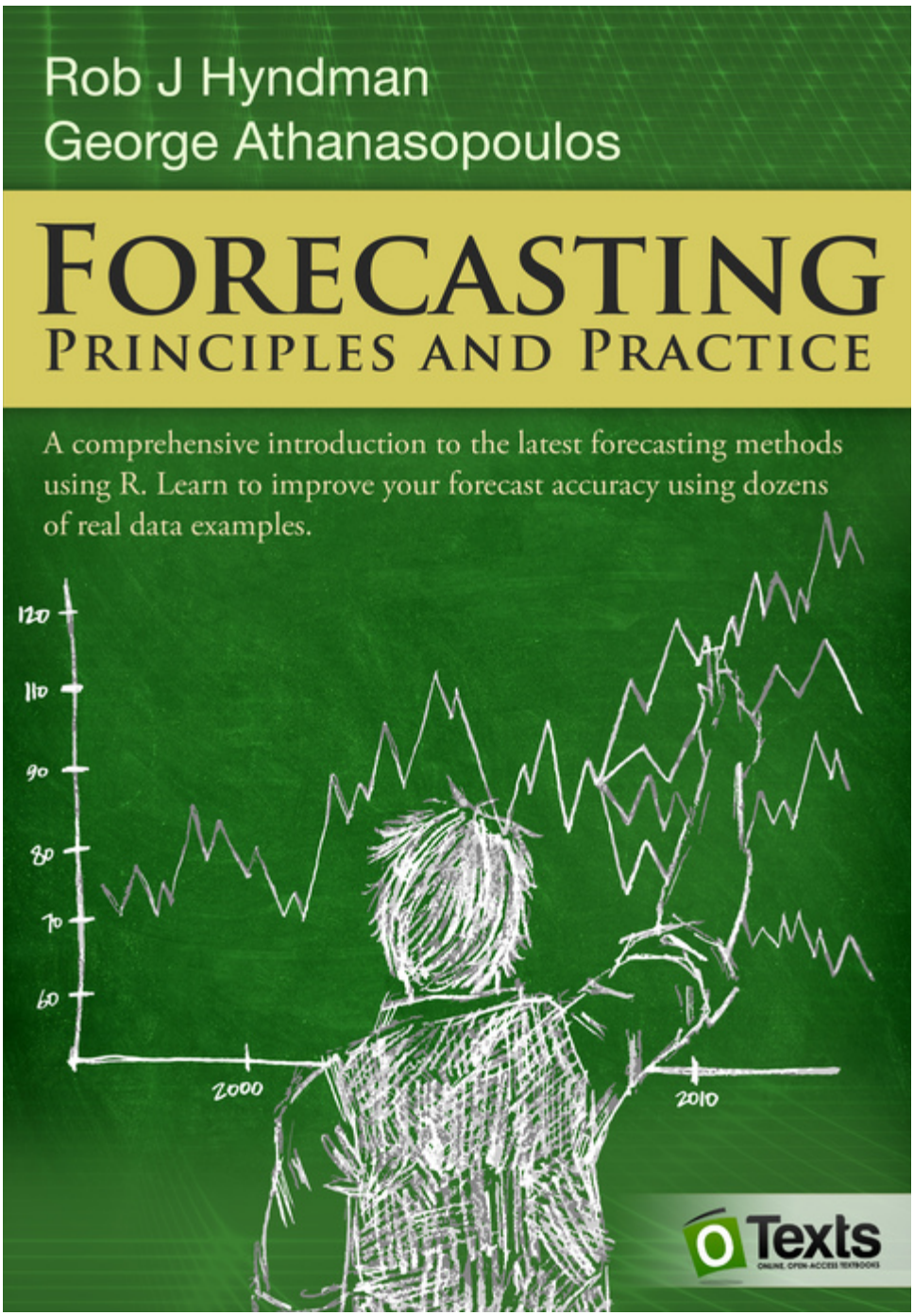


Forecasting: Principles and Practice

otexts.org |

George Athanasopoulos

Welcome

Welcome to our online textbook on forecasting. This textbook is intended to provide a comprehensive introduction to forecasting methods and to present enough information about each method for readers to be able to use them sensibly. We don't attempt to give a thorough discussion

of the theoretical details behind each method, although the references at the end of each chapter will fill in many of those details.

The book is written for three audiences: (1) people finding themselves doing forecasting in business when they may not have had any formal training in the area; (2) undergraduate students studying business; (3) MBA students doing a forecasting elective. We use it ourselves for a third-year subject for students undertaking a Bachelor of Commerce or a Bachelor of Business degree at Monash University, Australia.

For most sections, we only assume that readers are familiar with introductory statistics, and with high-school algebra. There are a couple of sections that also require knowledge of matrices, but these are flagged.

At the end of each chapter we provide a list of "further reading". In general, these lists comprise suggested textbooks that provide a more advanced or detailed treatment of the subject. Where there is no suitable textbook, we suggest journal articles that provide more information.

We use R throughout the book and we intend students to learn how to forecast with R. R is free and available on almost every operating system. It is a wonderful tool for all statistical analysis, not just for forecasting. See Using R for instructions on installing and using R.

All R examples in the book assume you have loaded the `fpp2` package, available on CRAN, using `library(fpp2)`. This will automatically load several other packages including `forecast` and `ggplot2`, as well as all the data used in the book. The `fpp2` package requires at least version 8.0 of the `forecast` package and version 2.0.0 of the `ggplot2` package.

We will use the `ggplot2` package for all graphics. If you want to learn how to modify the graphs, or create your own `ggplot2` graphics that are different from the examples shown in this book, please either read the ggplot2 book, or do the ggplot2 course on DataCamp.

There is also a DataCamp course based on this book which provides an introduction to some of the ideas in Chapters 2, 3, 7 and 8, plus a brief glimpse at a few of the topics in Chapters 9 and 11.

The book is different from other forecasting textbooks in several ways.

- It is free and online, making it accessible to a wide audience.
- It uses R, which is free, open-source, and extremely powerful software.
- The online version is continuously updated. You don't have to wait until the next edition for errors to be removed or new methods to be discussed. We will update the book frequently.
- There are dozens of real data examples taken from our own consulting practice. We have worked with hundreds of businesses and organizations helping them with forecasting issues, and this experience has contributed directly to many of the examples given here, as well as guiding our general philosophy of forecasting.
- We emphasise graphical methods more than most forecasters. We use graphs to explore the data, analyse the validity of the models fitted and present the forecasting results.

Changes in the second edition

The most important change in edition 2 of the book is that we have restricted our focus to *time series forecasting*. That is, we no longer consider the problem of cross-sectional prediction. Instead, all forecasting in this book concerns prediction of data at future times using observations collected in the past.

We have also simplified the chapter on exponential smoothing, and added new chapters on dynamic regression forecasting, hierarchical forecasting and practical forecasting issues. We have added new material on combining forecasts, handling complicated seasonality patterns, dealing with hourly, daily and weekly data, forecasting count time series, and we have added several new examples involving electricity demand, online shopping, and restaurant bookings. We have also revised all existing chapters to bring them up-to-date with the latest research, and we have carefully gone through every chapter to improve the explanations where possible, to add newer references, to add more exercises, and to make the R code simpler.

Helpful readers of the earlier versions of the book let us know of any typos or errors they had found. These were updated immediately online. No doubt we have introduced some new mistakes, and we will correct them online as soon as they are spotted. Please continue to let us know about such things.

Happy forecasting!

Rob J Hyndman
George Athanasopoulos

August 2017

Chapter 1 Getting started

Forecasting has fascinated people for thousands of years, sometimes being considered a sign of divine inspiration, and sometimes being seen as a criminal activity. The Jewish prophet Isaiah wrote in about 700 BC

> *Tell us what the future holds, so we may know that you are gods.*
> (Isaiah 41:23)

One hundred years later, in ancient Babylon, forecasters would foretell the future based on the distribution of maggots in a rotten sheep's liver. By 300 BC, people wanting forecasts would journey to Delphi in Greece to consult the Oracle, who would provide her predictions while intoxicated by ethylene vapours. Forecasters had a tougher time under the emperor Constantine, who issued a decree in AD357 forbidding anyone "to consult a soothsayer, a mathematician, or a forecaster … May curiosity to foretell the future be silenced forever." A similar ban on forecasting occurred in England in 1736 when it became an offence to defraud by charging money for predictions. The punishment was three months' imprisonment with hard labour!

The varying fortunes of forecasters arise because good forecasts can seem almost magical, while bad forecasts may be dangerous. Consider the following famous predictions about computing.

- *I think there is a world market for maybe five computers.*
  (Chairman of IBM, 1943)
- *Computers in the future may weigh no more than 1.5 tons.*
- *There is no reason anyone would want a computer in their home.*
  (President, DEC, 1977)

The last of these was made only three years before IBM produced the first personal computer. Not surprisingly, you can no longer buy a DEC computer. Forecasting is obviously a difficult activity, and businesses that do it well have a big advantage over those whose forecasts fail.

In this book, we will explore the most reliable methods for producing forecasts. The emphasis will be on methods that are replicable and testable, and have been shown to work.

1.1 What can be forecast?

Forecasting is required in many situations: deciding whether to build another power generation plant in the next five years requires forecasts of future demand; scheduling staff in a call center next week requires forecasts of call volumes; stocking an inventory requires forecasts of stock requirements. Forecasts can be required several years in advance (for the case of capital investments), or only a few minutes beforehand (for telecommunication routing). Whatever the circumstances or time horizons involved, forecasting is an important aid to effective and efficient planning.

Some things are easier to forecast than others. The time of the sunrise tomorrow morning can be forecast very precisely. On the other hand, tomorrow's lotto numbers cannot be forecast with any accuracy. The predictability of an event or a quantity depends on several factors including:

1. how well we understand the factors that contribute to
2. how much data are available;
3. whether the forecasts can affect the thing we are trying to forecast.

For example, forecasts of electricity demand can be highly accurate because all three conditions are usually satisfied. We have a good idea on the contributing factors: electricity demand is driven largely by temperatures, with smaller effects for calendar variation such as holidays, and economic conditions. Provided there is a sufficient history of data on electricity demand and weather conditions, and we have the skills to develop a good model linking electricity demand and the key driver variables, the forecasts can be remarkably accurate.

On the other hand, when forecasting currency exchange rates, only one of the conditions is satisfied: there is plenty of available data. However, we have a very limited understanding of the factors that affect exchange rates, and forecasts of the exchange rate have a direct effect on the rates themselves. If there are well-publicized forecasts that the exchange rate will increase, then people will immediately adjust the price they are willing to pay and so the forecasts are self-fulfilling. In a sense the exchange rates become their own forecasts. This is an example of the "efficient market hypothesis". Consequently, forecasting whether the exchange rate will rise or fall tomorrow is about as predictable as forecasting whether a tossed coin will come down as a head or a tail. In both situations, you will be correct about 50% of the time, whatever you forecast. In situations like this, forecasters need to be aware of their own limitations, and not claim more than is possible.

Often in forecasting, a key step is knowing when something can be forecast accurately, and when forecasts will be no better than tossing a coin. Good forecasts capture the genuine patterns and relationships which exist in the historical data, but do not replicate past events that will not occur again. In this book, we will learn how to tell the difference between a random fluctuation in the past data that should be ignored, and a genuine pattern that should be modelled and extrapolated.

Many people wrongly assume that forecasts are not possible in a changing environment. Every environment is changing, and a good forecasting model captures the way in which things are changing. Forecasts rarely assume that the environment is unchanging. What is normally assumed is that *the way in which the environment is changing* will continue into the future. That is, a highly volatile environment will continue to be highly volatile; a business with fluctuating sales will continue to have fluctuating sales; and an economy that has gone through booms and busts will continue to go through booms and busts. A forecasting model is intended to capture the way

things move, not just where things are. As Abraham Lincoln said, “If we could first know where we are and whither we are tending, we could better judge what to do and how to do it”.

Forecasting situations vary widely in their time horizons, factors determining actual outcomes, types of data patterns, and many other aspects. Forecasting methods can be very simple, such as using the most recent observation as a forecast (which is called the “naïve method”), or highly complex, such as neural nets and econometric systems of simultaneous equations. Sometimes, there will be no data available at all. For example, we may wish to forecast the sales of a new product in its first year, but there are obviously no data to work with. In situations like this, we use judgmental forecasting, discussed in Chapter 4. The choice of method depends on what data are available and the predictability of the quantity to be forecast.

1.2 Forecasting, planning and goals

Forecasting is a common statistical task in business, where it helps to inform decisions about the scheduling of production, transportation and personnel, and provides a guide to long-term strategic planning. However, business forecasting is often done poorly, and is frequently confused with planning and goals. They are three different things.

Forecasting
: is about predicting the future as accurately as possible, given all of the information available, including historical data and knowledge of any future events that might impact the forecasts.

Goals
: are what you would like to have happen. Goals should be linked to forecasts and plans, but this does not always occur. Too often, goals are set without any plan for how to achieve them, and no forecasts for whether they are realistic.

Planning
: is a response to forecasts and goals. Planning involves determining the appropriate actions that are required to make your forecasts match your goals.

Forecasting should be an integral part of the decision-making activities of management, as it can play an important role in many areas of a company. Modern organizations require short-term, medium-term and long-term forecasts, depending on the specific application.

Short-term forecasts
: are needed for the scheduling of personnel, production and transportation. As part of the scheduling process, forecasts of demand are often also required.

Medium-term forecasts
: are needed to determine future resource requirements, in order to purchase raw materials, hire personnel, or buy machinery and equipment.

Long-term forecasts
: are used in strategic planning. Such decisions must take account of market opportunities, environmental factors and internal resources.

An organization needs to develop a forecasting system that involves several approaches to predicting uncertain events. Such forecasting systems require the development of expertise in identifying forecasting problems, applying a range of forecasting methods, selecting appropriate methods for each problem, and evaluating and refining forecasting methods over time. It is also important to have strong organizational support for the use of formal forecasting methods if they are to be used successfully.

1.3 Determining what to forecast

In the early stages of a forecasting project, decisions need to be made about what should be forecast. For example, if forecasts are required for items in a manufacturing environment, it is necessary to ask whether forecasts are needed for:

1. every product line, or for groups of products?
2. every sales outlet, or for outlets grouped by region, or only for total sales?
3. weekly data, monthly data or annual data?

It is also necessary to consider the forecasting horizon. Will forecasts be required for one month in advance, for 6 months, or for ten years? Different types of models will be necessary, depending on what forecast horizon is most important.

How frequently are forecasts required? Forecasts that need to be produced frequently are better done using an automated system than with methods that require careful manual work.

It is worth spending time talking to the people who will use the forecasts to ensure that you understand their needs, and how the forecasts are to be used, before embarking on extensive work in producing the forecasts.

Once it has been determined what forecasts are required, it is then necessary to find or collect the data on which the forecasts will be based. The data required for forecasting may already exist. These days, a lot of data are recorded, and the forecaster's task is often to identify where and how the required data are stored. The data may include sales records of a company, the historical demand for a product, or the unemployment rate for a geographical region. A large part of a forecaster's time can be spent in locating and collating the available data prior to developing suitable forecasting methods.

1.4 Forecasting data and methods

The appropriate forecasting methods depend largely on what data are available.

If there are no data available, or if the data available are not relevant to the forecasts, then *qualitative forecasting* methods must be used. These methods are not purely guesswork—there are well-developed structured approaches to obtaining good forecasts without using historical data. These methods are discussed in Chapter 4.

Quantitative forecasting can be applied when two conditions are satisfied:

1. numerical information about the past is available;
2. it is reasonable to assume that some aspects of the past patterns will continue into the future.

There is a wide range of quantitative forecasting methods, often developed within specific disciplines for specific purposes. Each method has its own properties, accuracies, and costs that must be considered when choosing a specific method.

Most quantitative prediction problems use either time series data (collected at regular intervals over time) or cross-sectional data (collected at a single point in time). In this book we are concerned with forecasting future data, and we concentrate on the time series domain.

Time series forecasting

Examples of time series data include:

- Daily IBM stock prices
- Monthly rainfall
- Quarterly sales results for Amazon
- Annual Google profits

Anything that is observed sequentially over time is a time series. In this book, we will only consider time series that are observed at regular intervals of time (e.g., hourly, daily, weekly, monthly, quarterly, annually). Irregularly spaced time series can also occur, but are beyond the scope of this book.

When forecasting time series data, the aim is to estimate how the sequence of observations will continue into the future. Figure 1.1 shows the quarterly Australian beer production from 1992 to the second quarter of 2010.

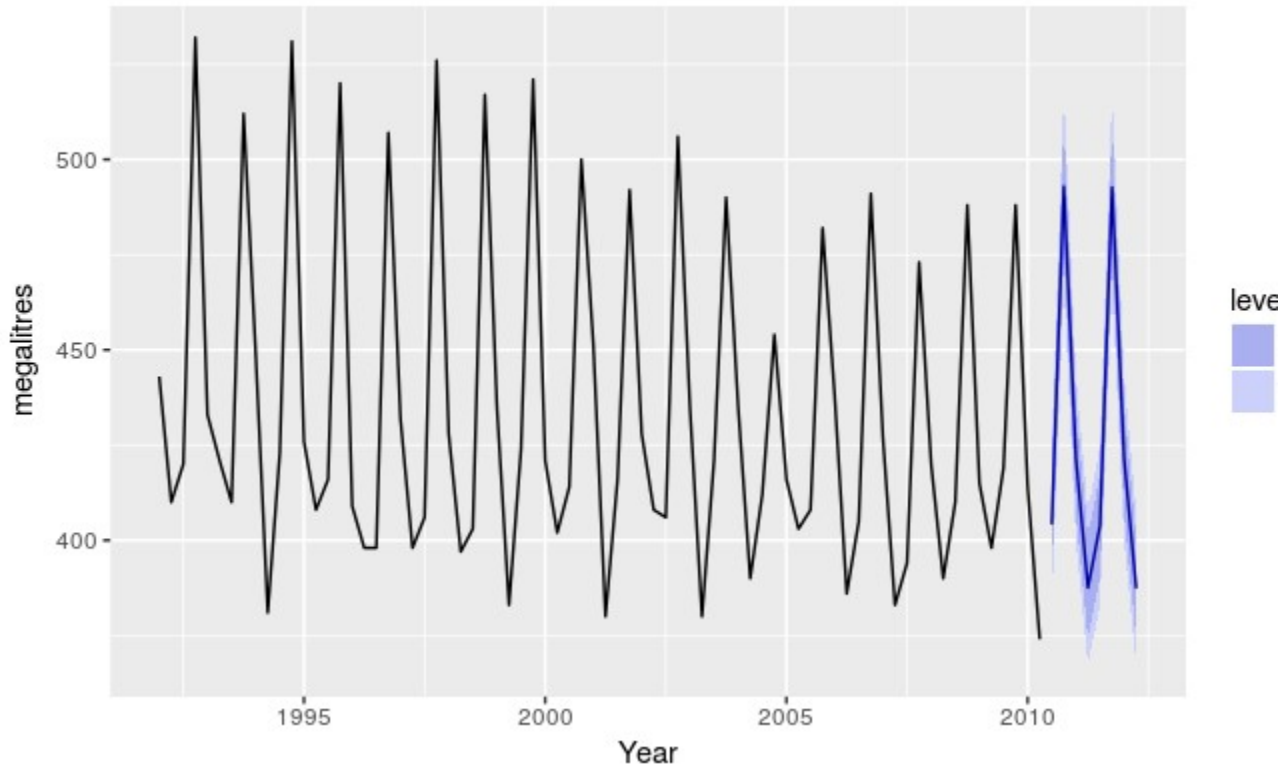


Figure 1.1: Australian quarterly beer production: 1992Q1–2010Q2, with two years of forecasts.

The blue lines show forecasts for the next two years. Notice how the forecasts have captured the seasonal pattern seen in the historical data and replicated it for the next two years. The dark shaded region shows 80% prediction intervals. That is, each future value is expected to lie in the dark shaded region with a probability of 80%. The light shaded region shows 95% prediction intervals. These prediction intervals are a very useful way of displaying the uncertainty in forecasts. In this case the forecasts are expected to be very accurate, and hence the prediction intervals are quite narrow.

The simplest time series forecasting methods use only information on the variable to be forecast, and make no attempt to discover the factors that affect its behaviour. Therefore they will

extrapolate trend and seasonal patterns, but they ignore all other information such as marketing initiatives, competitor activity, changes in economic conditions, and so on.

Time series models used for forecasting include ARIMA models, exponential smoothing and structural models. These models are discussed in Chapters 6, 7 and 8.

Predictor variables and time series forecasting

Predictor variables can also be used in time series forecasting. For example, suppose we wish to forecast the hourly electricity demand (ED) of a hot region during the summer period. A model with predictor variables might be of the form $$\begin{aligned} \text{ED} = & f(\text{current temperature, strength of economy, population,}\\ & \qquad\text{time of day, day of week, error}). \end{aligned}$$

The relationship is not exact—there will always be changes in electricity demand that cannot be accounted for by the predictor variables. The "error" term on the right allows for random variation and the effects of relevant variables that are not included in the model. We call this an "explanatory model" because it helps explain what causes the variation in electricity demand.

Because the electricity demand data form a time series, we could also use a time series model for forecasting. In this case, a suitable time series forecasting equation is of the form $$ \text{ED}_{t+1} = f(\text{ED}_{t}, \text{ED}_{t-1}, \text{ED}_{t-2}, \text{ED}_{t-3},\dots, \text{error}), $$ where t is the present hour, $t+1$ is the next hour, $t-1$ is the previous hour, $t-2$ is two hours ago, and so on. Here, prediction of the future is based on past values of a variable, but not on external variables which may affect the system. Again, the "error" term on the right allows for random variation and the effects of relevant variables that are not included in the model.

There is also a third type of model which combines the features of the above two models. For example, it might be given by $$ \text{ED}_{t+1} = f(\text{ED}_{t}, \text{current temperature, time of day, day of week, error}). $$ These types of mixed models have been given various names in different disciplines. They are known as dynamic regression models, panel data models, longitudinal models, transfer function models, and linear system models (assuming that f is linear). These models are discussed in Chapter 9.

An explanatory model is very useful because it incorporates information about other variables, rather than only historical values of the variable to be forecast. However, there are several reasons a forecaster might select a time series model rather than an explanatory model. First, the system may not be understood, and even if it was understood it may be extremely difficult to measure the relationships that are assumed to govern its behaviour. Second, it is necessary to know or forecast the future values of the various predictors in order to be able to forecast the variable of interest, and this may be too difficult. Third, the main concern may be only to predict what will happen, not to know why it happens. Finally, the time series model may give more accurate forecasts than an explanatory or mixed model.

The model to be used in forecasting depends on the resources and data available, the accuracy of the competing models, and the way in which the forecasting model is to be used.

Notation

We will use the subscript t for time. For example, y_t will denote the observation at time t. We will use T to denote the total number of observations in a time series.

1.5 Some case studies

The following four cases are from our consulting practice and demonstrate different types of forecasting situations and the associated problems that often arise.

Case 1

The client was a large company manufacturing disposable tableware such as napkins and paper plates. They needed forecasts of each of hundreds of items every month. The time series data showed a range of patterns, some with trends, some seasonal, and some with neither. At the time, they were using their own software, written in-house, but it often produced forecasts that did not seem sensible. The methods that were being used were the following:

1. average of the last 12 months data;
2. average of the last 6 months data;
3. prediction from a straight line regression over the last 12 months;
4. prediction from a straight line regression over the last 6 months;
5. prediction obtained by a straight line through the last observation with slope equal to the average slope of the lines connecting last year's and this year's values;
6. prediction obtained by a straight line through the last observation with slope equal to the average slope of the lines connecting last year's and this year's values, where the average is taken only over the last 6 months.

They required us to tell them what was going wrong and to modify the software to provide more accurate forecasts. The software was written in COBOL, making it difficult to do any sophisticated numerical computation.

Case 2

In this case, the client was the Australian federal government, who needed to forecast the annual budget for the Pharmaceutical Benefit Scheme (PBS). The PBS provides a subsidy for many pharmaceutical products sold in Australia, and the expenditure depends on what people purchase during the year. The total expenditure was around A$7 billion in 2009, and had been underestimated by nearly $1 billion in each of the two years before we were asked to assist in developing a more accurate forecasting approach.

In order to forecast the total expenditure, it is necessary to forecast the sales volumes of hundreds of groups of pharmaceutical products using monthly data. Almost all of the groups have trends and seasonal patterns. The sales volumes for many groups have sudden jumps up or down due to changes in what drugs are subsidised. The expenditures for many groups also have sudden changes due to cheaper competitor drugs becoming available.

Thus we needed to find a forecasting method that allowed for trend and seasonality if they were present, and at the same time was robust to sudden changes in the underlying patterns. It also needed to be able to be applied automatically to a large number of time series.

Case 3

A large car fleet company asked us to help them forecast vehicle re-sale values. They purchase new vehicles, lease them out for three years, and then sell them. Better forecasts of vehicle sales values would mean better control of profits; understanding what affects resale values may allow leasing and sales policies to be developed in order to maximize profits.

At the time, the resale values were being forecast by a group of specialists. Unfortunately, they saw any statistical model as a threat to their jobs, and were uncooperative in providing information. Nevertheless, the company provided a large amount of data on previous vehicles and their eventual resale values.

Case 4

In this project, we needed to develop a model for forecasting weekly air passenger traffic on major domestic routes for one of Australia's leading airlines. The company required forecasts of passenger numbers for each major domestic route and for each class of passenger (economy class, business class and first class). The company provided weekly traffic data from the previous six years.

Air passenger numbers are affected by school holidays, major sporting events, advertising campaigns, competition behaviour, etc. School holidays often do not coincide in different Australian cities, and sporting events sometimes move from one city to another. During the period of the historical data, there was a major pilots' strike during which there was no traffic for several months. A new cut-price airline also launched and folded. Towards the end of the historical data, the airline had trialled a redistribution of some economy class seats to business class, and some business class seats to first class. After several months, however, the seat classifications reverted to the original distribution.

1.6 The basic steps in a forecasting task

A forecasting task usually involves five basic steps.

Step 1: Problem definition.
: Often this is the most difficult part of forecasting. Defining the problem carefully requires an understanding of the way the forecasts will be used, who requires the forecasts, and how the forecasting function fits within the organization requiring the forecasts. A forecaster needs to spend time talking to everyone who will be involved in collecting data, maintaining databases, and using the forecasts for future planning.

Step 2: Gathering information.
: There are always at least two kinds of information required: (a) statistical data, and (b) the accumulated expertise of the people who collect the data and use the forecasts. Often, it will be difficult to obtain enough historical data to be able to fit a good statistical model. However, occasionally, very old data will be less useful due to changes in the system being forecast.

Step 3: Preliminary (exploratory) analysis.
: Always start by graphing the data. Are there consistent patterns? Is there a significant trend? Is seasonality important? Is there evidence of the presence of business cycles? Are there any outliers in the data that need to be explained by those with expert knowledge? How strong are the relationships among the variables available for analysis? Various tools have been developed to help with this analysis. These are discussed in Chapters 2 and 6.

Step 4: Choosing and fitting models.
: The best model to use depends on the availability of historical data, the strength of relationships between the forecast variable and any explanatory variables, and the way in which the forecasts are to be used. It is common to compare two or three potential models. Each model is itself an artificial construct that is based on a set of assumptions (explicit and implicit) and usually involves one or more parameters which must be "fitted" using the known historical data. We will discuss regression models (Chapter 5), exponential smoothing methods (Chapter 7), Box-Jenkins ARIMA models (Chapter 8), Dynamic

regression models (Chapter 9), Hierarchical forecasting (Chapter 10), and a variety of other topics including count time series, neural networks and vector autoregression in Chapter 11.

Step 5: Using and evaluating a forecasting model.

Once a model has been selected and its parameters estimated, the model is used to make forecasts. The performance of the model can only be properly evaluated after the data for the forecast period have become available. A number of methods have been developed to help in assessing the accuracy of forecasts. There are also organizational issues in using and acting on the forecasts. A brief discussion of some of these issues is given in Chapter 3.

1.7 The statistical forecasting perspective

The thing we are trying to forecast is unknown (or we wouldn't be forecasting it), and so we can think of it as a *random variable*. For example, the total sales for next month could take a range of possible values, and until we add up the actual sales at the end of the month, we don't know what the value will be. So until we know the sales for next month, it is a random quantity.

Because next month is relatively close, we usually have a good idea what the likely sales values could be. On the other hand, if we are forecasting the sales for the same month next year, the possible values it could take are much more variable. In most forecasting situations, the variation associated with the thing we are forecasting will shrink as the event approaches. In other words, the further ahead we forecast, the more uncertain we are.

We can imagine many possible futures, each yielding a different value for the thing we wish to forecast. Plotted below in black are the total international visitors to Australia from 1980 to 2015. Also shown are ten possible futures from 2016–2025.

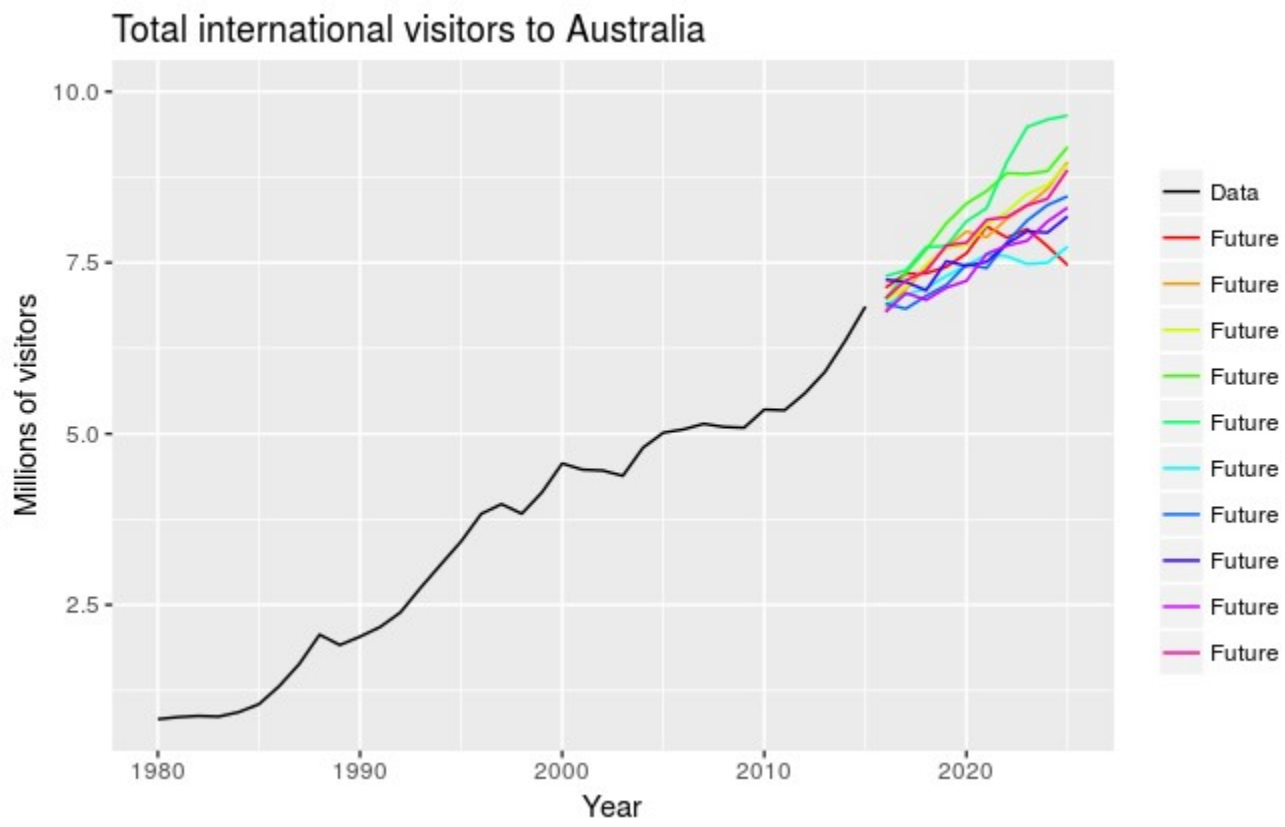


Figure 1.2: Total international visitors to Australia (1980-2015) along with ten possible futures.

When we obtain a forecast, we are estimating the *middle* of the range of possible values the random variable could take. Very often, a forecast is accompanied by a prediction interval giving a *range* of values the random variable could take with relatively high probability. For example, a 95% prediction interval contains a range of values which should include the actual future value with probability 95%.

Instead of plotting individual possible futures as shown in Figure 1.2, we normally show these prediction intervals instead. The plot below shows 80% and 95% intervals for the future Australian international visitors. The blue line is the average of the possible future values, which we call the "point forecasts".

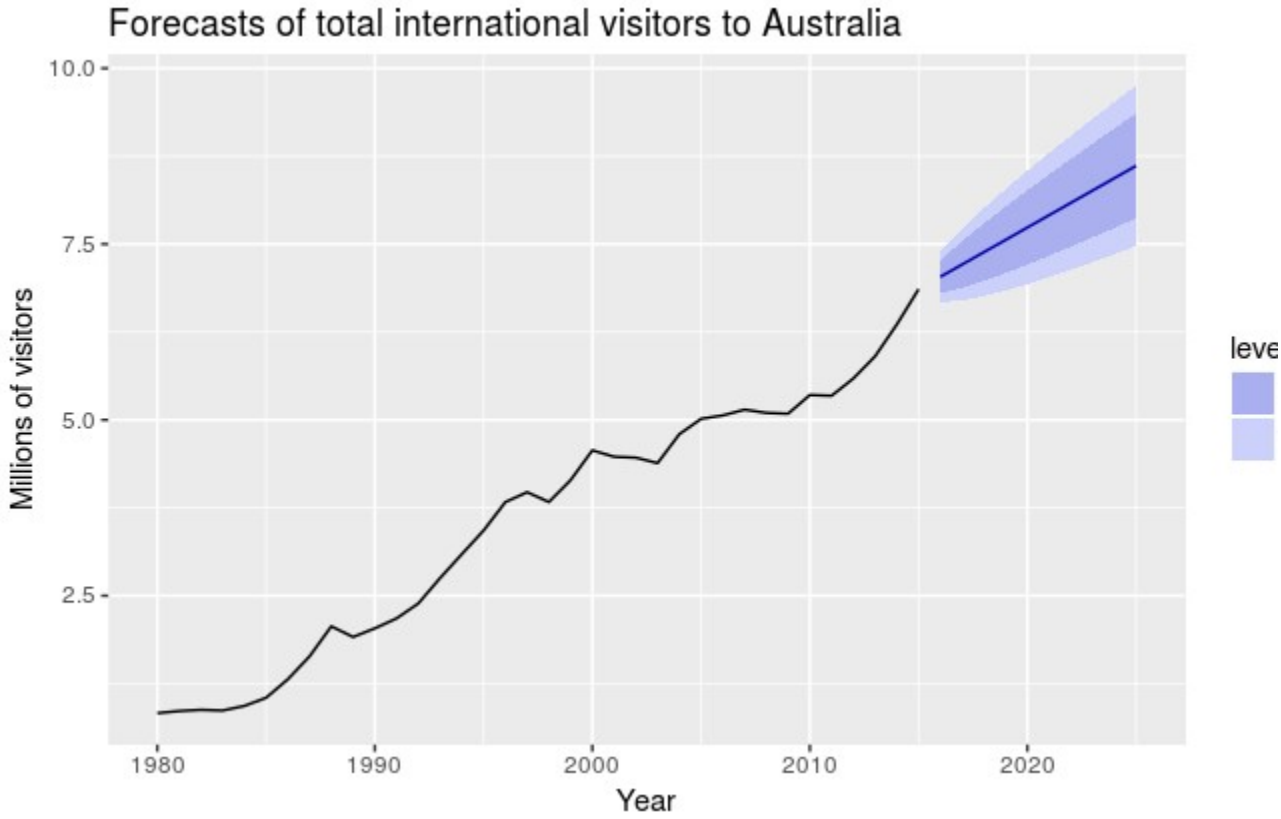


Figure 1.3: Total international visitors to Australia (1980–2015) along with 10-year forecasts and 80% and 95% prediction intervals.

A forecast is always based on some observations. Suppose we denote all the information we have observed as I and we want to forecast yt. We then write yt|I meaning "the random variable yt given what we know in I". The set of values that this random variable could take, along with their relative probabilities, is known as the "probability distribution" of yt|I. In forecasting, we call this the "forecast distribution".

When we talk about the "forecast", we usually mean the average value of the forecast distribution, and we put a "hat" over y to show this. Thus, we write the forecast of yt as ˆyt, meaning the average of the possible values that yt could take given everything we know. Occasionally, we will use ˆyt to refer to the *median* (or middle value) of the forecast distribution instead.

It is often useful to specify exactly what information we have used in calculating the forecast. Then we will write, for example, ˆyt|t−1 to mean the forecast of yt taking account of all previous observations (y1,…,yt−1). Similarly, ˆyT+h|T means the forecast of yT+h taking account of y1, …,yT (i.e., an h-step forecast taking account of all observations up to time T).

1.8 Exercises

1. For cases 3 and 4 in Section 1.5, list the possible predictor variables that might be useful, assuming that the relevant data are available.

2. For case 3 in Section 1.5, describe the five steps of forecasting in the context of this project.

1.9 Further reading

- Armstrong (2001) covers the whole field of forecasting, with each chapter written by different experts. It is highly opinionated at times (and we don't agree with everything in it), but is full of excellent general advice on tackling forecasting problems.

- Ord and Fildes (2012) is a forecasting textbook covering some of the same areas as this book, but with a different emphasis and not focussed around any particular software environment. It is written by two of the most highly respected forecasters in the world, with many decades of experience between them.

References

Armstrong, J S, ed. 2001. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer Academic Publishers.

Ord, J Keith, and Robert Fildes. 2012. *Principles of Business Forecasting*. South-Western College Pub.

Chapter 2 Time series graphics

The first thing to do in any data analysis task is to plot the data. Graphs enable many features of the data to be visualized, including patterns, unusual observations, changes over time, and relationships between variables. The features that are seen in plots of the data must then be incorporated, as far as possible, into the forecasting methods to be used. Just as the type of data determines what forecasting method to use, it also determines what graphs are appropriate.

But before we produce graphs, we need to set up our time series in R.

2.1 `ts` objects

A time series can be thought of as a list of numbers, along with some information about what times those numbers were recorded. This information is stored in a `ts` object in R.

Suppose you have annual observations for the last few years:

Year	Observation
2012	123
2013	39
2014	78
2015	52
2016	110

Year Observation

We turn this into a `ts` object using the `ts` function:

```
y <- ts(c(123,39,78,52,110), start=2012)
```

It is assumed that you have annual data, with one observation per year, so you only need to provide the starting year.

For observations that are more frequent than once per year, you simply add a `frequency` argument. For example, if your monthly data is already stored as a numerical vector `z`, then it can be converted to a `ts` object like this:

```
y <- ts(z, start=2003, frequency=12)
```

Almost all of the data used in this book is already stored as `ts` objects. But if you want to work with your own data, you will need to use the `ts` function before proceeding with the analysis.

Frequency of a time series

The "frequency" is the number of observations before the seasonal pattern repeats.[1] When using the `ts()` function in R, the following choices should be used.

Data	frequency
Annual	1
Quarterly	4
Monthly	12
Weekly	52

Actually, there are not 52 weeks in a year, but 365.25/7 = 52.18 on average, allowing for a leap year every fourth year. But most functions which use `ts` objects require integer frequency.

If the frequency of observations is greater than once per week, then there is usually more than one way of handling the frequency. For example, data with daily observations might have a weekly seasonality (frequency=7) or an annual seasonality (frequency=365.25). Similarly, data that are observed every minute might have an hourly seasonality (frequency=60), a daily seasonality (frequency=24x60=1440), a weekly seasonality (frequency=24x60x7=10080) and an annual seasonality (frequency=24x60x365.25=525960). If you want to use a `ts` object, then you need to decide which of these is the most important.

In chapter 11 we will look at handling these types of multiple seasonality, without having to choose just one of the frequencies.

2.2 Time plots

For time series data, the obvious graph to start with is a time plot. That is, the observations are plotted against the time of observation, with consecutive observations joined by straight lines. Figure 2.1 below shows the weekly economy passenger load on Ansett Airlines between Australia's two largest cities.

```
autoplot(melsyd[,"Economy.Class"]) +
  ggtitle("Economy class passengers: Melbourne-Sydney") +
  xlab("Year") + ylab("Thousands")
```

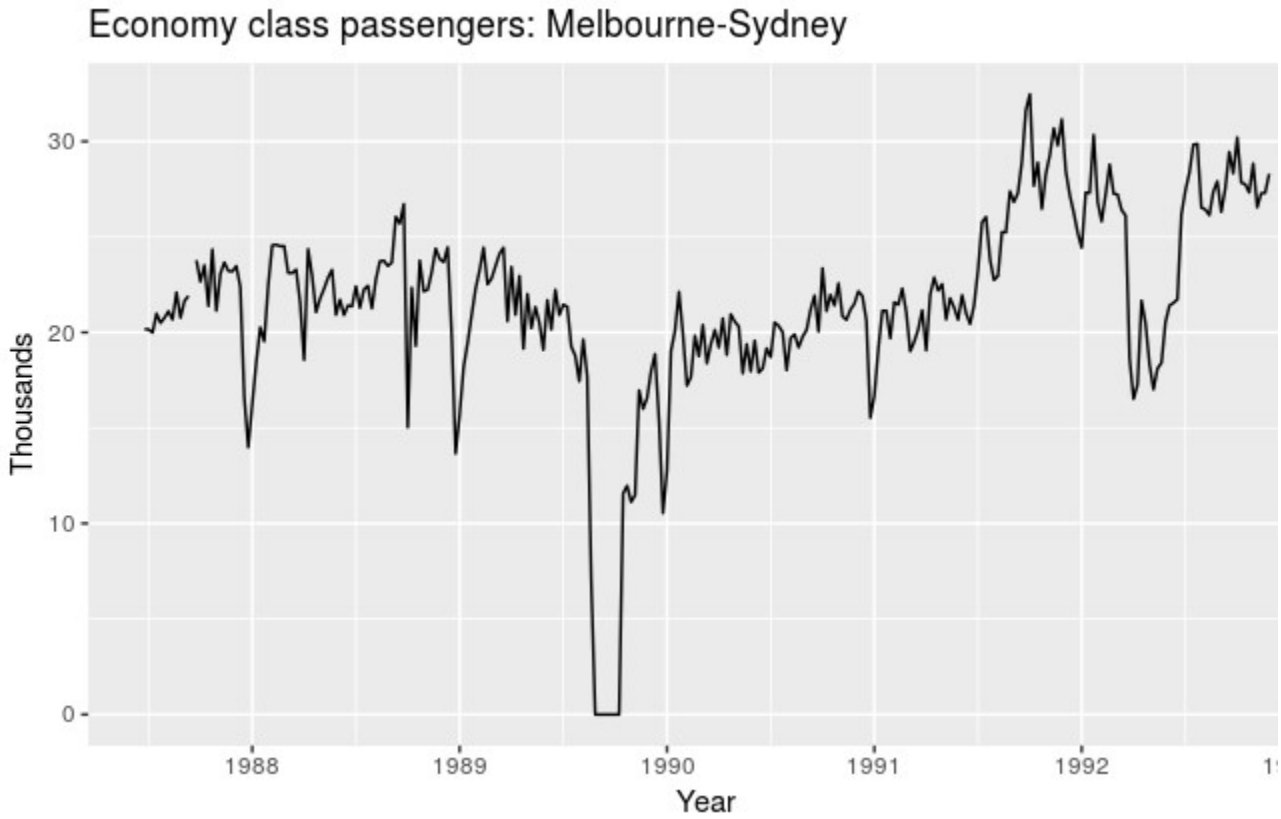


Figure 2.1: Weekly economy passenger load on Ansett Airlines.

We will use the `autoplot` command frequently. It automatically produces an appropriate plot of whatever you pass to it in the first argument. In this case, it recognizes `melsyd[,"Economy.Class"]` as a time series and produces a time plot.

The time plot immediately reveals some interesting features.

- There was a period in 1989 when no passengers were carried — this was due to an industrial dispute.
- There was a period of reduced load in 1992. This was due to a trial in which some economy class seats were replaced by business class seats.
- A large increase in passenger load occurred in the second half of 1991.
- There are some large dips in load around the start of each year. These are due to holiday effects.
- There is a long-term fluctuation in the level of the series which increases during 1987, decreases in 1989, and increases again through 1990 and 1991.
- There are some periods of missing observations.

Any model will need to take all these features into account in order to effectively forecast the passenger load into the future.

A simpler time series is shown in Figure 2.2.

```
autoplot(a10) +
  ggtitle("Antidiabetic drug sales") +
  ylab("$ million") + xlab("Year")
```

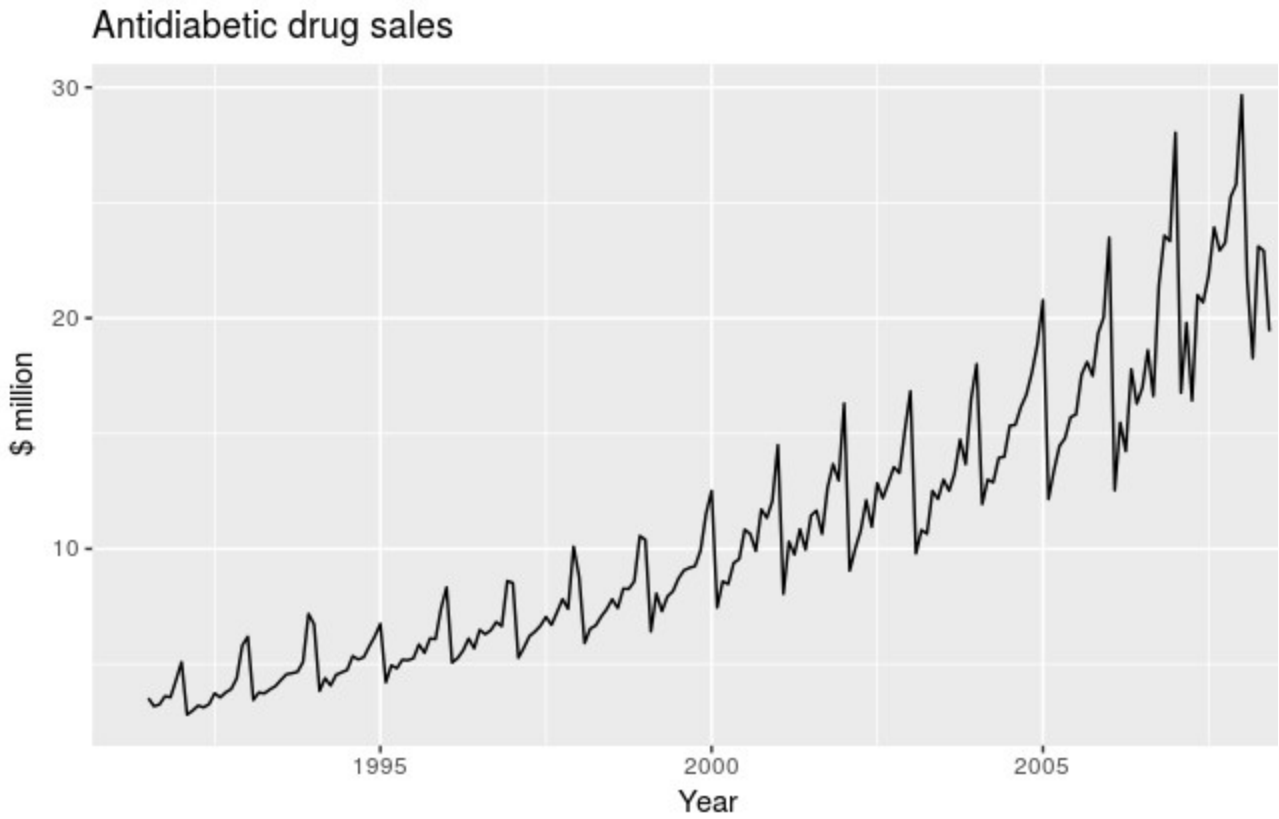


Figure 2.2: Monthly sales of antidiabetic drugs in Australia.

Here, there is a clear and increasing trend. There is also a strong seasonal pattern that increases in size as the level of the series increases. The sudden drop at the end of each year is caused by a government subsidisation scheme that makes it cost-effective for patients to stockpile drugs at the end of the calendar year. Any forecasts of this series would need to capture the seasonal pattern, and the fact that the trend is changing slowly.

2.3 Time series patterns

In describing these time series, we have used words such as “trend” and “seasonal” which need to be defined more carefully.

Trend
: A *trend* exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend as “changing direction”, when it might go from an increasing trend to a decreasing trend. There is a trend in the antidiabetic drug sales data shown in Figure 2.2.

Seasonal
: A *seasonal* pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency. The monthly sales of antidiabetic drugs above shows seasonality which is induced partly by the change in the cost of the drugs at the end of the calendar year.

Cyclic
: A *cycle* occurs when the data exhibit rises and falls that are not of a fixed frequency. These fluctuations are usually due to economic conditions, and are often related to the “business cycle”. The duration of these fluctuations is usually at least 2 years.

Many people confuse cyclic behaviour with seasonal behaviour, but they are really quite different. If the fluctuations are not of a fixed frequency then they are cyclic; if the frequency is unchanging and associated with some aspect of the calendar, then the pattern is seasonal. In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitudes of cycles tend to be more variable than the magnitudes of seasonal patterns. Cycles and seasonality are discussed further in Section 6.1.

Many time series include trend, cycles and seasonality. When choosing a forecasting method, we will first need to identify the time series patterns in the data, and then choose a method that is able to capture the patterns properly.

The following four examples show different combinations of the above components.

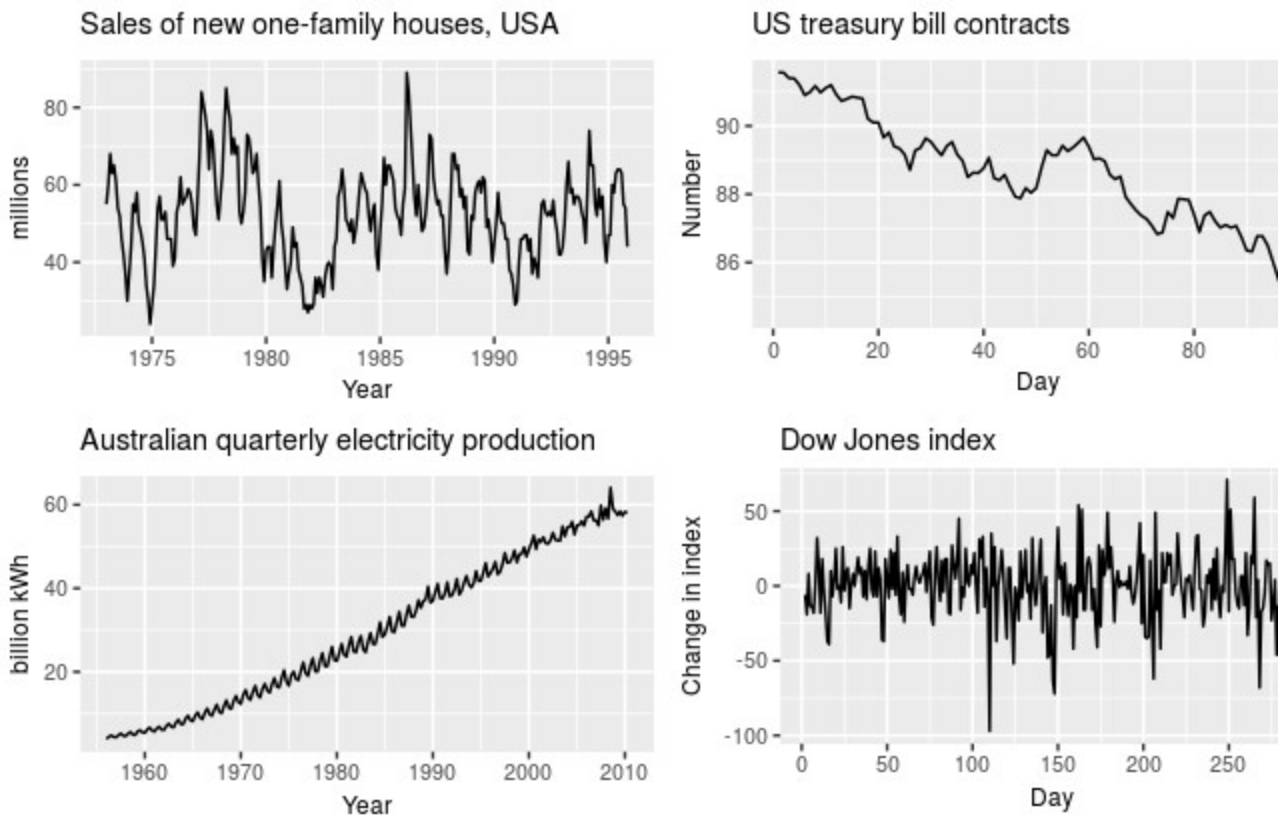


Figure 2.3: Four examples of time series showing different patterns.

1. The monthly housing sales (top left) show strong seasonality within each year, as well as some strong cyclic behaviour with a period of about 6–10 years. There is no apparent trend in the data over this period.

2. The US treasury bill contracts (top right) show results from the Chicago market for 100 consecutive trading days in 1981. Here there is no seasonality, but an obvious downward trend. Possibly, if we had a much longer series, we would see that this downward trend is actually part of a long cycle, but when viewed over only 100 days it appears to be a trend.

3. The Australian monthly electricity production (bottom left) shows a strong increasing trend, with strong seasonality. There is no evidence of any cyclic behaviour here.

4. The daily change in the Dow Jones index (bottom right) has no trend, seasonality or cyclic behaviour. There are random fluctuations which do not appear to be very predictable, and no strong patterns that would help with developing a forecasting model.

2.4 Seasonal plots

A seasonal plot is similar to a time plot except that the data are plotted against the individual "seasons" in which the data were observed. An example is given below showing the antidiabetic drug sales.

```
ggseasonplot(a10, year.labels=TRUE, year.labels.left=TRUE) +
  ylab("$ million") + ggtitle("Seasonal plot: antidiabetic drug sales")
```

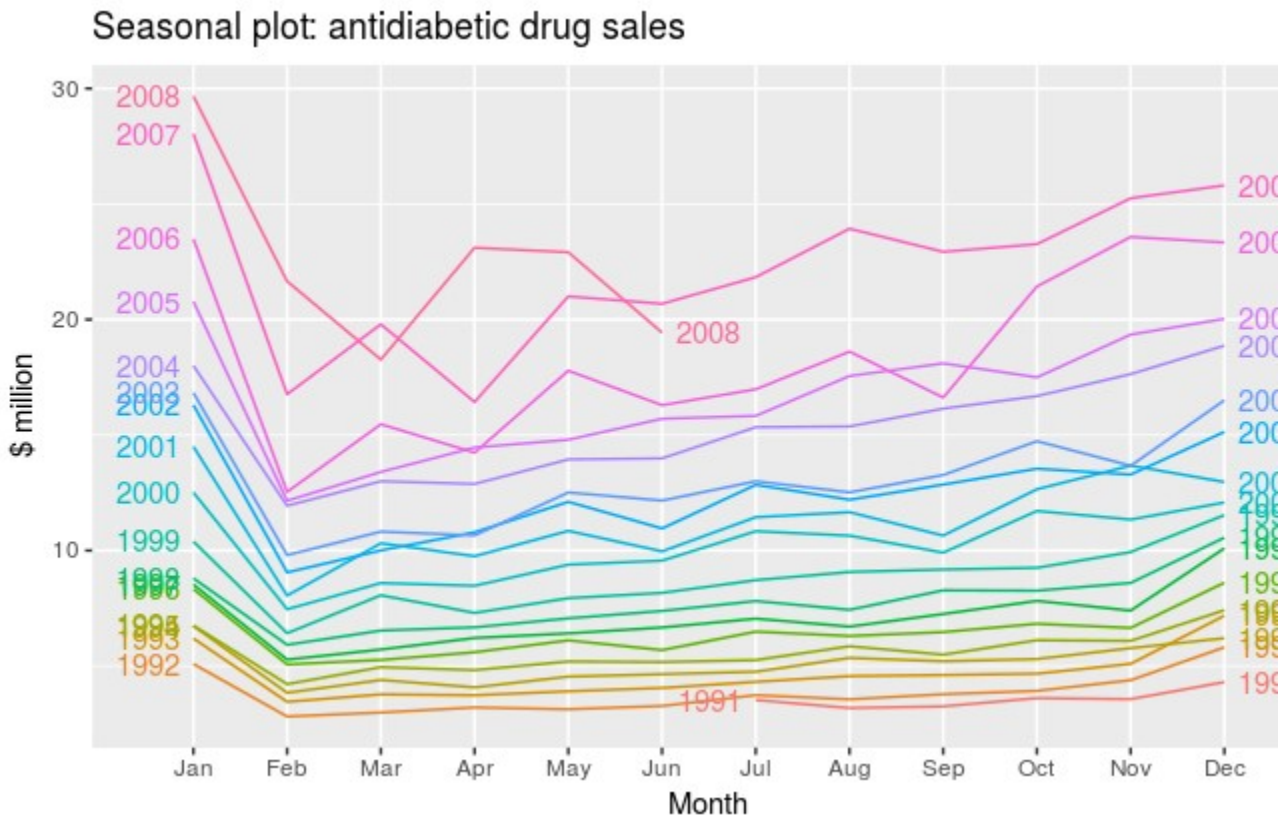


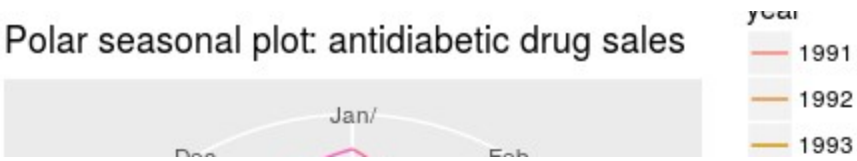
Figure 2.4: Seasonal plot of monthly antidiabetic drug sales in Australia.

These are exactly the same data as were shown earlier, but now the data from each season are overlapped. A seasonal plot allows the underlying seasonal pattern to be seen more clearly, and is especially useful in identifying years in which the pattern changes.

In this case, it is clear that there is a large jump in sales in January each year. Actually, these are probably sales in late December as customers stockpile before the end of the calendar year, but the sales are not registered with the government until a week or two later. The graph also shows that there was an unusually small number of sales in March 2008 (most other years show an increase between February and March). The small number of sales in June 2008 is probably due to incomplete counting of sales at the time the data were collected.

A useful variation on the seasonal plot uses polar coordinates, as shown below.

```
ggseasonplot(a10, polar=TRUE) +
  ylab("$ million") + ggtitle("Polar seasonal plot: antidiabetic drug sales")
```



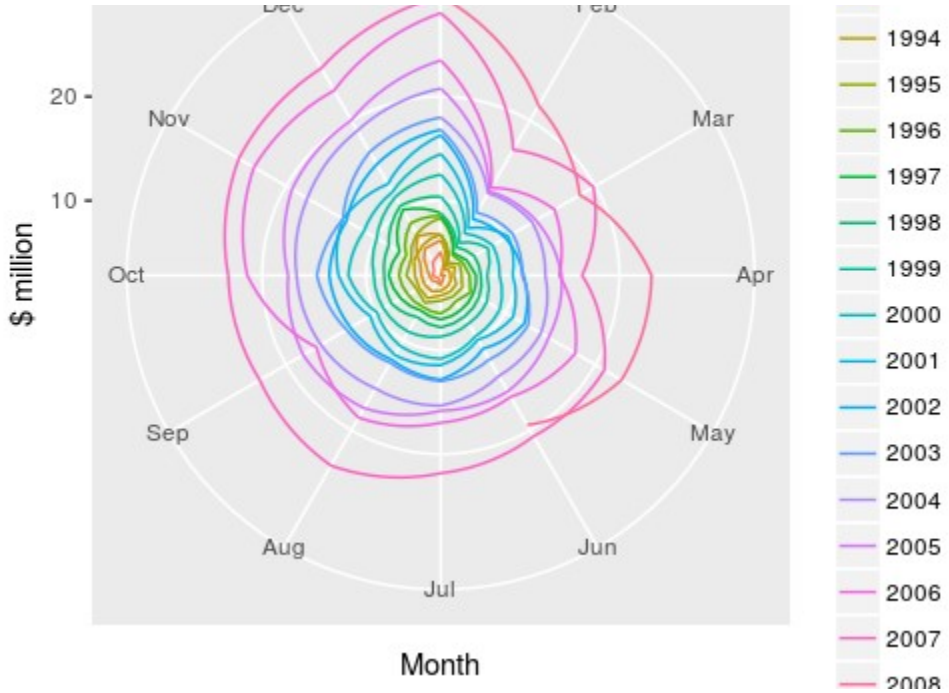


Figure 2.5: Polar seasonal plot of monthly antidiabetic drug sales in Australia.

2.5 Seasonal subseries plots

An alternative plot that emphasises the seasonal patterns is where the data for each season are collected together in separate mini time plots.

```
ggsubseriesplot(a10) + ylab("$ million") +
  ggtitle("Seasonal subseries plot: antidiabetic drug sales")
```

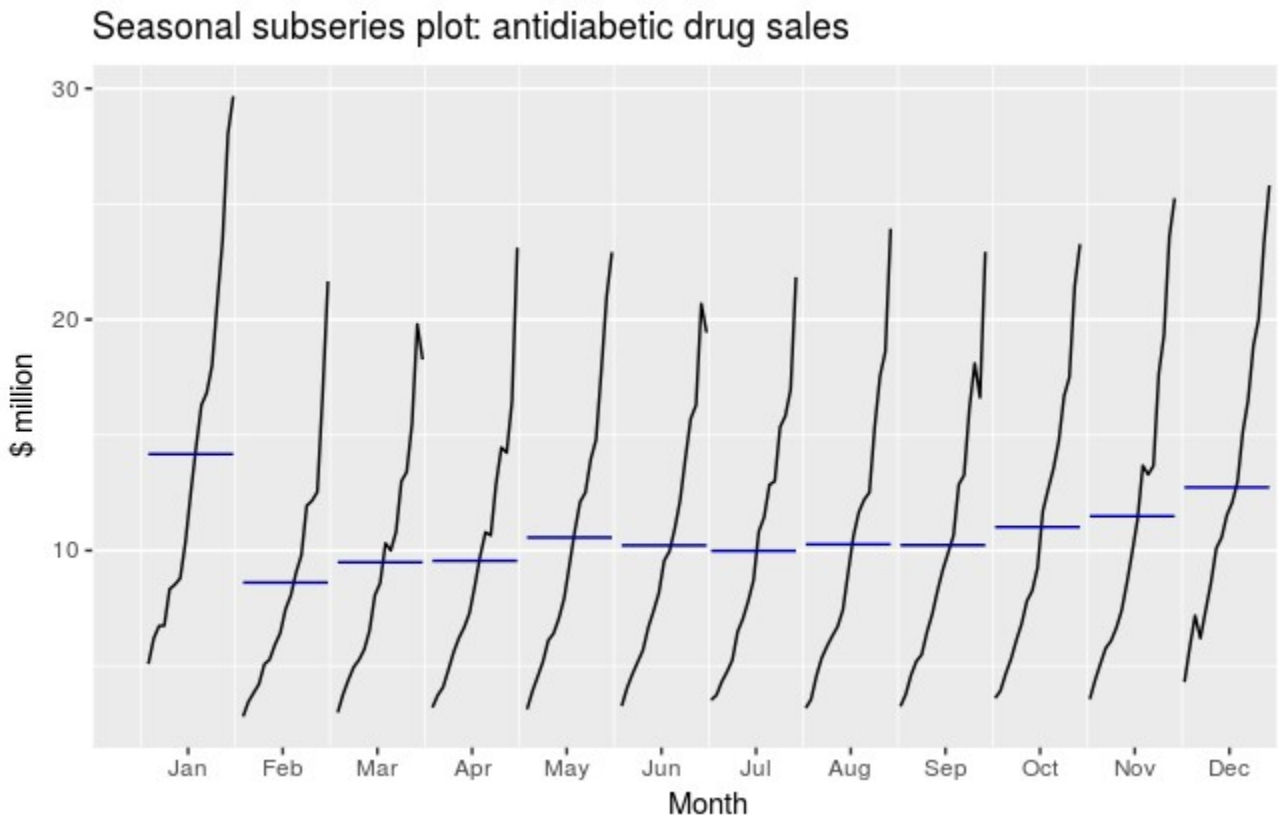


Figure 2.6: Seasonal subseries plot of monthly antidiabetic drug sales in Australia.

The horizontal lines indicate the means for each month. This form of plot enables the underlying seasonal pattern to be seen clearly, and also shows the changes in seasonality over time. It is especially useful in identifying changes within particular seasons. In this example, the plot is not particularly revealing; but in some cases, this is the most useful way of viewing seasonal changes over time.

2.6 Scatterplots

The graphs discussed so far are useful for visualizing individual time series. It is also useful to explore relationships *between* time series.

Figure 2.7 shows two time series: monthly takings (in $million) from accommodation at hotels, motels and guest houses in Victoria (top) and total room nights for each corresponding month (in thousands). “Room nights” is the total number of rooms booked multiplied by the number of nights people stayed in those rooms.

```
autoplot(motel[,2:1]/1000, facet=TRUE) +
  xlab("Year") + ylab("") +
  ggtitle("Total monthly accommodation: Victoria, Australia")
```

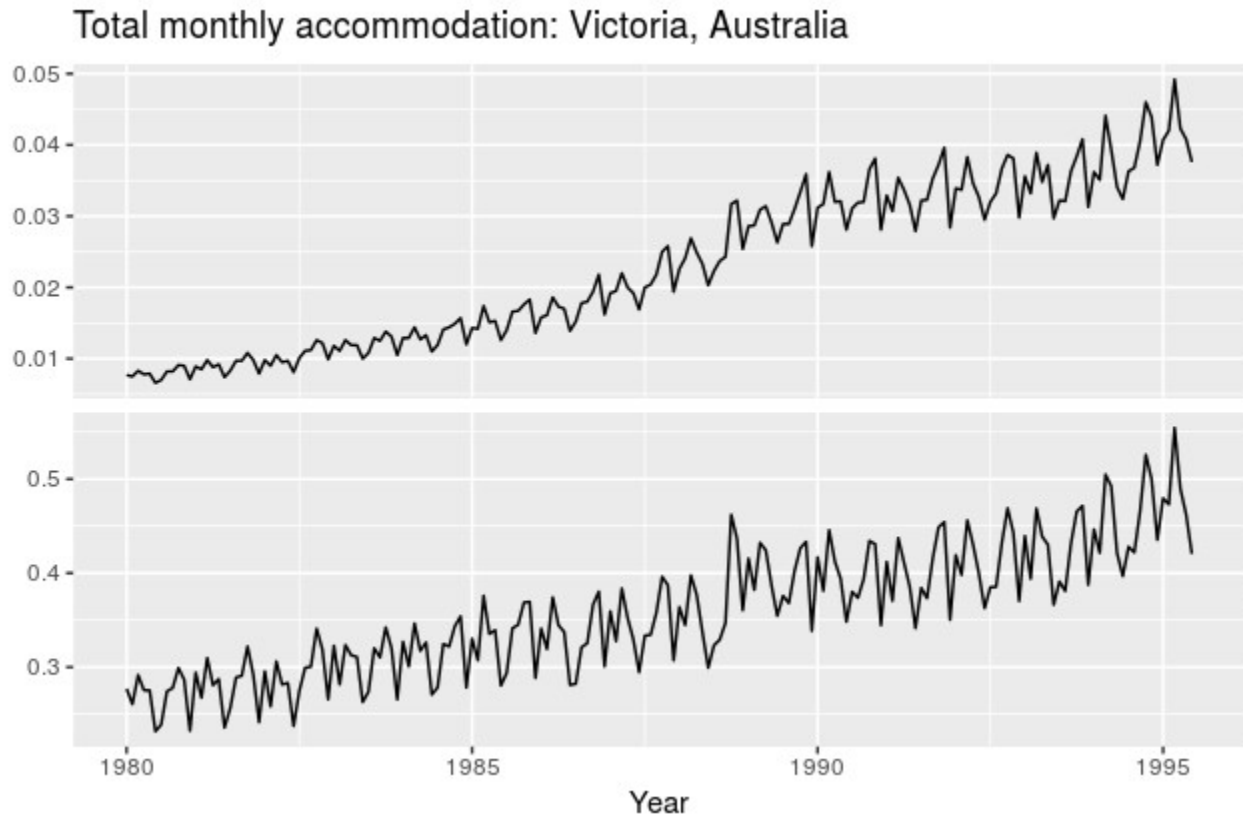


Figure 2.7: Monthly takings and room nights for accommodation in Victoria, Australia.

We can study the relationship between takings and room nights by plotting one series against the other.

```
qplot(Roomnights/1000, Takings/1000, data=as.data.frame(motel)) +
  ylab("Takings ($million)") + xlab("Room nights (thousands)")
```

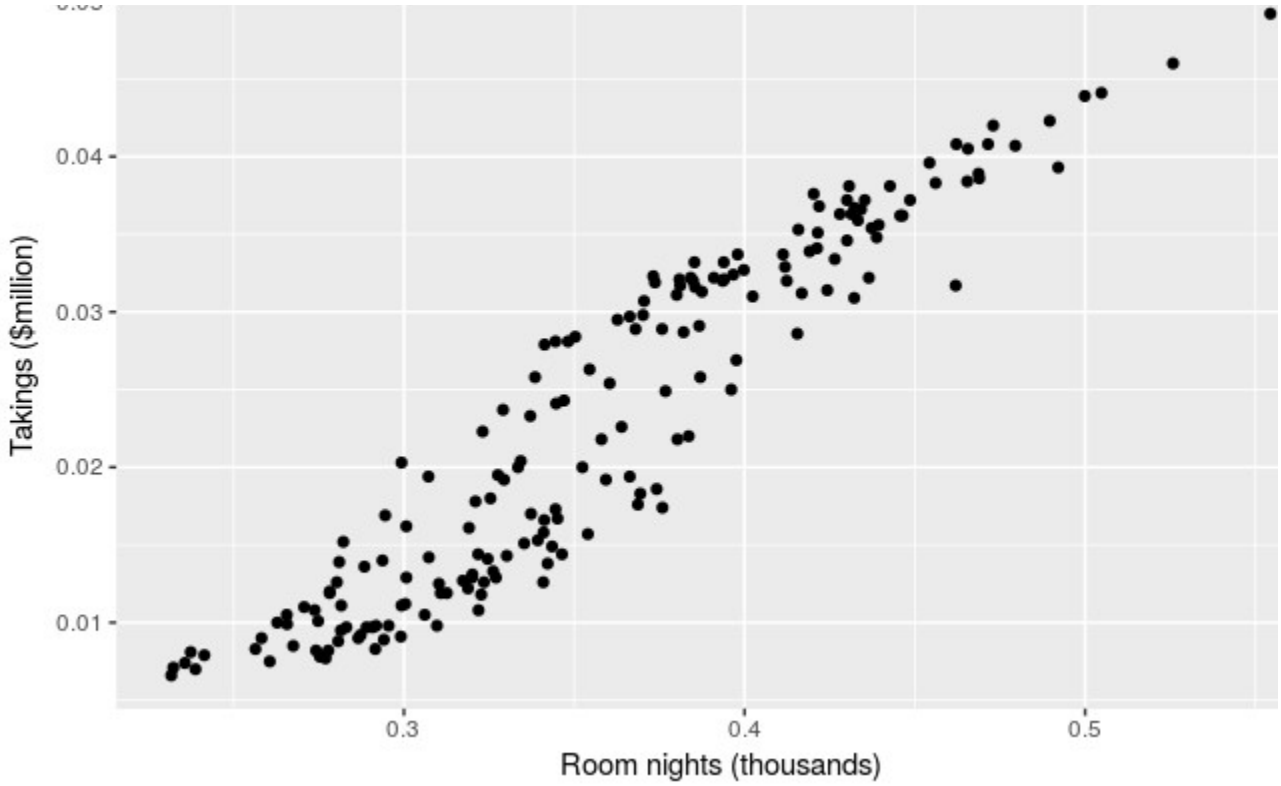


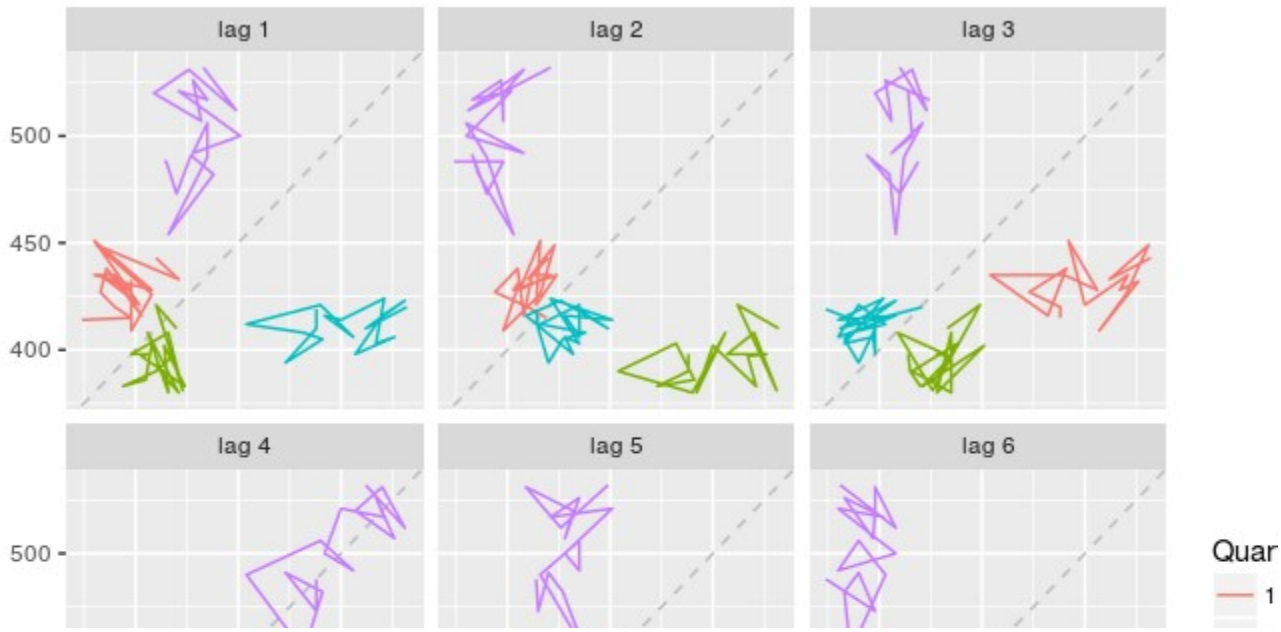
Figure 2.8: Monthly takings plotted against room nights for accommodation in Victoria, Australia.

This scatterplot helps us to visualize the relationship between the variables.

2.7 Lag plots

Figure 2.9 displays scatterplots of quarterly Australian beer production, where the horizontal axis shows lagged values of the time series. Each graph shows yt plotted against yt−k for different values of k.

```
beer2 <- window(ausbeer, start=1992)
gglagplot(beer2)
```



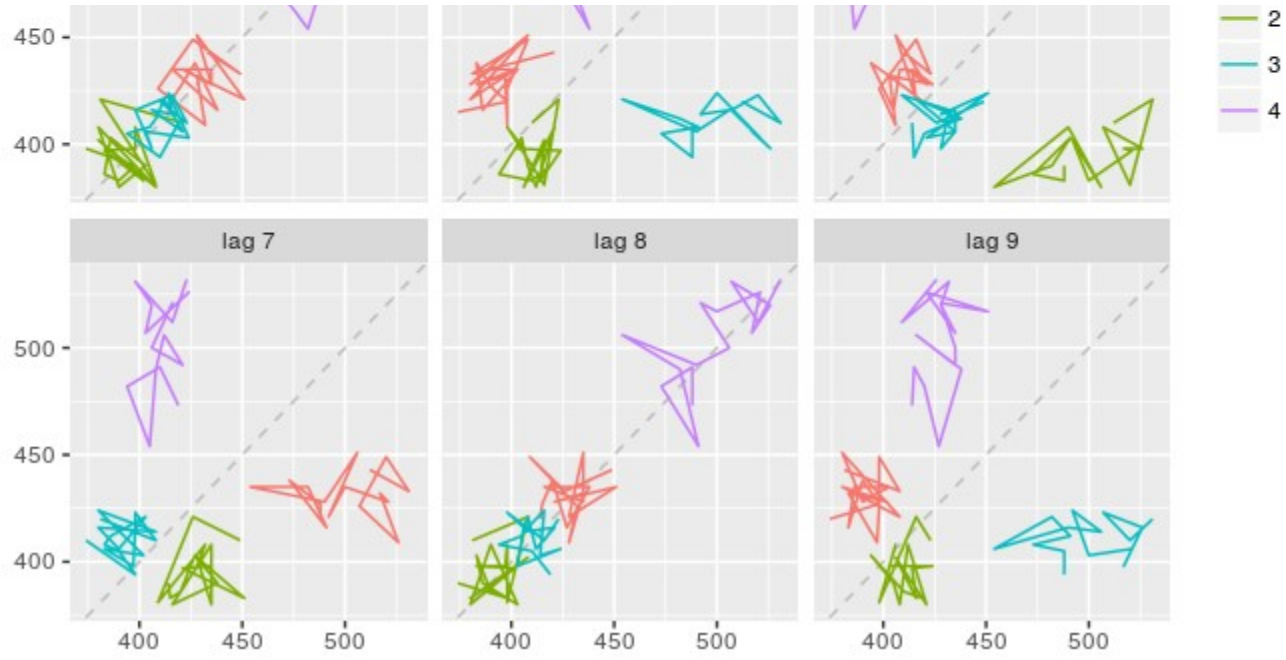


Figure 2.9: Lagged scatterplots for quarterly beer production.

Here the colours indicate the quarter of the variable on the vertical axis. The relationship is strongly positive at lags 4 and 8, reflecting the strong quarterly seasonality in the data. The negative relationship seen for lags 2 and 6 occurs because peaks (in Q4) are plotted against troughs (in Q2)

The `window` function used here is very useful when extracting a portion of a time series. In this case, we have extracted the data from `ausbeer`, beginning in 1992.

2.8 Autocorrelation

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between *lagged values* of a time series.

There are several autocorrelation coefficients, corresponding to each panel in the lag plot. For example, r1 measures the relationship between yt and yt−1, r2 measures the relationship between yt and yt−2, and so on.

The value of rk can be written as rk=T∑t=k+1(yt−¯y)(yt−k−¯y)T∑t=1(yt−¯y)2, where T is the length of the time series.

The first nine autocorrelation coefficients for the beer production data are given in the following table.

r1	r2	r3	r4	r5	r6	r7	r8	r9
-0.102	-0.657	-0.060	0.869	-0.089	-0.635	-0.054	0.832	-0.108

These correspond to the nine scatterplots in Figure 2.9. The autocorrelation coefficients are normally plotted to form the *autocorrelation function* or ACF. The plot is also known as a *correlogram*.

```
ggAcf(beer2)
```

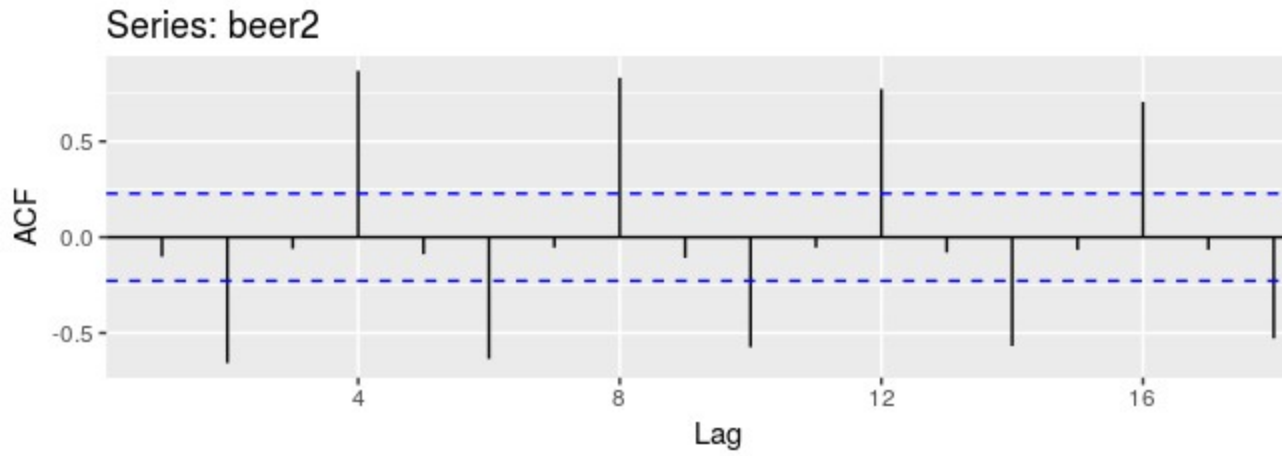


Figure 2.10: Autocorrelation function of quarterly beer production.

In this graph:

- r4 is higher than for the other lags. This is due to the seasonal pattern in the data: the peaks tend to be four quarters apart and the troughs tend to be two quarters apart.
- r2 is more negative than for the other lags because troughs tend to be two quarters behind peaks.
- The dashed blue lines indicate whether the correlations are significantly different from zero. These are explained in Section 2.9.

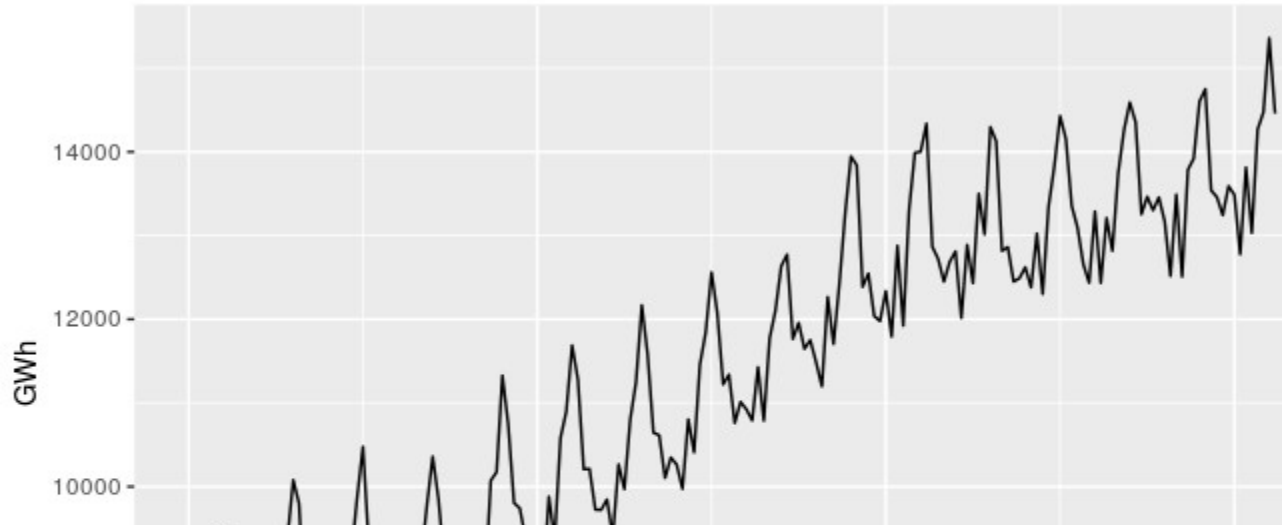
Trend and seasonality in ACF plots

When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase.

When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags.

When data are both trended and seasonal, you see a combination of these effects, as illustrated in Figure 2.12.

```
aelec <- window(elec, start=1980)
autoplot(aelec) + xlab("Year") + ylab("GWh")
```



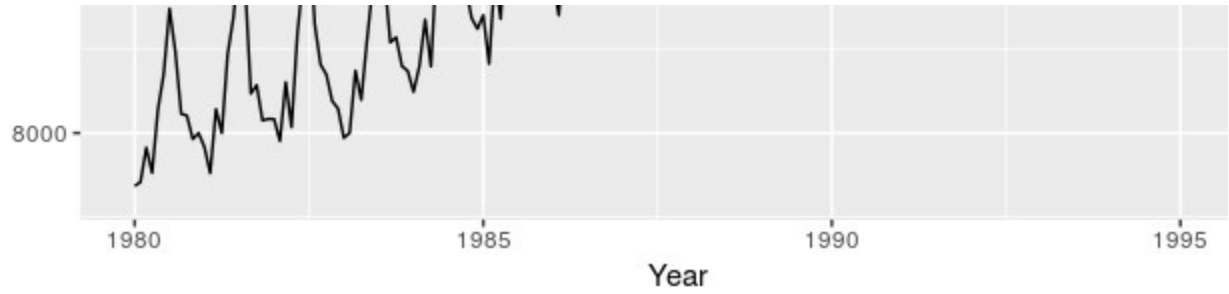


Figure 2.11: Monthly Australian electricity demand from 1980–1995.

```
ggAcf(aelec, lag=48)
```

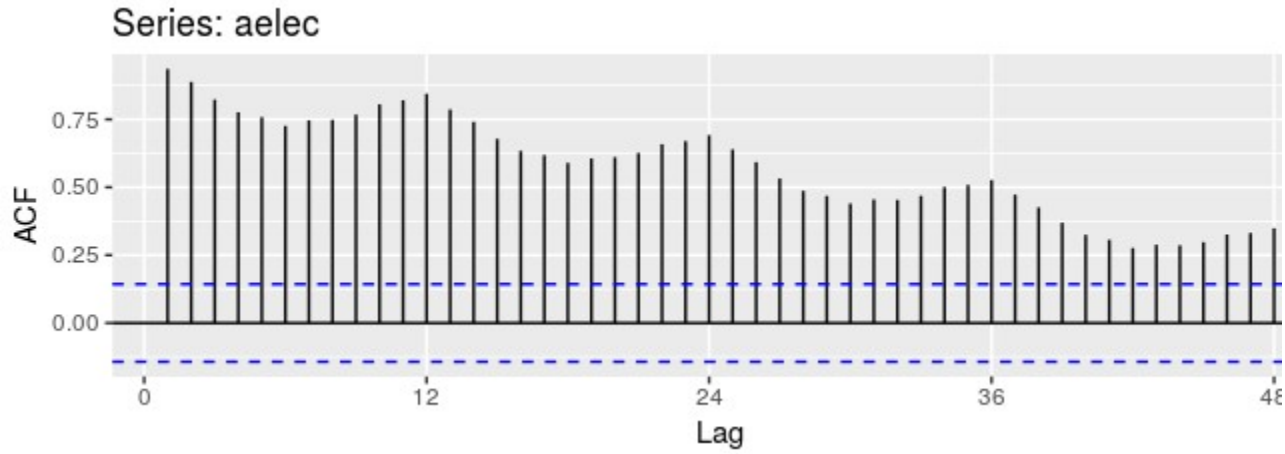


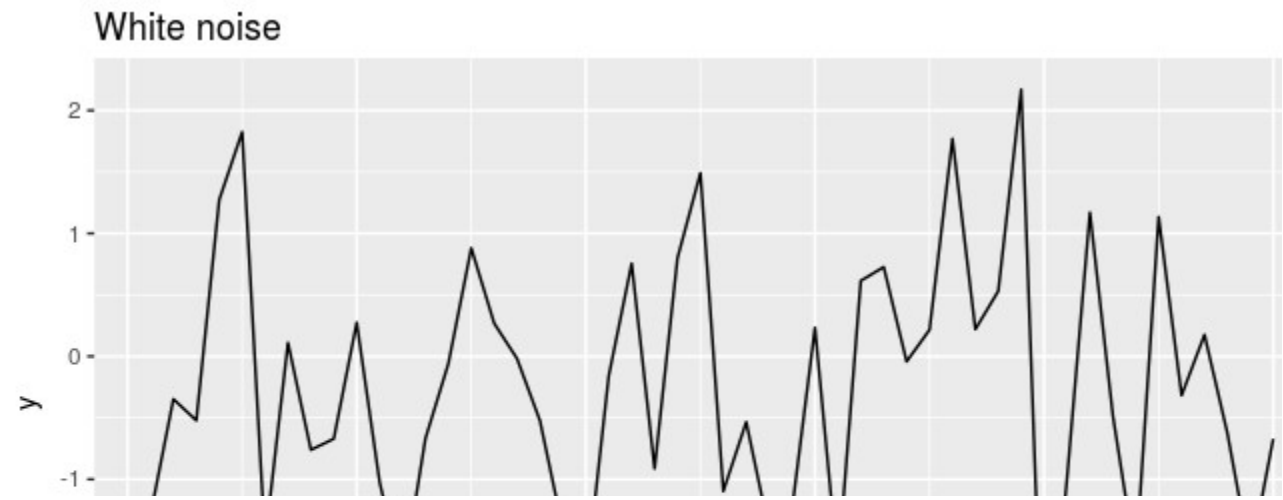
Figure 2.12: ACF of monthly Australian electricity demand.

The slow decrease in the ACF as the lags increase is due to the trend, while the “scalloped” shape is due the seasonality.

2.9 White noise

Time series that show no autocorrelation are called “white noise”. Figure 2.13 gives an example of a white noise series.

```
set.seed(30)
y <- ts(rnorm(50))
autoplot(y) + ggtitle("White noise")
```



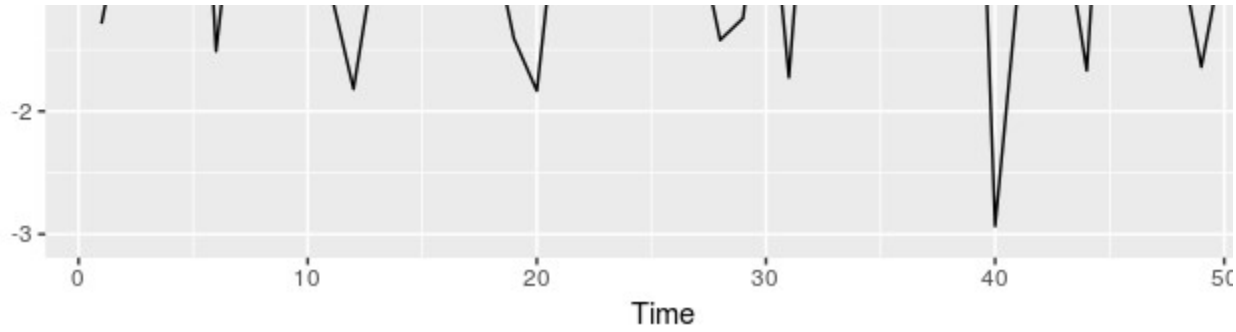


Figure 2.13: A white noise time series.

```
ggAcf(y)
```

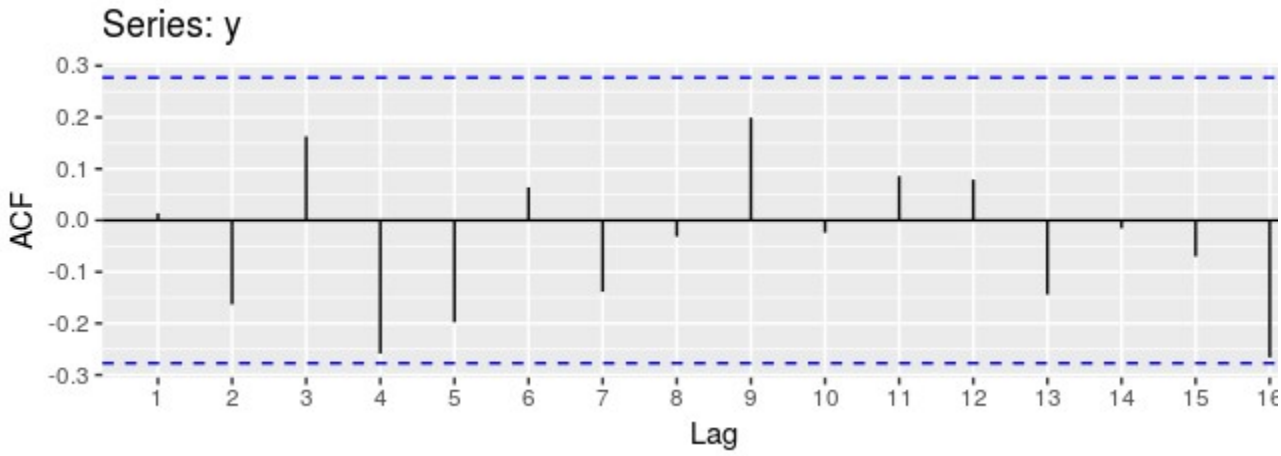


Figure 2.14: Autocorrelation function for the white noise series.

