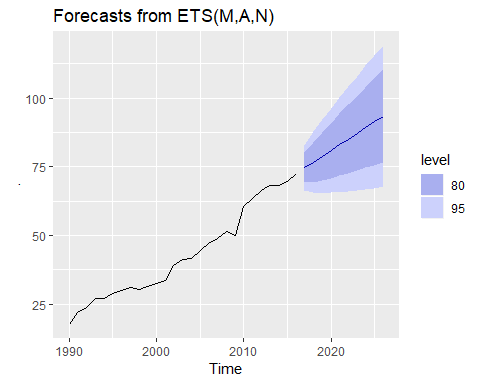
#> [1] 7.864  
mean(e2^2, na.rm=TRUE)

## [1] 9.622164

#> [1] 9.622

In this case, the ets model has a lower tsCV statistic based on MSEs. Below we generate and plot forecasts for the next 5 years generated from an ETS model.

air %>% ets() %>% forecast() %>% autoplot()



### Example: Comparing auto.arima() and ets() on seasonal data

In this case, we want to compare seasonal ARIMA and ETS models applied to the quarterly cement production data qcement. Because the series is relatively long, we can afford to use a training and a test set rather than time series cross-validation. The advantage is that this is much faster. We create a training set from the beginning of 1988 to the end of 2007 and select an ARIMA and an ETs model using the auto.arima() and ets() functions.

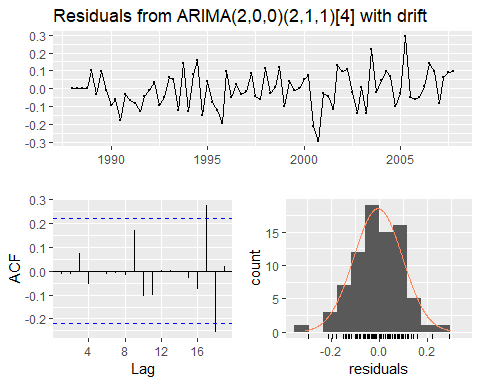
# Consider the qcement data beginning in 1988  
cement <- window(qcement, start=1988)  
  
# USe 20 years of the data as the training set  
train <- window(cement, end=c(2007,4))

The output below shows the ARIMA model selected and estimated by auto.arima. The ARIMA model does well in capturing all the dynamics in the data as the residuals seem to be white noise.

# Fit an ARIMa model to the training data  
(fit.arima <- auto.arima(train))

## Series: train   
## ARIMA(2,0,0)(2,1,1)[4] with drift   
##   
## Coefficients:  
## ar1 ar2 sar1 sar2 sma1 drift  
## 0.6470 0.1934 0.0726 -0.2403 -0.8699 0.0105  
## s.e. 0.1165 0.1183 0.1602 0.1387 0.1624 0.0030  
##   
## sigma^2 estimated as 0.01156: log likelihood=61.45  
## AIC=-108.89 AICc=-107.25 BIC=-92.58

checkresiduals(fit.arima)



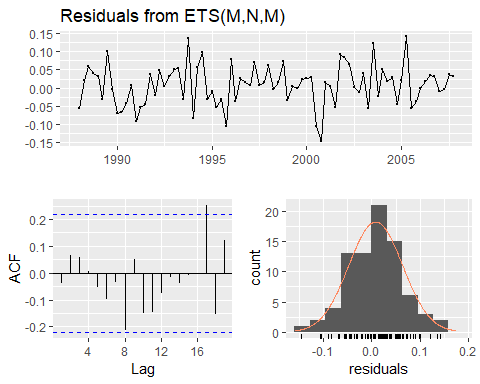
##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(2,0,0)(2,1,1)[4] with drift  
## Q\* = 3.5497, df = 3, p-value = 0.3144  
##   
## Model df: 6. Total lags used: 9

The output below also shows that the ETS model selected and estimated by ets. This models also does well in capturing all the dynamics in the data as the residuals also seem to be white noise.

