

MATH6011: Forecasting

“All models are wrong, but some models are useful.” – George E. P. Box (1919–2013)



About the course

The overall aims of this course are to (1) introduce students to time series models and associated forecasting methods; (2) show how such models and methods can be implemented using Python-based libraries; (3) give an appreciation of the different fields of application of time series analysis and forecasting; and (4) convey the value of such quantitatively based methods for solving realistic practical problems. Students who complete the module successfully should be able to (a) formulate time series models and construct Python-based versions; (b) use Python functions built in various libraries to fit and analyse such models to data; (c) appreciate both the capabilities and the limitations of such computer based techniques; and (d) produce well-structured assignment reports describing problem formulation and solution.

It should be emphasized that the course is a basic introduction to forecasting, and there is no pre-requisite for taking it. But students who have taken (or are taking) any statistics or data analysis related module might be familiar with some important concepts that will be discussed in the course. As to the use of Python, those taking MATH6005 Introduction to Python might have an advantage in working through the workshop sheets. However, note that unlike in MATH6005, where you will be learning how to program in Python, in this forecasting course, we will just be using existing Python-based libraries to illustrate the concepts that will be studied. So, for this course and the related assessment, no programming skill is required and all the basic tools needed to succeed are provided. You are however free, in case you are an advanced user of Python, to develop your own codes for the forecasting methods that will be introduced in the course.

The module uses *Makridakis, S., Wheelwright, S.C. and Hyndman, R.J. 1998, Forecasting: Methods and Applications 3rd Ed., New York: Wiley* as text book. Most of the material for the lecture notes is extracted from there. Also, most of the data sets used in the demonstrations are drawn from this book. (The full set from the book can also be downloaded under Course Content, if desired.) *Hyndman, R.J. and Athanasopoulos, G. 2014, Forecasting: principles and practice, Otexts.com*, although slightly lighter on some aspects, has recently superseded the book by Makridakis et al. Hence, some of the material of these notes has also been drawn from there. An additional advantage of the book by Hyndman and Athanasopoulos (2014) is that it is freely accessible to [read online](https://otexts.com/fpp2/) (at <https://otexts.com/fpp2/>). A few hard copies of the book have been

ordered and made available in the Hartley Library.

As for the Python packages that will be used in this course, the following preprint provides a detailed relevant discussion: *Alain Zemkoho, A basic time series forecasting course with python*, <https://www.southampton.ac.uk/abz1e14/papers/ForecastingPaper.pdf>. Also, the e-book *Vothihong, P., Czygan, M., Idris, I., Persson, M.V., and Martins, L.F. 2017, Python: End to End Data Analysis, Packt* also provides some details on how some of the relevant packages work. Various other resources on the packages used in this course can be found online. Other interesting references on forecasting include the following ones:

1. Anderson, R.A., Sweeney, D.J. and Williams, T.A. 1994, *An Introduction to Management Science*, 7th Ed. West Publishing Co.;
2. Draper, N.R. and Smith, H. 1981, *Applied Regression Analysis*, 2nd Ed. New York: Wiley;
3. Gilchrist, W.G. 1976, *Statistical Forecasting*, New York: Wiley;
4. Janert, P.K. 2011, *Data Analysis with Open Source Tools*, Sebastopol: O'Reilly;
5. Wetherill, G.B. 1981, *Intermediate Statistical Methods*, London: Chapman and Hall.

All notes, data, and Python codes used in the module will be made available on the course [Blackboard](#) site under Course Content, where they are grouped by chapter and workshop, respectively. The notes are meant to be worked through and each chapter is accompanied by a number of demos associated to the data and codes, illustrating the topic or method being discussed. They are an essential part of the text and must be carefully studied, possibly before the lectures. All the data used in the lectures is stored in Excel spreadsheets. Many of these spreadsheets also implement Excel-based versions of the methods that will be discussed in the course. This will help you develop a better understanding of the techniques from these graphical implementations.

Exercises included at the end of each chapter correspond to the worksheet for the workshop of the corresponding week. They will be worked through during the workshops (computer labs) that will take place in the different allocated in the timetable. The workshop exercises follow the same patterns as the demos, and use the same data sets in some cases, in order to give you the opportunity to get more familiar with the related material, as focus at lectures will be more on the mathematical background of the different concepts and methods.

Assessment: The assessment of the module is 100% by a single coursework assignment. You will be given the assignment and related instructions in the second week. Further details on the submission and other key dates will be provided during the course.

Feedback: A key opportunity to get feedback on your progress in the module will be during the weekly workshops. To benefit the most from the workshops, you are strongly encouraged to work on the problem sheets in advance before coming to the workshop. This will help us assess where you are struggling and provide immediate help. It is also the best way for you to get well prepared for the exercises in your coursework assignment. You are also encouraged to come to the office hours to discuss any particular aspect of the lectures/material you might be struggling to understand. No appointment is needed to come to the office hour. We will be able to provide some brief element of feedback by email (efforts will be made to reply by the next working day after reception) if you have any quick questions. We have also arranged three voluntary sessions (assignment surgeries), prior to the coursework submission deadline, where you could ask questions and get feedback on the module and the coursework preparations; relevant details will be provide on Blackboard. The final feedback on your performance on the coursework will be provided within 4 weeks after the submission deadline.

Instructors

Dr Alain Zemkoho

(a.b.zemkoho@soton.ac.uk)

School of Mathematical Sciences

Dr Selin Ahipasaoglu

(s.d.ahipasaoglu@soton.ac.uk)

School of Mathematical Sciences

Demonstrators^a: TBC

^aThey will support you during the workshops and possibly during the assignment surgeries. Please do not contact them for any assistance related to the coursework assignment.



Contents

1	Introduction and basic tools	1
1.1	Graphical and numerical summaries	2
1.1.1	Graphical summaries	2
1.1.2	Numerical data summaries	3
1.2	Decomposition	4
1.2.1	Trend estimation using moving averages	5
1.2.2	Additive decomposition	6
1.2.3	Multiplicative decomposition	6
1.2.4	Decomposition and forecasting	7
1.3	Data preparation	7
1.3.1	Length of the times series	8
1.3.2	Missing and erroneous data	8
1.3.3	Transformations	8
1.3.4	Calendar adjustments	9
1.4	Python implementation guidelines	9
1.5	Exercises	10
2	Basic forecasting methods	13
2.1	Accuracy analysis	13
2.1.1	Measures of Accuracy	13
2.1.2	ACF of forecast error	15
2.1.3	Prediction interval	15
2.2	Averaging methods	15
2.3	Exponential smoothing methods	16
2.3.1	Single exponential smoothing	16
2.3.2	Holt's linear exponential smoothing	17

2.3.3	Holt-Winter's method	17
2.4	Python implementation guidelines	19
2.5	Exercises	19
3	Forecasting using regression	23
3.1	The model and key statistics	24
3.1.1	Model description	24
3.1.2	Computing the coefficients of the regression model	25
3.1.3	Key statistics for model evaluation	26
3.2	Selection of explanatory variables	28
3.2.1	Adding variables to the initial model	28
3.2.2	Time related explanatory variables	29
3.2.3	Subset selection	29
3.3	Multiple linear regression for forecasting	30
3.3.1	Assumptions made and the validity of tests and forecasts	30
3.3.2	Multicollinearity	31
3.4	Python implementation guidelines	31
3.5	Exercises	31
4	The ARIMA method	35
4.1	Preliminary analysis	35
4.1.1	The partial autocorrelation function	35
4.1.2	A white noise model	36
4.1.3	Stationarity	36
4.2	ARIMA models	39
4.2.1	Autoregression (AR) models	40
4.2.2	Moving Average (MA) models	40
4.2.3	ARIMA (p, d, q) models	41
4.2.4	ARIMA(p, d, q)(P, D, Q)s models	41
4.3	Model selection and forecasting	42
4.3.1	Phase 1: Identification	42
4.3.2	Phase 2: Parameters selection/estimation and testing	43
4.3.3	Phase 3: Forecasting using the ARIMA model	44
4.4	Python implementation guidelines	45
4.5	Exercises	45



1. Introduction and basic tools

Many companies make use of judgmental forecasting techniques which rely on the knowledge of experienced employees and managers. Such a *qualitative* approach is common in the case where there is no historical data; for example, if one wants to forecast the sales of a new product. A typical situation where judgemental forecasting can also be crucial is *Brexit*. As there is no precedent to this situation, it is almost impossible to accurately identify any historical factor that can be used to make reliable predictions of its impact on the UK's economy. Often, qualitative forecasting is carried out within the framework of fairly formal and regularly scheduled meetings. It can be augmented by the kind of quantitative techniques discussed in this unit, and this combined approach has much to commend it. Further details on qualitative forecasting methods can be found in Chapter 3 of the book by Hyndman and Athanasopoulos (2014).

Our focus in this course will be on *quantitative* forecasting methods. A quantitative approach relies on sufficient reliable quantitative information being available. An *explanatory model* is one that attempts to explain the relationship between the variable to be forecast and a number of independent variables, e.g., the Gross National Product (GNP) of a country can be obtained as

$$GNP = f(\text{monetary and tax policies, inflation, capital spending, imports, exports}) + \text{Error},$$

where f stands for “function of”.

A *black-box model* is one that simply tries to relate future values of the variable of interest to previous values, without attempting to explain its behaviour in terms of other variables. A *time series model* is one that attempts to relate the value of a variable(s) at *one time point* with values of the variable(s) at *previous time points*, for example,

$$GNP_{t+1} = f(GNP_t, GNP_{t-1}, GNP_{t-2}, \dots) + \text{Error}.$$

Here, t denotes the time. Thus “simple” time series models, like the one above, are “black-box”. More complex time series models are explanatory in that they try to relate the value of the variable of interest not simply with its previous values but also with previous values of other “explanatory” variables.

This module will be concerned with such simple time series models, as well as some explanatory modelling (see chapter on regression). The methods that we will discuss are all based on extrapolation into the future of patterns shown in the past. Confidence in such forecasts is therefore based on confidence that such patterns will, in future, remain stable.

We begin with the preliminaries to forecasting that enable you to begin to find the best forecasting model to use with a particular time series. Before we start with the basic tools that we will be using in the course, let us recall that our focus will mostly be on two types of data (time series data and cross-sectional data) that we are now going to formally define.

Definition 1.0.1 *Time series data* are data from a unit (or a group of units) observed in several successive periods, whereas *cross-sectional data* are data from units observed at the same time or in the same time period. The latter may be single observations from a sample survey or from all units in a population.

Though our main focus here will be on the first type, the second will also be of a great use, especially in Chapter 3. Examples of time series are discussed in Demo 1.1 while an example of cross-sectional dataset is given in Demo 1.3.

1.1 Graphical and numerical summaries

This course is a practical introduction to the skills of forecasting. To give experience in encountering a variety of time series, real-life datasets are used, both during the course and for the final coursework.

1.1.1 Graphical summaries

Time plots

The essence of forecasting models is that patterns are projected forward in time, while random effects are discarded. The first thing to do when forecasting is therefore to make a time plot and look for patterns. The following may be observed:

- A *trend*, which is a long term increase or decrease in the variable of interest.
- A *seasonal/periodic pattern* appears when a time series is affected by seasonal factors such as time of the year or the day of the week.
- A *cyclical pattern*, which is one where there are rises and falls but not of regular period, generally thought of as longer in time, e.g., several years.

It should be noted that combinations of the above three types of pattern occur frequently. We will treat trend and cycles together in the following analysis, and will often refer to trend-cycles.

A time series is said to be *stationary* if the distribution of the fluctuations is not time dependent. In particular both the *variability about the mean*, as well as the *mean* must be independent of time. A stationary time series therefore has no trend, cycle or seasonality and no patterns that can be used for forecasting.

Demo 1.1: See the folder named *Demo 1.1* under the Chapter 1 material folder. There, you have the following time series: Australian monthly electricity (*Electricity.xls*), US treasury bills (*TreasuryBills.xls*), Australian clay brick (*ClayBricks.xls*). Produce the time plots of these data sets by respectively running the Python codes [TimePlotElectricity.py](#), [TimePlotTreasuryBills.py](#), and [TimePlotClayBricks.py](#). You will see the following patterns in these series:

- The first example, Australian monthly electricity production, displays a clear trend and seasonality. Note that both the seasonal *variability* as well as the *mean* show a trend.
- The US treasury bill contracts data shows a trend, but there is less certainty as to whether this trend will continue.

- The data on Australian clay brick production contains occasional large fluctuations which are difficult to explain, and hence predict, without knowing the underlying causes.

Seasonal plots

If an initial inspection of a time plot leads you to suspect that seasonality may exist, then several methods exist to demonstrate and to give evidence for its existence. A seasonal plot is one where the time series is cut into regular periods and the time plots of each period are overlaid on top of one another. It is an effective means of demonstrating seasonality, for example to a client who is not overly technically minded.

Demo 1.2: As already indicated in the previous Demo, the Australian electricity production data exhibits seasonality. For further evidence of this, produce its seasonal plots by running the code [SeasonalPlotElectricity.py](#) present in the Demo 1.2 folder of Chapter 1. You can also produce the time and seasonal plots for the Australian beer production data, where seasonality is less clear, by running the codes [TimePlotBeer.py](#) and [TimePlotElectricity.py](#), respectively.

If seasonal plots lead you to think that seasonality may exist, then further technical evidence can be provided by autocovariance and autocorrelations. These concepts are introduced in the next section. Before going to that, we provide another class of plots that is useful in analyzing bivariate-type data sets (i.e., data sets based on two variables).

Scatterplots

The graphs discussed so far are useful for time series data. Scatter plots are most useful for exploring relationships between variables in *cross-sectional data*.

The automobile data of 19 Japanese cars (see “Data” sheet in `JapaneseCars.xls` in Demo 1.3 folder) are not a time series. Thus making time or seasonal plots inappropriate for these data. However, these data are well suited to a scatter plot (see Demo 1.3) such as that of price against mileage. In the figure (Demo 1.3), we plot the variable that we might want to forecast (price) against one possible explanatory variable (mileage). Each point on the graph represents one specific vehicle. The plot shows the relationship between price and mileage: vehicles with high mileage per gallon are generally cheaper than less fuel-efficient vehicles. (Both price and fuel-efficiency are related to the vehicle and engine size.) Vehicles with low mileage per gallon are generally priced over a range from around 12,000 to 25,000, with three vehicles much more expensive than other vehicles of comparable efficiency. A scatter plot helps us visualize the relationship and suggests that a forecasting model must include mileage as an explanatory variable.

When there are several potential explanatory variables, it is useful to plot each variable against each other variable to analyze correlations (see definition below) between them. These plots can be arranged in a *scatterplot matrix*; see, e.g., the 4th question of the Exercises of Chapter 3 that will be discussed at the corresponding workshop.

Demo 1.3: Run the code [ScatterPlotJapaneseCars.py](#) to produce the scatter plot mapping the price of the 19 Japanese cars against their mileage.

1.1.2 Numerical data summaries

Covariance and correlation

The most commonly used statistic for bivariate data (i.e. when there are two variables) is the *covariance*, and the *correlation coefficient*. If we have n pairs of observations (X_i, Y_i) on two variables X and Y , then the formulas are respectively

$$Cov_{XY} = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})$$

and

$$r_{XY} = \frac{Cov_{XY}}{S_X S_Y} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}. \quad (1.1)$$

Here, \bar{X} is the mean and $S_X = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2}$ is the (sample) standard deviation. The correlation coefficient r_{XY} , is a *standardised* version of the covariance and its value is always between -1 and 1. Values close to each limit indicate a strong *linear* relation between the two variables. Various Python-based packages have built-in functions to calculate covariance and correlation; the next demo uses `corrcoef` from the `numpy` package, as it can generate the correlation matrix of more than two data sets.

Demo 1.4: Use the code [CorrelationJapanese.py](#) to calculate the correlation between price and mileage in the 19 Japanese cars in the data set `JapaneseCars.xls`. The scatterplot mapping the cars' prices against the mileage already showed that there is a strong negative correlation between the two variables. This is confirmed by the value obtained from [CorrelationJapanese.py](#), which gives a value of approximately -0.72 (a number strictly below the middle value of -0.50 .)

Autocovariance and autocorrelation

The use of covariance and correlation can be extended to a time series $\{Y_t\}$. We can compare Y_t with the previous *lagged* value Y_{t-1} . The *autocovariance*, c_k , and *autocorrelation* at lag k , r_k , are defined by

$$c_k = \frac{1}{n-1} \sum_{t=k+1}^n (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})$$

and

$$r_k = \frac{\sum_{t=k+1}^n (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^n (Y_t - \bar{Y})^2}, \quad (1.2)$$

respectively. The complete set of autocovariances is called the *autocovariance function*, and the set of autocorrelations, the *autocorrelation function* (ACF).

Demo 1.5: Run the code [AutocorrelationBeer.py](#) (in Demo 1.5 folder) to plot the ACF function of the Australian beer production data (`Beer.xls`). The curve further confirms that there is some seasonality in this time series, as peaks and troughs are approximately occurring at regular intervals.

1.2 Decomposition

If the existence of trend/seasonality in a time series has been demonstrated, then it is possible to decompose a time series to give estimates of the underlying trend and seasonal parts. It should be noted that the techniques used are not forecasting methods in themselves, but they can be employed in actual forecasting in a limited number of situations, see Subsection 1.2.4 for related details.