For white noise series, we expect each autocorrelation to be close to zero. Of course, they will not be exactly equal to zero as there is some random variation. For a white noise series, we expect 95% of the spikes in the ACF to lie within $\pm 2/\sqrt{T}$ where T is the length of the time series. It is common to plot these bounds on a graph of the ACF (the blue dashed lines above). If one or more large spikes are outside these bounds, or if substantially more than 5% of spikes are outside these bounds, then the series is probably not white noise.

In this example, T=50 and so the bounds are at $\pm 2/\sqrt{50} = \pm 0.28$. All of the autocorrelation coefficients lie within these limits, confirming that the data are white noise.

2.10 Exercises

1. Download some monthly Australian retail data from http://robjhyndman.com/data/retail.xlsx. These represent retail sales in various categories for different Australian states, and are stored in a MS-Excel file.

    1. You can read the data into R with the following script:

        ```
        retaildata <- readxl::read_excel("retail.xlsx", skip = 1)
        ```

        You may need to first install the `readxl` package.

    2. Select one of the time series as follows (but replace the column name with your own chosen column):

```
mytimeseries <- ts(retaildata[,"A3349873A"], frequency=12, start=c(1982,4))
```

3. Explore your chosen retail time series using the following functions:

   `autoplot`, `ggseasonplot`, `ggsubseriesplot`, `gglagplot`, `ggAcf`

   Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

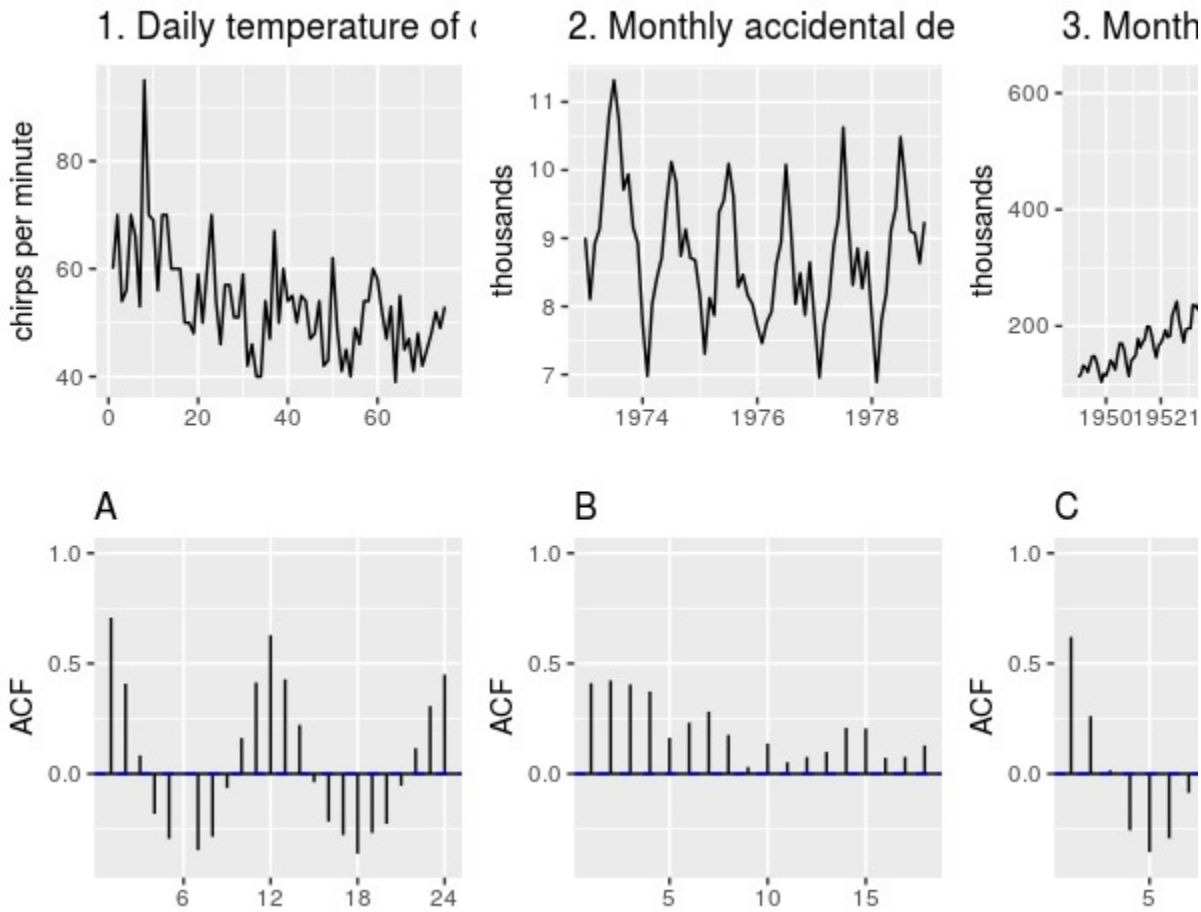
2. Repeat for the following series:

   `bicoal`, `chicken`, `dole`, `usdeaths`, `bricksq`, `lynx`, `ibmclose`

   Use the help files to find out what the series are.

3. The `arrivals` data set comprises quarterly international arrivals (in thousands) to Australia from Japan, New Zealand, UK and the US. Use `autoplot` and `ggseasonplot` and compare the differences between the arrivals from these four countries. Can you identify any unusual observations?

4. The following time plots and ACF plots correspond to four different time series. Your task is to match each time plot in the first row with one of the ACF plots in the second row.



2.11 Further reading

- Cleveland (1993) is a classic book on the principles of visualization for data analysis. While it is more than 20 years old, the ideas are timeless.
- Unwin (2015) is a modern introduction to graphical data analysis using R. It does not have much information on time series graphics, but plenty of excellent general advice on using graphics for data analysis.

References

Cleveland, William S. 1993. *Visualizing Data*. Hobart Press.

Unwin, Antony. 2015. *Graphical Data Analysis with R*. Chapman; Hall/CRC.

Chapter 3 The forecaster's toolbox

In this chapter, we discuss some general tools that are useful for many different forecasting situations. We will describe some benchmark forecasting methods, ways of making the forecasting task simpler using transformations and adjustments, methods for checking whether a forecasting method has adequately utilized the available information, and techniques for computing prediction intervals.

Each of the tools discussed in this chapter will be used repeatedly in subsequent chapters as we develop and explore a range of forecasting methods.

3.1 Some simple forecasting methods

Some forecasting methods are very simple and surprisingly effective. Here are four methods that we will use as benchmarks for other forecasting methods.

Average method

Here, the forecasts of all future values are equal to the mean of the historical data. If we let the historical data be denoted by y1,…,yT, then we can write the forecasts as ˆyT+h|T=¯y=(y1+⋯+yT)/T. The notation ˆyT+h|T is a short-hand for the estimate of yT+h based on the data y1,…,yT.

```
meanf(y, h)
# y contains the time series
# h is the forecast horizon
```

Naïve method

Naïve forecasts are where all forecasts are simply set to be the value of the last observation. That is, ˆyT+h|T=yT. This method works remarkably well for many economic and financial time series.

```
naive(y, h)
rwf(y, h) # Alternative
```

Seasonal naïve method

A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year). Formally, the forecast for time T+h is written as $\hat{y}_{T+h|T} = y_{T+h-km}$, where m= the seasonal period, $k = \lfloor (h-1)/m \rfloor + 1$, and $\lfloor u \rfloor$ denotes the integer part of u. That looks more complicated than it really is. For example, with monthly data, the forecast for all future February values is equal to the last observed February value. With quarterly data, the forecast of all future Q2 values is equal to the last observed Q2 value (where Q2 means the second quarter). Similar rules apply for other months and quarters, and for other seasonal periods.

```
snaive(y, h)
```

Drift method

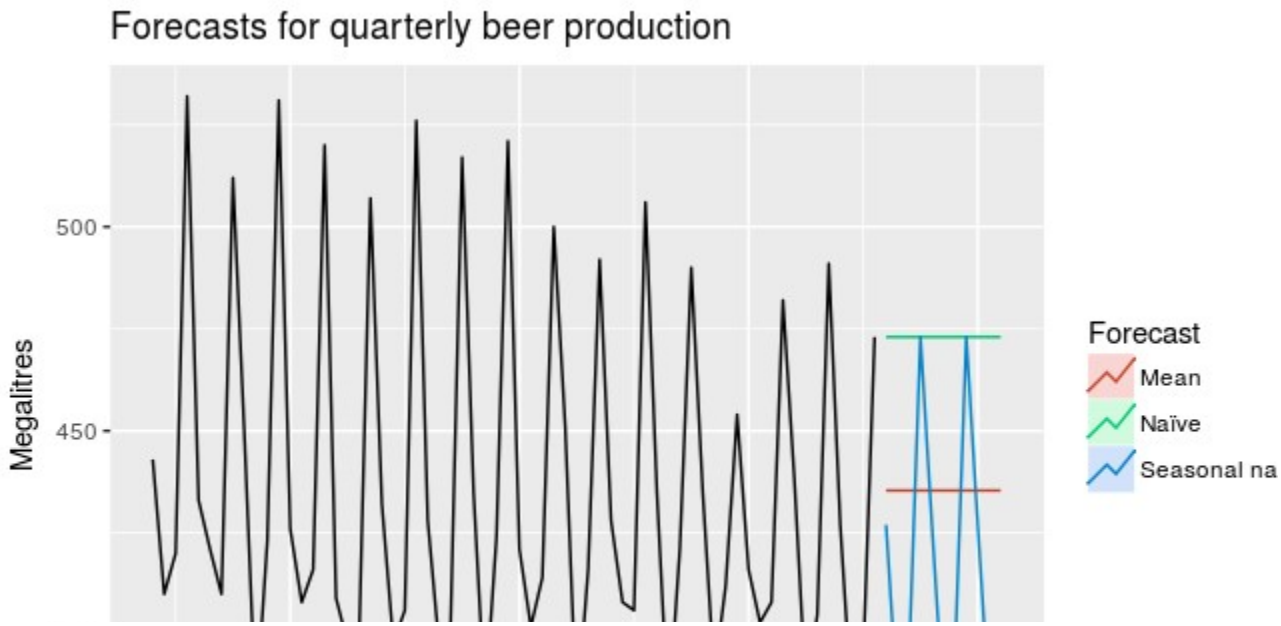
A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the "drift") is set to be the average change seen in the historical data. Thus the forecast for time T+h is given by $\hat{y}_{T+h|T} = y_{T+h} \frac{T-1}{\sum_{t=2}(y_t - y_{t-1})} = y_{T+h}(y_T - y_1 T-1)$. This is equivalent to drawing a line between the first and last observations, and extrapolating it into the future.

```
rwf(y, h, drift=TRUE)
```

Examples

Figure 3.1 shows the first three methods applied to the quarterly beer production data.

```
# Set training data from 1992-2007
beer2 <- window(ausbeer,start=1992,end=c(2007,4))
# Plot some forecasts
autoplot(beer2) +
  forecast::autolayer(meanf(beer2, h=11), PI=FALSE, series="Mean") +
  forecast::autolayer(naive(beer2, h=11), PI=FALSE, series="Naïve") +
  forecast::autolayer(snaive(beer2, h=11), PI=FALSE, series="Seasonal naïve") +
  ggtitle("Forecasts for quarterly beer production") +
  xlab("Year") + ylab("Megalitres") +
  guides(colour=guide_legend(title="Forecast"))
```



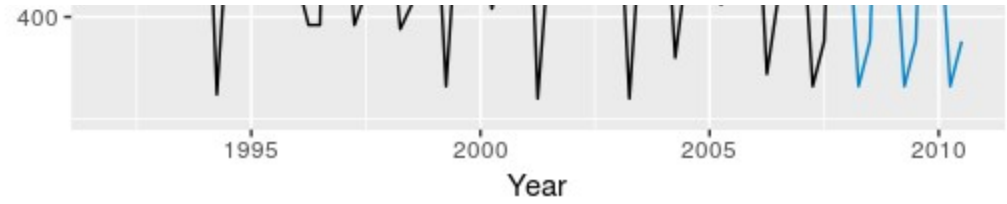


Figure 3.1: Forecasts of Australian quarterly beer production.

In Figure 3.2, the non-seasonal methods were applied to a series of 250 days of the Dow Jones Index.

```
# Set training data to first 250 days
dj2 <- window(dj,end=250)
# Plot some forecasts
autoplot(dj2) +
  forecast::autolayer(meanf(dj2, h=42), PI=FALSE, series="Mean") +
  forecast::autolayer(rwf(dj2, h=42), PI=FALSE, series="Naïve") +
  forecast::autolayer(rwf(dj2, drift=TRUE, h=42), PI=FALSE, series="Drift") +
  ggtitle("Dow Jones Index (daily ending 15 Jul 94)") +
  xlab("Day") + ylab("") +
  guides(colour=guide_legend(title="Forecast"))
```

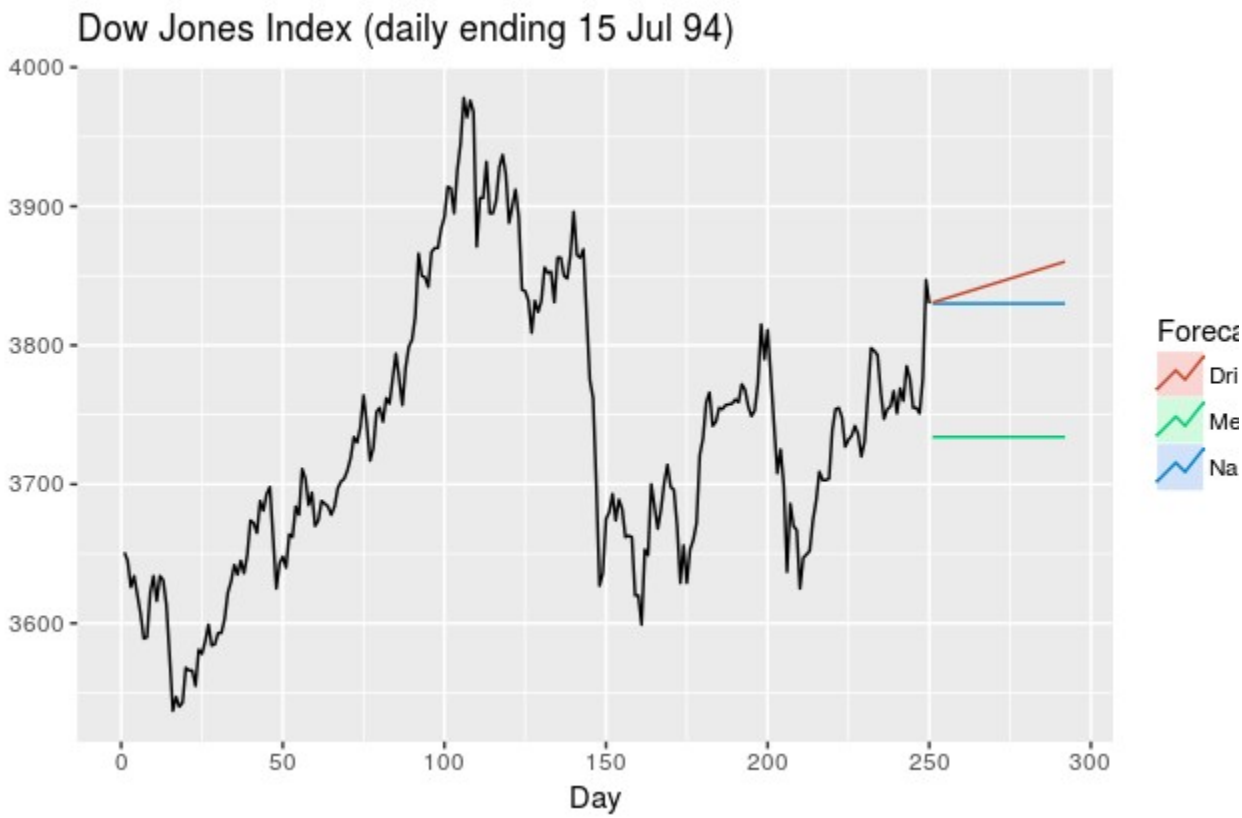


Figure 3.2: Forecasts based on 250 days of the Dow Jones Index.

Sometimes one of these simple methods will be the best forecasting method available; but in many cases, these methods will serve as benchmarks rather than the method of choice. That is, any forecasting methods we develop will be compared to these simple methods to ensure that the new method is better than these simple alternatives. If not, the new method is not worth considering.

3.2 Transformations and adjustments

Adjusting the historical data can often lead to a simpler forecasting model. Here, we deal with four kinds of adjustments: calendar adjustments, population adjustments, inflation adjustments and mathematical transformations. The purpose of these adjustments and transformations is to simplify the patterns in the historical data by removing known sources of variation or by making the pattern more consistent across the whole data set. Simpler patterns usually lead to more accurate forecasts.

Calendar adjustments

Some of the variation seen in seasonal data may be due to simple calendar effects. In such cases, it is usually much easier to remove the variation before fitting a forecasting model. The `monthdays` function will compute the number of days in each month or quarter.

For example, if you are studying the monthly milk production on a farm, there will be variation between the months simply because of the different numbers of days in each month, in addition to the seasonal variation across the year.

```
cbind(Monthly = milk, DailyAverage=milk/monthdays(milk)) %>%
  autoplot(facet=TRUE) + xlab("Years") + ylab("Pounds") +
    ggtitle("Milk production per cow")
```

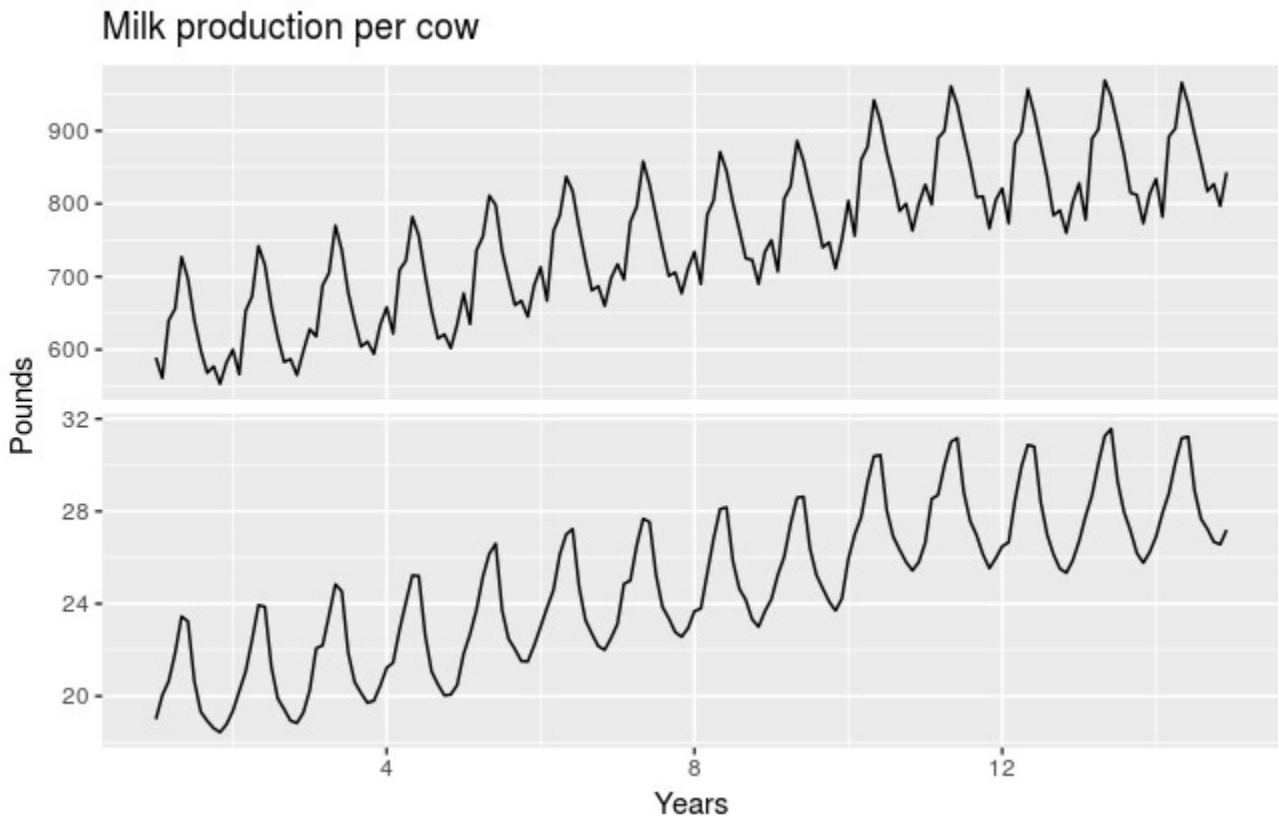


Figure 3.3: Monthly milk production per cow. Source: Cryer and Chan (2008).

Notice how much simpler the seasonal pattern is in the average daily production plot compared to the average monthly production plot. By looking at the average daily production instead of the average monthly production, we effectively remove the variation due to the different month lengths. Simpler patterns are usually easier to model and lead to more accurate forecasts.

A similar adjustment can be done for sales data when the number of trading days in each month varies. In this case, the sales per trading day can be modelled instead of the total sales for each month.

3.3 Residual diagnostics

Fitted values

Each observation in a time series can be forecast using all previous observations. We call these "fitted values" and they are denoted by $\hat{y}_{t|t-1}$, meaning the forecast of y_t based on observations $y_1,\ldots,y_{t-1}$. We use these so often, we sometimes drop part of the subscript and just write $\hat{y}_t$ instead of $\hat{y}_{t|t-1}$. Fitted values always involve one-step forecasts.

Actually, fitted values are often not true forecasts because any parameters involved in the forecasting method are estimated using all available observations in the time series, including future observations. For example, if we use the average method, the fitted values are given by $\hat{y}_t=\hat{c}$ where $\hat{c}$ is the average computed over all available observations, including those at times *after* t. Similarly, for the drift method, the drift parameter is estimated using all available observations. In this case, the fitted values are given by $\hat{y}_t=y_{t-1}+\hat{c}$ where $\hat{c}=(y_T-y_1)/(T-1)$. In both cases, there is a parameter to be estimated from the data. The "hat" above the c reminds us that this is an estimate. When the estimate of c involves observations after time t, the fitted values are not true forecasts. On the other hand, naïve or seasonal naïve forecasts do not involve any parameters, and so fitted values are true forecasts in such cases.

Residuals

The "residuals" in a time series model are what is left over after fitting a model. For many (but not all) time series models, the residuals are equal to the difference between the observations and the corresponding fitted values: $e_t=y_t-\hat{y}_t$.

Residuals are useful in checking whether a model has adequately captured the information in the data. A good forecasting method will yield residuals with the following properties:

1. The residuals are uncorrelated. If there are correlations between residuals, then there is information left in the residuals which should be used in computing forecasts.
2. The residuals have zero mean. If the residuals have a mean other than zero, then the forecasts are biased.

Any forecasting method that does not satisfy these properties can be improved. However, that does not mean that forecasting methods that satisfy these properties cannot be improved. It is possible to have several different forecasting methods for the same data set, all of which satisfy these properties. Checking these properties is important in order to see whether a method is using all of the available information, but it is not a good way to select a forecasting method.

If either of these properties is not satisfied, then the forecasting method can be modified to give better forecasts. Adjusting for bias is easy: if the residuals have mean m, then simply add m to all forecasts and the bias problem is solved. Fixing the correlation problem is harder, and we will not address it until Chapter @ref(????).

In addition to these essential properties, it is useful (but not necessary) for the residuals to also have the following two properties.

3. The residuals have constant variance.

4. The residuals are normally distributed.

These two properties make the calculation of prediction intervals easier (see the next section for an example). However, a forecasting method that does not satisfy these properties cannot necessarily be improved. Sometimes applying a Box-Cox transformation may assist with these properties, but otherwise there is usually little that you can do to ensure that your residuals have constant variance and a normal distribution. Instead, an alternative approach to obtaining prediction intervals is necessary. Again, we will not address how to do this until later in the book.

3.4 Evaluating forecast accuracy

Training and test sets

It is important to evaluate forecast accuracy using genuine forecasts. Consequently, the size of the residuals is not a reliable indication of how large true forecast errors are likely to be. The accuracy of forecasts can only be determined by considering how well a model performs on new data that were not used when fitting the model.

When choosing models, it is common practice to separate the available data into two portions, training and test data, where the training data is used to estimate any parameters of a forecasting method and the test data is used to evaluate its accuracy. Because the test data is not used in determining the forecasts, it should provide a reliable indication of how well the model is likely to forecast on new data.

The size of the test set is typically about 20% of the total sample, although this value depends on how long the sample is and how far ahead you want to forecast. The test set should ideally be at least as large as the maximum forecast horizon required. The following points should be noted.

- A model which fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.

Some references describe the test set as the “hold-out set” because these data are “held out” of the data used for fitting. Other references call the training set the “in-sample data” and the test set the “out-of-sample data”. We prefer to use “training data” and “test data” in this book.

Forecast errors

A forecast “error” is the difference between an observed value and its forecast. Here “error” does not mean a mistake, it means the unpredictable part of an observation.

When computed on the test set, the forecast error is the difference between the observed values and the forecasts obtaining using the training data: $e_{T+h}=y_{T+h}-\hat{}y_{T+h|T}$, where the training data is given by $\{y_1,\ldots,y_T\}$ and the test data is given by $\{y_{T+1},y_{T+2},\ldots\}$. Unlike residuals, forecast errors on the test set involve multi-step forecasts.

We measure forecast accuracy by summarising the forecast errors in different ways.

Scale-dependent errors

The forecast errors are on the same scale as the data. Accuracy measures that are based only on et are therefore scale-dependent and cannot be used to make comparisons between series that involve different units.

The two most commonly used scale-dependent measures are based on the absolute errors or squared errors: Mean absolute error: MAE=mean(|et|),Root mean squared error: RMSE=√mean (e2t).

When comparing forecast methods applied to a single time series, or to several time series with the same units, the MAE is popular as it is easy to both understand and compute. A forecast method that minimizes the MAE will lead to forecasts of the median, while minimizing the RMSE will lead to forecasts of the mean. Consequently, the RMSE is also widely used, despite being more difficult to interpret.

Percentage errors

The percentage error is given by pt=100et/yt. Percentage errors have the advantage of being unit-free, and so are frequently used to compare forecast performances between data sets. The most commonly used measure is: Mean absolute percentage error: MAPE=mean(|pt|). Measures based on percentage errors have the disadvantage of being infinite or undefined if yt=0 for any t in the period of interest, and having extreme values if any yt is close to zero. Another problem with percentage errors that is often overlooked is that they assume the unit of measurement has a meaningful zero.[2] For example, a percentage error makes no sense when measuring the accuracy of temperature forecasts on either the Fahrenheit or Celsius scales, because temperature has an arbitrary zero point.

They also have the disadvantage that they put a heavier penalty on negative errors than on positive errors. This observation led to the use of the so-called "symmetric" MAPE (sMAPE) proposed by Armstrong (1985, 348), which was used in the M3 forecasting competition. It is defined by sMAPE=mean(200|yt−ˆyt|/(yt+ˆyt)). However, if yt is close to zero, ˆyt is also likely to be close to zero. Thus, the measure still involves division by a number close to zero, making the calculation unstable. Also, the value of sMAPE can be negative, so it is not really a measure of "absolute percentage errors" at all.

Hyndman and Koehler (2006) recommend that the sMAPE not be used. It is included here only because it is widely used, although we will not use it in this book.

Scaled errors

Scaled errors were proposed by Hyndman and Koehler (2006) as an alternative to using percentage errors when comparing forecast accuracy across series with different units. They proposed scaling the errors based on the *training* MAE from a simple forecast method.

For a non-seasonal time series, a useful way to define a scaled error uses naïve forecasts: qj=ej1T−1T∑t=2|yt−yt−1|. Because the numerator and denominator both involve values on the scale of the original data, qj is independent of the scale of the data. A scaled error is less than one if it arises from a better forecast than the average naïve forecast computed on the training data. Conversely, it is greater than one if the forecast is worse than the average naïve forecast computed on the training data.

For seasonal time series, a scaled error can be defined using seasonal naïve forecasts: qj=ej1T−mT∑t=m+1|yt−yt−m|.

The *mean absolute scaled error* is simply MASE=mean(|qj|). Similarly, the *mean squared scaled error* (MSSE) can be defined where the errors (on both the training data and test data) are squared instead of using absolute values.

Examples

```
beer2 <- window(ausbeer,start=1992,end=c(2007,4))
beerfit1 <- meanf(beer2,h=10)
beerfit2 <- rwf(beer2,h=10)
beerfit3 <- snaive(beer2,h=10)
autoplot(window(ausbeer, start=1992)) +
  forecast::autolayer(beerfit1, series="Mean", PI=FALSE) +
  forecast::autolayer(beerfit2, series="Naïve", PI=FALSE) +
  forecast::autolayer(beerfit3, series="Seasonal naïve", PI=FALSE) +
  xlab("Year") + ylab("Megalitres") +
  ggtitle("Forecasts for quarterly beer production") +
  guides(colour=guide_legend(title="Forecast"))
```

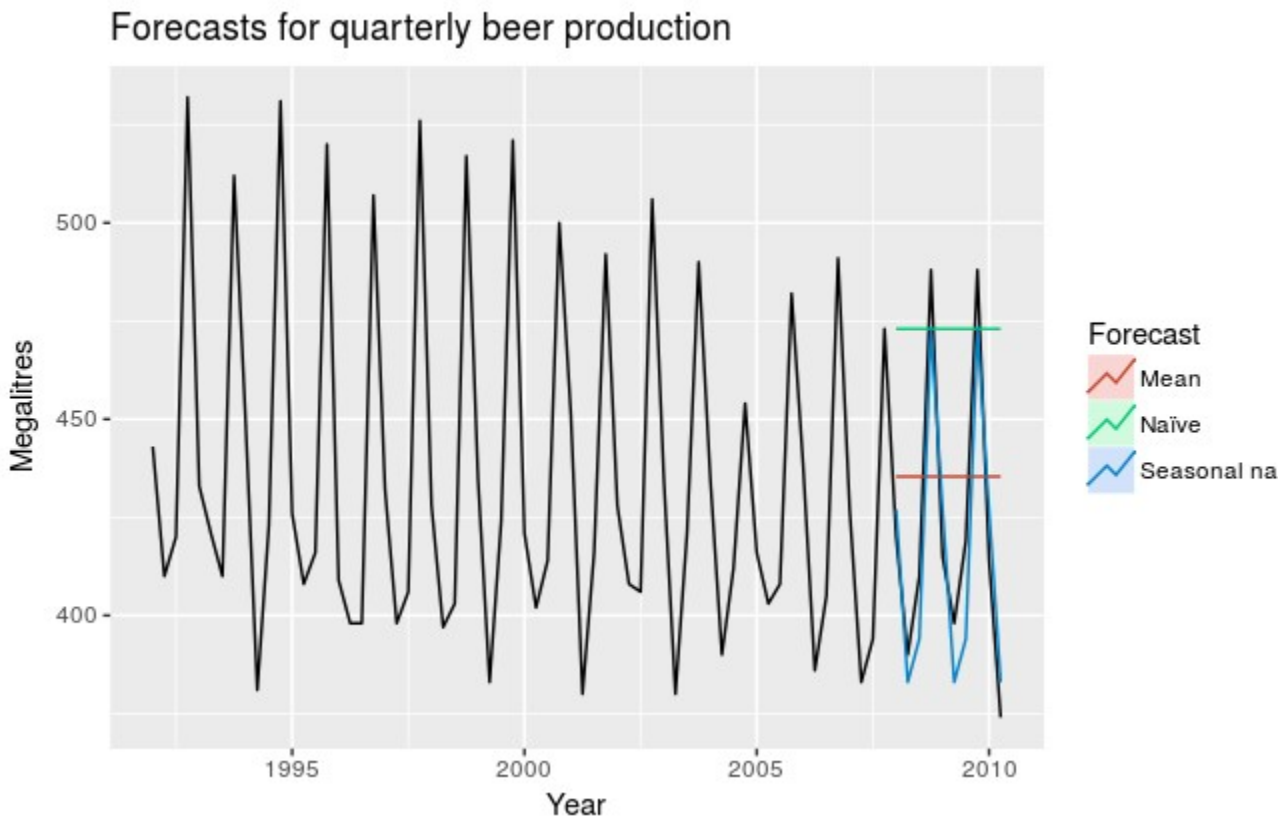


Figure 3.8: Forecasts of Australian quarterly beer production using data up to the end of 2007.

Figure 3.8 shows three forecast methods applied to the quarterly Australian beer production using data only to the end of 2007. The actual values for the period 2008–2010 are also shown. We compute the forecast accuracy measures for this period.

```
beer3 <- window(ausbeer, start=2008)
accuracy(beerfit1, beer3)
accuracy(beerfit2, beer3)
accuracy(beerfit3, beer3)
```

	RMSE	MAE	MAPE	MASE
Mean method	38.5	34.8	8.28	2.44
Naïve method	62.7	57.4	14.18	4.01
Seasonal naïve method	14.3	13.4	3.17	0.94

It is obvious from the graph that the seasonal naïve method is best for these data, although it can still be improved, as we will discover later. Sometimes, different accuracy measures will lead to different results as to which forecast method is best. However, in this case, all of the results point to the seasonal naïve method as the best of these three methods for this data set.

To take a non-seasonal example, consider the Dow Jones Index. The following graph shows the 250 observations ending on 15 July 1994, along with forecasts of the next 42 days obtained from three different methods.

```
dj2 <- window(dj, end=250)
djfc1 <- meanf(dj2, h=42)
djfc2 <- rwf(dj2, h=42)
djfc3 <- rwf(dj2, drift=TRUE, h=42)
autoplot(dj) +
  forecast::autolayer(djfc1, PI=FALSE, series="Mean") +
  forecast::autolayer(djfc2, PI=FALSE, series="Naïve") +
  forecast::autolayer(djfc3, PI=FALSE, series="Drift") +
  xlab("Day") + ylab("") +
  ggtitle("Dow Jones Index (daily ending 15 Jul 94)") +
  guides(colour=guide_legend(title="Forecast"))
```

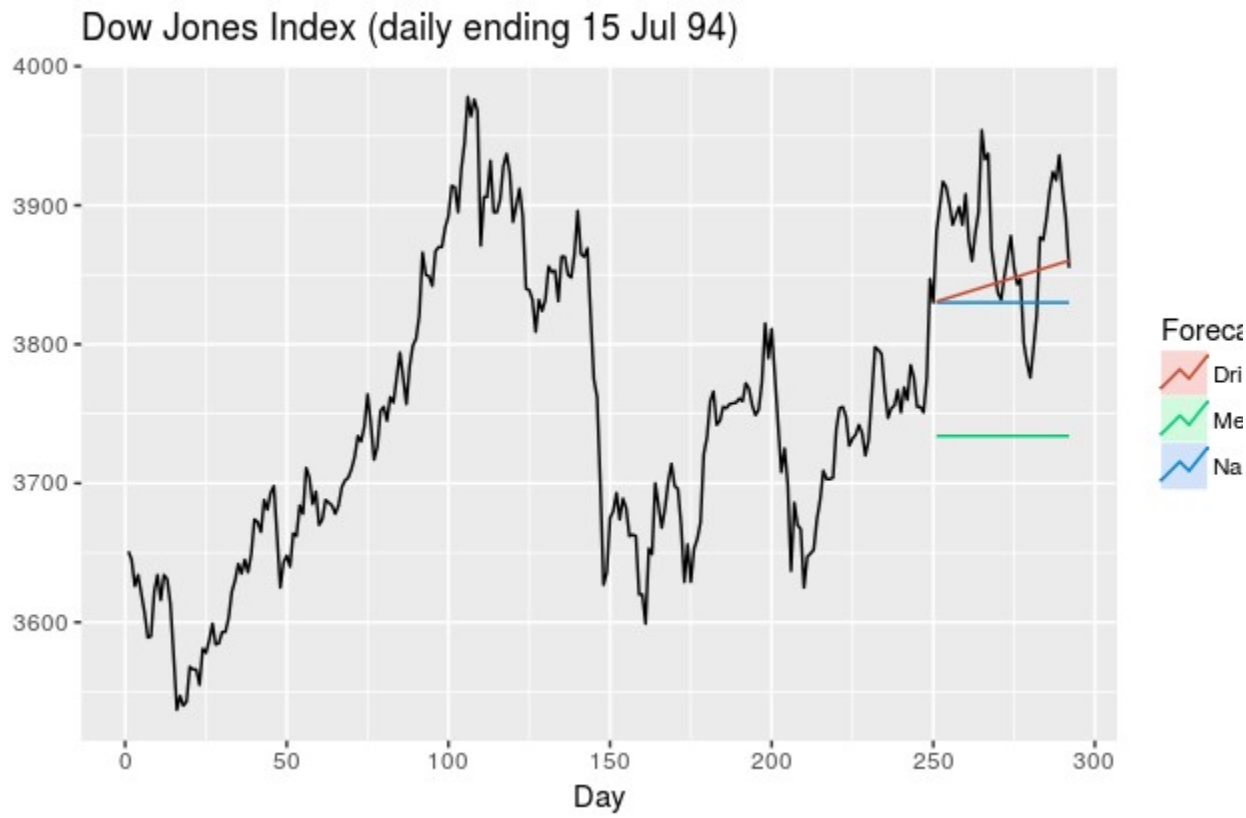


Figure 3.9: Forecasts of the Dow Jones Index from 16 July 1994.

```
dj3 <- window(dj, start=251)
accuracy(djfc1, dj3)
```

```
accuracy(djfc2, dj3)
accuracy(djfc3, dj3)
```

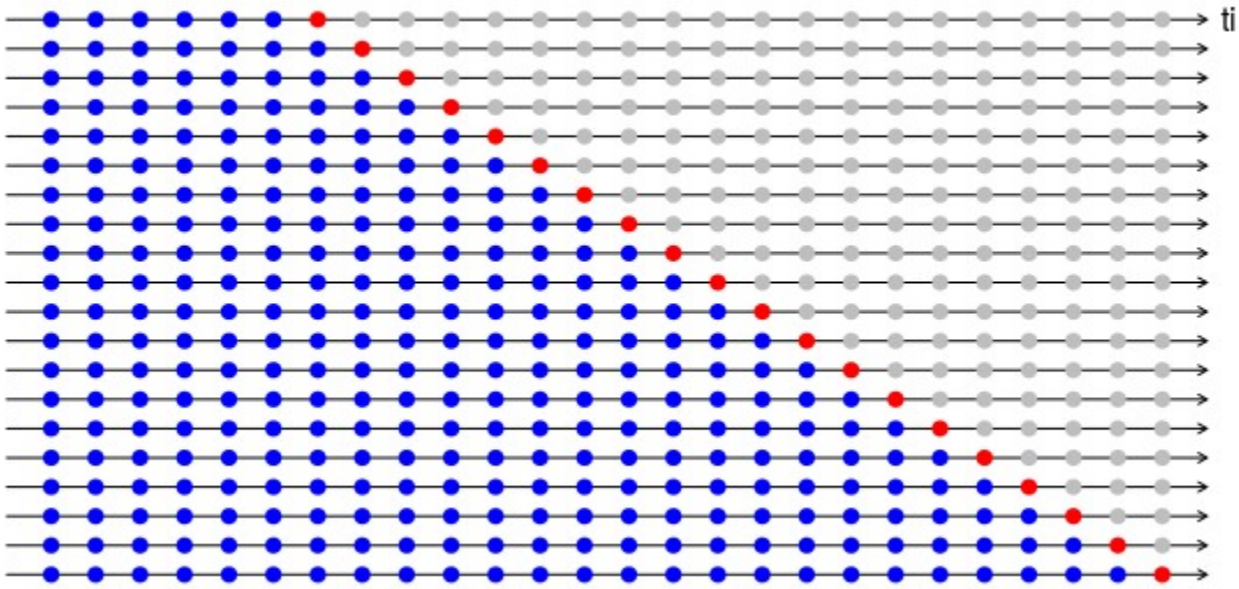
	RMSE	MAE	MAPE	MASE
Mean method	148.2	142.4	3.66	8.70
Naïve method	62.0	54.4	1.40	3.32
Drift method	53.7	45.7	1.18	2.79

Here, the best method is the drift method (regardless of which accuracy measure is used).

Time series cross-validation

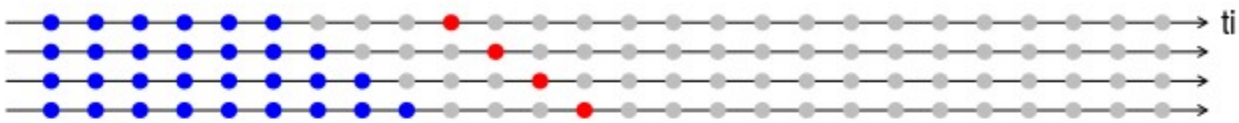
A more sophisticated version of training/test sets is time series cross-validation. In this procedure, there are a series of test sets, each consisting of a single observation. The corresponding training set consists only of observations that occurred *prior* to the observation that forms the test set. Thus, no future observations can be used in constructing the forecast. Since it is not possible to obtain a reliable forecast based on a very small training set, the earliest observations are not considered as test sets.

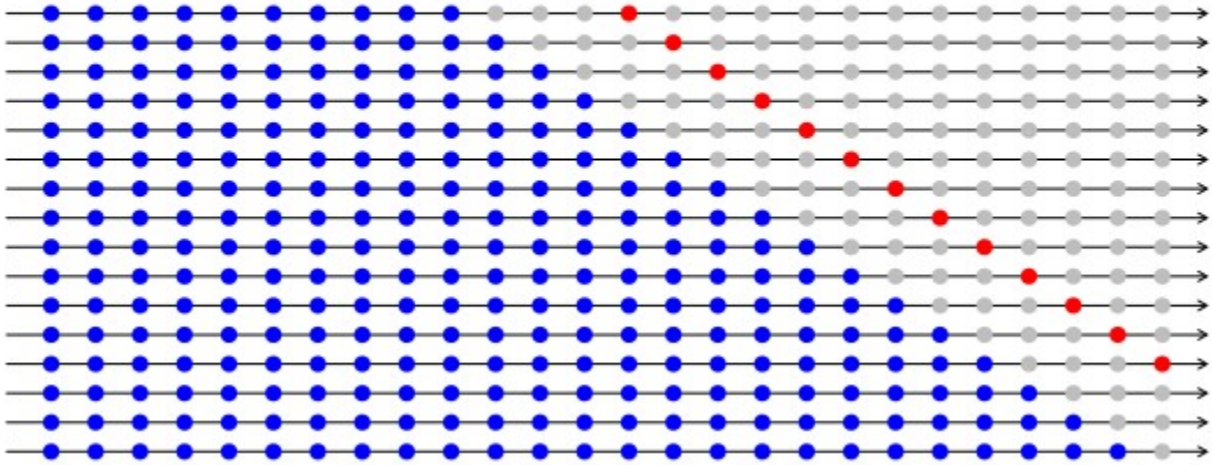
The following diagram illustrates the series of training and test sets, where the blue observations form the training sets, and the red observations form the test sets.



The forecast accuracy is computed by averaging over the test sets. This procedure is sometimes known as “evaluation on a rolling forecasting origin” because the “origin” at which the forecast is based rolls forward in time.

With time series forecasting, one-step forecasts may not be as relevant as multi-step forecasts. In this case, the cross-validation procedure based on a rolling forecasting origin can be modified to allow multi-step errors to be used. Suppose that we are interested in models that produce good 4-step-ahead forecasts. Then the corresponding diagram is shown below.





Time series cross validation is implemented with the `tsCV` function. In the following example, we compare the residual RMSE with the RMSE obtained via time series cross-validation.

```
e <- tsCV(dj, rwf, drift=TRUE, h=1)
sqrt(mean(e^2, na.rm=TRUE))
#> [1] 22.7
sqrt(mean(residuals(rwf(dj, drift=TRUE))^2, na.rm=TRUE))
#> [1] 22.5
```

As expected, the RMSE from the residuals is smaller, as the corresponding "forecasts" are based on a model fitted to the entire data set, rather than being true forecasts.

A good way to choose the best forecasting model is to find the model with the smallest RMSE computed using time series cross-validation.

The ugliness of the above R code makes this a good opportunity to introduce some alternative ways of stringing R functions together. In the above code, we are nesting functions within functions within functions, so you have to read the code from the inside out, making it difficult to understand what is being computed. Instead, we can use the pipe operator `%>%` as follows

```
dj %>% tsCV(forecastfunction=rwf, drift=TRUE, h=1) -> e
e^2 %>% mean(na.rm=TRUE) %>% sqrt
dj %>% rwf(drift=TRUE) %>% residuals -> res
res^2 %>% mean(na.rm=TRUE) %>% sqrt
```

The left hand side of each pipe is passed as the first argument to the function on the right hand side. This is consistent with the way we read from left to right in English. A consequence of using pipes is that all other arguments must be named, which also helps readability. When using pipes, it is natural to use the right arrow assignment `->` rather than the left arrow. For example, the third line above can be read as "Take the `dj` series, pass it to `rwf` with `drift=TRUE`, compute the resulting residuals, and store them as `res`".

In this book, we will use the pipe operator whenever it makes the code easier to read.

3.5 Prediction intervals

As discussed in Section 1.7, a prediction interval gives an interval within which we expect yt to lie with a specified probability. For example, assuming that the forecast errors are uncorrelated and normally distributed, a simple 95% prediction interval for the next observation in a time series is

$\hat{y}_t \pm 1.96\hat{\sigma}$, where $\hat{\sigma}$ is an estimate of the standard deviation of the forecast distribution. In this book we usually calculate 80% intervals and 95% intervals, although any percentage may be used.

When forecasting one step ahead, the standard deviation of the forecast distribution is almost the same as the standard deviation of the residuals. (In fact, the two standard deviations are identical if there are no parameters to be estimated, as is the case with the naïve method. For forecasting methods involving parameters to be estimated, the standard deviation of the forecast distribution is slightly larger than the residual standard deviation, although this difference is often ignored.)

For example, consider a naïve forecast for the Dow-Jones Index. The last value of the observed series is 3830, so the forecast of the next value of the DJI is 3830. The standard deviation of the residuals from the naïve method is 22.00 . Hence, a 95% prediction interval for the next value of the DJI is $3830 \pm 1.96(22.00) = [3787, 3873]$. Similarly, an 80% prediction interval is given by $3830 \pm 1.28(22.00) = [3802, 3858]$.

The value of the multiplier (1.96 or 1.28) determines the percentage of the prediction interval. The following table gives the values to be used for different percentages.

Table 3.1: Multipliers to be used for prediction intervals.

Percentage	Multiplier
50	0.67
55	0.76
60	0.84
65	0.93
70	1.04
75	1.15
80	1.28
85	1.44
90	1.64
95	1.96
96	2.05
97	2.17
98	2.33
99	2.58

The use of this table and the formula $\hat{y}_t \pm k\hat{\sigma}$ (where k is the multiplier) assumes that the residuals are normally distributed and uncorrelated. If either of these conditions does not hold, then this method of producing a prediction interval cannot be used.

The value of prediction intervals is that they express the uncertainty in the forecasts. If we only produce point forecasts, there is no way of telling how accurate the forecasts are. However, if we also produce prediction intervals, then it is clear how much uncertainty is associated with each forecast. For this reason, point forecasts can be of almost no value without the accompanying prediction intervals.

To produce a prediction interval, it is necessary to have an estimate of the standard deviation of the forecast distribution. For one-step forecasts, the residual standard deviation provides a good estimate of the forecast standard deviation. But multi-step forecasts, a more complicated method

of calculation is required. These calculations are usually done with standard forecasting software and need not trouble the forecaster (unless he or she is writing the software!).

A common feature of prediction intervals is that they increase in length as the forecast horizon increases. The further ahead we forecast, the more uncertainty is associated with the forecast, and thus the wider the prediction intervals. However, there are some (non-linear) forecasting methods that do not have this property.

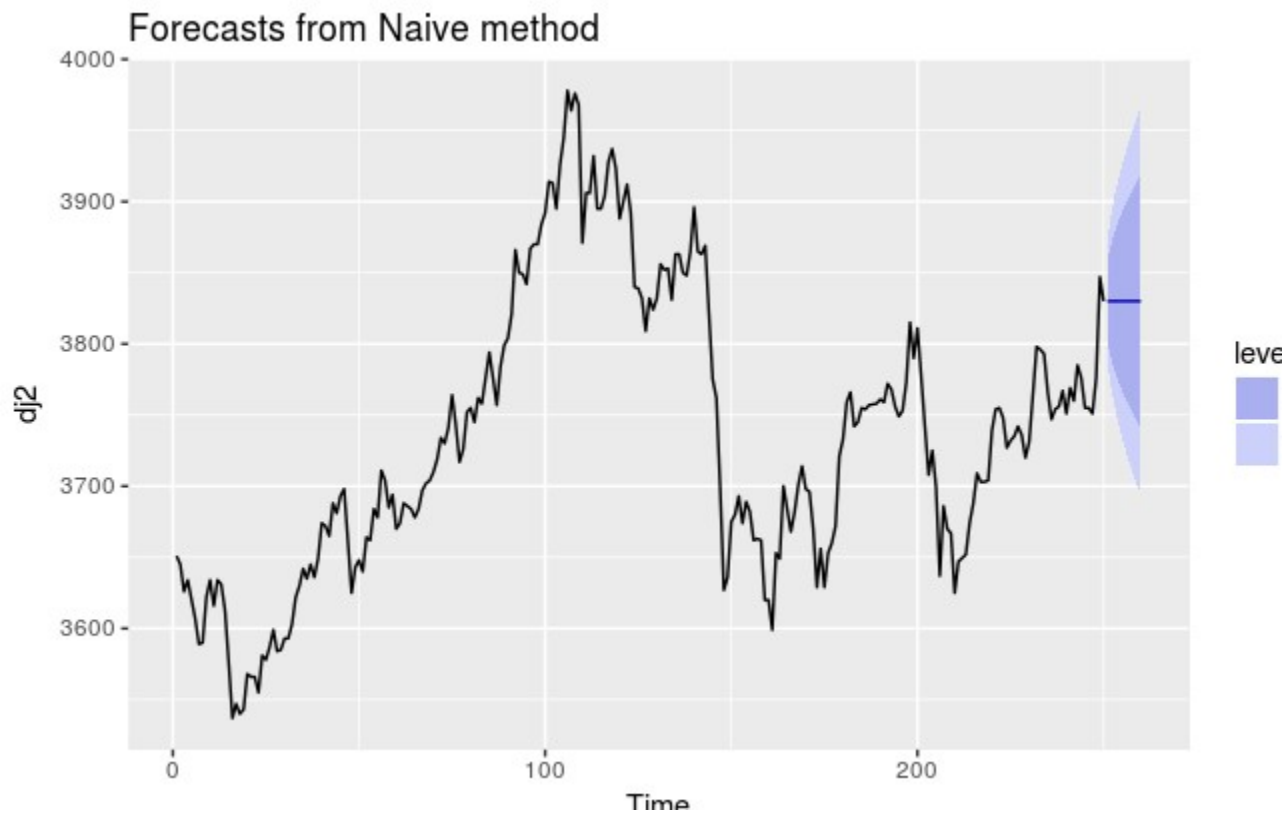
If a transformation has been used, then the prediction interval should be computed on the transformed scale, and the end points back-transformed to give a prediction interval on the original scale. This approach preserves the probability coverage of the prediction interval, although it will no longer be symmetric around the point forecast.

Prediction intervals are computed for you when using any of the benchmark forecasting methods. For example, here is the output when using the naïve method for the Dow-Jones index.

```
naive(dj2)
#>     Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
#> 251           3830  3802  3858  3787  3873
#> 252           3830  3790  3870  3769  3891
#> 253           3830  3781  3879  3755  3905
#> 254           3830  3774  3886  3744  3916
#> 255           3830  3767  3893  3734  3926
#> 256           3830  3761  3899  3724  3936
#> 257           3830  3755  3905  3716  3944
#> 258           3830  3750  3910  3708  3952
#> 259           3830  3745  3915  3701  3959
#> 260           3830  3741  3919  3694  3966
```

When plotted, the prediction intervals are shown as shaded region, with the strength of colour indicating the probability associated with the interval.

```
autoplot(naive(dj2))
```



3.6 The forecast package in R

This book uses the facilities in the `forecast` package in R (which is loaded automatically whenever you load the `fpp` package). This appendix briefly summarises some of the features of the package. Please refer to the help files for individual functions to learn more, and to see some examples of their use.

Functions that output a forecast object:

Many functions, including `meanf`, `naive`, `snaive` and `rwf`, produce output in the form of a `forecast` object (i.e., an object of class `forecast`). This allows other functions (such as `autoplot`) to work consistently across a range of forecasting models.

Objects of class `forecast` contain information about the forecasting method, the data used, the point forecasts obtained, prediction intervals, residuals and fitted values. There are several functions designed to work with these objects including `autoplot`, `summary` and `print`.

The following list shows all the functions that produce `forecast` objects.

- `meanf()`
- `naive()`, `snaive()`
- `rwf()`
- `croston()`
- `stlf()`
- `ses()`
- `holt()`, `hw()`
- `splinef()`
- `thetaf()`
- `forecast()`

`forecast()` function

So far we have used functions which produce a `forecast` object directly. But a more common approach, which we will focus on in the rest of the book, will be to fit a model to the data, and then use the `forecast()` function to produce forecasts from that model.

The `forecast()` function works with many different types of input. It generally takes a time series or time series model as its main argument, and produces forecasts appropriately. It always returns objects of class `forecast`.

If the first argument is of class `ts`, it returns forecasts from the automatic ETS algorithm discussed in Chapter 7.

Here is a simple example, applying `forecast()` to the `ausbeer` data:

```
forecast(ausbeer)
#>         Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
#> 2010 Q3            405   386   423   376   433
#> 2011 Q4            480   458   503   446   515
#> 2011 Q1            417   397   437   386   448
#> 2011 Q2            383   364   402   354   412
#> 2012 Q3            403   381   425   369   437
```

```
#> 2012 Q4            478   451   506   436   521
#> 2012 Q1            415   390   441   377   454
#> 2012 Q2            382   357   406   345   419
```

That works quite well if you have no idea what sort of model to use. But by the end of this book, you should not need to use `forecast()` in this "blind" fashion. Instead, you will fit a model appropriate to the data, and then use `forecast()` to produce forecasts from that model.

3.7 Exercises

1. For the following series, find an appropriate Box-Cox transformation in order to stabilize the variance.

    - `usnetelec`
    - `usgdp`
    - `mcopper`
    - `enplanements`

2. Why is a Box-Cox transformation unhelpful for the `cangas` data?

3. What Box-Cox transformation would you select for your retail data (from Exercise 1 in Section 2.10)?

4. Calculate the residuals from a seasonal naïve forecast applied to the quarterly Australian beer production data from 1992. The following code will help.

    ```
    beer <- window(ausbeer, start=1992)
    fc <- snaive(beer)
    autoplot(fc)
    res <- residuals(fc)
    autoplot(res)
    ```

    Test if the residuals are white noise and normally distributed.

    ```
    checkresiduals(fc)
    ```

    What do you conclude?

5. Repeat the exercise for the `WWWusage` and `bricksq` data. Use whichever of `naive` or `snaive` is more appropriate in each case.

6. Are the following statements true or false? Explain your answer.

    1. Good forecast methods should have normally distributed residuals.
    2. A model with small residuals will give good forecasts.
    3. The best measure of forecast accuracy is MAPE.
    4. If your model doesn't forecast well, you should make it more complicated.
    5. Always choose the model with the best forecast accuracy as measured on the test set.

7. For your retail time series (from Exercise 1 in Section 2.10)?

    1. Split the data into two parts using

        ```
        x1 <- window(mytimeseries, end=c(2010,12))
        x2 <- window(mytimeseries, start=2011)
        ```

2. Check that your data have been split appropriately by producing the following plot.

   ```
   autoplot(cbind(x1,x2))
   ```

3. Calculate forecasts using `snaive` applied to `x1`.

4. Compare the accuracy of your forecasts against the actual values stored in `x2`.

   ```
   f1 <- snaive(x1)
   accuracy(f1,x2)
   ```

5. Check the residuals.

   ```
   checkresiduals(f1)
   ```

   Do the residuals appear to be uncorrelated and normally distributed?

6. How sensitive are the accuracy measures to the training/test split?

8. Use the Dow Jones index (data set `dowjones`) to do the following:

   1. Produce a time plot of the series.
   2. Produce forecasts using the drift method and plot them.
   3. Show that the forecasts are identical to extending the line drawn between the first and last observations.
   4. Try using some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

9. Consider the daily closing IBM stock prices (data set `ibmclose`).

   1. Produce some plots of the data in order to become familiar with it.
   2. Split the data into a training set of 300 observations and a test set of 69 observations.
   3. Try using various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?
   4. Check the residuals of your preferred method. Do they resemble white noise?

10. Consider the sales of new one-family houses in the USA, Jan 1973 – Nov 1995 (data set `hsales`).

    1. Produce some plots of the data in order to become familiar with it.
    2. Split the `hsales` data set into a training set and a test set, where the test set is the last two years of data.
    3. Try using various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?
    4. Check the residuals of your preferred method. Do they resemble white noise?

3.8 Further reading

- Ord and Fildes (2012) provides further discussion of simple benchmark forecasting methods.
- A review of forecast evaluation methods is given in Hyndman and Koehler (2006), looking at the strengths and weakenesses of different approaches. This is the paper that introduced the MASE as a general-purpose forecast accuracy measure.

References

Ord, J Keith, and Robert Fildes. 2012. *Principles of Business Forecasting*. South-Western College Pub.

Hyndman, Rob J, and Anne B Koehler. 2006. “Another Look at Measures of Forecast Accuracy.” *International Journal of Forecasting* 22: 679–88.

Chapter 4 Judgmental forecasts

Forecasting using judgement is very common in practice. In many cases, judgmental forecasting is the only option, such as when there is a complete lack of historical data, or when a new product is being launched, or when a new competitor enters the market, or during completely new and unique market conditions. For example, in December 2012, the Australian government was the first in the world to pass legislation that banned the use of company logos on cigarette packets, and required all cigarette packets to be a dark green colour. Judgement must be applied in order to forecast the effect of such a policy, as there are no historical precedents.

There are also situations where the data are incomplete, or only become available after some delay. For example, central banks include judgement when forecasting the current level of economic activity, a procedure known as nowcasting, as GDP is only available on a quarterly basis.

Research in this area[3] has shown that the accuracy of judgmental forecasting improves when the forecaster has (i) important domain knowledge, and (ii) more timely, up-to-date information. A judgmental approach can be quick to adjust to such changes, information or events.

Over the years, the acceptance of judgmental forecasting as a science has increased, as has the recognition of its need. More importantly, the quality of judgmental forecasts has also improved, as a direct result of recognising that improvements in judgmental forecasting can be achieved by implementing well-structured and systematic approaches. It is important to recognise that judgmental forecasting is subjective and comes with limitations. However, implementing systematic and well-structured approaches can confine these limitations and markedly improve forecast accuracy.

There are three general settings in which judgmental forecasting is used: (i) there are no available data, so that statistical methods are not applicable and judgmental forecasting is the only feasible approach; (ii) data are available, statistical forecasts are generated, and these are then adjusted using judgement; and (iii) data are available and statistical and judgmental forecasts are generated independently and then combined. We should clarify that when data are available, applying statistical methods (such as those discussed in other chapters of this book), is preferable and should, at the very least, be used as a starting point. Statistical forecasts are generally superior to generating forecasts using only judgement. For the majority of the chapter, we focus on the first setting where no data are available, and in the very last section we discuss judgmentally adjusting statistical forecasts. We discuss combining forecasts in Chapter ??.

4.1 Beware of limitations

Judgmental forecasts are subjective, and therefore do not come free of bias or limitations.

Judgmental forecasts can be inconsistent. Unlike statistical forecasts, which can be generated by the same mathematical formulae every time, judgmental forecasts depend heavily on human cognition, and are vulnerable to its limitations. For example, a limited memory may render recent events more important than they actually are and may ignore momentous events from the more distant past; or a limited attention span may result in important information being missed; or a misunderstanding of causal relationships may lead to erroneous inferences. Furthermore, human judgement can vary due to the effect of psychological factors. One can imagine a manager who is in a positive frame of mind one day, generating forecasts that may tend to be somewhat optimistic, and in a negative frame of mind another day, generating somewhat less optimistic forecasts.

Judgement can be clouded by personal or political agendas, where targets and forecasts (as defined in Chapter 1 are not segregated. For example, if a sales manager knows that the forecasts she generates will be used to set sales expectations (targets), she may tend to set these low in order to show a good performance (i.e., exceed the expected targets). Even in cases where targets and forecasts are well segregated, judgement may be plagued by optimism or wishful thinking. For example, it would be highly unlikely that a team working towards launching a new product would forecast its failure. As we will discuss later, this optimism can be accentuated in a group meeting setting. "Beware of the enthusiasm of your marketing and sales colleagues"[4].

Another undesirable property which is commonly seen in judgmental forecasting is the effect of anchoring. In this case, the subsequent forecasts tend to converge or be very close to an initial familiar reference point. For example, it is common to take the last observed value as a reference point. The forecaster is influenced unduly by prior information, and therefore gives this more weight in the forecasting process. Anchoring may lead to conservatism and undervaluing new and more current information, and thereby create a systematic bias.

4.2 Key principles

Using a systematic and well structured approach in judgmental forecasting helps to reduce the adverse effects of the limitations of judgmental forecasting, some of which we listed in the previous section. Whether this approach involves one individual or many, the following principles should be followed.

Set the forecasting task clearly and concisely

Care is needed when setting the forecasting challenges and expressing the forecasting tasks. It is important that everyone be clear about what the task is. All definitions should be clear and comprehensive, avoiding ambiguous and vague expressions. Also, it is important to avoid incorporating emotive terms and irrelevant information that may distract the forecaster. In the Delphi method that follows (see Section 4.3), it may sometimes be useful to conduct a preliminary round of information gathering before setting the forecasting task.

Implement a systematic approach

Forecast accuracy and consistency can be improved by using a systematic approach to judgmental forecasting involving checklists of categories of information which are relevant to the forecasting task. For example, it is helpful to identify what information is important and how this information is to be weighted. When forecasting the demand for a new product, what factors should we account for and how should we account for them? Should it be the price, the quality and/or quantity of the competition, the economic environment at the time, the target population of the product? It is worthwhile to devote significant effort and resources to put together decision rules that will lead to the best possible systematic approach.

Document and justify

Formalising and documenting the decision rules and assumptions implemented in the systematic approach can promote consistency, as the same rules can be implemented repeatedly. Also, requesting a forecaster to document and justify their forecasts leads to accountability, which can lead to a reduced bias. Furthermore, formal documentation aids significantly in the systematic evaluation process that is suggested next.

Systematically evaluate forecasts

Systematically monitoring the forecasting process can identify unforeseen irregularities. In particular, keep records of forecasts and use them to obtain feedback as the forecasted period becomes observed. Although you may do your best as a forecaster, the environment you operate in is dynamic. Changes occur, and you need to monitor these in order to evaluate the decision rules and assumptions. Feedback and evaluation help forecasters learn and improve their forecast accuracy.

Segregate forecasters and users

Forecast accuracy may be impeded if the forecasting task is carried out by users of the forecasts, such as those responsible for implementing plans of action about which the forecast is concerned. We should clarify again here (as in Section 1.2), that forecasting is about predicting the future as accurately as possible, given all of the information available, including historical data and knowledge of any future events that may impact the forecasts. Forecasters and users should be clearly segregated. A classic case is that of a new product being launched. The forecast should be a reasonable estimate of the sales volume of a new product, which may differ considerably from what management expects or hopes the sales will be in order to meet company financial objectives. In this case, a forecaster may be delivering a reality check to the user.

It is important that forecasters communicate forecasts to potential users thoroughly. As we will see in Section 4.7, users may feel distant and disconnected from forecasts, and may not have full confidence in them. Explaining and clarifying the process and justifying the basic assumptions that led to the forecasts will provide some assurance to users.

The way in which forecasts may then be used and implemented will clearly depend on managerial decision making. For example, management may decide to adjust a forecast upwards (be over-optimistic), as the forecast may be used to guide purchasing and stock keeping levels. Such a decision may be taken after a cost-benefit analysis reveals that the cost of holding excess stock is much lower than that of lost sales. This type of adjustment should be part of setting goals or planning supply, rather than part of the forecasting process. In contrast, if forecasts are used as targets, they may be set low so that they can be exceeded more easily. Again, setting targets is different from producing forecasts, and the two should not be confused.

The example that follows comes from our experience in industry. It exemplifies two contrasting styles of judgmental forecasting — one that adheres to the principles we have just presented and one that does not.

Example: Pharmaceutical Benefits Scheme (PBS)

The Australian government subsidises the cost of a wide range of prescription medicines as part of the PBS. Each subsidised medicine falls into one of four categories: concession copayments, concession safety net, general copayments, and general safety net. Each person with a concession

card makes a concession copayment per PBS medicine ($5.80)[5], until they reach a set threshold amount labelled the concession safety net ($348). For the rest of the financial year, all PBS-listed medicines are free. Each general patient makes a general copayment per PBS medicine ($35.40) until the general safety net amount is reached ($1,363.30). For the rest of the financial year, they contribute a small amount per PBS-listed medicine ($5.80). The PBS forecasting process uses 84 groups of PBS-listed medicines, and produces forecasts of the medicine volume and the total expenditure for each group and for each of the four PBS categories, a total of 672 series. This forecasting process aids in setting the government budget allocated to the PBS, which is over $7 billion per year, or approximately 1% of GDP.

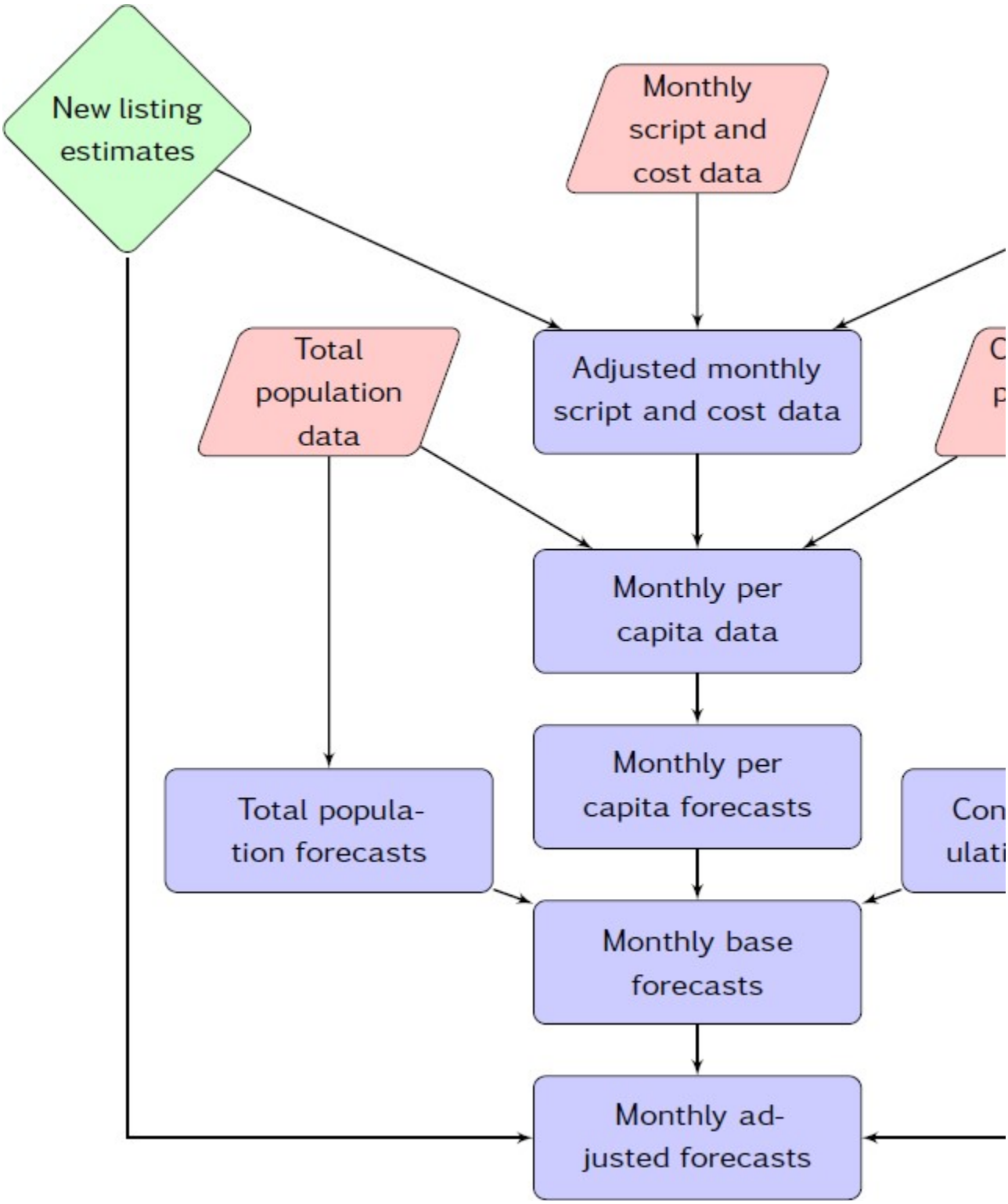


Figure 4.1: Process for producing PBS forecasts.

Figure 4.1 summarises the forecasting process. Judgmental forecasts are generated for new listings of medicines and for estimating the impact of new policies. These are shown by the green items. The pink items indicate the data used which were obtained from various government departments and associated authorities. The blue items show things that are calculated from the data provided. There were judgmental adjustments to the data to take account of new listings and new policies, and there were also judgmental adjustments to the forecasts. Because of the changing size of both the concession population and the total population, forecasts are produced on a per-capita basis, and then multiplied by the forecast population to obtain forecasts of total volume and expenditure per month.

One of us (Hyndman) was asked to evaluate the forecasting process a few years ago. We found that using judgement for new listings and new policy impacts gave better forecasts than using a statistical model alone. However, we also found that the forecasting accuracy and consistency could be improved through a more structured and systematic process, especially for policy impacts.

Forecasting new listings: Companies who are applying for their medicine to be added to the PBS are asked to submit detailed forecasts for various aspects of the medicine, such as projected patient numbers, market share of the new medicine, substitution effects, etc. The Pharmaceutical Benefits Advisory Committee provides guidelines describing a highly structured and systematic approach for generating these forecasts, and requires careful documentation for each step of the process. This structured process helps to reduce the likelihood and effects of deliberate self-serving biases. Two detailed evaluation rounds of the company forecasts are implemented by a sub-committee, one before the medicine is added to the PBS and one after it is added. Finally, comparisons of observations versus forecasts for some selected new listings are performed, 12 months and 24 months after the listings, and the results are sent back to the companies for comment.

Policy impact forecasts: In contrast to the highly structured process used for new listings, there were no systematic procedures for policy impact forecasts. On many occasions, forecasts of policy impacts were calculated by a small team, and were often heavily reliant on the work of one person. The forecasts were not usually subject to a formal review process. There were no guidelines for how to construct judgmental forecasts for policy impacts, and there was often a lack of adequate documentation about how these forecasts were obtained, the assumptions underlying them, etc.

Consequently, we recommended several changes:

- that guidelines for forecasting new policy impacts be developed, to encourage a more systematic and structured forecasting approach;
- that the forecast methodology be documented in each case, including all assumptions made in forming the forecasts;
- that new policy forecasts be made by at least two people from different areas of the organisation;
- that a review of forecasts be conducted one year after the implementation of each new policy by a review committee, especially for new policies that have a significant annual projected cost or saving. The review committee should include those involved in generating the forecasts, but also others.

These recommendations reflect the principles outlined in Section 4.2.

4.3 The Delphi method

The Delphi method was invented by Olaf Helmer and Norman Dalkey of the Rand Corporation in the 1950s for the purpose of addressing a specific military problem. The method relies on the key assumption that forecasts from a group are generally more accurate than those from individuals. The aim of the Delphi method is to construct consensus forecasts from a group of experts in a structured iterative manner. A facilitator is appointed in order to implement and manage the process. The Delphi method generally involves the following stages:

1. A panel of experts is assembled.
2. Forecasting tasks/challenges are set and distributed to the experts.
3. Experts return initial forecasts and justifications. These are compiled and summarised in order to provide feedback.
4. Feedback is provided to the experts, who now review their forecasts in light of the feedback. This step may be iterated until a satisfactory level of consensus is reached.
5. Final forecasts are constructed by aggregating the experts' forecasts.

Each stage of the Delphi method comes with its own challenges. In what follows, we provide some suggestions and discussions about each one of these.[6]

Experts and anonymity

The first challenge of the facilitator is to identify a group of experts who can contribute to the forecasting task. The usual suggestion is somewhere between 5 and 20 experts with diverse expertise. Experts submit forecasts and also provide detailed qualitative justifications for these.

A key feature of the Delphi method is that the participating experts remain anonymous at all times. This means that the experts cannot be influenced by political and social pressures in their forecasts. Furthermore, all experts are given an equal say and all are held accountable for their forecasts. This avoids the situation where a group meeting is held and some members do not contribute, while others dominate. It also prevents members exerting undue influence based on seniority or personality. There have been suggestions that even something as simple as the seating arrangements in a group setting can influence the group dynamics. Furthermore, there is ample evidence that a group meeting setting promotes enthusiasm and influences individual judgement, leading to optimism and overconfidence.[7]

A byproduct of anonymity is that the experts do not need to meet as a group in a physical location. An important advantage of this is that it increases the likelihood of gathering experts with diverse skills and expertise from varying locations. Furthermore, it makes the process cost-effective by eliminating the expense and inconvenience of travel, and it makes it flexible, as the experts only have to meet a common deadline for submitting forecasts, rather than having to set a common meeting time.

Setting the forecasting task in a Delphi

In a Delphi setting, it may be useful to conduct a preliminary round of information gathering from the experts before setting the forecasting tasks. Alternatively, as experts submit their initial forecasts and justifications, valuable information which is not shared between all experts can be identified by the facilitator when compiling the feedback.

Feedback

Feedback to the experts should include summary statistics of the forecasts and outlines of qualitative justifications. Numerical data summaries and graphical representations can be used to summarise the experts' forecasts.

As the feedback is controlled by the facilitator, there may be scope to direct attention and information from the experts to areas where it is most required. For example, the facilitator may direct the experts' attention to responses that fall outside the interquartile range, and the qualitative justification for such forecasts.

Iteration

The process of the experts submitting forecasts, receiving feedback, and reviewing their forecasts in light of the feedback, is repeated until a satisfactory level of consensus between the experts is reached. Satisfactory consensus does not mean complete convergence in the forecast value; it simply means that the variability of the responses has decreased to a satisfactory level. Usually two or three rounds are sufficient. Experts are more likely to drop out as the number of iterations increases, so too many rounds should be avoided.

Final forecasts

The final forecasts are usually constructed by giving equal weight to all of the experts' forecasts. However, the facilitator should keep in mind the possibility of extreme values which can distort the final forecast.

Limitations and variations

Applying the Delphi method can be time consuming. In a group meeting, final forecasts can possibly be reached in hours or even minutes — something which is almost impossible to do in a Delphi setting. If it is taking a long time to reach a consensus in a Delphi setting, the panel may lose interest and cohesiveness.

In a group setting, personal interactions can lead to quicker and better clarifications of qualitative justifications. A variation of the Delphi method which is often applied is the "estimate-talk-estimate" method, where the experts can interact between iterations, although the forecast submissions can still remain anonymous. A disadvantage of this variation is the possibility of the loudest person exerting undue influence.

The facilitator

The role of the facilitator is of the utmost importance. The facilitator is largely responsible for the design and administration of the Delphi process. The facilitator is also responsible for providing feedback to the experts and generating the final forecasts. In this role, the facilitator needs to be experienced enough to recognise areas that may need more attention, and to direct the experts' attention to these. Also, as there is no face-to-face interaction between the experts, the facilitator is responsible for disseminating important information. The efficiency and effectiveness of the facilitator can dramatically increase the probability of a successful Delphi method in a judgmental forecasting setting.

A useful judgmental approach which is often implemented in practice is forecasting by analogy. A common example is the pricing of a house through an appraisal process. An appraiser estimates

the market value of a house by comparing it to similar properties that have sold in the area. The degree of similarity depends on the attributes considered. With house appraisals, attributes such as land size, dwelling size, numbers of bedrooms and bathrooms, and garage space are usually considered.

Even thinking and discussing analogous products or situations can generate useful (and sometimes crucial) information. We illustrate this point with the following example.[8]

Example: Designing a high school curriculum

A small group of academics and teachers were assigned the task of developing a curriculum for teaching judgement and decision making under uncertainty for high schools in Israel. Each group member was asked to forecast how long it would take for the curriculum to be completed. Responses ranged between 18 and 30 months. One of the group members who was an expert in curriculum design was asked to consider analogous curricula developments around the world. He concluded that 40% of analogous groups he considered never completed the task. The rest took between 7 to 10 years. The Israel project was completed in 8 years.

Obviously, forecasting by analogy comes with challenges. We should aspire to base forecasts on multiple analogies rather than a single analogy, which may create biases. However, these may be challenging to identify. Similarly, we should aspire to consider multiple attributes. Identifying or even comparing these may not always be straightforward. As always, we suggest performing these comparisons and the forecasting process using a systematic approach. Developing a detailed scoring mechanism to rank attributes and record the process of ranking will always be useful.

A structured analogy

Alternatively, a structured approach comprising a panel of experts can be implemented, as was proposed by Green and Armstrong (2007). The concept is similar to that of a Delphi; however, the forecasting task is completed by considering analogies. First, a facilitator is appointed. Then the structured approach involves the following steps.

1. A panel of experts who are likely to have experience with analogous situations is assembled.
2. Tasks/challenges are set and distributed to the experts.
3. Experts identify and describe as many analogies as they can, and generate forecasts based on each analogy.
4. Experts list similarities and differences of each analogy to the target situation, then rate the similarity of each analogy to the target situation on a scale.
5. Forecasts are derived by the facilitator using a set rule. This can be a weighted average, where the weights can be guided by the ranking scores of each analogy by the experts.

As with the Delphi approach, anonymity of the experts may be an advantage in not suppressing creativity, but could hinder collaboration. Green and Armstrong found no gain in collaboration between the experts in their results. A key finding was that experts with multiple analogies (more than two), and who had direct experience with the analogies, generated the most accurate forecasts.

4.5 Scenario Forecasting

A fundamentally different approach to judgmental forecasting is scenario-based forecasting. The aim of this approach is to generate forecasts based on plausible scenarios. In contrast to the two

previous approaches (Delphi and forecasting by analogy) where the resulting forecast is intended to be a likely outcome, each scenario-based forecast may have a low probability of occurrence. The scenarios are generated by considering all possible factors or drivers, their relative impacts, the interactions between them, and the targets to be forecasted.

Building forecasts based on scenarios allows a wide range of possible forecasts to be generated and some extremes to be identified. For example it is usual for “best”, “middle” and “worst” case scenarios to be presented, although many other scenarios will be generated. Thinking about and documenting these contrasting extremes can lead to early contingency planning.

With scenario forecasting, decision makers often participate in the generation of scenarios. While this may lead to some biases, it can ease the communication of the scenario-based forecasts, and lead to a better understanding of the results.

4.6 New product forecasting

The definition of a new product can vary. It may be an entirely new product which has been launched, a variation of an existing product (“new and improved”), a change in the pricing scheme of an existing product, or even an existing product entering a new market.

Judgmental forecasting is usually the only available method for new product forecasting, as historical data are unavailable. The approaches we have already outlined (Delphi, forecasting by analogy and scenario forecasting) are all applicable when forecasting the demand for a new product.

Other methods which are more specific to the situation are also available. We briefly describe three such methods which are commonly applied in practice. These methods are less structured than those already discussed, and are likely to lead to more biased forecasts as a result.

Sales force composite

In this approach, forecasts for each outlet/branch/store of a company are generated by salespeople, and are then aggregated. This usually involves sales managers forecasting the demand for the outlet they manage. Salespeople are usually closest to the interaction between customers and products, and often develop an intuition about customer purchasing intentions. They bring this valuable experience and expertise to the forecast.

However, having salespeople generate forecasts violates the key principle of segregating forecasters and users, which can create biases in many directions. It is very common for the performance of a salesperson to be evaluated against the sales forecasts or expectations set beforehand. In this case, the salesperson acting as a forecaster may introduce some self-serving bias by generating low forecasts. On the other hand, one can imagine a very enthusiastic salesperson, full of optimism, generating high forecasts.

Moreover a successful salesperson is not necessarily a successful nor well-informed forecaster. A large proportion of salespeople will have no or very limited formal training in forecasting. Finally, salespeople will feel customer displeasure at first hand if, for example, the product runs out or is not introduced in their store. Such interactions will cloud their judgement.

Executive opinion

In contrast to the sales force composite, this approach involves staff at the top of the managerial structure generating aggregate forecasts. Such forecasts are usually generated in a group meeting, where executives contribute information from their own area of the company. Having executives from different functional areas of the company promotes great skill and knowledge diversity in the group.

This process carries all of the advantages and disadvantages of a group meeting setting which we discussed earlier. In this setting, it is important to justify and document the forecasting process. That is, executives need to be held accountable in order to reduce the biases generated by the group meeting setting. There may also be scope to apply variations to a Delphi approach in this setting; for example, the estimate-talk-estimate process described earlier.

Customer intentions

Customer intentions can be used to forecast the demand for a new product or for a variation on an existing product. Questionnaires are filled in by customers on their intentions to buy the product. A structured questionnaire is used, asking customers to rate the likelihood of them purchasing the product on a scale; for example, highly likely, likely, possible, unlikely, highly unlikely.

Survey design challenges, such as collecting a representative sample, applying a time- and cost-effective method, and dealing with non-responses, need to be addressed.[9]

Furthermore, in this survey setting we must keep in mind the relationship between purchase intention and purchase behaviour. Customers do not always do what they say they will. Many studies have found a positive correlation between purchase intentions and purchase behaviour; however, the strength of these correlations varies substantially. The factors driving this variation include the timings of data collection and product launch, the definition of "new" for the product, and the type of industry. Behavioural theory tells us that intentions predict behaviour if the intentions are measured just before the behaviour.[10] The time between intention and behaviour will vary depending on whether it is a completely new product or a variation on an existing product. Also, the correlation between intention and behaviour is found to be stronger for variations on existing and familiar products than for completely new products.

Whichever method of new product forecasting is used, it is important to thoroughly document the forecasts made, and the reasoning behind them, in order to be able to evaluate them when data become available.

4.7 Judgmental adjustments

In this final section, we consider the situation where historical data are available and are used to generate statistical forecasts. It is common for practitioners to then apply judgmental adjustments to these forecasts. These adjustments can potentially provide all of the advantages of judgmental forecasting which have been discussed earlier in this chapter. For example, they provide an avenue for incorporating factors that may not be accounted for in the statistical model, such as promotions, large sporting events, holidays, or recent events that are not yet reflected in the data. However, these advantages come to fruition only when the right conditions are present. Judgmental adjustments, like judgmental forecasts, come with biases and limitations, and we must implement methodical strategies in order to minimise them.

Use adjustments sparingly

Practitioners adjust much more often than they should, and many times for the wrong reasons. By adjusting statistical forecasts, users of forecasts create a feeling of ownership and credibility. Users often do not understand or appreciate the mechanisms that generate the statistical forecasts (as they will usually have no training in this area). By implementing judgmental adjustments, users feel that they have contributed to and completed the forecasts, and they can now relate their own intuition and interpretations to these. The forecasts have become their own.

Judgmental adjustments should not aim to correct for a systematic pattern in the data that is thought to have been missed by the statistical model. This has been proven to be ineffective, as forecasters tend to read non-existent patterns in noisy series. Statistical models are much better at taking account of data patterns, and judgmental adjustments only hinder accuracy.

Judgmental adjustments are most effective when there is significant additional information at hand or strong evidence of the need for an adjustment. We should only adjust when we have important extra information which is not incorporated in the statistical model. Hence, adjustments seem to be most accurate when they are large in size. Small adjustments (especially in the positive direction promoting the illusion of optimism) have been found to hinder accuracy, and should be avoided.

Apply a structured approach

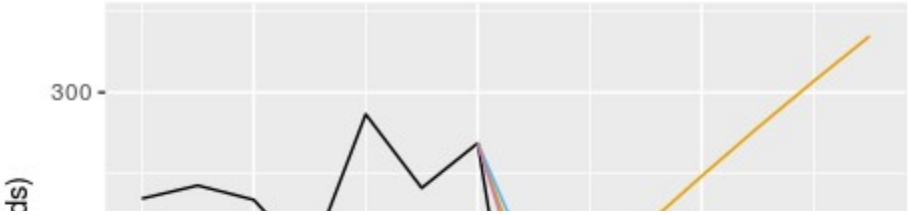
Using a structured and systematic approach will improve the accuracy of judgmental adjustments. Following the key principles outlined in Section @ref(sec-3-Key principles) is vital. In particular, having to document and justify adjustments will make it more challenging to override the statistical forecasts, and will guard against unnecessary adjustments.

It is common for adjustments to be implemented by a panel (see the example that follows). Using a Delphi setting carries great advantages. However, if adjustments are implemented in a group meeting, it is wise to consider the forecasts of key markets or products first, as panel members will get tired during this process. Fewer adjustments tend to be made as the meeting goes on through the day.

Example: Tourism Forecasting Committee (TFC)

Tourism Australia publishes forecasts for all aspects of Australian tourism twice a year. The published forecasts are generated by the TFC, an independent body which comprises experts from various government and private industry sectors; for example, the Australian Commonwealth Treasury, airline companies, consulting firms, banking sector companies, and tourism bodies.

The forecasting methodology applied is an iterative process. First, model-based statistical forecasts are generated by the forecasting unit within Tourism Australia, then judgmental adjustments are made to these in two rounds. In the first round, the TFC Technical Committee[11] (comprising senior researchers, economists and independent advisors) adjusts the model-based forecasts. In the second and final round, the TFC (comprising industry and government experts) makes final adjustments. In both rounds, adjustments are made by consensus.



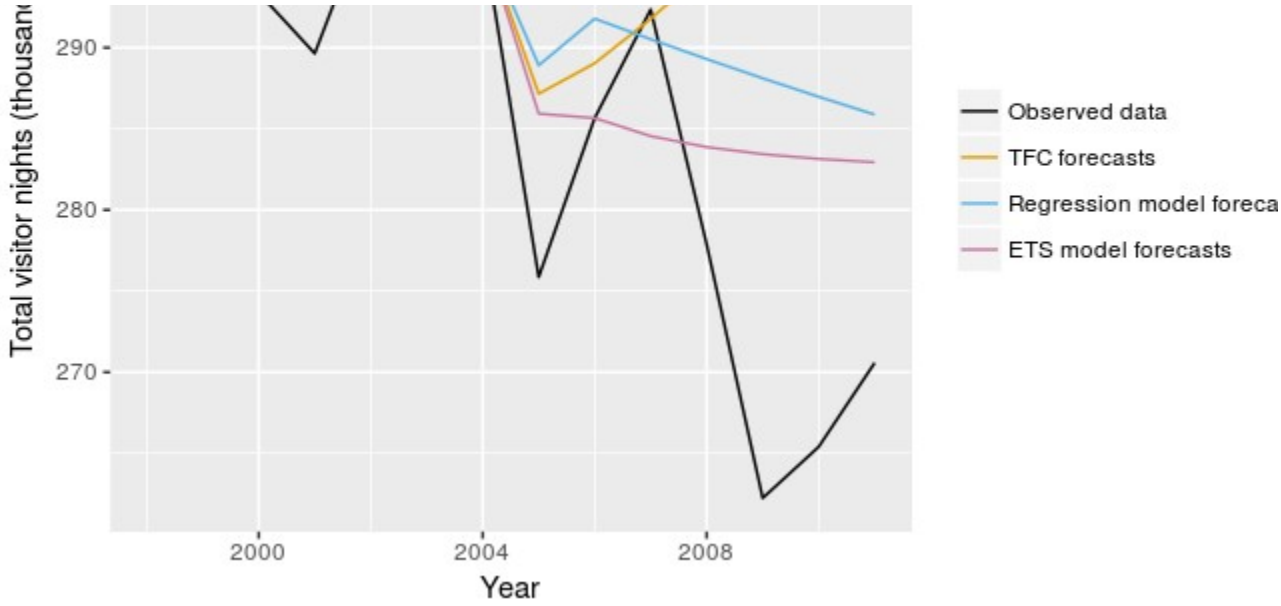


Figure 4.2: Long run annual forecasts for domestic visitor nights for Australia. We study regression models in Chapter 5, and ETS (ExponenTial Smoothing) models in Chapter 7.

In 2008, we (Athanasopoulos and Hyndman 2008) analysed forecasts for Australian domestic tourism. We concluded that the published TFC forecasts were optimistic, especially for the long-run, and we proposed alternative model-based forecasts. We now have access to observed data up to and including 2011. In Figure , we plot the published forecasts against the actual data. We can see that the published TFC forecasts have continued to be optimistic.

What can we learn from this example? Although the TFC clearly states in its methodology that it produces ‘forecasts’ rather than ‘targets’, could this be a case where these have been confused? Are the forecasters and users sufficiently well-segregated in this process? Could the iterative process itself be improved? Could the adjustment process in the meetings be improved? Could it be that the group meetings have promoted optimism? Could it be that domestic tourism should have been considered earlier in the day?

Ord, J Keith, and Robert Fildes. 2012. *Principles of Business Forecasting*. South-Western College Pub.

Goodwin, Paul, and George Wright. 2009. *Decision Analysis for Management Judgment*. 4th ed. Chichester: Wiley.