# Fit an ETS model to the training data  
(fit.ets <- ets(train))

## ETS(M,N,M)   
##   
## Call:  
## ets(y = train)   
##   
## Smoothing parameters:  
## alpha = 0.7341   
## gamma = 1e-04   
##   
## Initial states:  
## l = 1.6439   
## s = 1.031 1.0439 1.0103 0.9148  
##   
## sigma: 0.0581  
##   
## AIC AICc BIC   
## -2.1967020 -0.6411464 14.4774845

checkresiduals(fit.ets)



##   
## Ljung-Box test  
##   
## data: Residuals from ETS(M,N,M)  
## Q\* = 6.3457, df = 3, p-value = 0.09595  
##   
## Model df: 6. Total lags used: 9

The output below evaluates the forecasting performance of the two competing models over the test set. In this case the ETS model seems to be the slighlty more accuarate model based on the test set RMSE, MAPE and MASE.

# Generate forecasts and compare accuracy over the test set  
fit.arima %>% forecast(h = 4\*(2013-2007)+1) %>% accuracy(qcement)

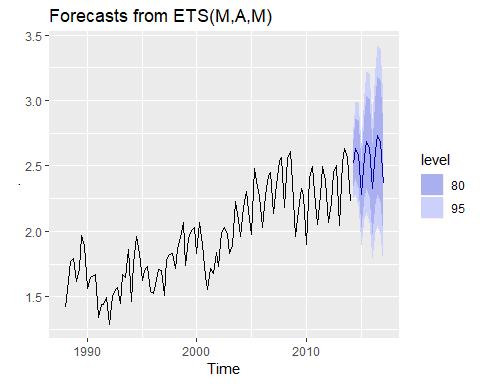
## ME RMSE MAE MPE MAPE  
## Training set -0.005847264 0.1005547 0.07954692 -0.6550065 4.347836  
## Test set -0.158355693 0.1978197 0.16749405 -7.3004639 7.653446  
## MASE ACF1 Theil's U  
## Training set 0.5434705 -0.01315073 NA  
## Test set 1.1443319 0.28848376 0.7226779

fit.ets %>% forecast(h = 4\*(2013-2007)+1) %>% accuracy(qcement)

## ME RMSE MAE MPE MAPE  
## Training set 0.01406512 0.1022079 0.07958478 0.4938163 4.371823  
## Test set -0.13495515 0.1838791 0.15395141 -6.2508975 6.986077  
## MASE ACF1 Theil's U  
## Training set 0.5437292 -0.03346295 NA  
## Test set 1.0518075 0.53438371 0.680556

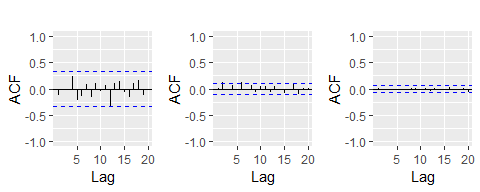
Below we generate and plot forecasts from an ets model for the next 3 years.

# Generate forecasts from an ETS model   
cement %>% ets() %>% forecast(h=12) %>% autoplot()



## Exercises

1. Figure @ref(fig:wnacfplus) shows the ACFs for 36 random numbers, 360 random numbers and 1,000 random numbers.
   1. Explain the differences among these figures. Do they all indicate that the data are white noise?



Left: ACF for a white noise series of 36 numbers. Middle: ACF for a white noise series of 360 numbers. Right: ACF for a white noise series of 1,000 numbers.

1. Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?
2. A classic example of a non-stationary series is the daily closing IBM stock price series (data set ibmclose). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.
3. For the following series, find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.
4. usnetelec
5. usgdp
6. mcopper
7. enplanements
8. visitors
9. For the enplanements data, write down the differences you chose above using backshift operator notation.
10. For your retail data (from Exercise 3 in Section @ref(ex-graphics)), find the appropriate order of differencing (after transformation if necessary) to obtain stationary data.
11. Use R to simulate and plot some data from simple ARIMA models.
    1. Use the following R code to generate data from an AR(1) model with and . The process starts with .

y <- ts(numeric(100))  
e <- rnorm(100)  
for(i in 2:100)  
y[i] <- 0.6\*y[i-1] + e[i]