The basic approach in analysing the underlying structure of a time series is to *decompose* it as

$$Y_t = f(S_t, T_t, E_t), \quad (1.3)$$

where Y_t is the observed value at time t and the variables are defined as follows:

- S_t is the seasonal component at time t ;
- T_t is the trend-cycle component at time t ;
- E_t is an irregular (random) component at time t .

There are several forms that the functional form f can take. But before we discuss the standard expressions, we first discuss the estimation of the trend component.

1.2.1 Trend estimation using moving averages

The first step in the decomposition is to produce an *estimate*, \hat{T}_t , of the trend-cycle. (Here and in what follows we use a circumflex to denote an estimate.) This is carried out by a *smoothing* technique known as *moving averages*. The basic idea is that values of observations which are close together in time will have trend-cycle components that are similar in value. Ignoring the seasonal component for the moment, the value of the trend-cycle component at some particular time point can then be obtained by taking an average of a set of observations about this time point. Because the values that are averaged depend on the time point, this is called a *moving average*.

There are many different forms that a moving average can take. Many have been constructed using ad-hoc arguments and reasoning. All boil down to being special cases of what is called a *k-point weighted moving average*:

$$M_t = \sum_{j=-m}^m a_j Y_{t+j}$$

where $m = (k-1)/2$ is called the *half-width*, and the a_j are called the *weights*.

Note that in this definition k must be an *odd* number. The simplest versions are the cases where all the weights are the same. This is then called a *simple moving average of order k*. For example, if $k = 3$, then

$$M_t = (Y_{t-1} + Y_t + Y_{t+1})/3.$$

If the weights are symmetrically balanced about the centre value (i.e. about $j = 0$ in the sum), then this is called a *centred moving average*.

Simple moving averages involving an even number of terms can be used, but are then not centred about an integer t . This can be redressed by averaging a second time only averaging the moving averages themselves. Thus, for example, if

$$M_{2.5} = (Y_1 + Y_2 + Y_3 + Y_4)/4 \text{ and } M_{3.5} = (Y_2 + Y_3 + Y_4 + Y_5)/4$$

are two consecutive 4-point moving averages, then we can centre them by taking their average

$$(M_{2.5} + M_{3.5})/2 = (Y_1 + 2Y_2 + 2Y_3 + 2Y_4 + Y_5)/8.$$

This example is called a 2×4 MA. It is simply a 5-point weighted moving average, with end weights each $1/8$, and with the other three weights being $1/4$.

If applied to quarterly data, this 2×4 MA, would give equal weight to all four quarters, as the 1st and last values would apply to the same quarter (but in different years). Thus this smoother would smooth out quarterly seasonal variation.

Similarly, a 2×12 MA would smooth out seasonal variation in monthly data. Question: What are the weights of a 2×12 MA smoother?

Demo 1.6: Fit the 7MA and 2×12 MA to the housing sales data by running the code [TrendEstimatesHouseSales.py](#). These series have been separately prepared in `HouseSales.xls`. However, simple Python codes can be written to do the calculations as well.

There is a problem applying a moving average at the two ends of a time series when we run out of observations to calculate the complete summation. When fewer than k observations are available the weights are usually *rescaled* so that they sum to unity.

An effect of a moving average is that it will underestimate trends at the ends of a time series. This means that the methods discussed so far are generally unsatisfactory for forecasting purposes when a trend is present.

Next, we use these trend estimation approach to introduce what might be called *classical decomposition* methods. These are techniques developed in the 1920's and form the basis of typical existing decomposition methods.

1.2.2 Additive decomposition

We first consider the additive case, where (1.3) takes the form

$$Y_t = S_t + T_t + E_t. \quad (1.4)$$

As most of our data sets and based on monthly observations, we assume throughout that the seasonal period is 12. The classical decomposition takes four steps:

Step 1: Compute the centred 2x12 MA. Denote this series by M_t (corresponding to \hat{T}_t in (1.4)). This series estimates the trend-cycle.

Step 2: De-trend the original series by subtraction:

$$D_t = Y_t - M_t = S_t + E_t.$$

Step 3: Calculate a *seasonal index* for each month by taking the average of all the values from each month j :

$$\hat{S}_j = \frac{1}{n_j} \sum_{k=1}^{n_j} D_{j+12(k-1)}.$$

In this formula, it is assumed that there are n_j values available for month j , so that the summation is over these n_j values.

Step 4: The estimated irregularity is obtained by subtraction of the seasonal component from the de-trended series:

$$\hat{E}_t = D_t - \hat{S}_{j(t)}.$$

Here $\hat{S}_{j(t)}$ denotes the seasonal index for the month corresponding to observation Y_t .

1.2.3 Multiplicative decomposition

For the multiplicative model

$$Y_t = S_t \times T_t \times E_t,$$

the method is called the *ratio of actual to moving averages*. There are again four steps:

Step 1: Compute the centred 2x12 MA. Denote this series by M_t (as in the previous case, this corresponds to \hat{T}_t). This step is exactly the same as in the additive model case.

Step 2: Calculate R_t , the ratio of actual to moving averages:

$$R_t = \frac{Y_t}{M_t}.$$

Step 3: Calculate a *seasonal index* for each month by taking the average of all the values each month, j :

$$\hat{S}_j = \frac{1}{n_j} \sum_{k=1}^{n_j} R_{j+12(k-1)}.$$

This step is exactly the same as in the additive case except that D is replaced by R .

Step 4: Calculate the error using

$$\hat{E}_t = \frac{R_t}{\hat{S}_t} = \frac{Y_t}{M_t \hat{S}_t}.$$

R The question arises as to which method of decomposition should be used for a particular dataset. The multiplicative model assumes that variability is amplified with trend, whereas constant variability is assumed by the additive model. This can be observed from time series plots. We will see in the next two demonstrations examples of use of the two models in these different conditions. An appropriate use of the model is confirmed by the randomness or otherwise of the remaining error terms (or residuals). For a general perception of the different patterns that can guide your choice, see Pegels classification in Figure 2.1 in the next chapter.

Demo 1.7: Use the code [DecomposeHouseSales.py](#) to produce the additive decomposition of the house sales. The trend-cycle, seasonal and irregular estimates are plotted in separate graphs. The data spreadsheet `HouseSales.xls` also shows how the decomposition can be produced in a step by step calculation.

Demo 1.8: Similarly to the previous demo, use [DecomposeAirlineSales.py](#) to produce the multiplicative decomposition components for an international airline's sales data.

1.2.4 Decomposition and forecasting

There have been many attempts to develop forecasts based directly on a decomposition. The individual components are projected into the future and recombined to form a forecast of the underlying series. Although this may appear a reasonable approach, in practice it rarely works well. The chief difficulty is in obtaining adequate forecasts of the components. The trend-cycle is the most difficult component to forecast. It is sometimes proposed that it be modeled by a simple function such as a straight line or some other parametric trend model. But such models are rarely adequate. The other components are somewhat easier to forecast. The seasonal component for future years can be based on the seasonal component from the last full period of data. But if the seasonal pattern is changing over time, this will be unlikely to be entirely adequate.

One approach that has been found to work reasonably well is to forecast the seasonally adjusted data using Holt's method (see next chapter), then adjust the forecasts using the seasonal component from the end of the data. Makridakis et al. (1982) found that forecasts obtained in this manner performed quite well compared with several other methods. However, in this course, decomposition will only be considered as a tool for understanding a time series rather than as a forecasting method in its own right. Time series decomposition provides graphical insight into the behavior of a time series. This can suggest possible causes of variation and help in identifying the structure of a series, thus leading to improved understanding of the problem and facilitating improved forecast accuracy. Decomposition is a useful tool in the forecaster's toolbox, to be applied as a preliminary step before selecting and applying a forecasting method.

1.3 Data preparation

We describe here the adjustments that may be needed to a dataset before it is ready for application of forecasting models, after preliminary analysis. *It should be emphasised that all such adjustments should be documented and justified as part of the process of analysis.*

1.3.1 Length of the times series

Consideration should be given to the length of the time series to be used for calculations of forecasts. Usually the entire available dataset is used, but sometimes changing conditions can produce radical changes in observed patterns. Sometimes forecasts cope well with such changes, but sometimes the methods do not cope well and it is better to truncate a dataset to more recent conditions. However, it is always necessary to have sufficient data to produce a forecast, and so the forecaster's judgment must be applied.

1.3.2 Missing and erroneous data

Real-life data is liable to contain human errors, most of which cannot be known with certainty by the forecaster. However, some clear outliers may be considered with high probability to be erroneous. For example, a missing or extra numeral will produce a resulting number that is ten times smaller or larger than neighbouring entries in a time series, and could seriously disrupt a forecast. If the source of the data can be referred to, it might then be possible to correct the error: if not, an estimate should be made.

If it is considered necessary to add an estimated value where missing or erroneous data is present, then this has to be carried out with due regard to the time series in question, and clear justification given. A local average value might be appropriate, but seasonality might also need to be considered.

1.3.3 Transformations

Sometimes a systematic adjustment of the data will lead to a simpler analysis: mathematical transforms may be applied in certain cases. There are two ideas that are helpful in selecting an appropriate transform.

First, it is usually easier to analyse a time series if the underlying mean varies in a linear way with time. Thus if the behaviour of the actual data has the form

$$Y_t = at^p + e_t,$$

where a and p are constants and e_t is a random “error”. Then, the transform

$$W_t = (Y_t)^{1/p} = (at^p + e_t)^{1/p} = bt + \delta_t,$$

where $b = a^{1/p}$, makes W_t look “more linear” than Y_t . Note that the transformed “error”, δ_t , will depend in a complicated way on e_t , a , p and t . However, in many situations the behaviour of δ_t will remain “random” looking and be no more difficult to interpret than the initial error e_t . The above is known as a *power transform*.

Another useful transform is the *logarithmic transform*:

$$W_t = \log_e(Y_t).$$

This can only be used if $Y_t > 0$, as the logarithm of a negative quantity is complex-valued.

The second idea is that the random errors are most easily handled if their variability is not time dependent but remains essentially constant. A good transformation should therefore be *variance stabilizing*, producing errors that have a constant variance. For example if

$$Y_t = a(t + e_t)^p,$$

where the e_t has a constant variance, then, the power transform

$$W_t = (Y_t)^{1/p} = a^{1/p}(t + e_t) = bt + \delta_t,$$

where $b = a^{1/p}$ and $\delta_t = be_t$, will not only linearise the trend, but will also be variance stabilizing, as δ_t will have constant variance.

Finally note that, although we analyse the transformed data, we are really actually interested in the original sequence. So it is necessary to *back transform* results into the original units. Thus, for example, in the last case, we might analyse W_t and estimate b , by, say \hat{b} , but we would back transform to estimate a by

$$\hat{a} = \hat{b}^p.$$

An important but somewhat difficult technical issue is that such transforms can destroy desirable properties like unbiasedness. A well known case concerns a random sample X_1, X_2, \dots, X_n , of size n . Here, the *sample variance* given by the formula

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

is known to be an unbiased estimator for the variance. However, s , the obvious estimator for the standard deviation is *not* unbiased. When n is large, this bias is, however, small.

Demo 1.9: Run the codes [SqrtTransformElectricity.py](#) and [LogTransformElectricity.py](#) to plot the Australian monthly electricity data using the square root and the (natural) log transforms, respectively. Each code produces the time plot of the transformed data, as well a histogram to facilitate the observation of the change.

1.3.4 Calendar adjustments

If data is for calendar months, then account might have to be taken of the length of a month. The difference between the longest and shortest months is about $(31 - 28)/30 = 10\%$. The adjustment needed is

$$W_t = \frac{\text{\# of days in an average month}}{\text{\# of days in month } i} \times Y_t = \frac{365.25/12}{\text{\# of days in month } i} \times Y_t.$$

Demo 1.10: Make separate time series plots of Y_t and W_t for the data on the monthly milk production (`MilkProduction.xls`) per cow. Then, run [CalendarAdjustmentMilkProduction.py](#) to produce a joint plot with both series to observe the changes.

1.4 Python implementation guidelines

To start with python, please look at the relevant file in the Chapter 0 folder on Blackboard, in case you are new at using the software. Once python is installed in your computer, you would just need to download the Chapter 1 folder and extract all its content and save on your local desktop. Subsequently, open your *spyder* editor and from there open each corresponding file mentioned in the demos and run them. This will allow you to develop a better understanding as you go through the concepts in this chapter and corresponding demonstration examples

The data sets to be used in the course will mostly be stored in excel files; hence, the function `read_excel` from `pandas` will be regularly used to read the necessary data sets:

```
from pandas import read_excel
```

For time and seasonal plots in Demos 1.1–1.3, the `pyplot` function from `matplotlib` will be used:

```
from matplotlib import pyplot
```

For the remaining demos, the relevant functions will be called from `numpy` and `statsmodels`; e.g.,

```
from numpy import sqrt
```

and

```
from statsmodels.graphics.tsaplots import plot_acf
```

will be used for a square root transformation (Demo 1.9) and autocorrelation (Demo 1.5) calculations, respectively. For the specific functions necessary for the other demos, look at the corresponding codes in Demonstration 1 folder (see the material for chapter 1/week 1).

1.5 Exercises

Recall that all the exercises mostly follow on the steps of the corresponding demonstrations in the notes above. The data sets necessary for this worksheet are available in the “Workshop 1” subfolder of the Chapter 1 material folder under Course Content (Blackboard). Whenever necessary, you could use similar steps as in the demo data files to prepare your data or proceed with some basic calculations. However, if you are an advanced Python user, you are welcome to write your own Python codes to do the calculations for you.

Exercise 1.1: Make time plots of the building materials (`BuildingMaterials.xls`) and cement production (`CementProduction.xls`) data sets. Give a title to the graph, dates on the X-axis and make sure the Y-axis is appropriately labelled (the files include details of the units involved, i.e. thousands of tonnes.) You can use adjust the colour scheme of the graphs to improve readability. What trends/cycles or seasonality do you observe?

Exercise 1.2: Produce seasonal plots for `BuildingMaterials.xls` and `CementProduction.xls`. What seasonality do you observe?

Exercise 1.3: Calculate the correlation coefficient for the two data sets `BuildingMaterials.xls` and `CementProduction.xls` over a suitable time period. Can a strong linear relationship (positive or negative) be observed?

Exercise 1.4: Plot the autocorrelation function for the building material (`BuildingMaterials.xls`). Does the plot exhibit any particular pattern?

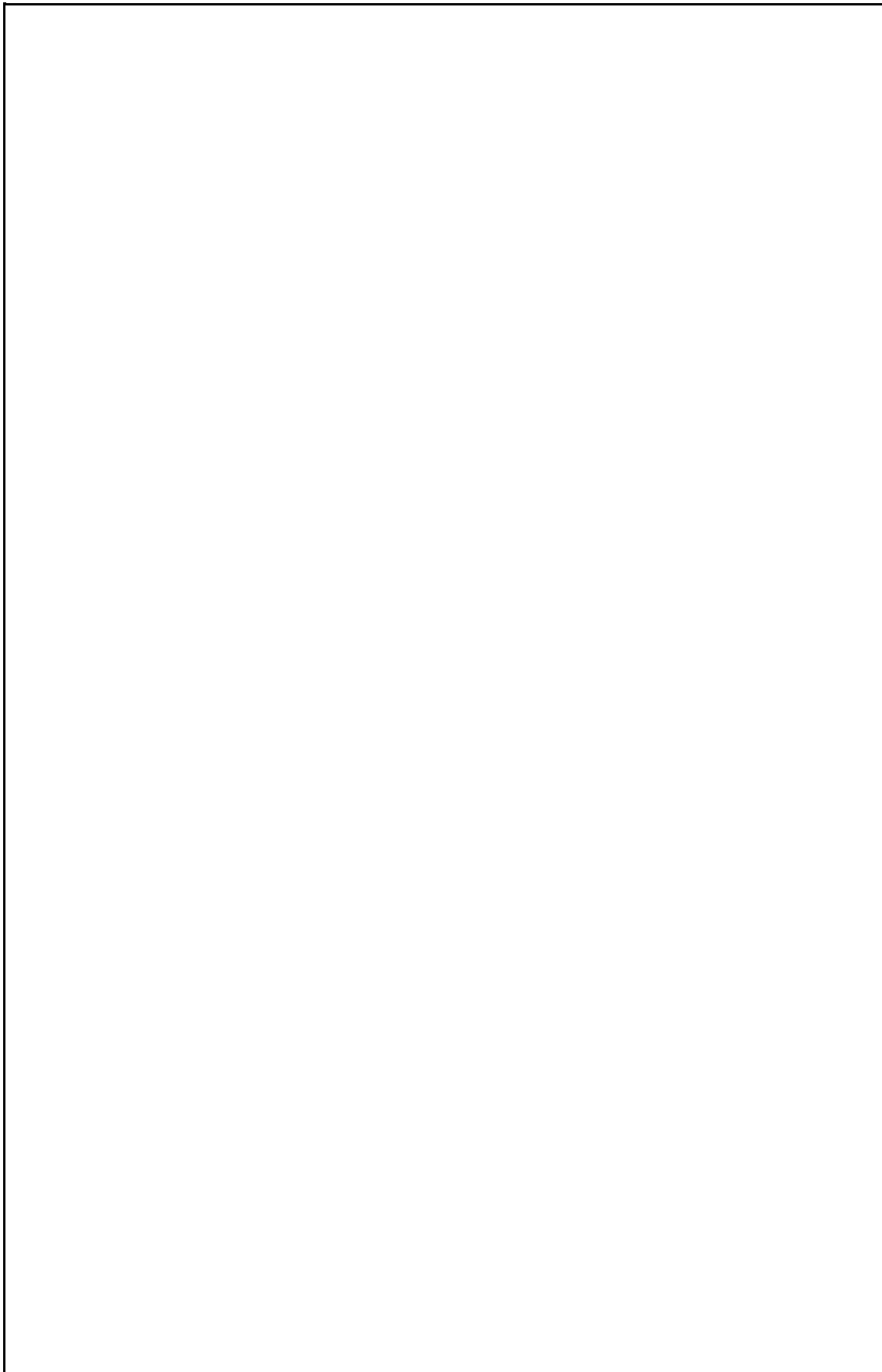
Exercise 1.5: Fit 7MA and 2x12MA moving averages to `BuildingMaterials.xls`. Which is the more appropriate for smoothing the data in this case?