Kahn, Kenneth B. 2006. *New Product Forecasting: An Applied Approach*. M.E. Sharp.

Fildes, Robert, and Paul Goodwin. 2007b. “Good and Bad Judgment in Forecasting: Lessons from Four Companies.” *Foresight: The International Journal of Applied Forecasting*, no. 8: 5–10.

Fildes, Robert, and Paul Goodwin. 2007a. “Against Your Better Judgment? How Organizations Can Improve Their Use of Management Judgment in Forecasting.” *Interfaces* 37 (6): 570–76.

Harvey, Nigel. 2001. “Improving Judgment in Forecasting.” In *Principles of Forecasting: A Handbook for Researchers and Practitioners*, edited by J Scott Armstrong, 59–80. Boston, MA: Kluwer Academic Publishers.

Rowe, Gene, and George Wright. 1999. “The Delphi Technique as a Forecasting Tool: Issues and Analysis.” *International Journal of Forecasting* 15: 353–75.

Rowe, Gene. 2007. “A Guide to Delphi.” *Foresight: The International Journal of Applied Forecasting*, no. 8: 11–16.

Sanders, Nada, Paul Goodwin, Dilek Önkal, Mustafa Sinan Gönül, Nigel Harvey, Anthony Lee, and Lucy Kjolso. 2005. “When and How Should Statistical Forecasts Be Judgmentally Adjusted?” *Foresight: The International Journal of Applied Forecasting* 1 (1): 5–23.

Eroglu, Cuneyt, and Keely L. Croxton. 2010. “Biases in Judgmental Adjustments of Statistical Forecasts: The Role of Individual Differences.” *International Journal of Forecasting* 26 (1): 116 –33.

Franses, Philip Hans, and Rianne Legerstee. 2013. “Do Statistical Forecasting Models for Sku-Level Data Benefit from Including Past Expert Knowledge?” *International Journal of Forecasting* 29 (1): 80–87.

Goodwin, Paul. 2000. “Correct or Combine? Mechanically Integrating Judgmental Forecasts with Statistical Methods.” *International Journal of Forecasting* 16 (2): 261–75.

Green, Kesten C., and J Scott Armstrong. 2007. “Structured Analogies for Forecasting.” *International Journal of Forecasting* 23 (3): 365–76.

Önkal, Dilek, Kadire Zeynep Sayim, and Mustafa Sinan Gönül. 2012. “Scenarios as Channels of Forecast Advice.” *Technological Forecasting and Social Change* 80: 772–88.

Morwitz, Vicki G., Joel H. Steckel, and Alok Gupta. 2007. “When Do Purchase Intentions Predict Sales?” *International Journal of Forecasting* 23 (3): 347–64.

Chapter 5 Linear regression models

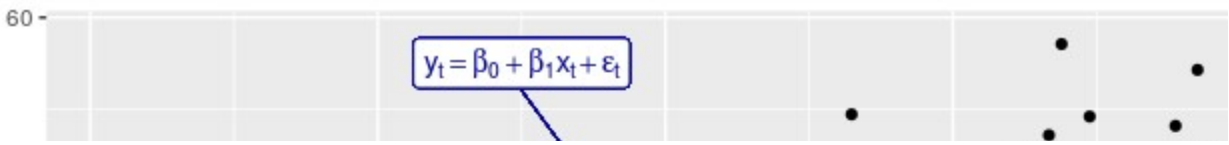
In this chapter we discuss linear regression models. The basic concept is that we forecast the time series of interest y assuming that it has a linear relationship with other time series x.

For example, we might wish to forecast monthly sales y with total advertising spend x as the predictor. Or we might forecast daily electricity demand y using temperature x1 and the day of week x2 as predictors.

The forecast variable y is sometimes also called the regressand, dependent or explained variable. The predictor variables x are sometimes also called the regressors, independent or explanatory variables. In this book we will always refer to them as the “forecast variable” and “predictor variables”.

Simple linear regression

In the simplest case, there is a linear relationship between the forecast variable and a single predictor variable, $y_t=\beta0+\beta1xt+\varepsilon t$. An example of data from such a model is shown in Figure 5.1. The parameters β0 and β1 denote the intercept and the slope of the line respectively. The intercept β0 represents the predicted value of y when x=0. The slope β1 represents the average predicted change in yx.



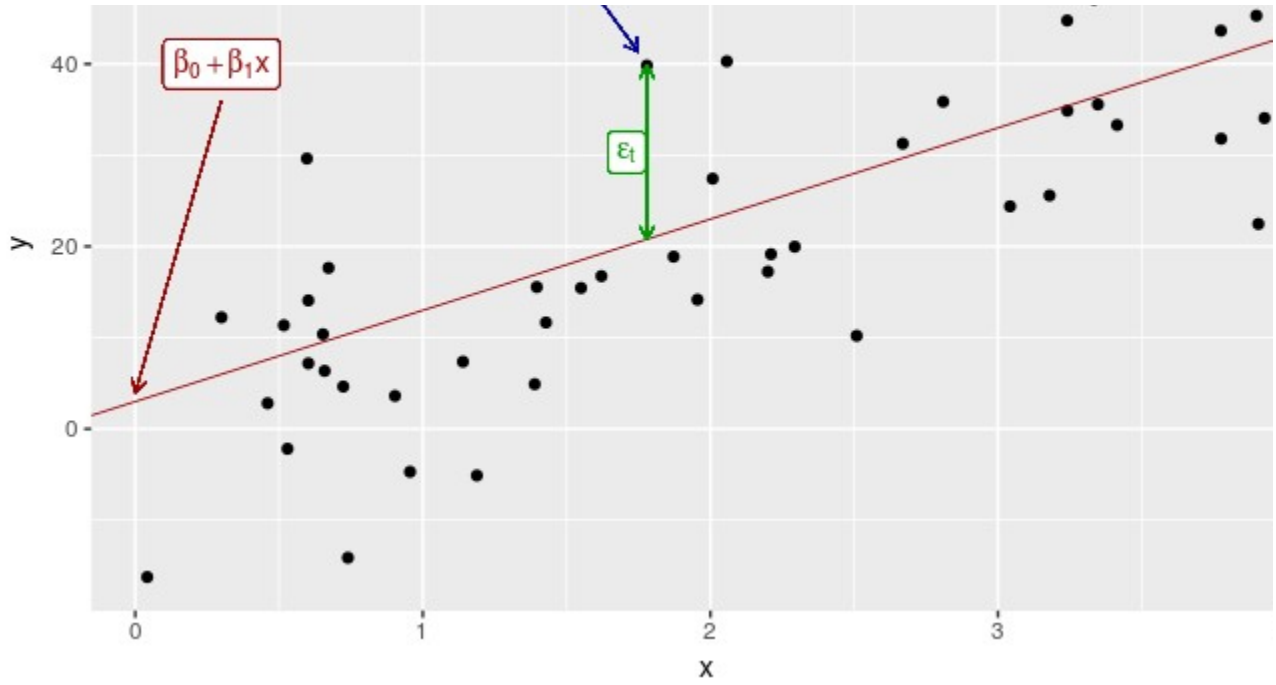


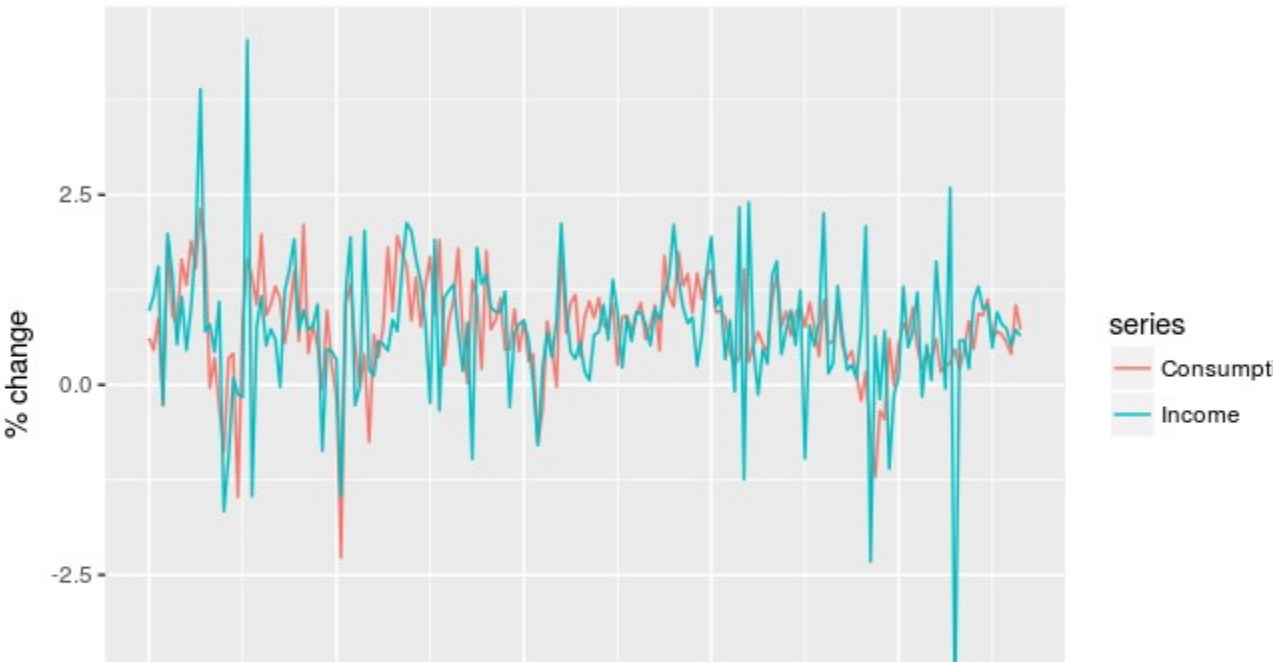
Figure 5.1: An example of data from a linear regression model.

Notice that the observations do not lie on the straight line but are scattered around it. We can think of each observation yt consisting of the systematic or explained part of the model, β0+β1xt, and the random "error", εt. The "error" term does not imply a mistake, but a deviation from the underlying straight line model. It captures anything that may affect yt other than xt.

Example: US consumption expenditure

Figure 5.2 shows time series of quarterly percentage changes (growth rates) of real personal consumption expenditure (y) and real personal disposable income (x) for the US from 1970 Q1 to 2016 Q3.

```
autoplot(uschange[,c("Consumption","Income")]) +
  ylab("% change") + xlab("Year")
```



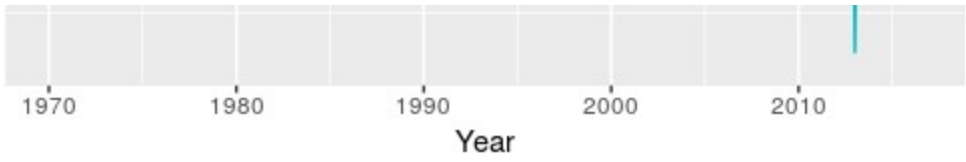


Figure 5.2: Percentage changes in personal consumption expenditure and personal income for the US.

A scatter plot of consumption against income is shown in Figure 5.3 along with the estimated regression line ^yt=0.55+0.28xt. (We put a "hat" above the y to indicate this is an estimate or prediction.)

```
uschange %>%
  as.data.frame %>%
  ggplot(aes(x=Income, y=Consumption)) +
    ylab("Consumption (quarterly % change)") +
    xlab("Income (quarterly % change)") +
    geom_point() +
    geom_smooth(method="lm", se=FALSE)
```

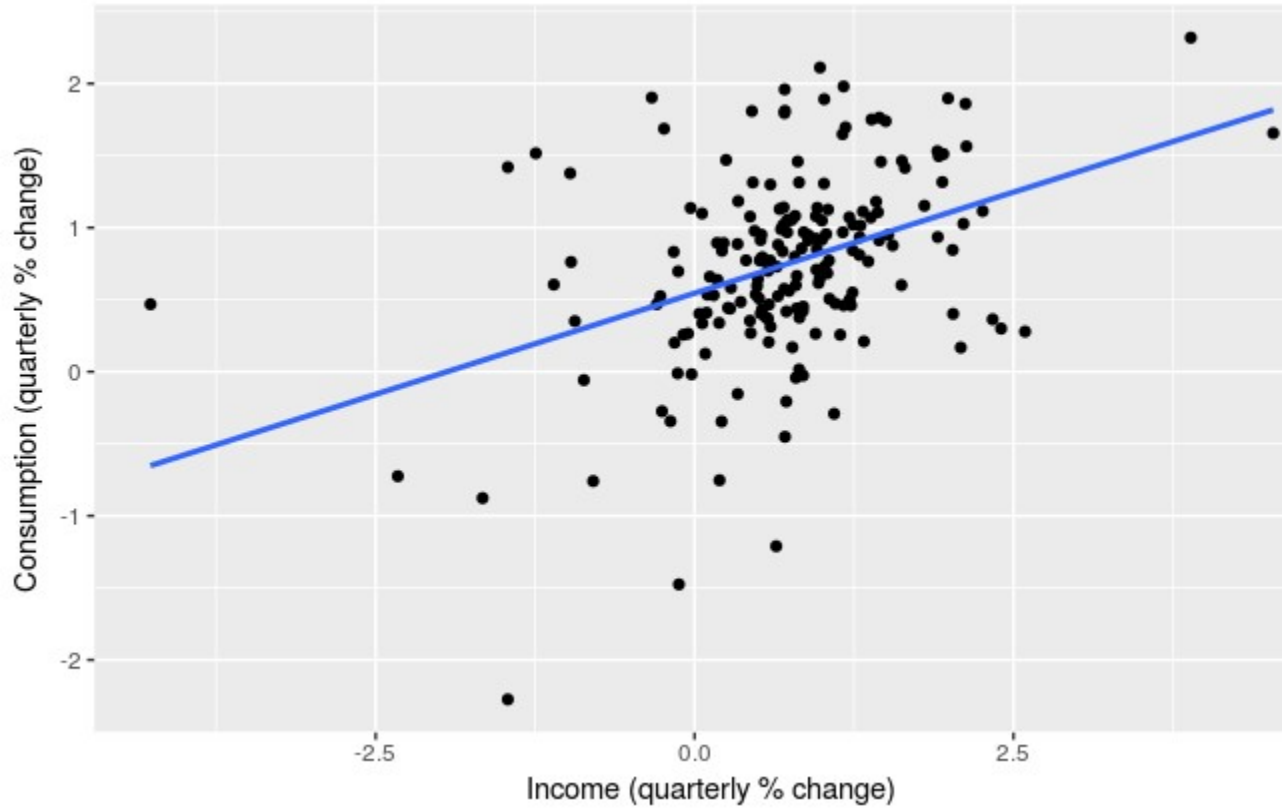


Figure 5.3: Scatterplot of consumption versus income and the fitted regression line.

This equation is estimated in R using the `tslm` function:

```
tslm(Consumption ~ Income, data=uschange)
#>
#> Call:
#> tslm(formula = Consumption ~ Income, data = uschange)
#>
#> Coefficients:
#> (Intercept)       Income
#>       0.545        0.281
```

We will discuss how `tslm` computes the coefficients in Section 5.2.

The fitted line has a positive slope, reflecting the positive relationship between income and consumption. The slope coefficient shows that a one unit increase in x (a 1% increase in personal disposable income) results on average in 0.28 units increase in y (an average 0.28% increase in personal consumption expenditure). Alternatively the estimated equation shows that a value of 1 for x (the percentage increase in personal disposable income) will result in a forecast value of $0.55+0.28\times1=0.83$ for y (the percentage increase in personal consumption expenditure).

The interpretation of the intercept requires that a value of x=0 makes sense. In this case when x=0 (i.e., when there is no change in personal disposable income since the last quarter) the predicted value of y is 0.55 (i.e., an average increase in personal consumption expenditure of 0.55%). Even when x=0 does not make sense, the intercept is an important part of the model. Without it, the slope coefficient can be distorted unnecessarily. The intercept should always be included unless the requirement is to force the regression line "through the origin". In what follows we assume that an intercept is always included in the model.

5.2 Least squares estimation

In practice, of course, we have a collection of observations but we do not know the values of the coefficients $\beta0,\beta1,\ldots,\beta k$. These need to be estimated from the data.

This is called "least squares" estimation because it gives the least value for the sum of squared errors. Finding the best estimates of the coefficients is often called "fitting" the model to the data, or sometimes "learning" or "training" the model. The line shown in Figure 5.3 was obtained in this way.

The `tslm` function fits a linear regression model to time series data. It is very similar to the `lm` function which is widely used for linear models, but `tslm` provides additional facilities for handling time series.

Example: US consumption expenditure (revisited)

A multiple linear regression model for US consumption is $yt=\beta0+\beta1x1,t+\beta2x2,t+\beta3x3,t+\beta4x4,t+\varepsilon t$, where y is the percentage change in real personal consumption expenditure, x1 is the percentage change in real personal disposable income, x2 is the percentage change in industrial production, x3 is the percentage change in personal savings and x4 is the change in the unemployment rate.

The following output provides information about the fitted model. The first column of `Coefficients` gives an estimate of each β coefficient and the second column gives its standard error (i.e., the standard deviation which would be obtained from repeatedly estimating the β coefficients on similar data sets). The standard error gives a measure of the uncertainty in the estimated β coefficient.

```
fit.consMR <- tslm(Consumption ~ Income + Production + Unemployment + Savings,
  data=uschange)
summary(fit.consMR)
#>
#> Call:
#> tslm(formula = Consumption ~ Income + Production + Unemployment +
#>     Savings, data = uschange)
#>
#> Residuals:
```

```
#>     Min      1Q  Median      3Q     Max
#> -0.8830 -0.1764 -0.0368  0.1525  1.2055
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.26729    0.03721    7.18  1.7e-11 ***
#> Income        0.71448    0.04219   16.93  < 2e-16 ***
#> Production    0.04589    0.02588    1.77    0.078 .
#> Unemployment -0.20477    0.10550   -1.94    0.054 .
#> Savings      -0.04527    0.00278  -16.29  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.329 on 182 degrees of freedom
#> Multiple R-squared:  0.754,  Adjusted R-squared:  0.749
#> F-statistic:  139 on 4 and 182 DF,  p-value: <2e-16
```

For forecasting purposes, the final two columns are of limited interest. The "t value" is the ratio of an estimated β coefficient to its standard error and the last column gives the p-value: the probability of the estimated β coefficient being as large as it is if there was no real relationship between the consumption and the corresponding predictor. This is useful when studying the effect of each predictor, but is not particularly useful when forecasting.

5.3 Some useful predictors

There are several very useful predictors that occur frequently when using regression for time series data.

Trend

It is very common for time series data to be trending. A linear trend can be modelled by simply using $x_{1,t}=t$ as a predictor, $y_t=\beta_0+\beta_1 t+\varepsilon_t$, where $t=1,\dots,T$. A trend variable can be specified in the `tslm` function using the `trend` predictor. In Section 5.8 we discuss how we can also model a nonlinear trends.

Dummy variables

So far, we have assumed that each predictor takes numerical values. But what about when a predictor is a categorical variable taking only two values (e.g., "yes" and "no"). Such a variable might arise, for example, when forecasting daily sales and you want to take account of whether the day is a **public holiday** or not. So the predictor takes value "yes" on a public holiday, and "no" otherwise.

This situation can still be handled within the framework of multiple regression models by creating a "dummy variable" taking value 1 corresponding to "yes" and 0 corresponding to "no". A dummy variable is also known as an "indicator variable".

A dummy variable can also be used to account for an **outlier** in the data. Rather than omit the outlier, a dummy variable removes its effect. In this case, the dummy variable takes value 1 for that observation and 0 everywhere else. An example is the case where a special event has occurred. For example when forecasting tourist arrivals to Brazil, we will need to account for the effect of the Rio de Janeiro summer Olympics in 2016.

If there are more than two categories, then the variable can be coded using several dummy variables (one fewer than the total number of categories). `tslm` will automatically handle this case

if you specify a factor variable as a predictor. There is usually no need to manually create the corresponding dummy variables.

Seasonal dummy variables

For example, suppose we are forecasting daily electricity demand and we want to account for the day of the week as a predictor. Then the following dummy variables can be created.

	d1,t	**d2,t**	**d3,t**	**d4,t**	**d5,t**	**d6,t**
Monday	1	0	0	0	0	0
Tuesday	0	1	0	0	0	0
Wednesday	0	0	1	0	0	0
Thursday	0	0	0	1	0	0
Friday	0	0	0	0	1	0
Saturday	0	0	0	0	0	1
Sunday	0	0	0	0	0	0
Monday	1	0	0	0	0	0
Tuesday	0	1	0	0	0	0
Wednesday	0	0	1	0	0	0
Thursday	0	0	0	1	0	0
Friday	0	0	0	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Notice that only six dummy variables are needed to code seven categories. That is because the seventh category (in this case Sunday) is specified when all dummy variables are all set to zero and is captured by the intercept.

Many beginners will try to add a seventh dummy variable for the seventh category. This is known as the “dummy variable trap” because it will cause the regression to fail. There will be one too many parameters to estimate when an intercept is also included. The general rule is to use one fewer dummy variables than categories. So for quarterly data, use three dummy variables; for monthly data, use 11 dummy variables; and for daily data, use six dummy variables, and so on.

The interpretation of each of the coefficients associated with the dummy variables is that it is *a measure of the effect of that category relative to the omitted category*. In the above example, the coefficient of d1,t associated with Monday will measure the effect of Monday compared to Sunday on the forecast variable. An example of interpreting estimated dummy variable coefficients capturing the quarterly seasonality of Australian beer production follows.

The `tslm` function will automatically handle this situation if you specify the predictor `season`.

Example: Australian quarterly beer production

Recall the Australian quarterly beer production data shown again in Figure 5.8 below.

```
beer2 <- window(ausbeer, start=1992)
autoplot(beer2) + xlab("Year") + ylab("Megalitres")
```

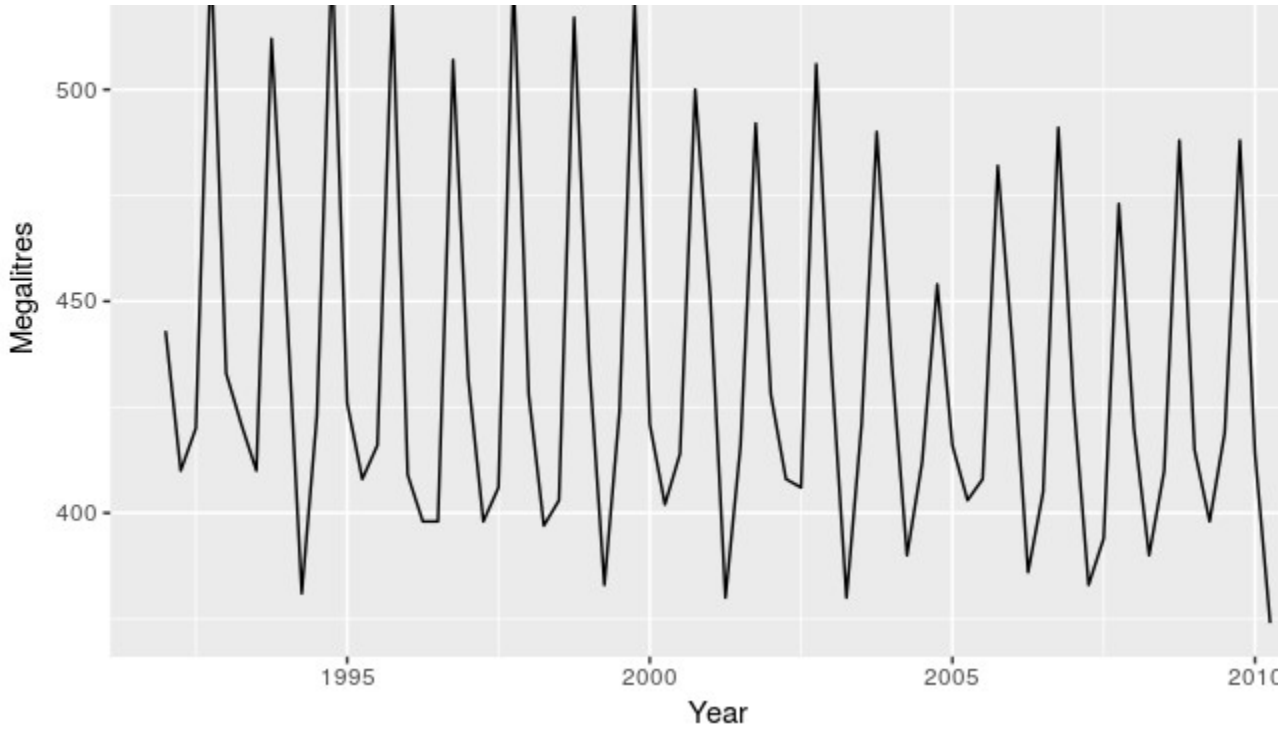


Figure 5.8: Australian quarterly beer production.

We want to forecast the value of future beer production. We can model this data using a regression model with a linear trend and quarterly dummy variables, $y_t=\beta_0+\beta_1 t+\beta_2 d_{2,t}+\beta_3 d_{3,t}+\beta_4 d_{4,t}+\varepsilon_t$, where $d_{i,t}=1$ if t is in quarter i and 0 otherwise. The first quarter variable has been omitted, so the coefficients associated with the other quarters are measures of the difference between those quarters and the first quarter.

```
fit.beer <- tslm(beer2 ~ trend + season)
summary(fit.beer)
#>
#> Call:
#> tslm(formula = beer2 ~ trend + season)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -42.90  -7.60  -0.46   7.99  21.79
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 441.8004     3.7335  118.33  < 2e-16 ***
#> trend        -0.3403     0.0666   -5.11  2.7e-06 ***
#> season2     -34.6597     3.9683   -8.73  9.1e-13 ***
#> season3     -17.8216     4.0225   -4.43  3.4e-05 ***
#> season4      72.7964     4.0230   18.09  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 12.2 on 69 degrees of freedom
#> Multiple R-squared:  0.924,  Adjusted R-squared:  0.92
#> F-statistic:  211 on 4 and 69 DF,  p-value: <2e-16
```

Note that `trend` and `season` are not objects in the R workspace; they are created automatically by `tslm` when specified in this way.

There is a downward trend of -0.34 megalitres per quarter. On average, the second quarter has production of 34.7 megalitres lower than the first quarter, the third quarter has production of 17.8 megalitres lower than the first quarter, and the fourth quarter has production of 72.8 megalitres higher than the first quarter.

```
autoplot(beer2, series="Data") +
  forecast::autolayer(fitted(fit.beer), series="Fitted") +
  xlab("Year") + ylab("Megalitres") +
  ggtitle("Quarterly Beer Production")
```

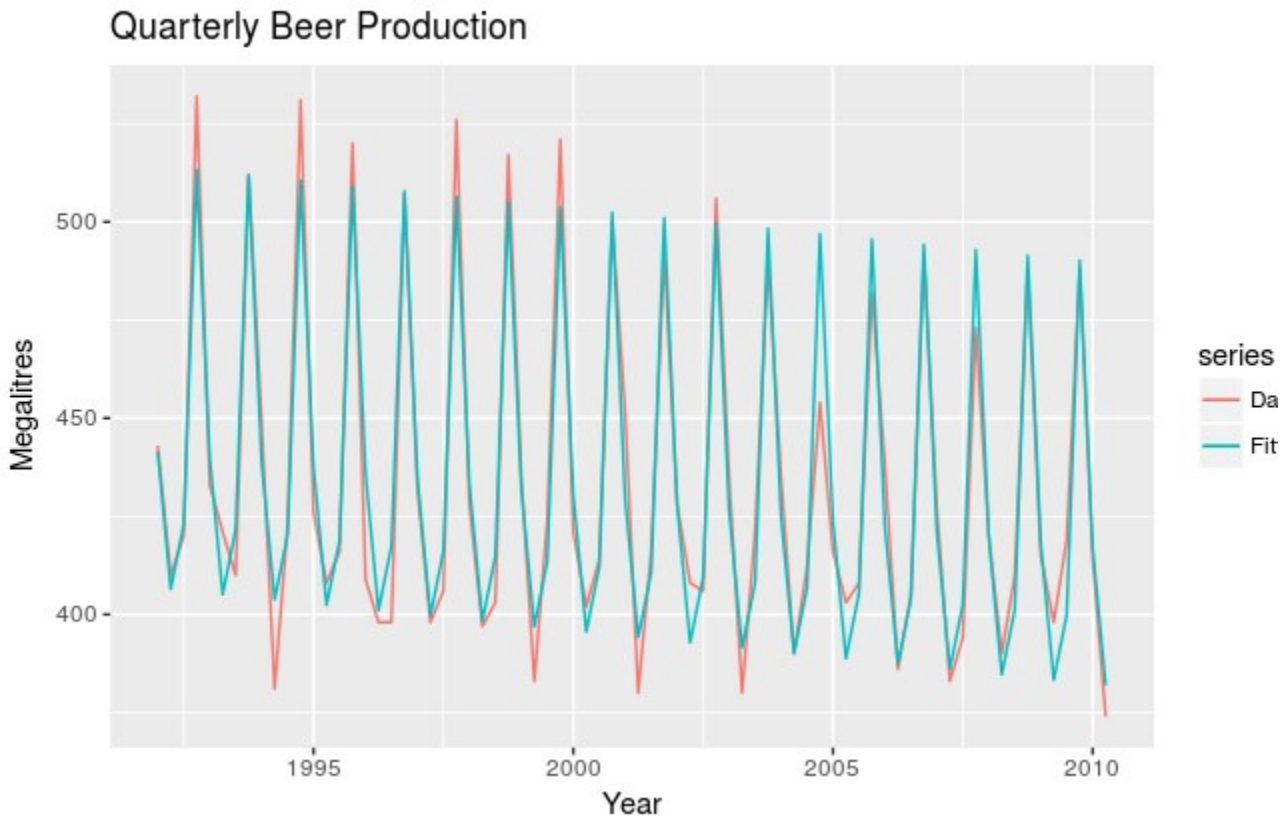
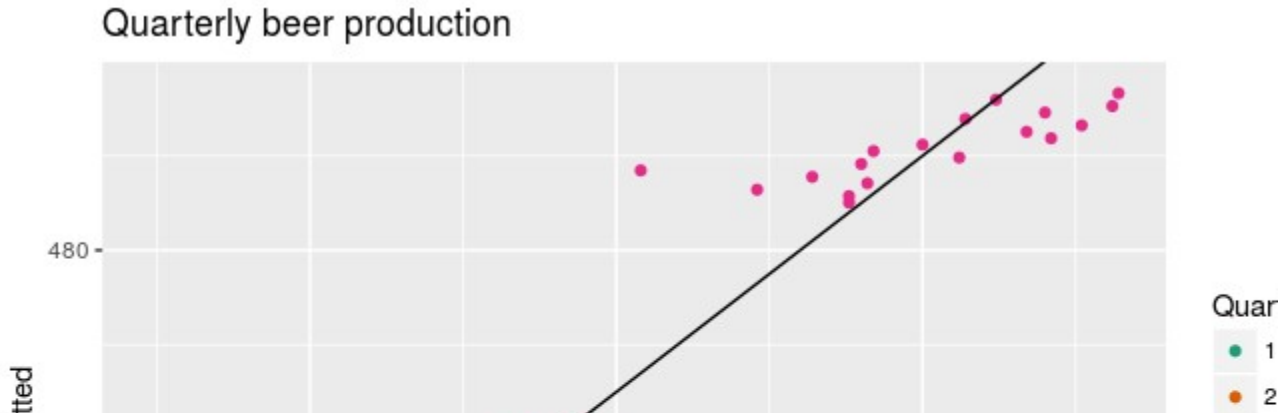


Figure 5.9: Time plot of beer production and predicted beer production.

```
cbind(Data=beer2, Fitted=fitted(fit.beer)) %>%
  as.data.frame %>%
  ggplot(aes(x=Data, y=Fitted, colour=as.factor(cycle(beer2)))) +
    geom_point() +
    ylab("Fitted") + xlab("Actual values") +
    ggtitle("Quarterly beer production") +
    scale_colour_brewer(palette="Dark2", name="Quarter") +
    geom_abline(intercept=0, slope=1)
```



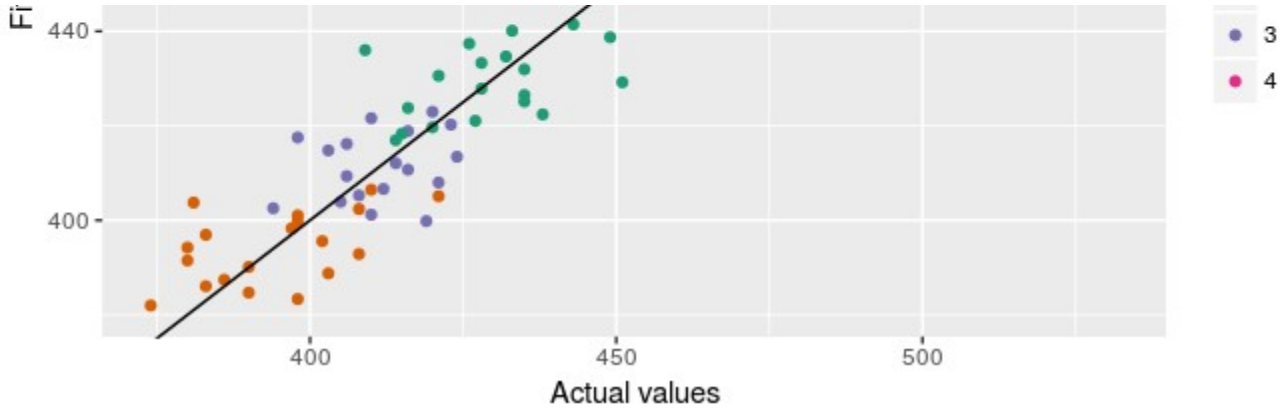


Figure 5.10: Actual beer production plotted against predicted beer production.

Fourier series

An alternative to using seasonal dummy variables, especially for long seasonal periods, is to use Fourier terms. Jean-Baptiste Fourier was a French mathematician, born in the 1700s, who showed that a series of sine and cosine terms of the right frequencies can approximate any periodic function. We can use them for seasonal patterns.

If m is the seasonal period, then the first few Fourier terms are given by x1,t=sin(2πtm),x2,t=cos(2πtm),x3,t=sin(4πtm),x4,t=cos(4πtm),x5,t=sin(6πtm),x6,t=cos(6πtm), and so on. If we have monthly seasonality, and we use the first 11 of these predictor variables, then we will get exactly the same forecasts as using 11 dummy variables.

The advantage of using Fourier terms is that we can often use fewer predictor variables than we need to with dummy variables, especially when m is large. This makes them useful for weekly data, for example, where m≈52. For short seasonal periods such as with quarterly data, there is little advantage in using Fourier series over seasonal dummy variables.

In R, these Fourier terms are produced using the `fourier` function. For example, the Australian beer data can be modelled like this.

```
fourier.beer <- tslm(beer2 ~ trend + fourier(beer2, K=2))
summary(fourier.beer)
#>
#> Call:
#> tslm(formula = beer2 ~ trend + fourier(beer2, K = 2))
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -42.90  -7.60  -0.46   7.99  21.79
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                446.8792     2.8732  155.53  < 2e-16 ***
#> trend                       -0.3403     0.0666   -5.11  2.7e-06 ***
#> fourier(beer2, K = 2)S1-4    8.9108     2.0112    4.43  3.4e-05 ***
#> fourier(beer2, K = 2)C1-4   53.7281     2.0112   26.71  < 2e-16 ***
#> fourier(beer2, K = 2)C2-4   13.9896     1.4226    9.83  9.3e-15 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 12.2 on 69 degrees of freedom
```

```
#> Multiple R-squared:  0.924,  Adjusted R-squared:  0.92
#> F-statistic:  211 on 4 and 69 DF,  p-value: <2e-16
```

The first argument to `fourier` just allows it to identify the seasonal period m and the length of the predictors to return. The second `K` argument specifies how many pairs of sin and cos terms to include. The maximum number allowed is K=m/2 where m is the seasonal period. Because we have used the maximum here, the results are identical to those obtained when using seasonal dummy variables.

If only the first two Fourier terms are used (x1,t and x2,t), the seasonal pattern will follow a simple sine wave. A regression model containing Fourier terms is often called a **harmonic regression** because the successive Fourier terms represent harmonics of the first two Fourier terms.

Intervention variables

It is often necessary to model interventions that may have affected the variable to be forecast. For example, competitor activity, advertising expenditure, industrial action, and so on, can all have an effect.

When the effect lasts only for one period, we use a spike variable. This is a dummy variable taking value one in the period of the intervention and zero elsewhere. A spike variable is equivalent to a dummy variable for handling an outlier.

Other interventions have an immediate and permanent effect. If an intervention causes a level shift (i.e., the value of the series changes suddenly and permanently from the time of intervention), then we use a step variable. A step variable takes value zero before the intervention and one from the time of intervention onward.

Another form of permanent effect is a change of slope. Here the intervention is handled using a piecewise linear trend; a trend that bends at the time of intervention and hence is nonlinear. We will discuss this in Section 5.8.

Trading days

The number of trading days in a month can vary considerably and can have a substantial effect on sales data. To allow for this, the number of trading days in each month can be included as a predictor.

For monthly or quarterly data, the `bizdays` function will compute the number of trading days in each period.

An alternative that allows for the effects of different days of the week has the following predictors: x1=# Mondays in month;x2=# Tuesdays in month;⋮x7=# Sundays in month.

Distributed lags

It is often useful to include advertising expenditure as a predictor. However, since the effect of advertising can last beyond the actual campaign, we need to include lagged values of advertising expenditure. So the following predictors may be used. x1=advertising for previous month;x2=advertising for two months previously;⋮xm=advertising for m months previously.

It is common to require the coefficients to decrease as the lag increases, although this is beyond the scope of this book.

Easter

Easter is different from most holidays because it is not held on the same date each year and the effect can last for several days. In this case, a dummy variable can be used with value one where the holiday falls in the particular time period and zero otherwise.

For example, with monthly data, when Easter falls in March then the dummy variable takes value 1 in March, when it falls in April, the dummy variable takes value 1 in April, and when it starts in March and finishes in April, the dummy variable is split proportionally between months.

The `easter` function will compute the dummy variable for you.

5.4 Evaluating the regression model

The differences between the observed y values and the corresponding fitted values are the training-set errors or "residuals" defined as, $et=yt−\hat{}yt=yt−\hat{}β0−\hat{}β1x1,t−\hat{}β2x2,t−⋯−\hat{}βkxk,t$

for $t=1,…,T$. Each residual is the unpredictable component of the associated observation.

The residuals have some useful properties including the following two: $T∑t=1et=0andT∑t=1xk,tet=0$for all k. As a result of these properties, it is clear that the average of the residuals is zero, and that the correlation between the residuals and the observations for the predictor variable is also zero. (This is not necessarily true when the intercept is omitted from the model.)

After selecting the regression variables and fitting a regression model, it is necessary to plot the residuals to check that the assumptions of the model have been satisfied. There are a series of plots that should be produced in order to check different aspects of the fitted model and the underlying assumptions.

ACF plot of residuals

With time series data, it is highly likely that the value of a variable observed in the current time period will be similar to its value in the previous period, or even the period before that, and so on. Therefore when fitting a regression model to time series data, it is very common to find autocorrelation in the residuals. In this case, the estimated model violates the assumption of no autocorrelation in the errors, and our forecasts may be inefficient — there is some information left over which should be accounted for in the model in order to obtain better forecasts. The forecasts from a model with autocorrelated errors are still unbiased, and so are not "wrong", but they will usually have larger prediction intervals than they need to. Therefore we should always look at an ACF plot of the residuals.

Another useful test of autocorrelation in the residuals designed to take account for the regression model is the **Breusch-Godfrey** test, also referred to as the LM (Lagrange Multiplier) test for serial correlation. It is used to test the joint hypothesis that there is no autocorrelation in the residuals up to a certain specified order. A small p-value indicates there is significant autocorrelation remaining in the residuals.

The Breusch-Godfrey test is similar to the Ljung-Box test, but it is specifically designed for use with regression models.

Histogram of residuals

It is always a good idea to check if the residuals are normally distributed. As explained earlier, this is not essential for forecasting, but it does make the calculation of prediction intervals much easier.

Example

Using the `checkresiduals` function introduced in Section 3.3, we can obtain all the useful residual diagnostics mentioned above. Figure 5.11 shows a time plot, the ACF and the histogram of the residuals from the model fitted to the beer production data as well as the Breusch-Godfrey test for testing up to 8th order autocorrelation. (The `checkresiduals` function will use the Breusch-Godfrey test for regression models, but the Ljung-Box test otherwise.)

The time plot shows a large negative value in the residuals at 2004 Q4, which suggests that something unusual may have happened in that quarter. It would be worth investigating to see if there were any unusual circumstances or events which may have significantly reduced beer production for that quarter.

The remaining residuals show that the model has captured the patterns in the data quite well. However, there is a small amount of autocorrelation left in the residuals, seen in the significant spike in the ACF plot but not detected to be significant by the Breusch-Godfrey test. This suggests that there may be some information remaining in the residuals and the model can be slightly improved to capture this. However, it is unlikely that it will make much difference to the resulting forecasts. The forecasts from the current model are still unbiased, but will have larger prediction intervals than they need to. In Chapter 9 we discuss dynamic regression models used for better capturing information left in the residuals.

The histogram shows that the residuals from modelling the beer data seem to be slightly negatively skewed, although that is due to the large negative value in 2004 Q4.

```
checkresiduals(fit.beer)
```

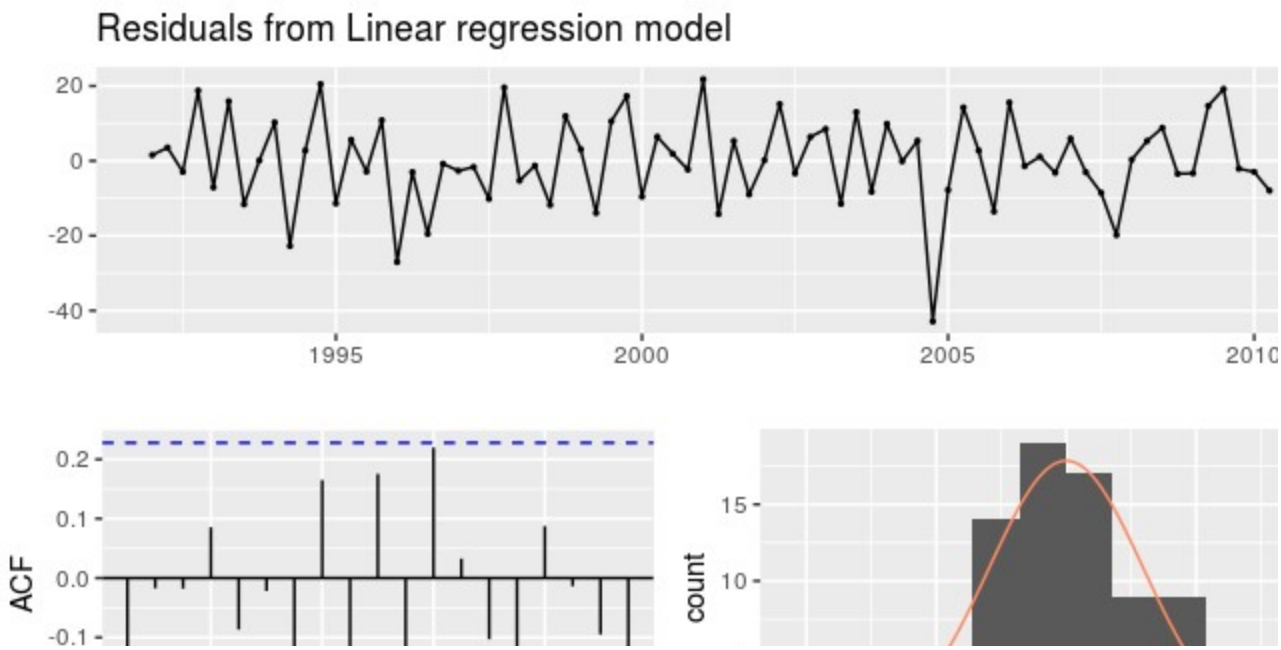


Figure 5.11: Analysing the residuals from a regression model for beer production.

```
#>
#>  Breusch-Godfrey test for serial correlation of order up to 8
#>
#> data:  Residuals from Linear regression model
#> LM test = 9, df = 8, p-value = 0.3
```

Residual plots against predictors

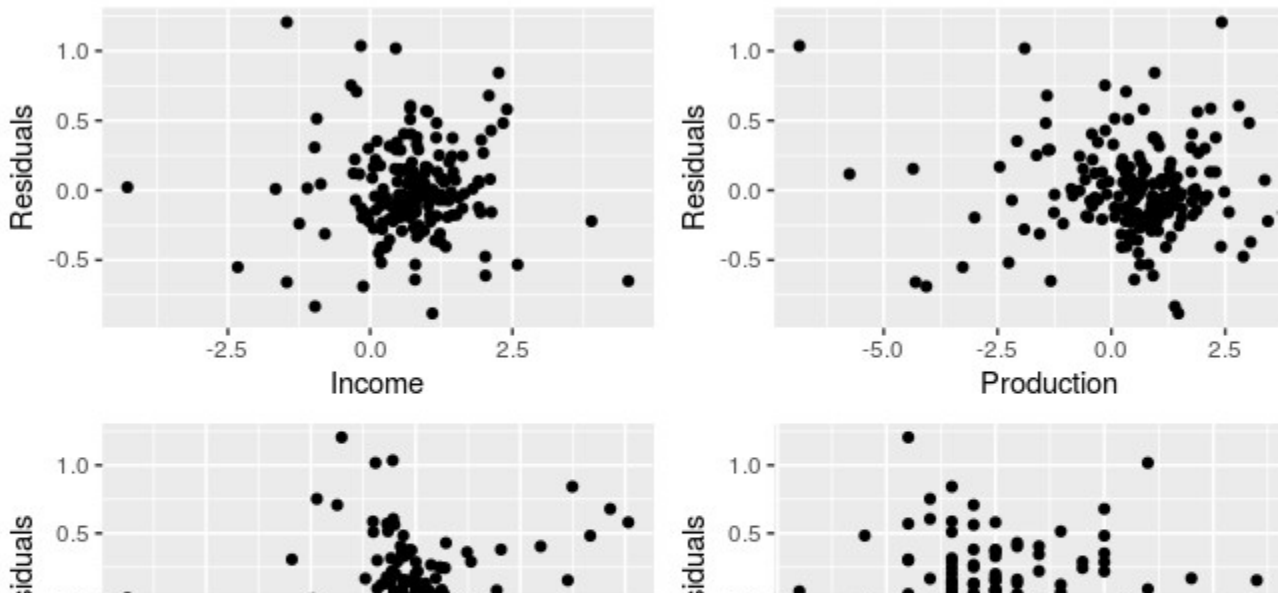
We would expect the residuals to be randomly scattered without showing any systematic patterns. A simple and quick way to check this is to examine a scatterplot of the residuals against each of the predictor variables. If these scatterplots show a pattern, then the relationship may be nonlinear and the model will need to be modified accordingly. See Section 5.8 for a discussion of nonlinear regression.

It is also necessary to plot the residuals against any predictors *not* in the model. If these show a pattern, then the predictor may need to be added to the model (possibly in a nonlinear form).

Example

The residuals from the multiple regression model for forecasting US consumption plotted against each predictor in Figure 5.12 seem to be randomly scattered and therefore we are satisfied with these in this case.

```
df <- as.data.frame(uschange)
df[,"Residuals"]  <- as.numeric(residuals(fit.consMR))
p1 <- ggplot(df, aes(x=Income, y=Residuals)) + geom_point()
p2 <- ggplot(df, aes(x=Production, y=Residuals)) + geom_point()
p3 <- ggplot(df, aes(x=Savings, y=Residuals)) + geom_point()
p4 <- ggplot(df, aes(x=Unemployment, y=Residuals)) + geom_point()
gridExtra::grid.arrange(p1, p2, p3, p4, nrow=2)
```



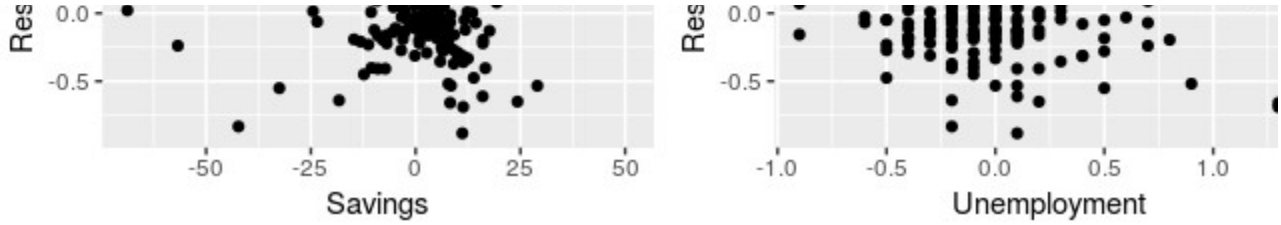


Figure 5.12: Scatterplots of residuals versus each predictor.

Residual plots against fitted values

A plot of the residuals against the fitted values should also show no pattern. If a pattern is observed, there may be "heteroscedasticity" in the errors which means that the variance of the residuals may not be constant. If this problem occurs, a transformation of the forecast variable (such as a logarithm or square root) may be required.

Example

Figure 5.13 shows the residuals against the fitted values from fitting a linear trend and seasonal dummies to the electricity data we first encountered in Section 3.2. The plot shows that the trend in the data is nonlinear and that the forecast variable requires a transformation as we observe a heteroscedastic pattern with variation increasing along the x-axis.

```
fit <- tslm(elec ~ trend + season)
cbind(Fitted=fitted(fit), Residuals=residuals(fit)) %>%
  as.data.frame %>%
  ggplot(aes(x=Fitted, y=Residuals)) + geom_point()
```

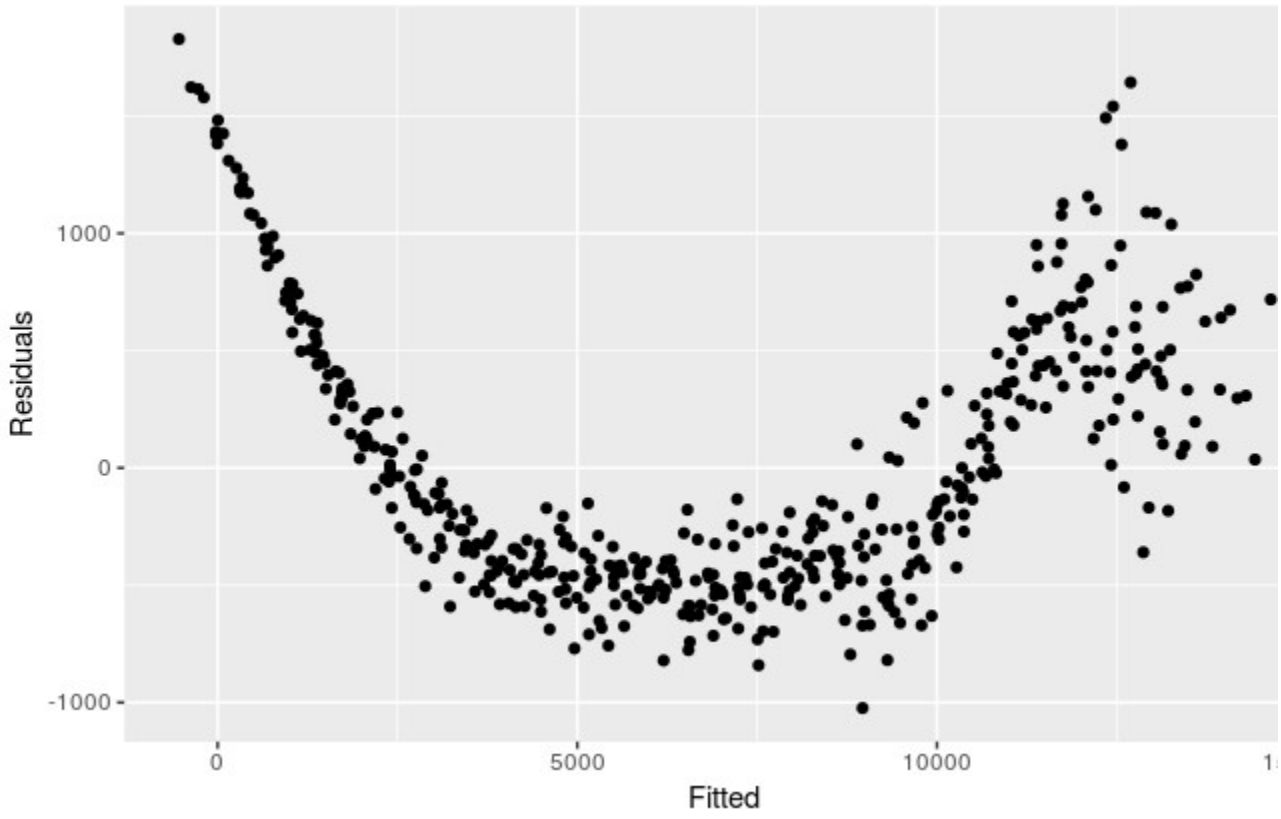


Figure 5.13: Scatterplot of residuals versus fitted.

```
##TODO
## A little too obvious. Can we find something more subtle?
## George: Will try to
```

Outliers and influential observations

Observations that take on extreme values compared to the majority of the data are called "outliers". Observations that have a large influence on the estimation results of a regression model are called "influential observations". Usually, influential observations are also outliers that are extreme in the x direction.

There are formal methods for detecting outliers and influential observations that are beyond the scope of this textbook. As we suggested at the beginning of Chapter 2, getting familiar with your data prior to performing any analysis is of vital importance. A scatter plot of y against each x is always a useful starting point in regression analysis and often helps to identify unusual observations.

One source for an outlier occurring is an incorrect data entry. Simple descriptive statistics of your data can identify minima and maxima that are not sensible. If such an observation is identified, and it has been incorrectly recorded, it should be immediately corrected or removed from the sample.

Outliers also occur when some observations are simply different. In this case it may not be wise for these observations to be removed. If an observation has been identified as a likely outlier, it is important to study it and analyze the possible reasons behind it. The decision of removing or retaining such an observation can be a challenging one (especially when outliers are influential observations). It is wise to report results both with and without the removal of such observations.

Example

Figure 5.14 highlights the effect of a single outlier when regressing US consumption on income (the example introduced in Section 5.1). In the left panel the outlier is only extreme in the direction of y as the percentage change in consumption has been incorrectly recorded as -4%. The red line is the regression line fitted to the data which includes the outlier, compared to the black line which is the line fitted to the data without the outlier. In the right panel the outlier now is also extreme in the direction of x with the 4% decrease in consumption corresponding to a 6% increase in income. In this case the outlier is very influential as the red line now deviates substantially from the black line.

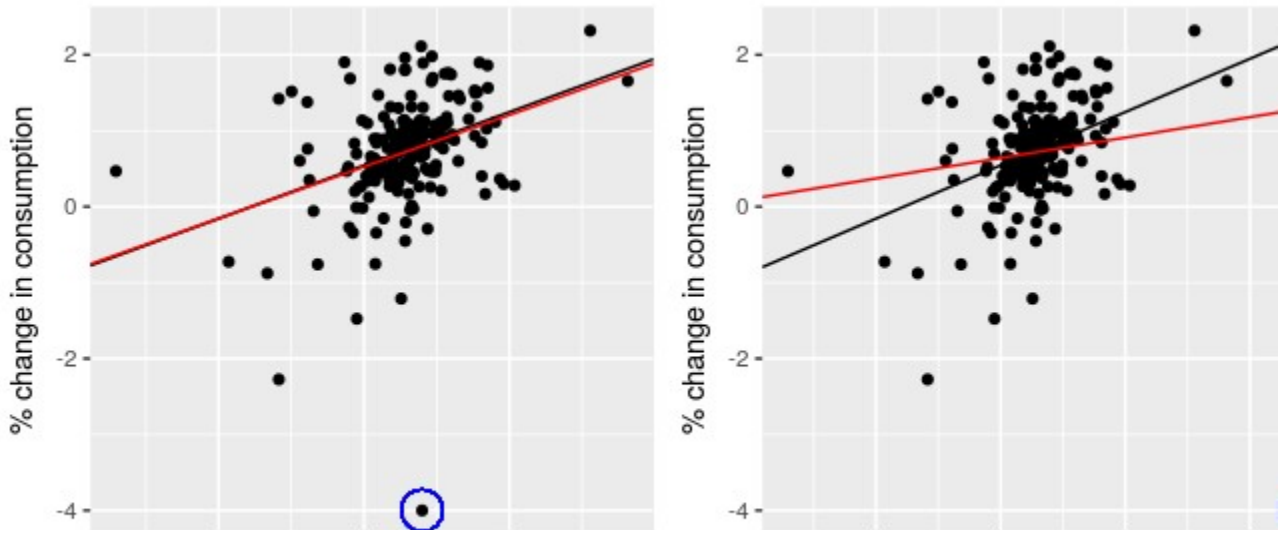


Figure 5.14: The effect of outliers and influential observations on regression

Goodness-of-fit

A common way to summarise how well a linear regression model fits the data is via the coefficient of determination or R2. This can be calculated as the square of the correlation between the observed y values and the predicted ˆy values. Alternatively, it can also be calculated as, R2=∑(ˆyt−¯y)2∑(yt−¯y)2, where the summations are over all observations. Thus, it reflects the proportion of variation in the forecast variable that is accounted for (or explained) by the regression model.

In simple linear regression, the value of R2 is also equal to the square of the correlation between y and x (provided an intercept has been included).

If the predictions are close to the actual values, we would expect R2 to be close to 1. On the other hand, if the predictions are unrelated to the actual values, then R2=0 (again, assuming there is an intercept). In all cases, R2 lies between 0 and 1.

The R2 value is commonly used, often incorrectly, in forecasting. R2 will never decrease when adding an extra predictor to the model and this can lead to over-fitting. There are no set rules for what is a good R2 value, and typical values of R2 depend on the type of data used. Validating a model's forecasting performance on the test data is much better than measuring the R2 value on the training data.

Example

Recall the example of US consumption data discussed in Section 5.2. The output for the estimated model is reproduced here for convenience:

```
fit.consMR <- tslm(Consumption ~ Income + Production + Unemployment + Savings,
  data=uschange)
summary(fit.consMR)
#>
#> Call:
#> tslm(formula = Consumption ~ Income + Production + Unemployment +
#>     Savings, data = uschange)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -0.8830 -0.1764 -0.0368  0.1525  1.2055
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.26729    0.03721    7.18  1.7e-11 ***
#> Income        0.71448    0.04219   16.93  < 2e-16 ***
#> Production    0.04589    0.02588    1.77    0.078 .
#> Unemployment -0.20477    0.10550   -1.94    0.054 .
#> Savings      -0.04527    0.00278  -16.29  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.329 on 182 degrees of freedom
```

```
#> Multiple R-squared:  0.754,  Adjusted R-squared:  0.749
#> F-statistic:  139 on 4 and 182 DF,  p-value: <2e-16
```

Figure 5.7 plots the actual consumption expenditure values versus the fitted values. The correlation between these variables is r=0.868 hence R2=75.4% (shown in the output above). In this case model does an excellent job as it explains most of the variation in the consumption data. Compare that to the R2 value of 16% obtained from the simple regression with the same data set in Section 5.1. Adding the three extra predictors has allowed a lot more of the variance in the consumption data to be explained.

Standard error of the regression

Another measure of how well the model has fitted the data is the standard deviation of the residuals, which is often known as the "residual standard error". This is shown in the above output with the value 0.329.

It is calculated by $\hat{\sigma}e=\sqrt{1T-k-1T}\sum t=1e2t$.

Notice that here we divide by $T-k-1$ in order to account for the number of estimated parameters in computing the residuals. Normally, we only estimate the mean (i.e., one parameter) when computing a standard deviation. The divisor is always T minus the number of parameters estimated in the calculation. Here, we divide by $T-k-1$ because we have estimated $k+1$ parameters (the intercept and a coefficient for each predictor variable) in computing the residuals.

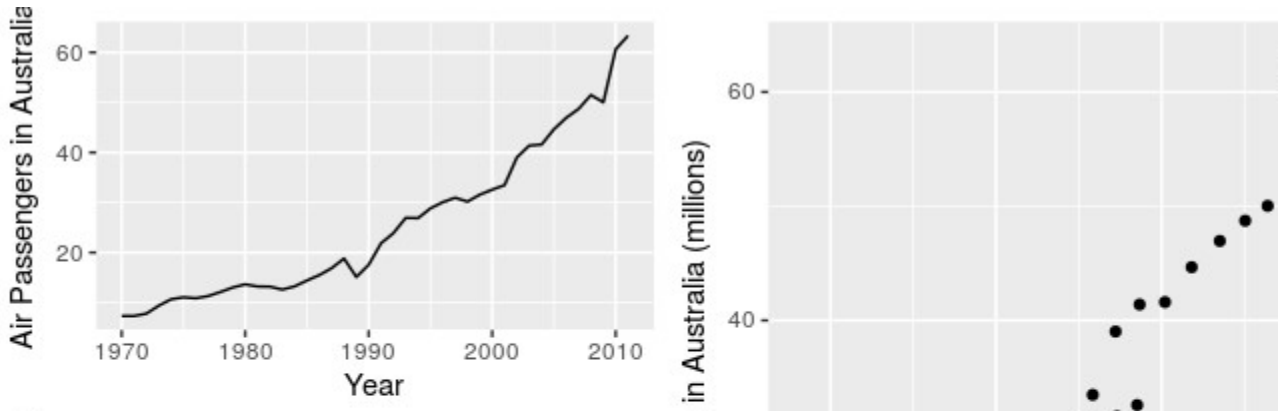
The standard error is related to the size of the average error that the model produces. We can compare this error to the sample mean of y or with the standard deviation of y to gain some perspective on the accuracy of the model.

We should warn here that the evaluation of the standard error can be highly subjective as it is scale dependent. The main reason we introduce it here is that it is required when generating forecast intervals, discussed in Section 5.6.

Spurious regression

More often than not, time series data are "non-stationary"; that is, the values of the time series do not fluctuate around a constant mean or with a constant variance. We will deal with time series stationarity in more detail in Chapter 8, but here we need to address the effect that non-stationary data can have on regression models.

For example consider the two variables plotted below in Figure 5.15. These appear to be related simply because they both trend upwards in the same manner. However, air passenger traffic in Australia has nothing to do with rice production in Guinea.



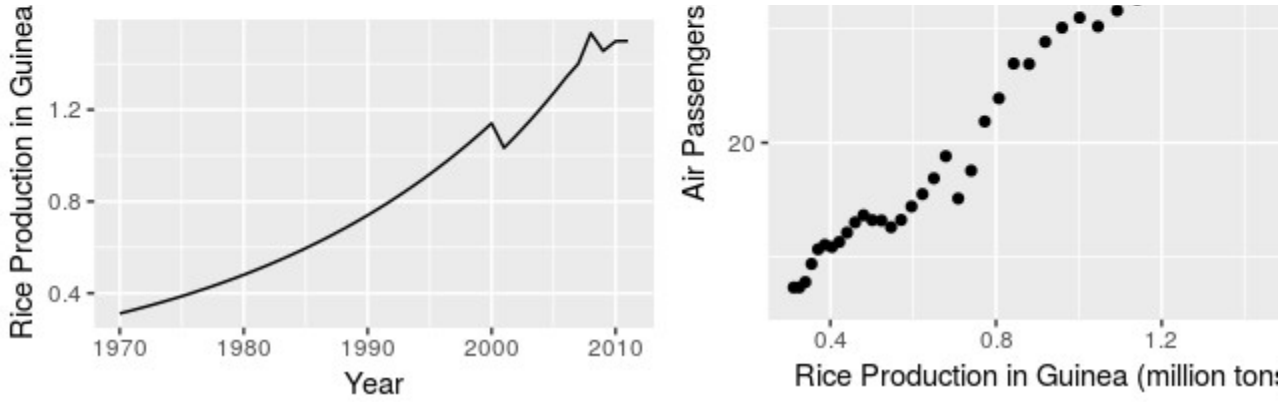
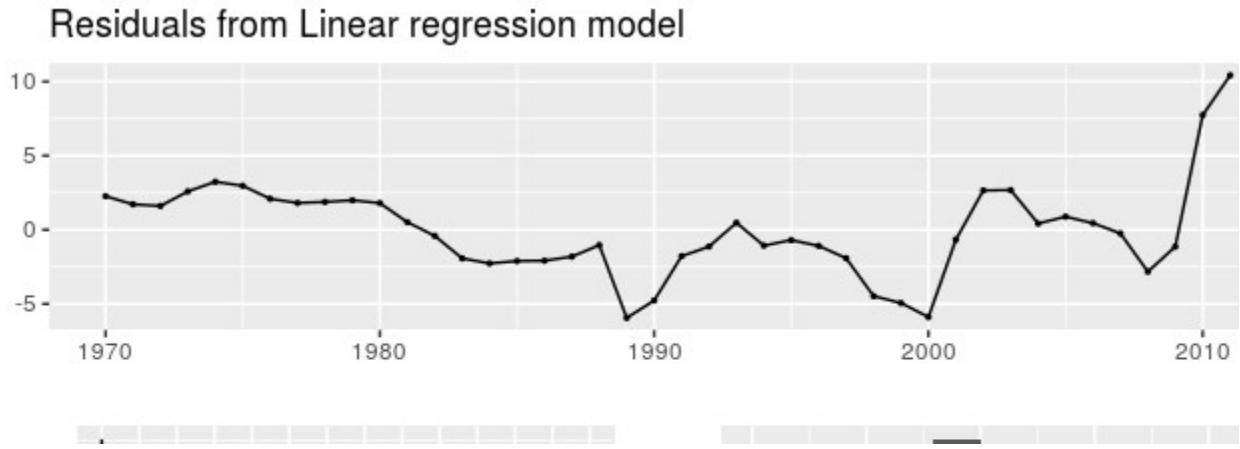


Figure 5.15: Trending time series data can appear to be related as shown in this example in which air passengers in Australia are regressed against rice production in Guinea.

Regressing non-stationary time series can lead to spurious regressions. The output of regressing Australian air passengers on rice production in Guinea is shown below. High R2 and high residual autocorrelation can be signs of spurious regression. We discuss the issues surrounding non-stationary data and spurious regressions in more detail in Chapter 9.

Cases of spurious regression might appear to give reasonable short-term forecasts, but they will generally not continue to work into the future.

```
fit <- tslm(aussies ~ guinearice)
summary(fit)
#>
#> Call:
#> tslm(formula = aussies ~ guinearice)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -5.945 -1.892 -0.327  1.862 10.421
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    -7.49       1.20   -6.23  2.3e-07 ***
#> guinearice     40.29       1.34   30.13  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.24 on 40 degrees of freedom
#> Multiple R-squared:  0.958,  Adjusted R-squared:  0.957
#> F-statistic:  908 on 1 and 40 DF,  p-value: <2e-16
checkresiduals(fit)
```



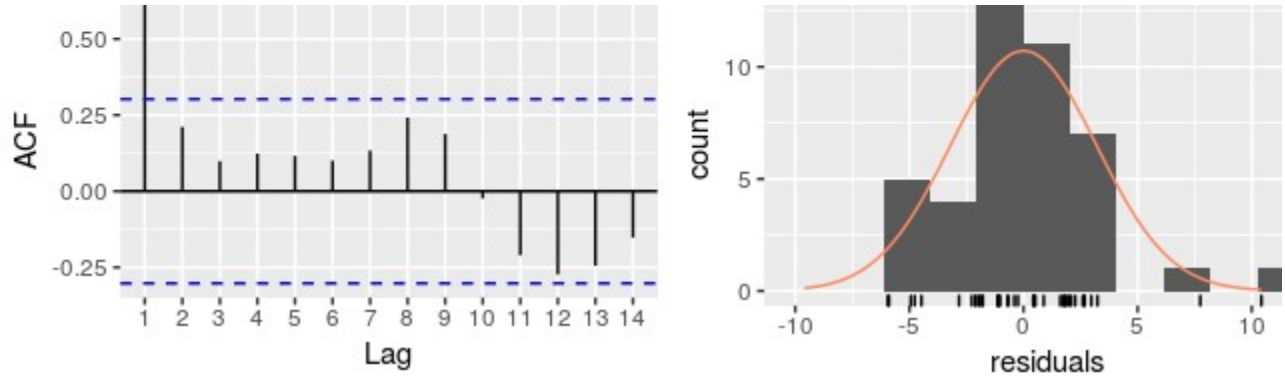


Figure 5.16: Residuals from a spurious regression.

```
#>
#>  Breusch-Godfrey test for serial correlation of order up to 10
#>
#> data:  Residuals from Linear regression model
#> LM test = 30, df = 10, p-value = 6e-04
```

5.5 Selecting predictors

When there are many possible predictors, we need some strategy to select the best predictors to use in a regression model.

A common approach that is *not recommended* is to plot the forecast variable against a particular predictor and if it shows no noticeable relationship, drop it. This is invalid because it is not always possible to see the relationship from a scatterplot, especially when the effects of other predictors have not been accounted for.

Another common approach which is also invalid is to do a multiple linear regression on all the predictors and disregard all variables whose p-values are greater than 0.05. To start with, statistical significance does not always indicate predictive value. Even if forecasting is not the goal, this is not a good strategy because the p-values can be misleading when two or more predictors are correlated with each other (see Section 5.9).

Instead, we will use a measure of predictive accuracy. Five such measures are introduced in this section.

Adjusted R2

Computer output for regression will always give the R2 value, discussed in Section 5.1. However, it is not a good measure of the predictive ability of a model. Imagine a model which produces forecasts that are exactly 20% of the actual values. In that case, the R2 value would be 1 (indicating perfect correlation), but the forecasts are not be very close to the actual values.

In addition, R2 does not allow for “degrees of freedom”. Adding *any* variable tends to increase the value of R2, even if that variable is irrelevant. For these reasons, forecasters should not use R2 to determine whether a model will give good predictions.

An equivalent idea is to select the model which gives the minimum sum of squared errors (SSE), given by SSE=T∑t=1e2t.

Minimizing the SSE is equivalent to maximizing R2 and will always choose the model with the most variables, and so is not a valid way of selecting predictors.

An alternative, designed to overcome these problems, is the adjusted R2 (also called "R-bar-squared"): $\bar{R}^2 = 1-(1-R^2)\frac{T-1}{T-k-1}$, where T is the number of observations and k is the number of predictors. This is an improvement on R2 as it will no longer increase with each added predictor. Using this measure, the best model will be the one with the largest value of $\bar{R}^2$. Maximizing $\bar{R}^2$ is equivalent to minimizing the standard error $\hat{\sigma}_e$ given \tag{5.3}ref(#eq:Regr-se).

Maximizing $\bar{R}^2$ works quite well as a method of selecting predictors, although it does tend to err on the side of selecting too many predictors.

Cross-validation

Section 3.4 introduced time series cross-validation as a general and useful tool for determining the predictive ability of a model. For regression models, it is also possible to use classical leave-one-out cross-validation to selection predictors (Bergmeir, Hyndman, and Koo 2015). This is faster and makes more efficient use of the data. The procedure uses the following steps:

1. Remove observation t from the data set, and fit the model using the remaining data. Then compute the error $(e^*_t = y_t - \hat{y}_t)$ for the omitted observation. (This is not the same as the residual because the tth observation was not used in estimating the value of $\hat{y}_t$.)
2. Repeat step 1 for $t=1,\dots,T$.
3. Compute the MSE from $e^*_1,\dots,e^*_T$. We shall call this the CV.

Although this looks this looks like a time-consuming procedure there are very fast methods of calculating CV, so that it takes no longer than fitting one model to the full data set. The equation for computing CV efficiently is given in Section 5.7.

Under this criterion, the best model is the one with the smallest value of CV.

```
CV(fit.consMR)
#>       CV      AIC     AICc      BIC    AdjR2
#>    0.116 -409.298 -408.831 -389.911    0.749
```

Akaike's Information Criterion

A closely-related method is Akaike's Information Criterion, which we define as $AIC = T\log(SSE_T) + 2(k+2)$, where T is the number of observations used for estimation and k is the number of predictors in the model. Different computer packages use slightly different definitions for the AIC, although they should all lead to the same model being selected. The k+2 part of the equation occurs because there are k+2 parameters in the model: the k coefficients for the predictors, the intercept and the variance of the residuals. The idea here is to penalize the fit of the model (SSE) with the number of parameters that need to be estimated.

The model with the minimum value of the AIC is often the best model for forecasting. For large values of T, minimizing the AIC is equivalent to minimizing the CV value.

Corrected Akaike's Information Criterion

For small values of T, the AIC tends to select too many predictors, and so a bias-corrected version of the AIC has been developed, $AICc=AIC+\frac{2(k+2)(k+3)}{T-k-3}$. As with the AIC, the AICc should be minimized.

Schwarz Bayesian Information Criterion

A related measure is Schwarz's Bayesian Information Criterion (known as SBIC, BIC or SC), $BIC=T\log(SSE_T)+(k+2)\log(T)$. As with the AIC, minimizing the BIC is intended to give the best model. The model chosen by BIC is either the same as that chosen by AIC, or one with fewer terms. This is because the BIC penalizes the number of parameters more heavily than the AIC. For large values of T, minimizing BIC is similar to leave-v-out cross-validation when $v=T\left[1-\frac{1}{(\log(T)-1)}\right]$.

Many statisticians like to use BIC because it has the feature that if there is a true underlying model, then with enough data the BIC will select that model. However, in reality there is rarely if ever a true underlying model, and even if there was a true underlying model, selecting that model will not necessarily give the best forecasts (because the parameter estimates may not be accurate). Consequently, we prefer to use the AICc, AIC, or CV statistics, which have forecasting as their objective (and which give equivalent models for large T).

Example: US consumption

To obtain all these measures in R, use `CV(fit)`. In the multiple regression example for forecasting US consumption we considered four predictors. With four predictors, there are 24=16 possible models. Now we can check if all four predictors are actually useful, or whether we can drop one or more of them. All 16 models were fitted and the results are summarised below in Table 5.1. A "1" indicates that the predictor was included in the model, and a "0" means that the predictor was not included in the model. Hence the first row shows the measures of predictive accuracy for a model including all four predictors.

The results have been sorted according to the AICc and therefore the best models are given at the top of the table, and the worst at the bottom of the table.

Table 5.1: All 16 possible models for forecasting US consumption with 4 predictors.

Income	Production	Savings	Unemployment	CV	AIC	AICc	BIC	AdjR2
1	1	1	1	0.116	-409	-409	-390	0.749
1	0	1	1	0.116	-408	-408	-392	0.746
1	1	1	0	0.118	-407	-407	-391	0.745
1	0	1	0	0.129	-389	-389	-376	0.716
1	1	0	1	0.278	-243	-243	-227	0.386
1	0	0	1	0.283	-238	-238	-225	0.365
1	1	0	0	0.289	-236	-236	-223	0.359
0	1	1	1	0.293	-234	-234	-218	0.356
0	1	1	0	0.300	-229	-229	-216	0.334
0	1	0	1	0.303	-226	-226	-213	0.324
0	0	1	1	0.306	-225	-224	-212	0.318

Income	Production	Savings	Unemployment	CV	AIC	AICc	BIC	AdjR2
0	1	0	0	0.314	-220	-219	-210	0.296
0	0	0	1	0.314	-218	-218	-208	0.288
1	0	0	0	0.372	-185	-185	-176	0.154
0	0	1	0	0.414	-164	-164	-154	0.052
0	0	0	0	0.432	-155	-155	-149	0.000

The best model contains all four predictors. However, a closer look at the results reveals some interesting features. There is clear separation between the models in the first four rows and the ones below. This indicates that Income and Savings are both more important variables than Production and Unemployment. Also, the first two rows have almost identical values of CV, AIC and AICc. So we could possibly drop the Production variable and get very similar forecasts. Note that Production and Unemployment are highly (negatively) correlated, as shown in Figure 5.5, so most of the predictive information in Production is also contained in the Unemployment variable.

Best subset regression

Where possible, all potential regression models should be fitted (as was done in the above example) and the best model should be selected based on one of the measures discussed. This is known as “best subsets” regression or “all possible subsets” regression.

It is recommended that one of CV, AIC or AICc be used for this purpose. If the value of T is large enough, they will all lead to the same model.

While $\bar{R}^2$ is very widely used, and has been around longer than the other measures, its tendency to select too many predictor variables makes it less suitable for forecasting than either CV, AIC or AICc. Also, the tendency of BIC to select too few variables makes it less suitable for forecasting than either CV, AIC or AICc.

Stepwise regression

If there are a large number of predictors, it is not possible to fit all possible models. For example, 40 predictors leads to $2^{40} > 1$ trillion possible models! Consequently, a strategy is required to limit the number of models to be explored.

An approach that works quite well is *backwards stepwise regression*:

- Start with the model containing all potential predictors.
- Remove one predictor at a time. Keep the model if it improves the measure of predictive accuracy.
- Iterate until no further improvement.

If the number of potential predictors is too large, then the backwards stepwise regression will not work and *forward stepwise regression* can be used instead. This procedure starts with a model that includes only the intercept. Predictors are added one at a time, and the one that most improves the measure of predictive accuracy is retained in the model. The procedure is repeated until no further improvement can be achieved.

Alternatively for either the backward or forward direction, a starting model can be one that includes a subset of potential predictors. In this case, an extra step needs to be included. For the backwards procedure we should also consider adding a predictor with each step, and for the

forward procedure we should also consider dropping a predictor with each step. These are referred to as *hybrid* procedures.

It is important to realise that any stepwise approach is not guaranteed to lead to the best possible model, but it almost always leads to a good model. For further details see James et al. (2014).

Beware of inference after selecting predictors

We do not discuss statistical inference of the predictors in this book (e.g., looking at p-values associated with each predictor). If you do wish to look at the statistical significance of the predictors, beware that *any* procedure involving selecting predictors first will invalidate the assumptions behind the p-values. The procedures we recommend for selecting predictors are helpful when the model is used for forecasting; they are not helpful if you wish to study the effect of any predictor on the forecast variable.

5.6 Forecasting with regression

Recall that predictions of y can be obtained using equation (5.2), $\hat{y}_t=\hat{\beta}_0+\hat{\beta}_1x_{1,t}+\hat{\beta}_2x_{2,t}+\cdots+\hat{\beta}_kx_{k,t}$, which comprises the estimated coefficients and ignores the error in the regression equation. Plugging in the values of the predictor variables $x_{1,t},\ldots,x_{k,t}$ for $t=1,\ldots,T$ returned the fitted (training-sample) values of y. What we are interested in here is forecasting *future* values of y.

Ex-ante versus ex-post forecasts

When using regression models for time series data, we need to distinguish between the different types of forecasts that can be produced, depending on what is assumed to be known when the forecasts are computed.

Ex ante forecasts are those that are made using only the information that is available in advance. For example, ex-ante forecasts for the percentage change in US consumption for quarters following the end of the sample, should only use information that was available *up to and including* 2016 Q3. These are genuine forecasts, made in advance using whatever information is available at the time. Therefore in order to generate ex-ante forecasts, the model requires future values (forecasts) of the predictors. To obtain these we can use one of the simple methods introduced in Section 3.1 or more sophisticated pure time series approaches that follow in Chapters 7 and 8. Alternatively, forecasts from some other source, such as a government agency, may be available and can be used.

Ex post forecasts are those that are made using later information on the predictors. For example, ex post forecasts of consumption may use the actual observations of the predictors, once these have been observed. These are not genuine forecasts, but are useful for studying the behaviour of forecasting models.

The model from which ex-post forecasts are produced should not be estimated using data from the forecast period. That is, ex-post forecasts can assume knowledge of the predictor variables (the x variables), but should not assume knowledge of the data that are to be forecast (the y variable).

A comparative evaluation of ex-ante forecasts and ex-post forecasts can help to separate out the sources of forecast uncertainty. This will show whether forecast errors have arisen due to poor forecasts of the predictor or due to a poor forecasting model.

Example: Australian quarterly beer production

Normally, we cannot use actual future values of the predictor variables when producing ex-ante forecasts because their values will not be known in advance. However, the special predictors introduced in Section 5.3 are all known in advance, as they are based on calendar variables (e.g., seasonal dummy variables or public holiday indicators) or deterministic functions of time (e.g. time trend). In such cases, there is no difference between ex ante and ex post forecasts.

```
beer2 <- window(ausbeer, start=1992)
fit.beer <- tslm(beer2 ~ trend + season)
fcast <- forecast(fit.beer)
autoplot(fcast) +
  ggtitle("Forecasts of beer production using linear regression")
```

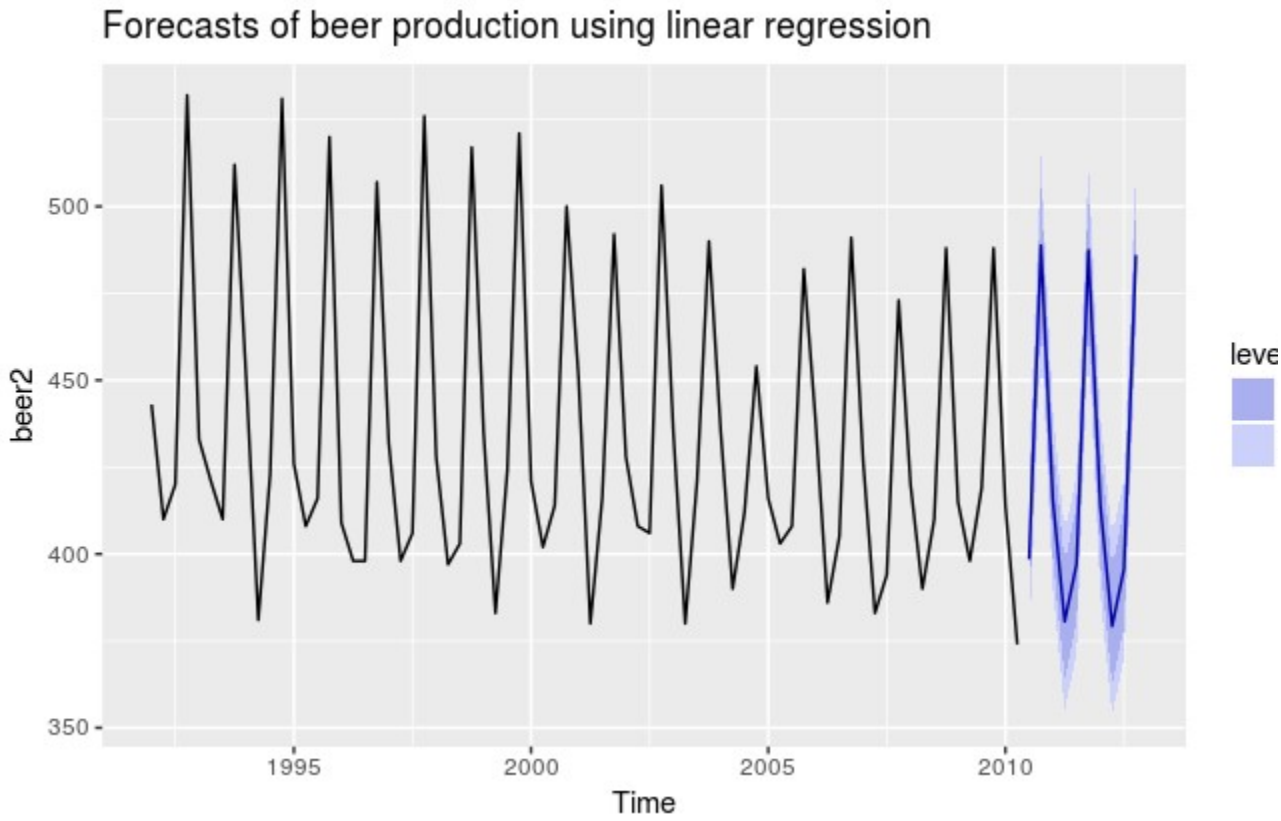


Figure 5.17: Forecasts from the regression model for beer production. The dark shaded region shows 80% prediction intervals and the light shaded region shows 95% prediction intervals.

Scenario based forecasting

In this setting, the forecaster assumes possible scenarios for the predictor variables that are of interest. For example, a US policy maker may be interested in comparing the predicted change in consumption when there is a constant growth of 1% and 0.5% respectively for income and savings with no change in the employment rate, versus a respective decline of 1% and 0.5%, for each of the four quarters following the end of the sample. The resulting forecasts are calculated below and shown in Figure 5.18. We should note that prediction intervals for scenario based forecasts do not include the uncertainty associated with the future values of the predictor variables. They assume that the values of the predictors are known in advance.

```
fit.consBest <- tslm(Consumption ~ Income + Savings + Unemployment, data = uschange)
h <- 4
newdata <-
```

```
  cbind(Income=c(1,1,1,1), Savings=c(0.5,0.5,0.5,0.5), Unemployment=c(0,0,0,0)) %>%
  as.data.frame
fcast.up <- forecast(fit.consBest, newdata=newdata)
newdata <-
  cbind(Income=rep(-1,h), Savings=rep(-0.5,h), Unemployment=rep(0,h)) %>%
  as.data.frame
fcast.down <- forecast(fit.consBest, newdata=newdata)

autoplot(uschange[,1]) + ylab("% change in US consumption") +
  forecast::autolayer(fcast.up, PI=TRUE, series="increase") +
  forecast::autolayer(fcast.down, PI=TRUE, series="decrease") +
  guides(colour=guide_legend(title="Scenario"))
```

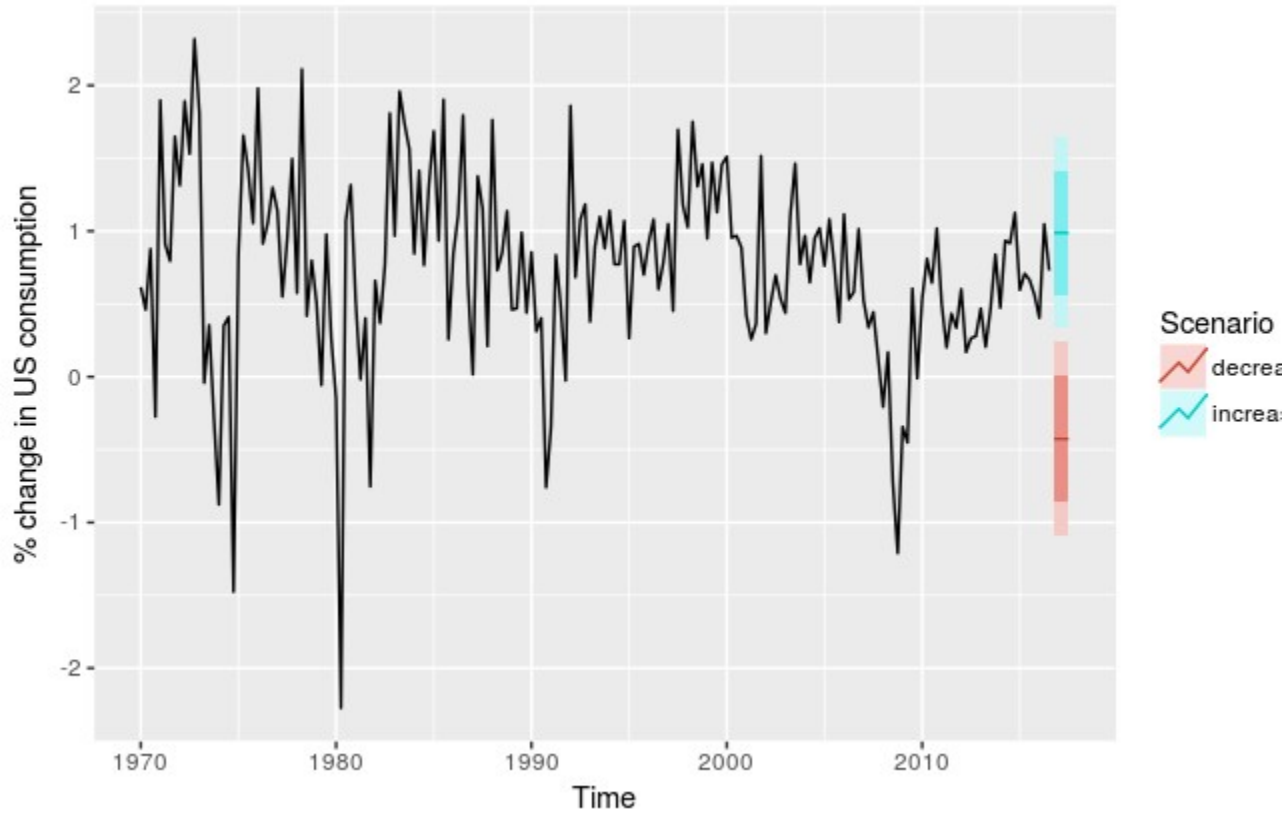


Figure 5.18: Forecasting percentage changes in personal consumption expenditure for the US under scenario based forecasting.

Building a predictive regression model

The great advantage of regression models is that they can be used to capture important relationships between the forecast variable of interest and the predictor variables. A major challenge however, is that in order to generate ex-ante forecasts the model requires future values of each predictor. If scenario based forecasting is of interest then these models are extremely useful. However, if ex-ante forecasting is the main focus, obtaining forecasts of the predictors can be very challenging (in many cases generating forecasts for the predictor variables can be more challenging than forecasting directly the forecast variable without using predictors).

An alternative formulation is to use as predictors their lagged values. Assuming that we are interested in generating a h-step ahead forecast we write $y_{t+h}=\beta_0+\beta_1 x_{1,t}+\cdots+\beta_k x_{k,t}+\varepsilon_{t+h}$ for $h=1,2\ldots$. The predictor set is formed by values of the xs that are observed h time periods prior to observing y. Therefore when the estimated model is projected into the future, i.e., beyond the end of the sample T, all predictor values are available.

Including lagged values of the predictors does not only make the model operational for easily generating forecasts, it also makes it intuitively appealling. For example, the effect of a policy change with the aim of increasing production may not have an instantaneous effect on consumption expenditure. It is most likely that this will happend with a lagging effect. We touched upon this in Section 5.3 when briefly introducing distributed lags as predictors. Several directions for generalising regression models to better incorporate the rich dynamics observed in time series are discussed In Section 9.

