

Applied Time Series Analysis with R

Stéphane Guerrier, Roberto Molinari and Haotian Xu

October 31 2018

Contents

Part I

Foundation

Chapter 1

Introduction

Welcome to “Applied Time Series Analysis with R”. This book is intended as a support for the course of STAT 463 (Applied Time Series Analysis) given at Penn State University. It contains an overview of the basic procedures to adequately approach a time series analysis with insight to more advanced analysis of time series. It firstly introduces the basic concepts and theory to appropriately use the applied tools that are presented in the second (and main) part of the book. In the latter part the reader will learn how to use descriptive analysis to identify the important characteristics of a time series and then employ modelling and inference techniques (made available through R functions) that allow to describe a time series and make predictions. The last part of the book will give introductory notions on more advanced analysis of time series where the reader will achieve a basic understanding of the tools available to analyse more complex characteristics of time series.

This document is **under development** and it is therefore preferable to always access the text online to be sure you are using the most up-to-date version. Due to its current development, you may encounter errors ranging from broken code to typos or poorly explained topics. If you do, please let us know! Simply add an issue to the GitHub repository used for this document (which can be accessed here <https://github.com/SMAC-Group/ts/issues>) and we will make the changes as soon as possible. In addition, if you know RMarkdown and are familiar with GitHub, make a pull request and fix an issue yourself.

1.1 Conventions

Throughout this book, R code will be typeset using a monospace font which is syntax highlighted. For example:

```
a = pi
b = 0.5
sin(a*b)
```

Similarly, R output lines (that usually appear in your Console) will begin with **##** and will not be syntax highlighted. The output of the above example is the following:

```
## [1] 1
```

Aside from R code and its outputs, this book will also insert some boxes that will draw the reader’s attention to important, curious or otherwise informative details. An example of these boxes was seen at the beginning of this introduction where an important aspect was pointed out to the reader regarding the “under construction” nature of this book. Therefore the following boxes and symbols can be used to represent information of different nature:

This is an important piece of information.

This is some additional information that could be useful to the reader.

This is something that the reader should pay caution to but should not create major problems if not considered.

This is a warning which should be heeded by the reader to avoid problems of different nature.

This is a tip for the reader when following or developing something based on this book.

Using the same convention as in `?`, the symbol `?` indicates a technically difficult section which may be skipped without interrupting the flow of the discussion.

1.2 Bibliographic Note

This is not the first (or the last) book that has been written on time series analysis. Indeed, this can be seen as a book that brings together and reorganizes information and material from other sources structuring and tailoring it to a course in basic time series analysis. The main and excellent references (which are far from being an exhaustive review of literature) that can be used to have a more in-depth view of different aspects treated in this book are `?`, `?` and `?`.

1.3 Acknowledgements

The text has benefited greatly from the contributions of many people who have provided extremely useful comments, suggestions and corrections. These are:

- Ziyang Wang
- Haoxian Zhong
- Zhihan Xiong
- Nathanael Claussen
- Justin Lee

The authors are particularly grateful to James Balamuta who introduced them to the use of the different tools provided by the RStudio environment and greatly contributed to an earlier version of this book:

- James Balamuta

1.4 License

You can redistribute it and/or modify this book under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA) 4.0 License.

Chapter 2

Basic Elements of Time Series

“Prévoir consiste à projeter dans l’avenir ce qu’on a perçu dans le passé.” – Henri Bergson

To make use of the R code within this chapter you will need to install (if not already done) and load the following libraries:

- `simts`;
- `astsa`;
- `mgcv`.

These libraries can be install as follows:

```
install.packages(c("devtools", "astsa", "mgcv"))
devtools::install_github("SMAC-Group/simts")
```

and simply load them using:

```
library(astsa)
library(mgcv)
library(simts)
```

We can start the discussion on the basic elements of time series by using a practical example from real data made available through the R software. The data represent the global mean land–ocean temperature shifts from 1880 to 2015 (with base index being the average temperatures from 1951 to 1980) and this time series is represented in the plot below.

```
# Load data
data(globtemp, package = "astsa")

# Construct gts object
globtemp = gts(globtemp, start = 1880, freq = 1, unit_ts = "C", name_ts = "Global Temperature Deviation")

# Plot time series
plot(globtemp)
```