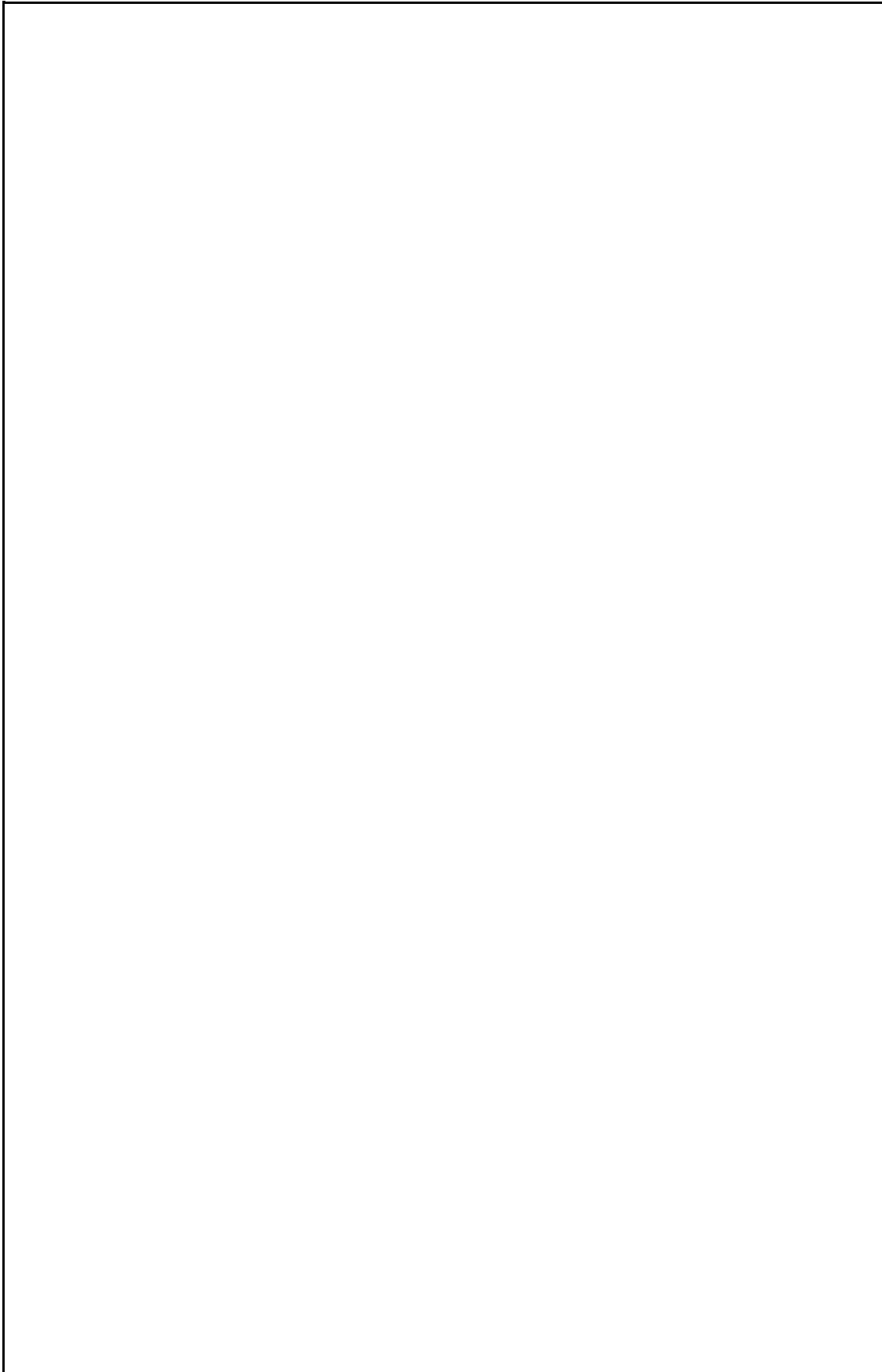
Exercise 1.6: Analyse `BuildingMaterials.xls` using the additive decomposition. Plot the trend-cycle, seasonal and irregular estimates.

Exercise 1.7: Analyse `CementProduction.xls` using the multiplicative decomposition. Plot the trend-cycle, seasonal and irregular estimates.

Exercise 1.8: Consider the brick deliveries data set, `BricksDeliveries.xls`:

1. The variance of the time series is decreasing. Can you find a transform that will stabilise it? Make time plots of the original and the transformed data.
2. Make calendar adjustments for this time series. Plot these against time plots of the original data. Did a calendar adjustment reduce the variance?







2. Basic forecasting methods

This chapter introduces basic forecasting methods based on *averaging* techniques. The basic forecasting methods to be considered are conventionally regarded as being divided in two groups: (i) *averaging methods* and (ii) *exponential smoothing methods*. Though it is convenient to follow this convention, it is important to realise at the outset that this distinction is artificial in that *all* the methods in this chapter are based on averages. They are thus all similar to the moving averages considered in the last chapter. The difference is that the averages are used here for *forecasting* rather than for *describing* past data. This point of potential confusion is made worse by the use of the name “exponential smoothing” for the second group. These methods are also based on weighted averages, where the weights decay in an exponential way from the most recent to the most distant data point. The term *smoothing* is being used simply to indicate that this weighted average smoothes the data irregularities. Thus, though the term smoothing here is used in the same sense as previously, the smoothing is being carried out in a different context from that used in the previous chapter. Before we start with the discussion of the methods, it is crucial to first present some tools that can be used to evaluate and compare the performance of forecasting techniques.

2.1 Accuracy analysis

2.1.1 Measures of Accuracy

The forecaster needs to choose the best model to use for forecasting any particular time series. We discuss here different measures for comparing different forecasting models on the basis of forecasting errors.

Let F_t be the forecast value and Y_t be the actual observation at time t . Then the forecast error at time t is defined as

$$e_t = Y_t - F_t. \quad (2.1)$$

Usually F_t is calculated from previous values of Y_t right up to and including the immediate preceding value Y_{t-1} . Thus F_t predicts just one step ahead. In this case F_t is called the *one-step* forecast and e_t is called the *one-step forecast error*. Usually we assess error not from one such e_t but from n values. Three measures of error are:

(i) the *mean error*

$$ME = \frac{1}{n} \sum_{t=1}^n e_t,$$

(ii) the *mean absolute error*

$$MAE = \frac{1}{n} \sum_{t=1}^n |e_t|,$$

(iii) and the *mean square error*

$$MSE = \frac{1}{n} \sum_{t=1}^n e_t^2.$$

The mean error is not very useful, as it tends to be near zero given that positive and negative errors tend to cancel each other. It is only of use in detecting systematic under or over forecasting; e.g., when pattern from original data is retained in the error time series.

The mean square error is a *squared* quantity; so one should be careful and not directly compare it with the MAE. Its square root is usually similar to the MAE.

The *relative* or *percentage error* can often also provide an interesting perspective of the error; it is defined as

$$PE_t = \left(\frac{Y_t - F_t}{Y_t} \right) \times 100,$$

while the *mean percentage error* is

$$MPE = \frac{1}{n} \sum_{t=1}^n PE_t$$

and the *mean absolute percentage error* is

$$MAPE = \frac{1}{n} \sum_{t=1}^n |PE_t|.$$

We illustrate these error measures in the following demonstration, which is based on two simple (or naïve) forecast methods.

Demo 2.1: For the Australian beer data introduced in Chapter 1, set up the naïve forecast method 1 and 2 denoted as *NF1* and *NF2*; see `BeerErrorsData.xls`. Then, calculate the ME, MAE, MSE, MPE, and MAPE based on *NF1* and *NF2*. Note that *NF1* is defined by

$$F_t = Y_t.$$

This simply takes the *present* Y value to be the forecast for the *next* period. The second naïve forecast, *NF2* which is defined next, takes into account some seasonal adjustment such that

$$F_{t+1} = Y_t - S_t + S_{(t-12)+1}.$$

Here the present Y value is adjusted by subtracting S_t , the current period's seasonality, and adding S_{t+1} , the next period's seasonality. The seasonality is obtained as

$$S_t = \frac{1}{m+1} (mS_{t-12} + Y_t) \quad \text{with } S_t = Y_t \quad \text{for } t = 1, \dots, 12,$$

where m is the number of complete years of data available. We also set $F_{t+1} = Y_t$ for $t = 1, \dots, 12$. See `BeerErrorsData.xls` for the data and values of *NF1* and *NF2* and run [ErrorMeasuresBeer.py](#) to generate a summary of the errors.

2.1.2 ACF of forecast error

It is often useful to regard the one-step forecast errors as a time series in its own right, and to calculate and plot the autocorrelation function (ACF) of this series. As mentioned above in the context of the errors, one would want the errors generated by a forecast method to be completely/purely random. If that is not the case, the ACF can retain some patterns observable in the original data set. Hence, having the ACF not completely random can be an indication that the forecasting method is not necessarily accurate. The ACF plot of the errors resulting from the implementation of NF1 and NF2 on the Australian beer production series has been done in the next demo:

Demo 2.2: The ACF for errors resulting from the naïve forecasts NF1 and NF2 for the Australian beer data is plotted; see [ACFErrorsBeer.py](#). Notice that there is pattern in the series (e.g., for errors resulting from NF1) and this has been picked up by the ACF with a high value at lag 12. With such a pattern evident in the errors, it can therefore be assumed that the forecasting model used here is not completely adequate. Do not read too much into the other autocorrelations as one should expect departures from zero even for the ACF of a random series. If time series is non-random then one or more of the autocorrelations will be significantly non-zero. In the ACF graph generated [ACFErrorsBeer.py](#), the horizontal lines displayed in the plot correspond to 95% and 99% confidence bands. The dashed line is 99% confidence band.

2.1.3 Prediction interval

Assuming that the errors are *normally distributed*, one can usefully assess the accuracy of a forecast by using \sqrt{MSE} as an estimate of the error. An approximate prediction interval for the next observation is

$$F_{t+1} \pm z\sqrt{MSE}, \quad (2.2)$$

where z is a quantile of the normal distribution. Typical values used are:

z	Probability
1.282	0.80
1.645	0.90
1.960	0.95
2.576	0.99

This enables, for example, 95% or 99% confidence intervals to be set up for any forecast.

2.2 Averaging methods

For this and following forecasting methods, we suppose that we are currently at time t and that we wish to use the data up to this time, i.e. $Y_1, Y_2, \dots, Y_{t-1}, Y_t$, to make forecasts $F_{t+1}, F_{t+2}, \dots, F_{t+m}$ of *future* values of Y . For averaging methods, it is only possible to make a one-step forecast, i.e. we can forecast only up to F_{t+1} . In the next section, however, we will discuss methods that can forecast further into the future, for time series with trend and/or seasonality.

The *moving average forecast of order k* , which we write as $MA(k)$, is defined as

$$F_{t+1} = \frac{1}{k} \sum_{i=t-k+1}^t Y_i.$$

This forecast is only useful if the data does not contain a trend-cycle or a seasonal component. In other words the data must be *stationary*, at least in the short term. Recall that a time series Y_t , which is a random variable, is said to be stationary if it has a probability distribution not depending on t .

A convenient way of implementing this forecast technique is to note that

$$F_{t+2} = \frac{1}{k} \sum_{i=t-k+2}^{t+1} Y_i = F_{t+1} + \frac{1}{k} (Y_{t+1} - Y_{t-k+1}).$$

This is known as an *updating formula* as it allows a forecast value to be obtained from the previous forecast value by a simpler calculation than using the defining expression.

The only point of note is that moving average forecasts give a progressively smoother forecast as the order increases, but a moving average of large order will be slow to respond to real but rapid changes. Thus, in choosing k , a balance has to be drawn between smoothness and ensuring that this *lag* is not unacceptably large.

2.3 Exponential smoothing methods

In this section, we introduce three exponential smoothing methods of forecasting that may be used to suit different conditions in a time series, i.e., depending on whether trend and seasonality are present or not. We start with the simplest form, *single exponential smoothing*.

2.3.1 Single exponential smoothing

The *single exponential forecast* or *simple exponential smoothing* (SES) is defined as

$$F_{t+1} = \alpha Y_t + (1 - \alpha) F_t,$$

where α is a given *weight* value to be selected subject to $0 \leq \alpha \leq 1$. Thus F_{t+1} is the weighted average of the *current* observation, Y_t , with the forecast, F_t , made at the *previous* time point $t - 1$.

Repeated application of the formula yields

$$F_{t+1} = (1 - \alpha)^t F_1 + \alpha \sum_{j=0}^{t-1} (1 - \alpha)^j Y_{t-j},$$

showing that the dependence of the current forecast on $Y_t, Y_{t-1}, Y_{t-2}, \dots$, falls away in an exponential way. The rate at which this dependence falls away is controlled by α . The larger the value of α , the quicker does the dependence on previous values fall away.

SES needs to be initialized. A simple choice is to use

$$F_1 = Y_1.$$

Other values are possible, but we shall not agonise over this too much as we are more concerned with the behaviour of the forecast once it has been in use for a while.

It should be noted that, as with averaging methods, SES can only produce a one-step forecast.

Demo 2.3: For the shampoo data (`ShampooSales.xls`), test the SES method as follows:

1. Select a few different values of α to see what influence the parameter has on the model.
2. Use an optimization software to optimize the selection of α .
3. Plot Y_t and the SES forecasting results and observe the differences.
4. Calculate the MSE of the SES from Y .

Scenarios of items 1–4 can be found in the Python code [SESShampooSales.py](#).

2.3.2 Holt's linear exponential smoothing

The Holt method, also known as the linear exponential method (LES) was introduced by Charles Holt in 1957. It is an extension of the simple/single exponential smoothing method to take into account a possible (local) linear trend. The trend makes it possible to forecast m time periods ahead. There are two smoothing constants α and β , $0 \leq \alpha, \beta \leq 1$. The equations are:

$$\begin{aligned} L_t &= \alpha Y_t + (1 - \alpha)(L_{t-1} + b_{t-1}), \\ b_t &= \beta (L_t - L_{t-1}) + (1 - \beta)b_{t-1}, \\ F_{t+m} &= L_t + b_t m. \end{aligned}$$

Here L_t and b_t are respectively (exponentially smoothed) estimates of the level and linear trend of the series at time t , whilst F_{t+m} is the linear forecast from t onwards.

Initial estimates are needed for L_1 and b_1 . Simple choices are

$$L_1 = Y_1 \text{ and } b_1 = 0.$$

If, however, zero is atypical of the initial slope, then a more careful estimate of the slope may be needed to ensure that the initial forecasts are not badly out.

It should be noted that to use this method, it is NOT necessary for a series to be completely linear, but some trend should be present.

Demo 2.4: For the shampoo data, test the Holt (LES) method as follows:

1. Select a few different combinations of α and β to see what influence they have in the model.
2. Use an optimization software to optimize the selection of α and β .
3. Plot Y_t and the LES forecasting results and observe the differences.
4. Calculate the MSE of the Holt method from Y .

Scenarios of items 1–4 can be found in the Python code [HoltShampooSales.py](#).

2.3.3 Holt-Winter's method

This is an extension of Holt's LES method to take into account seasonality denoted by S_t . There are two versions, *multiplicative* and *additive*, that we discuss below. Before, note that to choose between both approaches, the criteria discussed in Chapter 1 for the decomposition remains valid, i.e., the multiplicative method should be used when variability is amplified with trend, whereas the additive approach is more suitable in the case of constant variability. Whenever in doubt, both methods can be implemented and the best one chosen based on the error it generates.

Holt-Winter's method, multiplicative seasonality

The equations are

$$\begin{aligned} L_t &= \alpha \frac{Y_t}{S_{t-s}} + (1 - \alpha)(L_{t-1} + b_{t-1}), \\ b_t &= \beta (L_t - L_{t-1}) + (1 - \beta)b_{t-1} \\ S_t &= \gamma \frac{Y_t}{L_t} + (1 - \gamma)S_{t-s}, \\ F_{t+m} &= (L_t + b_t m) S_{t-s+m}, \end{aligned}$$

where s is the number of periods in one cycle of seasons, e.g., number of months or quarters in a year. To initialize we need one complete cycle of data, i.e., s values. Then set

$$L_s = \frac{1}{s}(Y_1 + Y_2 + \dots + Y_s)$$

and to initialize trend, we use $s + k$ time periods:

$$b_s = \frac{1}{k} \left(\frac{Y_{s+1} - Y_1}{s} + \frac{Y_{s+2} - Y_2}{s} + \dots + \frac{Y_{s+k} - Y_k}{s} \right).$$

If the series is long enough, then a good choice is to make $k = s$ so that two complete cycles are used. However, we can, at a pinch, use $k = 1$. Initial seasonal indices can be taken as

$$S_k = \frac{Y_k}{L_s}, \quad k = 1, \dots, s.$$

The parameters α , β , and γ should lie in the interval $[0, 1]$, and can be selected by minimising MAE, MSE or MAPE.