1. Produce a time plot for the series. How does the plot change as you change ?
2. Write your own code to generate data from an MA(1) model with and .
3. Produce a time plot for the series. How does the plot change as you change ?
4. Generate data from an ARMA(1,1) model with , and .
5. Generate data from an AR(2) model with , and . (Note that these parameters will give a non-stationary series.)
6. Graph the latter two series and compare them.
7. Consider the number of women murdered each year (per 100,000 standard population) in the United States. (Data set wmurders).
8. By studying appropriate graphs of the series in R, find an appropriate ARIMA() model for these data.
9. Should you include a constant in the model? Explain.
10. Write this model in terms of the backshift operator.
11. Fit the model using R and examine the residuals. Is the model satisfactory?
12. Forecast three times ahead. Check your forecasts by hand to make sure that you know how they have been calculated.
13. Create a plot of the series with forecasts and prediction intervals for the next three periods shown.
14. Does auto.arima give the same model you have chosen? If not, which model do you think is better?
15. Consider the total international visitors to Australia (in millions) for the period 1980-2015. (Data set austa.)
16. Use auto.arima to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.
17. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part (a). Remove the MA term and plot again.
18. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.
19. Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.
20. Plot forecasts from an ARIMA(0,2,1) model with no constant.
21. For the usgdp series:
22. if necessary, find a suitable Box-Cox transformation for the data;
23. fit a suitable ARIMA model to the transformed data using auto.arima();
24. try some other plausible models by experimenting with the orders chosen;
25. choose what you think is the best model and check the residual diagnostics;
26. produce forecasts of your fitted model. Do the forecasts look reasonable?
27. compare the results with what you would obtain using ets() (with no transformation).
28. Consider austourists, the quarterly number of international tourists to Australia for the period 1999–2010. (Data set austourists.)
29. Describe the time plot.
30. What can you learn from the ACF graph?
31. What can you learn from the PACF graph?
32. Produce plots of the seasonally differenced data . What model do these graphs suggest?
33. Does auto.arima give the same model that you chose? If not, which model do you think is better?
34. Write the model in terms of the backshift operator, then without using the backshift operator.
35. Consider the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period January 1973 – June 2013). (Data set usmelec.) In general there are two peaks per year: in mid-summer and mid-winter.
36. Examine the 12-month moving average of this series to see what kind of trend is involved.
37. Do the data need transforming? If so, find a suitable transformation.
38. Are the data stationary? If not, find an appropriate differencing which yields stationary data.
39. Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?
40. Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.
41. Forecast the next 15 years of electricity generation by the U.S. electric industry. Get the latest figures from <https://goo.gl/WZIItv> to check the accuracy of your forecasts.
42. How many years of forecasts do you think are sufficiently accurate to be usable?
43. For the mcopper data:
44. if necessary, find a suitable Box-Cox transformation for the data;
45. fit a suitable ARIMA model to the transformed data using auto.arima();
46. try some other plausible models by experimenting with the orders chosen;
47. choose what you think is the best model and check the residual diagnostics;
48. produce forecasts of your fitted model. Do the forecasts look reasonable?
49. compare the results with what you would obtain using ets() (with no transformation).
50. Choose one of the following seasonal time series: hsales, auscafe, qauselec, qcement, qgas.
51. Do the data need transforming? If so, find a suitable transformation.
52. Are the data stationary? If not, find an appropriate differencing which yields stationary data.
53. Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?
54. Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.
55. Forecast the next 24 months of data using your preferred model.
56. Compare the forecasts obtained using ets().
57. For the same time series you used in the previous exercise, try using a non-seasonal model applied to the seasonally adjusted data obtained from STL. The stlf() function will make the calculations easy (with method="arima"). Compare the forecasts with those obtained in the previous exercise. Which do you think is the best approach?
58. For your retail time series (Exercise 5 above):
59. develop an appropriate seasonal ARIMA model;
60. compare the forecasts with those you obtained in earlier chapters;
61. Obtain up-to-date retail data from the [ABS website](https://goo.gl/twfCyD) (Cat 8501.0, Table 11), and compare your forecasts with the actual numbers. How good were the forecasts from the various models?
62. Produce a time plot of the sheep population of England and Wales from 1867–1939 (data set sheep).
63. Assume you decide to fit the following model:

where is a white noise series. What sort of ARIMA model is this (i.e., what are , , and )?

1. By examining the ACF and PACF of the differenced data, explain why this model is appropriate.
2. The last five values of the series are given below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Year | 1935 | 1936 | 1937 | 1938 | 1939 |
| Millions of sheep | 1648 | 1665 | 1627 | 1791 | 1797 |

The estimated parameters are , , and . Without using the forecast function, calculate forecasts for the next three years (1940–1942).