These data have been used as a support in favour of the argument that the global temperatures are increasing and that global warming has occurred over the last half of the twentieth century. The first approach that one would take is to try and measure the average increase by fitting a model having the form:

$$X_t = f(t) + \varepsilon_t,$$

where X_t denotes the global temperatures deviation and $f(\cdot)$ is a “smooth” function such that $\mathbb{E}[X_t] - f(t) = 0$ for all t . In general, ε_t is assumed to follow a normal distribution for simplicity. The goal in this context would therefore be to evaluate if $f(t)$ (or a suitable estimator of this function) is an increasing function (especially over the last decades). In order to do so, we would require the residuals from the fitted model to be independently and identically distributed (iid). Let us fit a (nonparametric) model with the years (time) as explanatory variable using the code below:

```
time = gts_time(globtemp)
fit = gam(globtemp ~ s(time))
```

and check the residuals from this model using:

```
check(fit, simple = TRUE)
```

```
## Warning in if (class(model) == "fitsimts") {: the condition has length > 1
## and only the first element will be used

## Warning in check(fit, simple = TRUE): If 'lm' model is considered, only the
## full diagnostic plots can be provided, not the simple version.
```

It can be seen from the upper left plot that the trend appears to be removed and, if looking at the residuals as one would usually do in a regression framework, the residual plots seem to suggest that the modelling has done a relatively good job since no particular pattern seems to emerge and their distribution is quite close to being Gaussian.

However, is it possible to conclude from the plots that the data are *iid* (i.e. independent and identically distributed)? More specifically, can we assume that the residuals are independent? This is a fundamental question in order for inference procedures to be carried out in an appropriate manner and to limit false conclusions. Let us provide an example through a simulated data set where we know that there is an upward trend through time and our goal would be to show that this trend exists. In order to do so we consider a simple model where $f(t)$ has a simple parametric form, i.e. $f(t) = \beta \cdot t$ and we employ the following data generating process:

$$X_t = \beta \cdot t + Y_t,$$

where

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \varepsilon_t,$$

and where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. Intuitively, Y_t is not an *iid* sequence of random variables except in the case where $\phi_1 = \phi_2 = 0$. In the following chapters we shall see that this intuition is correct and that this model is known as an AR(2) model. Considering this, we simulate two cases where, in the first, the residuals are actually *iid* Gaussian while, in the second, the residuals are Gaussian but are dependent over time. In the first case, the only parameters that explain X_t are $\beta = 5 \cdot 10^{-3}$ and $\sigma^2 = 1$ since the residuals Y_t are *iid* (i.e. $\phi_1 = \phi_2 = 0$). In the second case however, aside from the mentioned parameters we also have $\phi_1 = 0.8897$, $\phi_2 = -0.4858$. In both cases, we perform the hypothesis test:

$$H_0 : \beta = 0$$

$$H_1 : \beta > 0$$

as our hope is to prove, similarly to the global temperature deviation example, that $f(t)$ is an increasing function. Our syntetic data are simulated as follows:

```
# Set seed for reproducibility
set.seed(9)

# Define sample size
```

```

n = 100

# Define beta
beta = 0.005

# Define sigma2
sigma2 = 1

# Simulation of Yt
Yt_case1 = gen_gts(WN(sigma2 = sigma2), n = n)
Yt_case2 = gen_gts(AR(phi = c(0.95, -0.5), sigma2 = sigma2), n = n)

# Define explanatory variable (time)
time = 1:n

# Simulation of Xt
Xt_case1 = beta*time + Yt_case1
Xt_case2 = beta*time + Yt_case2

# Fit a linear models
model1 <- lm(Xt_case1 ~ time + 0)
model2 <- lm(Xt_case2 ~ time + 0)

```

The “summary” of our model on the first dataset is given by

```

summary(model1)

##
## Call:
## lm(formula = Xt_case1 ~ time + 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.5985 -0.7023 -0.1398  0.4444  2.7098
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## time 0.003930   0.001647   2.386   0.019 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9583 on 99 degrees of freedom
## Multiple R-squared:  0.05436,    Adjusted R-squared:  0.04481
## F-statistic: 5.691 on 1 and 99 DF,  p-value: 0.01895