Holt-Winter's method, additive seasonality

The equations are

$$\begin{aligned} L_t &= \alpha(Y_t - S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1}), \\ b_t &= \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}, \\ S_t &= \gamma(Y_t - L_t) + (1 - \gamma)S_{t-s}, \\ F_{t+m} &= L_t + b_tm + S_{t-s+m}, \end{aligned}$$

where s is the number of periods in one cycle. The initial values of L_s and b_s can be as in the multiplicative case. The initial seasonal indices can be taken as

$$S_k = Y_k - L_s, \quad k = 1, 2, \dots, s.$$

Similarly, the parameters α , β , γ should lie in the interval $[0, 1]$, and can again be selected by minimising MAE, MSE or MAPE.

Demo 2.5: Apply the Holt-Winter method to the airline passengers data set contained in the spreadsheet `AirlineSales.xls`:

1. Proceed as in Demo 2.4, using different combinations of the parameters α , β and γ .
 2. Compare the forecast results for the Holt-Winter's method with those for LES.
- Run the code [HoltWinterAirlinesSales.py](#) for the results with certain scenarios.

A taxonomy of exponential smoothing methods

An important consideration in dealing with exponential smoothing methods having separate trend and seasonal aspects is whether or not the model should be additive (linear) or multiplicative (non-linear). Pegels (1969) has provided a simple but useful framework for analysing these scenarios as indicated by the patterns in Figure 2.1.

Clearly, the principle mentioned in the remark in Subsection 1.2.3 is reflected by the patterns in Figure 2.1; i.e., the multiplicative model assumes that variability is amplified with trend, whereas constant variability is assumed by the additive model. Pegels' classification in Figure 2.1 leads to the following general formulas that would lead to 9 different methods:

$$\begin{aligned} L_t &= \alpha P_t + (1 - \alpha)Q_t, \\ b_t &= \beta R_t + (1 - \beta)b_{t-1}, \\ S_t &= \gamma T_t + (1 - \gamma)S_{t-s}, \end{aligned}$$

where P , Q , R , and T vary according to which of the cells the method belongs to. For each corresponding scenario, the table in Figure 2.2 shows the appropriate values of P , Q , R , and T and the forecast formula for forecasting m periods ahead. Note that cell A-1 describes the SES method and cell B-1 describes Holt's method. The additive Holt-Winters' method is given by cell B-2 and the multiplicative Holt-Winters' method is given by cell B-3. To work through an example of one of the other cells, consider cell C-3, which refers to an exponential smoothing model that allows for multiplicative trend and multiplicative seasonality. Our focus in this course is on the standard cases of these methods discussed above, i.e., SES (A-1), LES (B-1), HW-Additive (B-2) and HW-Multiplicative (B-3). Further more recent extensions and improvements on these methods are discussed in Chapter 7 of the book by Hyndman and Athanasopoulos (2014).

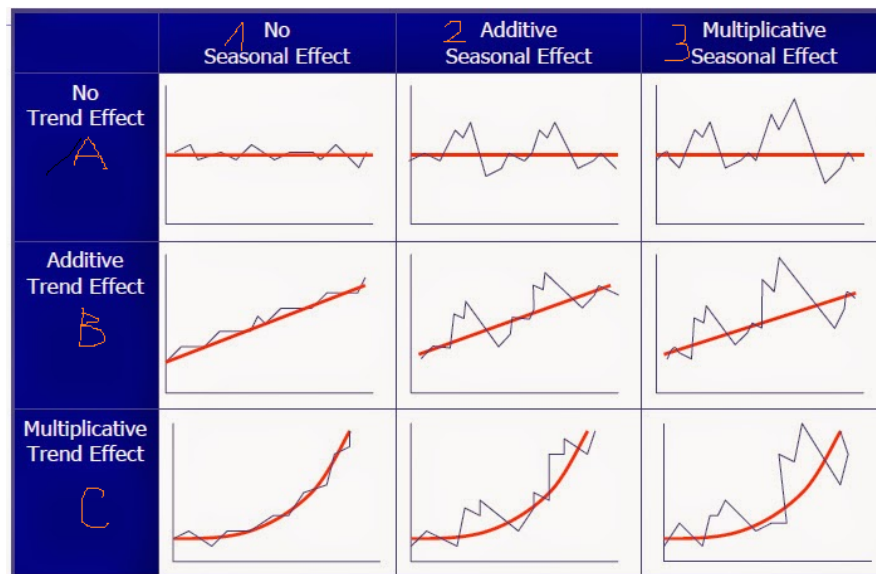


Figure 2.1: Patterns based on Pegels' (1969) classification

2.4 Python implementation guidelines

To implement the exponential smoothing methods discussed above, you will see in the demo codes referred to (see Demo 2.3, Demo 2.4, and Demo 2.5 folders), we use functions built in the

`statsmodels`

library. More precisely, you will see in the codes [SESShampooSales.py](#), [HoltShampooSales.py](#), and [HoltWinterAirlinesSales.py](#) that the command lines

```
from statsmodels.tsa.api import SimpleExpSmoothing
from statsmodels.tsa.api import Holt
from statsmodels.tsa.api import ExponentialSmoothing
```

are respectively used to call the *simple exponential method*, *Holt's linear exponential method*, and the *Holt-Winter method*, respectively. Each of the aforementioned demo codes produces plots made of two components: (a) *fitted values*, across the corresponding range of the original data set, in order to show how the produced results compare with the original time series for which forecasts are being developed; and (b) *forecast values* for a number of time periods ahead (e.g., 10, as in many of the demo codes; note that any other number of time points ahead can be chosen). For the later case, the accuracy will not be known until the actual observations are made in the future. However, in practice, you could split your data set in two parts, the training set (used to fit the forecast) and the testing set (usually 10% or more) that is then used to evaluate the accuracy of the model by comparing its values with those in the time points forecasted ahead. Also note that the values of the forecasts can be printed by applying the `print` command to `fcsti`, for example.

2.5 Exercises

See the Workshop 2 subfolder of the Material for Week 2 folder under Course Content.

Exercise 2.1: Set up forecasts NF1 and NF2 for the building material (`BuildingMaterials.xls`):

1. Calculate the ME, MAE, MSE, MPE, MAPE for these forecasts. How do you rate the two forecasts on the basis of these measures?

Trend	Seasonal Component		
	1 (none)	2 (additive)	3 (multiplicative)
A (none)	$P_t = Y_t$	$P_t = Y_t - S_{t-s}$	$P_t = Y_t/S_{t-s}$
	$Q_t = L_{t-1}$	$Q_t = L_{t-1}$	$Q_t = L_{t-1}$
		$T_t = Y_t - L_t$	$T_t = Y_t/L_t$
	$F_{t+m} = L_t$	$F_{t+m} = L_t + S_{t+m-s}$	$F_{t+m} = L_t S_{t+m-s}$
B (additive)	$P_t = Y_t$	$P_t = Y_t - S_{t-s}$	$P_t = Y_t/S_{t-s}$
	$Q_t = L_{t-1} + b_{t-1}$	$Q_t = L_{t-1} + b_{t-1}$	$Q_t = L_{t-1} + b_{t-1}$
	$R_t = L_t - L_{t-1}$	$R_t = L_t - L_{t-1}$	$R_t = L_t - L_{t-1}$
	$F_{t+m} = L_t + mb_t$	$F_{t+m} = L_t + mb_t + S_{t+m-s}$	$F_{t+m} = (L_t + mb_t)S_{t+m-s}$
C (multiplicative)	$P_t = Y_t$	$P_t = Y_t - S_{t-s}$	$P_t = Y_t/S_{t-s}$
	$Q_t = L_{t-1}b_{t-1}$	$Q_t = L_{t-1}b_{t-1}$	$Q_t = L_{t-1}b_{t-1}$
	$R_t = L_t/L_{t-1}$	$R_t = L_t/L_{t-1}$	$R_t = L_t/L_{t-1}$
	$F_{t+m} = L_t b_t^m$	$F_{t+m} = L_t b_t^m + S_{t+m-s}$	$F_{t+m} = L_t b_t^m S_{t+m-s}$

Figure 2.2: Formulas for calculations and forecasting using the Pegels' classification scheme

2. Plot the ACF of the one-step forecast errors of NF1 and NF2. Is there a pattern in the series? Are the forecasting models used here completely adequate?

Exercise 2.2: Proceeding as follows, employ the SES method to forecast the next month for the employment private services data set (`EmploymentPrivateServices.xls`):

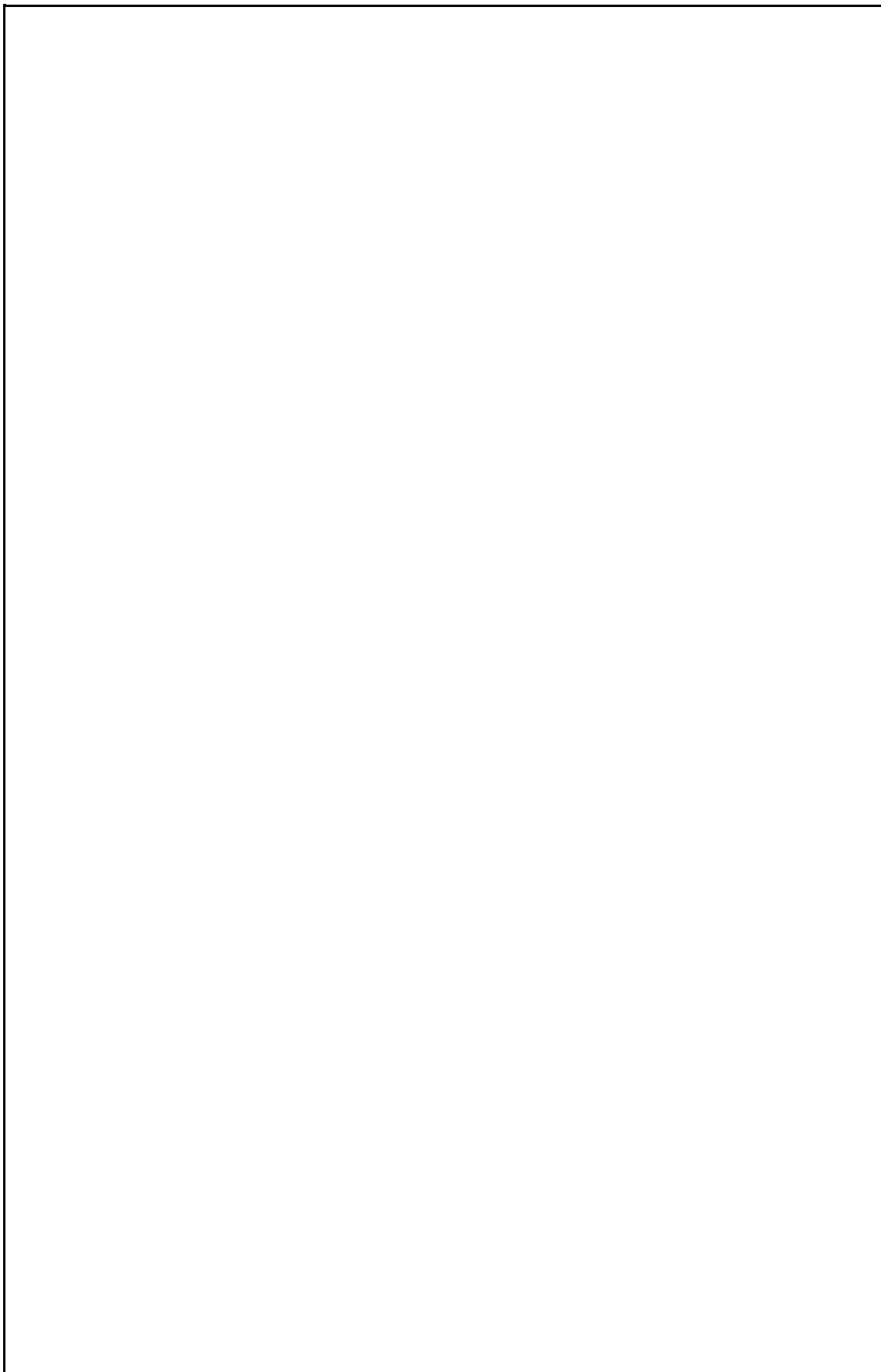
1. Set up the model with an initial value of 0.5 for α .
2. Plot the results from SES together with the original time series Y_t .
3. Also calculate the MSE of the SES method from Y .
4. Try different values of α . Then use the optimization solver built in `statsmodels` to automatically select the best value of α and corresponding model.
5. Plot the resulting forecast and the original time series, with dates on the X-axis. Give an appropriate label to the Y-axis.

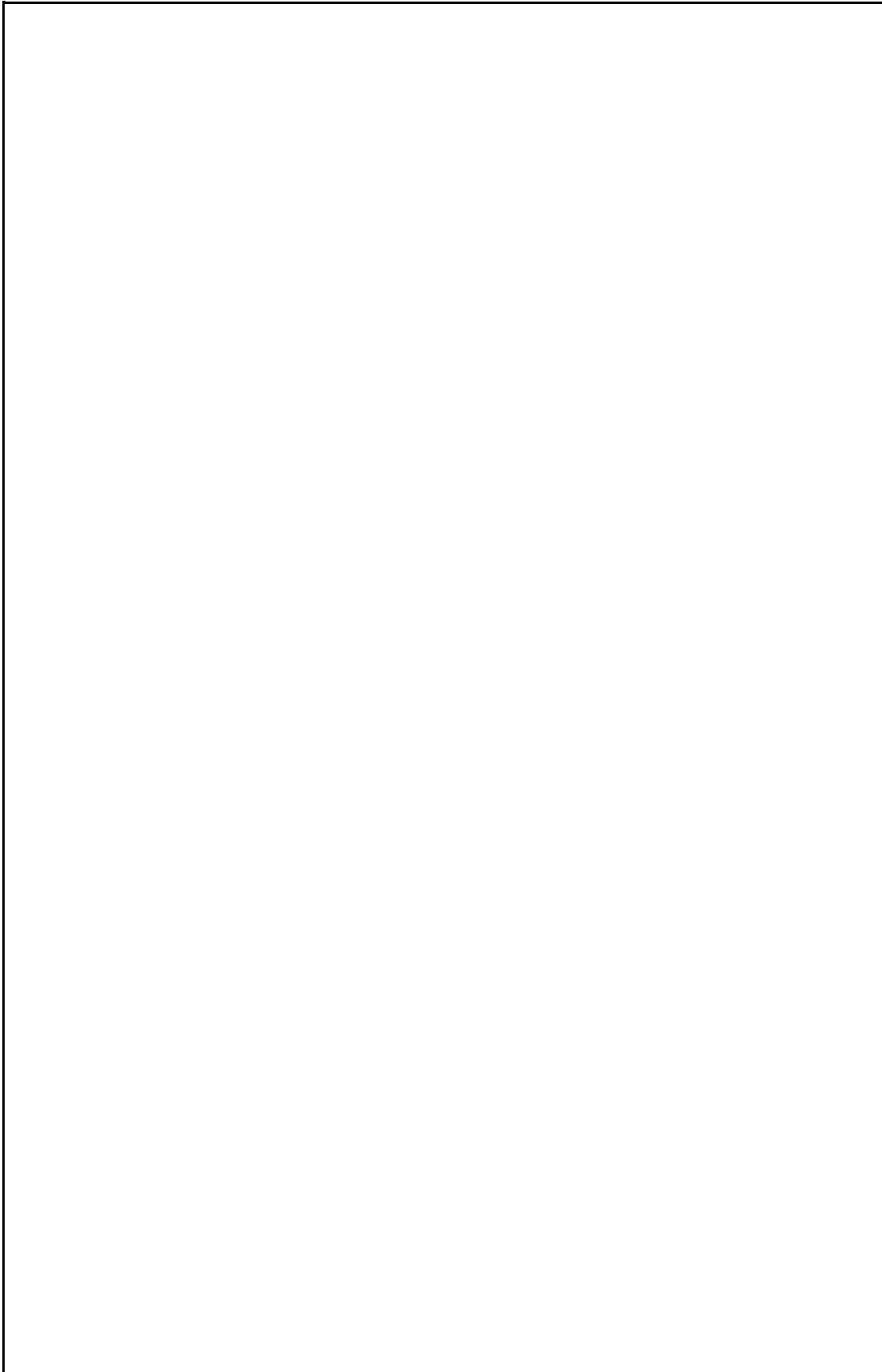
Exercise 2.3: Employ LES to forecast the next ten months for `employmentPrivateServices.xls`:

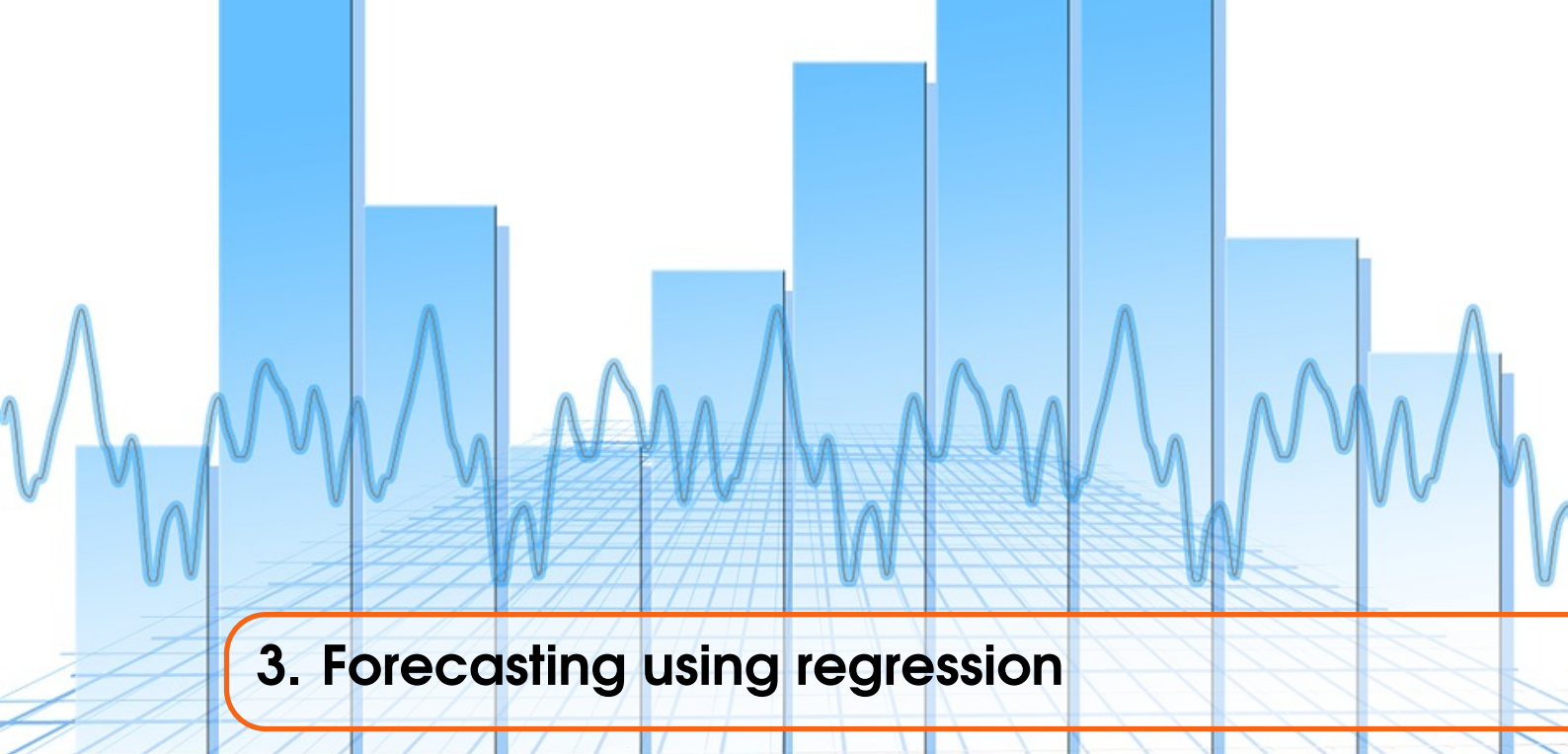
1. Set up the model with initial value of 0.5 for α and β .
2. Plot the results from LES together with the original time series Y_t .
3. Calculate also the MSE of the LES from Y .
4. Try different values of α and β . Then use the optimization solver built in `statsmodels` to automatically select the best value of α and β , as well as the corresponding model.
5. Plot the resulting forecast and the original time series, with dates on the X-axis. Give an appropriate label to the Y-axis.
6. What happens to the forecast if you vary α and β ? Especially note what happens for α and β taking the extreme values 0 and 1.

Exercise 2.4: Apply LES and Holt-Winter's forecasting method, both multiplicative and additive, to the cement production data (`CementProduction.xls`). This time, produce a full 12 months of forecasts with all these methods.

1. Follow procedure as in Demo 2.3, this time with 3 parameters α , β and γ , and plot results.
2. Compare results of the methods. What can you say about the residuals? What happens if you vary α , β and γ ? What if these parameters are 0 or 1?







3. Forecasting using regression

Forecasting models based on regression are examples of *explanatory models* constructed under the assumption that there is a possible relationship between the variable to be forecast and a number of independent variables.

The term *regression* refers to a certain type of statistical model that attempts to describe the relationship of one variable, called the *dependent* variable and usually denoted by Y , and a number of other variables X_1, X_2, \dots, X_k , called the *explanatory* or *independent* variables. We shall only consider the case of an additive error, e , where the relationship can be written as

$$Y = f(X_1, \dots, X_k; b_0, b_1, \dots, b_p) + e \quad (3.1)$$

with f being a given function, known as the *regression function*. The function will depend on *parameters* or *coefficients*, denoted by b_0, b_1, \dots, b_p . The parameters' values are not known and have to be estimated. The number of regression parameters $r = p+1$ is not necessarily the same as k . Finally there is an additional uncertain element in the relationship, represented by the random variable e . The probability distribution of e is usually specified, but this specification is not usually complete. For example, the distribution of e is often taken to be normal, $N(0, \sigma^2)$, but with the variance, σ^2 , unknown.

Irrespective of the form of e , we assume that its expected value is

$$E(e) = 0 \quad (3.2)$$

so that, assuming the explanatory variables are given, then the regression function is the expected value of Y :

$$E(Y) = f(X_1, \dots, X_k; b_0, b_1, \dots, b_p). \quad (3.3)$$

The simplest and most commonly used relationship is the *linear regression*:

$$Y = b_0 + b_1X_1 + \dots + b_kX_k + e. \quad (3.4)$$

This is called *simple linear regression* if there is only *one* explanatory variable, so that $k = 1$, i.e.,

$$Y = b_0 + b_1X_1 + e. \quad (3.5)$$

When $k > 1$, the relationship is called *multiple linear regression*.

The above regression models are completely general and not specifically connected with forecasting. As an example, consider the Japanese cars data discussed in Demos 1.3 and 1.4, where the price (Y) is plotted against the mileage (X), see scatter plot in Figure 3.1. Recall that this data set does not correspond to a time series; it is instead a cross sectional data set, see Definition 1.0.1. Using a regression line of the form (3.5), the aim is to find the line that fits best to the data. The selection of the best line is based on the minimization of these errors, cf. discussion in the next subsection.

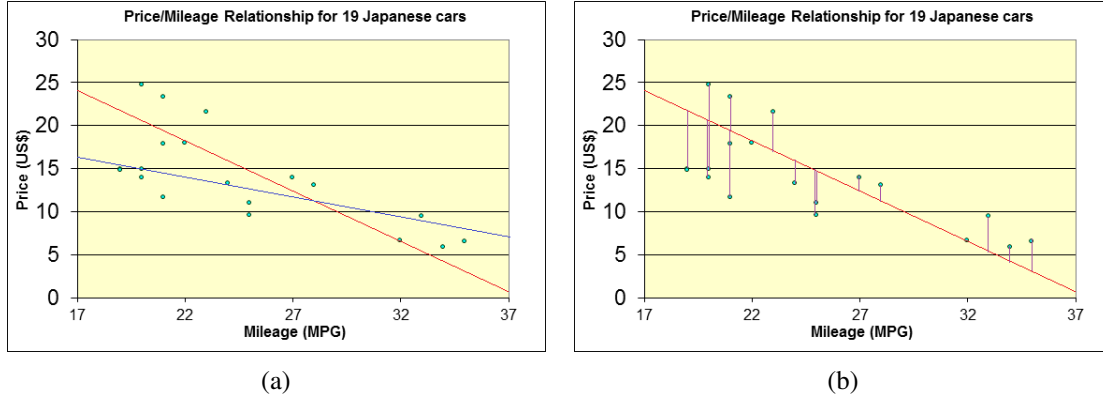


Figure 3.1: (a) shows two possible lines to establish a linear relationship between price (Y) and mileage (X), while in (b), the vertical segments represent the errors resulting from the choice of the red line.

Forecasting is certainly an area of application of regression techniques, and will be our main focus in this chapter. We will be examining the use of multiple linear regression (3.4) for forecasting, with $i = t$ (time). Before considering this application, we summarize the main features of linear regression as a classical statistical technique. There is a huge literature on linear regression. A good description at a reasonably accessible level is Wetherill (1981). For a derivation of the results, see Draper and Smith (1981), for example.

3.1 The model and key statistics

3.1.1 Model description

For convenience, we denote by a column vector the coefficients in (3.4):

$$b = (b_o, b_1, \dots, b_p)^\top$$

with the superscript T denoting the transpose. The values are usually *unknown* and have to be estimated from a set of n observed values $\{Y_i, X_{i1}, \dots, X_{ik}\}$, $i = 1, \dots, n$, where k is the number of explanatory variables. In linear regression, the relationship between the values of Y , and the X 's of each observation is assumed to be

$$Y_i = b_o + b_1 X_{i1} + \dots + b_k X_{ik} + e_i, \quad i = 1, \dots, n. \quad (3.6)$$

It is assumed that the explanatory variables X_{ij} are uncorrelated with the error terms e_i . It is also assumed that the errors e_i are uncorrelated with each other, and have a normal distribution with mean zero.

It is convenient to write this in the partial vector form

$$Y_i = X_i b + e_i, \quad i = 1, \dots, n, \quad (3.7)$$

where $X_i = (1, X_{i1}, X_{i2}, \dots, X_{ik})$ is a row vector. The full vector form is

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1k} \\ 1 & X_{21} & X_{22} & \dots & X_{2k} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{nk} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}. \quad (3.8)$$

More compactly, this means that we have

$$Y = Xb + e. \quad (3.9)$$

The values of the explanatory variables written in this matrix form is called the *design matrix*. In classical regression, this is usually taken to be non random, and not to depend explicitly on time.

Demo 3.1: Identify the observations Y and the design matrix X for the bank data example (Bank.xls). Note that this example is typical of situations where regression is used in forecasting, in that the observations $\{Y_i, X_{i1}, \dots, X_{ik}\}$, $i = 1, \dots, t$ come from a time series. We have emphasized this here by using t as the subscript indicating the current time point rather than n . [In this example, $k = 3$ and $n = 53$ as we will keep the last 6 observations for each variable for comparison with forecast results. You will note that the number of items left is approximately 10% of the total data points available.] You can run the code [TimePlotBank.py](#) for an initial view of on the nature of the data sets; [CorrelationBank.py](#) will allow you to see the correlations between the data sets.

3.1.2 Computing the coefficients of the regression model

To get the coefficients vector, $b = (b_0, \dots, b_k)$, of the regression model (3.4), the method commonly used for its estimation is called *least squares* (LS) method, where the aim is to find the minimum *sum of squares* of the error terms (see Figure 3.1b):

$$S^2 = \sum_{i=1}^n e_i^2 = (Y - Xb)^T (Y - Xb) \quad (3.10)$$

with respect to b . A more statistically satisfactory method is that of *maximum likelihood*. This latter technique requires an explicit form to be assumed for the distribution of e , such as the normal, whereas least squares is distribution free. In the special case where the errors are assumed to be normally distributed then least squares and maximum likelihood methods are essentially equivalent.

By a basic rule from calculus (i.e., the Fermat rule), b can be obtained from (3.10) by solving the system of equation $\nabla_b S^2 = 0$. This gives the value

$$\hat{b} = (X^T X)^{-1} X^T Y.$$

Following the expression in (3.9), the *estimate* of the regression function at the i th observed value $X = (1, X_{i1}, \dots, X_{ik})$ is written as \hat{Y}_i , and is calculated as

$$\hat{Y}_i = X_i \hat{b}. \quad (3.11)$$

Before we go into the details on how this formula can be used to concretely forecast (see Section 3.3), we first discuss how the performance of a regression model is evaluated (see Subsection 3.1.3) and how the explanatory variables can be selected (see Section 3.2).

3.1.3 Key statistics for model evaluation

When we define an equation relating Y (forecast variable) and X (explanatory variable), we are assuming that there is an underlying statistical model which we are estimating, cf. previous subsection. Certain statistical tests can be conducted to test the significance of the overall regression equation, to test the individual coefficients in the equation, and to develop prediction intervals for any forecasts that might be made using the regression model. In this subsection, we briefly introduce these statistics and how they can be used to assess the quality of a model.

The coefficient of determination

To proceed here, recall that Y represents the observed values, \hat{Y} , the estimate of Y obtained in (3.11) and \bar{Y} , the overall mean of the observations of Y , i.e., $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$. Based on these values, we respectively defined the following quantities:

$$\begin{aligned} SST &= \sum_{i=1}^n (Y_i - \bar{Y})^2, \\ SSR &= \sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2, \\ SSE &= \sum_{i=1}^n (Y_i - \hat{Y}_i)^2. \end{aligned}$$

SST represents the *total sum of squares (corrected for the overall mean)* while SSR is called the *regression sum of squares*. The latter quantity measures the reduction in total sum of squares due to fitting the terms involving the explanatory variables. As for SSE, it is called the (minimized) *residual or error sum of squares* and gives the variance of the total sum of squares not explained by the explanatory variables. In short, SSR and SSE stand for the *explained* and *unexplained* sum of squares, respectively. And we can easily show that the following relationship is satisfied:

$$SST = SSE + SSR. \quad (3.12)$$

The sample correlation, $r_{Y\hat{Y}}$, between the observations Y_i and the estimates \hat{Y}_i can be calculated from the usual sample correlation formula (1.1) and is called the *multiple correlation coefficient*. Its square, usually denoted as R^2 , turns out to be

$$R^2 = r_{Y\hat{Y}}^2 = \frac{SSR}{SST}. \quad (3.13)$$

This is called the *coefficient of determination*. R^2 is a measure of the proportion of the variance of the Y 's accounted for by the explanatory variables.

The F-test of overall significance

The sums of squares, SST, SSR and SSE, each have an associated number of *degrees of freedom* respectively denoted as dfT, dfR and dfE, defined by

$$\begin{aligned} \text{dfT} &= n - 1 && (\# \text{ of observations} - 1), \\ \text{dfR} &= k && (\# \text{ of coefficients} - 1), \\ \text{dfE} &= n - k - 1 && (\# \text{ of observations} - \# \text{ of coefficients}), \end{aligned} \quad (3.14)$$

where # stands for *number*. Similarly to the total sum of squares, dfT, dfR and dfE respectively represent the *total*, *explained* and *unexplained* degrees of freedom. And we obviously have

$$\text{dfT} = \text{dfR} + \text{dfE}.$$

Based on the values in (3.14), we can respectively define the *mean squares* as follows:

$$\begin{aligned} \text{total:} \quad \text{MST} &= \text{SST}/\text{dfT}, \\ \text{explained:} \quad \text{MSR} &= \text{SSR}/\text{dfR}, \\ \text{unexplained:} \quad \text{MSE} &= \text{SSE}/\text{dfE}. \end{aligned} \quad (3.15)$$

Under the assumption that the errors, e_i , are all independent and normally distributed, the distributional properties of all the quantities just discussed are well known.

It would be helpful to have a statistical test that would aid the forecaster in deciding on the significance of the relationship between the forecast variable Y and the explanatory variables, X_1, \dots, X_k . The F -test is such a test. Thus the F statistic is the ratio of two mean squares, i.e.,

$$F = \frac{\text{MSR (explained MS)}}{\text{MSE (unexplained MS)}}. \quad (3.16)$$

The numerator refers to the variance that is explained by the regression, and the denominator refers to the variance of what is not explained by the regression, namely the errors. In fact, the F statistic is intimately connected to the definition of the coefficient of determination, given that from equations (3.13)–(3.16), we have

$$F = \frac{R^2/k}{(1-R^2)/(n-k-1)}.$$

Most computer packages will also report the P -value along with F statistic. The P -value gives the probability of obtaining an F statistic as large as the one calculated for your data, if in fact the true slope is zero. So, if the P -value is small, then the regression is significant. It is customary to conclude that the regression is significant if the P -value is smaller than 0.05, although this threshold is arbitrary.

All the values derived in the current and previous subsection are conventionally set out in the following analysis of variance (ANOVA) table:

Source	Sum of Squares	df	MS	F	P
Regression	SSR	k	SSR/ k	MSR/MSE	P-value
Error	SSE	$n - k - 1$	SSE/ $(n - k - 1)$		
Total	SST	$n - 1$			

Demo 3.2: The code [ANOVABank.py](#) produces the regression ANOVA table for the bank data, with a clear indication that the model is significant. As in the previous demo, use only the first 53 observations; the remaining observations will be used for model validation later.

The coefficients of individual significance

Either the coefficient of determination, R^2 , or the F - *ratio* gives an overall measure of the significance of the explanatory variables. If overall significance is established then it is natural to try to identify which of the explanatory variables is having the most effect. *Individual coefficients can be tested in the presence of all the other explanatory variables relatively easily.*

Let us start by pointing out that in the practical process of estimating the coefficients of the regression model, b_j , $j = 0, 1, \dots, k$, in Subsection 3.1.2, their values pretty much depends on the values of observations considered. Hence, these values must both be considered as random variables. In other words, b will fluctuate from sample to sample.