Prediction intervals

With each forecast for the change in consumption Figure 5.18 95% and 80% prediction intervals are also included. The general formulation of how to calculate prediction intervals for multiple regression models is presented in Section 5.7. As this involves some advanced matrix algebra we present here the case for calculating prediction intervals for a simple regression, where a forecast can be generated using the equation, ^y=^β0+^β1x. Assuming that the regression errors are normally distributed, an approximate 95% prediction interval (also called a forecast interval) associated with this forecast is given by ^y±1.96^σe√1+1T+(x−¯x)2(T−1)s2x,

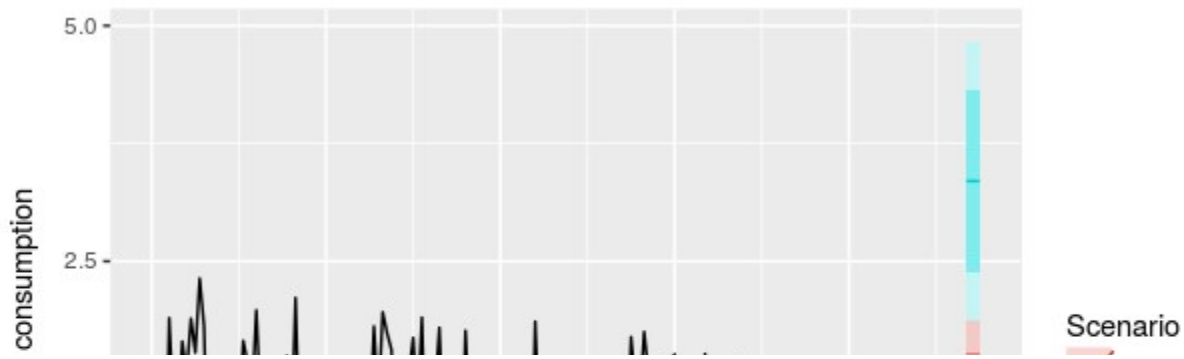
where T is the total number of observations, ¯x is the mean of the observed x values, sx is the standard deviation of the observed x values and ^σe is the standard error of the regression given by Equation (5.3). Similarly, an 80% forecast interval can be obtained by replacing 1.96 by 1.28. Other prediction intervals can be obtained by replacing the 1.96 with the appropriate value given in Table 3.1. If R is used to obtain forecast intervals, more exact calculations are obtained (especially for small values of T) than what is given by Equation (5.4).

Equation (5.4) shows that the forecast interval is wider when x is far from ¯x. That is, we are more certain about our forecasts when considering values of the predictor variable close to its sample mean. #### Example {-} `TODO: This is to higligh the extreme case. If not useful just remove.`

The estimated simple regression line in the US consumption example is ^yt=0.55+0.28xt.

Assuming that for the next four quarters personal income will increase by its historical mean value of ¯x=0.72%, consumption is forecast to increase by 0.75% and the corresponding 95% and 80% prediction intervals are [−0.45,1.94] and [−0.03,1.52] respectively (calcualted using R). If we assume an extreme increase of 10% in income then the prediction intervals are considerably wider as shown in Figure **??**.

```
autoplot(uschange[,"Consumption"]) +
  ylab("% change in US consumption") +
  forecast::autolayer(fcast.up, PI=TRUE, series="Average increase") +
  forecast::autolayer(fcast.down, PI=TRUE, series="Extreme increase") +
  guides(colour=guide_legend(title="Scenario"))
```



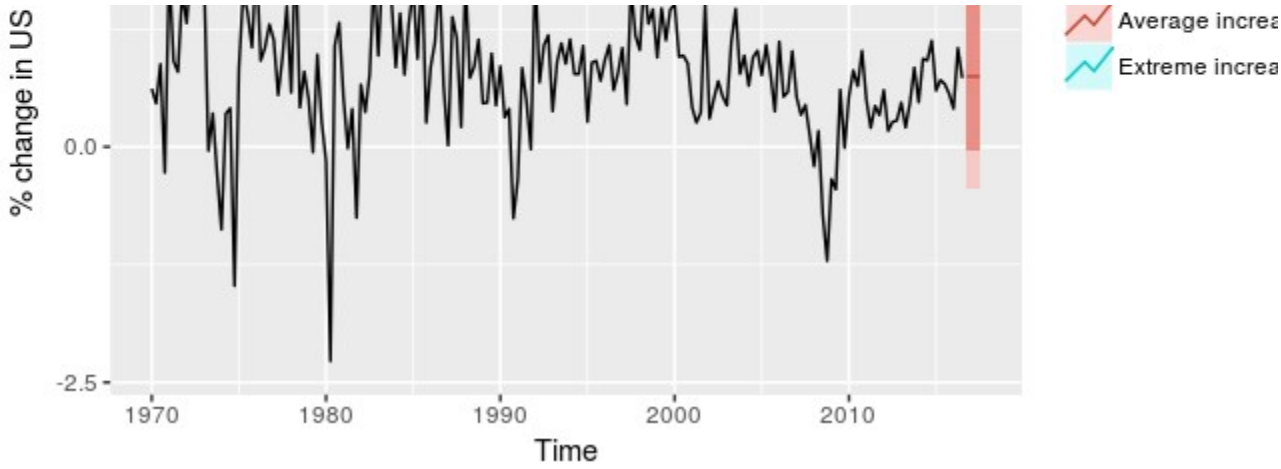


Figure 5.19: Prediction intervals if inomce increased by its historical mean versus an extreme increase of 10%.

5.7 Matrix formulation

Warning: this is a more advanced optional section and assumes knowledge of matrix algebra.

where ε has mean 0 and variance σ2I. Note that the X matrix has T rows reflecting the number of observations and k+1 columns reflecting the intercept which is represented by the column of ones plus the number of predictors.

Forecasts and prediction intervals

Let be a row vector containing the values of the predictors (in the same format as X) for which we want to generate a forecast . Then the forecast is given by ˆy=x∗ˆβ=x∗(X′X)−1X′Y and its variance is given by σ2[1+x∗(X′X)−1(x∗)′]. A 95% prediction interval can be calculated (assuming normally distributed errors) as ˆy±1.96ˆσe√1+x∗(X′X)−1(x∗)′. This takes account of the uncertainty due to the error term ε and the uncertainty in the coefficient estimates. However, it ignores any errors in . So if the future values of the predictors are uncertain, then the prediction interval calculated using this expression will be too narrow.

5.8 Nonlinear regression

Although the linear relationship assumed so far in this chapter is often adequate, there are many cases for which a nonlinear functional form is more suitable. To keep things simple in this section we assume that we only have one predictor x.

Simply transforming the forecast variable y and/or the predictor variable x and then estimating a regression model using the transformed variables is the simplest way of obtaining a nonlinear specification. We should note that these specifications and the specification that follow are still linear in the parameters. The most commonly used transformation is the (natural) logarithmic (see Section 3.2).

A **log-log** functional form is specified as logy=β0+β1logx+ε. In this model, the slope β1 is interpreted as an elasticity, β1 is the average percentage change in y resulting from a 1% increase in x. Other useful forms can also be specified. The **log-linear** form is specified by only transforming the forecast variable and the **linear-log** form is obtained by transforming the predictor.

Recall that in order to perform a logarithmic transformation to a variable, all its observed values must be greater than zero. In the case that variable x contains zeros we use the transformation log (x+1), i.e., we add one to the value of the variable and then take logarithms. This has a similar effect to taking logarithms but avoids the problem of zeros. It also has the neat side-effect of zeros on the original scale remaining zeros on the transformed scale.

There are cases for which simply transforming the data will not be adequate and a more general specification may be required. Then the model we use is $y=f(x)+\varepsilon$ where f is a nonlinear function. In standard (linear) regression, $f(x)=\beta 0+\beta 1x$. In the specification of nonlinear regression that follows, we allow f to be a more flexible nonlinear function of x, compared to simply a logarithmic or other transformation.

The notation $(x-c)+$ means the value $x-c$, if it is positive and 0 otherwise. This forces the slope to bend at point c. Additional bends can be included in the relationship by adding further variables of the above form.

An example of this follows by considering x=t and fitting a piecewise linear trend to a times series.

Piecewise linear relationships constructed in this way are a special case of **regression splines**. In general, a linear regression spline is obtained using $x1=x$ $x2=(x-c1)+\ldots$ $xk=(x-ck-1)+$ where c1, …,ck−1 are the knots (the points at which the line can bend). Selecting the number of knots (k−1) and where they should be positioned can be difficult and somewhat arbitrary. Some automatic knot selection algorithms are available in some software, but are not yet widely used.

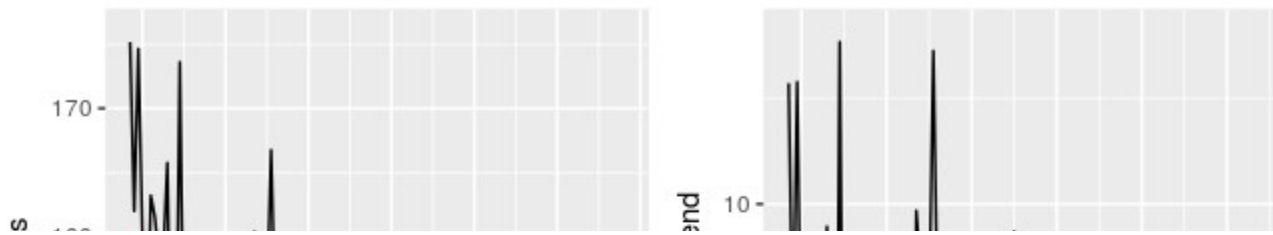
A smoother result is obtained using piecewise cubics rather than piecewise lines. These are constrained so they are continuous (they join up) and they are smooth (so there are no sudden changes of direction as we see with piecewise linear splines). In general, a cubic regression spline is written as $x1=x$ $x2=x2$ $x3=x3$ $x4=(x-c1)+\ldots$ $xk=(x-ck-3)+$. Cubic splines usually give a better fit to the data. However, forecasting values of y when x is outside the range of the historical data becomes very unreliable.

Forecasting with a nonlinear trend

In Section 5.3 fitting a linear trend to a time series by setting x=t was introduced. The simplest way of fitting a nonlinear trend is using quadratic or higher order trends obtained by specifying x1,t=t,x2,t=t2,…. However, it is not recommended that quadratic or higher order trends are used in forecasting. When they are extrapolated, the resulting forecasts are often very unrealistic.

Example: Boston marathon winning times

Figure 5.20 below in the left panel shows the Boston marathon winning times (in minutes) since it started in 1897. The time series shows a general downward trend as the winning times have been improving over the years. The panel on the right shows the residuals from fitting a linear trend to the data. The plot shows an obvious nonlinear pattern which has not been captured by the linear trend.



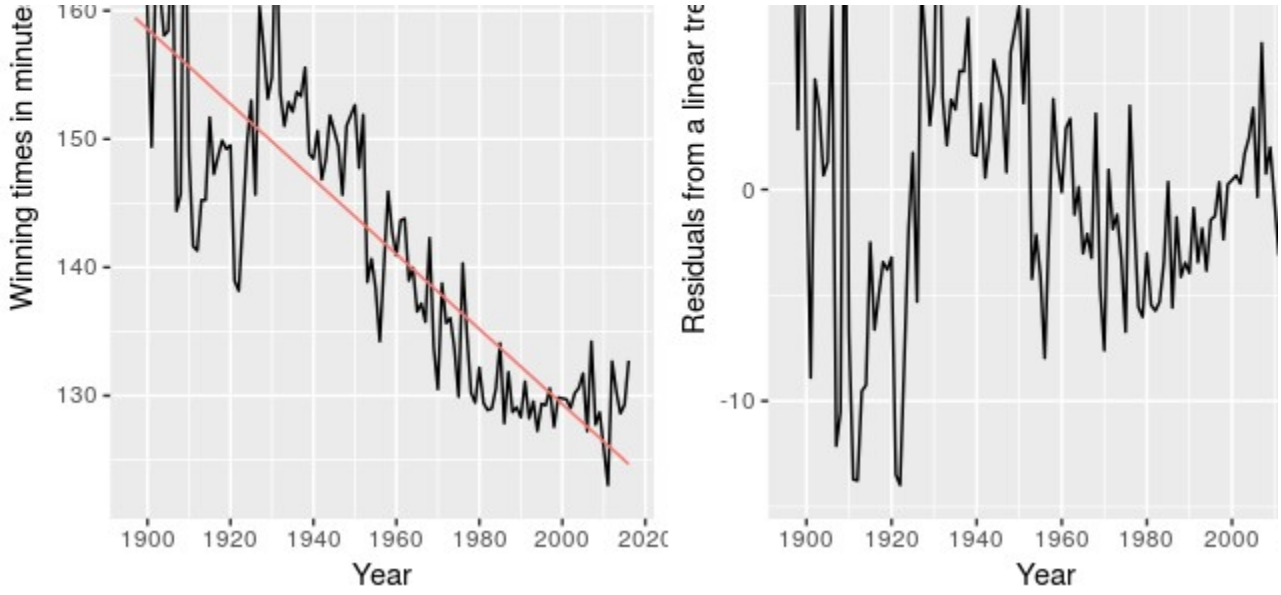


Figure 5.20: Fitting a linear trend to the Boston marathon winning times is inadequate

Fitting an exponential trend to the data can be achieved by transforming the y variable so that the model to be fitted is, $\log y_t=\beta_0+\beta_1 t+\varepsilon_t$. The fitted exponential trend and forecasts are shown in Figure 5.21 below. Although the exponential trend does not seem to fit the data much better than the linear trend, it gives a more sensible projection in that the winning times will decrease in the future but at a decaying rate rather than a linear non-changing rate.

Simply eyeballing the winning times reveals three very different periods. There is a lot of volatility in the winning times from the beginning up to about 1940 with the winning times decreasing overall but with significant increases in the decade of the 1920s. The almost linear decrease in times after 1940 is then followed by a flattening out after the 1980s with almost an upturn towards the end of the sample. Following these observations we specify as knots the years 1940 and 1980. We should warn here that subjectively identifying suitable knots in the data can lead to overfitting which can be detrimental to the forecast performance of a model and should be performed with caution.

```
h <- 10

fit.lin <- tslm(marathon ~ trend)
fcasts.lin <- forecast(fit.lin, h=h)

fit.exp <- tslm(marathon ~ trend, lambda = 0)
fcasts.exp <- forecast(fit.exp, h=h)

t.break1 <- 1940
t.break2 <- 1980

t <- time(marathon)
t1 <- ts(pmax(0,t-t.break1), start=1897)
t2 <- ts(pmax(0,t-t.break2), start=1897)

fit.pw <- tslm(marathon ~ t + t1 + t2)
  t.new <- t[length(t)]+seq(h)
  t1.new <- t1[length(t1)]+seq(h)
  t2.new <- t2[length(t2)]+seq(h)

newdata <- cbind(t=t.new,t1=t1.new,t2=t2.new)%>%as.data.frame
fcasts.pw <- forecast(fit.pw,newdata = newdata)
```

```
autoplot(marathon) +
  forecast::autolayer(fitted(fit.lin), series = "Linear") +
  forecast::autolayer(fitted(fit.exp), series="Exponential") +
  forecast::autolayer(fitted(fit.pw), series = "Piecewise") +
  forecast::autolayer(fcasts.pw, series="Piecewise") +
  forecast::autolayer(fcasts.lin$mean, series = "Linear") +
  forecast::autolayer(fcasts.exp$mean, series="Exponential") +
  xlab("Year") +  ylab("Winning times in minutes") +
  ggtitle("Boston Marathon") +
  guides(colour=guide_legend(title=" "))
```

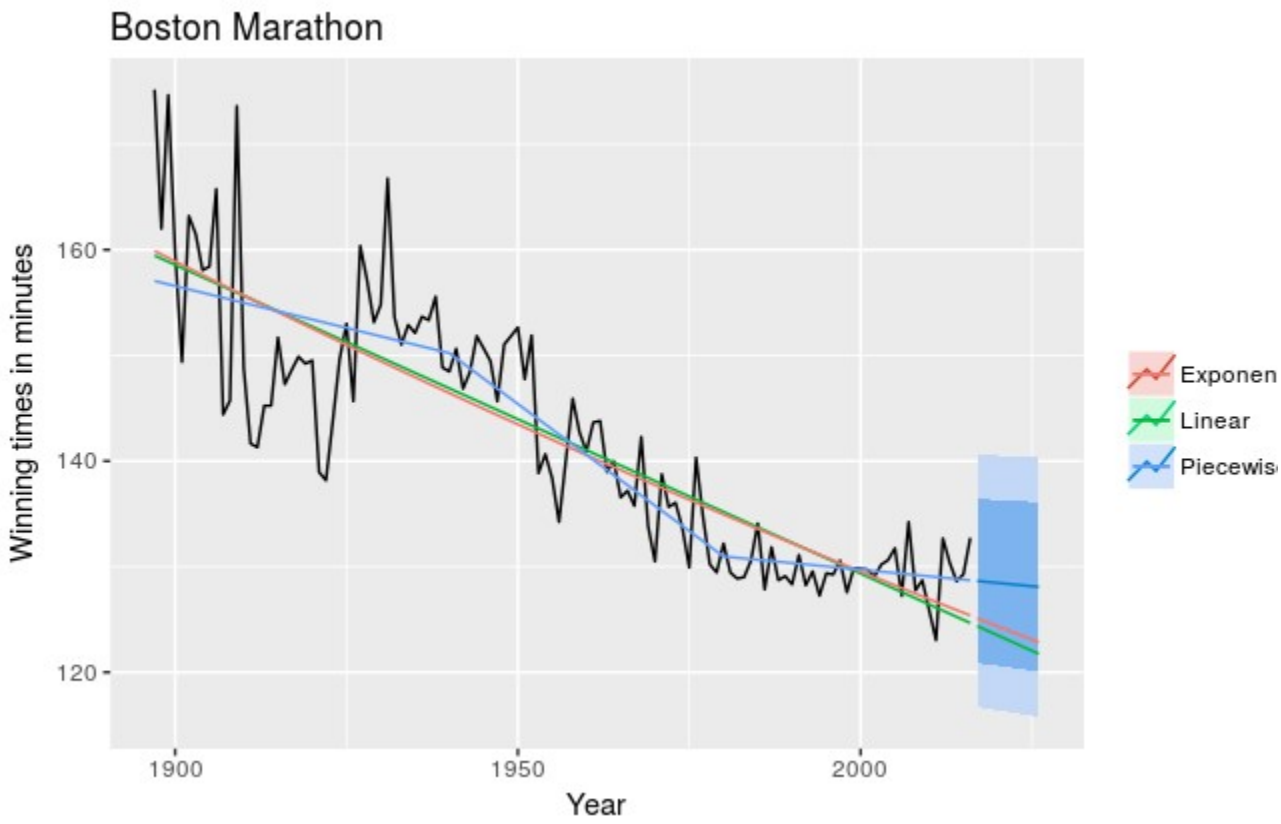


Figure 5.21: Projecting forecasts from a linear, exponential, piecewise linear trends and a cubic spline for the Boston marathon winning times

```
  ## TODO: Add cubic spline
```

Figure 5.21 above shows the fitted lines and forecasts from linear, exponential and piecewise linear trends. The piecewise linear trend fits the data better than both the exponential and linear trends and seems to return sensible forecasts. The residuals plotted in the Figure 5.22 below show that the model has mostly captured the trend well although there is some autocorrelation remaining (we will deal with this in Chapter (ch-advanced)). We should however again warn here that the projection forward from the piecewise linear trend can be very sensitive to the choice of location of the knots, especially the ones located towards the end of the sample. For example moving the knot placed at 1980 even five years forward will reveal very different forecasts. The wide prediction interval associated with the forecasts form the piecewise linear trend reflects the volatility observed in the historical winning times.

```
checkresiduals(fit.pw)
```

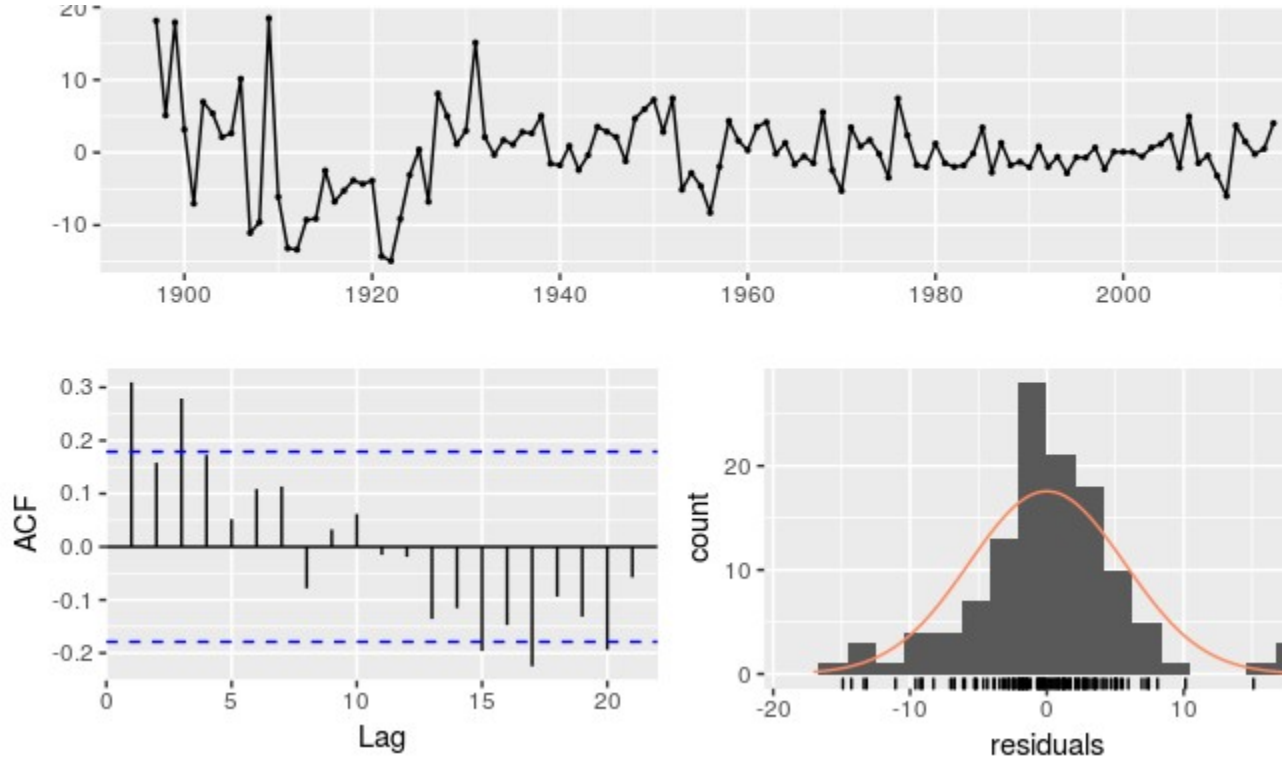


Figure 5.22: Residuals from the piecewise trend.

```
#>
#>  Breusch-Godfrey test for serial correlation of order up to 10
#>
#> data:  Residuals from Linear regression model
#> LM test = 20, df = 10, p-value = 0.02
```

5.9 Correlation, causation and forecasting

Correlation is not causation

It is important not to confuse correlation with causation, or causation with forecasting. A variable x may be useful for predicting a variable y, but that does not mean x is causing y. It is possible that x *is* causing y, but it may be that the relationship between them is more complicated than simple causality.

For example, it is possible to model the number of drownings at a beach resort each month with the number of ice-creams sold in the same period. The model can give reasonable forecasts, not because ice-creams cause drownings, but because people eat more ice-creams on hot days when they are also more likely to go swimming. So the two variables (ice-cream sales and drownings) are correlated, but one is not causing the other. It is important to understand that **correlations are useful for forecasting, even when there is no causal relationship between the two variables**.

However, often a better model is possible if a causal mechanism can be determined. In this example, both ice-cream sales and drownings will be affected by the temperature and by the numbers of people visiting the beach resort. Again, high temperatures do not actually *cause* people to drown, but they are more directly related to why people are swimming. So a better model for drownings will probably include temperatures and visitor numbers and exclude ice-cream sales.

Confounded predictors

A related issue involves confounding variables. Suppose we are forecasting monthly sales of a company for 2012, using data from 2000–2011. In January 2008 a new competitor came into the market and started taking some market share. At the same time, the economy began to decline. In your forecasting model, you include both competitor activity (measured using advertising time on a local television station) and the health of the economy (measured using GDP). It will not be possible to separate the effects of these two predictors because they are correlated. We say two variables are **confounded** when their effects on the forecast variable cannot be separated. Any pair of correlated predictors will have some level of confounding, but we would not normally describe them as confounded unless there was a relatively high level of correlation between them.

Confounding is not really a problem for forecasting, as we can still compute forecasts without needing to separate out the effects of the predictors. However, it becomes a problem with scenario forecasting as the scenarios should take account of the relationships between predictors. It is also a problem if some historical analysis of the contributions of various predictors is required.

Multicollinearity and forecasting

A closely related issue is **multicollinearity** which occurs when similar information is provided by two or more of the predictor variables in a multiple regression. It can occur in a number of ways.

Two predictors are highly correlated with each other (that is, they have a correlation coefficient close to +1 or -1). In this case, knowing the value of one of the variables tells you a lot about the value of the other variable. Hence, they are providing similar information.

A linear combination of predictors is highly correlated with another linear combination of predictors. In this case, knowing the value of the first group of predictors tells you a lot about the value of the second group of predictors. Hence, they are providing similar information.

The dummy variable trap is a special case of multicollinearity. Suppose you have quarterly data and use four dummy variables, d1,d2,d3 and d4. Then d4=1−d1−d2−d3, so there is perfect correlation between d4 and d1+d2+d3.

```
TODO: I have changed this below - change back if not improved
```

The dummy variable trap is a special case of multicollinearity, referred to as perfect multicollinearity. Suppose you have quarterly data and use four dummy variables, d1, d2, d3 and d4. Then d1+d2+d3+d4=1 and therefore there is perfect correlation between the column of ones representing the constant and the linear combination, in this case the sum, of the dummies.

In the case of perfect correlation (i.e., a correlation of +1 or -1, such as in the dummy variable trap), it is not possible to estimate the regression model. If there is high correlation (close to but not equal to +1 or -1), then the estimation of the regression coefficients is computationally difficult. In fact, some software (notably Microsoft Excel) may give highly inaccurate estimates of the coefficients. Most reputable statistical software will use algorithms to limit the effect of multicollinearity on the coefficient estimates, but you do need to be careful. The major software packages such as R, SPSS, SAS and Stata all use estimation algorithms to avoid the problem as much as possible.

The uncertainty associated with individual regression coefficients will be large. This is because they are difficult to estimate. Consequently, statistical tests (e.g., t-tests) on regression coefficients

are unreliable. (In forecasting we are rarely interested in such tests.) Also, it will not be possible to make accurate statements about the contribution of each separate predictor to the forecast.

Forecasts will be unreliable if the values of the future predictors are outside the range of the historical values of the predictors. For example, suppose you have fitted a regression model with predictors x1 and x2 which are highly correlated with each other, and suppose that the values of x1 in the fitting data ranged between 0 and 100. Then forecasts based on x1>100 or x1<0 will be unreliable. It is always a little dangerous when future values of the predictors lie much outside the historical range, but it is especially problematic when multicollinearity is present.

Note that if you are using good statistical software, if you are not interested in the specific contributions of each predictor, and if the future values of your predictor variables are within their historical ranges, there is nothing to worry about — multicollinearity is not a problem.

1. Electricity consumption was recorded for a small town on 12 consecutive days. The following maximum temperatures (degrees Celsius) and consumption (megawatt-hours) were recorded for each day. `TODO: change the econsumption to a ts of 12 concecutive days - change the lm to tslm below`

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Mwh	16.3	16.8	15.5	18.2	15.2	17.5	19.8	19.0	17.5	16.0	19.6	18.0
Temp	29.3	21.7	23.7	10.4	29.7	11.9	9.0	23.4	17.8	30.0	8.6	11.8

   1. Plot the data and find the regression model for Mwh with temperature as an explanatory variable. Why is there a negative relationship?
   2. Produce a residual plot. Is the model adequate? Are there any outliers or influential observations?
   3. Use the model to predict the electricity consumption that you would expect for the next day if the maximum temperature was 10∘ and compare it with the forecast if the with maximum temperature was 35∘. Do you believe these predictions?

   4. Give prediction intervals for your forecasts. The following R code will get you started:

```
plot(Mwh ~ temp, data=econsumption)
fit <- lm(Mwh ~ temp, data=econsumption)
plot(residuals(fit) ~ temp, data=econsumption)
forecast(fit, newdata=data.frame(temp=c(10,35)))
```

2. Data set `olympic` contains the winning times (in seconds) for the men's 400 meters final in each Olympic Games from 1896 to 2012.

   1. Plot the winning time against the year. Describe the main features of the scatterplot.
   2. Fit a regression line to the data. Obviously the winning times have been decreasing, but at what *average* rate per year?
   3. Plot the residuals against the year. What does this indicate about the suitability of the fitted line?
   4. Predict the winning time for the men's 400 meters final in the 2000, 2004, 2008 and 2012 Olympics. Give a prediction interval for each of your forecasts. What assumptions have you made in these calculations?
   5. Find out the actual winning times for these Olympics (see www.databaseolympics.com). How good were your forecasts and prediction intervals?

3. Type `easter(ausbeer)` and interpret what you see.

4. An elasticity coefficient is the ratio of the percentage change in the forecast variable (y) to the percentage change in the predictor variable (x). Mathematically, the elasticity is defined as $(dy/dx)\times(x/y)$. Consider the log-log model, $\log y=\beta 0+\beta 1\log x+\varepsilon$. Express y as a function of x and show that the coefficient $\beta 1$ is the elasticity coefficient.

5. The data set `fancy` concerns the monthly sales figures of a shop which opened in January 1987 and sells gifts, souvenirs, and novelties. The shop is situated on the wharf at a beach resort town in Queensland, Australia. The sales volume varies with the seasonal population of tourists. There is a large influx of visitors to the town at Christmas and for the local surfing festival, held every March since 1988. Over time, the shop has expanded its premises, range of products, and staff.

    1. Produce a time plot of the data and describe the patterns in the graph. Identify any unusual or unexpected fluctuations in the time series.
    2. Explain why it is necessary to take logarithms of these data before fitting a model.
    3. Use R to fit a regression model to the logarithms of these sales data with a linear trend, seasonal dummies and a "surfing festival" dummy variable.
    4. Plot the residuals against time and against the fitted values. Do these plots reveal any problems with the model?
    5. Do boxplots of the residuals for each month. Does this reveal any problems with the model?
    6. What do the values of the coefficients tell you about each variable?
    7. What does the Breusch-Godfrey test tell you about your model?
    8. Regardless of your answers to the above questions, use your regression model to predict the monthly sales for 1994, 1995, and 1996. Produce prediction intervals for each of your forecasts.
    9. Transform your predictions and intervals to obtain predictions and intervals for the raw data.
    10. How could you improve these predictions by modifying the model?

6. `TODO: you got to this before me ;-)` The `gasoline` series consists of weekly data for supplies of US finished motor gasoline product, from 2 February 1991 to 20 January 2017. The units are in "thousand barrels per day". Consider only the data to the end of 2004.
    1. Fit a harmonic regression with trend to the data. Select the appropriate number of Fourier terms to include by minimizing the AICc or CV value.
    2. Check the residuals of the final model using the `checkresiduals()` function. Even though the residuals fail the correlation tests, the results are probably not severe enough to make much difference to the forecasts and forecast intervals. (Note that the correlations are relatively small, even though they are significant.)

    3. To forecast using harmonic regression, you will need to generate the future values of the Fourier terms. This can be done as follows.

        ```
        fc <- forecast(fit, fourier(x, K, h))
        ```

        where `fit` is the fitted model using `tslm`, `K` is the number of Fourier terms used in creating `fit`, and `h` is the forecast horizon required.

        Forecast the next year of data.

    4. Plot the forecasts along with the actual data for 2005. What do you find?

5.11 Further reading

Regression with time series data

- Shumway, R. H. and D. S. Stoffer (2011). Time series analysis and its applications: with R examples. 3rd ed. New York: Springer.

Chapter 6 Time series decomposition

Time series data can exhibit a huge variety of patterns, and it is helpful to try to split a time series into several components, each representing one of the underlying categories of patterns.

In Section 2.3 we discussed three types of time series patterns: trend, seasonality and cycles. When we decompose a time series into components, we usually combine the trend and cycle into a single "trend-cycle" component (sometimes called the "trend" for simplicity). Thus we think of a time series as comprising three components: a trend-cycle component, a seasonal component, and a remainder component (containing anything else in the time series).

In this chapter, we consider some common methods for extracting these components from a time series. Often this is done to help improve understanding of the time series, but it can also be used to improve forecasts.

6.1 Time series components

If we assume an additive model, then we can write $y_t=S_t+T_t+R_t$, where y_t is the data at period t, S_t is the seasonal component at period t, T_t is the trend-cycle component at period t and R_t is the remainder component at period t. Alternatively, a multiplicative model would be written as $y_t=S_t\times T_t\times R_t$.

The additive model is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative model is more appropriate. Multiplicative models are common with economic time series.

An alternative to using a multiplicative model is to first transform the data until the variation in the series appears to be stable over time, then use an additive model. When a log transformation has been used, this is equivalent to using a multiplicative decomposition because $y_t=S_t\times T_t\times R_t$is equivalent tology$t=\log S_t+\log T_t+\log R_t$.

Electrical equipment manufacturing

We will look at several methods for obtaining the components S_t, T_t and R_t later in this chapter, but first, it is helpful to see an example. We will decompose the new orders index for electrical equipment shown in Figure 6.1. These data show the numbers of new orders for electrical equipment (computer, electronic and optical products) in the Euro area (16 countries). The data have been adjusted by working days and normalized so that a value of 100 corresponds to 2005.

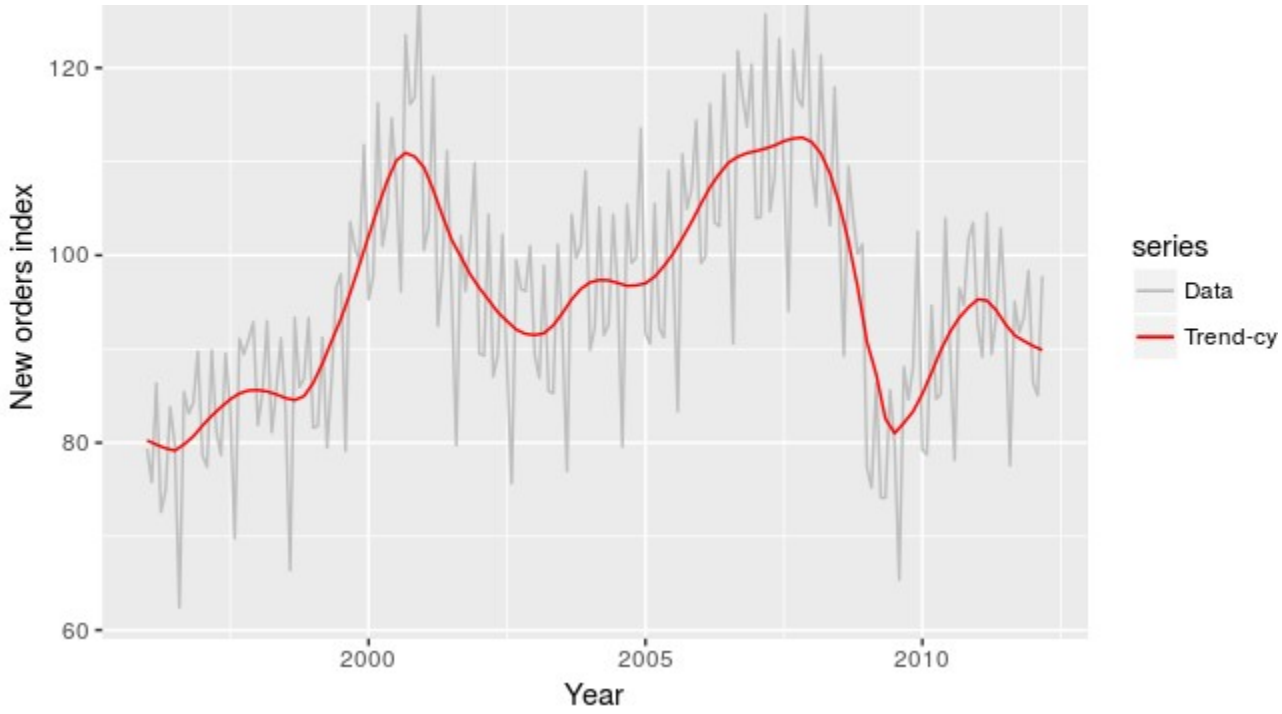
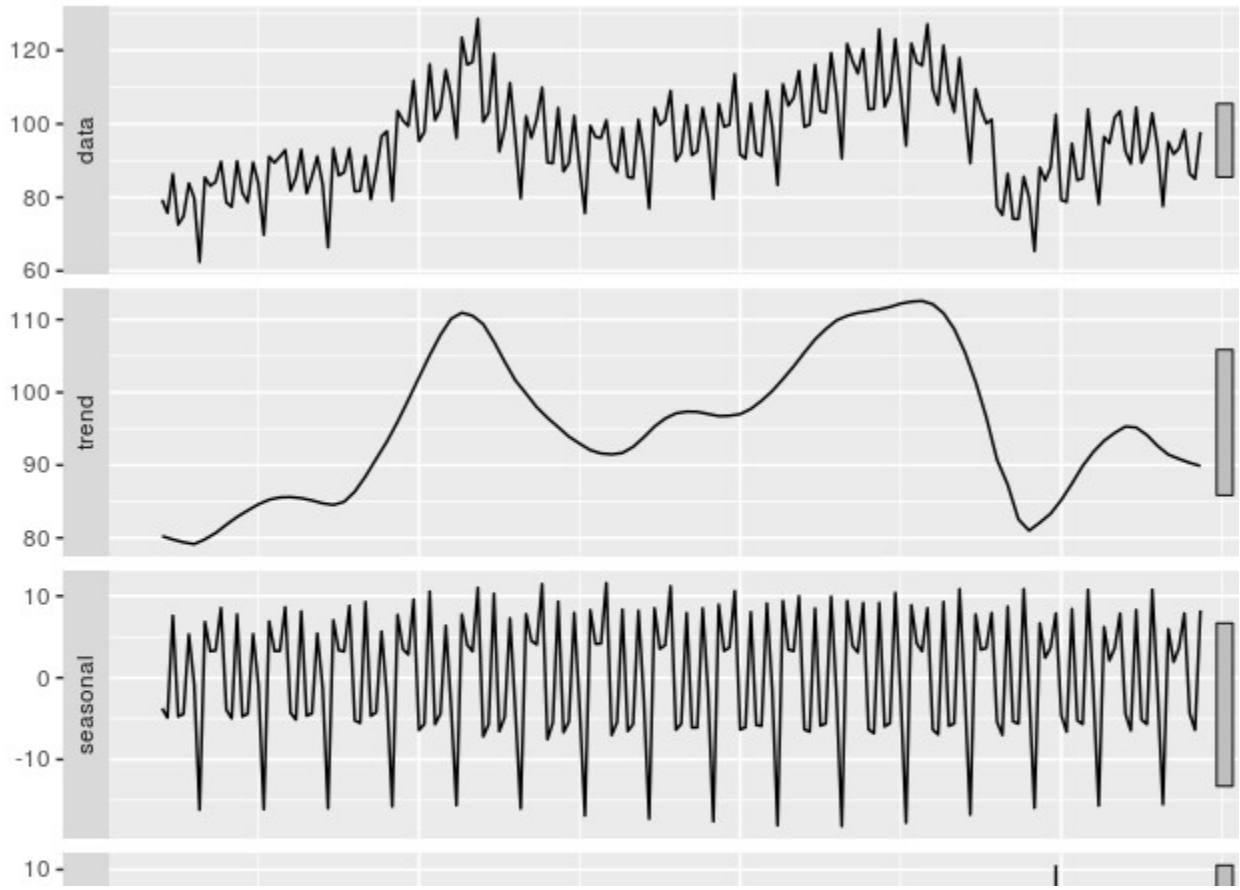


Figure 6.1: Electrical equipment orders: the trend-cycle component (red) and the raw data (grey).

Figure 6.1 shows the trend-cycle component, Tt, in red and the original data, yt, in grey. The trend-cycle shows the overall movement in the series, ignoring the seasonality and any small random fluctuations.

Figure 6.2 shows an additive decomposition of these data. The method used for estimating components in this example is STL, which is discussed in Section 6.6.



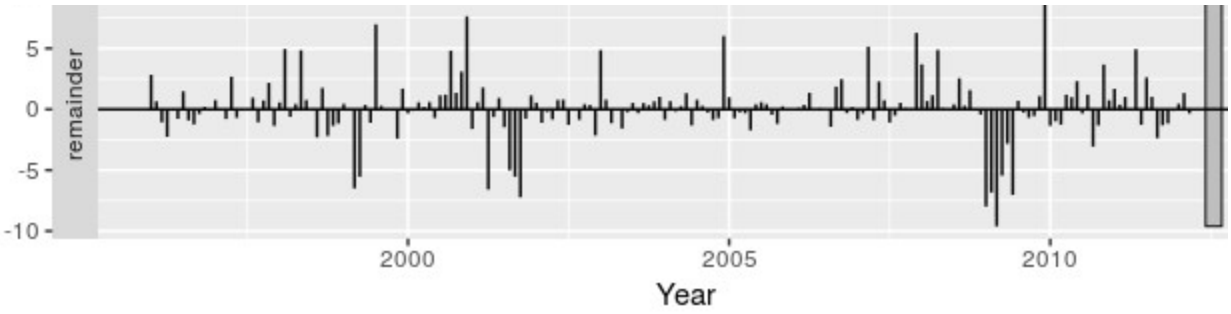


Figure 6.2: The electricial equipment orders (top) and its three additive components.

The three components are shown separately in the bottom three panels of Figure 6.2. These components can be added together to reconstruct the data shown in the top panel. Notice that the seasonal component changes very slowly over time, so that any two consecutive years have very similar patterns, but years far apart may have different seasonal patterns. The remainder component shown in the bottom panel is what is left over when the seasonal and trend-cycle components have been subtracted from the data.

The grey bars to the right of each panel show the relative scales of the components. Each grey bar represents the same length but because the plots are on different scales, the bars vary in size. The large grey bar in the bottom panel shows that the variation in the remainder component is small compared to the variation in the data, which has a bar about one quarter the size. If we shrunk the bottom three panels until their bars became the same size as that in the data panel, then all the panels would be on the same scale.

Seasonally adjusted data

If the seasonal component is removed from the original data, the resulting values are called the "seasonally adjusted" data. For an additive model, the seasonally adjusted data are given by yt−St, and for multiplicative data, the seasonally adjusted values are obtained using yt/St.

Figure 6.3 shows the seasonally adjusted electrical equipment orders.

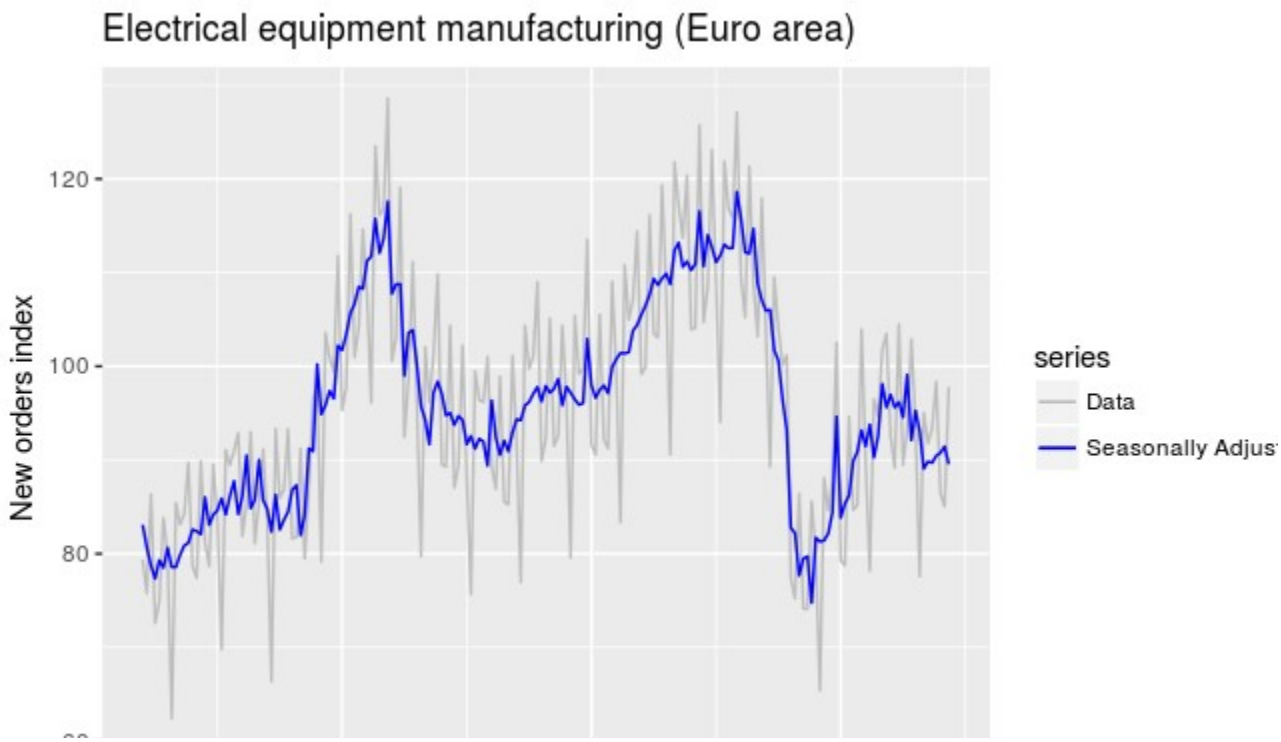


Figure 6.3: Seasonally adjusted electrical equipment orders (red) and the original data (grey).

If the variation due to seasonality is not of primary interest, the seasonally adjusted series can be useful. For example, monthly unemployment data are usually seasonally adjusted in order to highlight variation due to the underlying state of the economy rather than the seasonal variation. An increase in unemployment due to school leavers seeking work is seasonal variation, while an increase in unemployment due to large employers laying off workers is non-seasonal. Most people who study unemployment data are more interested in the non-seasonal variation. Consequently, employment data (and many other economic series) are usually seasonally adjusted.

Seasonally adjusted series contain the remainder component as well as the trend-cycle. Therefore, they are not “smooth”, and “downturns” or “upturns” can be misleading. If the purpose is to look for turning points in the series, and interpret any changes in the series, then it is better to use the trend-cycle component rather than the seasonally adjusted data.

6.2 Moving averages

The classical method of time series decomposition originated in the 1920s and was widely used until the 1950s. It still forms the basis of many time series decomposition methods, so it is important to understand how it works. The first step in a classical decomposition is to use a moving average method to estimate the trend-cycle, so we begin by discussing moving averages.

Moving average smoothing

A moving average of order m can be written as $\hat{T}_t=\frac{1}{m}\sum_{j=-k}^{k} y_{t+j}$, where $m=2k+1$. That is, the estimate of the trend-cycle at time t is obtained by averaging values of the time series within k periods of t. Observations that are nearby in time are also likely to be close in value, and the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component. We call this an “m-MA”, meaning a moving average of order m.

```
autoplot(elecsales) + xlab("Year") + ylab("GWh") +
  ggtitle("Annual electricity sales: South Australia")
```

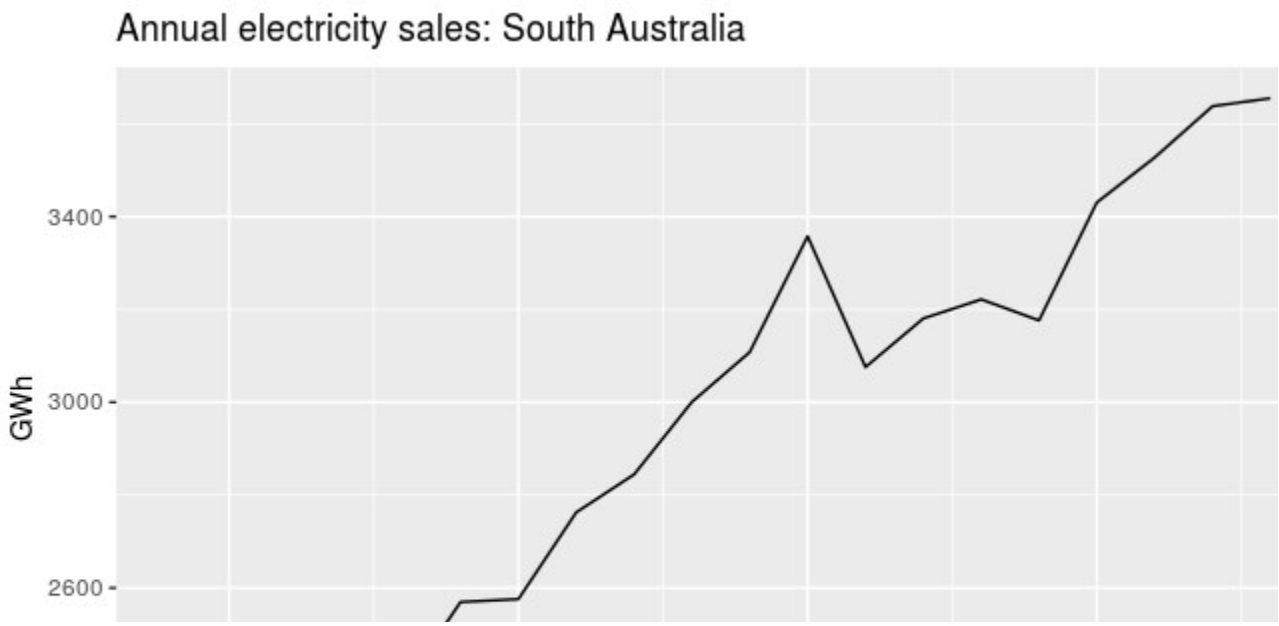


Figure 6.4: Residential electricity sales (excluding hot water) for South Australia: 1989–2008.

For example, consider Figure 6.4 which shows the volume of electricity sold to residential customers in South Australia each year from 1989 to 2008 (hot water sales have been excluded). The data are also shown in Table 6.1.

```
ma5 <- ma(elecsales, 5)
```

Table 6.1: Annual electricity sales to residential customers in South Australia. 1989–2008.

Year	Sales (GWh)	5-MA
1989	2354.34	
1990	2379.71	
1991	2318.52	2381.53
1992	2468.99	2424.56
1993	2386.09	2463.76
1994	2569.47	2552.60
1995	2575.72	2627.70
1996	2762.72	2750.62
1997	2844.50	2858.35
1998	3000.70	3014.70
1999	3108.10	3077.30
2000	3357.50	3144.52
2001	3075.70	3188.70
2002	3180.60	3202.32
2003	3221.60	3216.94
2004	3176.20	3307.30
2005	3430.60	3398.75
2006	3527.48	3485.43
2007	3637.89	
2008	3655.00	

In the second column of this table, a moving average of order 5 is shown, providing an estimate of the trend-cycle. The first value in this column is the average of the first five observations (1989 –1993); the second value in the 5-MA column is the average of the values for 1990–1994; and so on. Each value in the 5-MA column is the average of the observations in the five year period centered on the corresponding year. There are no values for either the first two years or the last two years, because we do not have two observations on either side. In the formula above, column 5-MA contains the values of ˆTt with k=2. To see what the trend-cycle estimate looks like, we plot it along with the original data in Figure 6.5.

```
autoplot(elecsales, series="Data") +
  forecast::autolayer(ma(elecsales,5), series="5-MA") +
  xlab("Year") + ylab("GWh") +
  ggtitle("Annual electricity sales: South Australia") +
  scale_colour_manual(values=c("Data"="grey50","5-MA"="red"),
                      breaks=c("Data","5-MA"))
```

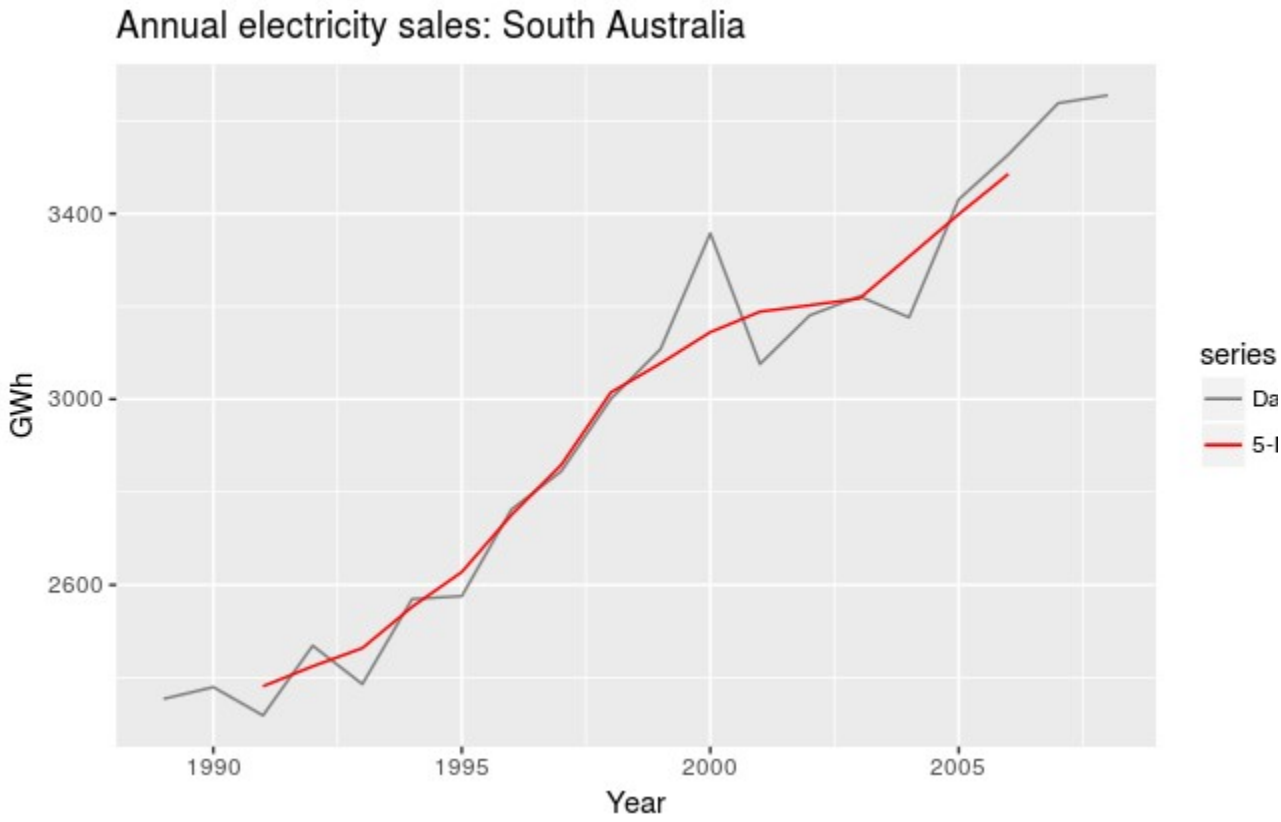
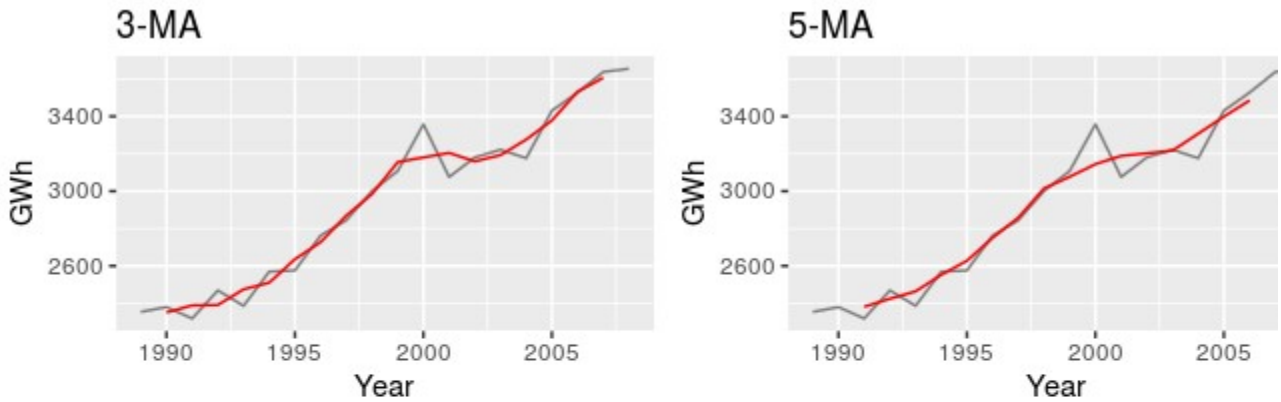


Figure 6.5: Residential electricity sales (black) along with the 5-MA estimate of the trend-cycle (red).

Notice that the trend-cycle (in red) is smoother than the original data and captures the main movement of the time series without all of the minor fluctuations. The moving average method does not allow estimates of Tt when t is close to the ends of the series; hence the red line does not extend to the edges of the graph on either side. Later we will use more sophisticated methods of trend-cycle estimation which do allow estimates near the endpoints.

The order of the moving average determines the smoothness of the trend-cycle estimate. In general, a larger order means a smoother curve. Figure 6.6 shows the effect of changing the order of the moving average for the residential electricity sales data.



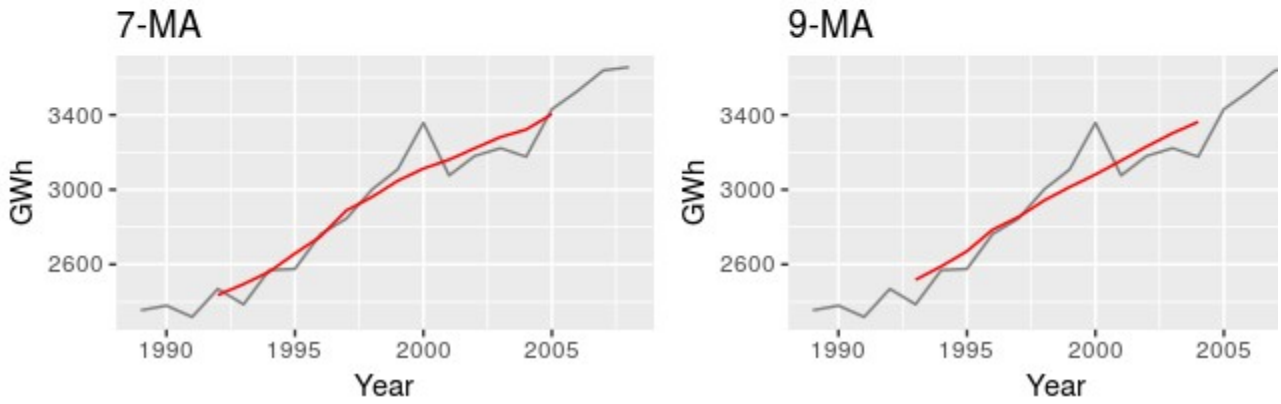


Figure 6.6: Different moving averages applied to the residential electricity sales data.

Simple moving averages such as these are usually of an odd order (e.g., 3, 5, 7, etc.) This is so they are symmetric: in a moving average of order m=2k+1, there are k earlier observations, k later observations and the middle observation that are averaged. But if m was even, it would no longer be symmetric.

Moving averages of moving averages

It is possible to apply a moving average to a moving average. One reason for doing this is to make an even-order moving average symmetric.

For example, we might take a moving average of order 4, and then apply another moving average of order 2 to the results. In the following table, this has been done for the first few years of the Australian quarterly beer production data.

```
beer2 <- window(ausbeer,start=1992)
ma4 <- ma(beer2, order=4, centre=FALSE)
ma2x4 <- ma(beer2, order=4, centre=TRUE)
```

Table 6.2: A moving average of order 4 applied to the quarterly beer data, followed by a moving average of order 2.

Year	Quarter	Observation	4-MA	2x4-MA
1992	Q1	443		
1992	Q2	410	451	
1992	Q3	420	449	450
1992	Q4	532	452	450
1993	Q1	433	449	450
1993	Q2	421	444	446
1993	Q3	410	448	446
1993	Q4	512	438	443
1994	Q1	449	441	440
1994	Q2	381	446	444
1994	Q3	423	440	443
1994	Q4	531	447	444
1995	Q1	426	445	446
1995	Q2	408	442	444

Year	Quarter	Observation	4-MA	2x4-MA
1995	Q3	416	438	440
1995	Q4	520	436	437
1996	Q1	409	431	434
1996	Q2	398	428	430
1996	Q3	398	434	431
1996	Q4	507	434	434

The notation "2×4-MA" in the last column means a 4-MA followed by a 2-MA. The values in the last column are obtained by taking a moving average of order 2 of the values in the previous column. For example, the first two values in the 4-MA column are 451.25=(443+410+420+532)/4 and 448.75=(410+420+532+433)/4. The first value in the 2x4-MA column is the average of these two: 450.00=(451.25+448.75)/2.

When a 2-MA follows a moving average of an even order (such as 4), it is called a "centered moving average of order 4". This is because the results are now symmetric. To see that this is the case, we can write the 2×4-MA as follows: $\hat{T}_t=\frac{1}{2}[\frac{1}{4}(y_{t-2}+y_{t-1}+y_t+y_{t+1})+\frac{1}{4}(y_{t-1}+y_t+y_{t+1}+y_{t+2})]=\frac{1}{8}y_{t-2}+\frac{1}{4}y_{t-1}+\frac{1}{4}y_t+\frac{1}{4}y_{t+1}+\frac{1}{8}y_{t+2}$.

It is now a weighted average of observations, but it is symmetric.

Other combinations of moving averages are also possible. For example, a 3×3-MA is often used, and consists of a moving average of order 3 followed by another moving average of order 3. In general, an even order MA should be followed by an even order MA to make it symmetric. Similarly, an odd order MA should be followed by an odd order MA.

Estimating the trend-cycle with seasonal data

The most common use of centered moving averages is in estimating the trend-cycle from seasonal data. Consider the 2×4-MA: $\hat{T}_t=\frac{1}{8}y_{t-2}+\frac{1}{4}y_{t-1}+\frac{1}{4}y_t+\frac{1}{4}y_{t+1}+\frac{1}{8}y_{t+2}$. When applied to quarterly data, each quarter of the year is given equal weight as the first and last terms apply to the same quarter in consecutive years. Consequently, the seasonal variation will be averaged out and the resulting values of $\hat{T}_t$ will have little or no seasonal variation remaining. A similar effect would be obtained using a 2×8-MA or a 2×12-MA to quarterly data.

In general, a 2×m-MA is equivalent to a weighted moving average of order m+1 where all observations take the weight 1/m, except for the first and last terms which take weights 1/(2m). So, if the seasonal period is even and of order m, use a 2×m-MA to estimate the trend-cycle. If the seasonal period is odd and of order m, use a m-MA to estimate the trend-cycle. For example, a 2×12-MA can be used to estimate the trend-cycle of monthly data and a 7-MA can be used to estimate the trend-cycle of daily data with a weekly seasonality.

Other choices for the order of the MA will usually result in trend-cycle estimates being contaminated by the seasonality in the data.

Example: Electrical equipment manufacturing

```
autoplot(elecequip, series="Data") +
  forecast::autolayer(ma(elecequip, 12), series="12-MA") +
  xlab("Year") + ylab("New orders index") +
  ggtitle("Electrical equipment manufacturing (Euro area)") +
```

```
scale_colour_manual(values=c("Data"="grey","12-MA"="red"),
  breaks=c("Data","12-MA"))
```

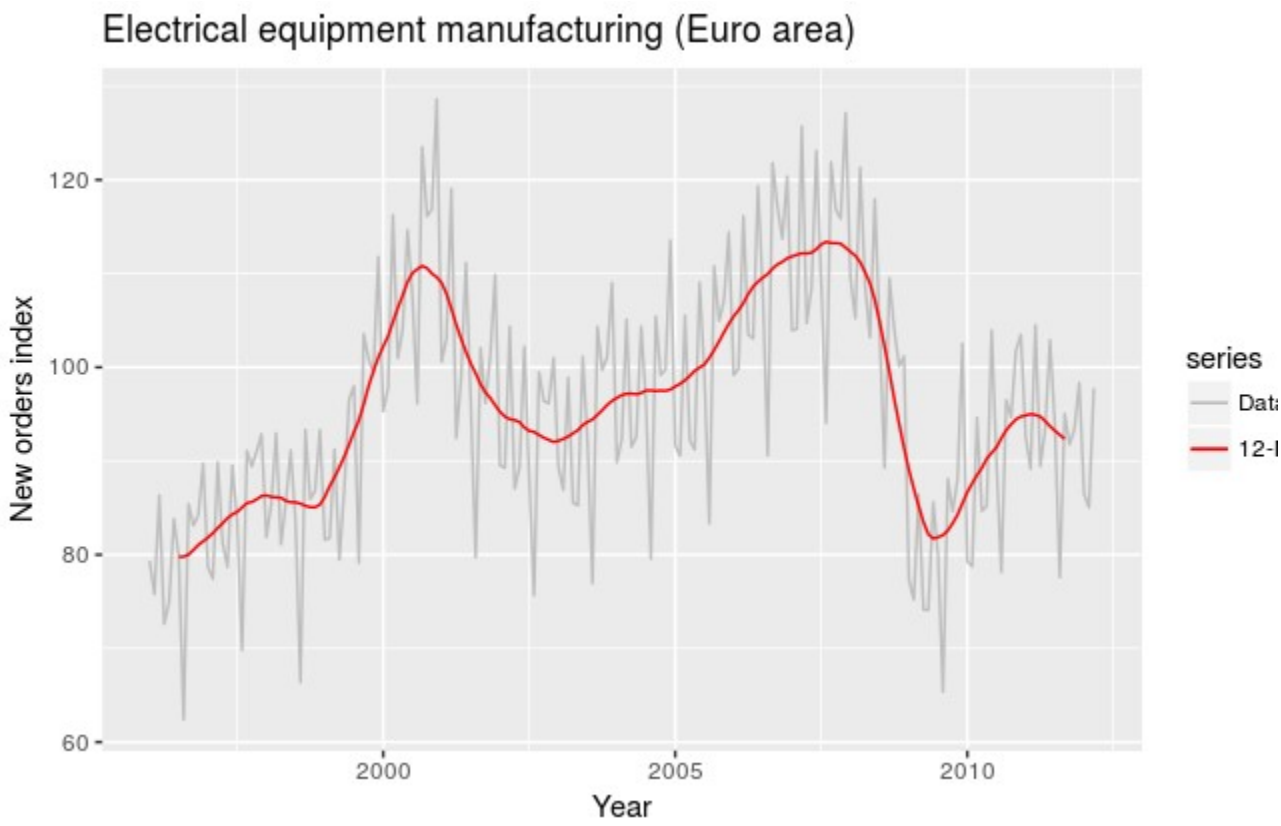


Figure 6.7: A 2x12-MA applied to the electrical equipment orders index.

Figure 6.7 shows a 2×12-MA applied to the electrical equipment orders index. Notice that the smooth line shows no seasonality; it is almost the same as the trend-cycle shown in Figure 6.1, which was estimated using a much more sophisticated method than moving averages. Any other choice for the order of the moving average (except for 24, 36, etc.) would have resulted in a smooth line that showed some seasonal fluctuations.

Weighted moving averages

Combinations of moving averages result in weighted moving averages. For example, the 2×4-MA discussed above is equivalent to a weighted 5-MA with weights given by [18,14,14,14,18]. In general, a weighted m-MA can be written as $\hat{T}t=k\sum j=-kajyt+j$, where k=(m−1)/2, and the weights are given by [a−k,…,ak]. It is important that the weights all sum to one and that they are symmetric so that aj=a−j. The simple m-MA is a special case where all of the weights are equal to 1/m.

A major advantage of weighted moving averages is that they yield a smoother estimate of the trend-cycle. Instead of observations entering and leaving the calculation at full weight, their weights slowly increase and then slowly decrease, resulting in a smoother curve.

Some specific sets of weights are widely used due to their mathematical properties. Some of these are given in Table 6.3.

Table 6.3: Commonly used weights in weighted moving averages. S=Spencer's weighted moving average. H=Henderson's weighted moving average

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11
3-MA	0.333	0.333										
5-MA	0.200	0.200	0.200									
2x12-MA	0.083	0.083	0.083	0.083	0.083	0.083	0.042					
3x3-MA	0.333	0.222	0.111									
3x5-MA	0.200	0.200	0.133	0.067								
S15-MA	0.231	0.209	0.144	0.066	0.009	-0.016	-0.019	-0.009				
S21-MA	0.171	0.163	0.134	0.094	0.051	0.017	-0.006	-0.014	-0.014	-0.009	-0.003	
H5-MA	0.558	0.294	-0.073									
H9-MA	0.330	0.267	0.119	-0.010	-0.041							
H13-MA	0.240	0.214	0.147	0.066	0.000	-0.028	-0.019					
H23-MA	0.148	0.138	0.122	0.097	0.068	0.039	0.013	-0.005	-0.015	-0.016	-0.011	-0.004

6.3 Classical decomposition

The classical decomposition method originated in the 1920s. It is a relatively simple procedure, and forms the starting point for most other methods of time series decomposition. There are two forms of classical decomposition: an additive decomposition and a multiplicative decomposition. These are described below for a time series with seasonal period m (e.g., m=4 for quarterly data, m=12 for monthly data, m=7 for daily data with a weekly pattern).

In classical decomposition, we assume that the seasonal component is constant from year to year. For multiplicative seasonality, the m values that form the seasonal component are sometimes called the “seasonal indices”.

Additive decomposition

Step 1
: If m is an even number, compute the trend-cycle component using a 2×m-MA to obtain $\hat{T}t$. If m is an odd number, compute the trend-cycle component using an m-MA to obtain $\hat{T}t$.

Step 2
: Calculate the detrended series: $yt-\hat{T}t$.

Step 3
: To estimate the seasonal component for each season, simply average the detrended values for that season. For example, with monthly data, the seasonal component for March is the average of all the detrended March values in the data. These seasonal component values are then adjusted to ensure that they add to zero. The seasonal component is obtained by stringing together these values for each year of data. This gives $\hat{S}t$.

Step 4
: The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $\hat{R}t=yt-\hat{T}t-\hat{S}t$.

Multiplicative decomposition

A classical multiplicative decomposition is very similar, except that the subtractions are replaced by divisions.

Step 1

If m is an even number, compute the trend-cycle component using a 2×m-MA to obtain ˆTt.
If m is an odd number, compute the trend-cycle component using an m-MA to obtain ˆTt.

Step 2

Calculate the detrended series: yt/ˆTt.

Step 3

To estimate the seasonal component for each season, simply average the detrended values for that season. For example, with monthly data, the seasonal index for March is the average of all the detrended March values in the data. These seasonal indexes are then adjusted to ensure that they add to m. The seasonal component is obtained by stringing together all of the seasonal indices for each year of data. This gives ˆSt.

Step 4

The remainder component is calculated by dividing out the estimated seasonal and trend-cycle components: ˆRt=yt/(ˆTtˆSt).

```
fit <- decompose(elecequip, type="multiplicative")
autoplot(fit) + xlab("Year") +
  ggtitle("Classical multiplicative decomposition of electrical equipment index")
```

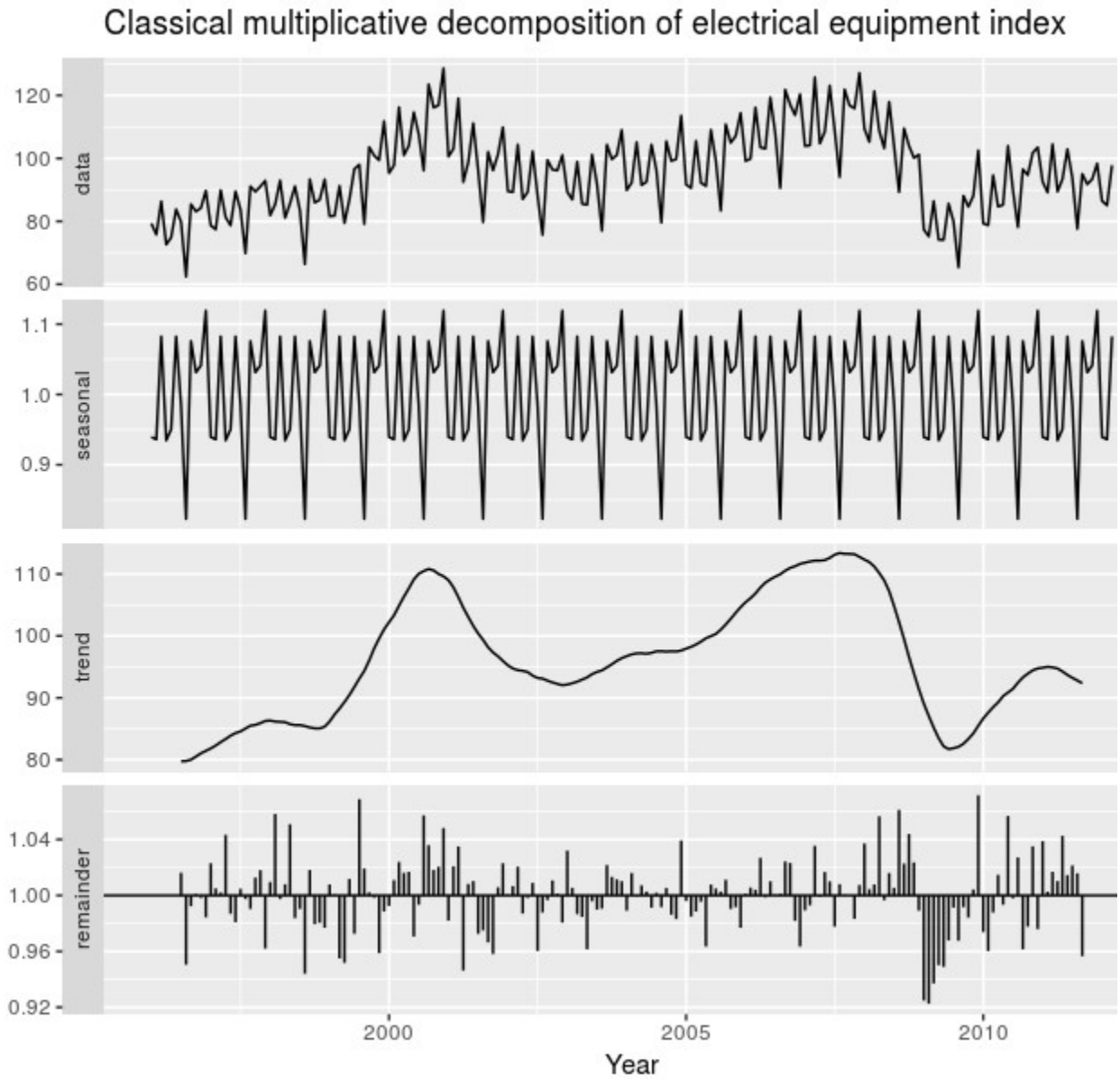


Figure 6.8: A classical multiplicative decomposition of the new orders index for electrical equipment.

Figure 6.8 shows a classical decomposition of the electrical equipment index. Compare this decomposition with that shown in Figure 6.1. The run of large negative remainder values in 2009 suggests that there is some "leakage" of the trend-cycle component into the remainder component. The trend-cycle estimate has over-smoothed the drop in the data, and the corresponding remainder values have been affected by the poor trend-cycle estimate.

Comments on classical decomposition

While classical decomposition is still widely used, it is not recommended, as there are now several much better methods. Some of the problems with classical decomposition are summarised below.

- The estimate of the trend-cycle is unavailable for the first few and last few observations. For example, if m=12, there is no trend-cycle estimate for the first six or the last six observations. Consequently, there is also no estimate of the remainder component for the same time periods.
- The trend-cycle estimate tends to over-smooth rapid rises and falls in the data (as seen in the above example).
- Classical decomposition methods assume that the seasonal component repeats from year to year. For many series, this is a reasonable assumption, but for some longer series it is not. For example, electricity demand patterns have changed over time as air conditioning has become more widespread. Specifically, in many locations, the seasonal usage pattern from several decades ago had its maximum demand in winter (due to heating), while the current seasonal pattern has its maximum demand in summer (due to air conditioning). The classical decomposition methods are unable to capture these seasonal changes over time.
- Occasionally, the values of the time series in a small number of periods may be particularly unusual. For example, the monthly air passenger traffic may be affected by an industrial dispute, making the traffic during the dispute very different from usual. The classical method is not robust to these kinds of unusual values.

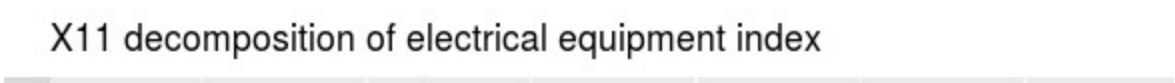
Another popular method for decomposing quarterly and monthly data is the X11 method which originated in the US Census Bureau and Statistics Canada.

This method is based on classical decomposition, but includes many extra steps and features in order to overcome the drawbacks of classical decomposition that were discussed in the previous section. In particular, trend-cycle estimates are available for all observations including the end points, and the seasonal component is allowed to vary slowly over time. X11 also has some sophisticated methods for handling trading day variation, holiday effects and the effects of known predictors. It handles both additive and multiplicative decomposition. The process is entirely automatic and tends to be highly robust to outliers and level shifts in the time series.

The details of the X11 method are described in Dagum and Bianconcini (2016). Here we will only demonstrate how to use the automatic procedure in R.

The X11 method is available using the `seas` function from the `seasonal` package for R.

```
library(seasonal)
fit <- seas(elecequip, x11="")
autoplot(fit) +
  ggtitle("X11 decomposition of electrical equipment index")
```



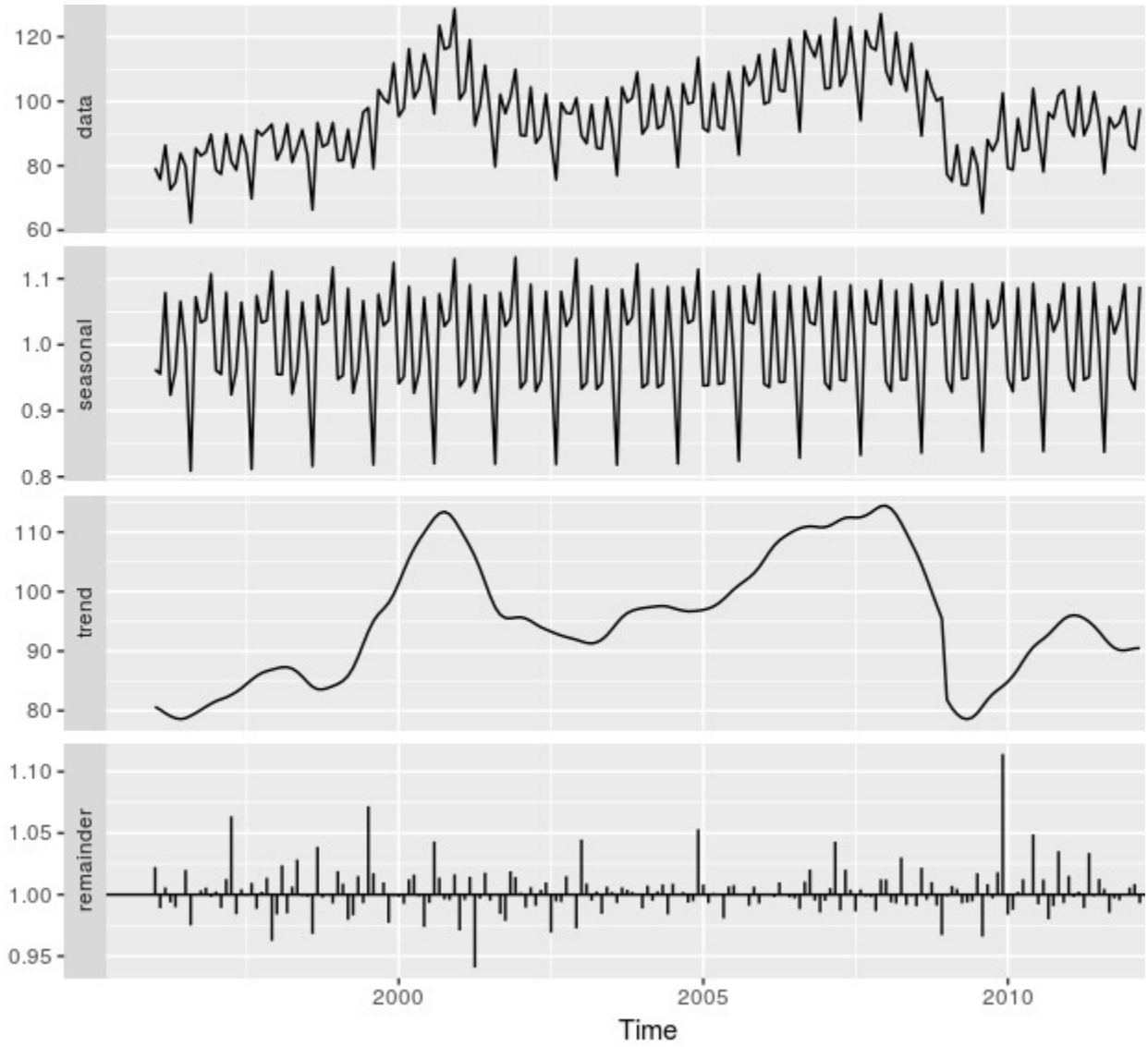


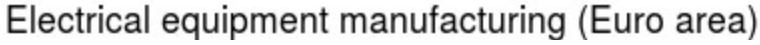
Figure 6.9: An X11 decomposition of the new orders index for electrical equipment.

Compare this decomposition with the STL decomposition shown in Figure 6.1 and the classical decomposition shown in Figure 6.8. The X11 trend-cycle has captured the sudden fall in the data in early 2009 better than either of the other two methods, and the unusual observation at the end of 2009 is now more clearly seen in the remainder component.

Given the output from the `seas` function, `seasonal()` will extract the seasonal component, `trendcycle()` will extract the trend-cycle component, `remainder()` will extract the remainder component, and `seasadj()` will compute the seasonally adjusted time series.

For example, Figure 6.10 shows the trend-cycle component and the seasonally adjusted data, along with the original data.

```
autoplot(elecequip, series="Data") +
  forecast::autolayer(trendcycle(fit), series="Trend") +
  forecast::autolayer(seasadj(fit), series="Seasonally Adjusted") +
  xlab("Year") + ylab("New orders index") +
  ggtitle("Electrical equipment manufacturing (Euro area)") +
  scale_colour_manual(values=c("gray","blue","red"),
                      breaks=c("Data","Seasonally Adjusted","Trend"))
```



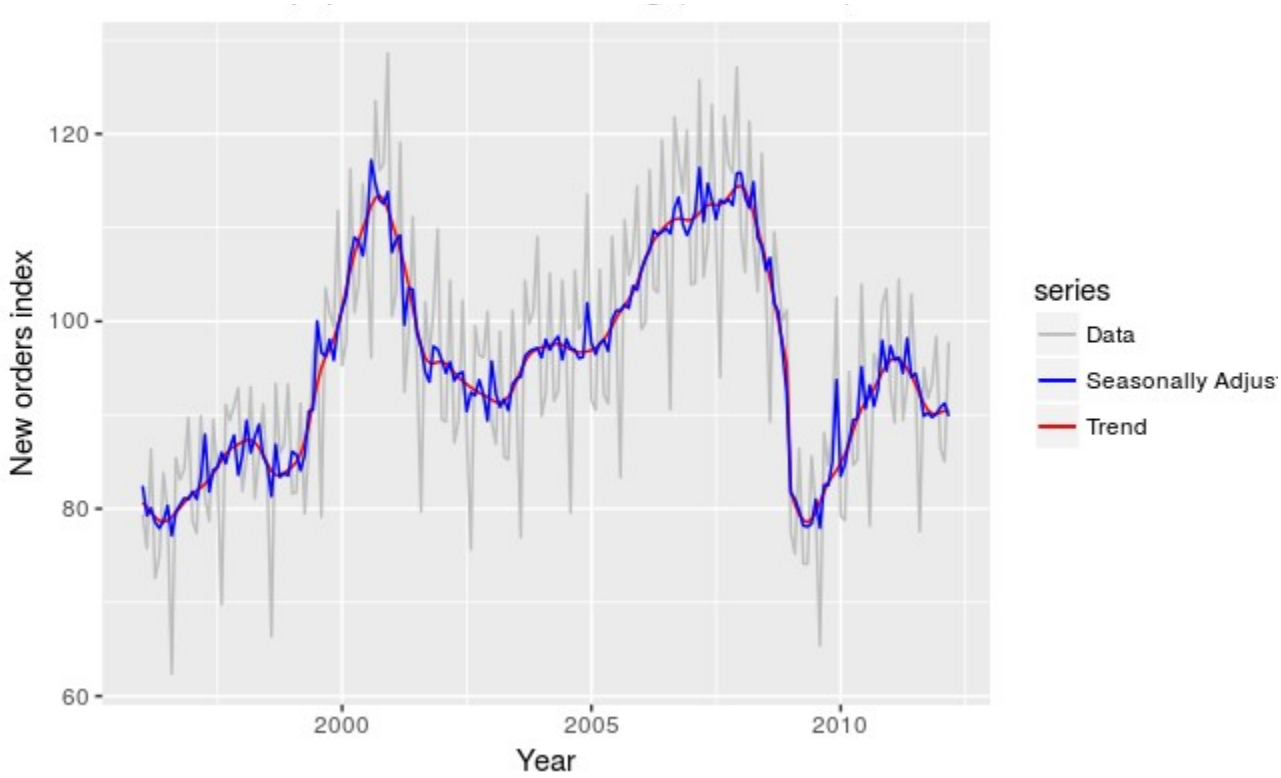


Figure 6.10: Electrical equipment orders: the original data (grey), the trend-cycle component (red) and the seasonally adjusted data (blue).

It can be useful to use seasonal plots and seasonal sub-series plots of the seasonal component. These help us to visualize the variation in the seasonal component over time. Figure 6.11 shows a seasonal sub-series plot of the seasonal component from Figure 6.9. In this case, there are only very small changes over time.

```
ggsubseriesplot(seasonal(fit)) + ylab("Seasonal")
```

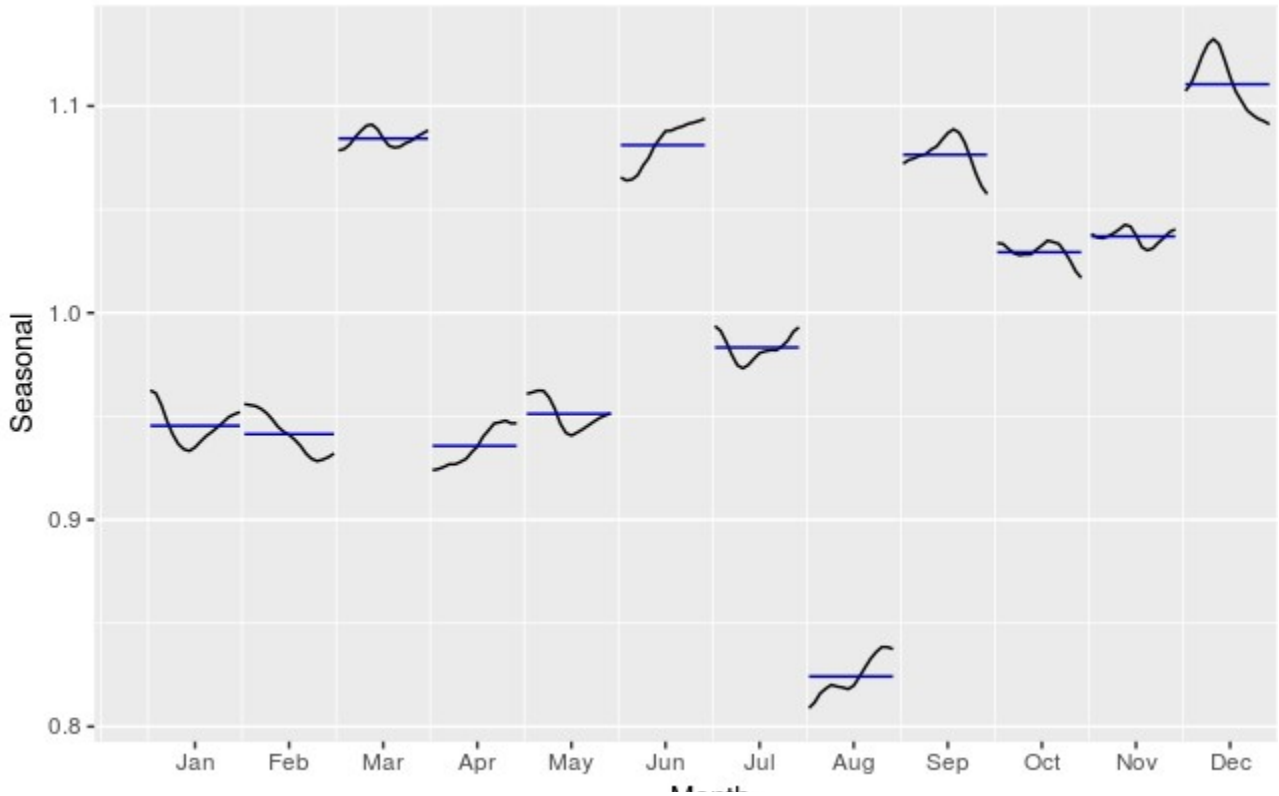


Figure 6.11: Seasonal sub-series plot of the seasonal component from the X11 decomposition shown in Figure 6.9.

6.5 SEATS decomposition

"SEATS" stands for "Seasonal Extraction in ARIMA Time Series" (ARIMA models are discussed in Chapter 8). This procedure was developed at the Bank of Spain, and is now widely used by government agencies around the world. The procedure works only with quarterly and monthly data. So seasonality of other kinds, such as daily data, or hourly data, or weekly data, require an alternative approach.