```

As can be seen, in the first case the estimated slope (≈ 0.004) is close to the true slope (0.005) and is significant (i.e. the p-value is smaller than the common rejection level 0.05) since the p-value of the above mentioned test is given by 0.0095. Hence, from this inference procedure we can conclude at the 5% significance level that the slope is significantly larger than zero and is roughly equal to 0.004 (which is relatively close to the truth). However, let us perform the same analysis when the residuals are not independent (the second case) by examining its “summary”:

```

summary(model2)

```

```

##
## Call:

```

```
## lm(formula = Xt_case2 ~ time + 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.6916 -1.1184  0.2323  1.1253  2.6198
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## time 0.0009877  0.0026435   0.374   0.709
##
## Residual standard error: 1.538 on 99 degrees of freedom
## Multiple R-squared:  0.001408,    Adjusted R-squared:  -0.008679
## F-statistic: 0.1396 on 1 and 99 DF,  p-value: 0.7095
```

In this case we can observe that the p-value of the above mentioned test is given by 0.3547 and is therefore greater than the arbitrary value of 0.05. Consequently, we don't have evidence to conclude that the slope coefficient is larger than zero (i.e. we fail to reject H_0) although it is actually so in reality. Therefore, the inference procedures can be misleading when not taking into account other possible significant variables or, in this case, forms of dependence that can hide true underlying effects. The above is only one example and there are therefore cases where, despite dependence in the residuals, the estimated slope would be deemed significant even when not considering this dependence structure. However, if we decided to repeat this experiment using a larger quantity of simulated samples, we would probably see that we fail to reject the null hypothesis much more frequently in the case where we don't consider dependence when there actually is.

These examples therefore highlight how the approach to analysing time series does not only rely on finding an appropriate model that describes the evolution of a variable as a function of time (which is deterministic). Indeed, one of the main focuses of time series analysis consists in modelling the dependence structure that describes how random variables impact each other as a function of time. In other words, a time series is a collection of random variables whose interaction and dependence structure is indexed by time. Based on this structure, one of the main goals of time series analysis is to correctly estimate the dependence mechanism and consequently deliver forecasts that are as accurate as possible considering the deterministic functions of time (and other variables) as well as the random dependence structure.

2.1 The Wold Decomposition

The previous discussion highlighted how a time series can be decomposed into a deterministic component and a random component. Leaving aside technical rigour, this characteristic of time series was put forward in Wold's Decomposition Theorem who postulated that a time series (Y_t) (where $t = 1, \dots, n$ represents the time index) can be very generically represented as follows:

$$Y_t = D_t + W_t,$$

where D_t represents the deterministic part (or *signal*) that can be modelled through the standard modelling techniques (e.g. linear regression) and W_t that, restricting ourselves to a general class of processes, represents the random part (*noise*) that requires the analytical and modelling approaches that will be tackled in this book.

Typically, we have $\mathbb{E}[Y_t] \neq 0$ while $\mathbb{E}[W_t] = 0$ (although we may have $\mathbb{E}[W_t|W_{t-1}, \dots, W_1] \neq 0$). Such models impose some parametric structure which represents a convenient and flexible way of studying time series as well as a means to evaluate *future* values of the series through forecasting. As we will see, predicting future values is one of the main aspects of time series analysis. However, making predictions is often a daunting task or as famously stated by Nils Bohr:

“Prediction is very difficult, especially about the future.”

There are plenty of examples of predictions that turned out to be completely erroneous. For example, three days before the 1929 crash, Irving Fisher, Professor of Economics at Yale University, famously predicted:

“Stock prices have reached what looks like a permanently high plateau”.

Another example is given by Thomas Watson, president of IBM, who said in 1943:

“I think there is a world market for maybe five computers.”

Let us now briefly discuss the two components of a time series.