1. Now fit the model in R and obtain the forecasts using forecast. How are they different from yours? Why?
2. Plot the annual bituminous coal production in the United States from 1920 to 1968 (data set ).
3. You decide to fit the following model to the series:
4. where is the coal production in year and is a white noise series. What sort of ARIMA model is this (i.e., what are , , and )?
   1. Explain why this model was chosen using the ACF and PACF.
   2. The last five values of the series are given below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Year | 1964 | 1965 | 1966 | 1967 | 1968 |
| Millions of tons | 467 | 512 | 534 | 552 | 545 |

The estimated parameters are , , , , and . Without using the forecast function, calculate forecasts for the next three years (1969–1971).

1. Now fit the model in R and obtain the forecasts from the same model. How are they different from yours? Why?
   1. Install the **rdatamarket** package in R using
   * install.packages("rdatamarket")
   1. Select a time series from <http://datamarket.com/data/list/?q=pricing:free>. Then copy its short URL and import the data using
   * x <- ts(rdatamarket::dmseries("shorturl")[,1], start=??, frequency=??)
   * (Replace ?? with the appropriate values.)
   1. Plot graphs of the data, and try to identify an appropriate ARIMA model.
   2. Do residual diagnostic checking of your ARIMA model. Are the residuals white noise?
   3. Use your chosen ARIMA model to forecast the next four years.
   4. Now try to identify an appropriate ETS model.
   5. Do residual diagnostic checking of your ETS model. Are the residuals white noise?
   6. Use your chosen ETS model to forecast the next four years.
   7. Which of the two models do you prefer?

## Further reading

The classic text which popularized ARIMA modelling was @BJ70. The most recent edition is @BJRL15, and it is still an excellent reference for all things ARIMA. @BDbook16 provides a good introduction to the mathematical background to the models, while @PTT01 describes some alternative automatic algorithms to the one used by auto.arima().

### Bibliograhy

Bibliography Box, G. E. P., & Jenkins, G. M. (1970). Time series analysis: Forecasting and control. San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: Forecasting and control (5th ed). Hoboken, New Jersey: John Wiley & Sons. [Amazon]

Brockwell, P. J., & Davis, R. A. (2016). Introduction to time series and forecasting (3rd ed). New York, USA: Springer. [Amazon]

Peña, D., Tiao, G. C., & Tsay, R. S. (Eds.). (2001). A course in time series analysis. New York, USA: John Wiley & Sons. [Amazon]

# Dynamic regression models

The time series models in the previous two chapters allow for the inclusion of information from past observations of a series, but not for the inclusion of other information that may also be relevant. For example, the effects of holidays, competitor activity, changes in the law, the wider economy, or other external variables, may explain some of the historical variation and may lead to more accurate forecasts. On the other hand, the regression models in Chapter 5 allow for the inclusion of a lot of relevant information from predictor variables, but do not allow for the subtle time series dynamics that can be handled with ARIMA models.

In this chapter, we consider how to extend ARIMA models in order to allow other information to be included in the models.

In Chapter 5, we consideredregression models of the form:

where is a linear function of the predictor variables (), and is usually assumed to be an uncorrelated error term (i.e., it is white noise). We considered tests such as the **Breusch-Godfrey test** for assessing whether was significantly correlated.

In this chapter, we will allow the errors from a regression to contain autocorrelation. To emphasise this change in perspective, we will replace with in the equation. The error series is assumed to follow an ARIMA model. For example, if follows an ARIMA(1,1,1) model, we can write

where is a white noise series.

Notice that the model has two error terms here — the error from the regression model, which we denote by and the error from the ARIMA model, which we denote by . ONly the ARIMA model errors are assumed to be white noise.

## Estimation

When we estimate parameters from the model, we need to minimise the sum of squared values. If we minimize the sum of squared values instead (which is what would happen if we estimated the regression model ignoring the autocorrelations in the errors), then several problems arise.

1. The estimated coefficients are no glonger the best estimates, as some information has been ignored in the calculation;
2. Any statistical tests associated with the model (e.g., t-tests on the coefficients) will be incorrect
3. The AICc values of the fitted models are no longer a good guide as to which is the best model for forecasting.
4. In most cases, the -values associated with the coefficients will be too small, and so some predictor variables will appear to be important when they are not. This is known as “spurious regression”.

Minimizing the sum of squared values avoids these problems. Alternatively, **maximum likelihood estimation** can be used; this will give very similar estimates of the coefficients.