The details are beyond the scope of this book. However, a complete discussion of the method is available in Dagum and Bianconcini (2016). Here we will only demonstrate how to use it via the `seasonal` package.

```
library(seasonal)
fit <- seas(elecequip)
autoplot(fit) +
  ggtitle("SEATS decomposition of electrical equipment index")
```

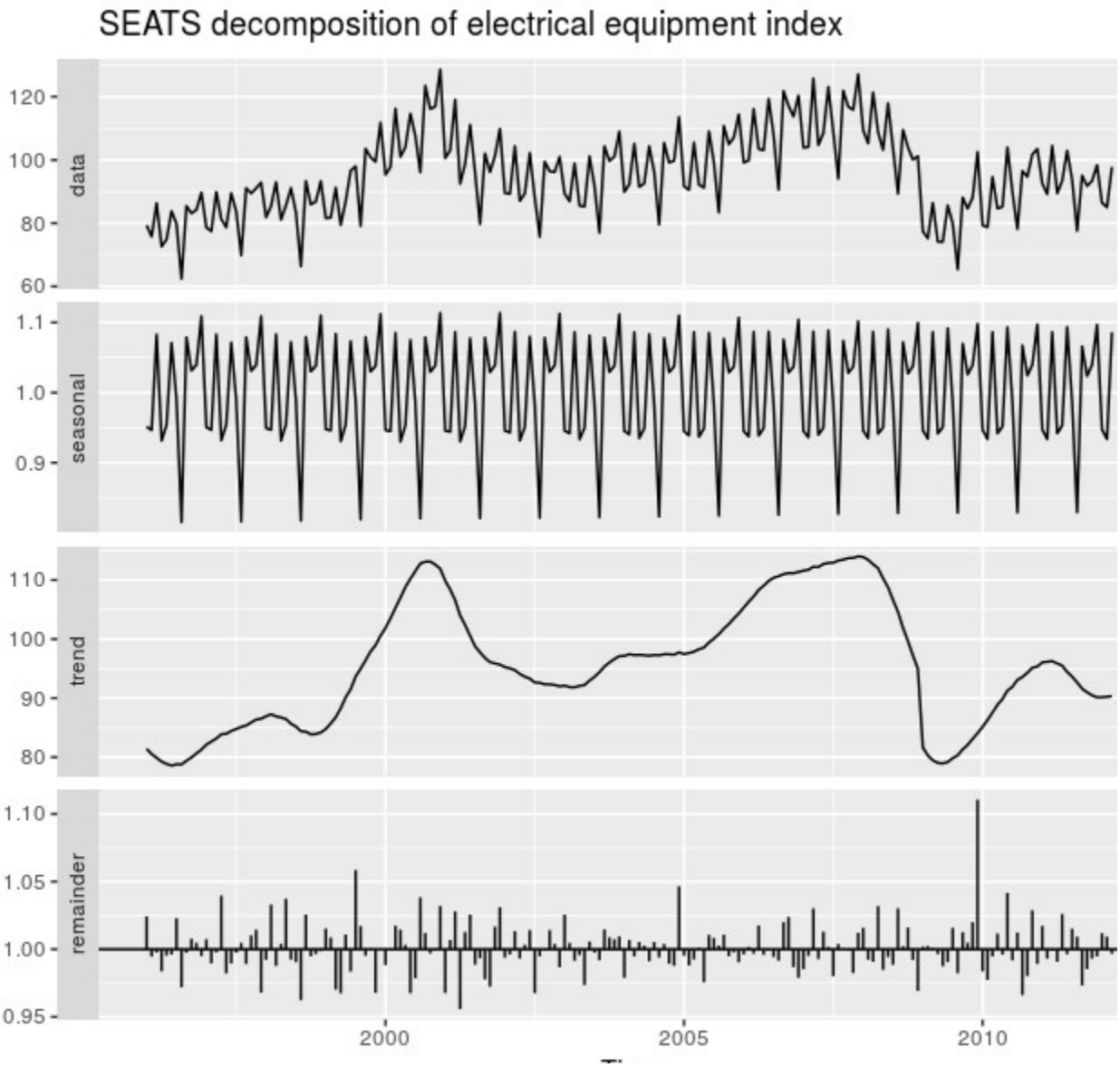


Figure 6.12: A SEATS decomposition of the new orders index for electrical equipment.

The result is quite similar to the X11 decompsotion shown in Figure 6.9.

As with the X11 method, we can use the `seasonal()`, `trendcycle()` and `remainder()` functions to extract the individual components, and `seasadj()` to compute the seasonally adjusted time series.

The `seasonal` package has many options for handling variations of X11 and SEATS. See the package website for a detailed introduction to the options and features available.

6.6 STL decomposition

STL is a very versatile and robust method for decomposing time series. STL is an acronym for "Seasonal and Trend decomposition using Loess", while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990).

STL has several advantages over the classical, SEATS and X-11 decomposition methods:

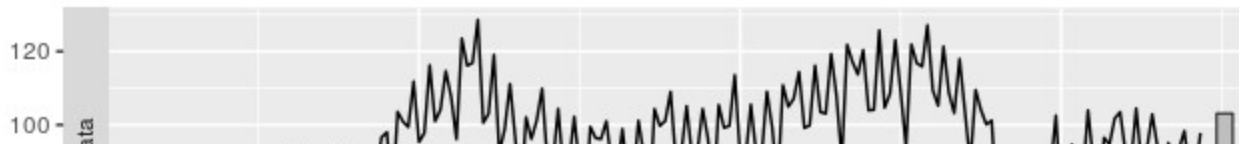
- Unlike SEATS and X-11, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.
- The smoothness of the trend-cycle can also be controlled by the user.
- It can be robust to outliers (i.e., the user can specify a robust decomposition), so that occasional unusual observations will not affect the estimates of the trend-cycle and seasonal components. They will, however, affect the remainder component.

On the other hand, STL has some disadvantages. In particular, it does not handle trading day or calendar variation automatically, and it only provides facilities for additive decompositions.

It is possible to obtain a multiplicative decomposition by first taking logs of the data, then back-transforming the components. Decompositions between additive and multiplicative can be obtained using a Box-Cox transformation of the data with $0<\lambda<1$. A value of $\lambda=0$ corresponds to the multiplicative decomposition while $\lambda=1$ is equivalent to an additive decomposition.

The best way to begin learning how to use STL is to see some examples and experiment with the settings. Figure 6.2 showed an example of STL applied to the electrical equipment orders data. Figure 6.13 shows an alternative STL decomposition where the trend-cycle is more flexible, the seasonal component does not change over time, and the robust option has been used. Here, it is more obvious that there has been a down-turn at the end of the series, and that the orders in 2009 were unusually low (corresponding to some large negative values in the remainder component).

```
elecequip %>%
  stl(t.window=13, s.window="periodic", robust=TRUE) %>%
  autoplot
```



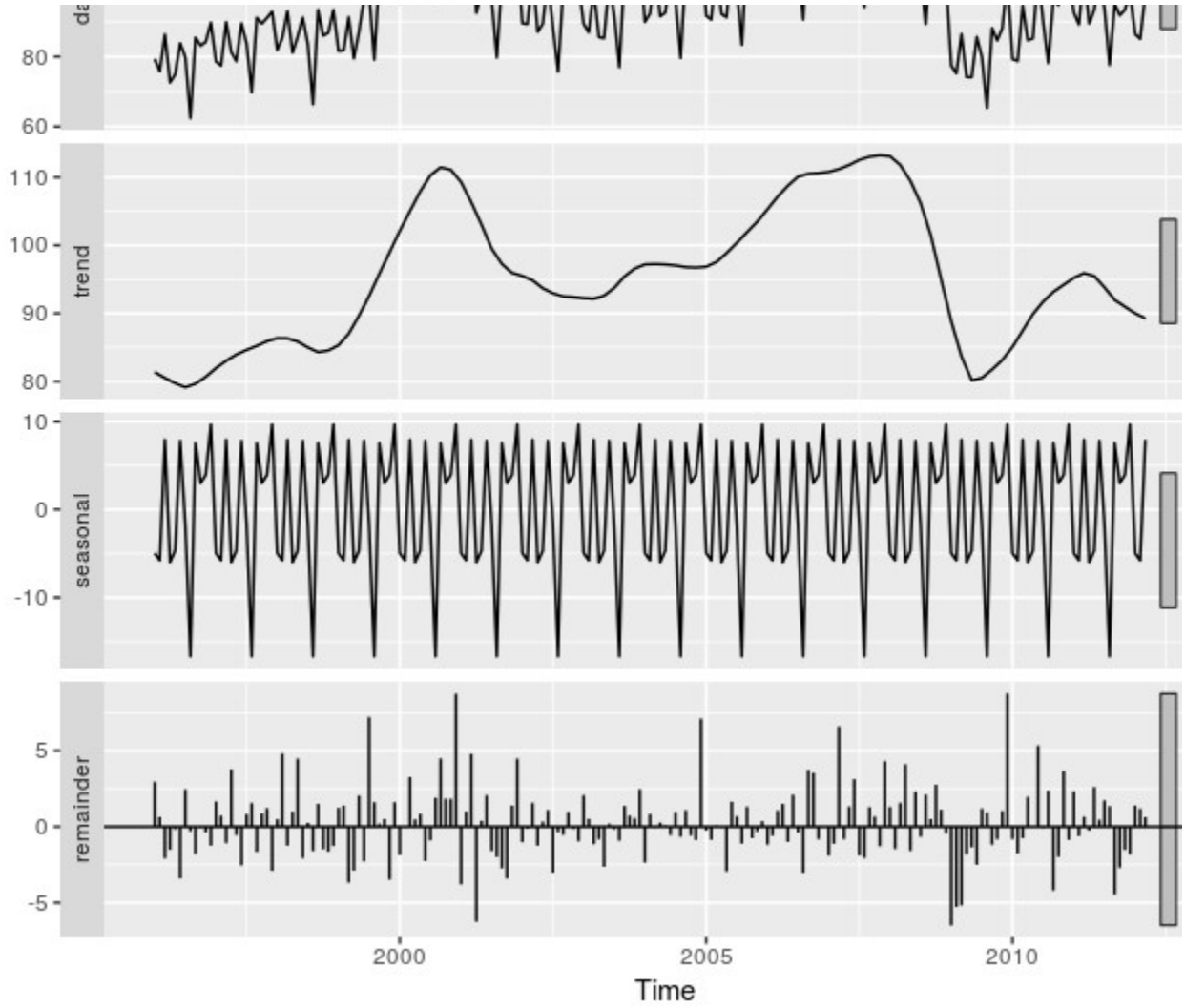


Figure 6.13: The electrical equipment orders (top) and its three additive components obtained from a robust STL decomposition with flexible trend-cycle and fixed seasonality.

The two main parameters to be chosen when using STL are the trend-cycle window (`t.window`) and the seasonal window (`s.window`). These control how rapidly the trend-cycle and seasonal components can change. Smaller values allow for more rapid changes. Both `t.window` and `s.window` should be odd numbers and refer to the number of consecutive years to be used when estimating the trend-cycle and seasonal components respectively. The user must specify `s.window` as there is no default. Setting it to to be infinite is equivalent to forcing the seasonal component to be periodic (i.e., identical across years). Specifying `t.window` is optional, and a default value will be used if it is omitted.

As with the other decomposition methods discussed in this book, to obtain the separate components plotted in Figure 6.8, use the `seasonal()` function for the seasonal component, the `trendcycle()` function for trend-cycle component, and the `remainder()` function for the remainder component. The `seasadj()` function can be used to compute the seasonally adjusted series.

6.7 Forecasting with decomposition

While decomposition is primarily useful for studying time series data, and exploring the historical changes over time, it can also be used in forecasting.

To forecast a decomposed time series, we forecast the seasonal component, ˆSt, and the seasonally adjusted component ˆAt, separately. It is usually assumed that the seasonal component is unchanging, or changing extremely slowly, so it is forecast by simply taking the last year of the estimated component. In other words, a seasonal naïve method is used for the seasonal component.

To forecast the seasonally adjusted component, any non-seasonal forecasting method may be used. For example, a random walk with drift model, or Holt's method (discussed in the next chapter), or a non-seasonal ARIMA model (discussed in Chapter 8), may be used.

Example: Electrical equipment manufacturing

```
fit <- stl(elecequip, t.window=13, s.window="periodic", robust=TRUE)
eeadj <- seasadj(fit)
autoplot(naive(eeadj)) + ylab("New orders index") +
  ggtitle("Naive forecasts of seasonally adjusted data")
```

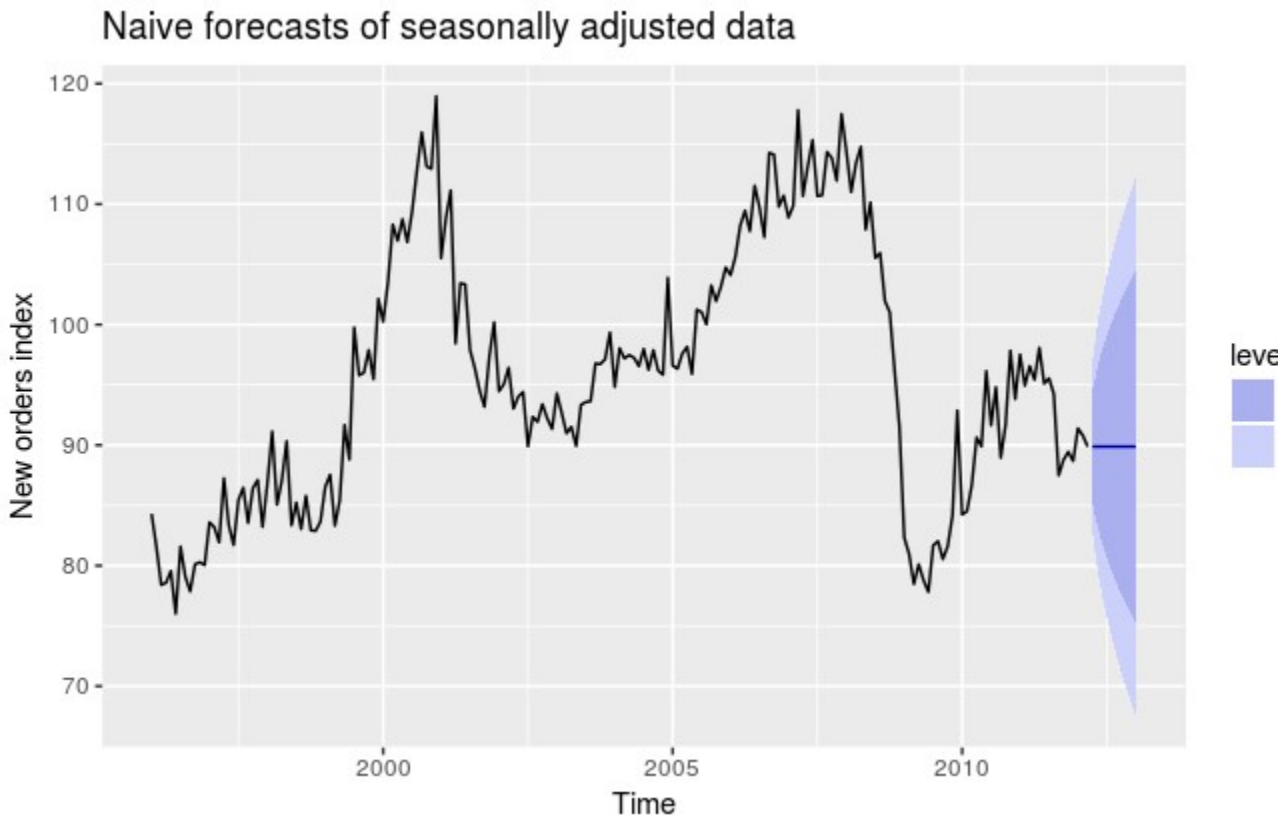


Figure 6.14: Naïve forecasts of the seasonally adjusted data obtained from an STL decomposition of the electrical equipment orders data.

Figure 6.14 shows naïve forecasts of the seasonally adjusted electrical equipment orders data. These are then "reseasonalized" by adding in the seasonal naïve forecasts of the seasonal component.

This is made easy with the `forecast` function applied to the `stl` object. You need to specify the method being used on the seasonally adjusted data, and the function will do the re-seasonalizing for you. The resulting forecasts of the original data are shown in Figure 6.15.

```
fcast <- forecast(fit, method="naive")
autoplot(fcast) + ylab("New orders index")
```

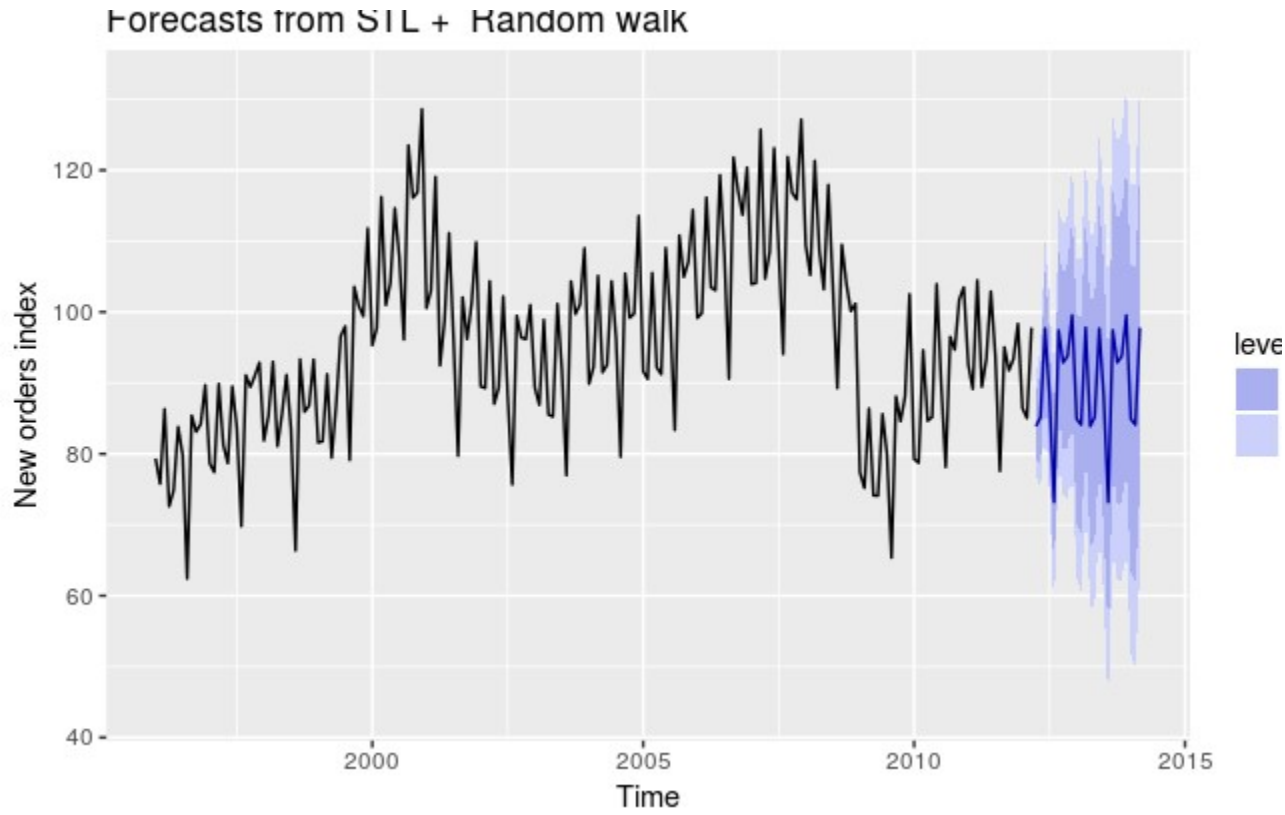


Figure 6.15: Forecasts of the electrical equipment orders data based on a naïve forecast of the seasonally adjusted data and a seasonal naïve forecast of the seasonal component, after an STL decomposition of the data.

The prediction intervals shown in this graph are constructed in the same way as the point forecasts. That is, the upper and lower limits of the prediction intervals on the seasonally adjusted data are "reseasonalized" by adding in the forecasts of the seasonal component. In this calculation, the uncertainty in the forecasts of the seasonal component has been ignored. The rationale for this choice is that the uncertainty in the seasonal component is much smaller than that for the seasonally adjusted data, and so it is a reasonable approximation to ignore it.

A short-cut approach is to use the `stlf` function. The following code will decompose the time series using STL, forecast the seasonally adjusted series, and return reseasonalize the forecasts.

```
fcast <- stlf(eeadj, method='naive')
```

The `stlf` function uses default values for `s.window` and `t.window`.

As well as the naïve method, several other possible forecasting methods are available with `stlf`, as described in the corresponding help file. If `method` is not specified, it will use the ETS approach (discussed in the next chapter) applied to the seasonally adjusted series. This usually produces quite good forecasts for seasonal time series, and some companies use it routinely for all their operational forecasts.

6.8 Exercises

1. Show that a 3×5 MA is equivalent to a 7-term weighted moving average with weights of 0.067, 0.133, 0.200, 0.200, 0.200, 0.133, and 0.067.

2. The `plastics` data set consists of the monthly sales (in thousands) of product A for a plastics manufacturer for five years.
    1. Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend-cycle?
    2. Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal indices.
    3. Do the results support the graphical interpretation from part (a)?
    4. Compute and plot the seasonally adjusted data.
    5. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?
    6. Does it make any difference if the outlier is near the end rather than in the middle of the time series?

3. Recall your retail time series data (from Exercise 1 in Section 2.10). Decompose the series using X11. Does it reveal any outliers, or unusual features that you had not noticed previously?

4. Figures 6.16 and 6.17 shows the result of decomposing the number of persons in the civilian labor force in Australia each month from February 1978 to August 1995.

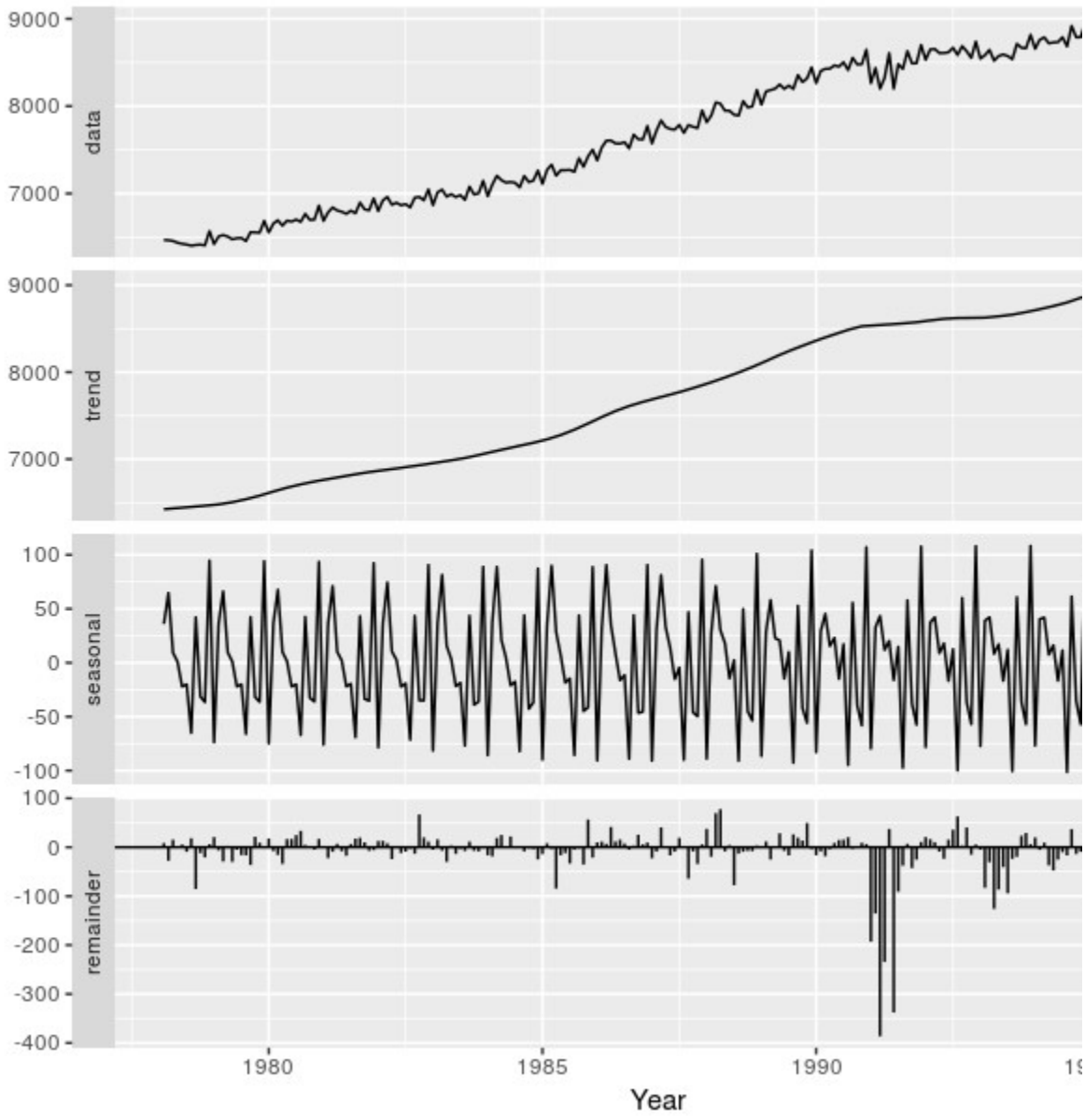


Figure 6.16: Decomposition of the number of persons in the civilian labor force in Australia each month from February 1978 to August 1995.

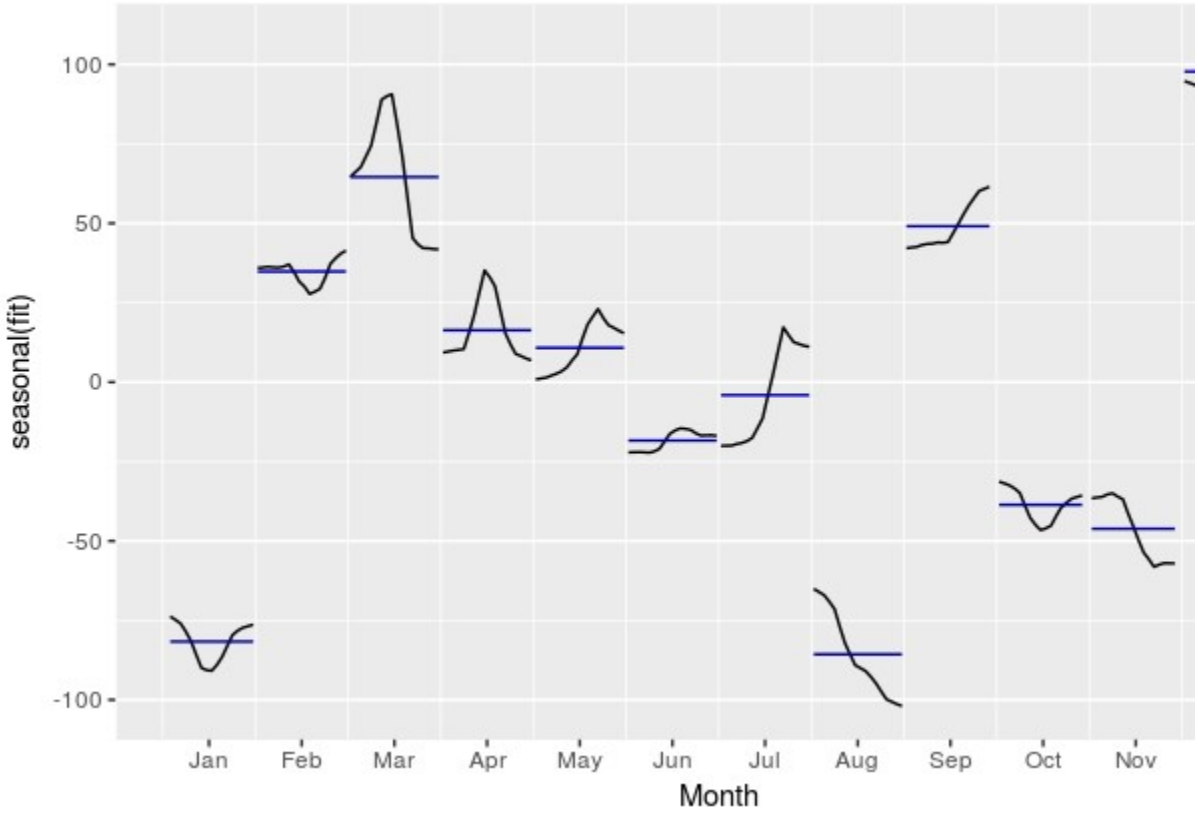


Figure 6.17: Seasonal component from the decomposition shown in Figure 6.16.

1. Write about 3–5 sentences describing the results of the seasonal adjustment. Pay particular attention to the scales of the graphs in making your interpretation.
2. Is the recession of 1991/1992 visible in the estimated components?

5. This exercise uses the `cangas` data (monthly Canadian gas production in billions of cubic metres, January 1960 – February 2005).
    1. Plot the data using `autoplot`, `ggsubseriesplot` and `ggseasonplot` to look at the effect of the changing seasonality over time. What do you think is causing it to change so much?
    2. Do an STL decomposition of the data. You will need to choose `s.window` to allow for the changing shape of the seasonal component.
    3. Compare the results with those obtained using SEATS and X11. How are they different?
6. We will use the `bricksq` data (Australian quarterly clay brick production. 1956–1994) for this exercise.
    1. Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality.)
    2. Compute and plot the seasonally adjusted data.
    3. Use a naïve method to produce forecasts of the seasonally adjusted data.
    4. Use `stlf` to reseasonalize the results, giving forecasts for the original data.
    5. Do the residuals look uncorrelated?
    6. Repeat with a robust STL decomposition. Does it make much difference?
    7. Compare forecasts from `stlf` with those from `snaive`, using a test set comprising the last 2 years of data. Which is better?

7. Use `stlf` to produce forecasts of the `writing` series with either `method="naive"` or `method="rwdrift"`, whichever is most appropriate. Use the `lambda` argument if you think a Box-Cox transformation is required.

8. Use `stlf` to produce forecasts of the `fancy` series with either `method="naive"` or `method="rwdrift"`, whichever is most appropriate. Use the `lambda` argument if you think a Box-Cox transformation is required.

6.9 Further reading

A detailed modern discussion of SEATS and X11 decomposition methods is provided by Dagum and Bianconcini (2016). Cleveland et al. (1990) introduced STL, and still provides the best description of the algorithm. For a discussion of forecasting using STL, see Theodosiou (2011).

References

Dagum, Estela Bee, and Silvia Bianconcini. 2016. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation:* Springer.

Cleveland, Robert B, William S Cleveland, Jean E McRae, and Irma J Terpenning. 1990. "STL: A Seasonal-Trend Decomposition Procedure Based on Loess." *Journal of Official Statistics* 6 (1): 3–73.

Theodosiou, Marina. 2011. "Forecasting Monthly and Quarterly Time Series Using STL Decomposition." *International Journal of Forecasting* 27 (4): 1178–95.

Chapter 7 Exponential smoothing

Exponential smoothing was proposed in the late 1950s (Brown (1959), Holt (1957), and Winters (1960) are key pioneering works), and has motivated some of the most successful forecasting methods. Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

This chapter is divided into two parts. In the first part (Sections 7.1–7.4) we present the mechanics of the most important exponential smoothing methods, and their application in forecasting time series with various characteristics. This helps us develop an intuition to how these methods work. In this setting, selecting and using a forecasting method may appear to be somewhat ad hoc. The selection of the method is generally based on recognising key components of the time series (trend and seasonal) and the way in which these enter the smoothing method (e.g., in an additive, damped or multiplicative manner).

In the second part of the chapter (Section 7.5) we present the statistical models that underlie exponential smoothing methods. These models generate identical point forecasts to the methods discussed in the first part of the chapter, but also generate prediction intervals. Furthermore, this statistical framework allows for genuine model selection between competing models.

7.1 Simple exponential smoothing

The simplest of the exponentially smoothing methods is naturally called “simple exponential smoothing” (SES)[12]. This method is suitable for forecasting data with no clear trend or seasonal pattern. For example, the data in Figure 7.1 do not display any clear trending behaviour or any seasonality. (There is a rise in the last few years, which might suggest a trend. We will consider whether a trended method would be better for this series later in this chapter.) We have already considered the naïve and the average as possible methods for forecasting such data (Section 3.1).

```
oildata <- window(oil, start=1996)
autoplot(oildata) +
  ylab("Oil (millions of tonnes)") + xlab("Year")
```

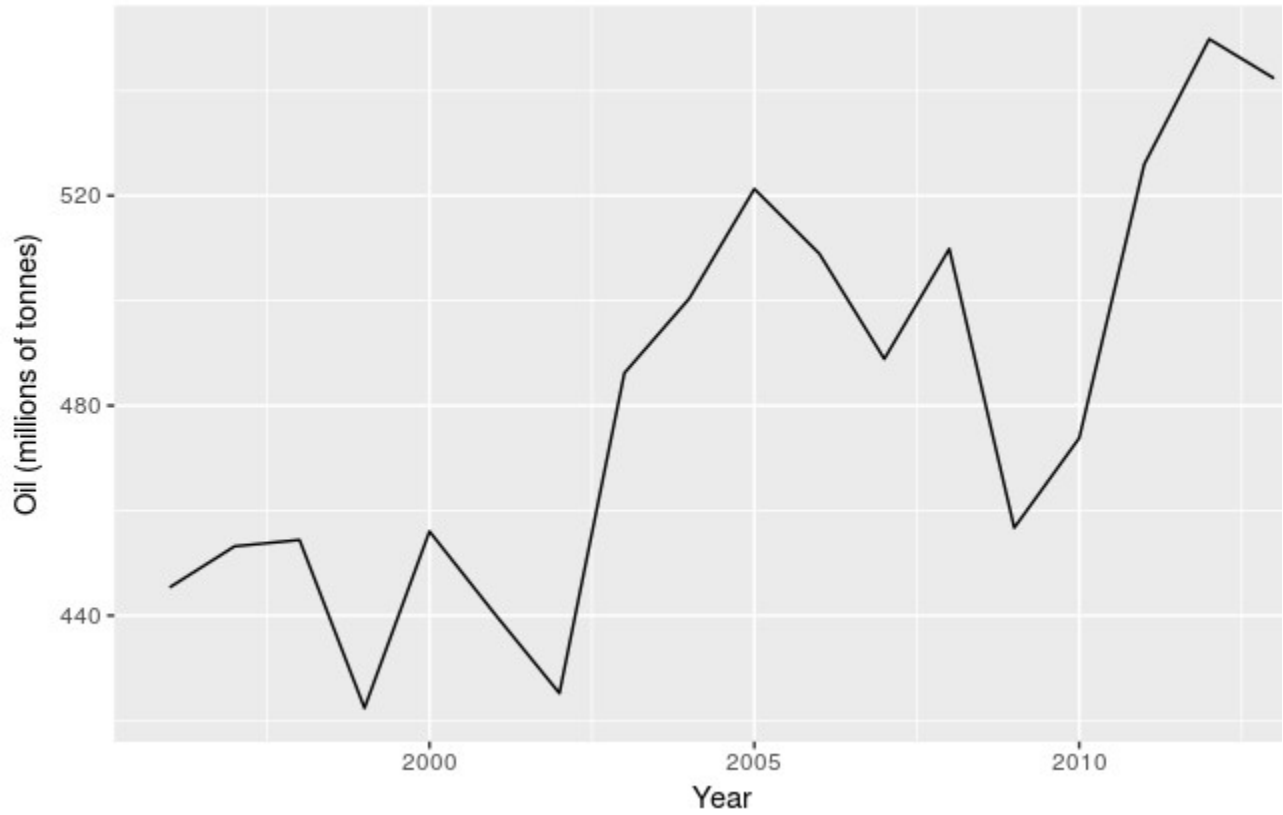


Figure 7.1: Oil production in Saudi Arabia from 1980 to 2013.

Using the naïve method, all forecasts for the future are equal to the last observed value of the series, $\hat{y}_{T+h|T}=y_T$, for $h=1,2,\ldots$. Hence, the naïve method assumes that the most recent observation is the only important one, and all previous observations provide no information for the future. This can be thought of as a weighted average where all of the weight is given to the last observation.

Using the average method, all future forecasts are equal to a simple average of the observed data, $\hat{y}_{T+h|T}=\frac{1}{T}\sum_{t=1}^{T}y_t$, for $h=1,2,\ldots$. Hence, the average method assumes that all observations are of equal importance, and gives them equal weights when generating forecasts.

We often want something between these two extremes. For example, it may be sensible to attach larger weights to more recent observations than to observations from the distant past. This is exactly the concept behind simple exponential smoothing. Forecasts are calculated using weighted averages, where the weights decrease exponentially as observations come from further in the past

— the smallest weights are associated with the oldest observations: $\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \alpha(1-\alpha)^3 y_{T-3} + \cdots$,

where $0 \leq \alpha \leq 1$ is the smoothing parameter. The one-step-ahead forecast for time T+1 is a weighted average of all of the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α.

Table 7.1 shows the weights attached to observations for four different values of α when forecasting using simple exponential smoothing. Note that the sum of the weights even for a small value of α will be approximately one for any reasonable sample size.

Table 7.1: Exponentially decaying weights attached to observations of the time series when generating forecasts using simple exponential smoothing.

	α=0.2	**α=0.4**	**α=0.6**	**α=0.8**
y_T	0.2000	0.4000	0.6000	0.8000
y_{T-1}	0.1600	0.2400	0.2400	0.1600
y_{T-2}	0.1280	0.1440	0.0960	0.0320
y_{T-3}	0.1024	0.0864	0.0384	0.0064
y_{T-4}	0.0819	0.0518	0.0154	0.0013
y_{T-5}	0.0655	0.0311	0.0061	0.0003

For any α between 0 and 1, the weights attached to the observations decrease exponentially as we go back in time, hence the name "exponential smoothing". If α is small (i.e., close to 0), more weight is given to observations from the more distant past. If α is large (i.e., close to 1), more weight is given to the more recent observations. For the extreme case where $\alpha=1$, $\hat{y}_{T+1|T} = y_T$, and the forecasts are equal to the naïve forecasts.

We present two equivalent forms of simple exponential smoothing, each of which leads to the forecast equation (7.1).

7.2 Trend methods

Holt's linear trend method

Holt (1957) extended simple exponential smoothing to allow the forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend): Forecast equation$\hat{y}_{t+h|t} = \ell_t + hb_t$Level equation$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$Trend equation$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$,

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, $0 \leq \alpha \leq 1$, and β^* is the smoothing parameter for the trend, $0 \leq \beta^* \leq 1$. (We denote this as β^* instead of β for reasons that will be explained in Section 7.5.)

As with simple exponential smoothing, the level equation here shows that ℓ_t is a weighted average of observation y_t and the one-step-ahead training forecast for time t, here given by $\ell_{t-1} + b_{t-1}$. The trend equation shows that b_t is a weighted average of the estimated trend at time t based on $\ell_t - \ell_{t-1}$ and b_{t-1}, the previous estimate of the trend.

The forecast function is no longer flat but trending. The h-step-ahead forecast is equal to the last estimated level plus h times the last estimated trend value. Hence the forecasts are a linear function of h.

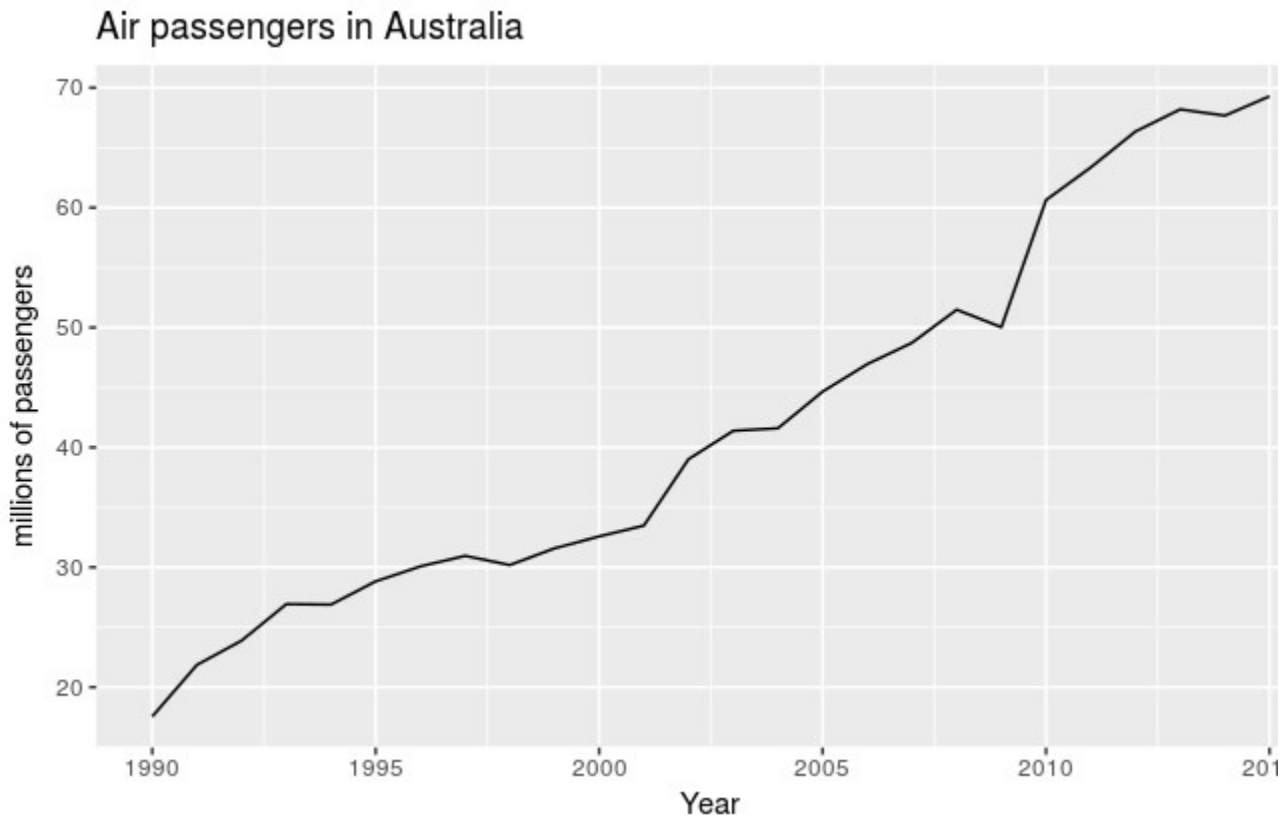


Figure 7.3: Total annual passengers of air carriers registered in Australia. 1990-2014.

Damped trend methods

The forecasts generated by Holt's linear method display a constant trend (increasing or decreasing) indefinitely into the future. Empirical evidence indicates that these methods tend to over-forecast, especially for longer forecast horizons. Motivated by this observation, Gardner Jr and McKenzie (1985) introduced a parameter that "dampens" the trend to a flat line some time in the future. Methods that include a damped trend have proven to be very successful, and are arguably the most popular individual methods when forecasts are required automatically for many series.

In conjunction with the smoothing parameters α and $\beta*$ (with values between 0 and 1 as in Holt's method), this method also includes a damping parameter $0<\phi<1$: $\hat{}yt+h|t=\ell t+(\phi+\phi 2+\cdots+\phi h)$ $bt\ell t=\alpha yt+(1-\alpha)(\ell t-1+\phi bt-1)bt=\beta*(\ell t-\ell t-1)+(1-\beta*)\phi bt-1$.

If $\phi=1$, the method is identical to Holt's linear method. For values between 0 and 1, ϕ dampens the trend so that it approaches a constant some time in the future. In fact, the forecasts converge to $\ell T+\phi bT/(1-\phi)$ as $h\rightarrow\infty$ for any value $0<\phi<1$. This means that short-run forecasts are trended while long-run forecasts are constant.

In practice, ϕ is rarely less than 0.8 as the damping has a very strong effect for smaller values. Values of ϕ close to 1 will mean that a damped model is not able to be distinguished from a non-damped model. For these reasons, we usually restrict ϕ to a minimum of 0.8 and a maximum of 0.98.

Example: Air Passengers (continued)

Figure 7.4 shows the forecasts for years 2014–2018 generated from Holt's linear trend method and the damped trend method.

```
fc <- holt(air, h=15)
fc2 <- holt(air, damped=TRUE, phi = 0.9, h=15)
autoplot(air) +
  forecast::autolayer(fc, PI=FALSE, series="Holt's method") +
  forecast::autolayer(fc2, PI=FALSE, series="Damped Holt's method") +
  ggtitle("Forecasts from Holt's method") +
  xlab("Year") + ylab("Air passengers in Australia (millions)") +
  guides(colour=guide_legend(title="Forecast"))
```

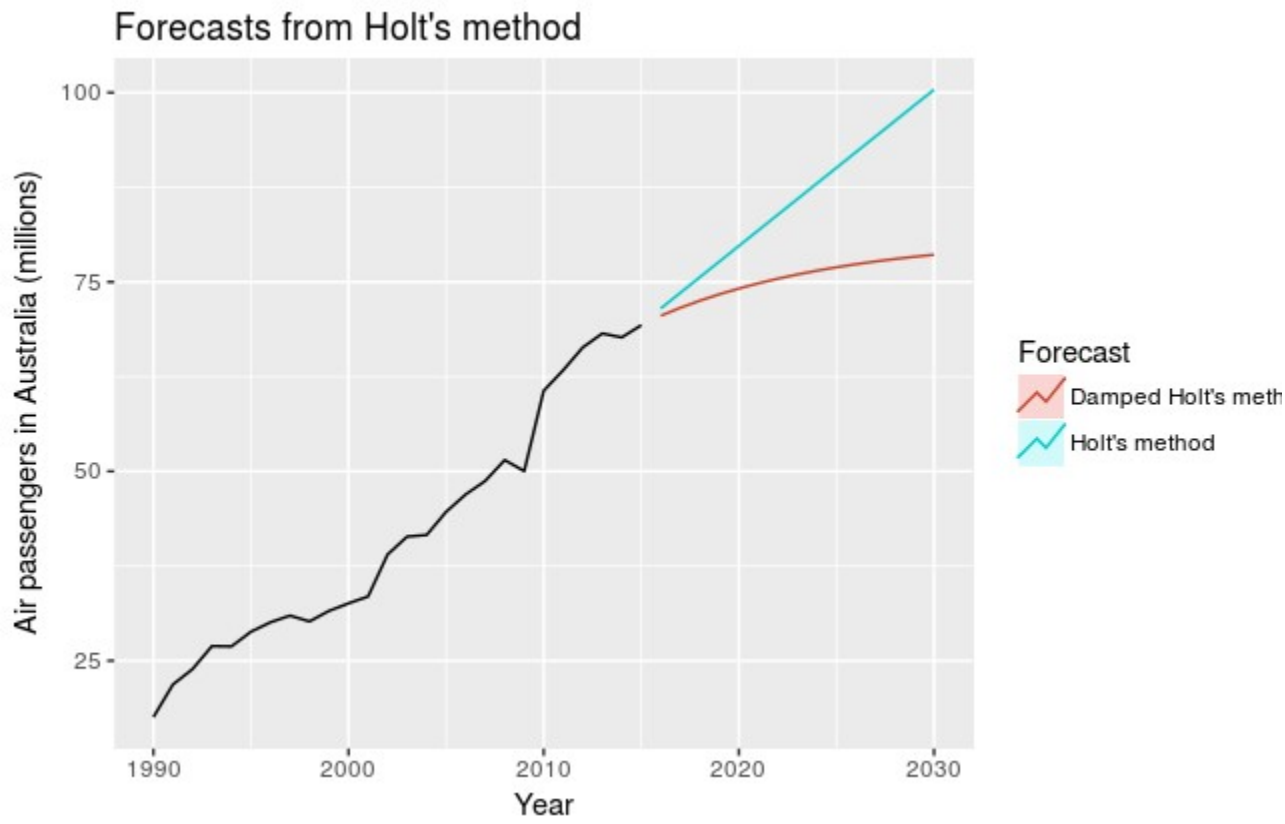


Figure 7.4: Forecasting Air Passengers in Australia (millions of passengers). For the damped trend method, ϕ=0.90.

We have set the damping parameter to a relatively low number (ϕ=0.90) to exaggerate the effect of damping for comparison. Usually, we would estimate ϕ along with the other parameters.

Example: Sheep in Asia

In this example, we compare the forecasting performance of the three exponential smoothing methods that we have considered so far in forecasting the sheep livestock population in Asia. The data spans the period 1970–2007 and is shown in Figure 7.5.

```
autoplot(livestock) +
  xlab("Year") + ylab("Livestock, sheep in Asia (millions)")
```

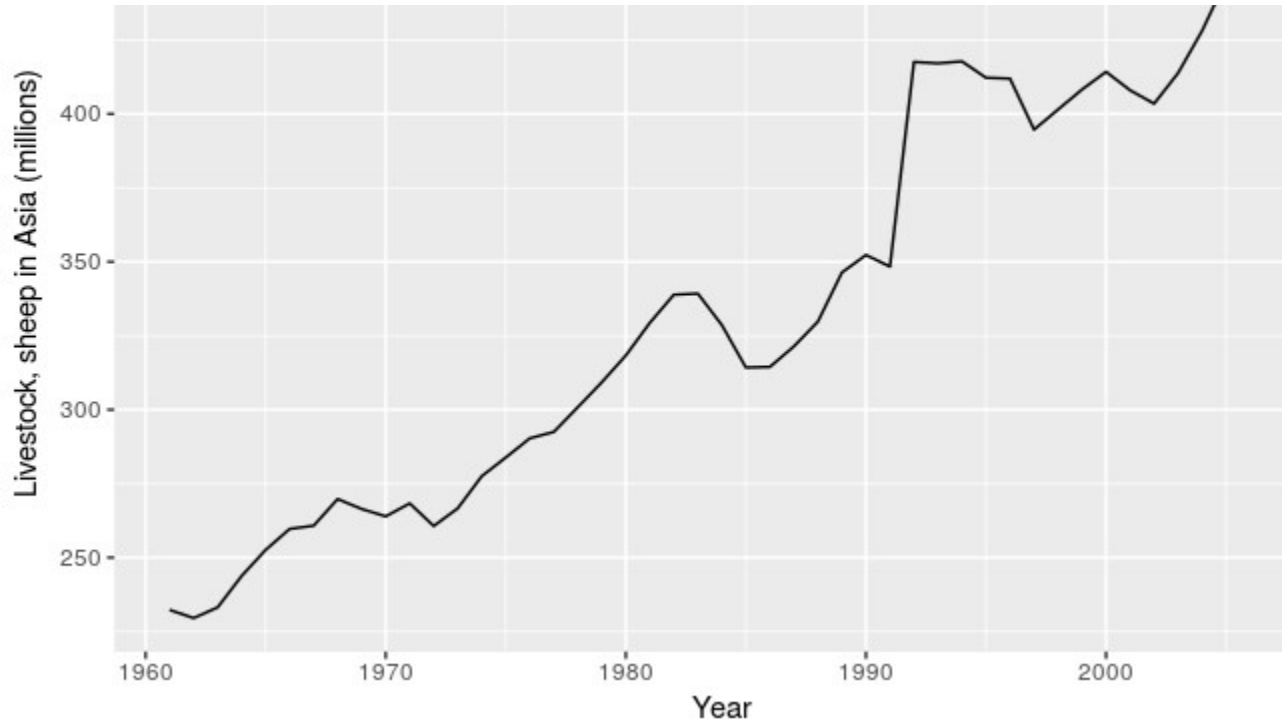


Figure 7.5: Annual sheep livestock numbers in Asia (in million head)

We will use time series cross-validation to compare the one-step forecast accuracy of the three methods.

```
e1 <- tsCV(livestock, ses, h=1)
e2 <- tsCV(livestock, holt, h=1)
e3 <- tsCV(livestock, holt, damped=TRUE, h=1)
# Compare MSE:
mean(e1^2, na.rm=TRUE)
#> [1] 171
mean(e2^2, na.rm=TRUE)
#> [1] 168
mean(e3^2, na.rm=TRUE)
#> [1] 165
# Compare MAE:
mean(abs(e1), na.rm=TRUE)
#> [1] 8.3
mean(abs(e2), na.rm=TRUE)
#> [1] 8.41
mean(abs(e3), na.rm=TRUE)
#> [1] 8.26
```

Based on MSE, Holt's method is best. But based on MAE, simple exponential smoothing is best. Conflicts such as this are common in forecasting comparisons. As forecasting tasks can vary by many dimensions (length of forecast horizon, size of test set, forecast error measures, frequency of data, etc.), it is unlikely that one method will be better than all others for all forecasting scenarios. What we require from a forecasting method are consistently sensible forecasts, and these should be frequently evaluated against the task at hand. In this case, the data are clearly trended, so we will prefer Holt's method, and apply it to the whole data set to get forecasts for future years.

```
fc <- holt(livestock)
# Estimated parameters:
fc[["model"]]
#> Holt's method
#>
```

```
#> Call:
#>  holt(y = livestock)
#>
#>   Smoothing parameters:
#>     alpha = 0.9999
#>     beta  = 1e-04
#>
#>   Initial states:
#>     l = 225.192
#>     b = 4.9532
#>
#>   sigma:  12
#>
#>  AIC AICc  BIC
#>  425  426  434
```

The smoothing parameter for the slope parameter is estimated to be essentially zero, indicating that the trend is not changing over time. The value of α is very close to one, showing that the level reacts strongly to each new observation.

```
autoplot(fc) +
  xlab("Year") + ylab("Livestock, sheep in Asia (millions)")
```

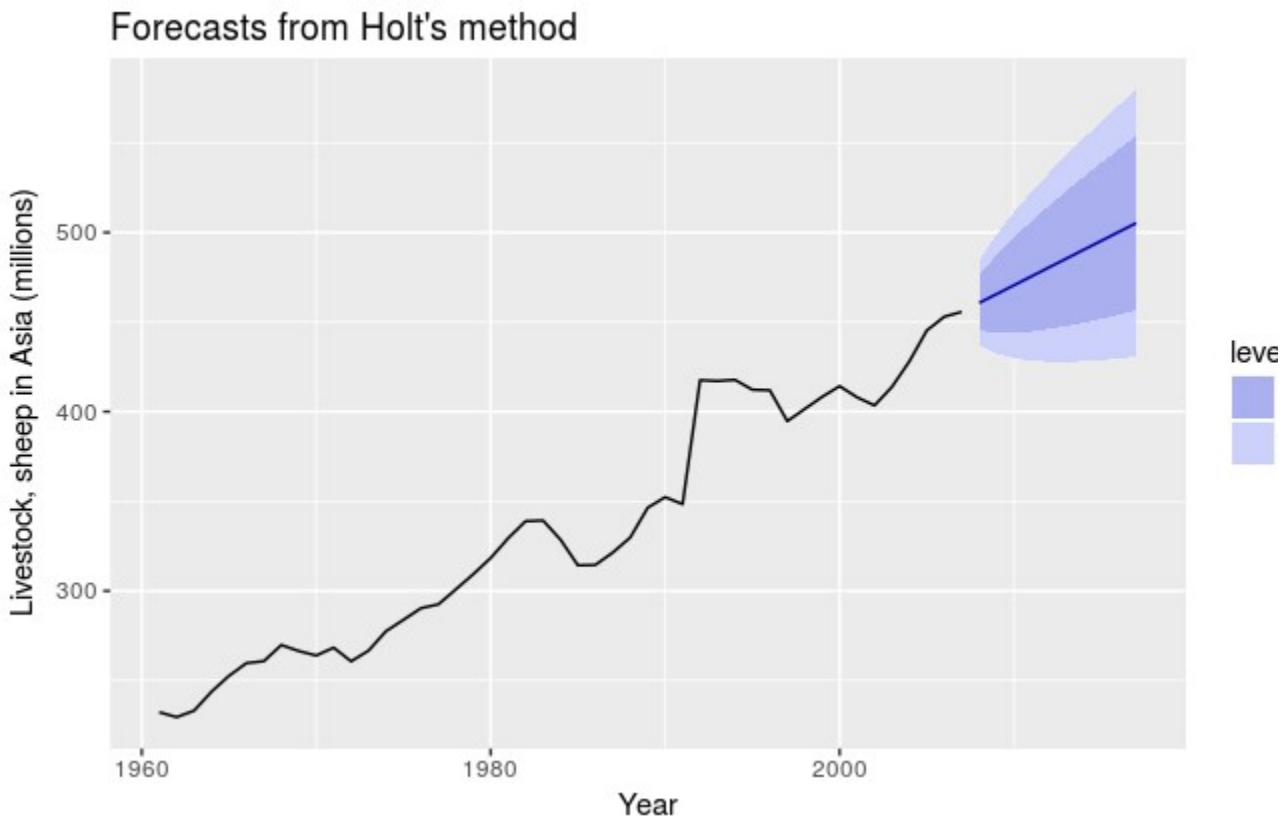


Figure 7.6: Forecasting livestock, sheep in Asia: comparing forecasting performance of non-seasonal method.

The resulting forecasts look sensible with increasing trend, and relatively wide prediction intervals reflecting the variation in the historical data. The prediction intervals are calculated using the methods described in Section 7.5.

7.3 Holt-Winters' seasonal method

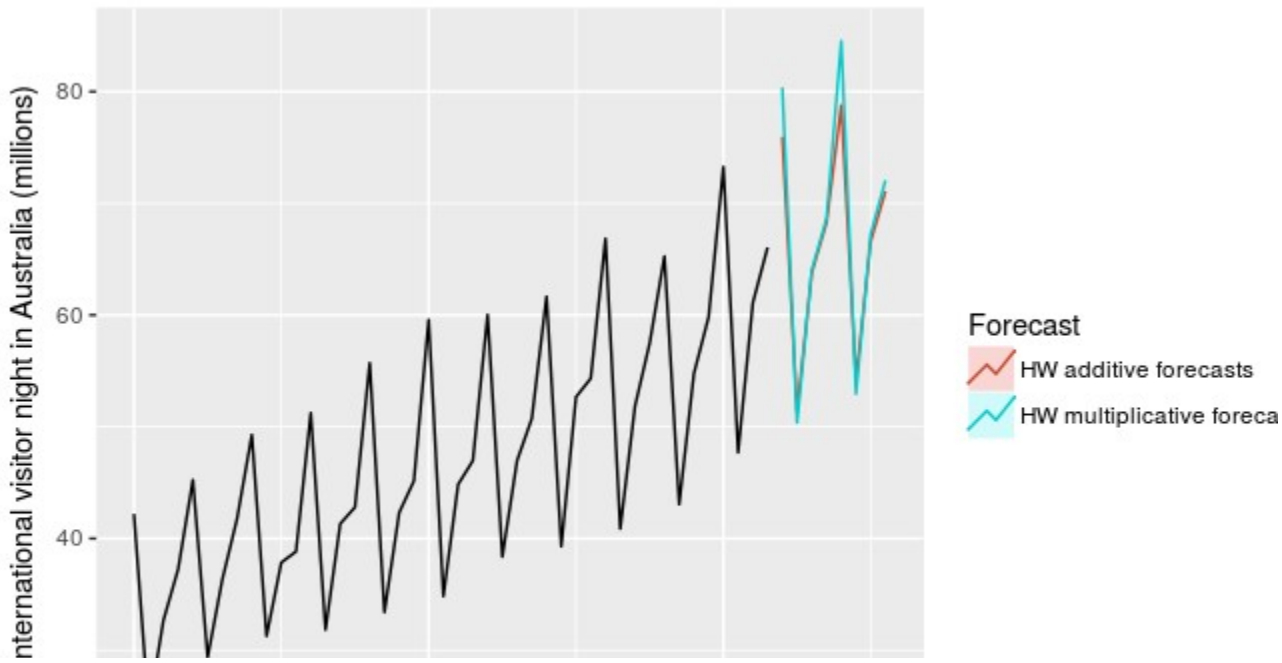
Holt (1957) and Winters (1960) extended Holt's method to capture seasonality. The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level ℓt, one for the trend bt, and one for the seasonal component st, with corresponding smoothing parameters α, $\beta*$ and γ. We use m to denote the frequency of the seasonality, i.e., the number of seasons in a year. For example, for quarterly data $m=4$, and for monthly data $m=12$.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. With the additive method, the seasonal component is expressed in absolute terms in the scale of the observed series, and in the level equation the series is seasonally adjusted by subtracting the seasonal component. Within each year, the seasonal component will add up to approximately zero. With the multiplicative method, the seasonal component is expressed in relative terms (percentages), and the series is seasonally adjusted by dividing through by the seasonal component. Within each year, the seasonal component will sum up to approximately m.

Example: International tourist visitor nights in Australia

In this example we employ the Holt-Winters method with both additive and multiplicative seasonality to forecast quarterly visitor nights in Australia spent by international tourists. Figure 7.7 shows the data from 2005, and the forecasts for 2016–2017. The data show an obvious seasonal pattern, with peaks observed in the March quarter of each year, corresponding to the Australian summer.

```
aust <- window(austourists,start=2005)
fit1 <- hw(aust,seasonal="additive")
fit2 <- hw(aust,seasonal="multiplicative")
autoplot(aust) +
  forecast::autolayer(fit1, PI=FALSE, series="HW additive forecasts") +
  forecast::autolayer(fit2, PI=FALSE, series="HW multiplicative forecasts") +
  xlab("Year") + ylab("International visitor night in Australia (millions)") +
  guides(colour=guide_legend(title="Forecast"))
```



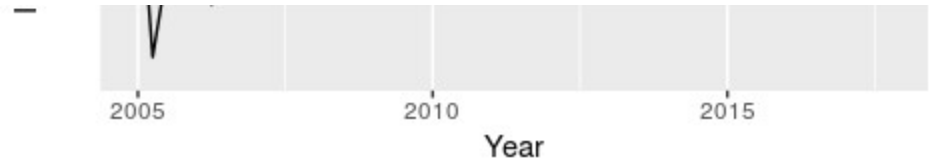


Figure 7.7: Forecasting international visitor nights in Australia using the Holt-Winters method with both additive and multiplicative seasonality.

Table 7.4: Applying Holt-Winters' method with additive seasonality for forecasting international visitor nights in Australia. Notice that the additive seasonal component sums to approximately zero. The smoothing parameters and initial estimates for the components have been estimated by minimizing RMSE (α=0.250, $\beta*$=0.0069, γ=0.456 and RMSE=1.786).

	t	**yt**	**ℓt**	**bt**	**st**	**ŷt**
2004 Q1	-3				-1.41	
2004 Q2	-2				-10.28	
2004 Q3	-1				9.64	
2004 Q4	0		32.26	0.70	2.05	
2005 Q1	1	42.21	32.86	0.70	9.46	42.60
2005 Q2	2	24.65	33.90	0.70	-9.65	23.28
2005 Q3	3	32.67	34.47	0.70	-1.65	33.19
2005 Q4	4	37.26	35.18	0.70	2.07	37.23
2006 Q1	5	45.24	35.86	0.70	9.41	45.35
2006 Q2	6	29.35	37.17	0.71	-8.54	26.91
2006 Q3	7	36.34	37.91	0.71	-1.60	36.23
2006 Q4	8	41.78	38.89	0.71	2.57	40.68
2007 Q1	9	49.28	39.66	0.71	9.53	49.01
2007 Q2	10	31.28	40.23	0.71	-8.79	31.83
2007 Q3	11	37.85	40.57	0.71	-2.28	39.34
2007 Q4	12	38.84	40.02	0.70	0.28	43.84
2008 Q1	13	51.24	40.97	0.70	9.98	50.25
2008 Q2	14	31.84	41.41	0.70	-9.26	32.87
2008 Q3	15	41.32	42.48	0.70	-1.59	39.82
2008 Q4	16	42.80	43.01	0.70	-0.02	43.46
2009 Q1	17	55.71	44.21	0.70	10.90	53.69
2009 Q2	18	33.41	44.35	0.70	-10.29	35.65
2009 Q3	19	42.32	44.77	0.70	-2.11	43.46
2009 Q4	20	45.16	45.39	0.70	-0.15	45.44
2010 Q1	21	59.58	46.73	0.70	12.08	56.99
2010 Q2	22	34.84	46.85	0.70	-11.34	37.14
2010 Q3	23	44.84	47.40	0.69	-2.39	45.44
2010 Q4	24	46.97	47.85	0.69	-0.59	47.95

	t	yt	ℓt	bt	st	ˆyt
2011 Q1	25	60.02	48.39	0.69	11.81	60.63
2011 Q2	26	38.37	49.24	0.69	-11.05	37.74
2011 Q3	27	46.98	49.79	0.69	-2.65	47.55
2011 Q4	28	50.73	50.69	0.69	-0.21	49.89
2012 Q1	29	61.65	51.00	0.69	11.10	63.19
2012 Q2	30	39.30	51.36	0.69	-11.66	40.64
2012 Q3	31	52.67	52.86	0.69	-1.15	49.40
2012 Q4	32	54.33	53.80	0.70	0.24	53.35
2013 Q1	33	66.83	54.81	0.70	11.66	65.60
2013 Q2	34	40.87	54.76	0.69	-13.02	43.84
2013 Q3	35	51.83	54.84	0.69	-2.28	54.30
2013 Q4	36	57.49	55.96	0.69	1.03	55.77
2014 Q1	37	65.25	55.88	0.69	10.27	68.31
2014 Q2	38	43.06	56.45	0.69	-13.24	43.55
2014 Q3	39	54.76	57.11	0.69	-2.32	54.85
2014 Q4	40	59.83	58.05	0.69	1.49	58.82
2015 Q1	41	73.26	59.80	0.69	12.21	69.00
2015 Q2	42	47.70	60.60	0.69	-13.04	47.25
2015 Q3	43	61.10	61.83	0.70	-1.36	58.98
2015 Q4	44	66.06	63.04	0.70	2.42	64.02
	h					ˆyT+h|T
2016 Q1	1					75.95
2016 Q2	2					51.40
2016 Q3	3					63.79
2016 Q4	4					68.27
2017 Q1	5					78.76
2017 Q2	6					54.21
2017 Q3	7					66.60
2017 Q4	8					71.07

Table 7.5: Applying Holt-Winters' method with multiplicative seasonality for forecasting international visitor nights in Australia. Notice that the multiplicative seasonal component sums to approximately m=4. The smoothing parameters and initial estimates for the components have been estimated by minimizing RMSE (α=0.479, $\beta*$=0.055, γ=0.0001 and RMSE=1.590).

	t	yt	ℓt	bt	st	ˆyt
2004 Q1	-3				0.97	
2004 Q2	-2				0.77	
2004 Q3	-1				1.24	
2004 Q4	0		33.03	0.92	1.02	
2005 Q1	1	42.21	33.95	0.92	1.24	42.20

	t	yt	ℓt	bt	st	ˆyt
2005 Q2	2	24.65	33.52	0.84	0.77	26.82
2005 Q3	3	32.67	34.11	0.83	0.97	33.18
2005 Q4	4	37.26	35.66	0.87	1.02	35.71
2006 Q1	5	45.24	36.47	0.87	1.24	45.41
2006 Q2	6	29.35	37.73	0.89	0.77	28.72
2006 Q3	7	36.34	38.15	0.86	0.97	37.29
2006 Q4	8	41.78	39.90	0.91	1.02	39.87
2007 Q1	9	49.28	40.25	0.88	1.24	50.74
2007 Q2	10	31.28	40.91	0.87	0.77	31.64
2007 Q3	11	37.85	40.54	0.80	0.97	40.34
2007 Q4	12	38.84	39.74	0.71	1.02	42.25
2008 Q1	13	51.24	40.82	0.73	1.24	50.28
2008 Q2	14	31.84	41.47	0.73	0.77	31.96
2008 Q3	15	41.32	42.49	0.74	0.97	40.75
2008 Q4	16	42.80	42.58	0.71	1.02	44.19
2009 Q1	17	55.71	44.02	0.75	1.24	53.81
2009 Q2	18	33.41	44.13	0.71	0.77	34.44
2009 Q3	19	42.32	44.35	0.69	0.97	43.30
2009 Q4	20	45.16	44.63	0.66	1.02	46.03
2010 Q1	21	59.58	46.55	0.73	1.24	56.30
2010 Q2	22	34.84	46.33	0.68	0.77	36.38
2010 Q3	23	44.84	46.74	0.67	0.97	45.39
2010 Q4	24	46.97	46.71	0.63	1.02	48.45
2011 Q1	25	60.02	47.79	0.65	1.24	58.84
2011 Q2	26	38.37	49.13	0.69	0.77	37.26
2011 Q3	27	46.98	49.26	0.66	0.97	48.11
2011 Q4	28	50.73	49.78	0.65	1.02	51.02
2012 Q1	29	61.65	50.03	0.63	1.24	62.69
2012 Q2	30	39.30	50.87	0.64	0.77	38.97
2012 Q3	31	52.67	52.96	0.72	0.97	49.74
2012 Q4	32	54.33	53.43	0.71	1.02	54.87
2013 Q1	33	66.83	53.96	0.70	1.24	67.30
2013 Q2	34	40.87	53.93	0.66	0.77	42.05
2013 Q3	35	51.83	54.15	0.63	0.97	52.71
2013 Q4	36	57.49	55.48	0.67	1.02	55.99
2014 Q1	37	65.25	54.40	0.58	1.24	69.80
2014 Q2	38	43.06	55.46	0.60	0.77	42.29
2014 Q3	39	54.76	56.37	0.62	0.97	54.13
2014 Q4	40	59.83	57.73	0.66	1.02	58.25
2015 Q1	41	73.26	58.65	0.67	1.24	72.58
2015 Q2	42	47.70	60.61	0.74	0.77	45.64
2015 Q3	43	61.10	62.27	0.80	0.97	59.24

	t	yt	ℓt	bt	st	ˆyt
2015 Q4	44	66.06	63.81	0.84	1.02	64.46
	h					ˆyT+h|T
2016 Q1	1					80.36
2016 Q2	2					50.38
2016 Q3	3					64.04
2016 Q4	4					68.64
2017 Q1	5					84.52
2017 Q2	6					52.95
2017 Q3	7					67.27
2017 Q4	8					72.06

The applications of both methods (with additive and multiplicative seasonality) are presented in Tables 7.4 and 7.5 respectively. Because both methods have exactly the same number of parameters to estimate, we can compare the training RMSE from both models. In this case, the method with multiplicative seasonality fits the data best. This was to be expected, as the time plot shows that the seasonal variation in the data increases as the level of the series increases. This is also reflected in the two sets of forecasts; the forecasts generated by the method with the multiplicative seasonality display larger and increasing seasonal variation as the level of the forecasts increases compared to the forecasts generated by the method with additive seasonality.

The estimated states for both models are plotted in Figure 7.8. The small value of γ for the multiplicative model means that the seasonal component hardly changes over time. The small value of $\beta*$ for the additive model means the slope component hardly changes over time (check the vertical scale).

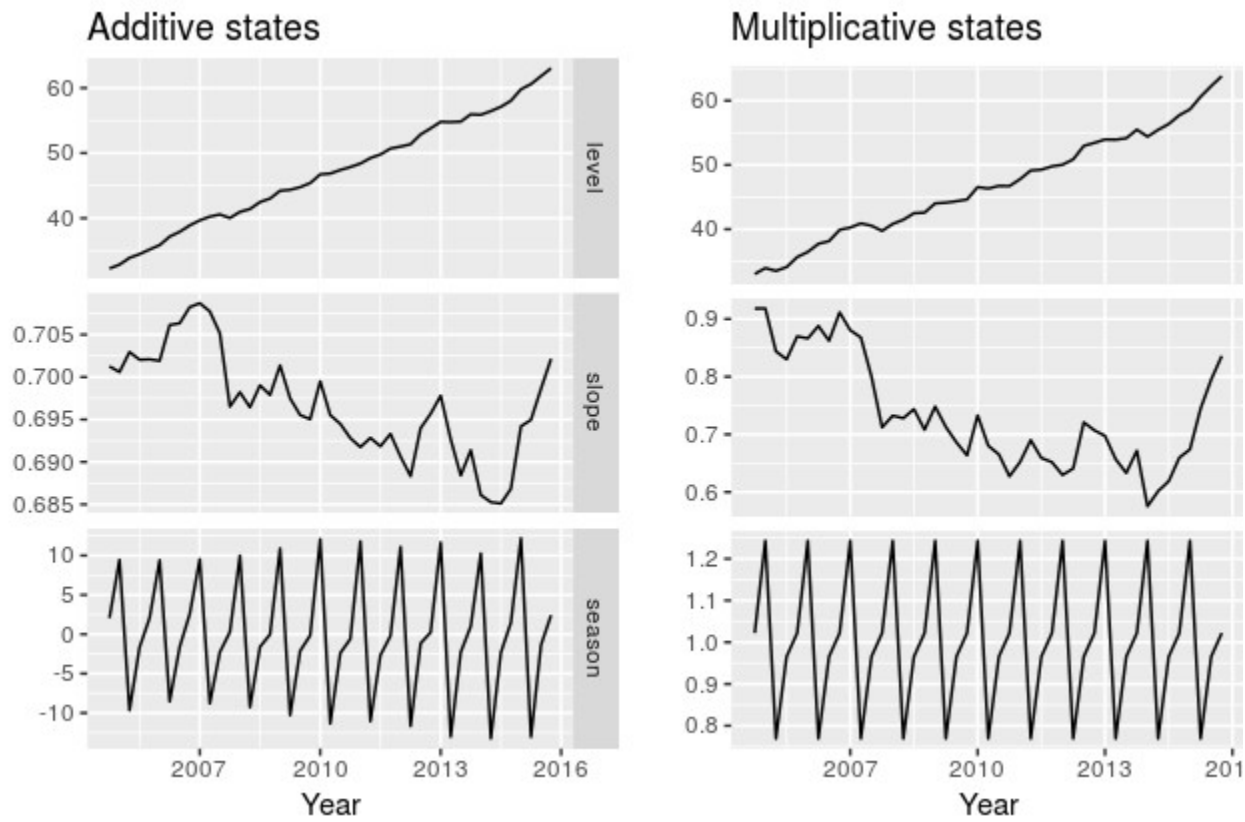


Figure 7.8: Estimated components for the Holt-Winters method with additive and multiplicative seasonal components.

7.4 A taxonomy of exponential smoothing methods

Exponential smoothing methods are not restricted to those we have presented so far. By considering variations in the combinations of the trend and seasonal components, nine exponential smoothing methods are possible, listed in Table 7.6. Each method is labelled by a pair of letters (T,S) defining the type of 'Trend' and 'Seasonal' components. For example, (A,M) is the method with an additive trend and multiplicative seasonality; (A_d,N) is the method with damped trend and no seasonality; and so on.

Table 7.6: A two-way classification of exponential smoothing methods.

Trend Component	**Seasonal**	**Component**	
	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
A_d (Additive damped)	(A_d,N)	(A_d,A)	(A_d)

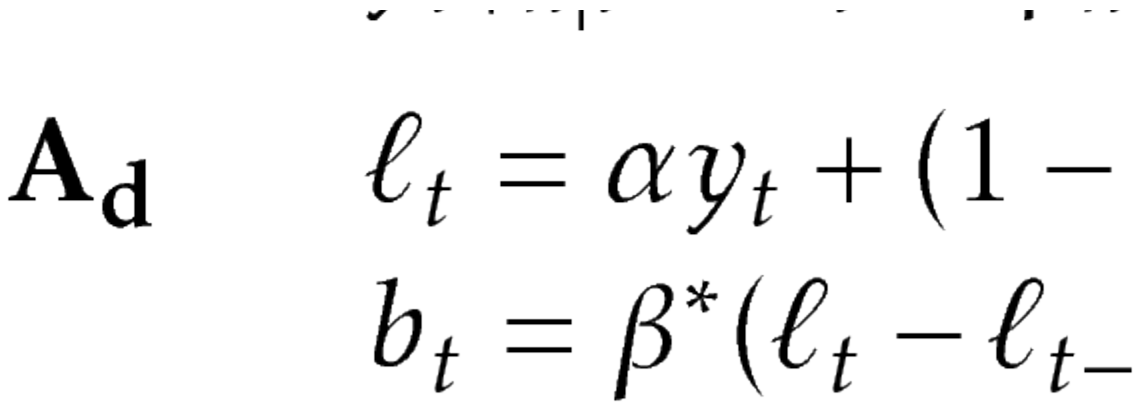
Some of these methods we have already seen:

Short hand	**Method**
(N,N)	Simple exponential smoothing
(A,N)	Holt's linear method
(A_d,N)	Additive damped trend method
(A,A)	Additive Holt-Winters' method
(A,M)	Multiplicative Holt-Winters' method
(A_d,M)	Holt-Winters' damped method

This type of classification was first proposed by Pegels (1969), who also included a method with a multiplicative trend. It was later extended by Gardner Jr (1985) to include methods with an additive damped trend and by Taylor (2003) to include methods with a multiplicative damped trend. We do not consider the multiplicative trend methods in this book as they tend to produce poor forecasts. See Hyndman, Koehler, et al. (2008a) for a more thorough discussion of all exponential smoothing methods.

Table 7.7 gives the recursive formulae for applying the nine exponential smoothing methods in Table 7.6. Each cell includes the forecast equation for generating h-step-ahead forecasts, and the smoothing equations for applying the method.

Trend	
	$\hat{y}_{t+h\|t} = \ell_t$
N	$\ell_t = \alpha y_t + (1 -$
	$\hat{y}_{t+h\|t} = \ell_t + hb_t$
A	$\ell_t = \alpha y_t + (1 -$
	$b_t = \beta^*(\ell_t - \ell_{t-}$
	$\hat{y}_{t+h\|t} = \ell_t + \phi_h$



7.5 Innovations state space models for exponential smoothing

In the rest of this chapter, we study the statistical models that underlie the exponential smoothing methods we have considered so far. The exponential smoothing methods presented in Table 7.7 are algorithms which generate point forecasts. The statistical models in this section generate the same point forecasts, but can also generate prediction (or forecast) intervals. A statistical model is a stochastic (or random) data generating process that can produce an entire forecast distribution. The general statistical framework we will introduce also provides a way of using the model selection criteria introduced in Chapter 5, thus allowing the choice of model to be made in an objective manner.

Each model consists of a measurement equation that describes the observed data, and some transition equations that describe how the unobserved components or states (level, trend, seasonal) change over time. Hence, these are referred to as "state space models".

For each method there exist two models: one with additive errors and one with multiplicative errors. The point forecasts produced by the models are identical if they use the same smoothing parameter values. They will, however, generate different prediction intervals.

To distinguish between a model with additive errors and one with multiplicative errors (and also to distinguish the models from the methods), we add a third letter to the classification of Table 7.6. We label each state space model as ETS(·,·,·) for (Error, Trend, Seasonal). This label can also be thought of as ExponenTial Smoothing. Using the same notation as in Table 7.6, the possibilities for each component are: Error =$\{A,M\}$, Trend =$\{N,A,A_d\}$ and Seasonal =$\{N,A,M\}$.

7.6 Estimation and model selection

Estimating ETS models

An alternative to estimating the parameters by minimizing the sum of squared errors is to maximize the "likelihood". The likelihood is the probability of the data arising from the specified model. Thus, a large likelihood is associated with a good model. For an additive error model, maximizing the likelihood gives the same results as minimizing the sum of squared errors.

However, different results will be obtained for multiplicative error models. In this section, we will estimate the smoothing parameters α, β, γ and ϕ, and the initial states $\ell 0$, b0, s0,s−1,…,s−m+1, by maximizing the likelihood.

The possible values that the smoothing parameters can take are restricted. Traditionally, the parameters have been constrained to lie between 0 and 1 so that the equations can be interpreted as weighted averages. That is, $0<\alpha,\beta*,\gamma*,\phi<1$. For the state space models, we have set $\beta=\alpha\beta*$ and $\gamma=(1-\alpha)\gamma*$. Therefore, the traditional restrictions translate to $0<\alpha<1$, $0<\beta<\alpha$ and $0<\gamma<1-\alpha$. In practice, the damping parameter ϕ is usually constrained further to prevent numerical difficulties in estimating the model. In R, it is restricted so that $0.8<\phi<0.98$.

Another way to view the parameters is through a consideration of the mathematical properties of the state space models. The parameters are constrained in order to prevent observations in the distant past having a continuing effect on current forecasts. This leads to some *admissibility* constraints on the parameters, which are usually (but not always) less restrictive than the usual region. For example, for the ETS(A,N,N) model, the usual parameter region is $0<\alpha<1$ but the admissible region is $0<\alpha<2$. For the ETS(A,A,N) model, the usual parameter region is $0<\alpha<1$ and $0<\beta<\alpha$ but the admissible region is $0<\alpha<2$ and $0<\beta<4-2\alpha$.

Model selection

A great advantage of the ETS statistical framework is that information criteria can be used for model selection. The AIC, AICc and BIC, introduced in Section 5.5, can be used here to determine which of the ETS models is most appropriate for a given time series.

For ETS models, Akaike's Information Criterion (AIC) is defined as $AIC=-2\log(L)+2k$, where L is the likelihood of the model and k is the total number of parameters and initial states that have been estimated (including the residual variance).

The AIC corrected for small sample bias (AICc) is defined as $AICc=AIC+k(k+1)T-k-1$, and the Bayesian Information Criterion (BIC) is $BIC=AIC+k[\log(T)-2]$.

Three of the combinations of (Error, Trend, Seasonal) can lead to numerical difficulties. Specifically, the models that can cause such instabilities are ETS(A,N,M), ETS(A,A,M), and ETS (A,A_d,M). We normally do not consider these particular combinations when selecting a model.

Models with multiplicative errors are useful when the data are strictly positive, but are not numerically stable when the data contain zeros or negative values. Therefore, multiplicative error models will not be considered if the time series is not strictly positive. In that case, only the six fully additive models will be applied.

The `ets()` function in R

The models can be estimated in R using the `ets()` function in the forecast package. Unlike the `ses`, `holt` and `hw` functions, the `ets` function does not produce forecasts. Rather, it estimates the model parameters and returns information about the fitted model.

The R code below shows the most important arguments that this function can take, and their default values. If only the time series is specified, and all other arguments are left at their default values, then an appropriate model will be selected automatically. We explain the arguments below. See the help file for a more complete description.

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL, beta=NULL, gamma=NULL,
    phi=NULL, lambda=NULL, biasadj=FALSE, additive.only=FALSE,
    restrict=TRUE, allow.multiplicative.trend=FALSE)
```

`y`
: The time series to be forecast.

`model`
: A three-letter code indicating the model to be estimated using the ETS classification and notation. The possible inputs are "N" for none, "A" for additive, "M" for multiplicative, or "Z" for automatic selection. If any of the inputs is left as "Z", then this component is selected according to the information criterion chosen. The default value of `ZZZ` ensures that all components are selected using the information criterion.

`damped`
: If `damped=TRUE`, then a damped trend will be used (either A or M). If `damped=FALSE`, then a non-damped trend will used. If `damped=NULL` (the default), then either a damped or a non-damped trend will be selected, depending on which model has the smallest value for the information criterion.

`alpha`, `beta`, `gamma`, `phi`
: The values of the smoothing parameters can be specified using these arguments. If they are set to `NULL` (the default setting for each of them), the parameters are estimated.

`lambda`
: Box-Cox transformation parameter. It will be ignored if `lambda=NULL` (the default value). Otherwise, the time series will be transformed before the model is estimated. When `lambda` is not `NULL`, `additive.only` is set to `TRUE`.

`biasadj`
: If `TRUE` and `lambda` is not `NULL`, then the back-transformed fitted values and forecasts will be bias-adjusted.

`additive.only`
: Only models with additive components will be considered if `additive.only=TRUE`. Otherwise, all models will be considered.

`restrict`
: If `restrict=TRUE` (the default), the models that cause numerical difficulties are not considered in model selection.

`allow.multiplicative.trend`
: Multiplicative trend models are also available, but not covered in this book. Set this argument to `TRUE` to allow these models to be considered.

Working with `ets` objects

The `ets()` function will return an object of class `ets`. There are many R functions designed to make working with `ets` objects easy. A few of them are described below.

`coef()`
: returns all fitted parameters.

`accuracy()`
: returns accuracy measures computed on the training data.

`summary()`
: prints some summary information about the fitted model.

`autoplot()` and `plot()`
: produce time plots of the components.

`residuals()`
: returns residuals from the estimated model.

`fitted()`

returns one-step forecasts for the training data.

`simulate()`
: will simulate future sample paths from the fitted model.

`forecast()`
: computes point forecasts and prediction intervals, as described in the next section.

Example: International tourist visitor nights in Australia

We now employ the ETS statistical framework to forecast tourist visitor nights in Australia by international arrivals over the period 2016–2019. We let the `ets()` function select the model by minimizing the AICc.

```
aust <- window(austourists, start=2005)
fit <- ets(aust)
summary(fit)
#> ETS(M,A,M)
#>
#> Call:
#>  ets(y = aust)
#>
#>   Smoothing parameters:
#>     alpha = 0.47
#>     beta  = 1e-04
#>     gamma = 1e-04
#>
#>   Initial states:
#>     l = 32.3107
#>     b = 0.6903
#>     s=1.02 0.962 0.767 1.25
#>
#>   sigma:  0.0328
#>
#>  AIC AICc  BIC
#>  220  225  236
#>
#> Training set error measures:
#>                    ME RMSE  MAE     MPE MAPE  MASE  ACF1
#> Training set 0.0434 1.57 1.26 -0.0399 2.71 0.412 -0.11
```

The model selected is ETS(M,A,M):

$$y_t = (\ell_{t-1} + b_{t-1})s_{t-m}(1+\varepsilon_t) \quad \ell_t = (\ell_{t-1} + b_{t-1})(1+\alpha\varepsilon_t) \quad b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t1})\varepsilon_t \quad s_t = s_{t-m}(1+\gamma\varepsilon_t).$$

The parameter estimates are $\alpha = 0.4700$, $\beta = 0.0001$, and $\gamma = 0.0001$. The output also returns the estimates for the initial states ℓ_0, b_0, s_0, s_{-1}, s_{-2} and s_{-3}. Compare these with the values obtained for the Holt-Winters method with multiplicative seasonality presented in Table 7.5. Although this model gives equivalent point forecasts to a multiplicative Holt-Winters' method, the parameters have been estimated differently. With the `ets()` function, the default estimation method is maximum likelihood rather than minimum sum of squares.

Figure 7.9 shows the states over time, while Figure 7.11 shows point forecasts and prediction intervals generated from the model. The small values of β and γ mean that the slope and seasonal components change very little over time. The narrow prediction intervals indicate that the series is relatively easy to forecast due to the strong trend and seasonality.

```
autoplot(fit)
```

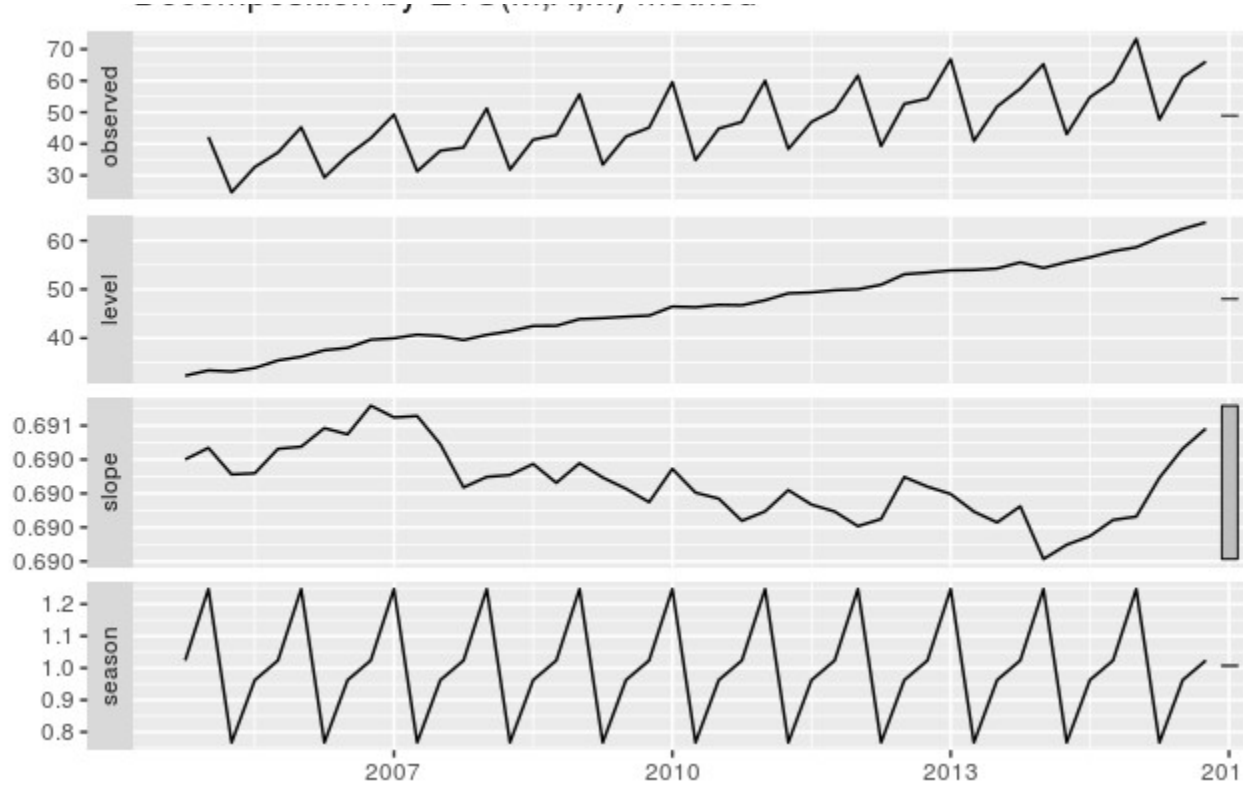
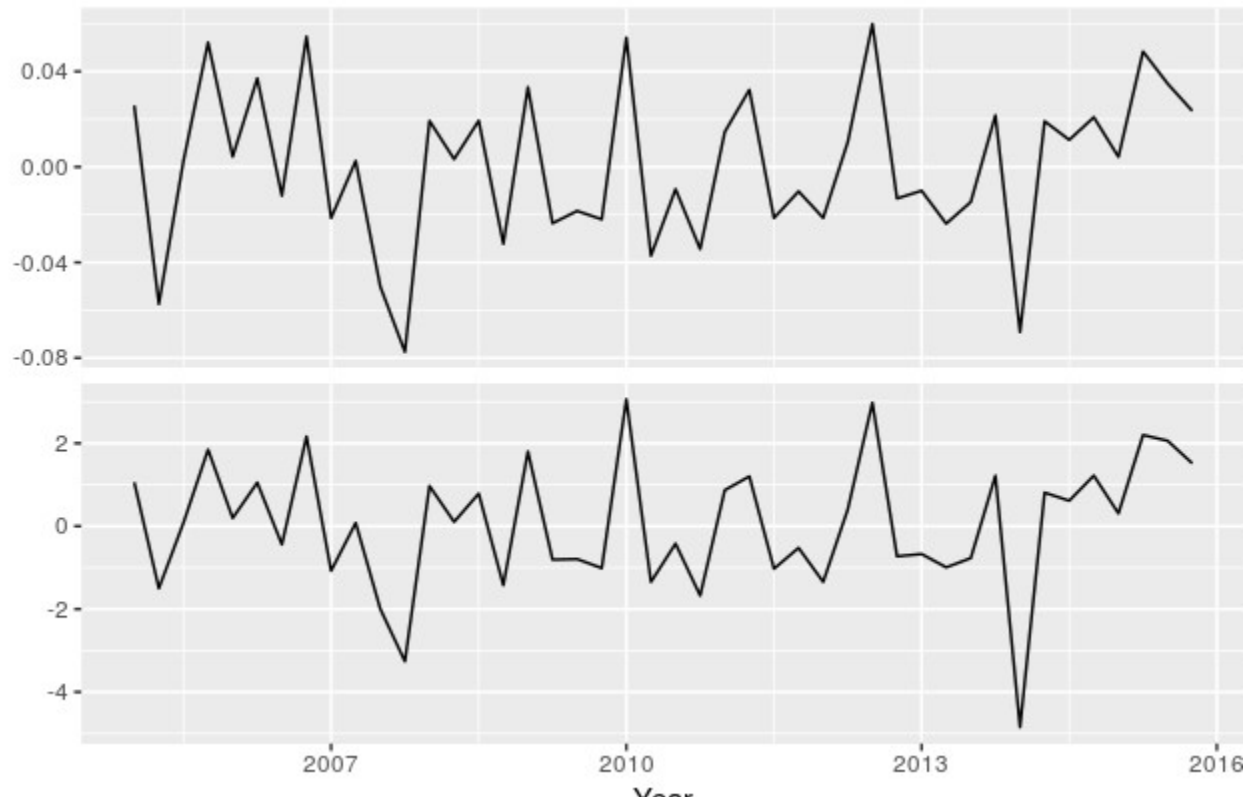


Figure 7.9: Graphical representation of the estimated states over time.

Because this model has multiplicative errors, the residuals are not equivalent to the one-step forecast errors. The residuals are given by ˆεt, while the one-step forecast errors are defined as yt−ˆyt|t−1. We can obtain both using the `residuals` function.