2.1.1 The Deterministic Component (Signal)

Before shifting our focus to the random component of time series, we will first just underline the main features that should be taken into account for the deterministic component. The first feature that should be analysed is the *trend* that characterises the time series, more specifically the behaviour of the variable of interest as a specific function of time (as the global temperature time series seen earlier). Let us consider another example borrowed from ? of time series based on real data, i.e. the quarterly earnings of Johnson & Johnson between 1960 and 1980 represented below.

```
# Load data
data(jj, package = "astsa")

# Construct gts object
jj = gts(jj, start = 1960, freq = 4, unit_ts = "$", name_ts = "Quarterly Earnings per Share", data_name

# Plot time series
plot(jj)
```

As can be seen from the plot, the earnings appear to grow over time, therefore we can imagine fitting a straight line to this data to describe its behaviour by considering the following model:

$$X_t = \alpha + \beta t + \varepsilon_t, \quad (2.1)$$

where ε_t is iid Gaussian. The results are presented in the graph below:

```
# Fit linear regression
time_jj = gts_time(jj)
fit_jj1 = lm(as.vector(jj) ~ time_jj)

# Plot results and add regression line
plot(jj)
lines(time_jj, predict(fit_jj1), col = "red")
legend("bottomright", c("Time series", "Regression line"),
      col = c("blue4", "red"), bty = "n", lwd = 1)
```

Although the line captures a part of the behaviour, it is quite clear that the trend of the time series is not linear as can be observed from the diagnostic plot below:

```
check(fit_jj1, simple = TRUE)
```

```
## Warning in check(fit_jj1, simple = TRUE): If 'lm' model is considered, only
## the full diagnostic plots can be provided, not the simple version.
```

It could therefore be more appropriate to define another function of time to describe it and, consequently, we add a quadratic term of time to obtain the following fit. Therefore, the model considered in (??) becomes:

$$X_t = \alpha + \beta_1 t + \beta_2 t^2 + \varepsilon_t, \quad (2.2)$$

The results of this regression are presented on the graphs below:

```
check(fit_jj2, simple = TRUE)
```

```
## Warning in check(fit_jj2, simple = TRUE): If 'lm' model is considered, only
## the full diagnostic plots can be provided, not the simple version.
```

We can see now that the quadratic function of time allows to better fit the observed time series and closely follow the observations. However, there still appears to be a pattern in the data that isn't captured by this quadratic model. This pattern appears to be repeated over time: peaks and valleys that seem to occur at regular intervals along the time series. This behaviour is known as *seasonality* which, in this case, can be explained by the effect of a specific quarter on the behaviour of the earnings. Indeed, it is reasonable to assume that the seasons have impacts on different variables measured over time (e.g. temperatures, earnings linked to sales that vary with seasons, etc.). Let us therefore take the quarters as an explanatory variable and add it to the model considered in (??), which becomes:

$$X_t = \alpha + \beta_1 t + \beta_2 t^2 + \sum_{i=1}^4 \gamma_i I_{t \in \mathcal{A}_i} + \varepsilon_t, \quad (2.3)$$

where

$$I_{t \in \mathcal{A}} \equiv \begin{cases} 1 & \text{if } t \in \mathcal{A} \\ 0 & \text{if } t \notin \mathcal{A} \end{cases},$$

and where

$$\mathcal{A}_i \equiv \{x \in \mathbb{N} | x = i \bmod 4\}.$$

The results are presented below:

```
check(fit_jj3, simple = TRUE)
```

```
## Warning in if (class(model) == "fitsimts") {: the condition has length > 1
## and only the first element will be used

## Warning in check(fit_jj3, simple = TRUE): If 'lm' model is considered, only
## the full diagnostic plots can be provided, not the simple version.
```

This final fit appears to well describe the behaviour of the earnings although there still appears to be a problem of heteroskedasticity (i.e. change in variance) and random seasonality (both of which will be treated further on in this text). Hence, *trend* and *seasonality* are the main features that characterize the deterministic component of a time series. However, as discussed earlier, these deterministic components often don't explain

all of the observed time series since there is often a random component characterizing data measured over time. Not considering the latter component can have considerable impacts on the inference procedures (as seen earlier) and it is therefore important to adequately analyse them (see next section).