Hence, assuming again that the errors e_i are $N(0, \sigma^2)$ variables, then the covariance matrix of \hat{b} is given by

$$\text{Var}(\hat{b}) = (X^\top X)^{-1} \sigma^2,$$

where σ^2 can be estimated as the MSE, i.e., $\hat{\sigma}^2 = MSE$. The best way of describing how much each estimate fluctuates is in the form of a confidence interval. We can obtain the confidence interval for each b_j as follows

$$\hat{b}_j \pm t^* \times SE(\hat{b}_j),$$

where $SE(\hat{b}_j)$, the standard error of \hat{b}_j , has a *t-distribution* with $n - k - 1$ degrees of freedom. As for t^* , it is a multiplying factor that depends on the number of observations used in the regression and the level of confidence required. Possible values for t^* are given in the table provided in Figure 3.2, see last page of this chapter.

A related idea is the *t-test* which is a test of whether a parameter is equal to zero. For $j = 0, 1, \dots, k$, the following *t-test* can be set-up for b_j :

$$t_j = \frac{\hat{b}_j}{SE(\hat{b}_j)}.$$

This statistic indicates if the values for b_j is significantly different from zero. Precisely, when b_j is significantly different from zero, the value of t_j will be large (in either the positive or negative direction depending on the sign of b_j).

Similarly to the *F-test*, the `statsmodels` function `ols` (ordinary least squares) generates *P-values* of each *t-statistic* (see next demo below). Each of these *P-values* is the probability, $Pr(T > |t_j|)$, of obtaining a value of $|t_j|$ as large as the one calculated for your data, if in fact the parameter is equal to zero. So, if a *P-value* is small, then the estimated parameter is significantly different from zero. As with *F-tests*, it is customary to conclude that an estimated parameter is significantly different from zero (i.e., *significant*, for short) if the *P-value* is smaller than 0.05, although this threshold is also arbitrary.

Demo 3.3: As in the previous demo, the code [IndividualSignificance.py](#) calculates \hat{b}_j and $SE(\hat{b}_j)$, as well as the *P-values*, $Pr(T > |t_j|)$, and the confidence intervals of the parameters.

3.2 Selection of explanatory variables

3.2.1 Adding variables to the initial model

In any regression model, including the ones used with time series, one may consider introducing additional explanatory variables to explain more of the variability of Y . This is especially desirable when the error sum of squares, *SSE* is large compared with *SSR* after fitting the initial set of explanatory variables.

One useful type of additional variable to consider are what are called *indicator* variables. This often arises when one wishes to include an explanatory variable that is *categorical* in form. A categorical variable is one that takes only a small set of distinct values. For example suppose that we have a categorical variable, W , taking just one of four values *Low*, *Medium*, *High*, *Very High*. If the effect of W on Y is predictable then it might be quite appropriate to assign the values 1, 2, 3, 4 to the categories Low, Medium, High, Very High and then account for the effect of W using just one coefficient a :

$$Y_i = b_o + b_1X_{i1} + \dots + b_kX_{ik} + aW + e_i, \quad i = 1, \dots, n.$$

However if the effect of each of the different possible values of the categorical variable on Y is not known then we can adopt the following different approach. If there are c categories then we introduce $(c - 1)$ indicator variables. In the example we therefore use $4 - 1 = 3$ indicator variables; i.e., W_1 , W_2 and W_3 . The observations are assumed to have the form

$$Y_i = b_o + b_1X_{i1} + \dots + b_kX_{ik} + a_1W_{i1} + a_2W_{i2} + a_3W_{i3} + e_i, \quad i = 1, \dots, n, \quad (3.17)$$

where, for $j = 1, 2, 3$, we have

$$W_{ij} = \begin{cases} 1 & \text{if the data point } i \text{ corresponds to category } j, \\ 0 & \text{otherwise.} \end{cases}$$

Note that for each index i , only one component of each of the variables W_{i1} , W_{i2} and W_{i3} is equal to unity, the other two being zero.



Note also that an indicator variable is *not* needed for the final category as its effect is absorbed by the overall constant b_0 . A typical application is to monthly data in which there is a seasonal component of uncertain effect. Thus *month* is the categorical variable. We need an indicator variable, D_i , for the *each of 11 months*. Note also that the use of indicator variables to represent the effect of categorical variables can greatly increase the number of coefficients to be estimated.

Demo 3.4: Introduce 11 monthly indicator variables for the Bank data and fit the new regression model to the first 53 data points; see [StatsWithIndicatorsBank.py](#).

3.2.2 Time related explanatory variables

If the observations $\{Y_i, X_{i1}, \dots, X_{ik}\}$, $i = 1, \dots, t$, come from a time series, then the explanatory variables are in this sense already time related. We may however include *time* itself as an explanatory variable, and even its powers. Precisely, for the model (3.17), including i , i^2 and i^3 as three additional variables leads to

$$Y_i = b_0 + b_1 X_{i1} + \dots + b_k X_{ik} + a_1 W_{i1} + a_2 W_{i2} + a_3 W_{i3} + a_4 i + a_5 i^2 + a_6 i^3 + e_i, \quad i = 1, \dots, t.$$

Demo 3.5: Introduce i as an explanatory variable as well as the 11 monthly indicator variables for the Bank data and fit the new regression model to the first 53 data points; you can run the code [StatsWithIndicatorsTimeBank.py](#) to generate the corresponding statistics.

3.2.3 Subset selection

When the number of explanatory variables is large, then the question arises as to whether some of the explanatory variables might be omitted because they have little influence on Y . Many ways have been suggested for selecting variables, including the following ones:

- (i) Best subset selection;
- (ii) Forward stepwise regression;
- (iii) Backward stepwise regression.

Makridakis et al. (1998) and Draper and Smith (1981) discuss this in more detail. Most packages offer routines to proceed with this. We do not discuss this further here. An important point is that when the design matrix is *non-orthogonal*, as invariably will be the case when the explanatory variable values arise from time series, then the rank order of significance of the coefficients as given by the P-values is not invariant, but depends on which coefficients happen to be included in the model. Thus any statement about the significance of a coefficient, is always conditional on which other coefficients have been fitted. Nevertheless an initial assessment can be made simply by ordering the coefficients according to their P-value. Based on this ranking, variables with large P-values can usually be omitted straight away.

Demo 3.6: Using [StatsComparisonBank.py](#), assess which explanatory variables are important for the bank data in the spreadsheet `Bank.xls`, including in the assessment the *time* variable, i , as well as the 11 monthly indicator variables 53 data points.

3.3 Multiple linear regression for forecasting

Suppose that the current time point is t , and that we have observations $\{Y_i, X_{i1}, \dots, X_{ik}\}$, $i = 1, \dots, t$ up to this point. We now wish to forecast Y for time points $i = t + 1, \dots, t + m$. However rather than use one of the forecasts of Chapter 2, such as Holt's LES forecast on Y directly, we might feel that the estimate of $E(Y_i)$ for $i = t + 1, \dots, t + m$, obtained from the multiple regression model of Y using X_1, \dots, X_k as explanatory variables, will be a better forecast.

To be able to do this we need forecasts

$$G_i = (G_{i1}, \dots, G_{ik}) \text{ of } X_i = (X_{i1}, \dots, X_{ik}), \text{ for } i = t + 1, \dots, t + m.$$

We can then use each of these forecasts in the predictor, \hat{Y} , as the forecast of $E(Y)$, i.e.,

$$F_i = \hat{Y}_i = G_i \hat{b} \text{ for } i = t + 1, \dots, t + m.$$

The forecasts G_i can be obtained in a number of ways, possibly exploiting relationships between the X_1, \dots, X_k themselves. We shall not consider these possibilities in any detail. Instead we shall assume that the X_j behave independently of one another, and can therefore each be forecast separately using a method like Holt's LES method, Holt-Winter method or whichever method is best suited to the time series in question.

We can however generate an estimate of the accuracy of these forecasts, using the standard error of the predictor \hat{Y}_i . Given G_i , we have

$$SE(\hat{Y}_i) = \sigma \sqrt{1 + G_i(X^T X)^{-1} G_i^T} \text{ for } i = t + 1, \dots, t + m,$$

where we can estimate σ using $\hat{\sigma} = \sqrt{MSE}$. Note that $SE(\hat{Y}_i)$ does not give a true measure of the total variability of \hat{Y}_i as it only expresses the variability of \hat{Y}_i given the value of G_i , and does not take into account the variability in G_i itself.

Demo 3.7: For the bank data in `Bank.xls`, produce forecasts for $D(EOM)$, for the time periods $i = 54, \dots, 59$, using the indicated explanatory variables and a selection of variables from the 11 monthly indicator variables and time. Use Holt's LES method to forecast the values of the explanatory variables for $i = 54, \dots, 59$ and also give confidence intervals for your forecasts of Y_i . The code [RegressionForecastsBank.py](#) can be used to generate the results.

3.3.1 Assumptions made and the validity of tests and forecasts

Multiple linear regression makes a number of assumptions, as listed below. If these are not correctly made, the above F and t-tests are not strictly valid, and forecast estimates may be unreliable. Consideration of the following should therefore be made for any particular dataset:

- (i) *Model form.* A linear relationship between the variables is assumed, but other relationships might be more appropriate; see, e.g., Section 8/6 of Makridakis et al. (1998) or Section 5/6 of Hyndman and Athanasopoulos (2014) for a description on polynomial or more general non-linear regression models.
- (ii) *Independence of residuals.* This is shown up by patterns in the ACF and PACF as well as by visual inspection. This may be improved by the addition of extra variables, as in Subsections 3.2.1 and 3.2.2. A complete approach to solving this problem is beyond the scope of this course, but may be achieved using a combination of regression with ARIMA (see Section 8/1 of the book by Makridakis et al. 1998).
- (iii) The model assumes that the residuals have the same variance, σ^2 , throughout the time series, a condition known as “homoscedasticity”. This may be corrected using a mathematical transformation; cf. Subsection 1.3.3.
- (iv) A normal distribution of error terms is assumed: violation affects the tests but not the ability to forecast. A mathematical transformation may also solve the problem.

3.3.2 Multicollinearity

Inspection should be made for multicollinearity between the explanatory variables X_1, \dots, X_k , as it can cause problems. Multicollinearity exists if there is perfect or near perfect correlation between the variables, or a linear combination of the variables. *It does not affect the ability of a model to predict*, but it can give computational problems. Multicollinearity also affects interpretation of the model results, in that the coefficients of the variables affected cannot be reliably interpreted. The effect of one variable on its own could also not be extracted. Scatterplots can be examined to look for such correlations: this method, however, is not guaranteed to find all multicollinearity, as linear combinations would not be discovered. (Principal component analysis could be applied, but the method is not included on this module.)

3.4 Python implementation guidelines

There python library

`statsmodels`

also includes various functions for regression analysis.

The main one that we will be using to generate statistics to analyse our model is

`ols`,

which means *ordinary least squares*. As you can see in the demo codes mentioned above, it is incredibly easy to use *ols*. For example, to build the basic model for our bank data, all you need is define your regression equation

```
formula = 'DEOM ~ AAA + Tto4 + D3to4',
```

where DEOM represents the dependent variable and AAA, Tto4, D3to4 are the dependent variables. *ols* then calls this model to produce the statistics, as follows:

```
results = ols(formula, data=series).fit(),
```

where *series* corresponds to the container of your data set.

The parameters resulting from these statistics, namely, the coefficients of the regression model (see second statistics table generated by the *ols* function), which are computed, can then be used to produce the forecasts using the formula presented in the introduction of this Section; cf. [RegressionForecastsBank.py](#).

3.5 Exercises

As for the previous workshops, the data files and demonstration spreadsheets are available on the Blackboard site (see Course Content).

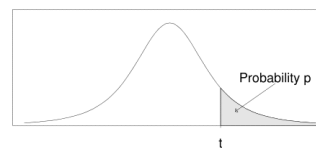
The aim of this assignment is to allow you to familiarise yourself with the demonstrations done in this chapter, using the same data set this time. The spreadsheet `Bank.xls` containing the data can be downloaded from the Chapter 3 data folder on blackboard. The file contains the bank data in the “data” sheet. The following questions are strongly connected to Demos 3.1 – 3.7:

1. Identify the observations Y and the design matrix X for each of the models built. Note that the main variables here are $DEOM=Y$, $AAA=X_1$, $3to4=X_2$, and $D3to4=X_3$.
2. Further inspect all the variables by building the time plots and assess which methods could potentially be the most suitable for each individual variable.
3. Produce scatterplots of each of the data sets DEOM, AAA, 3to4 and D3to4 against each other and address the following points:

- (a) What relationships can you observe between Y and the variables in the design matrix X ?
- (b) What relationships can be observed between the different explanatory variables?
- (c) How should such relationships be taken into account in the forecasting process?
4. Try different regression models, including indicator or time variables. Which model do you think is the most satisfactory, and why?
5. How are the forecasts calculated?
6. Calculate forecasts and confidence intervals using different model scenarios and analyse the results. What important observations can you make.

Appendix of Chapter 3 – Table of critical values for t

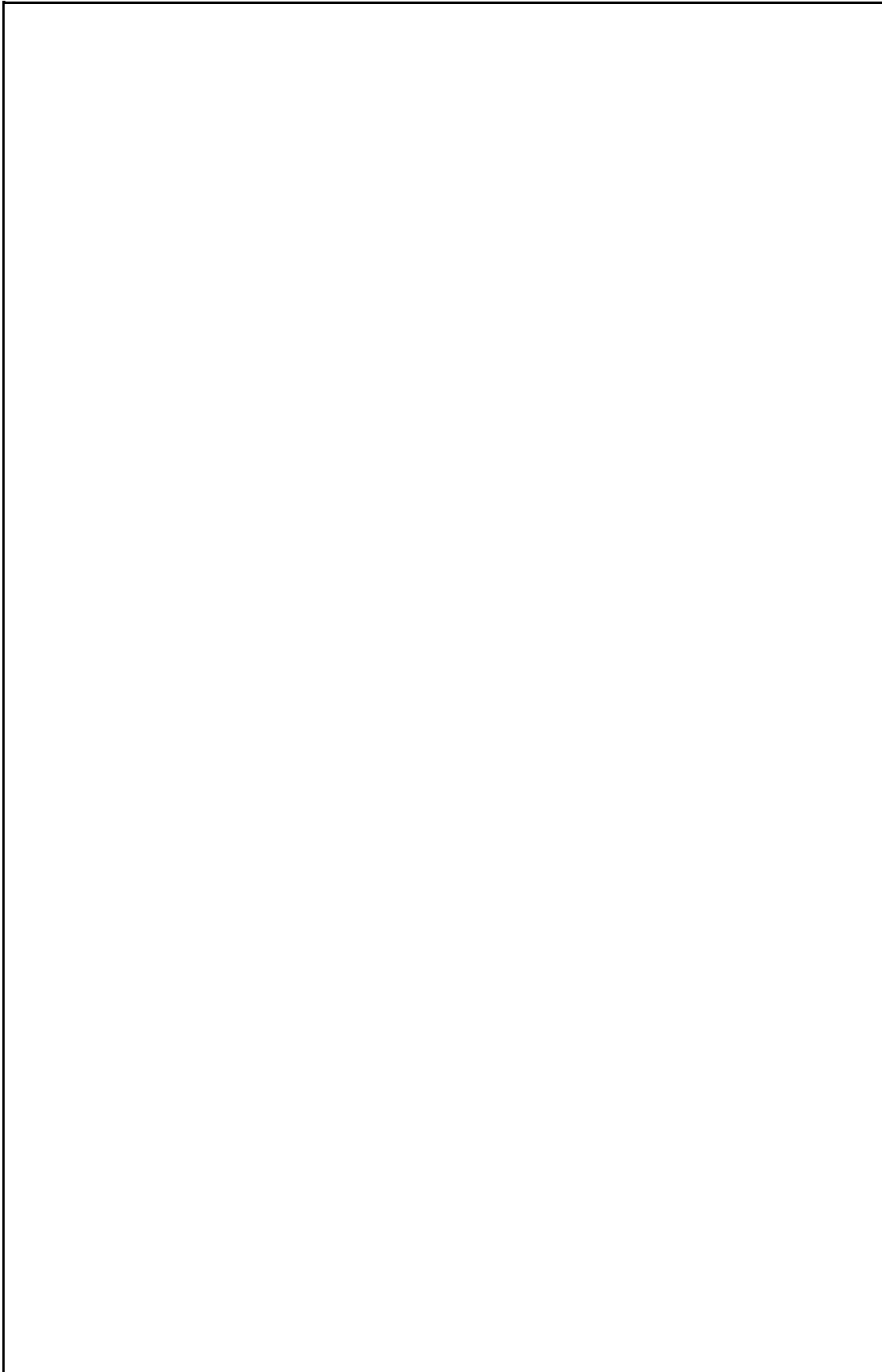
Table entry is the point t with the probability p lying above it. The first column gives the degrees of freedom. Use symmetry for negative t values.



df	Tail probability p				
	0.1	0.05	0.025	0.01	0.005
1	3.08	6.31	12.71	31.82	63.66
2	1.89	2.92	4.30	6.96	9.92
3	1.64	2.35	3.18	4.54	5.84
4	1.53	2.13	2.78	3.75	4.60
5	1.48	2.02	2.57	3.36	4.03
6	1.44	1.94	2.45	3.14	3.71
7	1.41	1.89	2.36	3.00	3.50
8	1.40	1.86	2.31	2.90	3.36
9	1.38	1.83	2.26	2.82	3.25
10	1.37	1.81	2.23	2.76	3.17
11	1.36	1.80	2.20	2.72	3.11
12	1.36	1.78	2.18	2.68	3.05
13	1.35	1.77	2.16	2.65	3.01
14	1.34	1.76	2.14	2.62	2.98
15	1.34	1.75	2.13	2.60	2.95
16	1.34	1.75	2.12	2.58	2.92
17	1.33	1.74	2.11	2.57	2.90
18	1.33	1.73	2.10	2.55	2.88
19	1.33	1.73	2.09	2.54	2.86
20	1.33	1.72	2.09	2.53	2.85
21	1.32	1.72	2.08	2.52	2.83
22	1.32	1.72	2.07	2.51	2.82
23	1.32	1.71	2.07	2.50	2.81
24	1.32	1.71	2.06	2.49	2.80
25	1.32	1.71	2.06	2.49	2.79
30	1.31	1.70	2.04	2.46	2.75
40	1.30	1.68	2.02	2.42	2.70
50	1.30	1.68	2.01	2.40	2.68
60	1.30	1.67	2.00	2.39	2.66
70	1.29	1.67	1.99	2.38	2.65
80	1.29	1.66	1.99	2.37	2.64
90	1.29	1.66	1.99	2.37	2.63
100	1.29	1.66	1.98	2.36	2.63
∞	1.28	1.64	1.96	2.33	2.58
	80%	90%	95%	98%	99%
	Confidence level				

Figure 3.2: Table of critical values for t statistics from Makridakis et al. (1998), page 621

This box can be used for notes on Chapter 3





4. The ARIMA method

Unlike in the previous chapter, we come back to a black-box method by considering a particular set of models used in advanced forecasting and time series analysis: AutoRegressive Integrated Moving Average (ARIMA) models. Box and Jenkins (1970) and more recently Box, Jenkins and Reinsel (1994) present the theoretical basis for these models. This type of method is based on the autocorrelation function (ACF), which was introduced in Chapter 1, see precisely (1.2). This provides a more refined method of forecasting than, for example, a moving averages method.