```
cbind('Residuals' = residuals(fit),
      'Forecast errors' = residuals(fit, type='response')) %>%
  autoplot(facet=TRUE) + xlab("Year") + ylab("")
```



Year

Figure 7.10: Residuals and one-step forecast errors from the ETS(M,A,M) model.

7.7 Forecasting with ETS models

Point forecasts are obtained from the models by iterating the equations for t=T+1,…,T+h and setting all εt=0 for t>T.

For example, for model ETS(M,A,N), yT+1=(ℓT+bT)(1+εT+1). Therefore Similarly, yT+2= (ℓT+1+bT+1)(1+εT+1)=[(ℓT+bT)(1+αεT+1)+bT+β(ℓT+bT)εT+1](1+εT+1).

Therefore, ˆyT+2|T=ℓT+2bT, and so on. These forecasts are identical to the forecasts from Holt's linear method, and also to those from model ETS(A,A,N). Thus, the point forecasts obtained from the method and from the two models that underlie the method are identical (assuming that the same parameter values are used).

ETS point forecasts are equal to the medians of the forecast distributions. For models with only additive components, the forecast distributions are normal, so the medians and means are equal. For ETS models with multiplicative errors, or with multiplicative seasonality, the point forecasts will not be equal to the means of the forecast distributions.

A big advantage of the models is that prediction intervals can also be generated — something that cannot be done using the methods. The prediction intervals will differ between models with additive and multiplicative methods.

For some models, there are exact formulae that enable prediction intervals to be calculated (see Hyndman, Koehler, et al. (2008a) for the formulas). A more general approach that works for all models is to simulate future sample paths, conditional on the last estimate of the states, and to obtain prediction intervals from the percentiles of these simulated future paths.

To obtain forecasts from an ETS model, we use the `forecast` function. The R code below shows the possible arguments that this function takes when applied to an ETS model. We explain each of the arguments in what follows.

```
forecast(object, h=ifelse(object$m>1, 2*object$m, 10),
level=c(80,95), fan=FALSE, simulate=FALSE, bootstrap=FALSE, npaths=5000,
PI=TRUE, lambda=object$lambda, biasadj=NULL, ...)
```

`object`
: The object returned by the `ets()` function.

`h`
: The forecast horizon — the number of periods to be forecast.

`level`
: The confidence level for the prediction intervals.

`fan`
: If `fan=TRUE`, `level=seq(50,99,by=1)`. This is suitable for fan plots.

`simulate`
: If `simulate=TRUE`, prediction intervals are produced by simulation rather than using algebraic formulae. Simulation will also be used (even if `simulate=FALSE`) where there are no algebraic formulae available for the particular model.

`bootstrap`
: If `bootstrap=TRUE` and `simulate=TRUE`, then the simulated prediction intervals use re-sampled errors rather than normally distributed errors.

`npaths`
: The number of sample paths used in computing simulated prediction intervals.

`PI`
: If `PI=TRUE`, then prediction intervals are produced; otherwise only point forecasts are calculated.

`lambda`
: The Box-Cox transformation parameter. This is ignored if `lambda=NULL`. Otherwise, the forecasts are back-transformed via an inverse Box-Cox transformation.

`biasadj`
: If `lambda` is not `NULL`, the backtransformed forecasts (and prediction intervals) are bias-adjusted.

```
fit %>% forecast(h=8) %>%
  autoplot() +
  ylab("International visitor night in Australia (millions)")
```

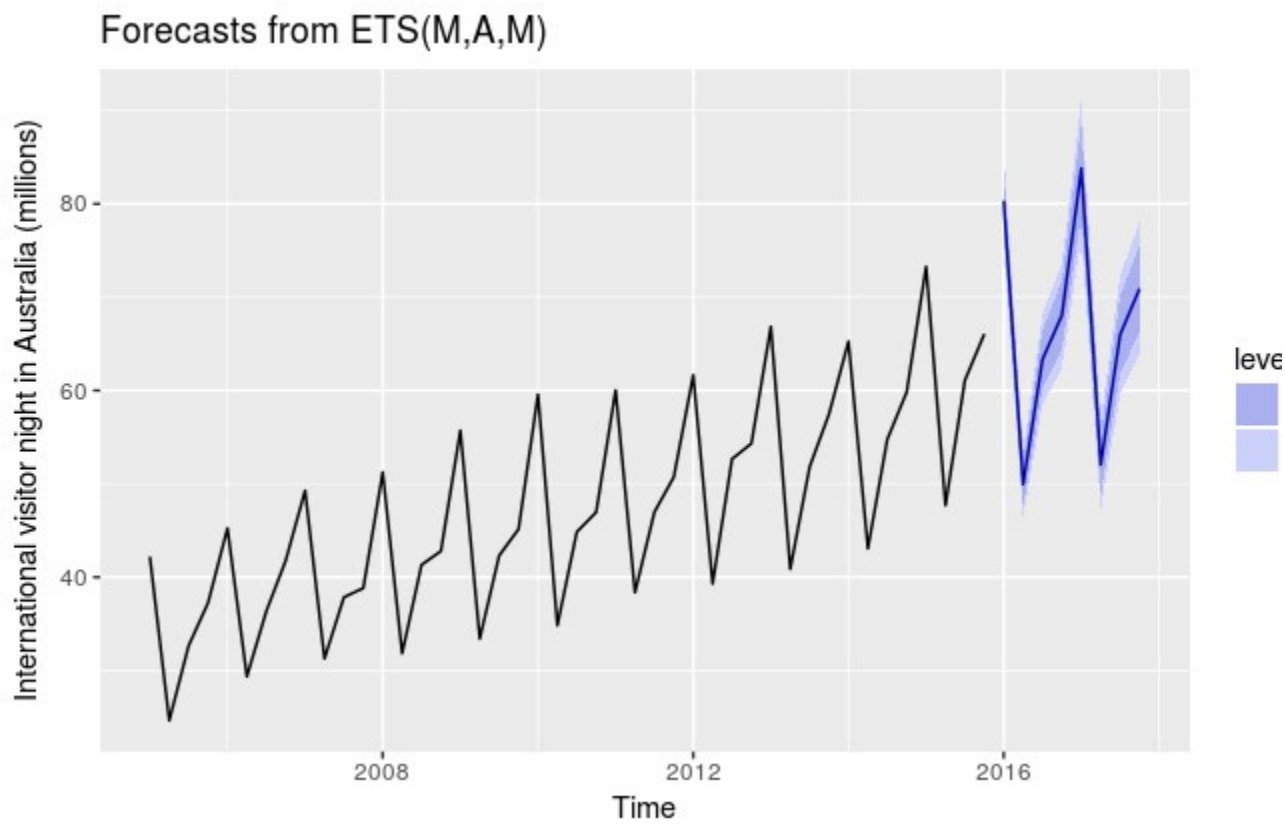


Figure 7.11: Forecasting international visitor nights in Australia using an ETS(M,A,M) model.

1. Consider the `pigs` series — the number of pigs slaughtered in Victoria each month.
    1. Use the `ses` function in R to find the optimal values of α and ℓ0, and generate forecasts for the next four months.
    2. Compute a 95% prediction interval for the first forecast using ˆy±1.96s where s is the standard deviation of the residuals. Compare your interval with the interval produced by R.

2. Write your own function to implement simple exponential smoothing. The function should take arguments `y` (the time series), `alpha` (the smoothing parameter α) and `level` (the initial level ℓ0). It should return the forecast of the next observation in the series. Does it give the same forecast as `ses`?

3. Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the `optim` function to find the optimal values of α and $\ell 0$. Do you get the same values as the `ses` function?

4. Combine your previous two functions to produce a function which both finds the optimal values of α and $\ell 0$, and produces a forecast of the next observation in the series.

5. Data set `books` contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days' sales for paperback and hardcover books.
    1. Plot the series and discuss the main features of the data.
    2. Use the `ses` function to forecast each series, and plot the forecasts.
    3. Compute the RMSE values for the training data in each case.
6.
    1. Now apply Holt's linear method to the `paperback` and `hardback` series and compute four-day forecasts in each case.
    2. Compare the RMSE measures of Holt's method for the two series to those of simple exponential smoothing in the previous question. (Remember that Holt's method is using one more parameter than SES.) Discuss the merits of the two forecasting methods for these data sets.
    3. Compare the forecasts for the two series using both methods. Which do you think is best?
    4. Calculate a 95% prediction interval for the first forecast for each series, using the RMSE values and assuming normal errors. Compare your intervals with those produced using `ses` and `holt`.

7. For this exercise, use the price of a dozen eggs in the United States from 1900–1993 (data set `eggs`). Experiment with the various options in the `holt()` function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to develop an intuition of what each argument is doing to the forecasts.

    [Hint: use `h=100` when calling `holt()` so you can clearly see the differences between the various options when plotting the forecasts.]

    Which model gives the best RMSE?

8. Recall your retail time series data (from Exercise 1 in Section 2.10).
    1. Why is multiplicative seasonality necessary for this series?
    2. Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.
    3. Compare the RMSE of the one-step forecasts from the two methods. Which do you prefer?
    4. Check that the residuals from the best method look like white noise.
    5. Now find the test set RMSE, while training the model to the end of 2010. Can you beat the seasonal naïve approach from Exercise 7 in Section 3.7?

9. For the same retail data, try an STL decomposition applied to the Box-Cox transformed series, followed by ETS on the seasonally adjusted data. How does that compare with your best previous forecasts on the test set?

10. For this exercise, use the quarterly UK passenger vehicle production data from 1977Q1 –2005Q1 (data set `ukcars`).
    1. Plot the data and describe the main features of the series.
    2. Decompose the series using STL and obtain the seasonally adjusted data.

3. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. (This can be done in one step using `stlf` with arguments `etsmodel="AAN", damped=TRUE`.
4. Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data (as before but with `damped=FALSE`).
5. Now use `ets()` to choose a seasonal model for the data.
6. Compare the RMSE of the ETS model with the RMSE of the models you obtained using STL decompositions. Which gives the better in-sample fits?
7. Compare the forecasts from the three approaches? Which seems most reasonable?
8. Check the residuals of your preferred model.

11. For this exercise, use the monthly Australian short-term overseas visitors data, May 1985–April 2005. (Data set: `visitors`.)
    1. Make a time plot of your data and describe the main features of the series.
    2. Split your data into a training set and a test set comprising the last two years of available data. Forecast the test set using Holt-Winters' multiplicative method.
    3. Why is multiplicative seasonality necessary here?
    4. Forecast the two-year test set using each of the following methods:
        1. an ETS model;
        2. an additive ETS model applied to a Box-Cox transformed series;
        3. a seasonal naive method;
        4. an STL decomposition applied to the Box-Cox transformed data followed by an ETS model applied to the seasonally adjusted (transformed) data.
    5. Which method gives the best forecasts? Does it pass the residual tests?
    6. Compare the same five methods using time series cross-validation with the `tsCV` function instead of using a training and test set. Do you come to the same conclusions?

12. Compare `ets`, `snaive` and `stlf` on the following six time series. For `stlf`, you might need to use a Box-Cox transformation. Use a test set of three years to decide what gives the best forecasts. `ausbeer`, `bricksq`, `dole`, `a10`, `h02`, `usmelec`.

13.

    1. Use `ets()` on some of these series:

       `bicoal`, `chicken`, `dole`, `usdeaths`, `lynx`, `ibmclose`, `eggs`.

       Does it always give good forecasts?

    2. Find an example where it does not work well. Can you figure out why?

7.9 Further reading

Two articles by Ev Gardner (Gardner Jr 1985; Gardner Jr 2006) provide a great overview of the history of exponential smoothing, and its many variations. A full book treatment of the subject providing the mathematical details is given by Hyndman, Koehler, et al. (2008b).

References

Gardner Jr, Everette S. 1985. "Exponential Smoothing: The State of the Art" 4 (1): 1–28.

Gardner Jr, Everette S. 2006. "Exponential Smoothing: The State of the Art — Part II." *Interantional Journal of Forecasting* 22: 637–66.

Hyndman, Rob J, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008b. *Forecasting with Exponential Smoothing: The State Space Approach*. Berlin: Springer-Verlag. www.exponentialsmoothing.net.

Chapter 8 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 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.

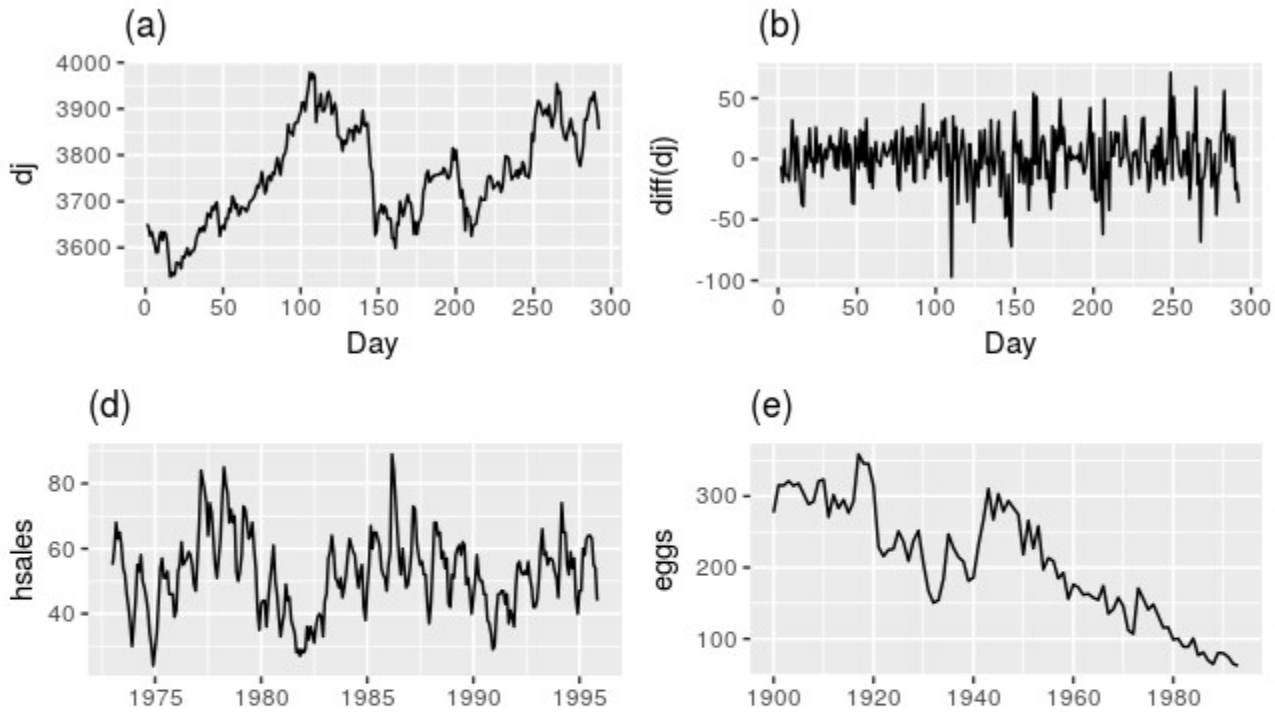
Before we introduce ARIMA models, we must first discuss the concept of stationarity and the technique of differencing time series.

8.1 Stationarity and differencing

A stationary time series is one whose properties do not depend on the time at which the series is observed.[14] 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.



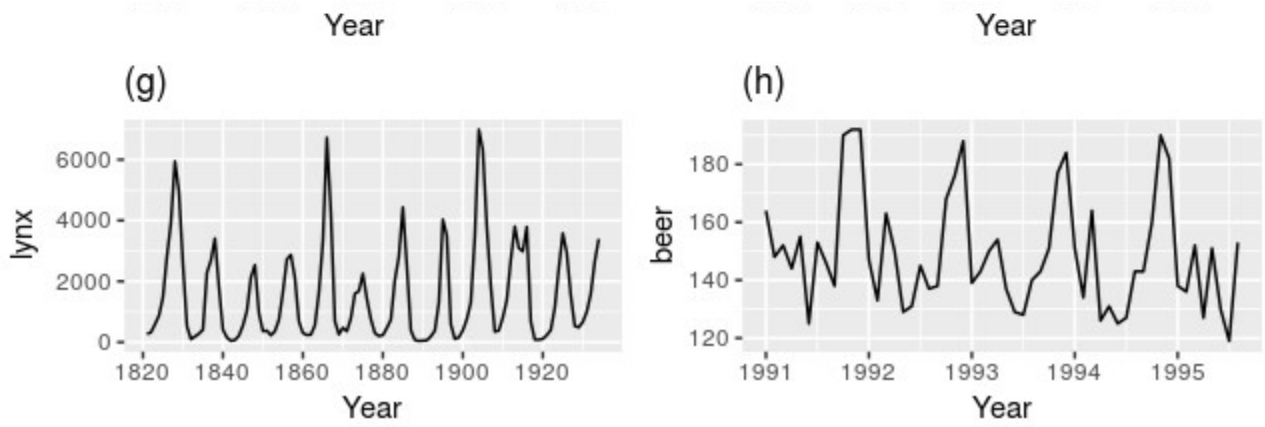


Figure 8.1: Which of these series are stationary? (a) Dow Jones index on 292 consecutive days; (b) Daily change in the Dow Jones index on 292 consecutive days; (c) Annual number of strikes in the US; (d) Monthly sales of new one-family houses sold in the US; (e) Annual price of a dozen eggs in the US (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 8.1. 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.

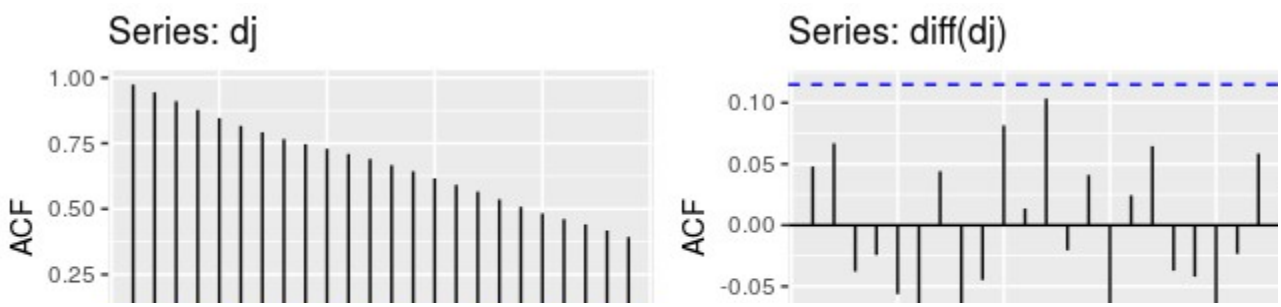
At first glance, the strong cycles in series (g) might appear to make it non-stationary. But these cycles are aperiodic — they are caused when the lynx population becomes too large for the available feed, so that they stop breeding and the population falls to very low numbers, then the regeneration of 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 8.1, 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 r1 is often large and positive.



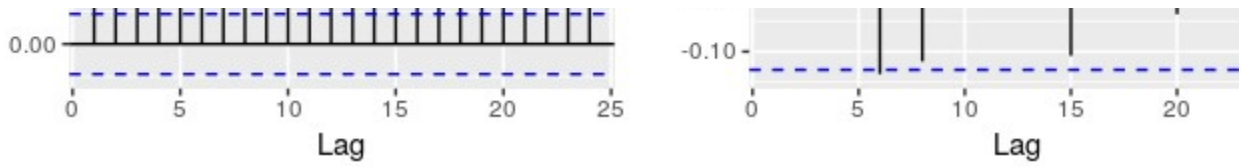


Figure 8.2: The ACF of the Dow-Jones index (left) and of the daily changes in the Dow-Jones index (right).

```
Box.test(diff(dj), lag=10, type="Ljung-Box")
#>
#>  Box-Ljung test
#>
#> data:  diff(dj)
#> X-squared = 10, df = 10, p-value = 0.2
```

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 Q∗ statistic has a *p*-value of 0.153 (for h=10). 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 y′t=yt−yt−1. The differenced series will have only T−1 values, since it is not possible to calculate a difference y′1 for the first observation.

When the differenced series is white noise, the model for the original series can be written as yt−yt−1=etoryt=yt−1+et. 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 yt−yt−1=c+etoryt=c+yt−1+et. The value of c is the average of the changes between consecutive observations. If c is positive, then the average change is an increase in the value of yt. Thus, yt will tend to drift upwards. However, if c is negative, yt will tend to drift downwards.

This is the model behind the drift method discussed in Section 3.1.

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: yt″

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 corresponding observation from the previous year. 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}+e_t. Forecasts from this model are equal to the last observation from the relevant season. That is, this model gives seasonal naïve forecasts.

```
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")
```

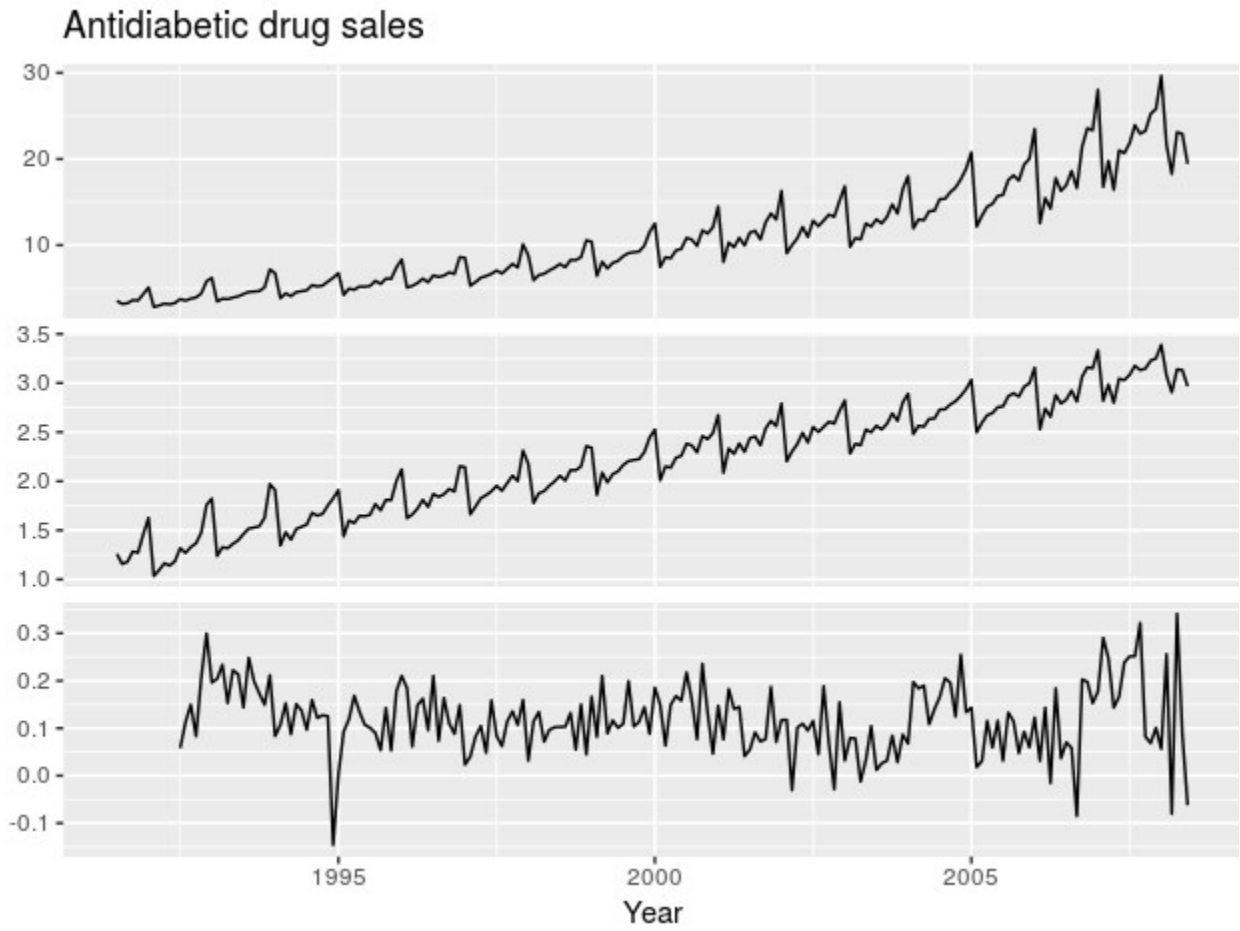


Figure 8.3: Logs and seasonal differences of the A10 (antidiabetic) sales data. The logarithms stabilize the variance, while the seasonal differences remove the seasonality and trend.

Figure 8.3 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 8.4. 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")
```

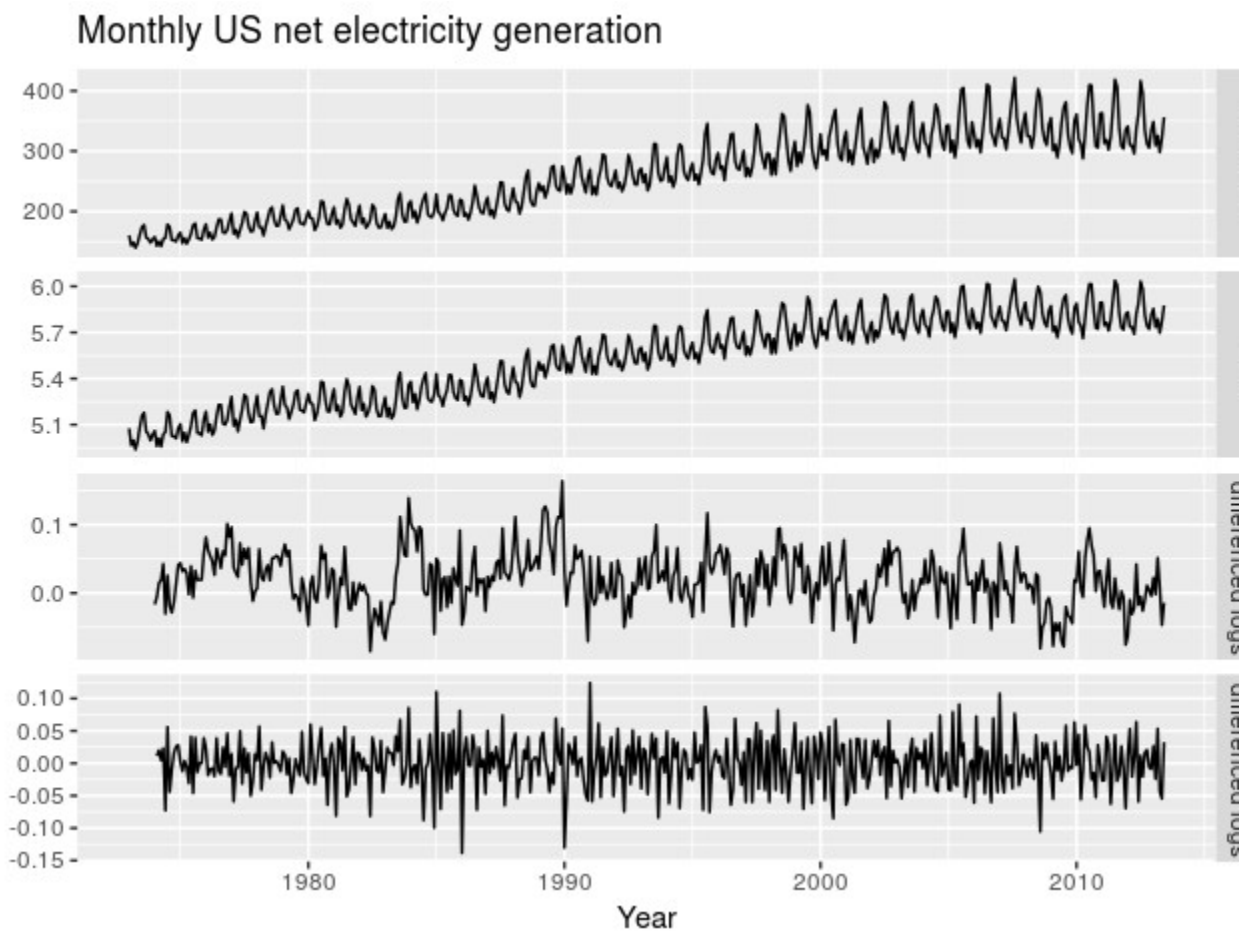


Figure 8.4: Top panel: US net electricity generation (billion kWh). Other panels show the same data after transforming and differencing.

There is a degree of subjectivity in selecting which differences to apply. The seasonally differenced data in Figure 8.3 do not show substantially different behaviour from the seasonally differenced data in Figure 8.4. 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 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 y'_t = y_t - y_{t-m} denotes a seasonally differenced series, then the twice-differenced series is

\begin{align*} 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{align*}

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: y'_t = \phi y_{t-1} + \beta_1 y'_{t-1} + \beta_2 y'_{t-2} + \cdots + \beta_k y'_{t-k}, where y'_t denotes the first-differenced series, y'_t=y_t-y_{t-1}. and k is the number of lags to include in the regression (often set to be about 3). If the original series, y_t, needs differencing, then the coefficient \hat\phi should be approximately zero. If y_t is already stationary, then \hat{\phi}<0. 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.

```
adf.test(x, alternative = "stationary")
```

In R, the default value of k is set to \lfloor(T - 1)^{1/3}\rfloor, where T is the length of the time series and \lfloor x\rfloor means the largest integer not greater than x.

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.

```
kpss.test(x)
```

A useful R function is `ndiffs()`, which uses these tests to determine the appropriate number of first differences required for a non-seasonal time series.

More complicated tests are required for seasonal differencing, and are beyond the scope of this book. A useful R function for determining whether seasonal differencing is required is `nsdiffs()`, which uses seasonal unit root tests to determine the appropriate number of seasonal differences required.

The following code can be used to find how to make a seasonal series stationary. The resulting series, stored as `xstar`, has been differenced appropriately.

```
ns <- nsdiffs(x)
if(ns > 0)
  xstar <- diff(x, lag=frequency(x), differences=ns)
else
  xstar <- x
nd <- ndiffs(xstar)
if(nd > 0)
  xstar <- diff(xstar, differences=nd)
```

8.2 Backshift notation

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**: $B(By_t) = B^2y_t = y_{t-2}$. 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: $y''_t = y_t - 2y_{t-1} + y_{t-2} = (1-2B+B^2)y_t = (1-B)^2y_t$. In general, a dth-order difference can be written as $(1-B)^d y_t$.

Backshift notation is very 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 $(1-B)(1-B^m)y_t = (1-B-B^m+B^{m+1})y_t = y_t - y_{t-1} - y_{t-m} + y_{t-m-1}$,

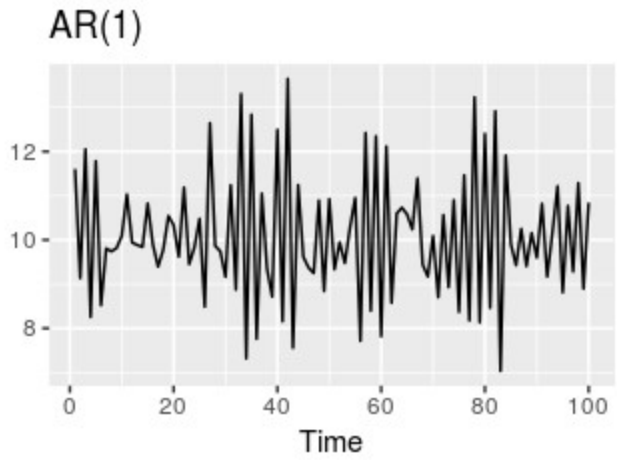
the same result we obtained earlier.

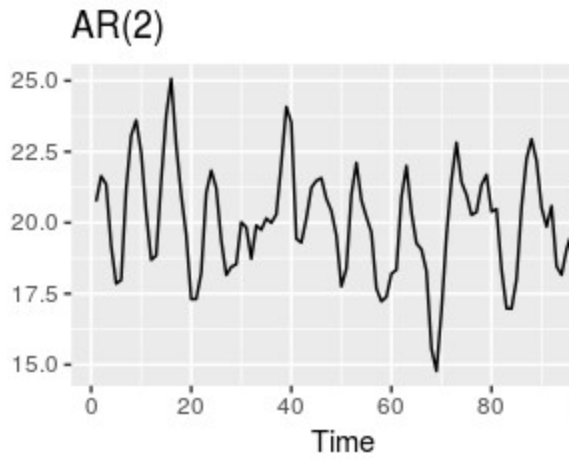
8.3 Autoregressive models

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 *auto*regression indicates that it is a regression of the variable against itself.

Thus, an autoregressive model of order p can be written as $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$, where e_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**.

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 AR(1) model and an AR(2) model. Changing the parameters $\phi_1, \ldots, \phi_p$ results in different time series patterns. The variance of the error term e_t will only change the scale of the series, not the patterns.





For an AR(1) model:

- when $\phi 1=0$, y_t is equivalent to white noise;
- when $\phi 1=1$ and $c=0$, y_t is equivalent to a random walk;
- when $\phi 1=1$ and $c\neq 0$, y_t is equivalent to a random walk with drift;
- when $\phi 1<0$, y_t tends to oscillate between positive and negative values;

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

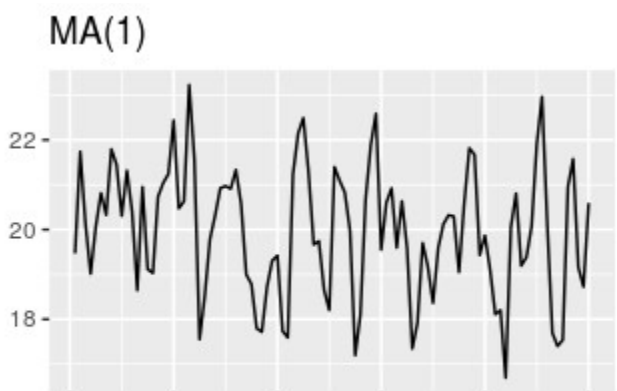
- For an AR(1) model: $-1<\phi 1<1$.
- For an AR(2) model: $-1<\phi 2<1$, $\phi 1+\phi 2<1$,

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

8.4 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. $y_t=c+e_t+\theta 1 e_{t-1}+\theta 2 e_{t-2}+\cdots+\theta q e_{t-q}$, where e_t is white noise. We refer to this as an **MA(q) model**. Of course, we do not *observe* the values of e_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* we discussed in Chapter 6. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.



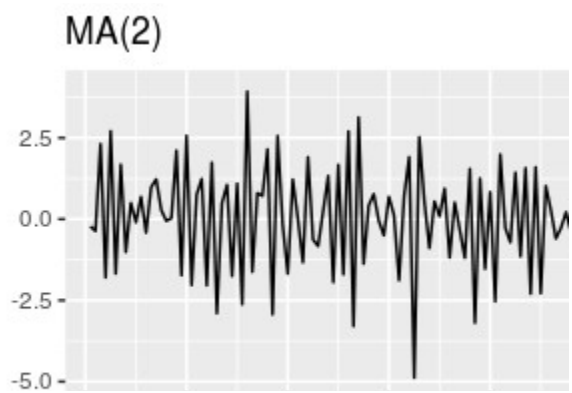


Figure 8.6 shows some data from an MA(1) model and an MA(2) model. Changing the parameters $\theta 1,\ldots,\theta q$ results in different time series patterns. As with autoregressive models, the variance of the error term et will only change the scale of the series, not the patterns.

It is possible to write any stationary AR(p) model as an MA(∞) model. For example, using repeated substitution, we can demonstrate this for an AR(1) model: $yt=\phi 1yt-1+et=\phi 1(\phi 1yt-2+et-1)+et=\phi 21yt-2+\phi 1et-1+et=\phi 31yt-3+\phi 21et-2+\phi 1et-1+etetc$.

Provided $-1<\phi 1<1$, the value of $\phi k1$ will get smaller as k gets larger. So eventually we obtain $yt=et+\phi 1et-1+\phi 21et-2+\phi 31et-3+\cdots$, 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(q) 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.

- 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.

8.5 Non-seasonal ARIMA models

where y′t is the differenced series (it may have been differenced more than once). The "predictors" on the right hand side include both lagged values of yt and lagged errors. We call this an **ARIMA (p,d,q) model**, where

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.

Selecting appropriate values for p, d and q 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

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

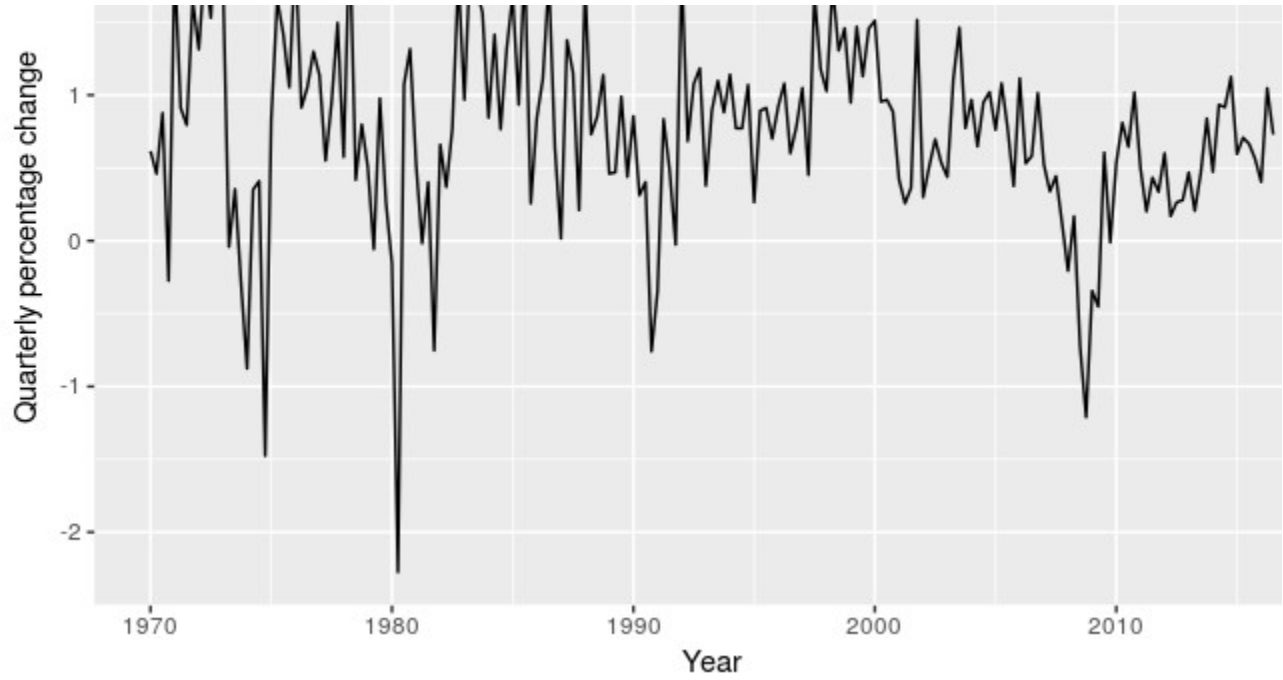


Figure 8.7: Quarterly percentage change in US consumption expenditure.

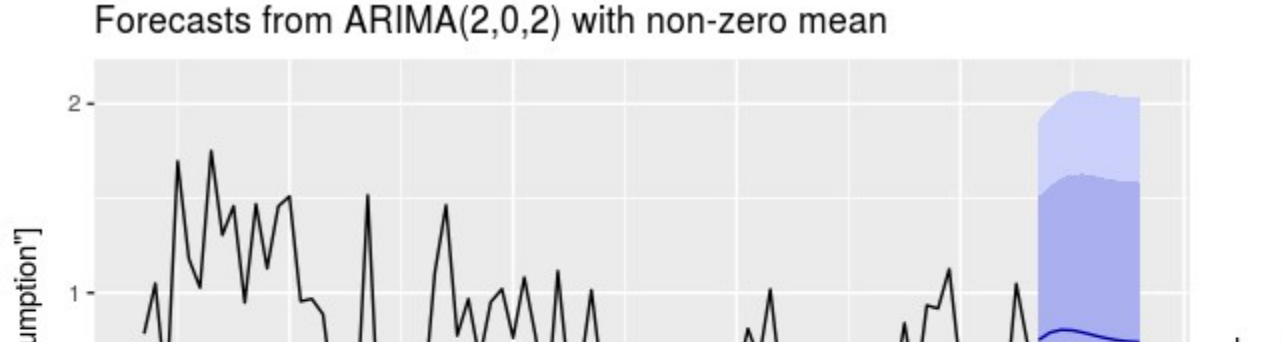
Figure 8.7 shows quarterly percentage changes in US consumption expenditure. Although it is a quarterly series, there does not appear to be a seasonal pattern, so we will fit a non-seasonal ARIMA model.

The following R code was used to select a model automatically. (We use the argument `seasonal=FALSE` for now, but we will consider seasonal ARIMA models in Section 8.9)

```
(fit <- auto.arima(uschange[,"Consumption"], seasonal=FALSE))
#> Series: uschange[, "Consumption"]
#> ARIMA(2,0,2)                    with non-zero mean
#>
#> Coefficients:
#>          ar1     ar2      ma1    ma2   mean
#>       1.391  -0.581  -1.180  0.558  0.746
#> s.e.  0.255   0.208   0.238  0.140  0.084
#>
#> sigma^2 estimated as 0.351:  log likelihood=-165
#> AIC=342   AICc=343   BIC=362
```

This is an ARIMA(2,0,2) model: $y_t = c + 1.391y_{t-1} - 0.581y_{t-2} - 1.180e_{t-1} + 0.558e_{t-2} + e_t$, where $c = 0.746 \times (1 - 1.391 + 0.581) = 0.142$ and e_t is white noise with a standard deviation of $0.593 = \sqrt{0.351}$. Forecasts from the model are shown in Figure 8.8.

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



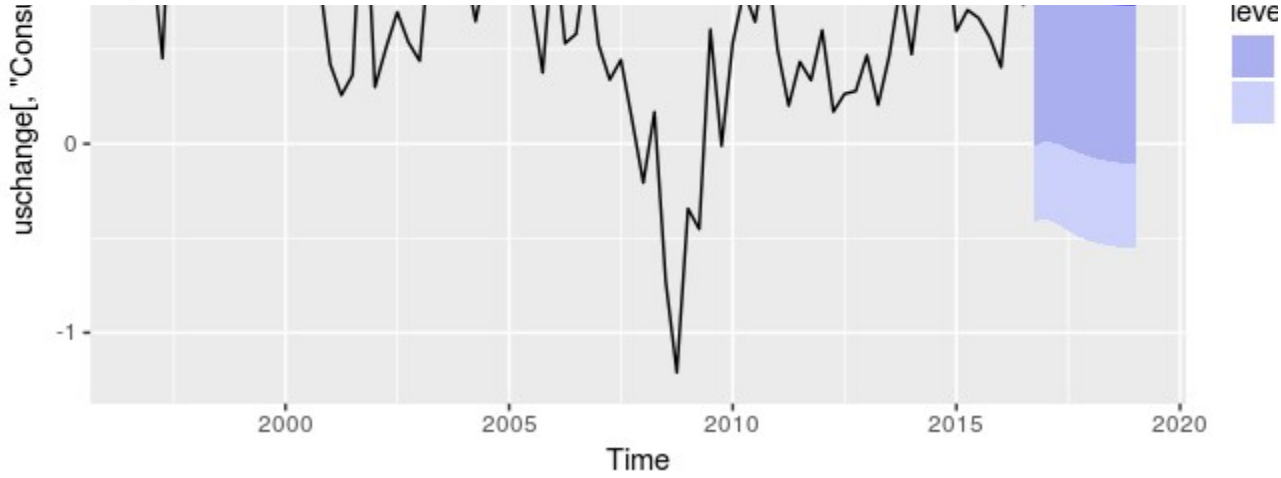


Figure 8.8: Forecasts of quarterly percentage changes in US consumption expenditure.

Understanding ARIMA models

The `auto.arima()` function is very useful, but anything automated can be a little dangerous, and it is worth understanding something of the behaviour of the models even when you rely 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 c=0 and d=0, the long-term forecasts will go to zero.
- If c=0 and d=1, the long-term forecasts will go to a non-zero constant.
- If c=0 and d=2, the long-term forecasts will follow a straight line.
- If c≠0 and d=0, the long-term forecasts will go to the mean of the data.
- If c≠0 and d=1, the long-term forecasts will follow a straight line.
- If c≠0 and d=2, the long-term forecasts will follow a quadratic trend.

The value of d also has an effect on the prediction intervals — the higher the value of d, the more rapidly the prediction intervals increase in size. For d=0, the long-term forecast standard deviation will go to the standard deviation of the historical data, so the prediction intervals will all be essentially the same.

This behaviour is seen in Figure 8.8 where d=0 and c≠0. 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 p is important if the data show cycles. To obtain cyclic forecasts, it is necessary to have p≥2, along with some additional conditions on the parameters. For an AR(2) model, cyclic behaviour occurs if $\phi_2 1+4\phi_2<0$. In that case, the average period of the cycles is[15] $2\pi \arccos(-\phi_1(1-\phi_2)/(4\phi_2))$.

ACF and PACF plots

It is usually not possible to tell, simply from a time plot, what values of p and q 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 p and q.

Recall that an ACF plot shows the autocorrelations which measure the relationship between y_t and y_{t-k} for different values of k. Now if y_t and y_{t-1} are correlated, then y_{t-1} and y_{t-2} must also be

correlated. However, then yt and yt−2 might be correlated, simply because they are both connected to yt−1, rather than because of any new information contained in yt−2 that could be used in forecasting yt.

To overcome this problem, we can use **partial autocorrelations**. These measure the relationship between yt and yt−k after removing the effects of lags 1,2,3,…,k−1. 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, αk, the kth partial autocorrelation coefficient, is equal to the estimate of ϕk in an AR(k) model. In practice, there are more efficient algorithms for computing αk than fitting all of these autoregressions, but they give the same results.

Figures 8.9 and 8.10 shows the ACF and PACF plots for the US consumption data shown in Figure 8.7. The partial autocorrelations have the same critical values of ±1.96/√T as for ordinary autocorrelations, and these are typically shown on the plot as in Figure 8.9.

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

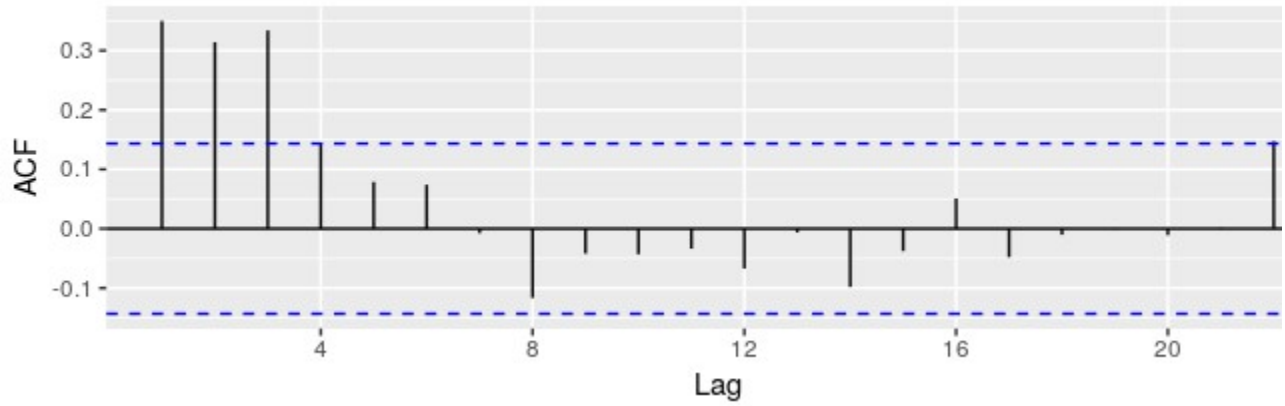


Figure 8.9: ACF of quarterly percentage change in US consumption. A convenient way to produce a time plot, ACF plot and PACF plot in one command is to use the `ggtsdisplay` function in R.

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

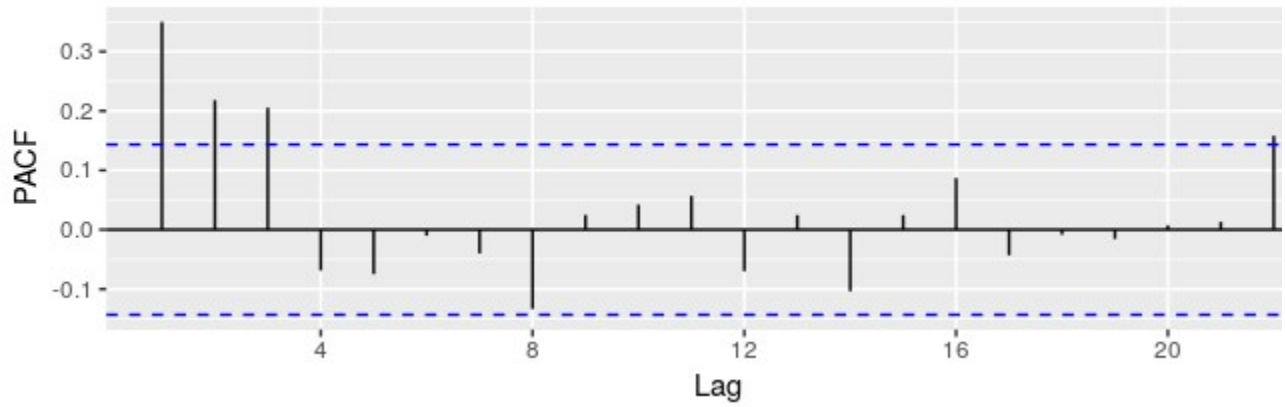


Figure 8.10: PACF of quarterly percentage change in US consumption.

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

The data may follow an ARIMA(p,d,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 p in the PACF, but none beyond lag p.

The data may follow an ARIMA(0,d,q) 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 q in the ACF, but none beyond lag q.

In Figure 8.9, 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
#>       0.227  0.160  0.203  0.745
#> s.e.  0.071  0.072  0.071  0.103
#>
#> sigma^2 estimated as 0.349:  log likelihood=-165
#> AIC=340   AICc=341   BIC=356
```

This model is actually slightly better than the model identified by `auto.arima` (with an AICc value of 340.67 compared to 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
#>       0.227  0.160  0.203  0.745
#> s.e.  0.071  0.072  0.071  0.103
#>
#> sigma^2 estimated as 0.349:  log likelihood=-165
#> AIC=340   AICc=341   BIC=356
```

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.

8.6 Estimation and order selection

Maximum likelihood estimation

Once the model order has been identified (i.e., the values of p, d and q), we need to estimate the parameters c, $\phi 1,\ldots,\phi p$, $\theta 1,\ldots,\theta q$. 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 $T\sum t=1e2t$. (For the regression models considered in Chapter 5, 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 p, d and q, R will try to maximize the log likelihood when finding parameter estimates.

8.7 ARIMA modelling in R

How does `auto.arima()` work?

The `auto.arima()` function in R uses a variation of the Hyndman-Khandakar algorithm (Hyndman and Khandakar 2008), 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 $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of p and q are then chosen by minimizing the AICc after differencing the data d times. Rather than considering every possible combination of p and q, the algorithm uses a stepwise search to traverse the model space.
    1. Four initial models are fitted:
        - ARIMA(0,d,0),
        - ARIMA(2,d,2),
        - ARIMA(1,d,0),
        - ARIMA(0,d,1). A constant is included unless d=2. If $d \leq 1$, an additional model
        - ARIMA(0,d,0) 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 p and/or q from the current model by ±1;
        - include/exclude c 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.

The arguments to `auto.arima()` provide for many variations on the algorithm. What is described here is the default behaviour.

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 argument `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 c unless d=0, 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.

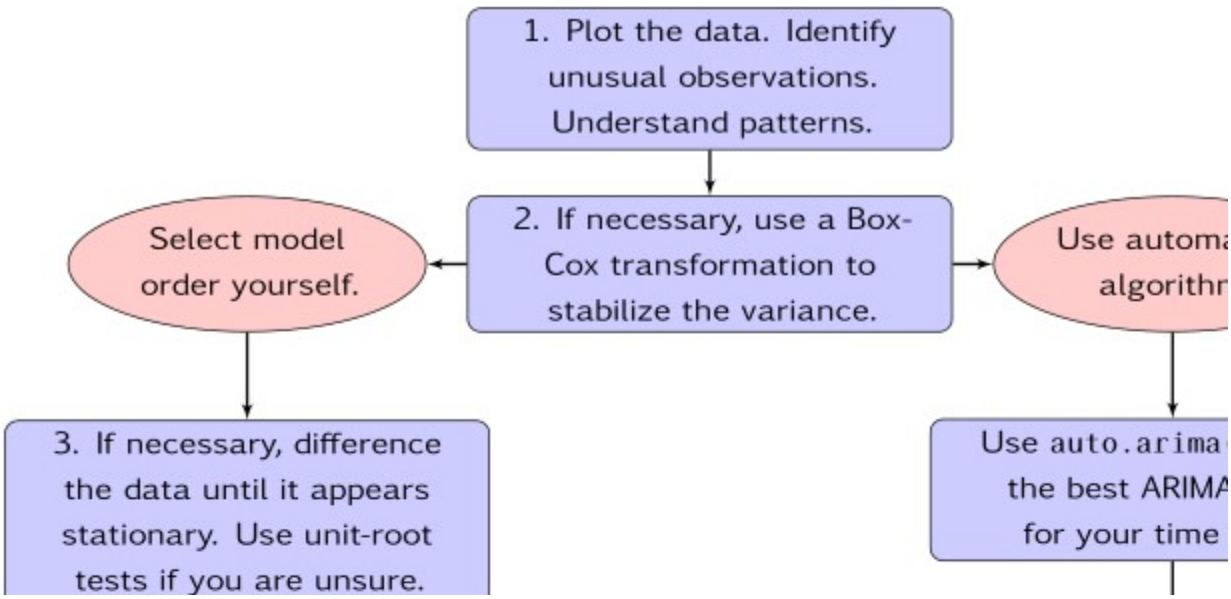
Modelling procedure

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 unusual observations.
2. If necessary, transform the data (using a 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(p,d,0) or ARIMA(0,d,q) 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 steps 3–5. So even if you use it, you will still need to take care of the other steps yourself.

The process is summarised in Figure 8.11.



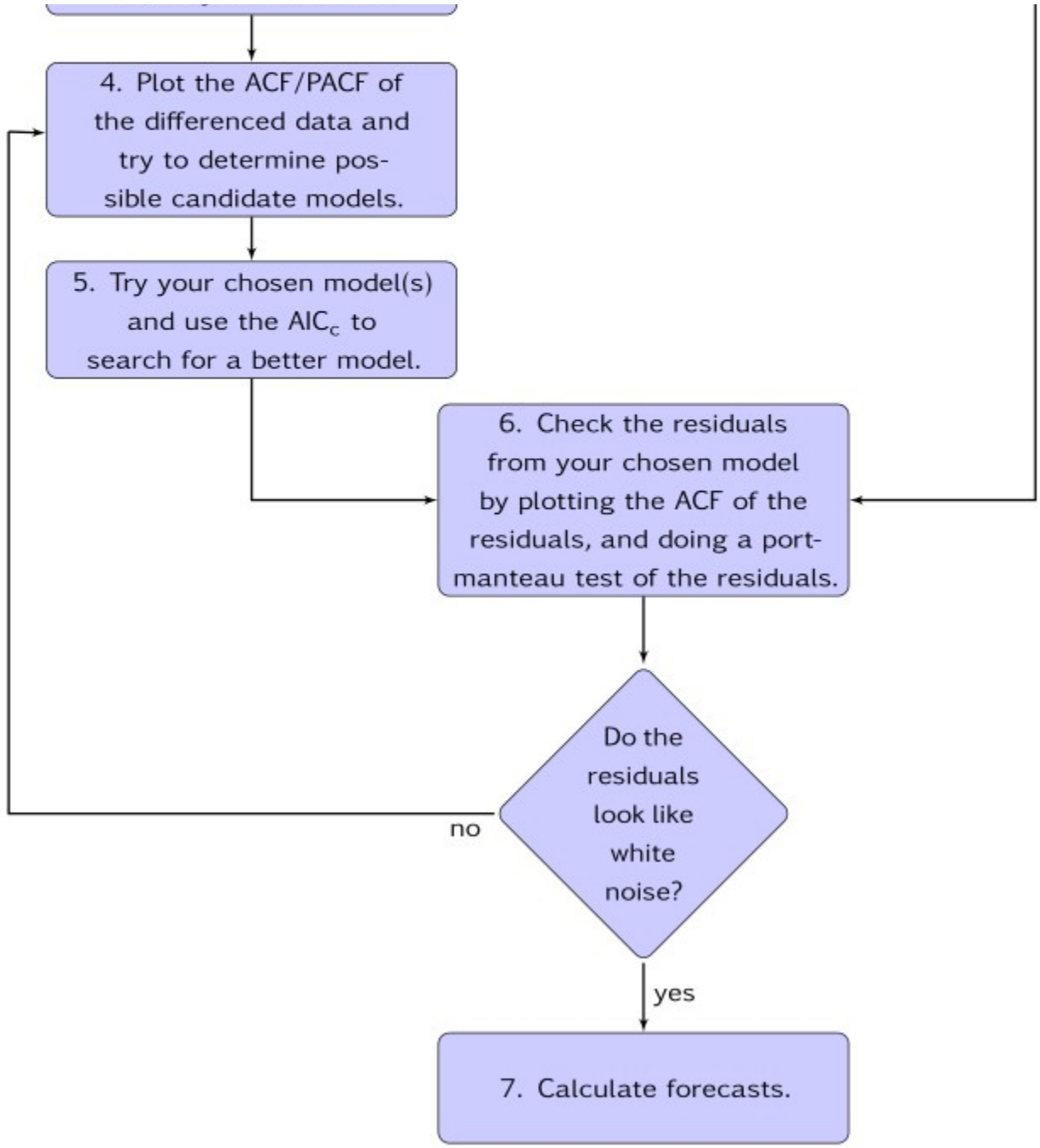


Figure 8.11: General process for forecasting using an ARIMA model.

Example: Seasonally adjusted electrical equipment orders

We will apply this procedure to the seasonally adjusted electrical equipment orders data shown in Figure 8.12.

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

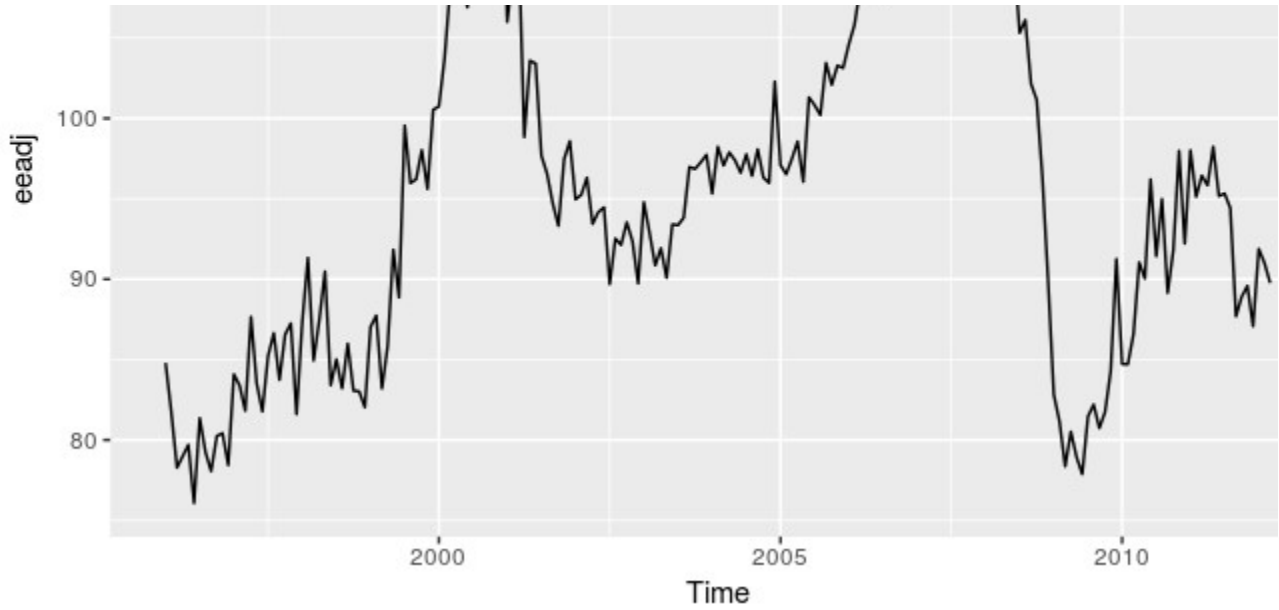


Figure 8.12: Seasonally adjusted electrical equipment orders index in the Euro area.

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

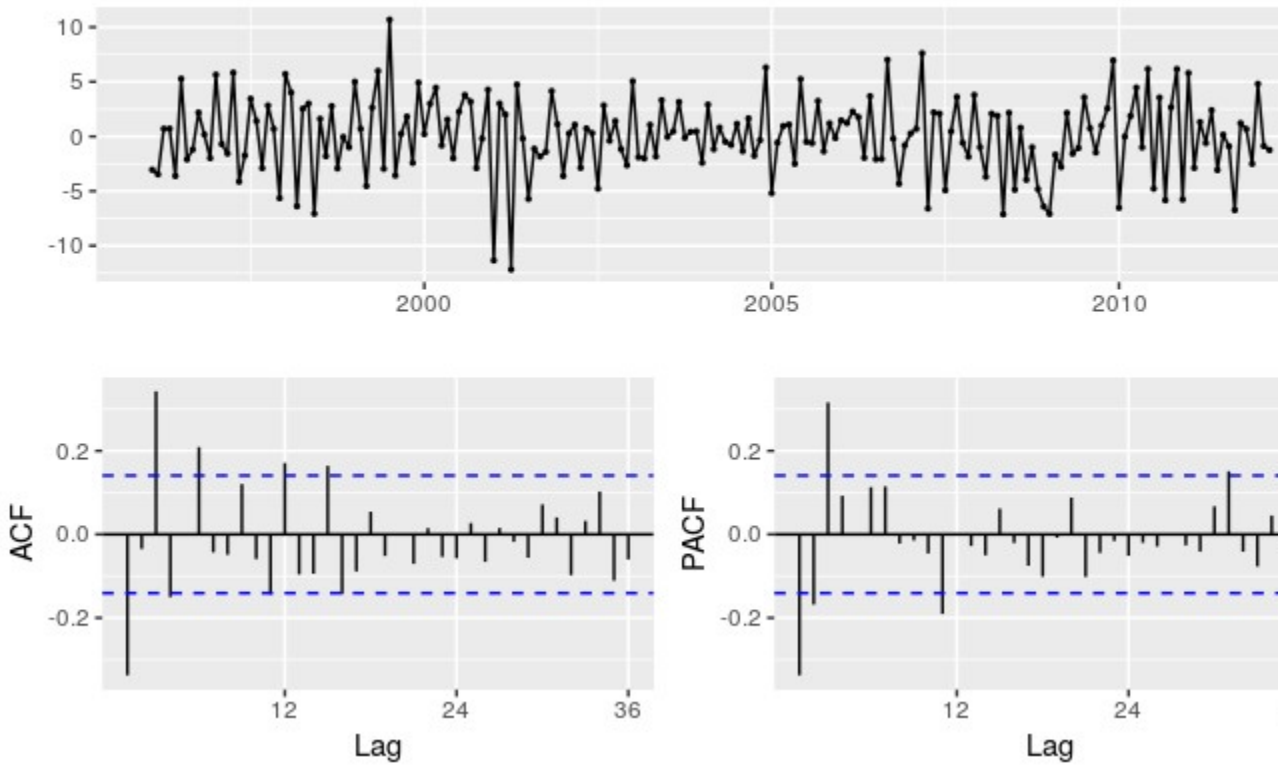


Figure 8.13: Time plot and ACF and PACF plots for the differenced seasonally adjusted electrical equipment data.

1. The time plot shows some sudden changes, particularly the big drop in 2008/2009. These changes are due to the global economic environment. Otherwise there is nothing unusual about the time plot and there appears 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.

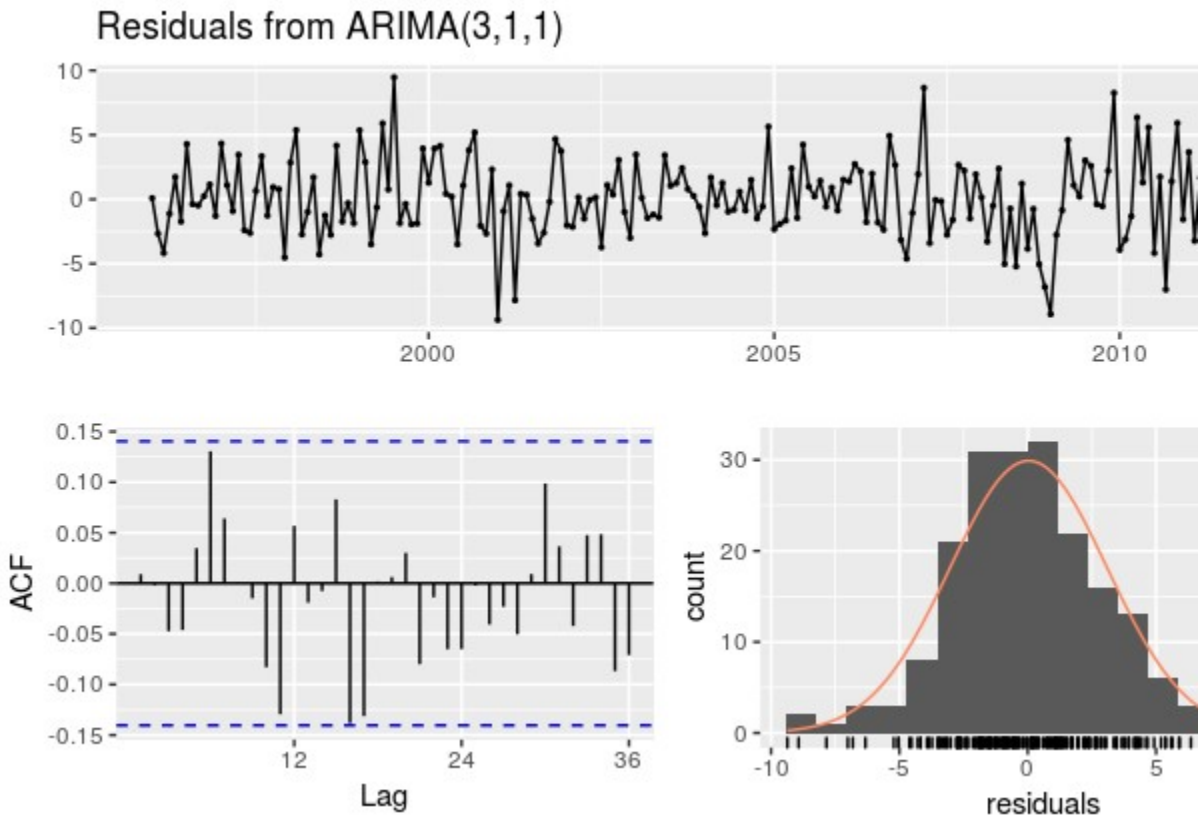
3. The data are clearly non-stationary, as the series wanders up and down for long periods. Consequently, we will take a first difference of the data. The differenced data are shown in Figure 8.13. These look stationary, and so we will not consider further differences.
4. The PACF shown in Figure 8.13 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
#>       0.004  0.092  0.370  -0.392
#> s.e.  0.220  0.098  0.067   0.243
#>
#> sigma^2 estimated as 9.58:  log likelihood=-493
#> AIC=995   AICc=996   BIC=1012
#>
#> Training set error measures:
#>                     ME RMSE  MAE      MPE MAPE  MASE    ACF1
#> Training set 0.0329 3.05 2.36 -0.00647 2.48 0.288 0.00898
```

6. 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 portmanteau 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* = 20, df = 20, p-value = 0.2
#>
#> Model df: 4.   Total lags used: 24
```

7. Forecasts from the chosen model are shown in Figure 8.14.

```
autoplot(forecast(fit))
```

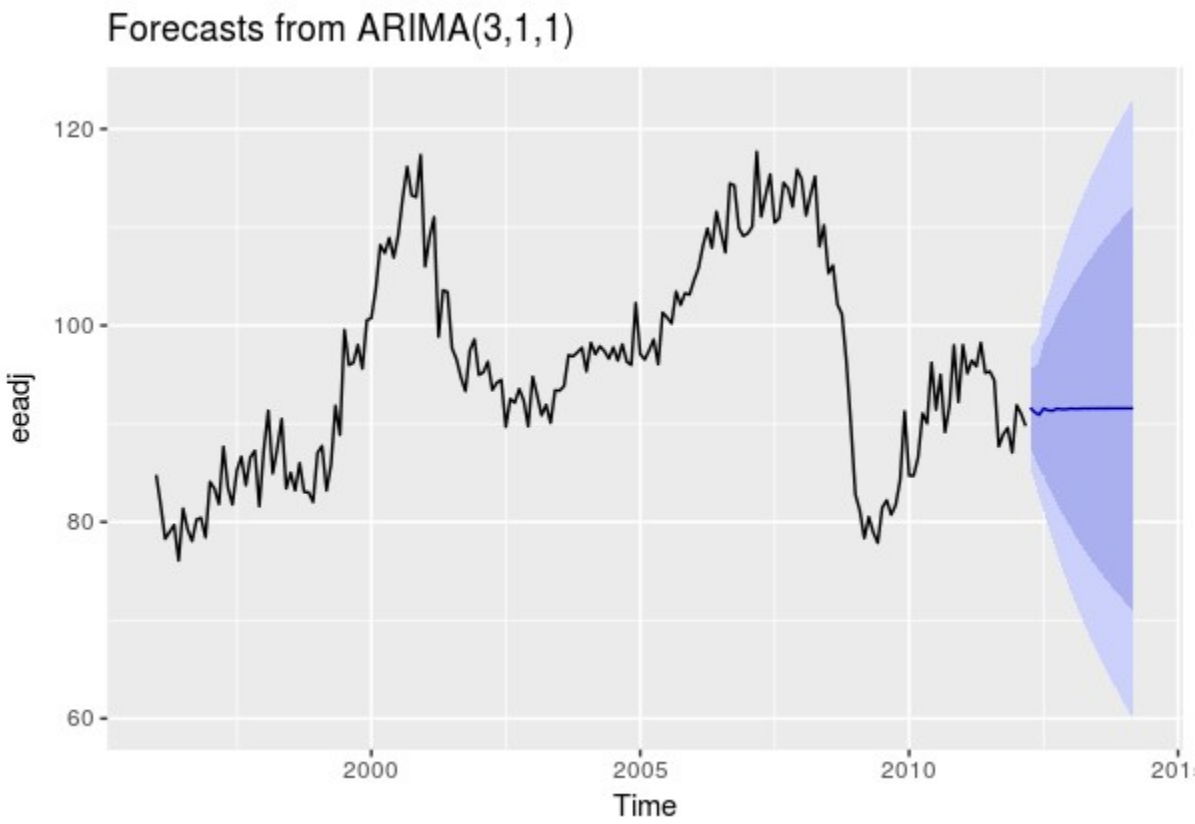


Figure 8.14: Forecasts for the seasonally adjusted electrical orders index.

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(3,1,1) model if we had set `approximation=FALSE`.

Understanding constants in R

A non-seasonal ARIMA model can be written as $(1-\phi_1 B-\cdots-\phi_p B^p)(1-B)d_{y_t}=c+(1+\theta_1 B+\cdots+\theta_q B^q)e_t$, or equivalently as $(1-\phi_1 B-\cdots-\phi_p B^p)(1-B)^d(y_t-\mu t^d/d!)=(1+\theta_1 B+\cdots+\theta_q B^q)e_t$,

where $c=\mu(1-\phi_1-\cdots-\phi_p)$ and μ is the mean of $(1-B)^d y_t$. R uses the parametrization of equation (8.5).

Thus, the inclusion of a constant in a non-stationary ARIMA model is equivalent to inducing a polynomial trend of order d in the forecast function. (If the constant is omitted, the forecast function includes a polynomial trend of order d−1.) When d=0, we have the special case that μ is the mean of y_t.

By default, the `Arima()` function sets $c=\mu=0$ when $d>0$ and provides an estimate of μ when $d=0$. 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 $p+q>0$.

The argument `include.mean` only has an effect when $d=0$ and is `TRUE` by default. Setting `include.mean=FALSE` will force $\mu=c=0$.

The argument `include.drift` allows $\mu\neq 0$ when $d=1$. For $d>1$, no constant is allowed as a quadratic or higher order trend is particularly dangerous when forecasting. The parameter μ is called the "drift" in the R output when $d=1$.

There is also an argument `include.constant` which, if `TRUE`, will set `include.mean=TRUE` if $d=0$ and `include.drift=TRUE` when $d=1$. 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 $d=0$ or $d=1$, a constant will be included if it improves the AICc value; for $d>1$ the constant is always omitted. If `allowdrift=FALSE` is specified, then the constant is only allowed when $d=0$.

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 y_t is on the left hand side and all other terms are on the right.
2. Rewrite the equation by replacing t with $T+h$.
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 $h=1$, these steps are then repeated for $h=2,3,\ldots$ until all forecasts have been calculated.

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: $(1-\hat{}\phi 1B-\hat{}\phi 2B2-\hat{}\phi 3B3)(1-B)yt=(1+\hat{}\theta 1B)et$, where , , and . Then we expand the left hand side to obtain $[1-(1+\hat{}\phi 1)B+(\hat{}\phi 1-\hat{}\phi 2)B2+(\hat{}\phi 2-\hat{}\phi 3)B3+\hat{}\phi 3B4]yt=(1+\hat{}\theta 1B)et$, and applying the backshift operator gives $yt-(1+\hat{}\phi 1)yt-1+(\hat{}\phi 1-\hat{}\phi 2)yt-2+(\hat{}\phi 2-\hat{}\phi 3)yt-3+\hat{}\phi 3yt-4=et+\hat{}\theta 1et-1$. Finally, we move all terms other than to the right hand side: $yt=(1+\hat{}\phi 1)yt-1-(\hat{}\phi 1-\hat{}\phi 2)yt-2-(\hat{}\phi 2-\hat{}\phi 3)yt-3-\hat{}\phi 3yt-4+et+\hat{}\theta 1et-1$.

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.

8.9 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 **(p,d,q)** **(P,D,Q)m**

↑ ↑

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

where m= 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)4 model (without a constant) is for quarterly data (m=4), and can be written as $(1-\phi 1B)\ (1-\Phi 1B4)(1-B)(1-B4)yt=(1+\theta 1B)\ (1+\Theta 1B4)et$.

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, …).

Similarly, an ARIMA(0,0,0)(1,0,0)12 model will show:

- exponential decay in the seasonal lags of the ACF;
- a single significant spike at lag 12 in the PACF.

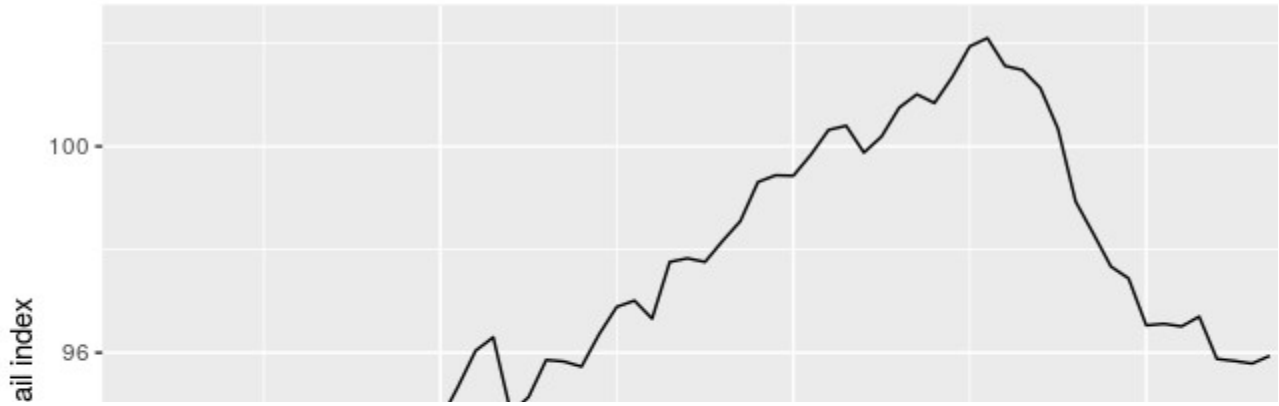
In considering the appropriate seasonal orders for an ARIMA model, restrict attention to the seasonal lags.

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 8.15.

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



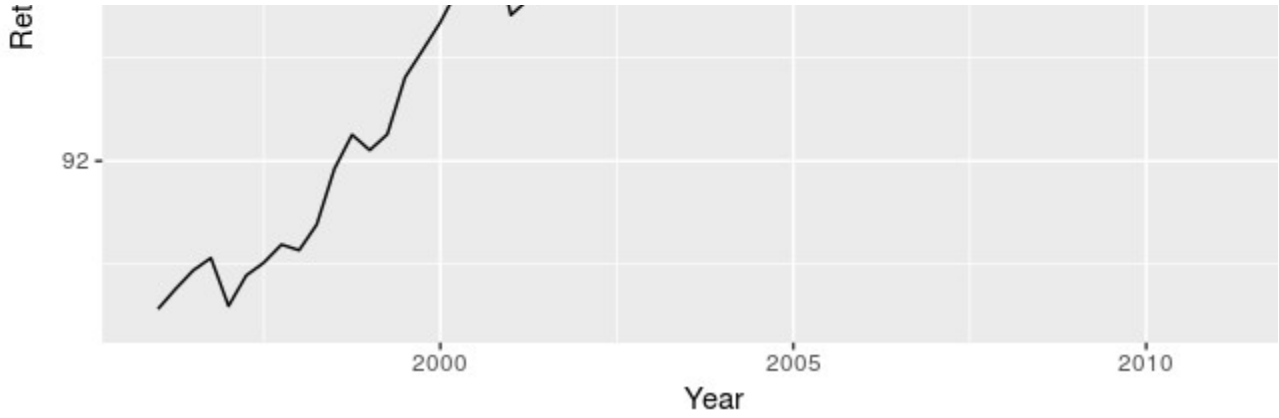


Figure 8.15: Quarterly retail trade index in the Euro area (17 countries), 1996–2011, covering wholesale and retail trade, and the repair of motor vehicles and motorcycles. (Index: 2005 = 100).

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 8.16. These also appear to be non-stationary, so we take an additional first difference, shown in Figure 8.17.

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

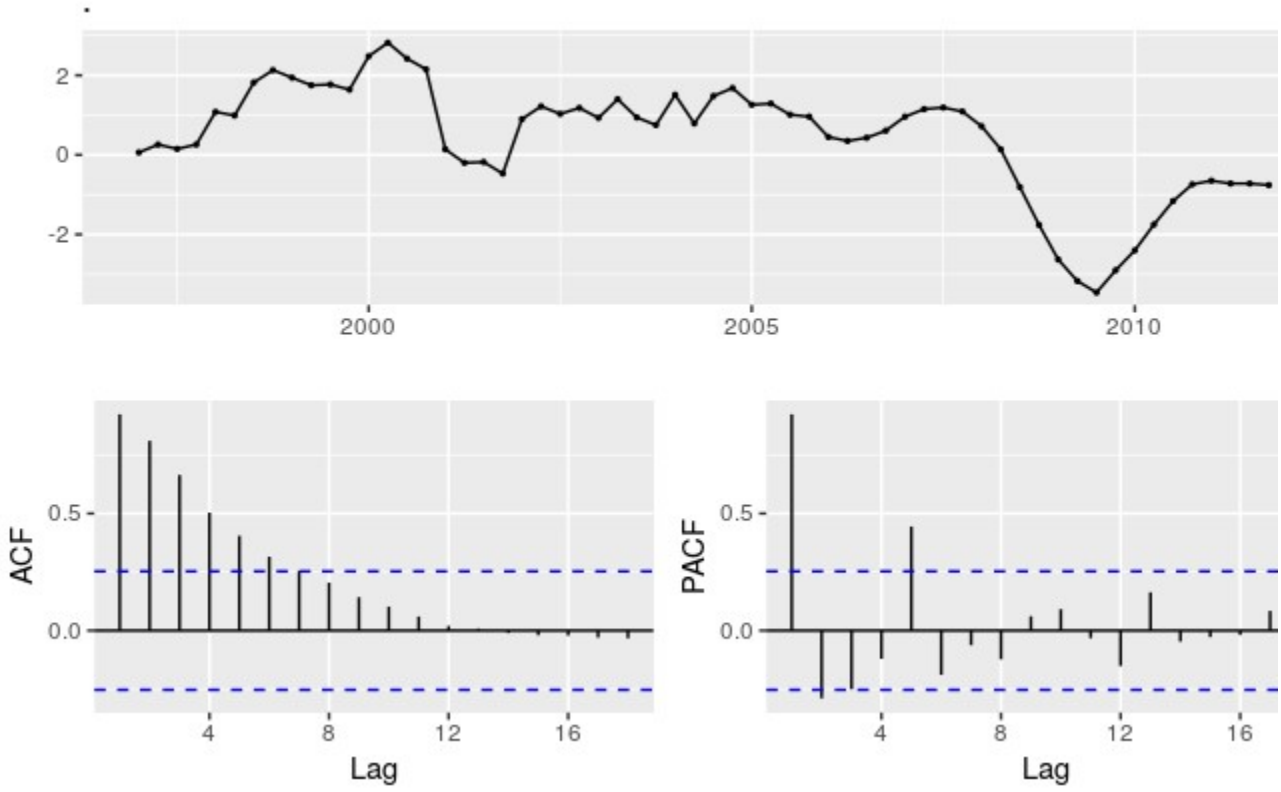
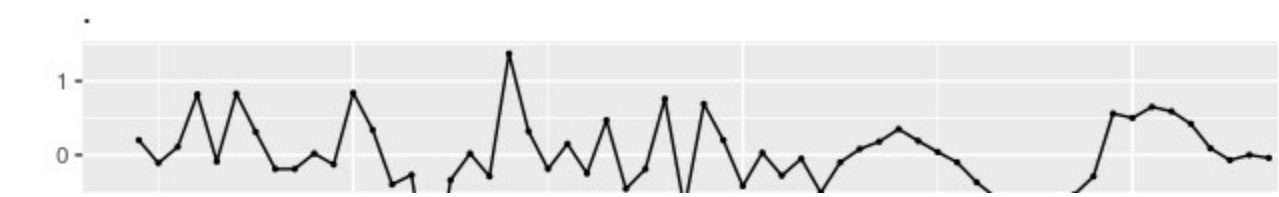


Figure 8.16: Seasonally differenced European retail trade index.

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



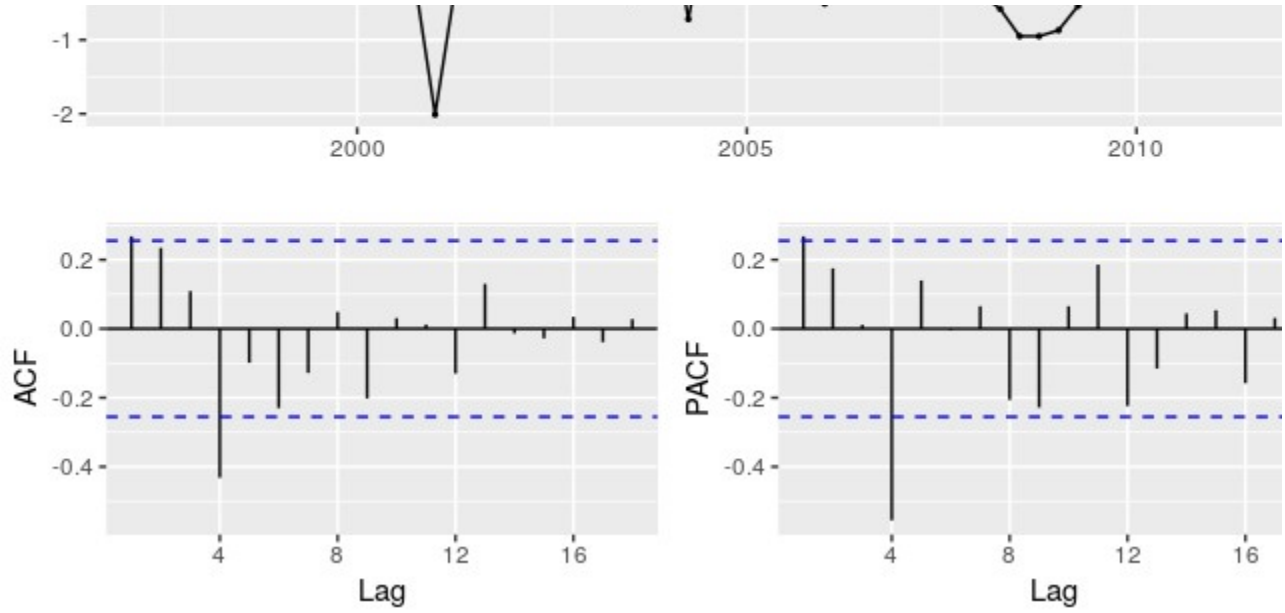
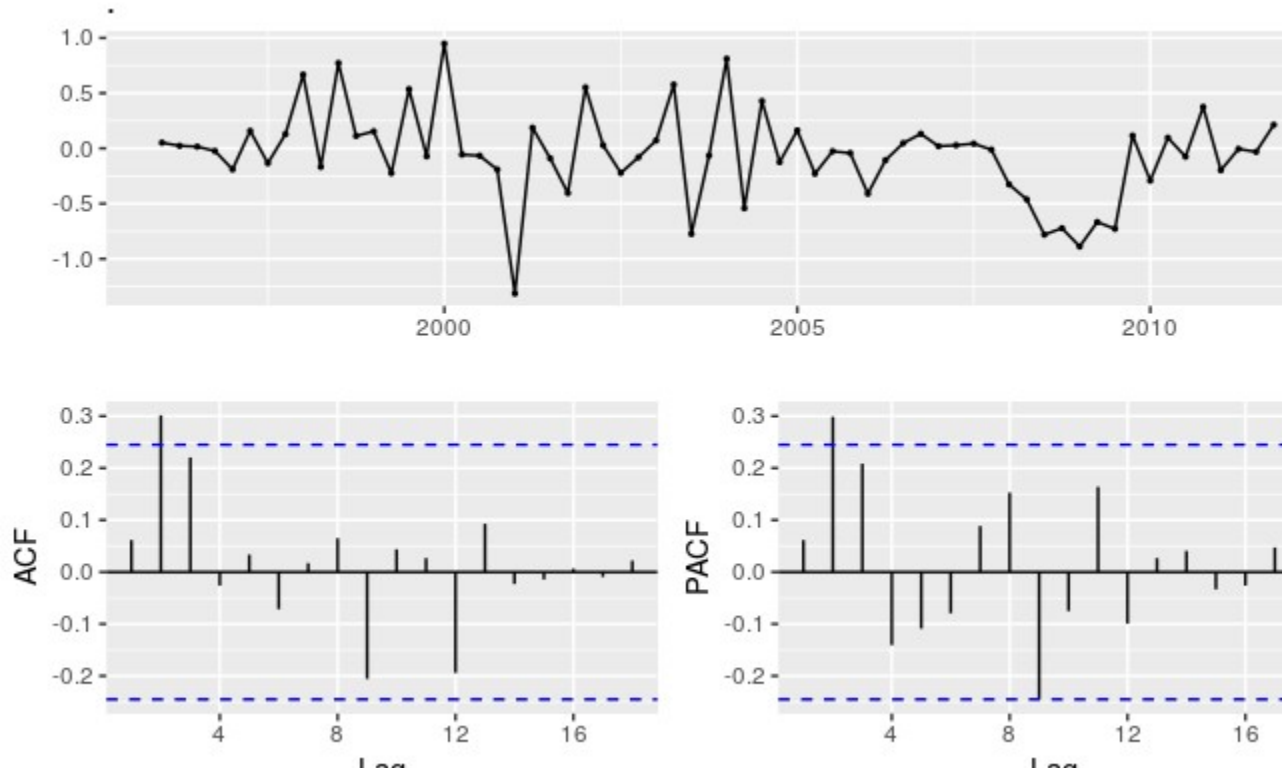


Figure 8.17: Double differenced European retail trade index.

Our aim now is to find an appropriate ARIMA model based on the ACF and PACF shown in Figure 8.17. The significant spike at lag 1 in the ACF suggests a non-seasonal MA(1) component, and the significant spike at lag 4 in the ACF suggests a seasonal MA(1) component. Consequently, we begin with an ARIMA(0,1,1)(0,1,1)4 model, indicating a first and seasonal difference, and non-seasonal and seasonal MA(1) components. The residuals for the fitted model are shown in Figure 8.18. (By analogous logic applied to the PACF, we could also have started with an ARIMA(1,1,0)(1,1,0)4 model.)

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



Lag Lag

Figure 8.18: Residuals from the fitted ARIMA(0,1,3)(0,1,1)4 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)4 model is 74.36, while that for the ARIMA(0,1,3)(0,1,1)4 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)4 model. Its residuals are plotted in Figure 8.19. 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)
```

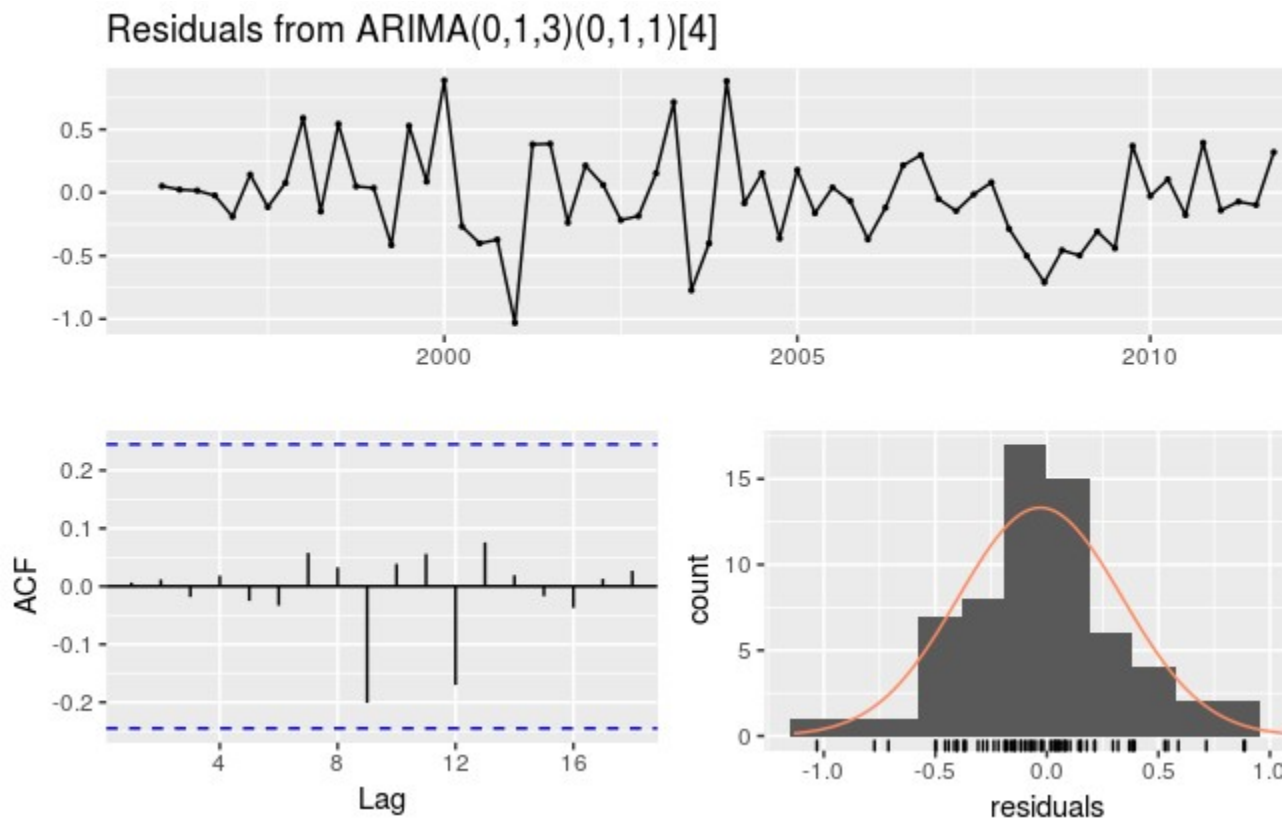


Figure 8.19: Residuals from the fitted ARIMA(0,1,3)(0,1,1)4 model for the European retail trade index data.