2.1.2 The Random Component (Noise)

From this section onwards we will refer to *time series as being solely the random noise component*. Keeping this in mind, a *time series* is a particular kind of *stochastic process* which, generally speaking, is a collection of random variables indexed by a set of numbers. Not surprisingly, the index of reference for a time series is given by *time* and, consequently, a time series is a collection of random variables indexed (or “measured”) over time such as, for example, the daily price of a financial asset or the monthly average temperature in a given location. In terms of notation, a time series is often represented as

$$(X_1, X_2, \dots, X_T) \quad \text{or} \quad (X_t)_{t=1, \dots, T}.$$

The time index t is contained within either the set of reals, \mathbb{R} , or integers, \mathbb{Z} . When $t \in \mathbb{R}$, the time series becomes a *continuous-time* stochastic process such as a Brownian motion, a model used to represent the random movement of particles within a suspended liquid or gas. However, within this book, we will limit ourselves to the cases where $t \in \mathbb{Z}$, better known as *discrete-time* processes. Discrete-time processes are measured sequentially at fixed and equally spaced intervals in time. This implies that we will uphold two general assumptions for the time series considered in this book:

1. t is not random, e.g. the time at which each observation is measured is known, and
2. the time between two consecutive observations is constant.

This book will also focus on certain representations of time series based on parametric probabilistic models. For example, one of the fundamental probability models used in time series analysis is called the *white noise* model and is defined as

$$X_t \stackrel{iid}{\sim} N(0, \sigma^2).$$

This statement simply means that (X_t) is normally distributed and independent over time. Ideally, this is the type of process that we would want to observe once we have performed a statistical modelling procedure. However, despite it appearing to be an excessively simple model to be considered for time series, it is actually a crucial component to construct a wide range of more complex time series models (see Chapter ??). Indeed, unlike the white noise process, time series are typically *not* independent over time. For example, if we suppose that the temperature in State College is unusually low on a given day, then it is reasonable to assume that the temperature the day after will also be low.

With this in mind, let us now give a quick overview of the information that can be retrieved on a time series from a simple descriptive representation.

2.2 Exploratory Data Analysis for Time Series

When dealing with relatively small time series (e.g. a few thousands or less), it is often useful to look at a graph of the original data. A graph can be an informative tool for “detecting” some features of a time series such as trends and the presence of outliers. This is indeed what was done in the previous paragraphs when analysing the global temperature data or the Johnson & Johnson data.

To go more in depth with respect to the previous paragraphs, a trend is typically assumed to be present in a time series when the data exhibit some form of long term increase or decrease or combination of increases or decreases. Such trends could be linear or non-linear and represent an important part of the “signal” of a model (as seen for the Johnson & Johnson time series). Here are a few examples of non-linear trends:

1. **Seasonal trends** (periodic): These are the cyclical patterns which repeat after a fixed/regular time period. This could be due to business cycles (e.g. bust/recession, recovery).
2. **Non-seasonal trends** (periodic): These patterns cannot be associated to seasonal variation and can for example be due to an external variable such as, for example, the impact of economic indicators on stock returns. Note that such trends are often hard to detect based on a graphical analysis of the data.
3. **“Other” trends**: These trends have typically no regular patterns and are over a segment of time, known as a “window”, that change the statistical properties of a time series. A common example of such trends is given by the vibrations observed before, during and after an earthquake.

Moreover, when observing “raw” time series data it is also interesting to evaluate if some of the following phenomena occur:

1. **Change in Mean**: Does the mean of the process shift over time?
2. **Change in Variance**: Does the variance of the process evolve with time?
3. **Change in State**: Does the time series appear to change between “states” having distinct statistical properties?
4. **Outliers**: Does the time series contain some “extreme” observations? (Note that this is typically difficult to assess visually.)

Example 2.1. In the figure below, we present an example of displacement recorded during an earthquake as well as an explosion.

```
data(EQ5, package = "astsa")
data(EXP6, package = "astsa")

# Construct gts object
eq5 <- gts(EQ5, start = 0, freq = 1, unit_ts = "p/s", name_ts = "Earthquake Arrival Phases", data_name = "EQ5")
exp6 <- gts(EXP6, start = 0, freq = 1, unit_ts = "p/s", name_ts = "Explosion Arrival Phases", data_name = "EXP6")

# Plot time series
plot(eq5)

plot(exp6)
```

From the graph, it can be observed that the statistical properties of the time series appear to change over time. For instance, the variance of the time series shifts at around $t = 1150$ for both series. The shift in variance also opens “windows” where there appear to be distinct states. In the case of the explosion data, this is particularly relevant around $t = 50, \dots, 250$ and then again from $t = 1200, \dots, 1500$. Even within these windows, there are “spikes” that could be considered as outliers most notably around $t = 1200$ in the explosion series.