As with other forecasting methods, different phases of activity can be distinguished when using the Box-Jenkins method. These phases are described in Figure 4.3. The use of ARIMA models requires further preliminary analysis techniques, in addition to those already considered in the previous chapters. This chapter proceeds with descriptions of such advanced techniques, before describing ARIMA models and their selection.

4.1 Preliminary analysis

We begin by considering analyses of correlations: techniques that can be used in Phase 1 analysis and Phase 2 testing of ARIMA models; cf. Figure 4.3.

4.1.1 The partial autocorrelation function

Partial autocorrelations are used to measure the degree of association between observations at times t and $t - k$, Y_t and Y_{t-k} , when the effects of other time lags, $1, \dots, k - 1$, are removed. The usefulness of partial autocorrelations may be understood in the following example. Suppose there is significant autocorrelation between Y_t and Y_{t-1} . This implies that there is also significant correlation between Y_t and Y_{t-2} , since they are one time step apart. There is therefore significant autocorrelation between Y_t and Y_{t-2} , because they are both correlated with Y_{t-1} . However, it can only be known if there is an independent relationship between Y_t and Y_{t-2} if the effect of Y_{t-1} can be *partially out*.

Partial autocorrelations calculate true correlations between Y_t and Y_{t-1}, \dots, Y_k using a regression equation known as an *autoregression* (AR):

$$Y_t = b_0 + b_1 Y_{t-1} + b_2 Y_{t-2} + \dots + b_k Y_{t-k}.$$

Clearly this formula expresses a multiple linear regression as discussed in the previous chapter. Hence, the partial autocorrelation coefficients, b_i , $i = 1, \dots, k$, are estimated using the same technique described there; cf. Subsection 3.1.2.

Demo 4.1: Using the code [AcfPacfPlotDowJones.py](#) generate both the ACF and PACF of the Dow Jones data set in `DowJones.xls`. What patterns do you see in the ACF and PACF plots?

The PACF together with the ACF introduced in Section 1.1.2 have many applications; in particular, within this chapter we will use this concepts to determine whether a time series is white noise, stationary or seasonal. The definitions of white noise and stationarity together with related illustrations are provided below. Also, the ACF and PACF can be useful in identifying ARIMA models, in particular the pure AR and MA ones; see Subsections 4.2.1 and 4.2.2, respectively for further details.

4.1.2 A white noise model

When testing for suitability, a forecasting model is deemed to be sufficiently well suited to a particular application *if the forecast errors are purely random*. The residuals are then described as *white noise*. A simple example of a white noise model is given by

$$Y_t = c + e_t,$$

where c represents a constant overall level and e_t is a random error component.

Theoretically, all autocorrelations coefficients for series of random numbers must be zero. But as we have finite samples, each of the sample's autocorrelation will not be exactly zero. It has been shown that, if a time series is white noise, both autocorrelation coefficients and partial autocorrelation coefficients are approximately independent and normally distributed with mean zero and standard deviation $1/\sqrt{n}$, where n is the number of observations in the series. Hence, it is useful to plot the ACF and PACF with range $\pm 1.96/\sqrt{n}$, when analysing what coefficients are significant or to determine whether data are white noise. If any coefficients lie outside this range, the data are probably not white noise.

Demo 4.2: The code [WhiteNoise.py](#) generates the time plot, histogram, ACF, and PACF plot for an example of artificial white noise time series; the series made of 1,000 values randomly generated using the `gauss()` function from the `random` module. It has a mean of 0 standard deviation (sigma) of 1. Examine the ACF and PACF of the white noise model. Are they *truly* white noise?

4.1.3 Stationarity

The first step in the model selection process of an ARIMA model is to ensure that the time series at hand is *stationary*. If this is not the case, some action, namely *differencing* (see the next two subsection below), would have to be performed to make sure that we have stationarity.

Before we discuss some approaches to recognize that a time series is *non-stationary*, let us recall that a *stationary time series* is one whose properties do not depend on the time at which the series is observed. So 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 period of time.

Some cases can be confusing—a time series with cyclic behaviour (but not trend or seasonality) is stationary. That is because the cycles are not of 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.

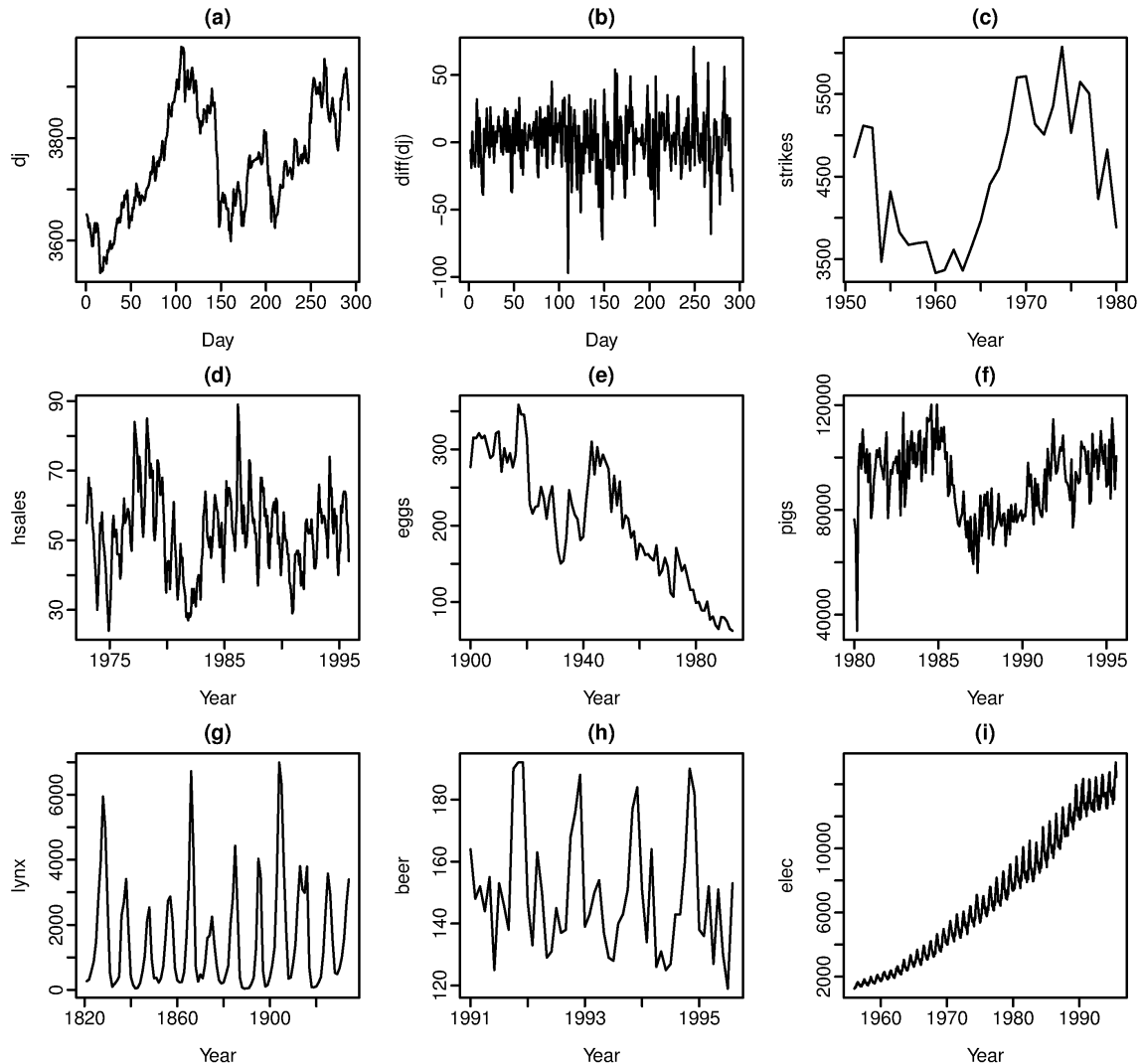


Figure 4.1: Timplots for time series to illustrate stationarity—Source: Page 214 of Hyndman and Athanasopoulos (2014)

Consider the nine series plotted in Figure 4.1. 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 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.

Plots of the ACF and PACF of a time series can also give clear evidence of non-stationarity. The autocorrelations of stationary data drop to zero quite quickly, while those for non-stationary data can take a number of time lags to become zero. The PACF of a non-stationary time series will

typically have a large spike close to 1 at lag 1. This can clearly be observed in Figure 4.2, where the left and right pictures correspond to the time series in pictures (a) and (b), respectively. Further details the data and ACF and PACF of the Dow Jones data can be found in the next demo.

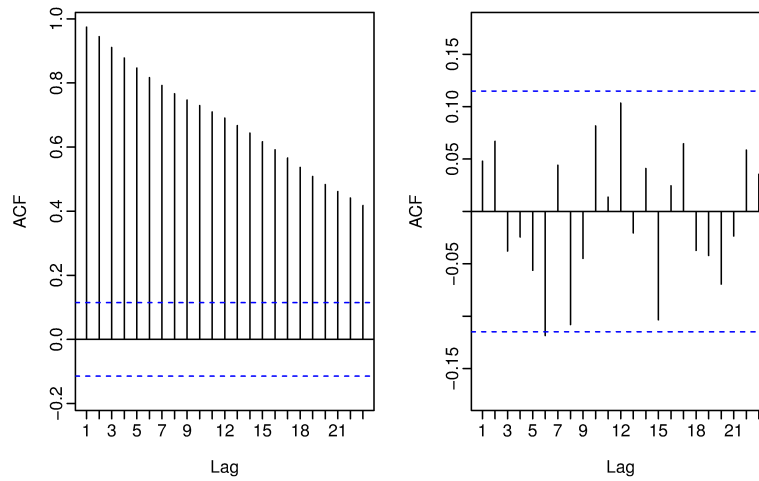


Figure 4.2: The ACF of the Dow-Jones index (left) and of the daily changes in the Dow-Jones index (right)–Source: Page 215 of Hyndman and Athanasopoulos (2014)

Demo 4.3: Look again at the ACF and PACF of the Dow Jones data: [AcfPacfPlotDowJones.py](#). Is the series stationary or not?



One way to determine more objectively if 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. The unit root test is out of the scope of the course, but those interested can look at Subsection 7/2/3 of the book of Makridakis et al. (1998).

Removing non-stationarity: the method of differencing

It is important to remove trends, or non-stationarity, from time series data prior to model building, since such autocorrelations dominate the ACF. One way of removing non-stationarity is through the method of *differencing*. The differenced series is defined as the *change* between each observation in the original time series:

$$Y'_t = Y_t - Y_{t-1}$$

Occasionally, such taking of first differences is insufficient to remove non-stationarity. In that case, second-order differences usually produce the desired effect:

$$Y''_t = Y'_t - Y'_{t-1}.$$

Demo 4.4: Apply first order differencing to the Dow Jones data and examine the ACF and PACF of the new series. What do these new ACF and PACF tell you about the data? Results can be obtained by running [1stDifferencingDowJones.py](#).

Seasonality and seasonal differencing

In this subsection we discuss seasonal differencing as a method to remove seasonality from a non-stationary time series. Before we proceed, let us recall that in Section 1.1, we introduced time

and seasonal plots as two possible steps to demonstrate that a time series is seasonal. Before, we introduce seasonal differencing, we would like to mention that ACF and PACF can also be used to further confirm that a time series is seasonal. This can be seen in the first part of Demo 4.2 below, where the corresponding spreadsheet illustrates seasonality with period $s = 12$. Clearly, the similar pattern repeats itself after every 12 time lags; in particular, in the ACF, spikes appear at the 12th, 24th, etc., time lags.

A seasonal difference is the difference between an observation and the corresponding observation from the previous year, quarter or month as appropriate, where s is the number of time periods back. For example, with monthly data, $s = 12$ and the seasonal difference is obtained as

$$Y'_t = Y_t - Y_{t-12}.$$

Seasonal differencing can be repeated to obtain second-order seasonal differencing, although this is rarely needed. It can happen that both seasonal and first differences are applicable: in this case, it makes no difference to the result which is done first. Seasonal differences can be more easily observed after first differencing, especially in the case of a strong trend.

Demo 4.5: Consider the Australian monthly electricity production data set and proceed with the following steps (see [SeasonalDifferencingElectricity.py](#)):

1. Produce the ACF and PACF of the series. Then apply seasonal differencing and see the changes.
2. Apply both seasonal and first differencing to the series. Does white noise result?

Backshift notation

The backshift notation is commonly used to represent ARIMA models. It uses the operator B , which shifts data back one period:

$$BY_t = Y_{t-1}.$$

Two applications of B shift the data back two periods:

$$B(BY_t) = B^2Y_t = Y_{t-2}.$$

For monthly data, we can use the notation B^{12} to “shift back to the same month last year”:

$$B^{12}Y_t = Y_{t-12}.$$

In general, B^s represents “shift back s time periods”. Note that a first difference is represented by $1 - B$ given that we have

$$Y'_t = Y_t - Y_{t-1} = Y_t - BY_t = (1 - B)Y_t.$$

Likewise, a 2^{nd} order difference is given by $(1 - B)^2$.

The “backshift” notation is convenient because terms can be multiplied together to see the combined effect. For example, a seasonal difference followed by a first difference can be written as:

$$(1 - B)(1 - B^s)Y_t = (1 - B - B^s + B^{s+1})Y_t = Y_t - Y_{t-1} - Y_{t-s} + Y_{t-s+1}$$

In the next section, we will use the backshift notation to describe a multitude of ARIMA models that can possibly be used.

4.2 ARIMA models

Here, we provide the main classes of the ARIMA model, starting from pure autoregression and moving average ones.

4.2.1 Autoregression (AR) models

Models that use the AR equation as described in Section 5.1.1 are termed *AR models*. They are classified by the number of time lags, p , used in the autoregression. In general, a p th order AR model, or $AR(p)$ model, is written as:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + e_t,$$

where c is a constant, ϕ_j are parameters to be determined and e_t is the error term. There are constraints on the allowable values of ϕ_j :

- For $p = 1$, $-1 < \phi_1 < 1$.
- For $p = 2$, $-1 < \phi_2 < 1$, $\phi_2 + \phi_1 < 1$ and $\phi_2 - \phi_1 < 1$.
- For $p \geq 3$, more complicated conditions hold.

An example of an $AR(1)$ model is

$$Y_t = 3 + 0.7Y_{t-1} + e_t,$$

where e_t is normally distributed with mean 0 and variance 1.

Demo 4.6: Build this $AR(1)$ model, make a timeplot and look at the ACF and PACF. Try varying the variance to 0.1 and see the results (see `DataAR1model.xls` and [AcfPacfAR1model.py](#)).

For an $AR(1)$ model, typically the ACF shows autocorrelations dying down to zero, while there is only one peak in the PACF at lag 1. With real-life or empirical data, random effects will give a more varied picture, as shown by the higher variance example above.

The general expression of the AR model can be rewritten in backshift notation as

$$(1 - \phi_1 B - \dots - \phi_p B^p)Y_t = c + e_t.$$

4.2.2 Moving Average (MA) models

As well as regressing on the observations at previous time lags, as in AR models, we can also regress on the error terms at previous time lags. Such models are called Moving Average (MA) models. (Nothing to do with the moving average described in Chapter 2.) Again, an MA model is classified by the number of time lags, q , used in the regression. A general $MA(q)$ is written as:

$$Y_t = c + e_t - \theta_1 e_{t-1} - \dots - \theta_q e_{t-q},$$

where c is constant, θ_j are parameters to be determined and e_t are the error terms.