```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from ARIMA(0,1,3)(0,1,1)[4]
#> Q* = 0.5, df = 4, p-value = 1
#>
#> Model df: 4.   Total lags used: 8
```

Thus, we now have a seasonal 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 8.20. 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
```

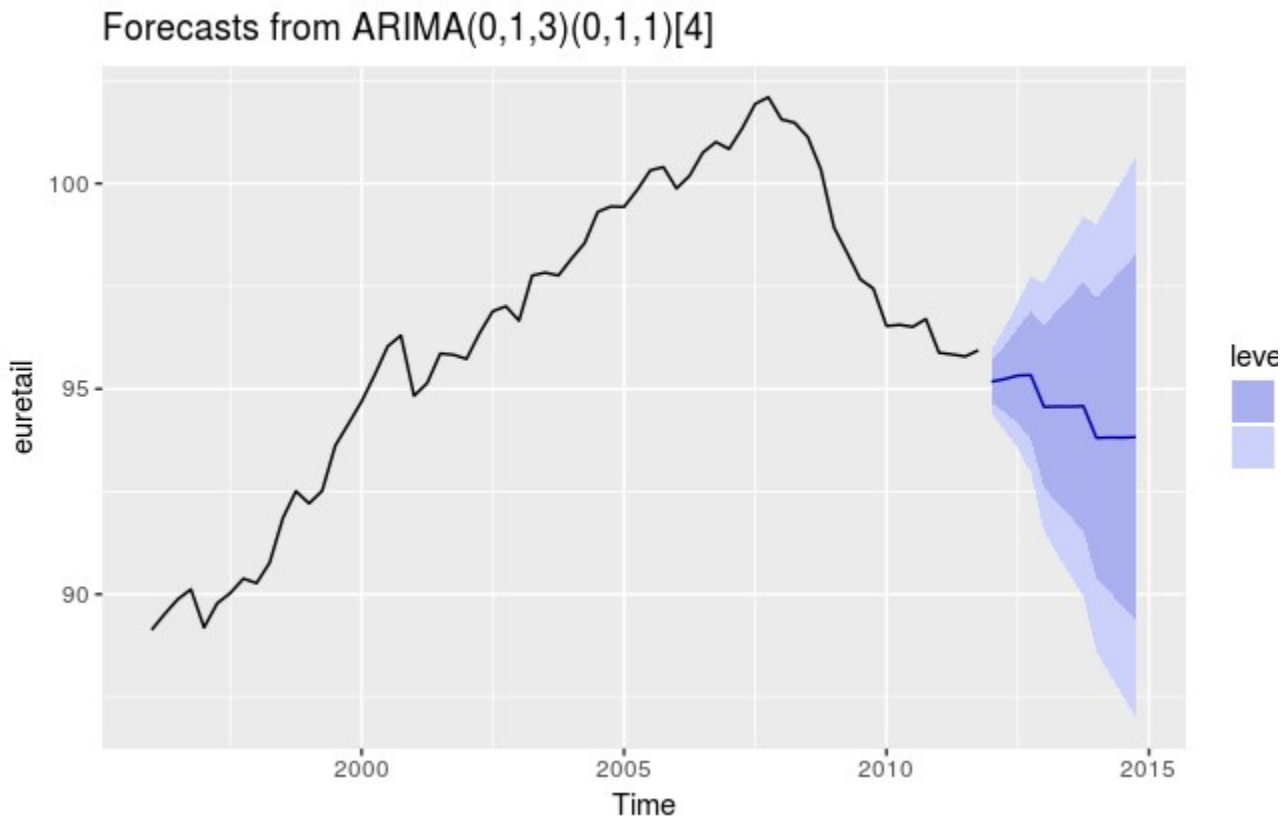


Figure 8.20: Forecasts of the European retail trade index data using the ARIMA(0,1,3)(0,1,1)4 model. 80% and 95% prediction intervals are shown.

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
#>       0.736  -0.466  0.216  -0.843
#> s.e.  0.224   0.199  0.210   0.188
#>
#> sigma^2 estimated as 0.159:  log likelihood=-29.6
#> AIC=69.2   AICc=70.4   BIC=79.6
```

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
```

```
#>       0.263  0.369  0.420  -0.664
#> s.e.  0.124  0.126  0.129   0.155
#>
#> sigma^2 estimated as 0.156:  log likelihood=-28.6
#> AIC=67.3   AICc=68.4   BIC=77.7
```

This time it returned the same model we had identified.

8.10 ARIMA vs ETS

It is a 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 counterparts. 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 ETS models with seasonality or non-damped trend or both have two unit roots (i.e., they need two levels of differencing to make them stationary). All other ETS models have one unit root (they need one level of differencing to make them 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)	$\theta 1 = \alpha - 1$
ETS(A,A,N)	ARIMA(0,2,2)	$\theta 1 = \alpha + \beta - 2$
		$\theta 2 = 1 - \alpha$
ETS(A,A,N)	ARIMA(1,1,2)	$\phi 1 = \phi$
		$\theta 1 = \alpha + \phi\beta - 1 - \phi$
		$\theta 2 = (1 - \alpha)\phi$
ETS(A,N,A)	ARIMA(0,0,m)(0,1,0)m	
ETS(A,A,A)	ARIMA(0,1,m+1)(0,1,0)m	
ETS(A,A,A)	ARIMA(1,0,m+1)(0,1,0)m	

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

1. Figure 8.25 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?

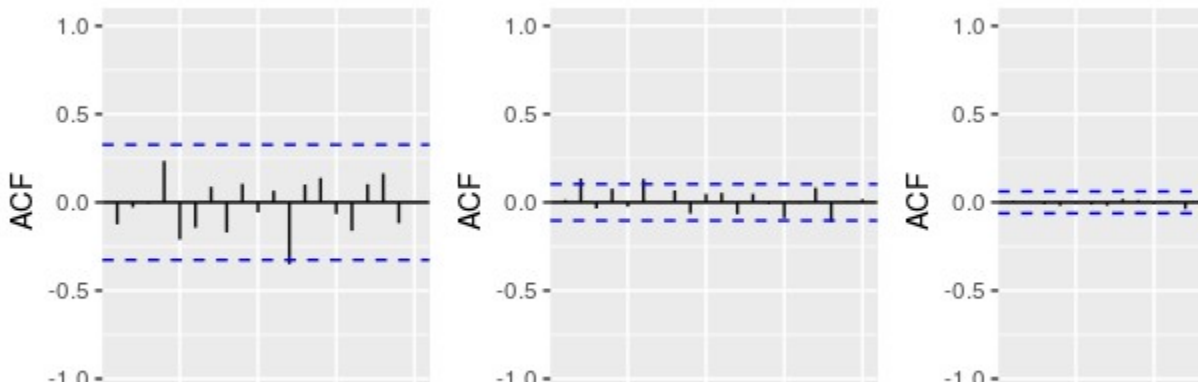


Figure 8.25: Left: ACF for a white noise series of 36 numbers. Middle: ACF for a white noise series of 360 numbers. Right: ACF for a white noise series of 1,000 numbers.

   2. 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.

   1. `usnetelec`
   2. `usgdp`
   3. `mcopper`
   4. `enplanements`
   5. `visitors`

4. For the `enplanements` data, write down the differences you chose above using backshift operator notation.

5. For your retail data (Exercise 1 of Section 2.10), find the appropriate order of differencing (after transformation if necessary) to obtain stationary data.

6. 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 $\phi 1=0.6$ and $\sigma 2=1$. The process starts with $y1=0$.

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

   2. Produce a time plot for the series. How does the plot change as you change $\phi 1$?
   3. Write your own code to generate data from an MA(1) model with $\theta 1=0.6$ and $\sigma 2=1$.
   4. Produce a time plot for the series. How does the plot change as you change $\theta 1$?
   5. Generate data from an ARMA(1,1) model with $\phi 1=0.6$, $\theta 1=0.6$ and $\sigma 2=1$.
   6. Generate data from an AR(2) model with $\phi 1=-0.8$, $\phi 2=0.3$ and $\sigma 2=1$. (Note that these parameters will give a non-stationary series.)
   7. 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`).

   1. By studying appropriate graphs of the series in R, find an appropriate ARIMA(p,d,q) model for these data.
   2. Should you include a constant in the model? Explain.
   3. Write this model in terms of the backshift operator.
   4. Fit the model using R and examine the residuals. Is the model satisfactory?

5. Forecast three times ahead. Check your forecasts by hand to make sure that you know how they have been calculated.
6. Create a plot of the series with forecasts and prediction intervals for the next three periods shown.
7. Does `auto.arima` give the same model you have chosen? If not, which model do you think is better?

8. For the `usgdp` series:
    1. if necessary, find a suitable Box-Cox transformation for the data;
    2. fit a suitable ARIMA model to the transformed data using `auto.arima()`;
    3. try some other plausible models by experimenting with the orders chosen;
    4. choose what you think is the best model and check the residual diagnostics;
    5. produce forecasts of your fitted model. Do the forecasts look reasonable?
    6. compare the results with what you would obtain using `ets()` (with no transformation).
9. Consider the quarterly number of international tourists to Australia for the period 1999 –2010. (Data set `austourists`.)
    1. Describe the time plot.
    2. What can you learn from the ACF graph?
    3. What can you learn from the PACF graph?
    4. Produce plots of the seasonally differenced data (1−B4)Yt. What model do these graphs suggest?
    5. Does `auto.arima` give the same model that you chose? If not, which model do you think is better?
    6. Write the model in terms of the backshift operator, then without using the backshift operator.
10. 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.
    1. Examine the 12-month moving average of this series to see what kind of trend is involved.
    2. Do the data need transforming? If so, find a suitable transformation.
    3. Are the data stationary? If not, find an appropriate differencing which yields stationary data.
    4. 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?
    5. 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.
    6. 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.
    7. How many years of forecasts do you think are sufficiently accurate to be usable?
11. For the `mcopper` data:
    1. if necessary, find a suitable Box-Cox transformation for the data;
    2. fit a suitable ARIMA model to the transformed data using `auto.arima()`;
    3. try some other plausible models by experimenting with the orders chosen;
    4. choose what you think is the best model and check the residual diagnostics;
    5. produce forecasts of your fitted model. Do the forecasts look reasonable?
    6. compare the results with what you would obtain using `ets()` (with no transformation).
12. Choose one of the following seasonal time series: `hsales`, `auscafe`, `qauselec`, `qcement`, `qgas`.
    1. Do the data need transforming? If so, find a suitable transformation.
    2. Are the data stationary? If not, find an appropriate differencing which yields stationary data.
    3. 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?

4. 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.
5. Forecast the next 24 months of data using your preferred model.
6. Compare the forecasts obtained using `ets()`.

13. 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?

14. For your retail time series (Exercise 5 above):
    1. develop an appropriate seasonal ARIMA model;
    2. compare the forecasts with those you obtained in earlier chapters;
    3. Obtain up-to-date retail data from the ABS website (Cat 8501.0, Table 11), and compare your forecasts with the actual numbers. How good were the forecasts from the various models?
15.
    1. Produce a time plot of the sheep population of England and Wales from 1867–1939 (data set `sheep`).
    2. Assume you decide to fit the following model: $y_t = y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) + \phi_2(y_{t-2} - y_{t-3}) + \phi_3(y_{t-3} - y_{t-4}) + e_t$, where e_t is a white noise series. What sort of ARIMA model is this (i.e., what are p, d, and q)?
    3. By examining the ACF and PACF of the differenced data, explain why this model is appropriate.
    4. 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 $\phi_1 = 0.42$, $\phi_2 = -0.20$, and $\phi_3 = -0.30$. Without using the `forecast` function, calculate forecasts for the next three years (1940–1942).

    5. Now fit the model in R and obtain the forecasts using `forecast`. How are they different from yours? Why?

16.
    1. Plot the annual bituminous coal production in the United States from 1920 to 1968 (data set).
    2. You decide to fit the following model to the series: $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \phi_4 y_{t-4} + e_t$ where y_t is the coal production in year t and e_t is a white noise series. What sort of ARIMA model is this (i.e., what are p, d, and q)?
    3. Explain why this model was chosen using the ACF and PACF.
    4. 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 $c=162.00$, $\phi_1=0.83$, $\phi_2=-0.34$, $\phi_3=0.55$, and $\phi_4=-0.38$. Without using the `forecast` function, calculate forecasts for the next three years (1969–1971).

5. Now fit the model in R and obtain the forecasts from the same model. How are they different from yours? Why?

17.

1. Install the **rdatamarket** package in R using

   ```
   install.packages("rdatamarket")
   ```

2. 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.)
3. Plot graphs of the data, and try to identify an appropriate ARIMA model.
4. Do residual diagnostic checking of your ARIMA model. Are the residuals white noise?
5. Use your chosen ARIMA model to forecast the next four years.
6. Now try to identify an appropriate ETS model.
7. Do residual diagnostic checking of your ETS model. Are the residuals white noise?
8. Use your chosen ETS model to forecast the next four years.
9. Which of the two models do you prefer?

8.12 Further reading

The classic text which popularized ARIMA modelling was Box and Jenkins (1970). The most recent edition is Box et al. (2015), and it is still an excellent reference for all things ARIMA. Brockwell and Davis (2016) provides a good introduction to the mathematical background to the models, while Peña, Tiao, and Tsay (2001) describes some alternative automatic algorithms to the one used by `auto.arima()`.

References

Box, George E P, and Gwilym M Jenkins. 1970. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Box, George E P, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis: Forecasting and Control*. 5th ed. Hoboken, New Jersey: John Wiley & Sons.

Brockwell, Peter J, and Richard A Davis. 2016. *Introduction to Time Series and Forecasting*. 3rd ed. New York: Springer.

Peña, Daniel, George C Tiao, and Ruey S Tsay, eds. 2001. *A Course in Time Series Analysis*. New York: John Wiley & Sons.

Chapter 9 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 allow 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 section, we consider how to extend ARIMA models in order to allow other information to be included in the models. We begin by simply combining regression models and ARIMA models to give a regression with ARIMA errors. These are then extended to the general class of dynamic regression models. In Chapter 5 we considered regression models of the form $y_t=\beta_0+\beta_1 x_{1,t}+\cdots+\beta_k x_{k,t}+e_t$, where y_t is a linear function of the k predictor variables ($x_{1,t}, \ldots, x_{k,t}$), and e_t 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 e_t 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 $y_t=\beta_0+\beta_1 x_{1,t}+\cdots+\beta_k x_{k,t}+n_t, (1-\phi_1 B)(1-B)n_t=(1+\theta_1 B)e_t,$

where e_t is a white noise series.

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

9.1 Estimation

When we estimate the parameters from the model, we need to minimise the sum of squared e_t values. If we minimise the sum of squared n_t 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 $\hat{\beta}_0, \ldots, \hat{\beta}_k$ are no longer 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 p-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".

Minimising the sum of squared e_t 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 y_t and all of the

predictors (x1,t,…,xk,t) 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 yt and the predictors that is stationary, then the estimated coefficients will be consistent.[16]

We therefore first difference the non-stationary variables in the model. It is often desirable to maintain the form of the relationship between yt 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 $y'_t=\beta_1 x'_{1,t}+\cdots+\beta_k x'_{k,t}+n'_t$, $(1-\phi_1 B)\, n'_t=(1+\theta_1 B)e_t$,

where $y'_t=y_t-y_{t-1}$, $x'_{t,i}=x_{t,i}-x_{t-1,i}$ and $n'_t=n_t-n_{t-1}$, which is a regression model in differences with ARMA errors.

9.2 Regression with ARIMA errors in R

The R function `Arima()` will fit 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 model before the model is estimated. For example, the R command

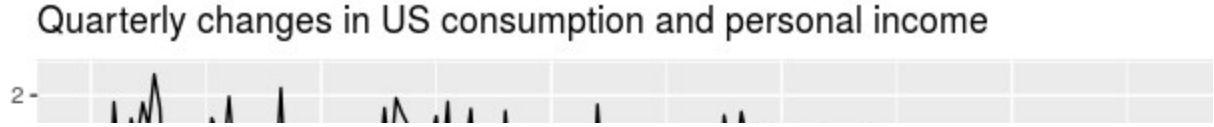
The `auto.arima()` function will also handle regression terms via the `xreg` argument. The user must specify the predictor variables to include, but `auto.arima` will select the best ARIMA model for the errors.

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 lowest AICc value selected.

Example: US Personal Consumption and Income

Figure 9.1 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")
```



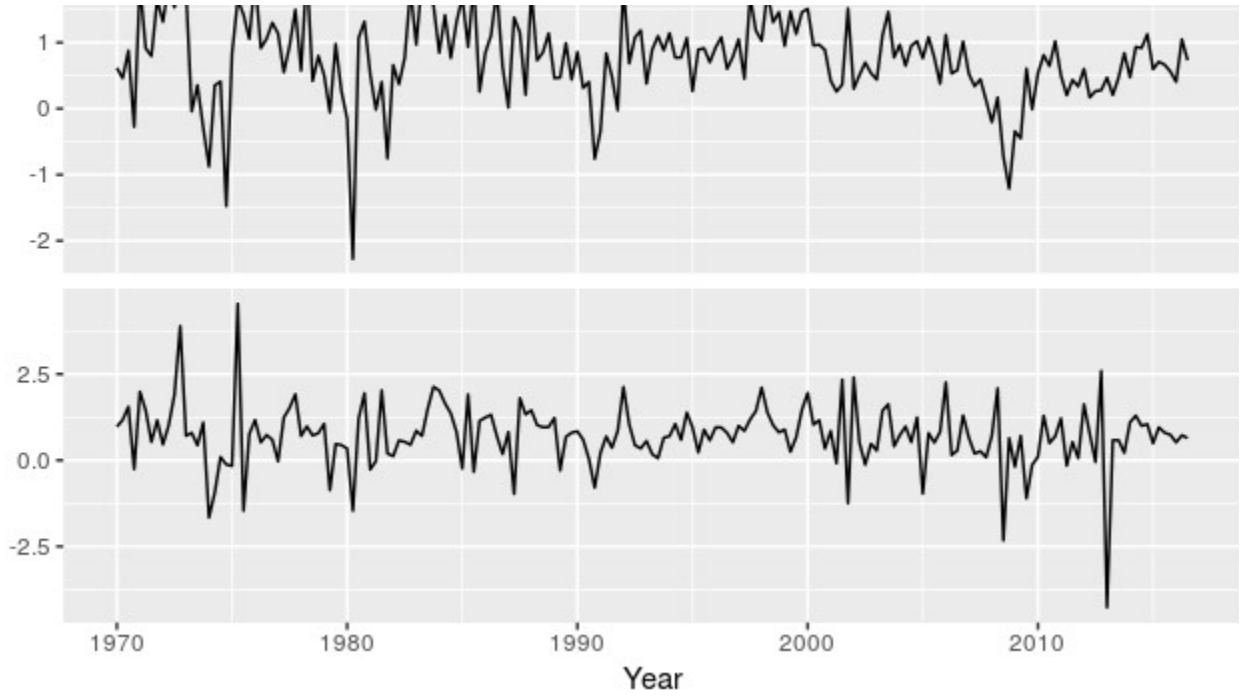


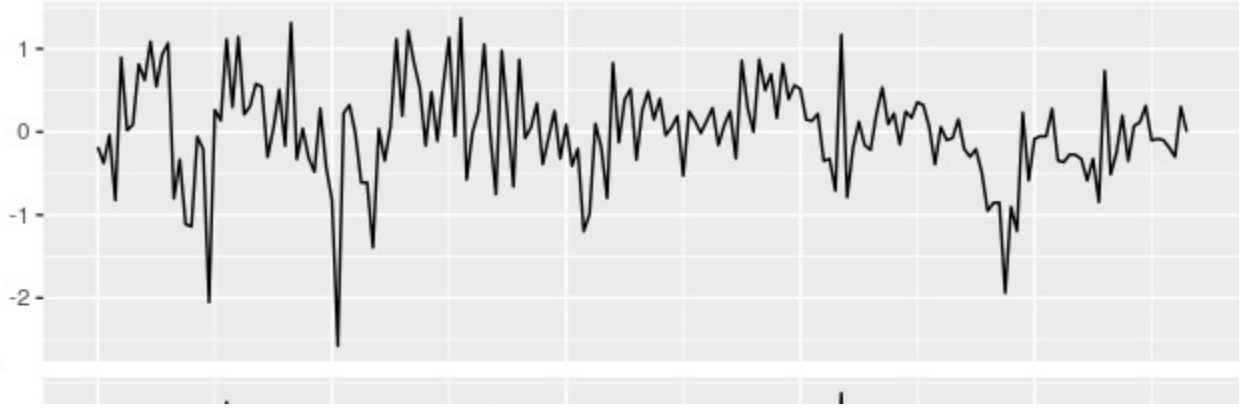
Figure 9.1: Percentage changes in quarterly personal consumption expenditure and personal disposable income for the USA, 1970 to 2010.

```
(fit <- auto.arima(uschange[,"Consumption"], xreg=uschange[,"Income"]))
#> Series: uschange[, "Consumption"]
#> Regression with ARIMA(1,0,2)           errors
#>
#> Coefficients:
#>         ar1     ma1    ma2  intercept   xreg
#>       0.692  -0.576  0.198      0.599  0.203
#> s.e.  0.116   0.130  0.076      0.088  0.046
#>
#> sigma^2 estimated as 0.322:  log likelihood=-157
#> AIC=326   AICc=326   BIC=345
```

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 $y_t = 0.60 + NA x_t + n_t, n_t = 0.69 n_{t-1} + e_t - 0.58 e_{t-1} + 0.20 e_{t-2}, e_t \sim NID(0, 0.322)$.

We can recover both the nt and et series using the `residuals()` function.

```
cbind("Regression Errors" = residuals(fit, type="regression"),
      "ARIMA errors" = residuals(fit, type="innovation")) %>%
  autoplot(facets=TRUE)
```



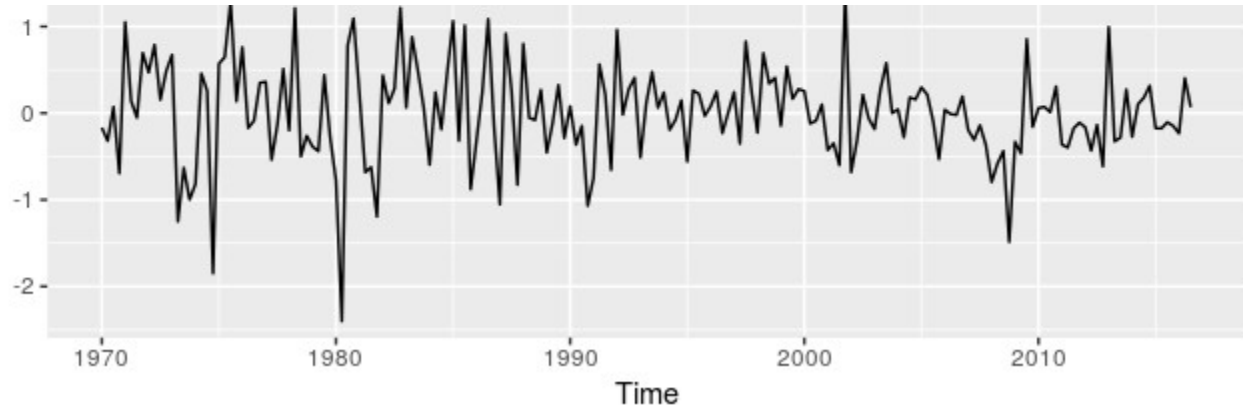


Figure 9.2: Regression errors (nt) and ARIMA errors (et) from the fitted model.

It is the ARIMA errors that should resemble a white noise series.

```
checkresiduals(fit)
```

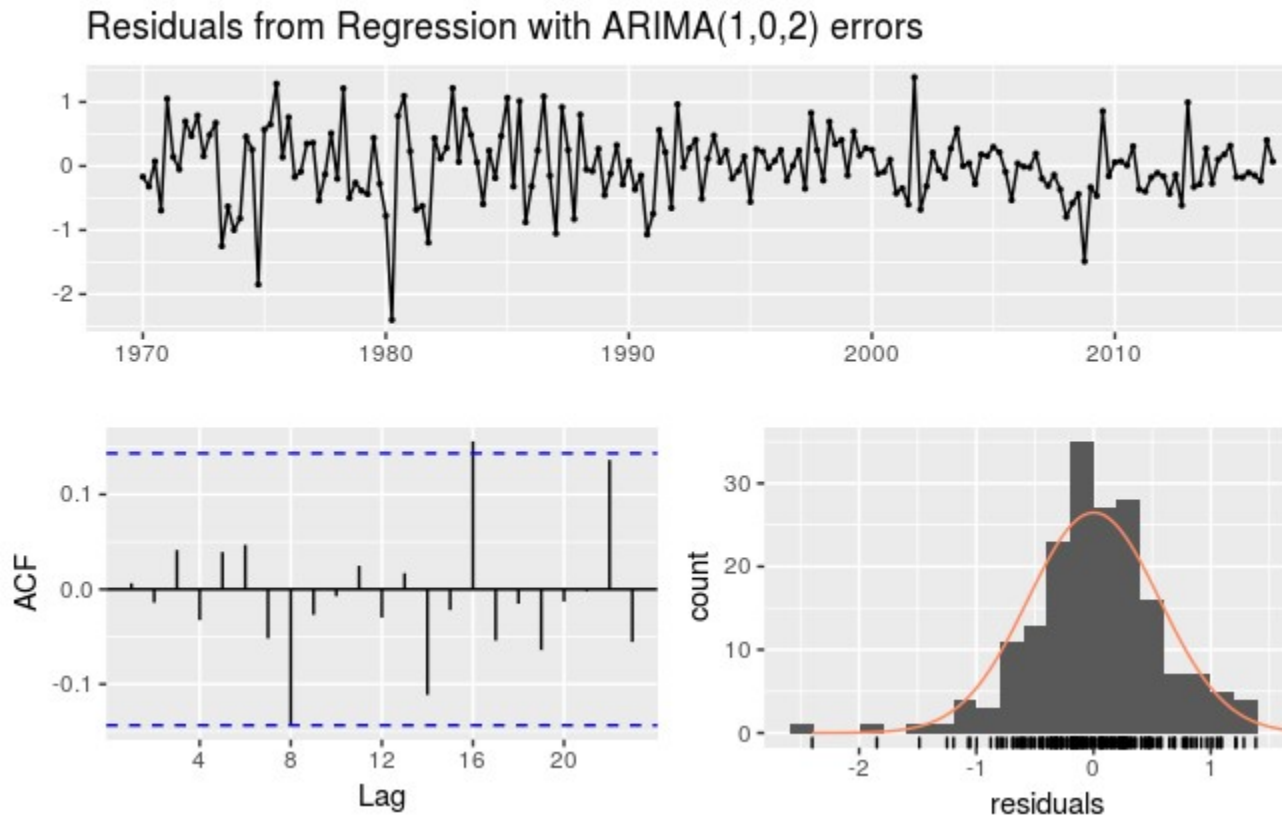


Figure 9.3: The residuals (i.e., the ARIMA errors) are not significantly different from white noise.

```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from Regression with ARIMA(1,0,2) errors
#> Q* = 6, df = 3, p-value = 0.1
#>
#> Model df: 5.   Total lags used: 8
```

9.3 Forecasting

To forecast a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model, and combine the results. As with ordinary regression models, in order to obtain forecasts we first need to forecast the predictors. When the predictors are known into the future (e.g., calendar-related variables such as time, day-of-week, etc.), this is straightforward. But when the predictors are themselves unknown, we must either model them separately, or use assumed future values for each predictor.

To continue the previous example, we will calculate forecasts for the next eight quarters assuming that the future percentage changes in personal disposable income will be equal to the mean percentage change from the last forty years.

```
fcast <- forecast(fit, xreg=rep(mean(uschange[,2]),8))
autoplot(fcast) + xlab("Year") +
  ylab("Percentage change")
```

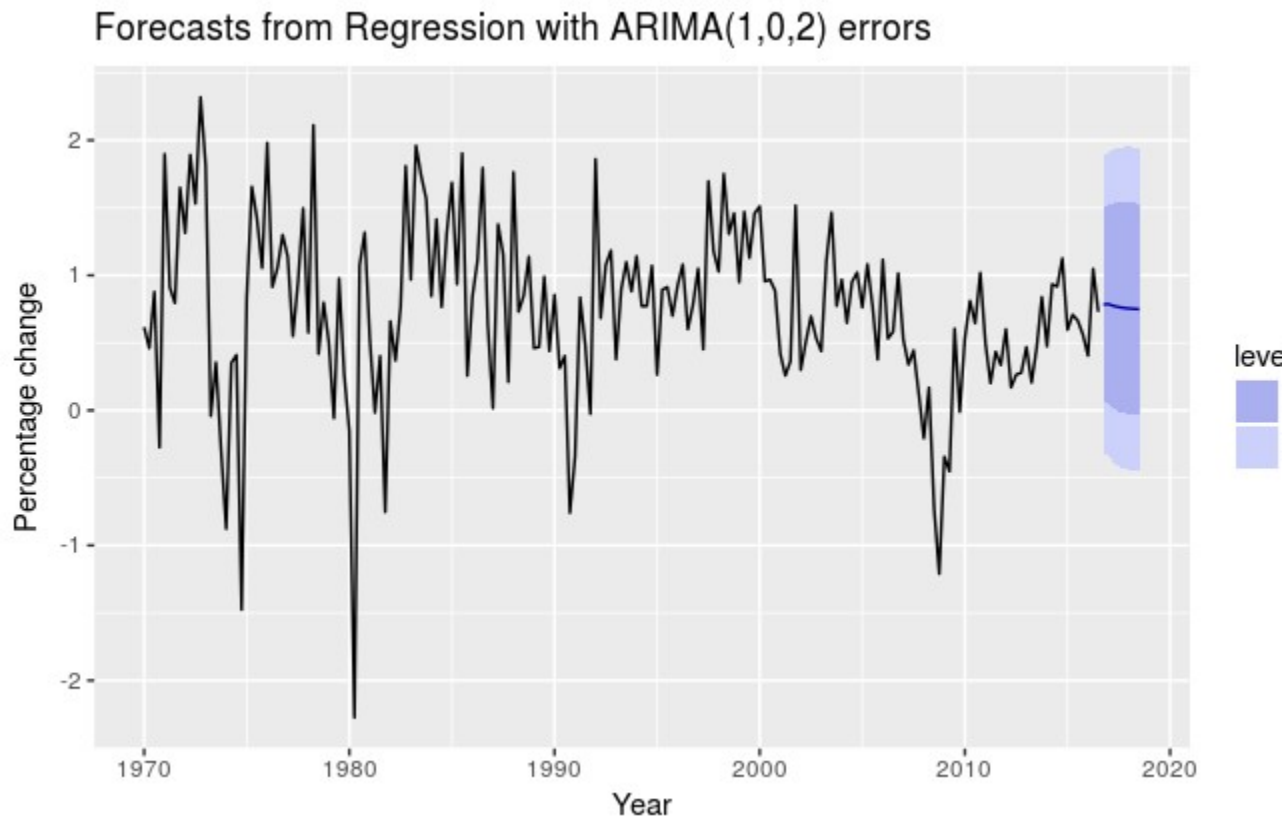


Figure 9.4: Forecasts obtained from regressing the percentage change in consumption expenditure on the percentage change in disposable income, with an ARIMA(1,0,2) error model.

The prediction intervals for this model are narrower than those for the model developed in Section 8.5 because we are now able to explain some of the variation in the data using the income predictor.

It is important to realise that the prediction intervals from regression models (with or without ARIMA errors) do not take into account the uncertainty in the forecasts of the predictors. So they should be interpreted as being conditional on the assumed (or estimated) future values of the predictor variables.

9.4 Stochastic and deterministic trends

Although these models appear quite similar (they only differ in the number of differences that need to be applied to nt), their forecasting characteristics are quite different.

Example: International visitors to Australia

```
autoplot(austa) + xlab("Year") +
  ylab("millions of people") +
  ggtitle("Total annual international visitors to Australia")
```

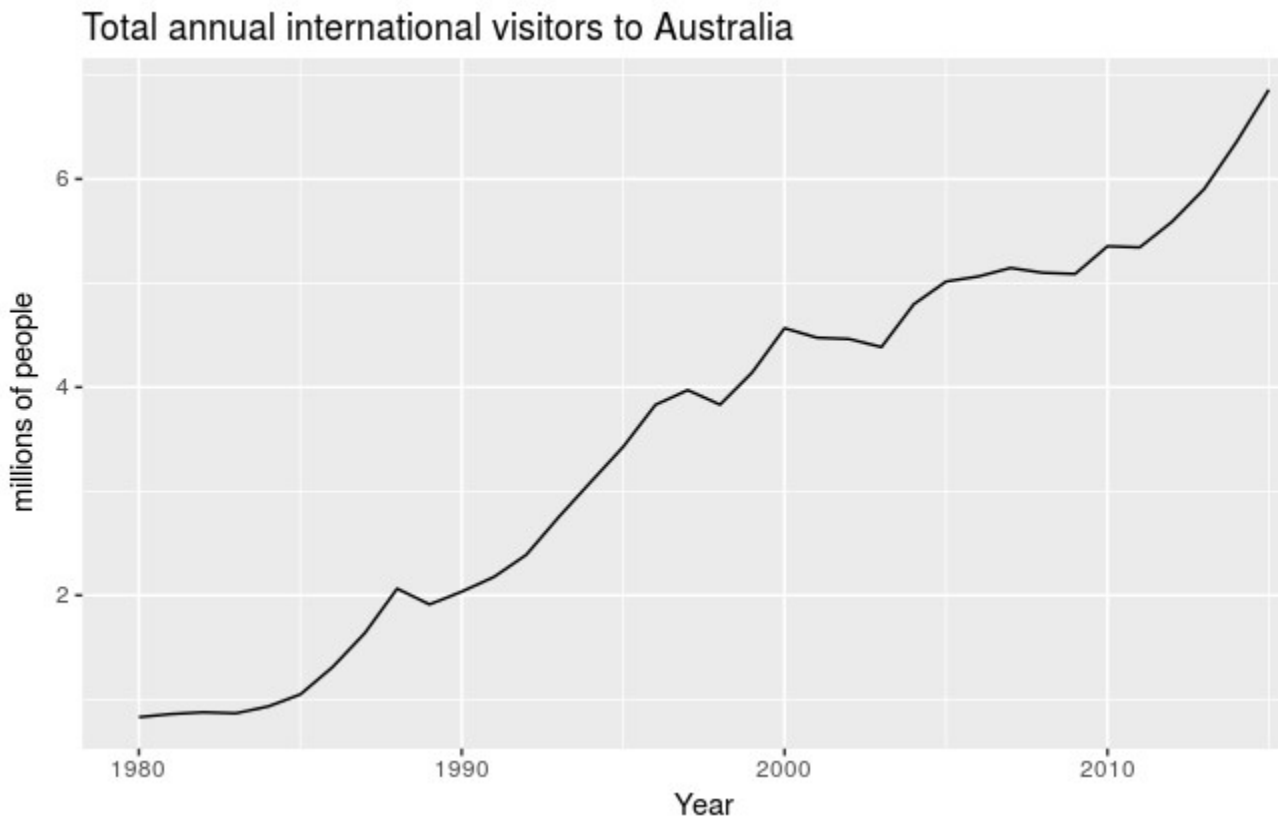


Figure 9.5: Annual international visitors to Australia, 1980–2010.

Figure 9.5 shows the total number of international visitors to Australia each year from 1980 to 2010. We will fit both a deterministic and a stochastic trend model to these data.

The deterministic trend model is obtained as follows:

```
trend <- seq_along(austa)
(fit1 <- auto.arima(austa, d=0, xreg=trend))
#> Series: austa
#> Regression with ARIMA(2,0,0) errors
#>
#> Coefficients:
#>         ar1     ar2  intercept   xreg
#>       1.11  -0.380      0.416  0.171
#> s.e.  0.16   0.158      0.190  0.009
#>
#> sigma^2 estimated as 0.0298:  log likelihood=13.6
#> AIC=-17.2   AICc=-15.2   BIC=-9.28
```

This model can be written as $y_t = 0.42 + 0.17t + n_t$ $n_t = 1.11n_{t-1} - 0.38n_{t-2} + e_t$ $e_t \sim NID(0,0.0298)$.

The estimated growth in visitor numbers is 0.17 million people per year.

Alternatively, the stochastic trend model can be estimated.

```
(fit2 <- auto.arima(austa, d=1))
#> Series: austa
#> ARIMA(0,1,1) with drift
#>
#> Coefficients:
#>         ma1  drift
#>       0.301  0.173
#> s.e.  0.165  0.039
#>
#> sigma^2 estimated as 0.0338:  log likelihood=10.6
#> AIC=-15.2   AICc=-14.5   BIC=-10.6
```

This model can be written as $y_t - y_{t-1} = 0.17 + n't$, or equivalently $y_t = y_0 + 0.17t + n_t$ $n_t = n_{t-1} + 0.30e_{t-1} + e_t$ $e_t \sim NID(0, 0.0338)$.

In this case, the estimated growth in visitor numbers is also 0.17 million people per year. Although the growth estimates are similar, the prediction intervals are not, as Figure 9.6 shows. In particular, stochastic trends have much wider prediction intervals because the errors are non-stationary.

```
fc1 <- forecast(fit1, xreg=data.frame(trend=length(austa)+1:10))
fc2 <- forecast(fit2, h=10)
autoplot(austa) +
  forecast::autolayer(fc2, series="Stochastic trend") +
  forecast::autolayer(fc1, series="Deterministic trend") +
  ggtitle("Forecasts from deterministic and stochastic trend models") +
  xlab("Year") + ylab("Visitors to Australia (millions)") +
  guides(colour=guide_legend(title="Forecast"))
```

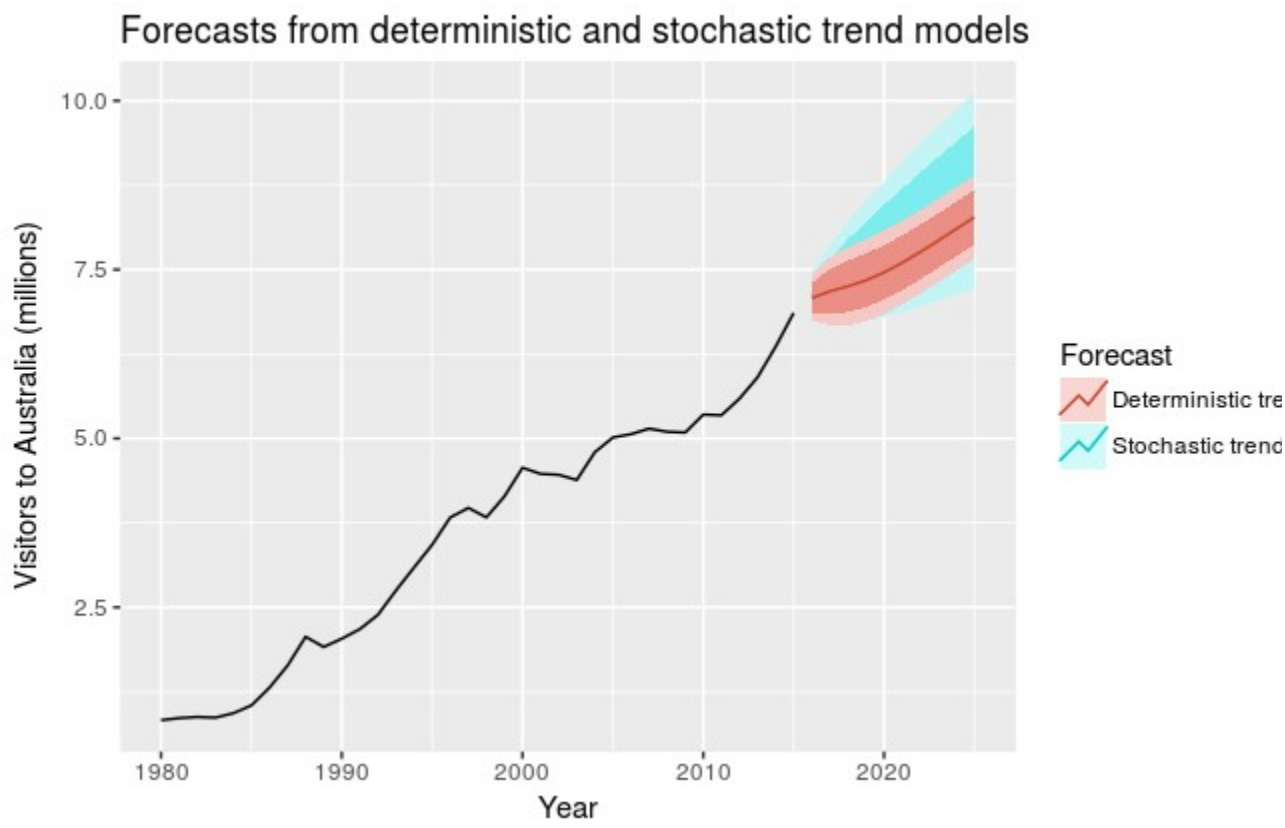


Figure 9.6: Forecasts of annual international visitors to Australia using a deterministic trend model and a stochastic trend model.

There is an implicit assumption with deterministic trends that the slope of the trend is not going to change over time. On the other hand, stochastic trends can change, and the estimated growth is only assumed to be the average growth over the historical period, not necessarily the rate of growth that will be observed into the future. Consequently, it is safer to forecast with stochastic trends, especially for longer forecast horizons, as the prediction intervals allow for greater uncertainty in future growth.

9.5 Dynamic harmonic regression

When there are long seasonal periods, a dynamic regression with Fourier terms is often better than other models we have considered in this book.

For example, daily data can have annual seasonality of length 365, weekly data has seasonal period of approximately 52, while half-hourly data can have several seasonal periods, the shortest of which is the daily pattern of period 48.

Seasonal versions of ARIMA and ETS models are designed for shorter periods such as 12 for monthly data or 4 for quarterly data. The `ets()` function restricts seasonality to be a maximum period of 24 to allow hourly data but not data with a larger seasonal frequency. The problem is that there are m−1 parameters to be estimated for the initial seasonal states where m is the seasonal period. So for large m, the estimation becomes almost impossible.

The `Arima()` and `auto.arima()` functions will allow a seasonal period up to m=350, but in practice will usually run out of memory whenever the seasonal period is more than about 200. In any case, seasonal differencing of very high order does not make a lot of sense — for daily data it involves comparing what happened today with what happened exactly a year ago and there is no constraint that the seasonal pattern is smooth.

So for such time series, we prefer a harmonic regression approach where the seasonal pattern is modelled using Fourier terms with short-term time series dynamics handled by an ARMA error.

The advantages of this approach are:

- it allows any length seasonality;
- for data with more than one seasonal period, you can include Fourier terms of different frequencies;
- the seasonal pattern is smooth for small values of K (but more wiggly seasonality can be handled by increasing K);
- the short-term dynamics are easily handled with a simple ARMA error.

The only real disadvantage (compared to a seasonal ARIMA model) is that the seasonality is assumed to be fixed — the pattern is not allowed to change over time. But in practice, seasonality is usually remarkably constant so this is not a big disadvantage except for very long time series.

NEED TO ADD EXAMPLE

Sometimes, the impact of a predictor which is included in a regression model will not be simple and immediate. For example, an advertising campaign may impact sales for some time beyond the end of the campaign, and sales in one month will depend on the advertising expenditure in each of the past few months. Similarly, a change in a company’s safety policy may reduce accidents

immediately, but have a diminishing effect over time as employees take less care when they become familiar with the new working conditions.

In these situations, we need to allow for lagged effects of the predictor. Suppose that we have only one predictor in our model. Then a model which allows for lagged effects can be written as $y_t=\beta_0+\gamma_0 x_t+\gamma_1 x_{t-1}+\cdots+\gamma_k x_{t-k}+n_t$, where n_t is an ARIMA process. The value of k can be selected using the AICc, along with the values of p and q for the ARIMA error.

Example: TV advertising and insurance quotations

A US insurance company advertises on national television in an attempt to increase the number of insurance quotations provided (and consequently the number of new policies). Figure 9.7 shows the number of quotations and the expenditure on television advertising for the company each month from January 2002 to April 2005.

```
autoplot(insurance, facets=TRUE) +
  xlab("Year") + ylab("") +
  ggtitle("Insurance advertising and quotations")
```

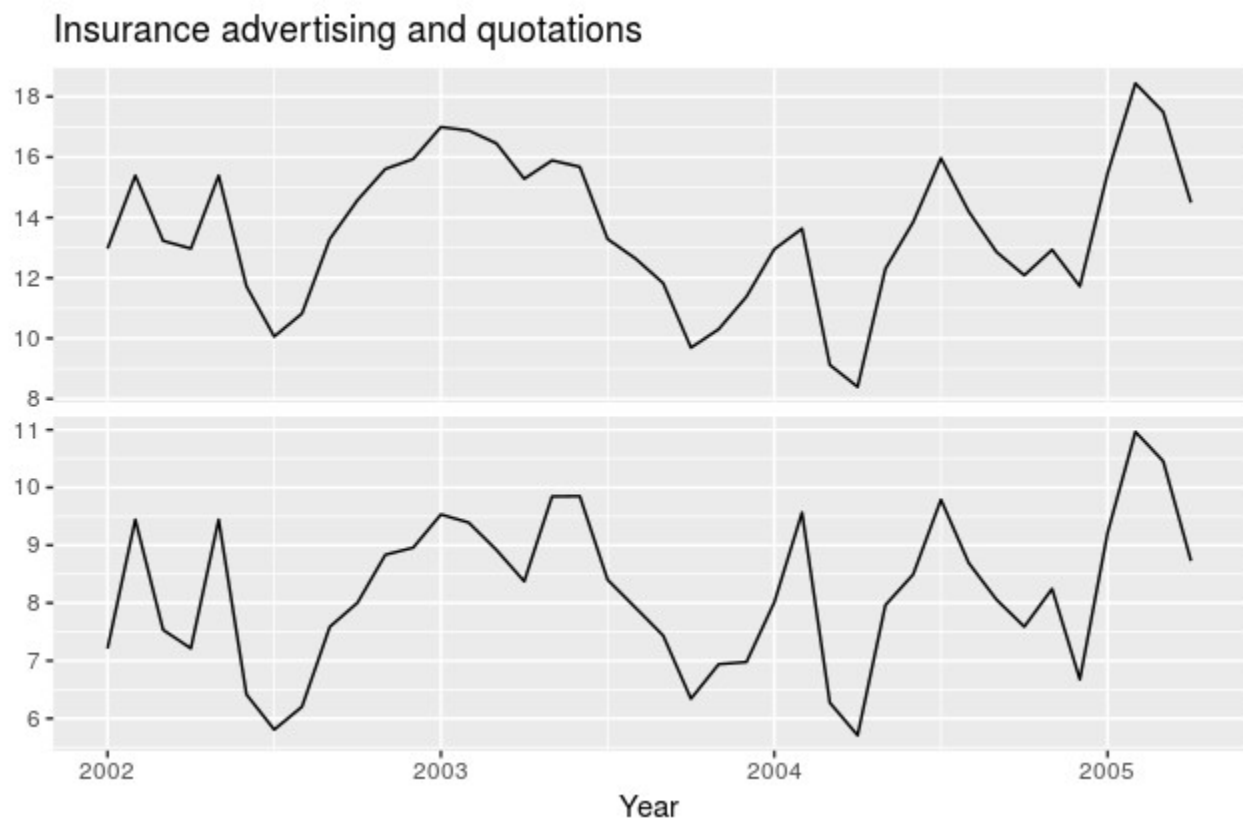


Figure 9.7: Numbers of insurance quotations provided per month and the expenditure on advertising per month.

We will consider including advertising expenditure for up to four months; that is, the model may include advertising expenditure in the current month, and the three months before that. When comparing models, it is important that they all use the same training set. In the following code, we exclude the first three months in order to make fair comparisons. The best model is the one with the smallest AICc value.

```
# Lagged predictors. Test 0, 1, 2 or 3 lags.
Advert <- cbind(
  AdLag0 = insurance[,"TV.advert"],
```

```
  AdLag1 = lag(insurance[,"TV.advert"],-1),
  AdLag2 = lag(insurance[,"TV.advert"],-2),
  AdLag3 = lag(insurance[,"TV.advert"],-3))[1:NROW(insurance),]

# Choose optimal lag length for advertising based on AICc
# Restrict data so models use same fitting period
fit1 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1], d=0)
fit2 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:2], d=0)
fit3 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:3], d=0)
fit4 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:4], d=0)

# Best model fitted to all data (based on AICc)
# Refit using all data
(fit <- auto.arima(insurance[,1], xreg=Advert[,1:2], d=0))
#> Series: insurance[, 1]
#> Regression with ARIMA(3,0,0)            errors
#>
#> Coefficients:
#>         ar1     ar2    ar3  intercept  AdLag0  AdLag1
#>       1.41  -0.932  0.359      2.039   1.256   0.162
#> s.e.  0.17   0.255  0.159      0.993   0.067   0.059
#>
#> sigma^2 estimated as 0.217:  log likelihood=-23.9
#> AIC=61.8   AICc=65.3   BIC=73.6
```

The chosen model includes advertising only in the current month and the previous month, and has AR(3) errors. The model can be written as $y_t = 2.04 + 1.26x_t + 0.16x_{t-1} + n_t$, where y_t is the number of quotations provided in month t, x_t is the advertising expenditure in month t, $n_t = 1.41n_{t-1} - 0.93n_{t-2} + 0.36n_{t-3} + e_t$, and e_t is white noise.

We can calculate forecasts using this model if we assume future values for the advertising variable. If we set the future monthly advertising to 8 units, we get the following forecasts.

```
fc8 <- forecast(fit, h=20,
  xreg=cbind(AdLag0=rep(8,20), AdLag1=c(Advert[40,1],rep(8,19))))
autoplot(fc8) + ylab("Quotes") +
  ggtitle("Forecast quotes with future advertising set to 8")
```

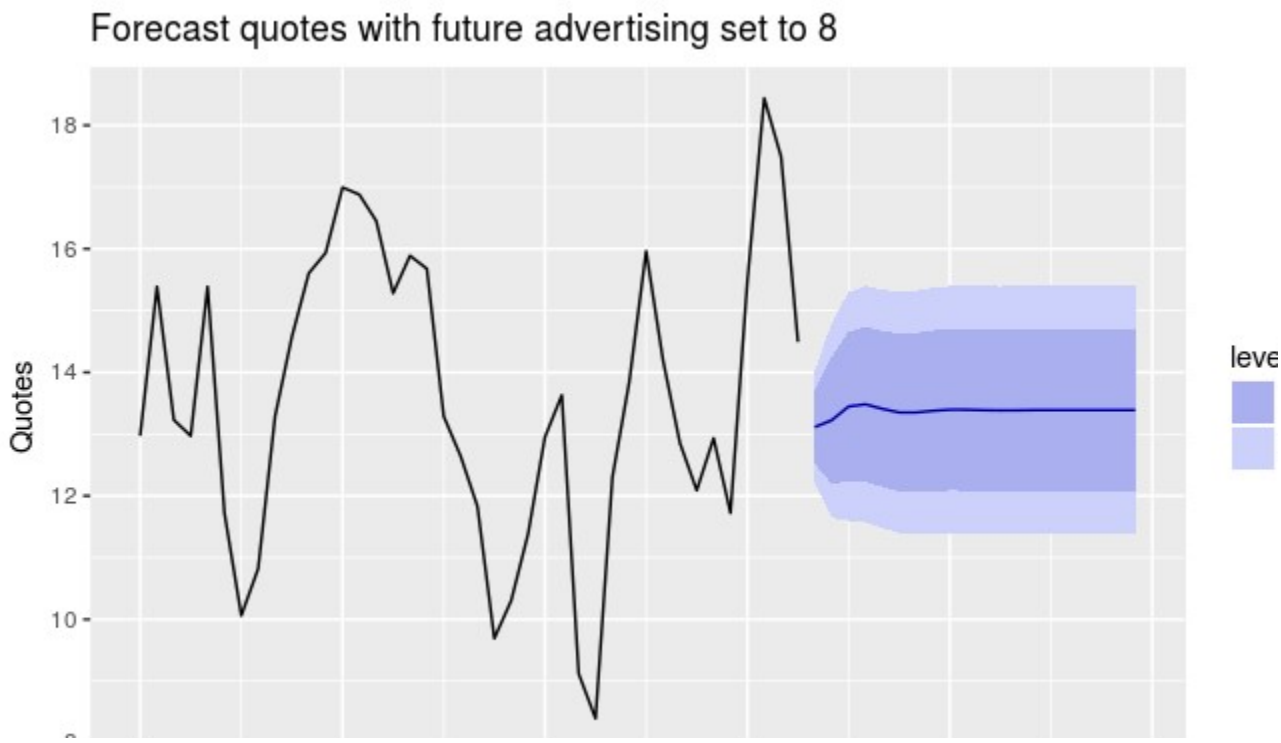


Figure 9.8: Forecasts of monthly insurance quotes, assuming that the future advertising expenditure is 8 units in each future month.

1. Consider monthly sales and advertising data for an automotive parts company (data set `advert`).
    1. Fit a standard regression model yt=a+bxt+nt where yt denotes sales and xt denotes advertising using the `tslm()` function.
    2. Show that the residuals have significant autocorrelation.
    3. What difference does it make you use the function instead:

        ```
        Arima(advert[,'sales'], xreg=advert[,'advert'], order=c(0,0,0))
        ```

    4. Refit the model using `auto.arima()`. How much difference does the error model make to the estimated parameters? What ARIMA model for the errors is selected?
    5. Check the residuals of the fitted model.
    6. Assuming the advertising budget for the next six months is exactly 80 units per month, produce sales forecasts with prediction intervals for the next six months.
2. This exercise uses data set `huron` giving the level of Lake Huron from 1875–1972.
    1. Fit a piecewise linear trend model to the Lake Huron data with a knot at 1920 and an ARMA error structure.
    2. Forecast the level for the next 30 years.
3. This exercise concerns `motel`: the total monthly takings from accommodation and the total room nights occupied at hotels, motels, and guest houses in Victoria, Australia, between January 1980 and June 1995. Total monthly takings are in thousands of Australian dollars; total room nights occupied are in thousands.
    1. Use the data to calculate the average cost of a night's accommodation in Victoria each month.
    2. Estimate the monthly CPI.
    3. Produce time series plots of both variables and explain why logarithms of both variables need to be taken before fitting any models.
    4. Fit an appropriate regression model with ARIMA errors. Explain your reasoning in arriving at the final model.
    5. Forecast the average price per room for the next twelve months using your fitted model. (Hint: You will need to produce forecasts of the CPI figures first.)
4. We fitted a harmonic regression model to part of the `gasoline` series in Exercise 6 in Section 5.10. We will now revisit this model, and extend it to include more data and ARMA errors.
    1. Using `tslm`, fit a harmonic regression with a piecewise linear time trend to the full `gasoline` series. Select the position of the knots in the trend and the appropriate number of Fourier terms to include by minimizing the AICc or CV value.
    2. Now refit the model using `auto.arima` to allow for correlated errors, keeping the same predictor variables as you used with `tslm`.
    3. Check the residuals of the final model using the `checkresiduals()` function. Do they look sufficiently like white noise to continue? If not, try modifying your model, or removing the first few years of data.
    4. Once you have a model with white noise residuals, produce forecasts for the next year.

5. Electricity consumption is often modelled as a function of temperature. Temperature is measured by daily heating degrees and cooling degrees. Heating degrees is 18∘C minus the average daily temperature when the daily average is below 18∘C; otherwise it is zero. This provides a measure of our need to heat ourselves as temperature falls. Cooling degrees measures our need to cool ourselves as the temperature rises. It is defined as the average daily temperature minus 18∘C when the daily average is above 18∘C; otherwise it is zero. Let yt denote the monthly total of kilowatt-hours of electricity used, let x1,t denote the monthly total of heating degrees, and let x2,t denote the monthly total of cooling degrees.

   An analyst fits the following model to a set of such data: $y*t=b1x*1,t+b2x*2,t+nt$, where $(1−B)(1−B12)nt=1−\theta1B1−\phi12B12−\phi24B24et$ and $y*t=\log(Yt)$, $x*1,t=\sqrt{x1,t}$ and $x*2,t=\sqrt{x2,t}$.

   1. What sort of ARIMA model is identified for Nt?
   2. The estimated coefficients are

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

Parameter	**Estimate**	**s.e.**	**Z**	**P-value**
b1	0.0077	0.0015	4.98	0.000
b2	0.0208	0.0023	9.23	0.000
$\theta1$	0.5830	0.0720	8.10	0.000
$\phi12$	-0.5373	0.0856	-6.27	0.000
$\phi24$	-0.4667	0.0862	-5.41	0.000

   Explain what the estimates of b1 and b2 tell us about electricity consumption.

   3. Write the equation in a form more suitable for forecasting.
   4. Describe how this model could be used to forecast electricity demand for the next 12 months.
   5. Explain why the Nt term should be modelled with an ARIMA model rather than modeling the data using a standard regression package. In your discussion, comment on the properties of the estimates, the validity of the standard regression results, and the importance of the Nt model in producing forecasts.
6. For the retail time series considered in earlier chapters:
   1. Develop an appropriate dynamic regression model with Fourier terms for the seasonality. Use the AIC to select the number of Fourier terms to include in the model. (You will probably need to use the same Box-Cox transformation you identified previously.)
   2. Check the residuals of the fitted model. Does the residual series look like white noise?
   3. Compare the forecasts with those you obtained earlier using alternative models.

9.8 Further reading

A detailed discussion of dynamic regression models is provided in Pankratz (1991). A generalization of dynamic regression models, known as “transfer function models”, is discussed in Box et al. (2015).

References

Pankratz, Alan E. 1991. *Forecasting with Dynamic Regression Models*. New York: John Wiley & Sons.

Box, George E P, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis: Forecasting and Control*. 5th ed. Hoboken, New Jersey: John Wiley & Sons.

Chapter 10 Forecasting hierarchical or grouped time series

Warning: this is a more advanced section and assumes knowledge of some basic matrix algebra. We have however tried to simplify and explain in as much as detail as possible all the notation used starting from basic concepts and building up.

In this chapter we discuss forecasting large collections of time series that follow either a hierarchical or grouped aggregation structure. We clasify these as hierarchical or grouped time series in Sections 10.1 and 10.2 respectively. The challenge is that we require forecasts that are **coherent** across the aggregation structure. I.e., forecasts that add up in a manner consistent with the aggregation structure the collection of time series follow. Commonly used approaches are based on selecting a single level of aggregation, generating forecasts at that level and then combining these to produce coherent forecasts for the other time series. These approaches are presented in Sections 10.4-10.6. In Section 10.8 we introduce the concept of reconciliation where forecasts of all the times series in the collection are first generated and these are then reconciled so that they become coherent. This appraoch has the advantage that it incorporates information and the correlation structure across all the series in the collection.

The hts package

Forecasting hierarchical and grouped time series is implemented using the **hts** package. Forecasts are simply obtained as usual with the `forecast` function. The R code below shows the possible arguments that this function takes when applied to a hierarchical or grouped time series.

```
forecast(object, h = ifelse(frequency(object$bts)>1, 2*frequency(object$bts), 10),
fmethod = c("ets", "arima", "rw"),
method = c("comb", "bu", "mo", "tdgsa", "tdgsf", "tdfp"),
weights = c("wls", "ols", "mint", "nseries"), ...)
```

We explain each of the arguments in the sections that follow in detail. Below is a brief introduction.

`object`
: A hierarchical or grouped time series object to be forecast. We present how to generate a hierarchical time series object in Section 10.1 and a grouped time series object in Section 10.2 using the `hts` function.

`fmethod`
: The forecasting model used for generating base forecasts. Possible values are `"ets"`, `"arima"` and `rw`. These models have been studied in Chapters 7, 8 and 3 respectively. What is meant by base forecasts becomes clear in Section 10.3.

`method`

The method used for generating coherent forecasts. The possible values this argument can take are explained in Sections 10.4-10.8.

`weights`

Weight associated with the reconcilication approach presented in Section 10.8.

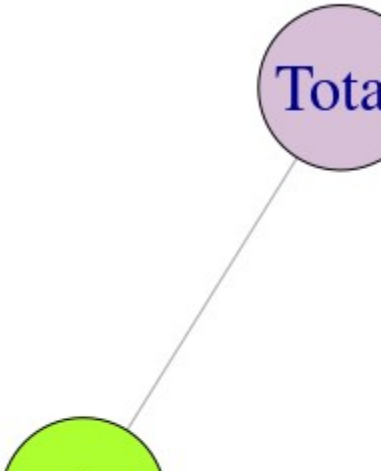
10.1 Hierarchical time series

Time series can often be naturally disaggregated by various attributes of interest. For example, the total number of bicycles sold by a cycling manufacturer can be disaggregated by product type such as: road bikes, mountain bikes, children bikes and hybrids. Each of these can be disaggregated into finer categories. For example children's bikes can be divided into balance bikes for children under the age of four, single speed bikes for children between the ages of four and six and other bikes for children over the age of six. Hybrid bikes can be divided into city, commuting, comfort, and trekking bikes; and so on. Such a collection of time series follow a hierarchical aggregation structure and we refer to these as *hierarchical time series*.

Commonly found in business and economics are hierarchical time series based on geographical locations. For example the total sales of a manufacturing company can be disaggregated by country, then within each country by state, within each state by region and so on down to the outlet level.

A feature that distinguishes hierarchical time series (from grouped time series that follow in Section 10.2) is that they have a unique structure with which they aggregate. Figure 10.1 shows a K=2-level hierarchical structure. At the top of the hierarchy, at level 0, is the "Total", the most aggregate level of the data. We denote as yt the tth observation of the "Total" series for t=1,…,T. The "Total" is disaggregated into two series at level 1 and each of these into three and two series respectively at the bottom-level of the hierarchy. Below the top most aggregate level, we denote as yj,t the tth observation of the series which corresponds to node j. For example yA,t denotes the tth observation of the series corresponding to node A at level 1, yAB,t denotes the tth observation of the series corresponding to node AB at level 2, and so on.

The total number of series in the hierarchy is n=1+2+5=8. We denote as m the number of series at the bottom-level, a dimension that is important in what follows. In this case m=5. Note that always n>m.



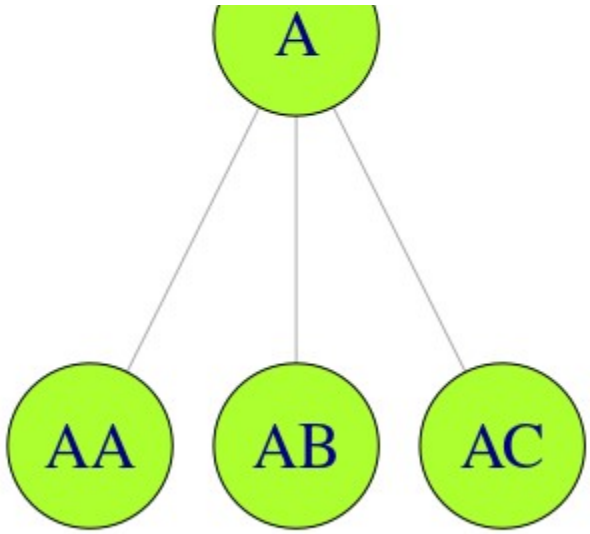


Figure 10.1: A two level hierarchical tree diagram.

For any time t, the observations at the bottom-level of the hierarchy will aggregate to the observations of the series above. For example, yt=yAA,t+yAB,t+yAC,t+yBA,t+yBB,t and yA,t=yAA,t+yAB,t+yAC,tandyB,t=yBA,t+yBB,t. Substituting (10.2) into (10.1) we also get . These equations can be thought of as aggregation constraints or summing equalities and can be more efficienlty represented using matrix notation. We construct an matrix S referred to as the *summing matrix* which dictates how the bottom-level series are aggregated, consistent with the aggregation structure. For the hierarchical structure in Figure 10.1 we write [ytyA,tyB,tyAA,tyAB,tyAC,tyBA,tyBB,t]=[1111111100000111000001000001000001000001] [yAA,tyAB,tyAC,tyBA,tyBB,t] or in more compact notation

where yt is a n-dimensional vector of all the observations in the hierarchy at time t, S is the summing matrix as defined above, and yK,t is an m-dimensional vector of all the observations in the bottom-level of the hierarchy at time t. Note that the first row in the summing S represents equation (10.1) above, the second and third row represent (10.2). The rows below these comprise an m-dimensional identity matrix Im so that each bottom-level observation on the right hand side of the equation is equal to itself in the left hand side.

Example: Australian tourism hierarchy

Australia is divided into eight geographical areas (some referred to as states and others as territories) with each one having its own government and some economic and administrative autonomy. Each of these can be further subdivided into smaller areas of interest referred to as zones. Business planners and tourism authorities are interested in forecasts for the whole of Australia, the states and the territories, and also the regions. In this example we concentrate on quarterly domestic tourism demand, measured as the number of visitor nights Australians spend away from home, for the six states of Australia, namely: New South Wales (NSW), Queensland (QLD), South Australia (SAU), Victoria (VIC), Western Australia (WA) and other (OTH). For each of these we consider visitor nights within the following zones.

State	Zones
NSW	Metro (NSWMetro), North Coast (NSWNthCo), South Coast (NSWSthCo), South Inner (NSWSthIn), North Inner (NSWNthIn)
QLD	Metro (QLDMetro), Central (QLDCntrl), North Coast (QLDNthCo)
SAu	Metro (SAUMetro), Costal (SAUCoast), Inner (SAUInner)
VIC	Metro (VICMetro), West Coast (VICWstCo), East Coast (VICEstCo), Inner (VICInner)
WAu	Metro (WAUMetro), Costal (WAUCoast), Inner (WAUInner)
OTH	Metro (OTHMetro), Non-Metro (OTHNoMet)

In summary, we consider five zones for NSW, four zones for VIC, and three zones for each QLD, SAU and WAU. Note that Metro zones contain the capital cities and surrounding areas around these generally considered to be metro areas. For OTH we consider Metro (OTHMetro) and non-Metro (OTHNoMet) areas across the rest of Australia. For further details on these geographical areas please refer to Appendix C in Wickramasuriya, Athanasopoulos, and Hyndman (2015).

To create a hierarchical time series we use the `hts` function as shown in the code below. The function requires as inputs the bottom-level time series and information about the hierarchical structure. `vn2` is a time series matrix containing the bottom-level series. There are alternative ways to pass to the function the structure of the hiearchy. In this case we are using the `characters` input. The first three characters of each column name of `vn2` capture the categories at the first level of the hierarchy (States). The following five characters capture the bottom-level categories (Zones).

```
vn2 <- read.csv("Ch10Data/vn2.csv")
vn2 <- ts(vn2, start = 1998, frequency = 4)
require(hts)
tourism.hts <- hts(vn2, characters = c(3, 5))
# I keep vn2 here as we will only use this line when we add vn2 in the fpp2 package
```

The top plot in Figure 10.2 shows the total number of visitor nights for the total of Australia while the plots bellow show the visitor nights disaggregated by state. These reveal diverse and rich dynamics at the aggregate national level and the first level of disaggregation across each state. The `aggts` function extracts time series from a `hts` object for any level of aggregation.

```
tourismL0 <- aggts(tourism.hts, levels = 0)
p1<-autoplot(tourismL0) +
  xlab("Year") +
  ylab("Visitor nights ('000)")+
  ggtitle("Total")

tourismL1 <- aggts(tourism.hts, levels = 1)
p2<-autoplot(tourismL1[,c(1,3,5)]) +
  xlab("Year") +
  ylab("Visitor nights ('000)")+
  scale_colour_discrete(guide = guide_legend(title = "State"))

p3<-autoplot(tourismL1[,c(2,4,6)]) +
  xlab("Year") +
  ylab("Visitor nights ('000)")+
  scale_colour_discrete(guide = guide_legend(title = "State"))

lay=rbind(c(1,1),c(2,3))
gridExtra::grid.arrange(p1, p2,p3, layout_matrix=lay)
```

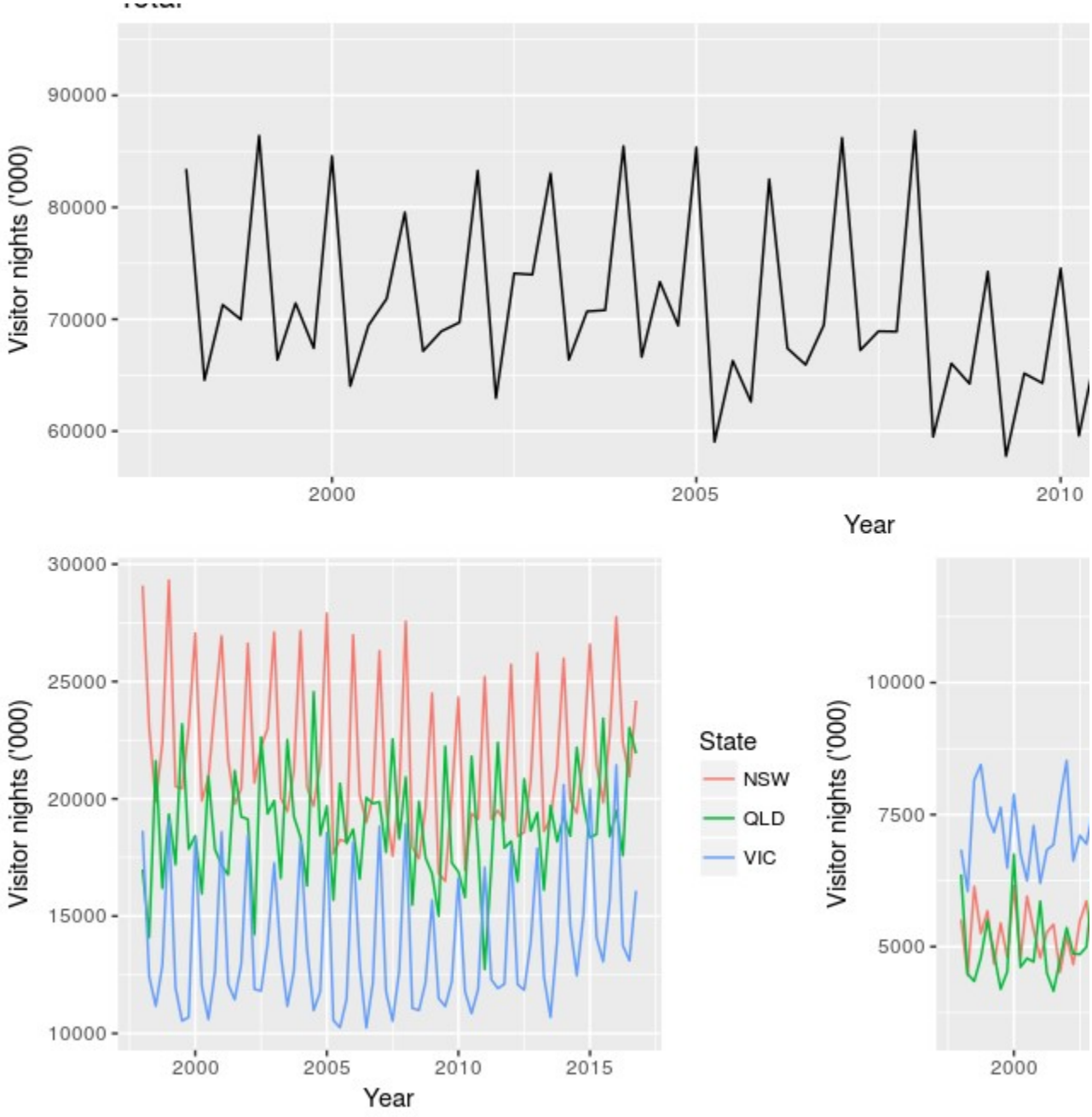
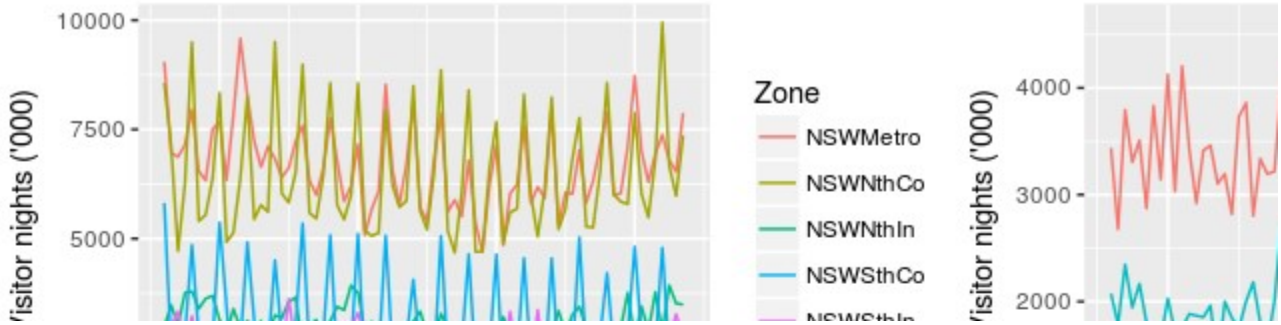


Figure 10.2: Australian domestic visitor nights over the period 1998 Q1 to 2016 Q4 disaggregated by State.

The plots in Figure 10.3 below show the bottom-level time series, i.e., the visitor nights for each zone. These help us visualise the diverse individual dynamics within each zones and assist in identifying unique and important time series. Notice for example the costal WAU zone which shows significant growth over the last few years.



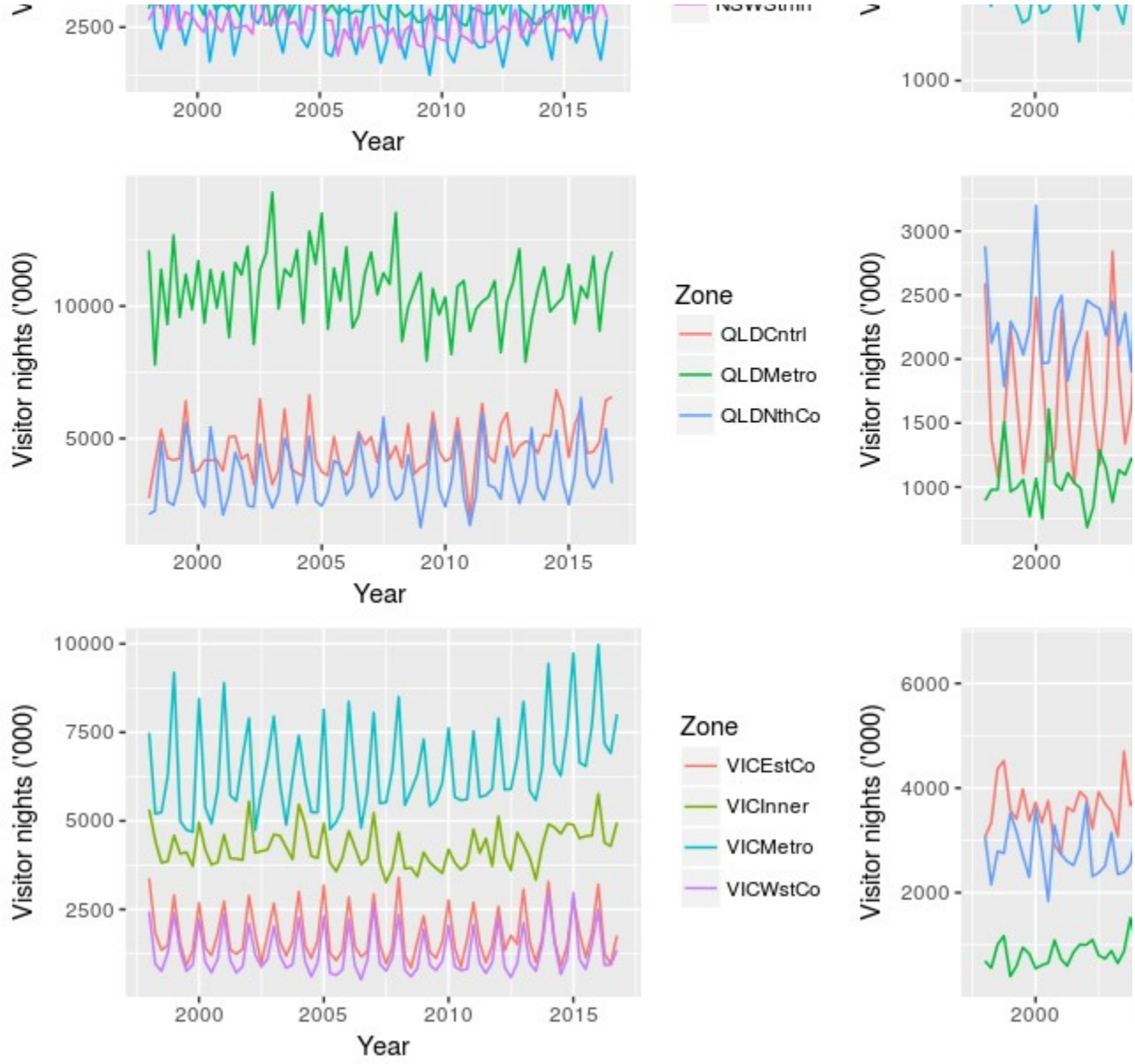
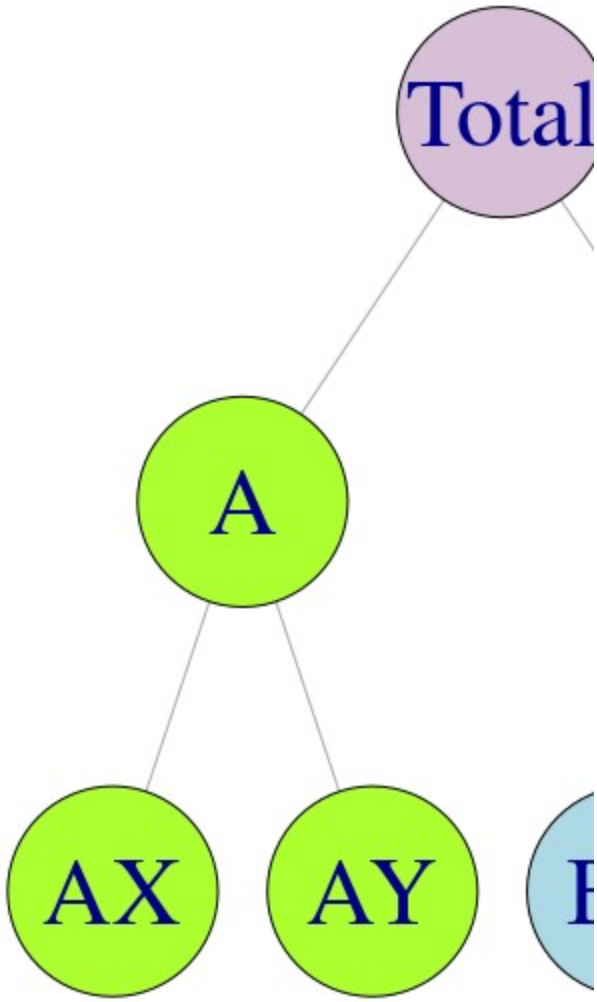


Figure 10.3: Australian domestic visitor nights over the period 1998 Q1 to 2016 Q4 disaggregated by Zones.

10.2 Grouped time series

Another possibility is that series can be naturally grouped together based on attributes without necessarily imposing a hierarchical structure. For example the bicycles sold by the warehouse can be for males, females or unisex. Frames can be carbon, aluminium or steel. They can be single speed or have multiple gears. We can also get a similar structure when we combine two hierarchical structures. For example the bicycle manufacturer may disaggregate sales by product and also by geographical location. We refer to these as *grouped time series*. With grouped time series we still have a hierarchical structure however the structure does not naturally disaggregate in a unique way. For example we can disaggregate the bicycles by product type and then geographical location but also vice versa.

Figure 10.4 below shows a K=2-level grouped structure. At the top of the grouped structure, is the "Total", the most aggregate level of the data, again represented by yt. The "Total" can be disaggregated by attributes (A, B) forming series yA,t and yB,t, or by attributes (X, Y) forming series yX,t and yY,t. At the bottom the data are disaggregated by both attributes.


Total
A
AX
AY


Total

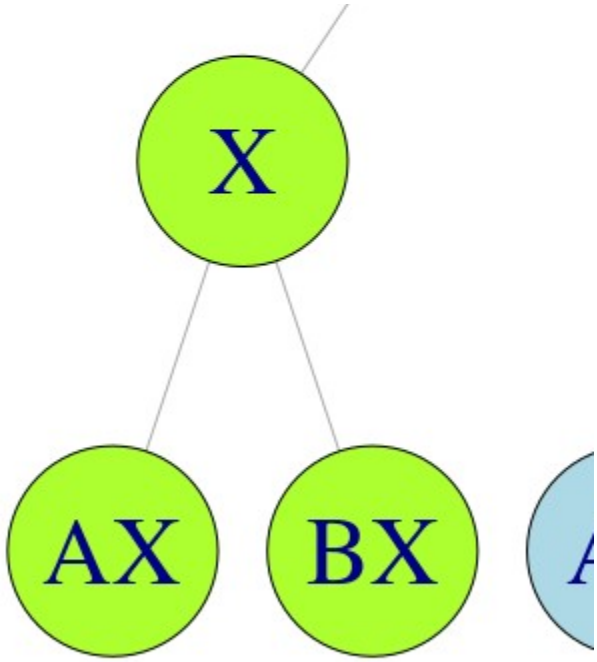


Figure 10.4: Alternative representations of a two level grouped structure.

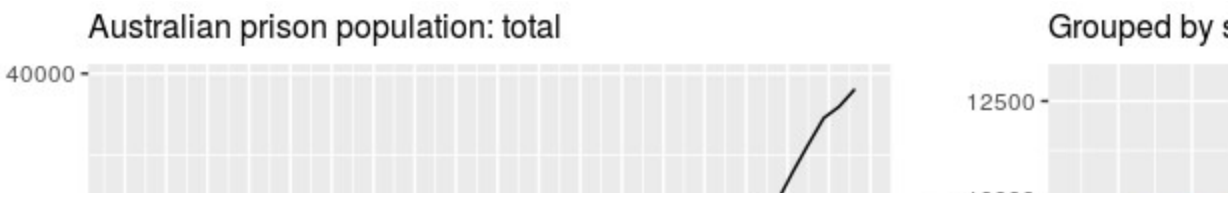
This example shows that there are alternative aggregation paths for grouped structures. For any time t, as with the hierarchical structure, yt=yAX,t+yAY,t+yBX,t+yBY,t. However, for the first level of the grouped structure, yA,t=yAX,t+yAY,tyB,t=yBX,t+yBY,t but also yX,t=yAX,t+yBX,tyY,t=yAY,t+yBY,t.

These equalities can again be represented by the summing matrix S which recall is of dimension n×m. The total number of series is n=9 with m=4 series at the bottom-level. For the grouped structure in Figure 10.4 we write [ytyA,tyB,tyX,tyY,tyAX,tyAY,tyBX,tyBY,t]= [111111000011101001011000010000100001][yAX,tyAY,tyBX,tyBY,t] where now the second and third rows of S represent (10.4) and the fourth and fifth rows represent (10.5).

Grouped time series can be thought of as hierarchical time series that do not impose a unique hierarchical structure in the sense that the order by which the series can be grouped is not unique.

Example: Australian prison population

The left plot in the top row of Figure 10.5 shows the total number of prisoners in Australia over the period 2005 Q1 to 2016 Q4. This represents the top-level series in the grouping structure. The rest of the plots show the prison population grouped by (i) state[17] (ii) legal status, whether prisoners have already been sentenced or are in remand waiting for a sentence, and (iii) gender.



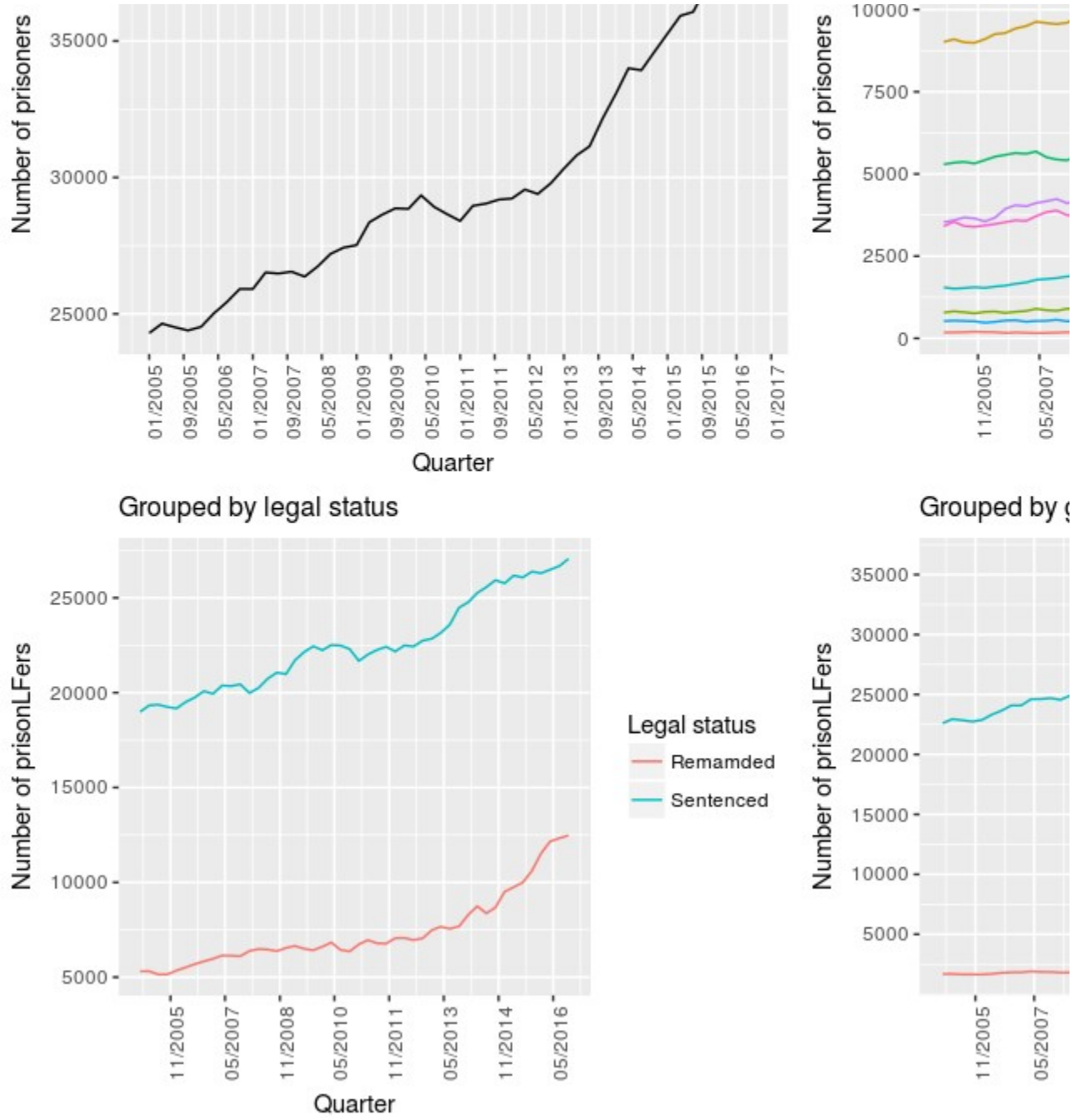


Figure 10.5: Total Australian adult prison population and Australian prison population grouped by state, by legal status and by gender.

To create a grouped time series we use the `gts` function as shown below. Similarly to the hts function the gts function requires as inputs the bottom-level time series and information about the grouping structure. `prison` is a time series matrix containing the bottom-level times series. Similarly to the hts function the information about the grouping structure can be passed in using the `characters` input. An alternative is to be more explicit about the labelling of the series and use the `groups` input. The code below shows examples for both these.

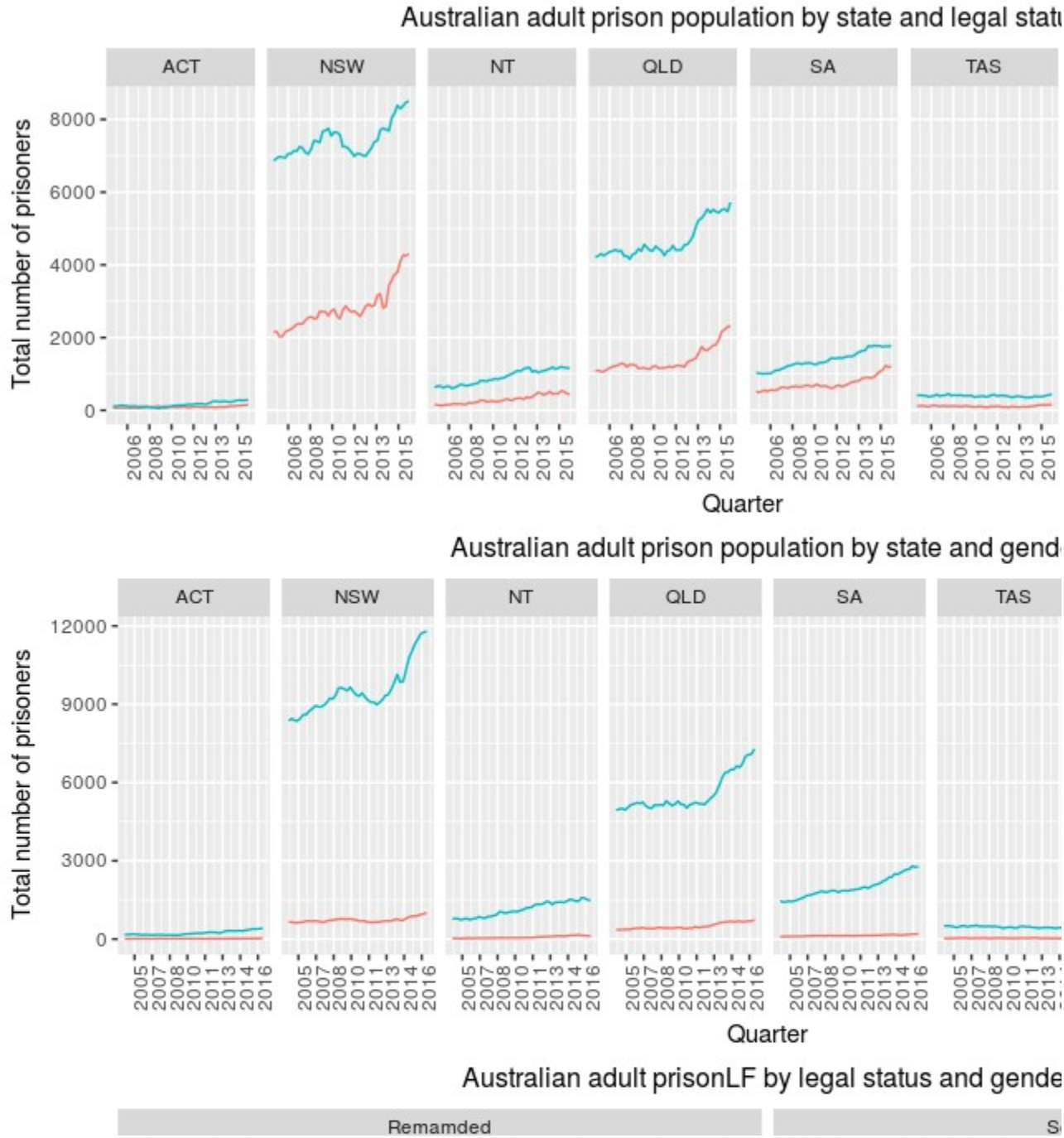
```
TODO: talk about interactions here - but possibly need to wait for update of the hts

prison <- read.csv("Ch10Data/prison.csv", strip.white = TRUE, check.names=FALSE)
prison <- ts(prison, start=c(2005,1), end=c(2016,4), frequency=4)
# Need to make data into prison time series matrix to add to fpp2
```

```
# Using the `characters` input
prison.gts <- gts(prison, characters = c(1,1,1))
#> Argument gnames is missing and the default labels are used.