Extreme observations or outliers are commonly observed in real time series data, this is illustrated in the following example.

Example 2.2. We consider here a data set coming from the domain of hydrology. The data concerns monthly precipitation (in mm) over a certain period of time (1907 to 1972) and is interesting for scientists in order to study water cycles. The data are presented in the graph below:

```
# Load hydro dataset
data("hydro")

# Simulate based on data
hydro = gts(as.vector(hydro), start = 1907, freq = 12, unit_ts = "in."),
```



```

name_ts = "Precipitation", data_name = "Hydrology data")

# Plot hydro
plot(hydro)

```

We can see how most observations lie below 2mm but there appear to be different observations that go beyond this and appear to be larger than the others. These could be possible outliers that can greatly affect the estimation procedure if not taken adequately into account.

Next, we consider an example coming from high-frequency finance. The figure below presents the returns or price innovations (i.e. the changes in price from one observation to the next) for Starbucks's stock on July 1, 2011 for about 150 seconds (left panel) and about 400 minutes (right panel).

```

# Load "high-frequency" Starbucks returns for July 01 2011
data(sbx.xts, package = "highfrequency")

# Plot returns
par(mfrow = c(1,2))

plot(gts(sbx.xts[1:89]),
     main = "Starbucks: 150 Seconds",
     ylab = "Returns")

plot(gts(sbx.xts),
     main = "Starbucks: 400 Minutes",
     ylab = "Returns")

```

It can be observed on the left panel that observations are not equally spaced. Indeed, in high-frequency data the intervals between two points are typically not constant and are, even worse, random variables. This implies that the time when a new observation will be available is in general unknown. On the right panel, one can observe that the variability of the data seems to change during the course of the trading day. Such a phenomenon is well known in the finance community since a lot of variation occurs at the start (and the end) of the day while the middle of the day is associated with small changes. Moreover, clear extreme observations can also be noted in this graph at around 11:00.

Example 2.3. Finally, let us consider the limitations of a direct graphical representation of a time series when the sample size is large. Indeed, due to visual limitations, a direct plotting of the data will probably result in an uninformative aggregation of points between which it is unable to distinguish anything. This is illustrated in the following example.

We consider here the data coming from the calibration procedure of an Inertial Measurement Unit (IMU) which, in general terms, is used to enhance navigation precision or reconstruct three dimensional movements:

These sensors are used in a very wide range of applications such as robotics, virtual reality , vehicle stability control, human and animal motion capture and so forth:

The signals coming from these instruments are measured at high frequencies over a long time and are often characterized by linear trends and numerous underlying stochastic processes.

The code below retrieves some data from an IMU and plots it directly:

To access the IMU time series represented below you must install the `imudata` package which can be found at [this link](#).

```

# Load IMU data
data(imu6, package = "imudata")

# Construct gst object
Xt = gts(imu6[,1], data_name = "Gyroscope data", unit_time = "hour",
        freq = 100*60*60, name_ts = "Angular rate",
        unit_ts = bquote(rad^2/s^2))

# Plot time series
plot(Xt)

```

Although a linear trend and other processes are present in this signal (time series), it is practically impossible to understand or guess anything from the plot. For this reason, other types of representations are available to understand the behaviour of a time series and will be discussed in the next chapter. Having discussed these representations (and the relative issues with these representations) let us present the basic parametric models that are used to build even more complex models to describe and predict the behaviour of a time series.

2.3 Dependence in Time Series

In this section we briefly discuss the concept of dependence (within time series). As mentioned earlier, it is straightforward to assume that observations measured through time are dependent on each other (in that observations at time t have some form of impact on observations at time $t + 1$ or beyond). Due to this characteristic, one of the main interests in time series is prediction where, if $(X_t)_{t=1,\dots,T}$ is an identically distributed but not independent sequence, we often want to know the value of X_{T+h} for $h > 0$ (i.e. an estimator of $\mathbb{E}[X_{T+h}|X_T, \dots]$). In order to tackle this issue, we first need to understand the dependence between X_1, \dots, X_T and, even before this, we have to formally define what **independence** is.