An important consideration when estimating a regression with ARMA errors is that all of the variables in the model must first be stationary. Thus, we first have to check that and all of the predictors appear to be stationary.

If we estimate the model when any of these are non-stationary, the estimated coefficients will not be consistent estimates (and therefore may not be meaningful). One exception to this is the case where non-stationary variables are co-integrated. If there exists a linear combination of the non-stationary and the predictors that is stationary, then the estimated coefficients will be consistent.[[3]](#footnote-106)

We therefore first difference the non-stationary variables in the model. It is often desirable to maintain the form of the relationship between and the predictors, and consequently it is common to difference all of the variables if any of them need differencing. The resulting model is then called a “model in differences”, as distinct from a “model in levels”, which is what is obtained when the original data are used without differencing.

If all of the variables in the model are stationary, then we only need to consider ARMA errors for the residuals. It is easy to see that a regression model with ARIMA errors is equivalent to a regression model in differences with ARMA errors.

For example, if the above regression model with ARIMA(1,1,1) errors is differenced we obtain the model

where , and , which is a regression model in differences with ARMA errors.

## Regression with ARIMA errors in R

The R function Arima() willfit a regression model with ARIMA errors if the argument xreg is used. The order argument specifies the order of the ARIMA error model.If differencing is specified, then the differencing is applied to all variables in the regression.

fit <- Arima(y, xreg=x, order=c(1,1,0))

will fit the model , where is an AR(1) error. This is equivalent to the model

where is an ARIMA(1,1,0) error. Notice that the constant term disappearsdue to the differencing. To include a constant in the differenced model, specify include.drift=TRUE.

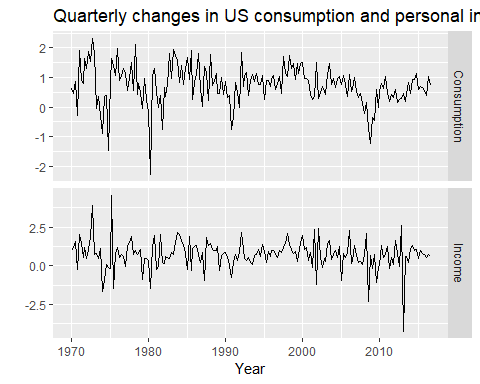
The auto.arima() function will also handle regression terms via the xreg argument. The user must specify the predictor variables to include, but auto.arima will select the best ARIMA model for the errors.

The AICc is calculated for the final model, and this value can be used to determine the best predictors. That is, the procedure should be repeated for all subsets of predictors to be considered, and the model with the loweest AICc value selected.

### Example: US Personal Consumption and Income

Figure @ref(fig:usconsump) shows the quarterly changes in personal consumption expenditure and personal disposable income from 1970 to 2010. We would like to forecast changes in expenditure based on changes in income. A change in income does not necessarily translate to an instant change in consumption (e.g., after the loss of a job, it may take a few months for expenses to be reduced to allow for the new circumstances). However, we will ignore this complexity in this example and try to measure the instantaneous effect of the average change of income on the average change of consumption expenditure.

autoplot(uschange[,1:2],facets=TRUE)+  
 xlab("Year")+ylab("")+  
 ggtitle("Quarterly changes in US consumption and personal income")



Percentage changes in quarterly personal consumption expenditure and personal disposable income for the USA, 1970 to 2010.

(fit <- auto.arima(uschange[,"Consumption"], xreg=uschange[,"Income"]))

## Series: uschange[, "Consumption"]   
## Regression with ARIMA(1,0,2) errors   
##   
## Coefficients:  
## ar1 ma1 ma2 intercept xreg  
## 0.6922 -0.5758 0.1984 0.5990 0.2028  
## s.e. 0.1159 0.1301 0.0756 0.0884 0.0461  
##   
## sigma^2 estimated as 0.3219: log likelihood=-156.95  
## AIC=325.91 AICc=326.37 BIC=345.29

The data are clearly already stationary (as we are considering percentage changes rather than raw expenditure and income), so there is no need for any differencing. The fitted model is

1. arc cos is the inverse cosine function. You should be able to find it on your calculator. It may be labelled acos or cos. [↑](#footnote-ref-50)
2. As already pointed out, comparing information criteria is only valid for ARIMA models of the same orders of differencing. [↑](#footnote-ref-90)
3. Forecasting with cointegrated models is discussed by @Harris03. [↑](#footnote-ref-106)