# Using the `groups` input # 8 states, 2 legal, 2 gender
s <- rep(c("NSW", "VIC", "QLD", "SA", "WA", "NT", "ACT", "TAS"), 32/8)
l <- rep(rep(c("Rem","Sen"),each=8),2) # Sentenced is number 2 here
g <- rep(c("M","F"),each=32/2)
s_l <- as.character(interaction(s,l,sep=""))
s_g <- as.character(interaction(s,g,sep=""))
g_l <- as.character(interaction(g,l,sep=""))
gc <-rbind(s,l,g,s_l, s_g, g_l)
prison.gts <- gts(prison,groups=gc)
```

Figure 10.6 shows the Australian prison population grouped by all possible combinations of two attributes at a time. The top plot shows the prison population grouped by state and legal status, the middle panel shows the prison poplulation grouped by state and gender and the bottom panel shows the prison population grouped by legal status and gender.



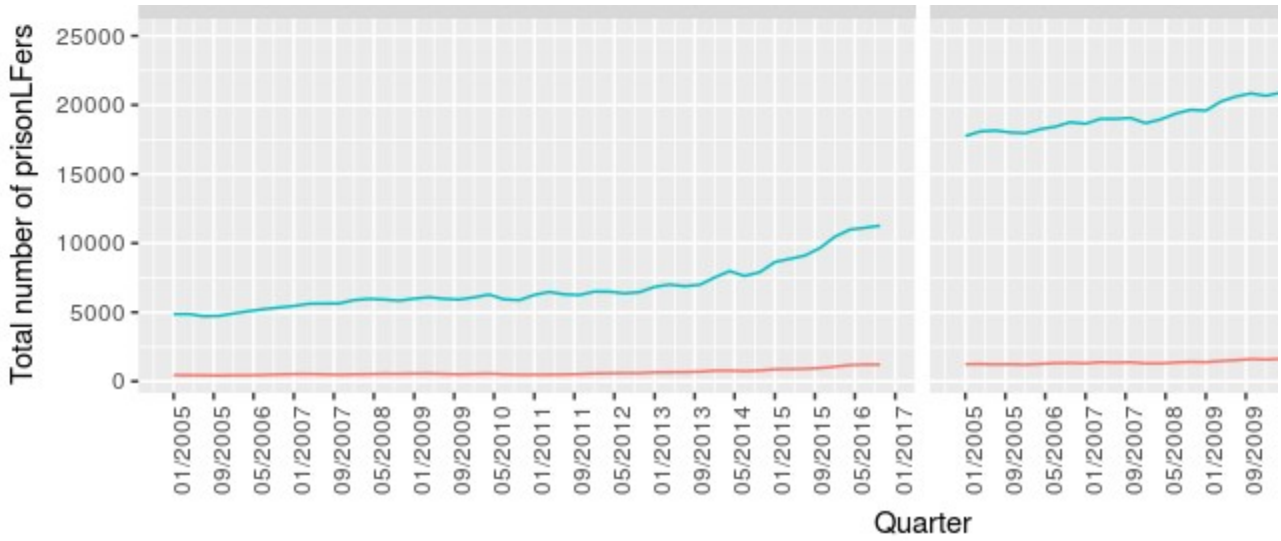


Figure 10.6: Australian adult prison population grouped by pairs of attributes.

Figure 10.7 shows the Australian adult population grouped by all three attributes: state, legal status and gender. These form the bottom-level series of the grouped structure for the Australian prison population.

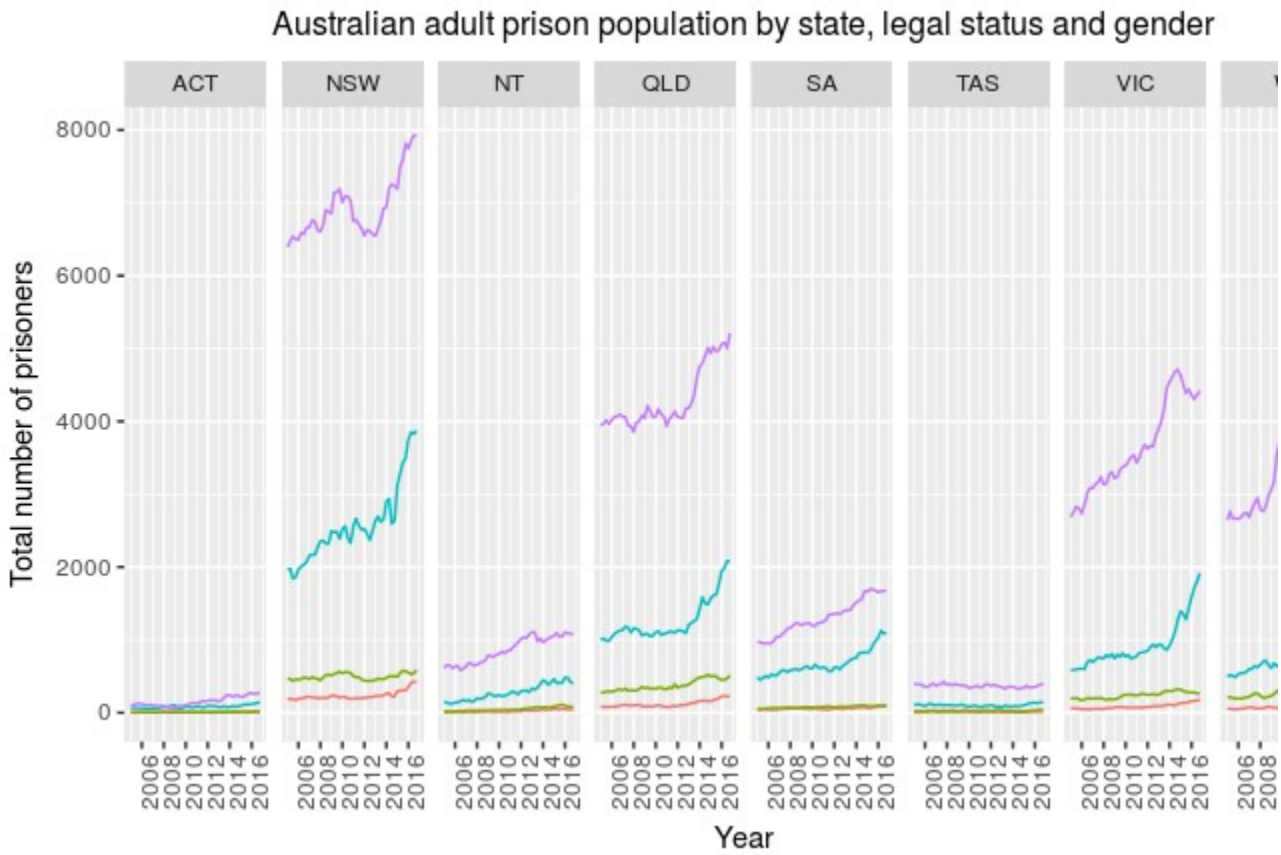


Figure 10.7: Bottom-level time series for the Australian adult prison population, grouped by state, legal status and gender.

10.3 Base and coherent forecasts

The aim is to generate a set of coherent forecasts across all the series of a hierarchical or a grouped structure.

Commonly used approaches are based on first generating forecasts for a single level of aggregation, and then producing coherent forecasts by aggregating these up using a bottom-up approach, disaggregating them down using a top-down approach, or a combination of the two using a middle-out approach. The details of these methods are presented in Sections 10.4 to 10.6. An alternative and a more advnanced approach is to first generate forecasts for all the time series and then reconcile these so that they become coherent. The details of this approach which we refer to as the optimal reconciliation approach are presented in Section 10.8.

In what follows in this section we introduce and formally define the concepts of base and coherent forecasts in order to build a unified framework that incorporates all of the approaches outlined above for both hierarchical and grouped time series.

Denote as $\hat{y}_h$ the h-step-ahead forecast generated for the "Total" series having observed the time series up to time T.[18] Likewise denote as $\hat{y}_{j,h}$ the h-step-ahead forecast generated for the series at node j having observed the time series up to time T. We refer to these as **base forecasts**. These are forecasts generated for each time series in the aggregation structure using a suitable forecasting method. The `hts` package has three inbuit options to produce base forecasts. These are controlled by the `fmethod` method argument

```
forecast(..., fmethod = c("ets", "arima", "rw"), ...).
```

as discussed in the Introduction of this Chapter 10.

Although the data follows exactly the aggregation structure (as reflected by the summing matrix S) base forecasts will generally not. It is only under very special circumstances such as using a very simple method to forecasts all the time series (for example using naïve forecasts for all series) that the base forecasts will be coherent.

The base forecasts can however be combined, as briefly described above, to produce a set of forecasts that are coherent. We denote the set of **coherent forecasts** for the "Total" series and all series below the top-level as $\tilde{y}_h$ and $\tilde{y}_{j,h}$ respectively.

10.4 The bottom-up approach

A commonly applied method for generating coherent forecasts is the bottom-up approach. This approach involves first generating base forecasts for each series at the bottom-level and then aggregating these upwards to produce forecasts for all the series in the structure.

For example, for the hierarchy of Figure 10.1 we first generate h-step-ahead base forecasts for each of the bottom-level series: $\hat{y}_{AA,h}$, $\hat{y}_{AB,h}$, $\hat{y}_{AC,h}$, $\hat{y}_{BA,h}$ and $\hat{y}_{BB,h}$.

Aggregating these up the hierarchy we get h-step-ahead coherent forecasts for the rest of the series: $\tilde{y}_h=\hat{y}_{AA,h}+\hat{y}_{AB,h}+\hat{y}_{AC,h}+\hat{y}_{BA,h}+\hat{y}_{BB,h}$, $\tilde{y}_{A,h}=\hat{y}_{AA,h}+\hat{y}_{AB,h}+\hat{y}_{AC,h}$ and $\tilde{y}_{B,h}=\hat{y}_{BA,h}+\hat{y}_{BB,h}$.

As in equation (10.3) we can employ the summing matrix here and write
$[\tilde{y}_h\tilde{y}_{A,h}\tilde{y}_{B,h}\tilde{y}_{AA,h}\tilde{y}_{AB,h}\tilde{y}_{AC,h}\tilde{y}_{BA,h}\tilde{y}_{BB,h}]=$
$[11111111000001110000100000100000100000 1][\hat{y}_{AA,h}\hat{y}_{AB,h}\hat{y}_{AC,h}\hat{y}_{BA,h}\hat{y}_{BB,h}]$.

In general, using more compact notation, the bottom-up approach can be represented as ˜yh=Sˆ yK,h where ~yt is an n-dimensional vector of coherent h-step-ahead forecasts of each time series within any aggregation structure and ˆyK,h is an m-dimensional vector of h-step-ahead base forecasts for each of the bottom-level series of any aggregation structure. Note that for the bottom-up approach the coherent forecasts for the bottom-level series are equal to the base forecasts, i.e., ˜yK,t=ˆyK,t.

The greatest advantage of this approach is that we are forecasting at the bottom-level of a structure and therefore no information is lost due to aggregation. On the other hand bottom-level data can be quite noisy and more challenging to model and forecast.

The bottom-up approach is implemented in the `forecast` package by setting

```
forecast(..., method = "bu", ...).
```

10.5 Top-down approaches

Top-down approaches involve first generating forecasts for the “Total” series on the top of the aggregation structure and then disaggregating these downwards. We let be a set of proportions which dictate how the base forecasts of the “Total” series are to be distributed to revised forecasts for each series at the bottom-level of the structure. For example for the hierarchy of Figure 10.1 using proportions we get,
˜yAA,t=p1ˆyt, ˜yAB,t=p2ˆyt, ˜yAC,t=p3ˆyt, ˜yBA,t=p4ˆyt and ˜yBB,t=p5ˆyt. Using matrix notation we can stack the set of proportions in a m-dimensional column vector [19] and write Once the bottom-level h-step-ahead forecasts have been generated these can be aggregated to generate coherent forecasts for the rest of the series. In general using the summing matrix and for a specified set of proportions, top-down approaches can be represented as, Note that for all top-down approaches the top-level coherent forecasts are equal to the top-level base forecasts, i.e., .

10.6 Middle-out approach

The middle-out approach combines bottom-up and top-down approaches. First the “middle level” is chosen and base forecasts are generated for all the series of this level. For the series above the middle level, coherent forecasts are generated using the bottom-up approach by aggregating the “middle-level” base forecasts upwards. For the series below the “middle level”, coherent forecasts are generated using a top-down approach by disaggregating the “middle level” base forecasts downwards.

This approach is implemented in the `forecast` package by setting

```
forecast(..., method = "mo",...).
```

10.7 The projection matrix

Denote as \(\bm{\hat{y}}_h\) a set of \(h\)-step-ahead base forecasts generated for each series in a hierarchial or grouped structure and stacked the same way as the data. For example for the hierarchy of Figure 10.1 \[\bm{\hat{y}}_h=\begin{bmatrix} \hat{y}_h \\ \yhat{A}{h} \\ \yhat{B}{h} \\ \yhat{AA}{h} \\ \yhat{AB}{h} \\ \yhat{AC}{h} \\ \yhat{BA}{h} \\ \yhat{BB}{h} \\ \end{bmatrix}. \]

In general, all forecasting approaches for either hierarchical or grouped structures can be represented as \[\begin{equation} \bm{\tilde{y}}_h=\bm{S}\bm{P}\bm{\hat{y}}_h \tag{10.6} \end{equation}\]

where reading from right to left, \(\bm{\hat{y}}_h\) is the set of \(h\)-step-ahead base forecasts as defined above, \(\bm{P}\) is a matrix that projects the base forecasts into the bottom-level, and the summing matrix \(\bm{S}\) sums these up using the aggregation structure to produce a set of coherent forecasts \(\bm{\tilde{y}}_h\).

The \(\bm{P}\) matrix is defined according to the approach implemented. For example if the bottom-up approach is used to forecast the hierarchy of Figure 10.1, \[\bm{P}= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ \end{bmatrix}. \] Notice that the \(\bm{P}\) comprises two partitions. The first three columns which zero out the base forecasts of the series above the bottom-level and the \(m\)-dimensional identity matrix which picks only the base forecasts of the bottom-level, to then be summed up the hierarchy by the \(\bm{S}\) matrix.

If any of the top-down approaches were used then \[\bm{P}= \begin{bmatrix} p_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ p_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ p_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ p_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ p_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ \end{bmatrix}. \] The first column includes a set of proportions that distribute the base forecasts of the top-level to the bottom-level. These are then summed up the hierarchy by the \(\bm{S}\) matrix. The rest of the columns zero out the base forecasts below the very top-level of aggregation.

For a middle out approach the \(\bm{P}\) matrix will be a combination of the above two. Using a set of proportions, the base forecasts of some pre-chosen level will be disaggregated to the bottom-level to then be summed up the hierarchy with the summing matrix, with all other base forecasts being zeroed out.

10.8 The optimal reconciliation approach

All the approaches we have considered so far in this chapter involve choosing a particular level in the aggregation structure, generating base forecasts for that level, and then either aggregating these up or disaggregating them down to generate coherent forecasts for the rest of the series. The above examples clearly reflect this through the P matrix.

In this section we introduce an approach that instead involves generating base forecasts for each series in the aggregation structure. The base forecasts are then reconciled to generate a set of coherent forecasts that are as close as possible to the base forecasts. Therefore we describe the resulting set of coherent forecasts as optimally reconciled coherent forecasts.

The general idea is derived from wanting to find a P matrix that minimises the forecast error of a set of coherent forecasts from forecasting each time series in the structure. In what follows we present a simplified summary of the approach. There are a few necessary steps that need to be followed in order to get a good flavour of the approach. These unfortunately complicate the presentation. However, following through these steps we get to the specification of P in equation (10.7) which is labelled the *MinT* estimator as it *Min*imises the *T*race of the forecast errors of the coherent forecasts across the whole structure. For further details and discussion please refer to Wickramasuriya, Athanasopoulos, and Hyndman (2015).

Let eT+h=yT+h−˜yh be the forecast errors after having produced a set of coherent forecasts across the whole structure, stacked in the same order as the data. Note that this expression is a

generalisation of a forecast error as defined in Section 3.4 using matrix notation. It can be easily shown through Equation (10.6) that for a set of base forecasts that are unbiased,[20] defining a P matrix such that SPS=S, generates a set of coherent forecasts ˜yh that are also unbiased. Note that this will not hold for any top-down approach.[21]

Wickramasuriya, Athanasopoulos, and Hyndman (2015) show in Lemma 1 that Var[yT+h−˜yh] =SPWhP′S′ where Wh=E[(yT+h−ˆyh)(yT+h−ˆyh)′] is the variance-covariance matrix of the h-step-ahead base forecast errors. This is a very important result as it shows that the forecast error variance of the coherent forecasts is a function of the error variance of base forecasts . The objective is to find a matrix P that minimises the error variance of the coherent forecasts. Wickramasuriya, Athanasopoulos, and Hyndman (2015) show in Theorem 1 that the optimal matrix P that minimises the such that , is given by P=(S′W−1hS)−1S′W−1h

referred to as the *MinT* estimator.

Note that the MinT estimator involves Wh the forecast error variance of the h-step-ahead base forecasts. As this is challenging to estimate we provide below three simplifying specifications which have been shown to work well in both simulations and in practice.

1. Set Wh=khI ∀h where kh>0.[22] This is the most simplifying assumption to make. Note that in this case P is independent of the data and no further estimation is required. The disadvantage is however that this specification does not account for the differences in the scale between the levels of the structure which naturally exist due to aggregation. The two specifications that follow do account for this. This approach is implemented in the `forecast` package by setting

   ```
   forecast(..., method = "comb", weights = "ols", ...).
   ```

   The weights here are referred to as the OLS (ordinary least squares) estimator as setting Wh=khI in (10.7) gives the least squares estimator we introduced in Section 5.7 with X=S and y=ˆy.

2. Set Wh=khdiag(ˆW1) ∀h where kh>0 and ˆW1=1TT∑t=1ˆetˆe′t where ˆet is an n-dimensional vector of residuals of the models that generated the base forecasts stacked in the same order as the data. Each element in this vector is the same as defined in Section 3.3. The approach is implemented by setting

   ```
   forecast(..., method = "comb", weights = "wls", ...).
   ```

   This specification scales the base forecasts using the variance of the residuals and it is therefore referred to as the WLS (weighted least squares) estimator using *variance scaling*.

3. Set Wh=khΛ, ∀h where kh>0 and Λ=diag(S1) where 1 is a unit column vector of dimension n. This specification assumes that the bottom-level base forecast errors each have variance kh and are uncorrelated between nodes. Hence each element of the diagonal Λ matrix contains the number of forecast error variances contributing to that aggregation level. This estimator only depends on the structure of the hierarchy or the grouped time series. It is therefore referred to as the specification that applies *structural scaling*. Notice that the structural scaling assumes equivariant forecast errors only at the bottom-level of the structure and not across all levels which is unrealistically assumed by the first specification. Furthermore, applying the structural scaling specification is particularly useful in cases where residuals are not available and therefore variance scaling cannot be applied. For example, in cases where the base forecasts are generated by judgemental forecasting introduced in Chapter 4. This approach is implemented by setting

```
forecast(..., method = "comb", weights = "nseries", ...).
```

4. An alterantive to the above simplifying specifications is to direclty estimate the full covariance matrix. The most obvious and simple way would be to use the sample covariance. This is implemented by setting

   ```
   forecast(..., method = "comb", weights = "mint", covariance = "sam", ...).
   ```

   However, for cases that m≥T this is not a good estimator. Instead we use a shrinkage estimator which shrinks the sample covariance to a diagonal matrix. This is implemented by setting

   ```
   forecast(..., method = "comb", weights = "mint", covariance = "shr", ...).
   ```

   For more details on MinT please refer to Wickramasuriya, Athanasopoulos, and Hyndman (2015).

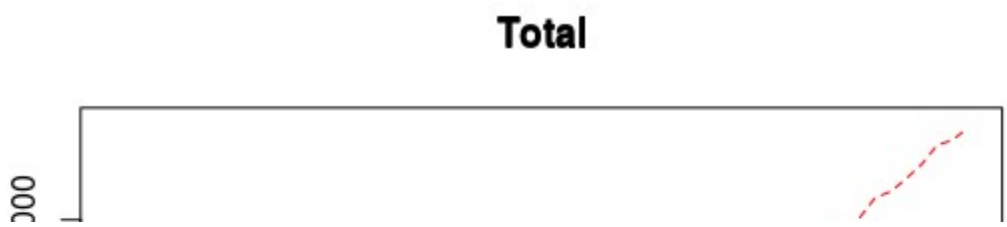
In summary, unlike any other existing approach, the optimal reconciliation forecasts are generated using all the information available within a hierarchical or a grouped structure. This is very important as particular aggregation levels or groupings may reveal features of the data that are of interest to the user and are important to be modelled. These features may be completely hidden or not easily identifiable at other levels. For example, consider a hierarchical structure reflecting the geographical division of a country into states, regions, down to a very fine grid of statistical local areas. There are significant differences between the seasonal patterns in the number of tourists visiting a state or a region that is mainly seen as a summer destination versus a state or a region that caters for winter activities. These differences will be smoothed at the country level due to aggregation and on the other hand it may be extremely challenging to identify at the very bottom-level of a statistical local area due to noise. Another example for a grouped structure is the difference in the sales of clothes between genders. Such differences will be completely smoothed out at the very top-level of aggregation considering total sales, or may be very challenging to identify due to noise at the very bottom-level.

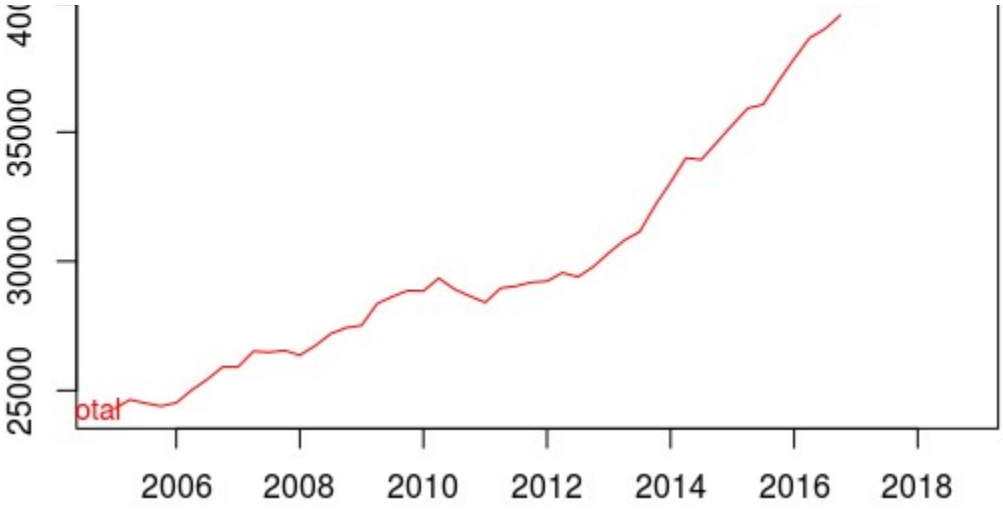
Example: Forecasting Australian prison population (continued)

Figure 10.8 plots coherent forecasts generated by the optimal reconciliation approach with the WLS estimator that uses variance scaling.

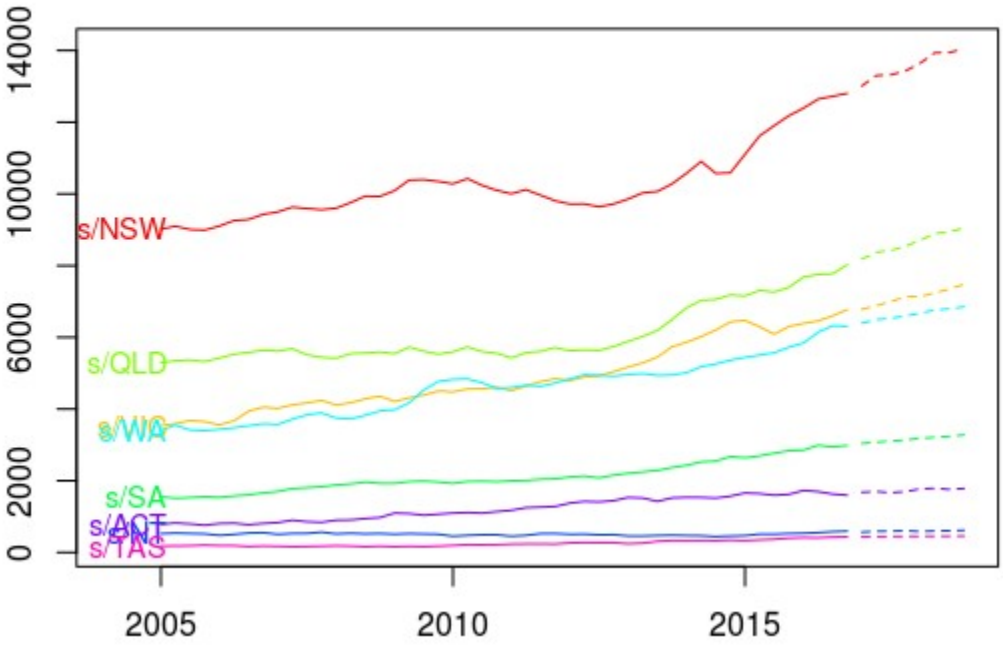
```
fcsts = forecast(prison.gts, h = 8, method = "comb", weights = "wls", fmethod = "ets")

plot(fcsts,levels = 0, color_lab=TRUE)
title(main = "Total")
plot(fcsts,levels = 1, color_lab=TRUE)
title(main = "Grouped by state")
plot(fcsts,levels = 2, color_lab=TRUE)
title(main = "Grouped by leagal status")
plot(fcsts,levels = 3, color_lab=TRUE)
title(main = "Grouped by gender")
```

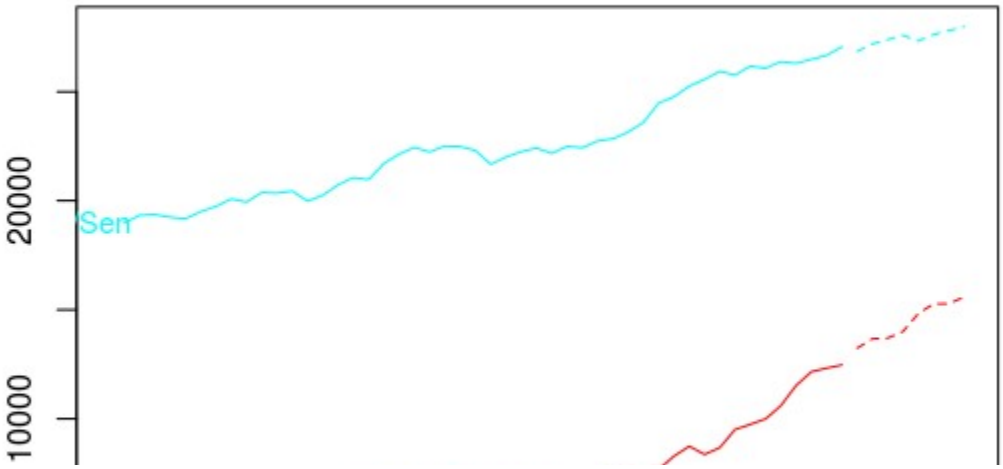




Grouped by state



Grouped by leagal status



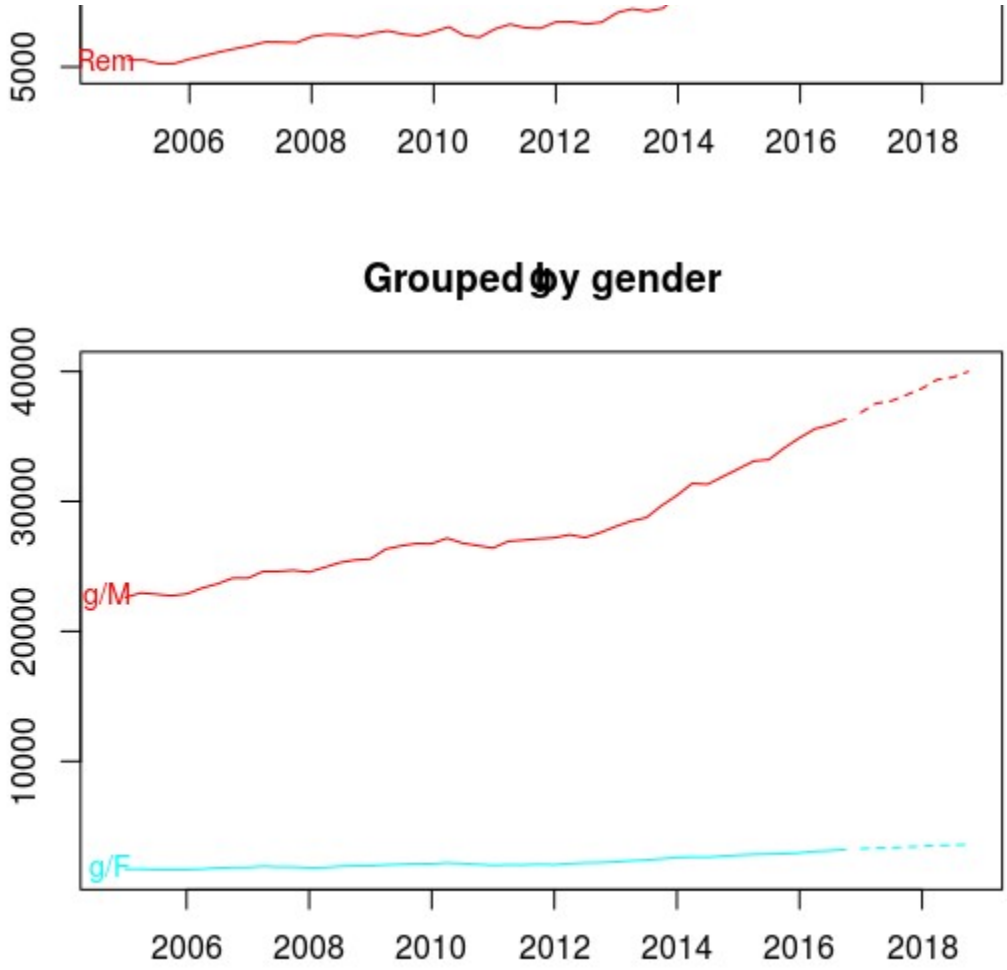
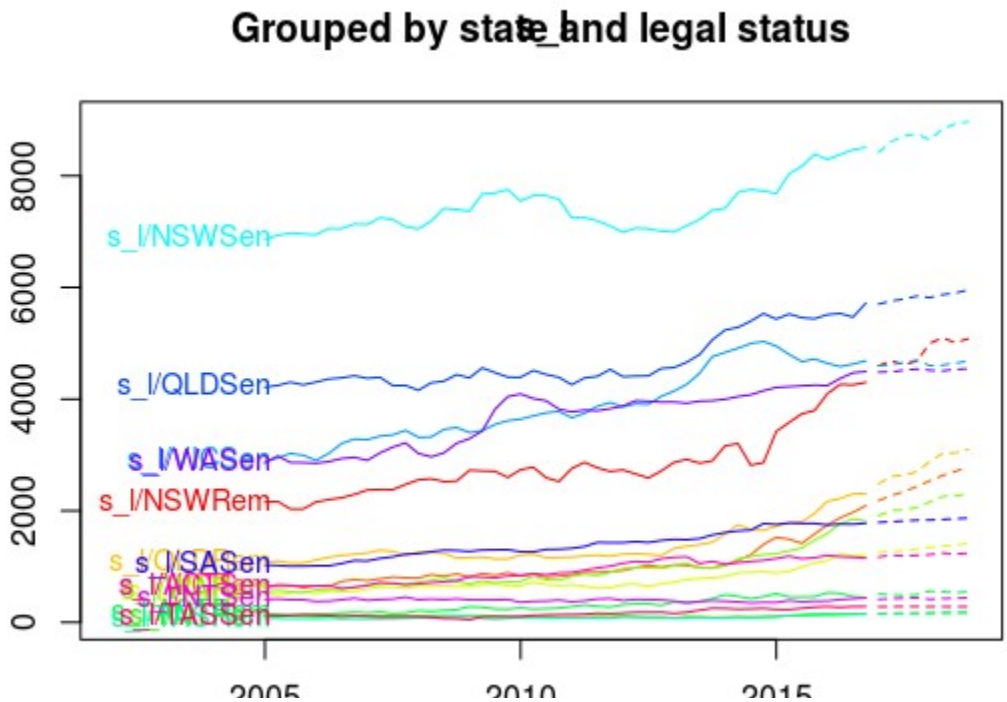
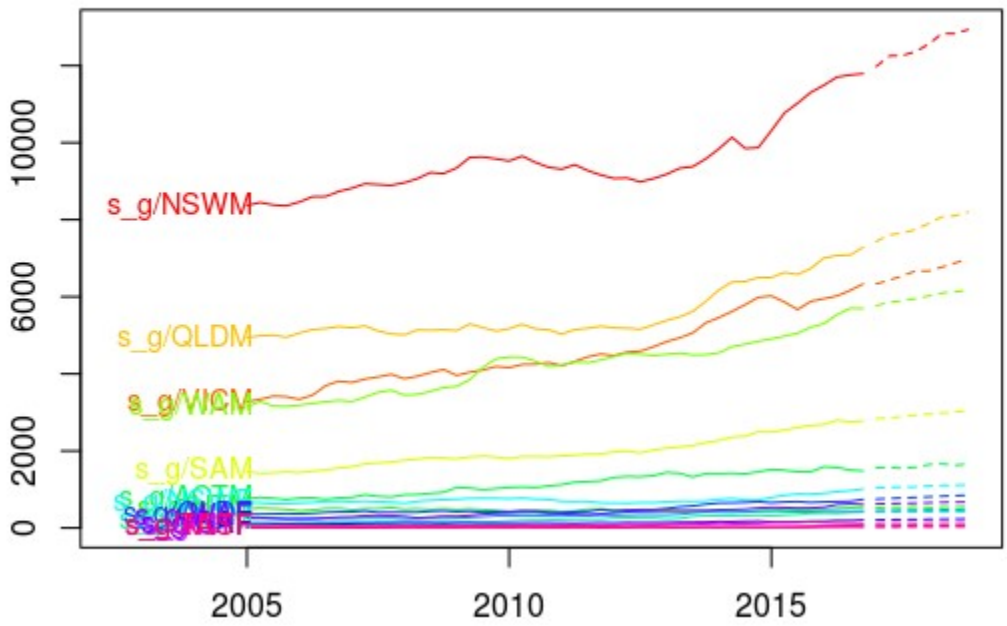
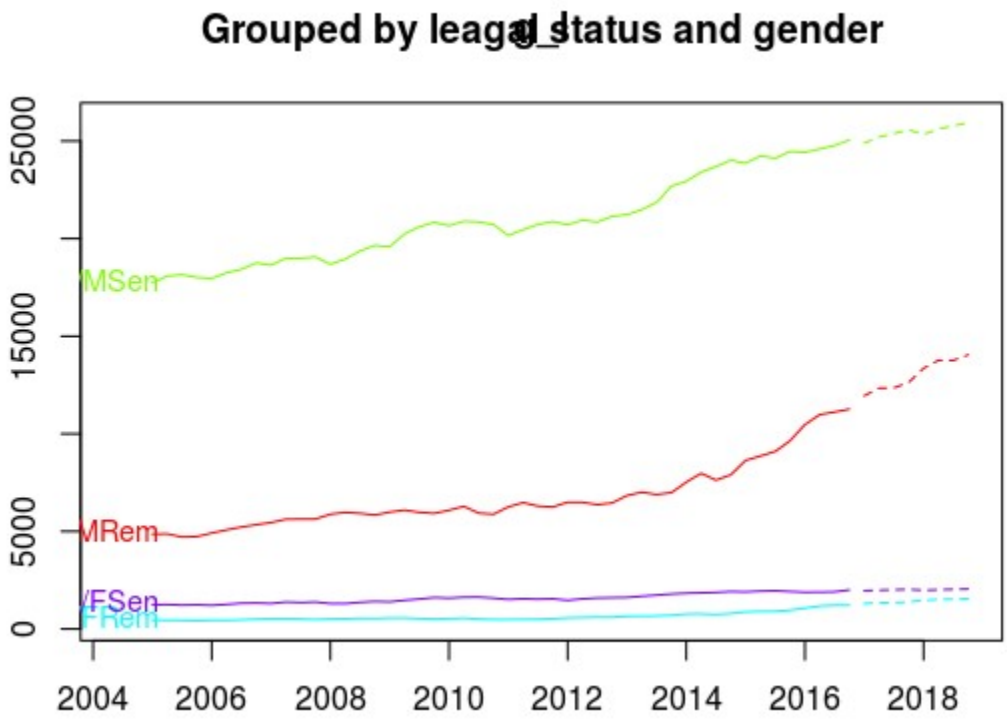


Figure 10.8: 8-step-ahead forecasts for the total Australian adult prison population andfor the population grouped by state, by legal status and by gender.

Figures 10.9 and 10.10 plot the coherent forecasts for all the interactions of the attributes.




Grouped by state and gender
s_g/NSWM
s_g/QLDM
s_g/SAM
0
2000
6000
10000
2005
2010
2015


Grouped by league status and gender
0
5000
15000
25000
2004
2006
2008
2010
2012
2014
2016
2018


Bottom
8000

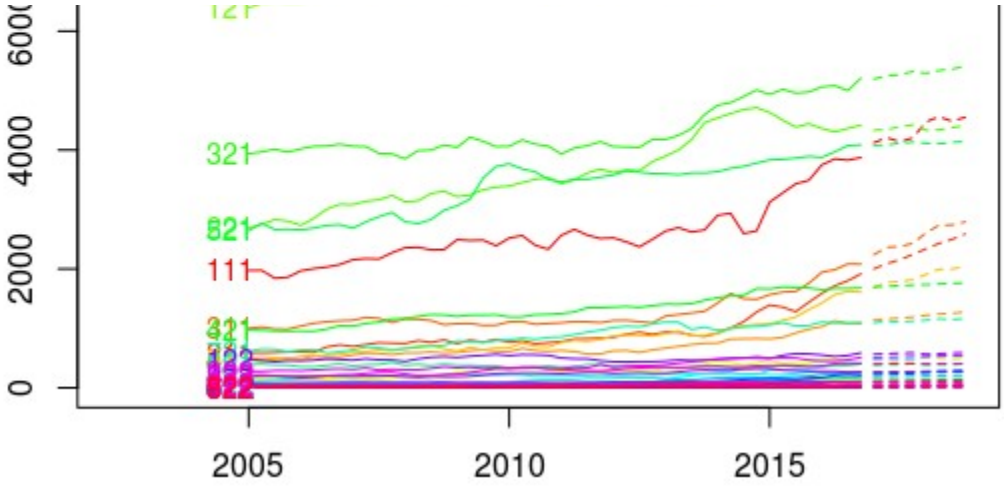


Figure 10.9: Forecasts for the Australian adult prison population grouped by pairs of attributes..

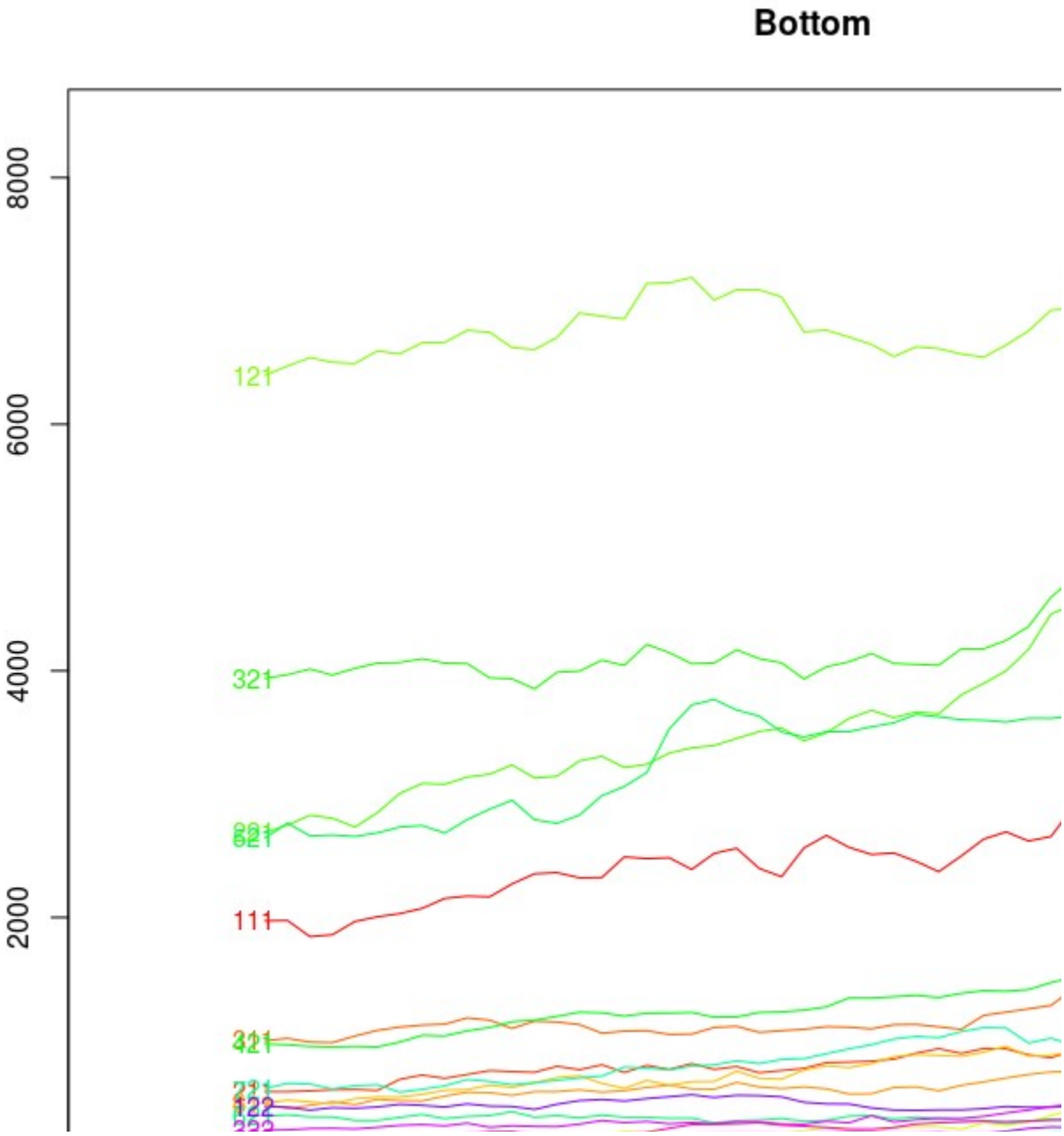


Figure 10.10: Forecasts for the bottom-level time series of the Australian adult prison population, grouped by state, legal status and gender.

10.9 Exercises

1. Exercise: write out the S matrices for the Australian tourism hierarchy and the Australian grouped structure. Use the `smatrix` command to verify your answers.

2. Generate forecasts from the bottom up approach for the tourism data. Plot the forecatsts and comment on there nature. Play with the window.

3. Set as training set and test set and compare the forecast accuracy of bu v opt.

Chapter 11 Advanced forecasting methods

In this chapter, we briefly discuss four more advanced forecasting methods that build on the models discussed in earlier chapters.

11.1 Complex seasonality

So far, we have considered relatively simple seasonal patterns such as quarterly and monthly data. However, higher frequency data often exhibits more complicated seasonal patterns. For example, daily data may have a weekly pattern as well as an annual pattern. Hourly data usually has three types of seasonality: a daily pattern, a weekly pattern, and an annual pattern. Even weekly data can be difficult as it typically has an annual pattern with seasonal period of $365.25/7 \approx 52.179$ on average.

Such multiple seasonal patterns are becoming more common with high frequency data recording. Further examples where multiple seasonal patterns can occur include call volume in call centres, daily hospital admissions, requests for cash at ATMs, electricity and water usage, and access to computer web sites.

Most of the methods we have considered so far are unable to deal with these seasonal complexities. Even the `ts` class in R can only handle one type of seasonality, which is usually assumed to take integer values.

To deal with such series, we will use the `msts` class which handles multiple seasonality time series. Then you can specify all of the frequencies that might be relevant. It is also flexible enough to handle non-integer frequencies.

You won't necessarily want to include all of these frequencies — just the ones that are likely to be present in the data. For example, if you have only 180 days of data, you can probably ignore the annual seasonality. If the data are measurements of a natural phenomenon (e.g., temperature), you might also be able to ignore the weekly seasonality.

Figure 11.1 shows the number of retail banking call arrivals per 5-minute interval between 7:00am and 9:05pm each weekday over a 33 week period. There is a strong daily seasonal pattern with frequency 169, and a weak weekly seasonal pattern with frequency 169×5=845. (Call volumes on Mondays tend to be higher than the rest of the week.) If a longer series of data were available, there may also be an annual seasonal pattern.

```
p1 <- autoplot(calls) +
  ylab("Call volume") + xlab("Weeks") +
  scale_x_continuous(breaks=seq(1,33,by=2))
p2 <- autoplot(window(calls, end=4)) +
  ylab("Call volume") + xlab("Weeks") +
  scale_x_continuous(minor_breaks = seq(1,4,by=0.2))
gridExtra::grid.arrange(p1,p2)
```

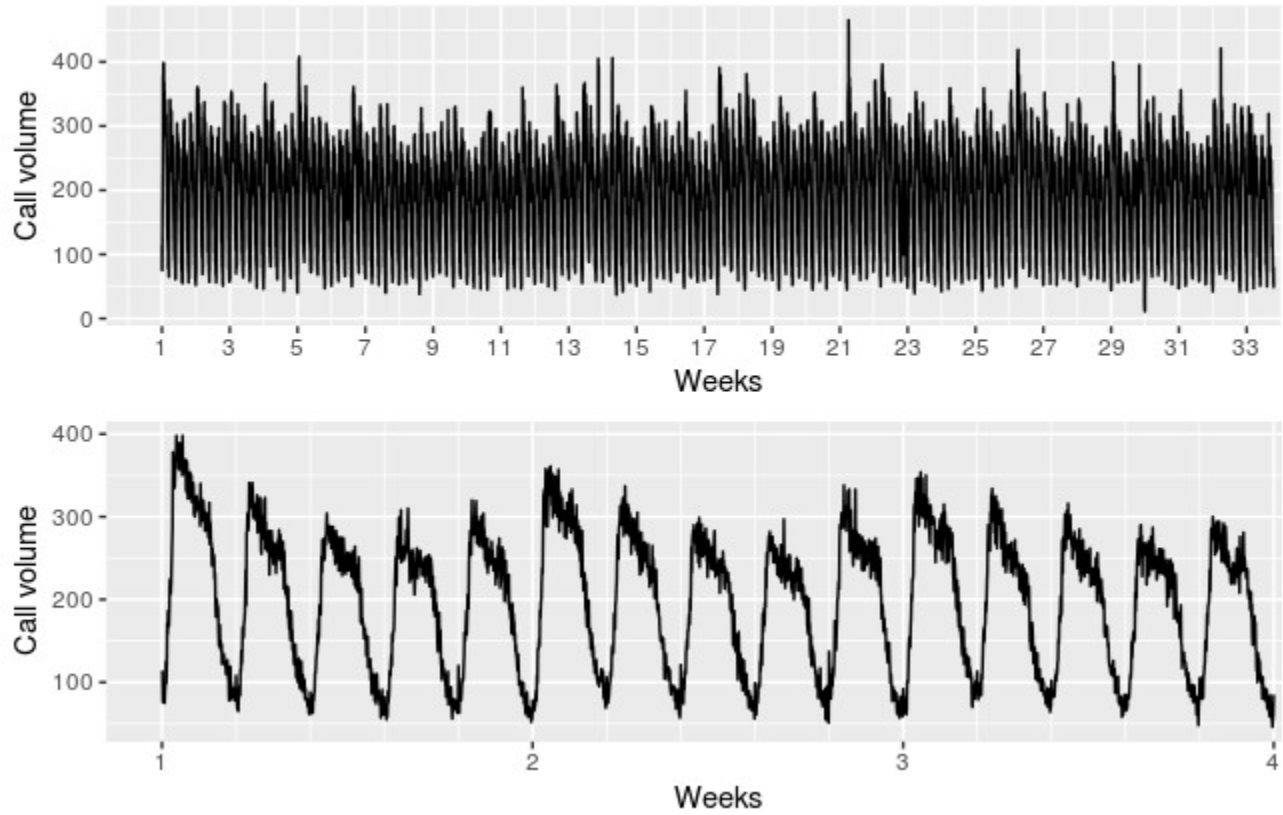


Figure 11.1: Five-minute call volume handled on weekdays between 7am and 9:05pm in a large North American commercial bank. Top panel shows data from 3 March 2003 to 23 May 2003. Bottom panel shows only the first three weeks.

Dynamic harmonic regression with multiple seasonal periods

With multiple seasonalities, we can use Fourier terms as we did in earlier chapters. Because there are multiple seasonalities, we need to add Fourier terms for each seasonal period. In this case, the seasonal periods are 169 and 845, so the Fourier terms are of the form $\sin(2\pi kt169)$,$\cos(2\pi kt169)$,$\sin(2\pi kt845)$,and$\cos(2\pi kt845)$, for $k=1,2,\ldots$. The `fourier` function can generate these for you.

We will fit a dynamic regression model with an ARMA error structure. The total number of Fourier terms for each seasonal period have been chosen to minimize the AICc. We will use a log transformation (`lambda=0`) to ensure the forecasts and prediction intervals remain positive.

```
#fit <- auto.arima(calls, seasonal=FALSE, lambda=0,
#         xreg=fourier(calls, K=c(10,10)))
#fc <- forecast(fit, xreg=fourier(calls, K=c(10,10), h=2*169))
#autoplot(fc, include=5*169) +
#  ylab("Call volume") + xlab("Weeks")
```

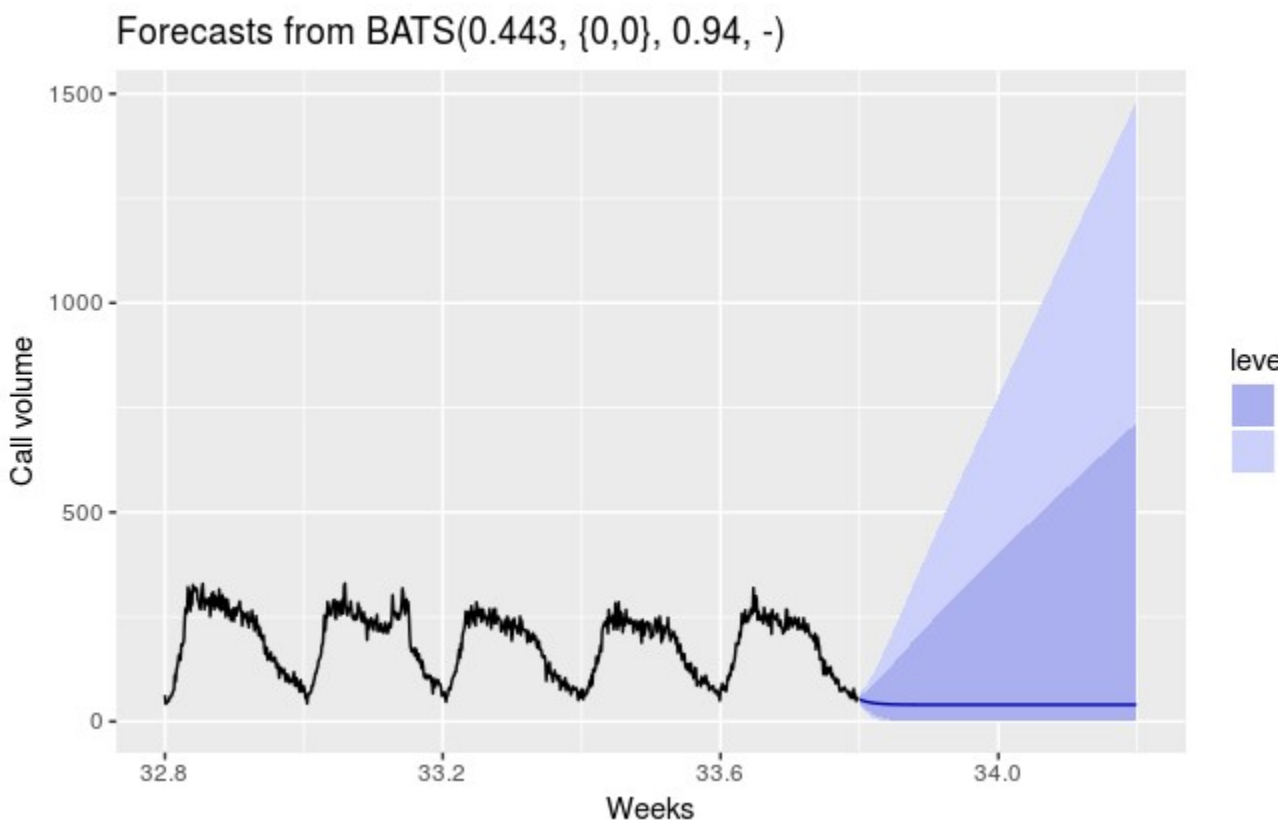
This is a very large model (containing 120+??=?? parameters).

TBATS models

An alternative approach developed by De Livera, Hyndman, and Snyder (2011) uses a combination of Fourier terms with an exponential smoothing state space model and a Box-Cox transformation, in a completed automated manner. As with any automated modelling framework, it does not always work, but it can be a useful approach in some circumstances.

A TBATS model differs from dynamic harmonic regression in that the seasonality is allowed to change slowly over time in a TBATS model, while harmonic regression terms force the seasonal patterns to repeat periodically without changing. One drawback of TBATS models is that they can be very slow to estimate, especially with long time series. So we will consider a subset of the `calls` data to save time.

```
calls %>%
  subset(start=length(calls)-1000) %>%
  tbats -> fit2
fc2 <- forecast(fit2, h=2*169)
autoplot(fc2, include=5*169) +
  ylab("Call volume") + xlab("Weeks")
```

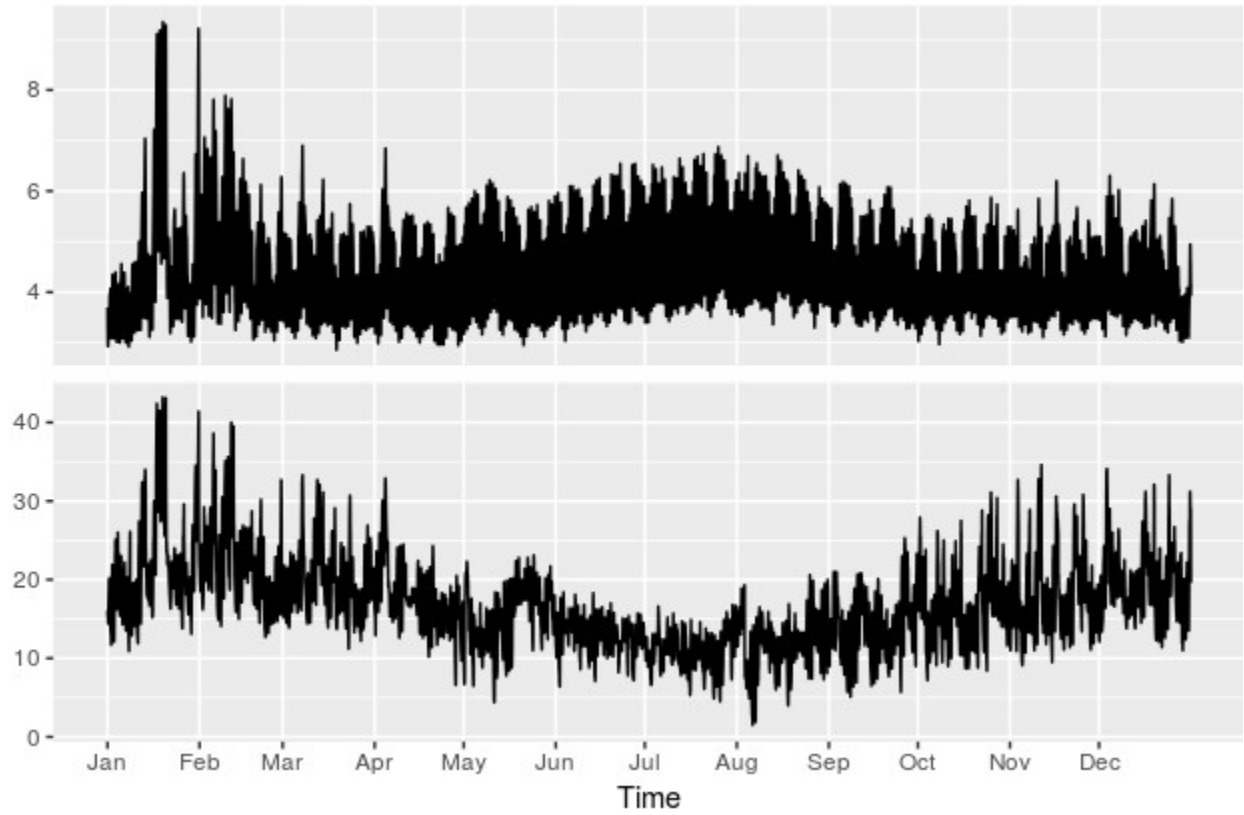


Complex seasonality with covariates

TBATS models do not allow for covariates, although they can be included in dynamic harmonic regression models. One common application of such models is electricity demand modelling.

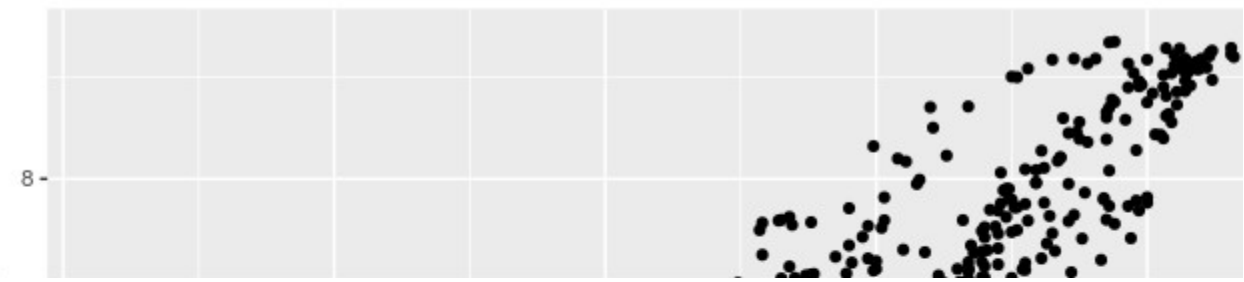
Figure ?? shows half-hourly electricity demand in Victoria, Australia, during 2014, along with temperatures in the same period.

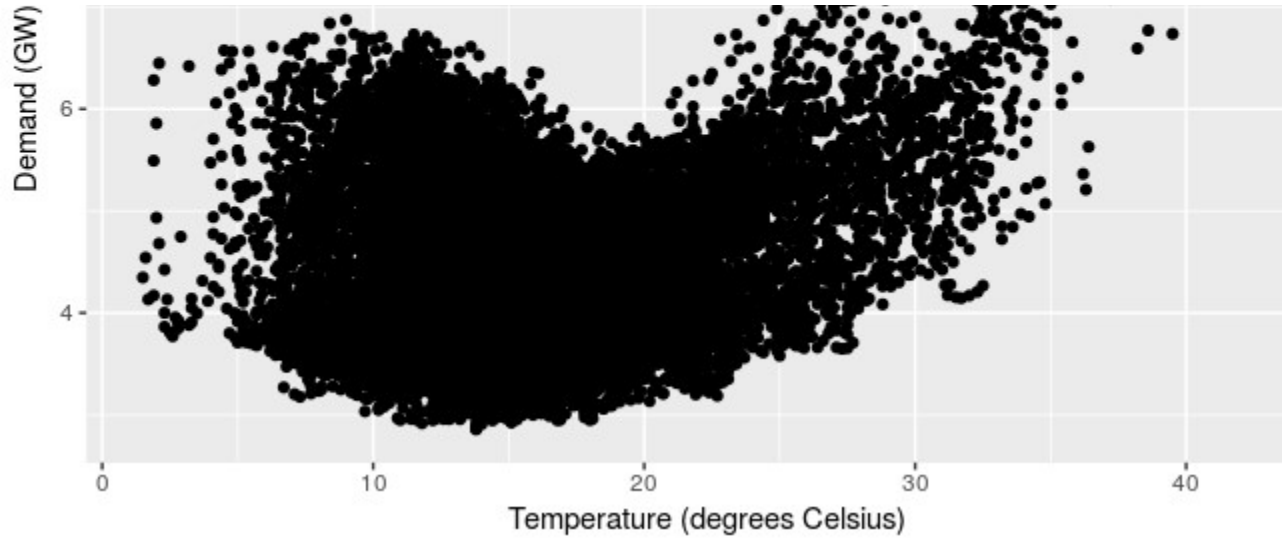
```
autoplot(elecdemand[,c("Demand","Temperature")], facet=TRUE) +
  scale_x_continuous(minor_breaks=NULL,
    breaks=2014+cumsum(c(0,31,28,31,30,31,30,31,31,30,31,30))/365,
    labels=month.abb) +
  xlab("Time") + ylab("")
```



Plotting electricity demand against temperature shows that there is a nonlinear relationship between the two, with demand increasing for low temperatures (due to heating) and increasing for high temperatures (due to cooling).

```
elecdemand %>%
  as.data.frame %>%
  ggplot(aes(x=Temperature, y=Demand)) + geom_point() +
    xlab("Temperature (degrees Celsius)") +
    ylab("Demand (GW)")
```





We will fit a regression model with a piecewise linear function of temperature (containing a knot at 18 degrees), and harmonic regression terms to allow for the daily seasonal pattern.

```
cooling <- pmax(elecdemand[,"Temperature"], 18)
fit <- auto.arima(elecdemand[,"Demand"],
         xreg = cbind(fourier(elecdemand, c(10,10,0)),
               heating=elecdemand[,"Temperature"],
               cooling=cooling))
```

Forecasting with such models is difficult because we require future values of the predictor variables. Future values of the Fourier terms are easy to compute, but future temperatures are, of course, unknown. We could use temperature forecasts obtain from a meteorological model if we are only interested in forecasting up to a week ahead. Alternatively, we could use scenario forecasting and plug in possible temperature patterns. In the following example, we have used a repeat of the last week of temperatures to generate future possible demand values.

```
#temps <- subset(elecdemand[,"Temperature"], start=NROW(elecdemand)-7*48-1)
#fc <- forecast(fit, xreg=cbind(fourier(temps, c(10,10,0)),
#                           heating=temps, cooling=pmax(temps,18)))
#autoplot(fc)

#checkresiduals(fc)
```

11.2 Forecasting counts

croston

Poisson models?

Other intermittent demand models

11.3 Vector autoregressions

One limitation with the models we have considered so far is that they impose a unidirectional relationship — the forecast variable is influenced by the predictor variables, but not vice versa. However, there are many cases where the reverse should also be allowed for — where all variables affect each other. In Chapter 9, the changes in personal consumption expenditure (Ct) were forecast based on the changes in personal disposable income (It). However, in this case a bi-

directional relationship may be more suitable: an increase in It will lead to an increase in Ct and vice versa.

An example of such a situation occurred in Australia during the Global Financial Crisis of 2008 –2009. The Australian government issued stimulus packages that included cash payments in December 2008, just in time for Christmas spending. As a result, retailers reported strong sales and the economy was stimulated. Consequently, incomes increased.

Such feedback relationships are allowed for in the vector autoregressive (VAR) framework. In this framework, all variables are treated symmetrically. They are all modelled as if they influence each other equally. In more formal terminology, all variables are now treated as "endogenous". To signify this we now change the notation and write all variables as ys: y1,t denotes the tth observation of variable y1, y2,t denotes the tth observation of variable y2, and so on.

A VAR model is a generalisation of the univariate autoregressive model for forecasting a collection of variables; that is, a vector of time series.[23] It comprises one equation per variable considered in the system. The right hand side of each equation includes a constant and lags of all the variables in the system. To keep it simple, we will consider a two variable VAR with one lag. We write a 2-dimensional VAR(1) as
$y_{1,t}=c_1+\phi_{11,1}y_{1,t-1}+\phi_{12,1}y_{2,t-1}+e_{1,t}y_{2,t}=c_2+\phi_{21,1}y_{1,t-1}+\phi_{22,1}y_{2,t-1}+e_{2,t}$

where $e_{1,t}$ and $e_{2,t}$ are white noise processes that may be contemporaneously correlated. Coefficient $\phi_{ii,\ell}$ captures the influence of the ℓth lag of variable yi on itself, while coefficient $\phi_{ij,\ell}$ captures the influence of the ℓth lag of variable yj on yi.

If the series modelled are stationary we forecast them by directly fitting a VAR to the data (known as a "VAR in levels"). If the series are non-stationary we take differences to make them stationary and then we fit a VAR model (known as a "VAR in differences"). In both cases, the models are estimated equation by equation using the principle of least squares. For each equation, the parameters are estimated by minimising the sum of squared ei,t values.

The other possibility which is beyond the scope of this book and therefore we do not explore here, is that series may be non-stationary but they are cointegrated, which means that there exists a linear combination of them that is stationary. In this case a VAR specification that includes an error correction mechanism (usually referred to as a vector error correction model) should be included and alternative estimation methods to least squares estimation should be used.[24]

Forecasts are generated from a VAR in a recursive manner. The VAR generates forecasts for *each* variable included in the system. To illustrate the process, assume that we have fitted the 2-dimensional VAR(1) described in equations ?? – for all observations up to time T. Then the one-step-ahead forecasts are generated by
$\hat{y}_{1,T+1|T}=\hat{c}_1+\hat{\phi}_{11,1}y_{1,T}+\hat{\phi}_{12,1}y_{2,T}\hat{y}_{2,T+1|T}=\hat{c}_2+\hat{\phi}_{21,1}y_{1,T}+\hat{\phi}_{22,1}y_{2,T}$. This is the same form as ?? – except that the errors have been set to zero and parameters have been replaced with their estimates. For , the forecasts are given by
$\hat{y}_{1,T+2|T}=\hat{c}_1+\hat{\phi}_{11,1}\hat{y}_{1,T+1}+\hat{\phi}_{12,1}\hat{y}_{2,T+1}\hat{y}_{2,T+2|T}=\hat{c}_2+\hat{\phi}_{21,1}\hat{y}_{1,T+1}+\hat{\phi}_{22,1}\hat{y}_{2,T+1}$.

Again, this is the same form as ?? – except that the errors have been set to zero, parameters have been replaced with their estimates, and the unknown values of y1 and y2 have been replaced with their forecasts. The process can be iterated in this manner for all future time periods.

There are two decisions one has to make when using a VAR to forecast. They are, how many variables (denoted by K) and how many lags (denoted by p) should be included in the system. The number of coefficients to be estimated in a VAR is equal to K+pK2 (or 1+pK per equation). For example, for a VAR with K=5 variables and p=3 lags, there are 16 coefficients per equation

making for a total of 80 coefficients to be estimated. The more coefficients to be estimated the larger the estimation error entering the forecast.

In practice it is usual to keep K small and include only variables that are correlated to each other and therefore useful in forecasting each other. Information criteria are commonly used to select the number of lags to be included.

VARs are implemented in the **vars** package in R. It contains a function `VARselect` to choose the number of lags p using four different information criteria: AIC, HQ, SC and FPE. We have met the AIC before, and SC is simply another name for the BIC (SC stands for Schwarz Criterion after Gideon Schwarz who proposed it). HQ is the Hannan-Quinn criterion and FPE is the "Final Prediction Error" criterion.[25] Care should be taken using the AIC as it tends to choose large numbers of lags. Instead, for VAR models, we prefer to use the BIC.

A criticism VARs face is that they are atheoretical. They are not built on some economic theory that imposes a theoretical structure to the equations. Every variable is assumed to influence every other variable in the system, which makes direct interpretation of the estimated coefficients very difficult. Despite this, VARs are useful in several contexts:

- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

Example: A VAR model for forecasting US consumption

```
library(vars)

VARselect(uschange[,1:2], lag.max=8, type="const")[["selection"]]
#> AIC(n)  HQ(n)  SC(n) FPE(n)
#>      5      1      1      5

var <- VAR(uschange[,1:2], p=3, type="const")
serial.test(var, lags.pt=10, type="PT.asymptotic")
#>
#>  Portmanteau Test (asymptotic)
#>
#> data:  Residuals of VAR object var
#> Chi-squared = 30, df = 30, p-value = 0.2

summary(var)
#>
#> VAR Estimation Results:
#> ========================
#> Endogenous variables: Consumption, Income
#> Deterministic variables: const
#> Sample size: 184
#> Log Likelihood: -380.155
#> Roots of the characteristic polynomial:
#> 0.782 0.498 0.498 0.444 0.286 0.286
#> Call:
#> VAR(y = uschange[, 1:2], p = 3, type = "const")
#>
#>
```

```
#> Estimation results for equation Consumption:
#> ============================================
#> Consumption = Consumption.l1 + Income.l1 + Consumption.l2 + Income.l2 + Consumption.l
#>
#>                Estimate Std. Error t value Pr(>|t|)
#> Consumption.l1   0.1910     0.0796    2.40  0.01752 *
#> Income.l1        0.0784     0.0545    1.44  0.15245
#> Consumption.l2   0.1595     0.0825    1.93  0.05479 .
#> Income.l2       -0.0271     0.0572   -0.47  0.63683
#> Consumption.l3   0.2265     0.0797    2.84  0.00502 **
#> Income.l3       -0.0145     0.0543   -0.27  0.78906
#> const            0.2908     0.0830    3.50  0.00058 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> Residual standard error: 0.595 on 177 degrees of freedom
#> Multiple R-Squared: 0.213,   Adjusted R-squared: 0.187
#> F-statistic: 7.99 on 6 and 177 DF,  p-value: 1.22e-07
#>
#>
#> Estimation results for equation Income:
#> =======================================
#> Income = Consumption.l1 + Income.l1 + Consumption.l2 + Income.l2 + Consumption.l3 + I
#>
#>                Estimate Std. Error t value Pr(>|t|)
#> Consumption.l1   0.4535     0.1156    3.92  0.00013 ***
#> Income.l1       -0.2730     0.0791   -3.45  0.00070 ***
#> Consumption.l2   0.0217     0.1198    0.18  0.85666
#> Income.l2       -0.0900     0.0831   -1.08  0.27979
#> Consumption.l3   0.3538     0.1157    3.06  0.00257 **
#> Income.l3       -0.0538     0.0787   -0.68  0.49570
#> const            0.3875     0.1205    3.22  0.00154 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> Residual standard error: 0.864 on 177 degrees of freedom
#> Multiple R-Squared: 0.175,   Adjusted R-squared: 0.148
#> F-statistic: 6.28 on 6 and 177 DF,  p-value: 5.31e-06
#>
#>
#>
#> Covariance matrix of residuals:
#>             Consumption Income
#> Consumption       0.355  0.185
#> Income            0.185  0.747
#>
#> Correlation matrix of residuals:
#>             Consumption Income
#> Consumption       1.000  0.359
#> Income            0.359  1.000
```

The R output on the following page shows the lag length selected by each of the information criteria available in the **vars** package. There is a large discrepancy between a VAR(5) selected by the AIC and a VAR(1) selected by the BIC. This is not unusual. As a result we first fit a VAR(1), selected by the BIC. In similar fashion to the univariate ARIMA methodology we test that the residuals are uncorrelated using a Portmanteau test[26] The null hypothesis of no serial correlation in the residuals is rejected for both a VAR(1) and a VAR(2) and therefore we fit a VAR(3) as now the null is not rejected. The forecasts generated by the VAR(3) are plotted in Figure **??**.

```
fcst <- forecast(var)
autoplot(fcst, xlab="Year")
#> Warning: Ignoring unknown parameters: xlab

#> Warning: Ignoring unknown parameters: xlab
```

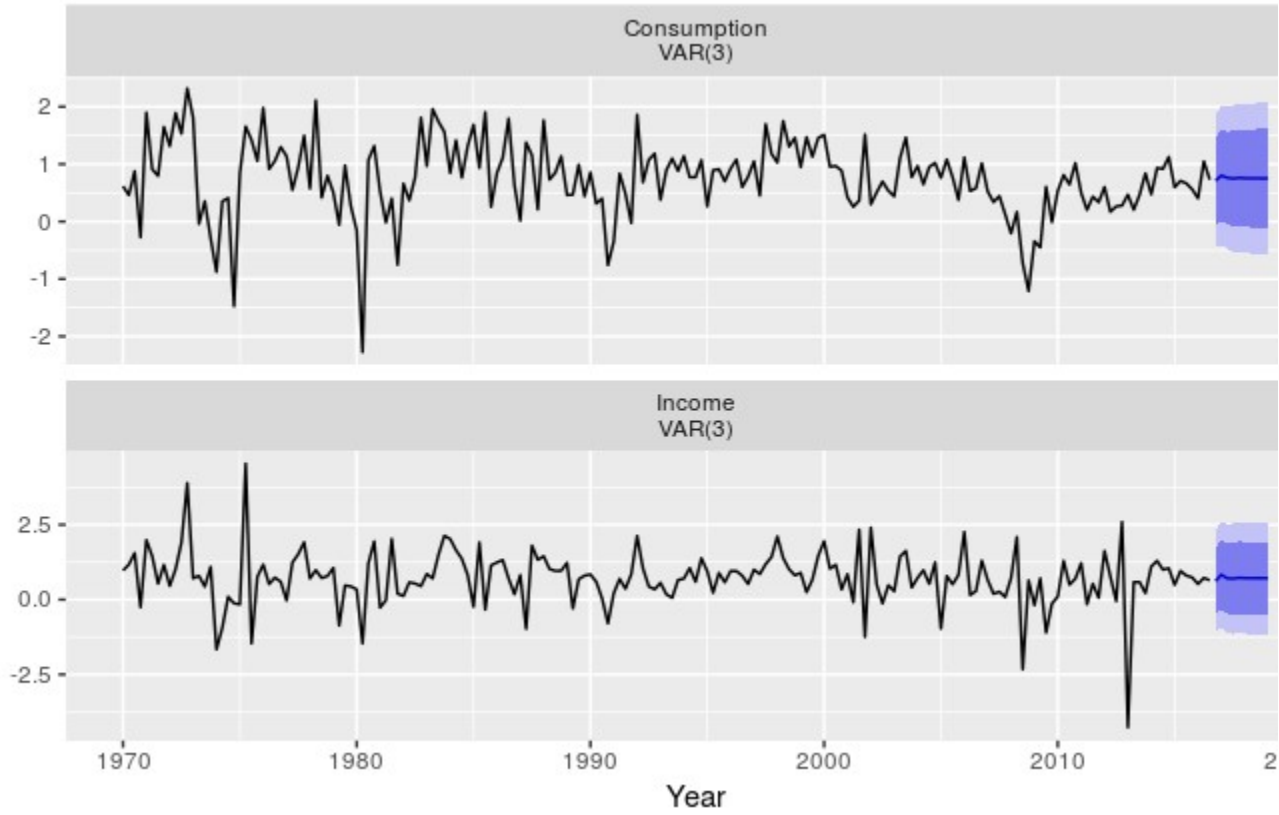


Figure 11.2: Forecasts for US consumption and income generated from a VAR(3).

11.4 Neural network models

Artificial neural networks are forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

Neural network architecture

A neural network can be thought of as a network of "neurons" organised in layers. The predictors (or inputs) form the bottom layer, and the forecasts (or outputs) form the top layer. There may be intermediate layers containing "hidden neurons".

The very simplest networks contain no hidden layers and are equivalent to linear regression. Figure ?? shows the neural network version of a linear regression with four predictors. The coefficients attached to these predictors are called "weights". The forecasts are obtained by a linear combination of the inputs. The weights are selected in the neural network framework using a "learning algorithm" that minimises a "cost function" such as MSE. Of course, in this simple example, we can use linear regression which is a much more efficient method for training the model.

```
[shorten \>=1pt,-\>,draw=black!50, node distance=] =[<-,shorten \<=1pt]
=[circle,fill=black!25,minimum size=17pt,inner sep=0pt] =[neuron,
```

```
fill=green!50]; =[neuron, fill=red!50]; =[neuron, fill=blue!50]; = [text
width=4em, text centered]

/ in <span>1,...,4</span> (I-) at (0,-) ;

\(O) ;

in <span>1,...,4</span> (I-) edge (O);

(input) <span>Input layer</span>; ;

\@ref(fig-10-nnet)
```

Once we add an intermediate layer with hidden neurons, the neural network becomes non-linear. A simple example is shown in Figure ??.

```
[shorten \>=1pt,-\>,draw=black!50, node distance=] =[<-,shorten \<=1pt]
=[circle,fill=black!25,minimum size=17pt,inner sep=0pt] =[neuron,
fill=green!50]; =[neuron, fill=red!50]; =[neuron, fill=blue!50]; = [text
width=4em, text centered]

/ in <span>1,...,4</span> (I-) at (0,-) ;

/ in <span>1,...,3</span> node[hidden neuron] (H-) at (,-cm) ;

\(O) ;

in <span>1,...,4</span> in <span>1,...,3</span> (I-) edge (H-);

in <span>1,...,3</span> (H-) edge (O);

(hl) <span>Hidden layer</span>; ; ;

\@ref(fig-10-nnet1)
```

This is known as a *multilayer feed-forward network* where each layer of nodes receives inputs from the previous layers. The outputs of nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output. For example, the inputs into hidden neuron j in Figure ?? are linearly combined to give zj=bj+4∑i=1wi,jxi. In the hidden layer, this is then modified using a nonlinear function such as a sigmoid, s(z)=11+e−z, to give the input for the next layer. This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

The parameters b1,b2,b3 and w1,1,…,w4,3 are "learned" from the data. The values of the weights are often restricted to prevent them becoming too large. The parameter that restricts the weights is known as the "decay parameter" and is often set to be equal to 0.1.

The weights take random values to begin with, which are then updated using the observed data. Consequently, there is an element of randomness in the predictions produced by a neural network. Therefore, the network is usually trained several times using different random starting points, and the results are averaged.

The number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance. We will consider how these can be chosen using cross-validation later in this chapter.

The `avNNet` function from the **caret** package fits a feed-forward neural network with one hidden layer. The network specified here contains three nodes (`size=3`) in the hidden layer. The decay parameter has been set to 0.1. The argument `repeats=25` indicates that 25 networks were trained

and their predictions are to be averaged. The argument `linout=TRUE` indicates that the output is obtained using a linear function. In this book, we will always specify `linout=TRUE`.

11.5 Exercises

1. Use the tbats function to model your retail time series.
    1. Check the residuals and produce forecasts.
    2. Does this completely automated approach work for these data?
    3. Have you saved any degrees of freedom by using Fourier terms rather than seasonal differencing?

2. Consider the weekly data on US finished motor gasoline products supplied (thousands of barrels per day) (series `gasoline`):

    1. Fit a tbats model to these data.
    2. Check the residuals and produce forecasts.
    3. Could you model these data using any of the other methods we have considered in this book?

3. Experiment with using `nnetar()` on your retail data and other data we have considered in previous chapters.

Using R

This book uses R and is designed to be used with R. R is free, available on almost every operating system, and there are thousands of add-on packages to do almost anything you could ever want to do. We recommend you use R with RStudio.

Installing R and RStudio

1. Download and install R.
2. Download and install RStudio.
3. Run RStudio. On the "Packages" tab, click on "Install packages" and install the package `fpp2` (make sure "install dependencies" is checked).

That's it! You should now be ready to go.

R examples in this book

We provide R code for most examples in shaded boxes like this:

```
autoplot(a10)
h02 %>% ets %>% forecast %>% summary
```

These examples assume that you have the `fpp2` package loaded. So you should use the command `library(fpp2)` before you try any examples provided here. (This needs to be done at the start of every R session.) Sometimes we also assume that the R code that appears earlier in the same section of the book has also been run; so it is best to work through the R code in the order provided within each section.

Getting started with R

If you have never previously used R, please first do the free online course "Introduction to R" from DataCamp.

While this course does not cover time series or forecasting, it will get you used to the basics of the R language. Other DataCamp courses for R may also be useful. The Coursera R Programming course is also highly recommended.

You will learn how to use R for forecasting using the exercises in this book.

Armstrong, J S, ed. 2001. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer Academic Publishers.

Armstrong, J Scott. 1985. *Long-Range Forecasting: From Crystal Ball to Computer*. Wiley.

Athanasopoulos, George, Roman A Ahmed, and Rob J Hyndman. 2009. "Hierarchical Forecasts for Australian Domestic Tourism." *International Journal of Forecasting* 25 (January): 146–66. doi:10.1016/j.ijforecast.2008.07.004.

Athanasopoulos, George, and Rob J Hyndman. 2008. "Modelling and Forecasting Australian Domestic Tourism." *Tourism Management* 29 (1): 19–31.

Bergmeir, Christoph, Rob J Hyndman, and Bonsoo Koo. 2015. "A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction." Working paper 10/15. Monash University Econometrics & Business Statistics. http://robjhyndman.com/working-papers/cv-time-series/.

Box, George E P, and Gwilym M Jenkins. 1970. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Box, George E P, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis: Forecasting and Control*. 5th ed. Hoboken, New Jersey: John Wiley & Sons.

Brockwell, Peter J, and Richard A Davis. 2016. *Introduction to Time Series and Forecasting*. 3rd ed. New York: Springer.

Brown, Robert Goodell. 1959. *Statistical Forecasting for Inventory Control*. McGraw/Hill.

Buehler, Roger, Deanna Messervey, and Dale Griffin. 2005. "Collaborative Planning and Prediction: Does Group Discussion Affect Optimistic Biases in Time Estimation?" *Organizational Behavior and Human Decision Processes* 97 (1): 47–63.

Cleveland, Robert B, William S Cleveland, Jean E McRae, and Irma J Terpenning. 1990. "STL: A Seasonal-Trend Decomposition Procedure Based on Loess." *Journal of Official Statistics* 6 (1): 3 –73.

Cleveland, William S. 1993. *Visualizing Data*. Hobart Press.

Cryer, Jonathan D, and Kung-Sik Chan. 2008. *Time Series Analysis: With Applications in R*. Springer Texts in Statistics. New York: Springer.

Dagum, Estela Bee, and Silvia Bianconcini. 2016. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation:* Springer.

De Livera, Alysha M, Rob J Hyndman, and Ralph D Snyder. 2011. “Forecasting Time Series with Complex Seasonal Patterns Using Exponential Smoothing.” *Journal of the American Statistical Association* 106 (496): 1513–27.

Eroglu, Cuneyt, and Keely L. Croxton. 2010. “Biases in Judgmental Adjustments of Statistical Forecasts: The Role of Individual Differences.” *International Journal of Forecasting* 26 (1): 116 –33.

Fildes, Robert, and Paul Goodwin. 2007a. “Against Your Better Judgment? How Organizations Can Improve Their Use of Management Judgment in Forecasting.” *Interfaces* 37 (6): 570–76.

———. 2007b. “Good and Bad Judgment in Forecasting: Lessons from Four Companies.” *Foresight: The International Journal of Applied Forecasting*, no. 8: 5–10.

Franses, Philip Hans, and Rianne Legerstee. 2013. “Do Statistical Forecasting Models for Sku-Level Data Benefit from Including Past Expert Knowledge?” *International Journal of Forecasting* 29 (1): 80–87.

Gardner Jr, Everette S. 1985. “Exponential Smoothing: The State of the Art” 4 (1): 1–28.

———. 2006. “Exponential Smoothing: The State of the Art — Part II.” *Interantional Journal of Forecasting* 22: 637–66.

Gardner Jr, Everette S, and Ed McKenzie. 1985. “Forecasting Trends in Time Series.” *Management Science* 31 (10): 1237–46.

Goodwin, Paul. 2000. “Correct or Combine? Mechanically Integrating Judgmental Forecasts with Statistical Methods.” *International Journal of Forecasting* 16 (2): 261–75.

Goodwin, Paul, and George Wright. 2009. *Decision Analysis for Management Judgment*. 4th ed. Chichester: Wiley.

Green, Kesten C., and J Scott Armstrong. 2007. “Structured Analogies for Forecasting.” *International Journal of Forecasting* 23 (3): 365–76.

Gross, C W, and J E Sohl. 1990. “Disaggregation methods to expedite product line forecasting.” *Journal of Forecasting* 9: 233–54.

Groves, Robert M., Floyd J. Fowler, Jr., Mick P. Couper, James M. Lepkowski, Eleanor Singer, and Roger Tourangeau. 2009. *Survey Methodology*. 2nd ed. Wiley.

Harris, Richard, and Robert Sollis. 2003. *Applied Time Series Modelling and Forecasting*. Chichester, UK: John Wiley & Sons.

Harvey, Nigel. 2001. “Improving Judgment in Forecasting.” In *Principles of Forecasting: A Handbook for Researchers and Practitioners*, edited by J Scott Armstrong, 59–80. Boston, MA: Kluwer Academic Publishers.

Holt, Charles E. 1957. “Forecasting Seasonals and Trends by Exponentially Weighted Averages.” O.N.R. Memorandum 52. Carnegie Institute of Technology, Pittsburgh USA.

Hyndman, Rob J, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. 2011. “Optimal Combination Forecasts for Hierarchical Time Series.” *Computational Statistics and Data Analysis* 55 (September). Citeseer: 2579–89. doi:10.1016/j.csda.2011.03.006.

Hyndman, Rob J, and Yeasmin Khandakar. 2008. “Automatic Time Series Forecasting: The Forecast Package for R.” *Journal of Statistical Software* 27 (1): 1–22. https://www.jstatsoft.org/article/view/v027i03.

Hyndman, Rob J, and Anne B Koehler. 2006. “Another Look at Measures of Forecast Accuracy.” *International Journal of Forecasting* 22: 679–88.

Hyndman, Rob J, A B Koehler, J K Ord, and R D Snyder. 2008a. *Forecasting with Exponential Smoothing*. Berlin, Germany: Springer-Verlag.

Hyndman, Rob J, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008b. *Forecasting with Exponential Smoothing: The State Space Approach*. Berlin: Springer-Verlag. www.exponentialsmoothing.net.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. New York: Springer.

Kahn, Kenneth B. 2006. *New Product Forecasting: An Applied Approach*. M.E. Sharp.

Kahneman, D., and D. Lovallo. 1993. “Timid Choices and Bold Forecasts: A Cognitive Perspective on Risk Taking.” *Management Science* 39 (1): 17–31.

Lawrence, Michael, Paul Goodwin, Marcus O’Connor, and Dilek Önkal. 2006. “Judgmental Forecasting: A Review of Progress over the Last 25 Years.” *International Journal of Forecasting* 22 (3): 493–518.

Morwitz, Vicki G., Joel H. Steckel, and Alok Gupta. 2007. “When Do Purchase Intentions Predict Sales?” *International Journal of Forecasting* 23 (3): 347–64.

Ord, J Keith, and Robert Fildes. 2012. *Principles of Business Forecasting*. South-Western College Pub.

Önkal, Dilek, Kadire Zeynep Sayim, and Mustafa Sinan Gönül. 2012. “Scenarios as Channels of Forecast Advice.” *Technological Forecasting and Social Change* 80: 772–88.

Pankratz, Alan E. 1991. *Forecasting with Dynamic Regression Models*. New York: John Wiley & Sons.

Pegels, C C. 1969. “Exponential Smoothing: Some New Variations.” *Management Science* 12: 311–15.

Peña, Daniel, George C Tiao, and Ruey S Tsay, eds. 2001. *A Course in Time Series Analysis*. New York: John Wiley & Sons.

Randall, Donna M, and James A Wolff. 1994. “The Time Interval in the Intention-Behaviour Relationship: Meta-Analysis.” *British Journal of Social Psychology* 33: 405–18.

Rowe, Gene. 2007. “A Guide to Delphi.” *Foresight: The International Journal of Applied Forecasting*, no. 8: 11–16.

Rowe, Gene, and George Wright. 1999. “The Delphi Technique as a Forecasting Tool: Issues and Analysis.” *International Journal of Forecasting* 15: 353–75.

Sanders, Nada, Paul Goodwin, Dilek Önkal, Mustafa Sinan Gönül, Nigel Harvey, Anthony Lee, and Lucy Kjolso. 2005. “When and How Should Statistical Forecasts Be Judgmentally Adjusted?” *Foresight: The International Journal of Applied Forecasting* 1 (1): 5–23.

Taylor, J W. 2003. “Exponential Smoothing with a Damped Multiplicative Trend.” *International Journal of Forecasting* 19: 715–25.

Theodosiou, Marina. 2011. “Forecasting Monthly and Quarterly Time Series Using STL Decomposition.” *International Journal of Forecasting* 27 (4): 1178–95.

Unwin, Antony. 2015. *Graphical Data Analysis with R*. Chapman; Hall/CRC.

Wickramasuriya, Shanika L, George Athanasopoulos, and Rob J Hyndman. 2015. “Forecasting Hierarchical and Grouped Time Series Through Trace Minimization.” Working paper 15/15. Monash University Econometrics & Business Statistics.

Winters, P R. 1960. “Forecasting Sales by Exponentially Weighted Moving Averages.” *Management Science* 6: 324–42.