Definition 2.1 (Independence of Events). Two events A and B are independent if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B),$$

with $\mathbb{P}(A)$ denoting the probability of event A occurring and $\mathbb{P}(A \cap B)$ denoting the joint probability (i.e. the probability that events A and B occur jointly). In general, A_1, \dots, A_n are independent if

$$\mathbb{P}(A_1 \dots A_n) = \mathbb{P}(A_1) \dots \mathbb{P}(A_n) \quad \forall A_i \in S, \quad i = 1, \dots, n$$

where S is the sample space.

Definition 2.2 (Independence of Random Variables). Two random variables X and Y with Cumulative Distribution Functions (CDF) $F_X(x)$ and $F_Y(y)$, respectively, are independent if and only if their joint CDF $F_{X,Y}(x, y)$ is such that

$$F_{X,Y}(x, y) = F_X(x)F_Y(y).$$

In general, random variables X_1, \dots, X_n with CDF $F_{X_1}(x_1), \dots, F_{X_n}(x_n)$ are respectively independent if and only if their joint CDF $F_{X_1, \dots, X_n}(x_1, \dots, x_n)$ is such that

$$F_{X_1, \dots, X_n}(x_1, \dots, x_n) = F_{X_1}(x_1) \dots F_{X_n}(x_n).$$

Definition 2.3 (iid sequence). The sequence X_1, X_2, \dots, X_T is said to be independent and identically distributed (i.e. iid) if and only if

$$\mathbb{P}(X_i < x) = \mathbb{P}(X_j < x) \quad \forall x \in \mathbb{R}, \forall i, j \in \{1, \dots, T\},$$

and

$$\mathbb{P}(X_1 < x_1, X_2 < x_2, \dots, X_T < x_T) = \mathbb{P}(X_1 < x_1) \dots \mathbb{P}(X_T < x_T) \quad \forall T \geq 2, x_1, \dots, x_T \in \mathbb{R}.$$

The basic idea behind the above definitions of independence is the fact that the probability of an event regarding variable X_i remains unaltered no matter what occurs for variable X_j (for $i \neq j$). However, for time series, this is often not the case and X_t often has some impact on X_{t+h} for some h (not too large). In order to explain (and predict) the impact of an observation on future observations, a series of models have been adopted through the years thereby providing a comprehensive framework to explain dependence through time. The following paragraphs introduce some of these basic models.

2.4 Basic Time Series Models

In this section, we introduce some simple time series models that constitute the building blocks for the more complex and flexible classes of time series commonly used in practice. Before doing so it is useful to define Ω_t as all the information available up to time $t - 1$, i.e.

$$\Omega_t \equiv (X_{t-1}, X_{t-2}, \dots, X_0).$$

As we will see further on, this compact notation is quite useful.

2.4.1 White Noise

As we saw earlier, the white noise model is the building block for most time series models and, to better specify the notation used throughout this book, this model is defined as

$$W_t \stackrel{iid}{\sim} N(0, \sigma_w^2).$$

This definition implies that:

1. $\mathbb{E}[W_t | \Omega_t] = 0$ for all t ,
2. $\text{cov}(W_t, W_{t-h}) = \mathbf{1}_{h=0} \sigma^2$ for all t, h .

More specifically, $h \in \mathbb{N}^+$ is the time difference between lagged variables. Therefore, in this process there is an absence of temporal (or serial) correlation and it is homoskedastic (i.e. it has a constant variance). Going into further details, white noise can be categorized into two sorts of processes: *weak* and *strong*. The process (W_t) is a *weak* white noise if

1. $\mathbb{E}[W_t] = 0$ for all t ,
2. $\text{var}(W_t) = \sigma_w^2$ for all t ,
3. $\text{cov}(W_t, W_{t-h}) = 0$ for all t and for all $h \neq 0$.