As with AR models, there are restrictions on the allowable values of θ_j :

- For $q = 1$, $-1 < \theta_1 < 1$.
- For $q = 2$, $-1 < \theta_2 < 1$, $\theta_2 + \theta_1 < 1$ and $\theta_2 - \theta_1 < 1$.
- For $q \geq 3$, more complicated conditions hold.

An example of an $MA(1)$ model is:

$$Y_t = 10 + e_t - 0.7e_{t-1},$$

where e_t is normally distributed with mean 0 and variance 1.

Demo 4.7: Generate this $MA(1)$ model, make a timeplot and look at the ACF and PACF. Again, try varying the variance to 0.1 and see the results (`DataMA1model.xls` and [AcfPacfMA1model.py](#)).

For an $MA(1)$ model, typically the ACF shows only one peak in the ACF at lag 1, while the PACF shows partial autocorrelations dying down to zero. With real-life or empirical data, random

effects will give a slightly more varied picture, as with the higher variance example above. Thus it can be seen that the MA(1) model is a mirror image of the AR(1) model, as far as the ACF and PACF are concerned.

The general expression of the MA model in the can be written backshift notation as

$$Y_t = c + (1 - \theta_1 B - \dots - \theta_q B^q) e_t.$$

4.2.3 ARIMA (p, d, q) models

AR and MA models can be combined with taking *differences* to give a wide variety of effects. These are known as the ARIMA(p, d, q) series of models. These combine:

- AR(p) autoregression to time lag p ;
- I(d) differences of order d ;
- MA(q) regression on errors to lag q .

In other words, p is the number of autoregression terms, d is the number of nonseasonal differences needed for stationarity and q is the number of lagged forecast errors in the prediction equation. As example, note that ARIMA(2, 1, 1) can be written as

$$Y_t = c + (1 + \phi_1) Y_{t-1} - \phi_1 Y_{t-2} + e_t - \theta_1 e_{t-1},$$

which is equivalent to the following equation in backshift notation:

$$(1 - \phi_1 B)(1 - B)Y_t = c + (1 - \theta_1 B)e_t.$$

The general expression of the ARIMA(p, d, q) model in the can be written backshift notation as

$$(1 - \phi_1 B - \dots - \phi_p B^p)(1 - B)^d Y_t = c + (1 - \theta_1 B - \dots - \theta_q B^q) e_t,$$

where ϕ_i, θ_i are parameters to be determined.

4.2.4 ARIMA(p, d, q)(P, D, Q)s models

The above models can (finally) be combined also with seasonal differences of order s . Just as *consecutive* data points might show AR, MA or mixed ARIMA properties, so might data separated *by a whole season* show the same properties. The ARIMA notation is extended to such seasonal components thus:

$$\text{ARIMA}(p, d, q)(P, D, Q)s,$$

where s is the number of time periods per season, with

- seasonal autoregression to time lag P ;
- seasonal differences of order D ;
- seasonal regression on errors to lag Q .

P, D and Q can be further described in a way similar to the corresponding terms in the nonseasonal framework. The mathematical expression of this model in terms of the backshift notation can be written as

$$\begin{aligned} (1 - \phi_1 B - \dots - \phi_p B^p)(1 - \Phi_1 B^s - \dots - \Phi_P B^{sP})(1 - B)^d(1 - B^s)^D Y_t \\ = (1 - \theta_1 B - \dots - \theta_q B^q)(1 - \Theta_1 B^s - \dots - \Theta_Q B^{sQ}) e_t. \end{aligned} \quad (4.1)$$

To conclude this section, let us recall that the (Box-Jenkins) ARIMA models provide a powerful approach to time series analysis that pays special attention to correlation between observations, a feature ignored by the more basic forecasting models. However, the simpler techniques should not be ignored. A large and careful study carried out by Makridakis seems to show that, very often, the simpler techniques described in this module are just as powerful. There is a good reason for this. The SES, LES and Holt-Winter's Additive methods are equivalent to special (simple) cases of ARIMA models. These simple models may fit many data sets quite well.

- How to detect ARIMA models in the general case, i.e., no-AR or MA-type models which cannot be identified with ACF and PACF?

4.3 Model selection and forecasting

The process of model selection and forecasting can be subdivided into three phases, as described in the Figure 4.3 provided below; i.e., model identification, parameters estimation and testing, and forecasting. Next, we respectively provided details on how each of these steps is conducted.

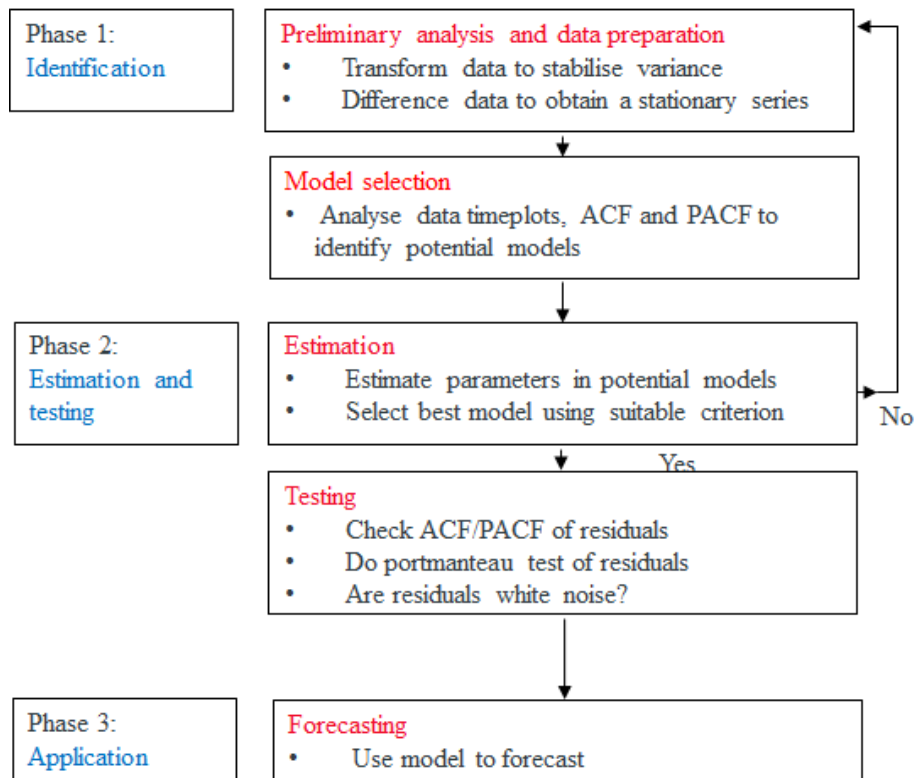


Figure 4.3: The three phases of the ARIMA method

4.3.1 Phase 1: Identification

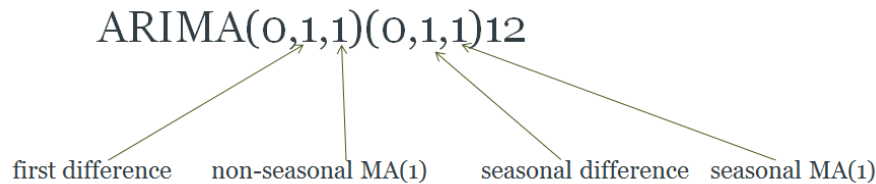
Here, we suggest an approach to identify which type of ARIMA model is appropriate for the data set being studied.

1. Make a timeplot. Look out for patterns or unusual features. Decide if it's necessary to stabilise the variance and if so, apply a power transformation (see Subsection 1.3.3).
2. Consider whether the time series is now stationary, using the timeplot as well as ACF and PACF. For example, have you got a *white noise* or not?
3. If the time series is non-stationary, try differencing. If there are seasonal effects, take seasonal differences. If non-stationarity still remains, take first differences, and second if necessary.
4. When stationarity has been achieved, look at the ACF/PACF of what remains. Do the patterns suggest AR or MA models, seasonal or not?

In the next demo, we illustrate the model identification with an example of a seasonal time series.

Demo 4.8: Consider the *printing and writing paper sales* data set in `PrintingWriting.xls`. First, run the ACF and PACF programs for the original data; What do you observe in the graphs? Spikes

in the ACF at time lags 12, 24, 36 suggest seasonality, so apply seasonal differencing of 12 months. Now, looking at the ACF, stationarity has not yet been achieved: the correlations do not die down quickly. Apply first differencing and look at the results. Something like the MA(1) pattern can be seen with a spike in the ACF at lag 1 and partial correlations slowly dying down in the PACF. Also, something similar could be happening seasonally, with a spike in the ACF at lag 12, and annual partial correlations dying down from lag 12. So, a seasonal MA(1) model is also suggested. Hence we get a possible identification of a model for the data as



See [AcfPacfPlotPrintingWriting.py](#), [SeasonalDifferencingPrintingWriting.py](#), and [Seasonal1stDifferencingPrintingWriting.py](#) for the preliminary analysis and model selection. [SARIMAPrintingWriting.py](#) generates many model statistics (we discuss some further below) and forecast results.

This model is known as the “airline model” as it was first applied to airline data by Box and Jenkins (1970). It is one of the most commonly used seasonal ARIMA models.

R The identification of higher order models, with parameters $p, q, P, Q \geq 2$, is a difficult task needing much experience. It is beyond the scope of this course to provide such experience in ARIMA forecasting. However, Section 7/3 of the book by Makridakis et al. (1998) gives some insights into higher order models.

4.3.2 Phase 2: Parameters selection/estimation and testing

As shown in Section 4.2, ARIMA models can be written in what is called *backshift* notation, which is described in Subsection 4.1.3. There, we can observe that a complete model

$$\text{ARIMA}(p, d, q)(P, D, Q)s$$

involves two classes of parameters; i.e., the ones determining the model (p, d, q, P, D, Q , and s) and the regression ones:

$$\phi_1, \phi_2, \dots, \phi_p, \Phi_1, \Phi_2, \dots, \Phi_P, \theta_1, \theta_2, \dots, \theta_q, \Theta_1, \Theta_2, \dots, \Theta_Q, \sigma, \quad (4.2)$$

where σ is the standard deviation of the errors, e_t .

The first class of parameters are required before the regression ones can be determined. For s , seasonal and ACF plots can be used as discussed in Chapter 1, to find the corresponding value, in case seasonality is present in the time series. Obviously, d and D are based on differencing, while initial guesses for p, q, P , and Q can be determined based on patterns in the ACF and PACF plots, as per Demo 4.8. Subsequently, other values can be tested and monitored with the AIC (see below) – models with the smallest AIC are usually the best!

Finally, as for the regression-type parameters (4.2), you can easily see that the ARIMA models (4.1) can be expressed as a form of multiple linear regression where the independent variables correspond to the values of the time series at different time lags. With such an equation, it is possible to calculate the values of these parameters by least squares methods in the previous chapter; cf. Subsection 3.1.2.

Selecting the best model using the AIC

It may happen that several ARIMA models give a good fit to the data. In fact, a better fit can always be gained simply by increasing the number of parameters in a model. However, this could mean that just randomness is being modelled, rather than innate characteristics. It is therefore helpful when comparing models to use a measure of good fit that penalises overfitting. One such measure is Akaike's Information Criterion (AIC) (Akaike 1974):

$$\text{AIC} = -2\log L + 2m,$$

where L is the likelihood, and $m = p + q + P + Q$. Hence, by varying the choices of p , q , P , and Q , the lowest attainable AIC is thus sought.

Portmanteau tests for white noise

Instead of examining each individual autocorrelation coefficient to test whether *white noise* is achieved, "portmanteau" tests can be used which consider the coefficients taken together. One such test is the Ljung-Box test (Ljung and Box, 1978), which employs the Q^* statistic

$$Q^* = n(n+2) \sum_{k=1}^h (n-k)^{-1} r_k^2,$$

where r_k represents the correlation values, n the number of observations and h is the maximum time lags considered. If the residuals from a model are white noise, Q^* has a chi-square (χ^2) distribution with $(h-m)$ degrees of freedom, where m is the number of parameters in the model. (When applied to raw data, no parameters having been fitted, the value $m = 0$ is used.) If the Q^* statistic lies in the right-hand 5% tail of the χ^2 distribution (i.e. with a P-value of < 0.05), it is normally concluded that the data are not white noise. However, it is recommended that the evidence of such a test should not be used alone to accept a model, as some poorly fitting models may not thereby be rejected.

4.3.3 Phase 3: Forecasting using the ARIMA model

Although there are many software tools that can be used to generate forecasts based on the ARIMA models introduced above, it remains important to have a general understanding on how the aforementioned steps finally lead to the results. The process can follow the following three steps:

1. Expand the ARIMA formula so that Y_t is expressed in the form of a conventional regression equation; i.e., with Y_t on the left hand side and all other terms are on the right.
2. Rewrite the equation by replacing t by $T+h$, where T represent the current time and h stands for the number of periods ahead for which forecasts should determined.
3. On the right hand side of the equation, replace future observations by their forecasts, future errors by zero, and past errors by the corresponding residuals.

To illustrate this, we consider the *airline model* ARIMA(0, 1, 1)(0, 1, 1)₁₂ discussed in Demo 4.8. It follows from the seasonal ARIMA model in (4.1) that its backshift notation is

$$(1-B)(1-B^{12})Y_t = (1-\theta_1 B)(1-\Theta_1 B^{12})e_t.$$

Step 1 above leads to the following expression of Y_t :

$$Y_t = Y_{t-1} + Y_{t-12} - Y_{t-13} + e_t - \theta_1 e_{t-1} - \Theta_1 e_{t-12} + \theta_1 e_{t-13}. \quad (4.3)$$

As for step 2, if we want to calculate the forecast at time $t+1$ (one period ahead), which means taking $T = t$ and $h = 1$, we replace t by $t+1$ in (4.3) and get

$$Y_{t+1} = Y_t + Y_{t-11} - Y_{t-12} + e_{t+1} - \theta_1 e_t - \Theta_1 e_{t-11} + \theta_1 e_{t-12}. \quad (4.4)$$

The term e_{t+1} will not be known because the expected value of future random errors has to be taken as zero, but from the fitted model it will be possible to replace the values e_t , e_{t-11} , and e_{t-12} by their empirically determined values; i.e., the residuals for times t , $t - 11$ and $t - 12$, respectively. Of course, as we forecast further and further ahead, there will be no empirical values for the e terms after a while, and so their expected values will all be zero.

For the Y values, at the start of the forecasting process, we will know the values Y_t , Y_{t-11} and Y_{t-12} . After a while, however, the Y values in equation (4.4) will be forecasted values rather than known past values.

Given that the values of the parameters θ_1 , and Θ_1 can be obtained using the regression method discussed in the previous chapter, replacing them in (4.4) will lead to the values of the desired forecast at a desired period.

4.4 Python implementation guidelines

There python library

`statsmodels`

also includes various tools to computing forecasts based on the ARIMA method. One of them is called ARIMA and can be imported into a python code as follows:

```
from statsmodels.tsa.arima_model import ARIMA.
```

Subsequently, if the data set is loaded in `series`, then ARIMA can be used with

```
model = ARIMA(series, order=(p, d, q)),
```

where p , d , and q represent the parameters of the model, which can be obtained from a preliminary analysis of the ACF and PACF as done in Demo 4.8, for example.

However, if a time series is seasonal, ARIMA might not allow the seasonal component to be incorporated in the model. Hence, to proceed in this case, the seasonal ARIMA model known as SARIMAX (Seasonal AutoRegressive Integrated Moving Averages with eXogenous regressors).

```
model = sm.tsa.statespace.SARIMAX(series, order=(p, d, q),
                                   seasonal_order=(P, D, Q, s)),
```

where `sm` corresponds to the `statsmodels.api`; p , d , and q are non-seasonal parameters of the model, while P , D , and Q are seasonal parameters. As for the parameter s , it represent the number of period in the seasonality of the time series. We have $s = 12$ for monthly observations, as it is the case for most the data sets in this course.

4.5 Exercises

The questions below are closely related to the demos in this chapter, but the data sets used here is the building material (`BuildingMaterials.xls`) and cement production (`CementProduction.xls`) ones, available in the Workshop 4 folder. For each of these data sets, do the following:

1. Examine the timeplot of the building material data set. Would it be appropriate to apply a transform to stabilise the variance?
2. Try out the square root and log transforms. (See Subsection 1.3.3 of the lecture notes; cf. Demo 1.9).
3. Examine the ACF and PACF of the time series; what patterns do you see? Is it stationary?
4. Apply first order differencing (as per Demo 4.4, `1stDifferencingDowJones.py`).
5. There is evidence for seasonality, with peaks in the ACF and PACF at the 12th time lag. So apply both seasonal and first differencing. Does white noise result? (cf. Demo 4.8; see [Seasonal1stDifferencingPrintingWriting.py](#)).

6. Now to consider likely ARIMA models (see Demos 4.6–4.8). Look at the resulting ACFs and PACFs from differencing. Is an AR or MA model suggested? (See [AcfPacfAR1model.py](#) and [AcfPacfAM1model.py](#).) Decide on a likely ARIMA model as a starting point.
7. Run this ARIMA model as per [SARIMAPrintingWriting.py](#), and note the AIC and Ljung-Box results. Does the model pass the Ljung-Box tests? Try out some other similar models with the same differencing (varying p , q , P , and Q), and find the best one that passes the Ljung-Box tests. Draw plots of the forecast with suitable confidence intervals.