Note that this definition does not imply that W_t and W_{t-h} are independent (for $h \neq 0$) but simply uncorrelated. However, the notion of independence is used to define a *strong* white noise as

1. $\mathbb{E}[W_t] = 0$ and $\text{var}(W_t) = \sigma^2 < \infty$, for all t ,
2. $F(W_t) = F(W_{t-h})$ for all t, h (where $F(W_t)$ denotes the marginal distribution of W_t),
3. W_t and W_{t-h} are independent for all t and for all $h \neq 0$.

It is clear from these definitions that if a process is a strong white noise it is also a weak white noise. However, the converse is not true as shown in the following example:

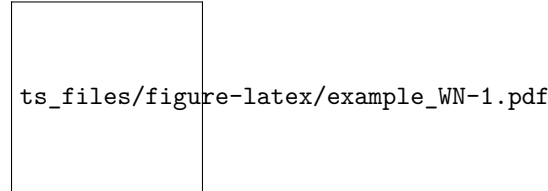
Example 2.4. Let $Y_t \sim F_{t+2}$, where F_{t+2} denotes a Student distribution with $t + 2$ degrees of freedom. Assuming the sequence (Y_1, \dots, Y_n) to be independent, we let $X_t = \sqrt{\frac{t}{t+2}} Y_t$. Then, the process (X_t) is obviously not a strong white noise as the distribution of X_t changes with t . However this process is a weak white noise since we have:

- $\mathbb{E}[X_t] = \sqrt{\frac{t}{t+2}} \mathbb{E}[Y_t] = 0$ for all t .

- $\text{var}(X_t) = \frac{t}{t+2} \text{var}(Y_t) = \frac{t}{t+2} \frac{t+2}{t} = 1$ for all t .
- $\text{cov}(X_t, X_{t+h}) = 0$ (by independence), for all t , and for all $h \neq 0$.

This distinction is therefore important and will be extremely relevant when discussing the concept of “stationarity” further on in this book. In general, the white noise model is assumed to be Gaussian in many practical cases and the code below presents an example of how to simulate a Gaussian white noise process.

```
n = 1000                                # process length
sigma2 = 1                              # process variance
Xt = gen_gts(n, WN(sigma2 = sigma2))
plot(Xt)
```



This model can be found in different applied settings and is often accompanied by some of the models presented in the following paragraphs.

2.4.2 Random Walk

The term *random walk* was first introduced by Karl Pearson in the early nineteen-hundreds and a wide range of random walk models have been defined over the years. For example, one of the simplest forms of a random walk process can be explained as follows: suppose that you are walking on campus and your next step can either be to your left, your right, forward or backward (each with equal probability). Two realizations of such processes are represented below:

```
set.seed(5)
RW2dimension(steps = 10^2)
```

```
RW2dimension(steps = 10^4)
```

Such processes inspired Karl Pearson’s famous quote that

“the most likely place to find a drunken walker is somewhere near his starting point.”

Empirical evidence of this phenomenon is not too hard to find on a Friday or Saturday night. This two-dimensional process may easily be extended to three dimensions and a simulated example of such a process is presented in the animation below:

In this text, we only consider one very specific form of random walk, namely the Gaussian random walk which can be defined as:

$$X_t = X_{t-1} + W_t,$$

where W_t is a Gaussian white noise process with initial condition $X_0 = c$ (typically $c = 0$.) This process can be expressed differently by *backsubstitution* as follows:

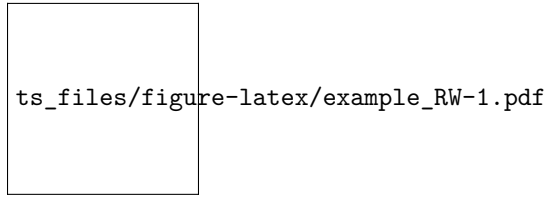
$$\begin{aligned}
X_t &= X_{t-1} + W_t \\
&= (X_{t-2} + W_{t-1}) + W_t \\
&\vdots \\
X_t &= \sum_{i=1}^t W_i + X_0 = \sum_{i=1}^t W_i + c
\end{aligned}$$

A random variable following a random walk can therefore be expressed as the cumulated sum of all the random variables that precede it. The code below presents an example of how to simulate a such process.

```

n = 1000                                # process length
gamma2 = 1                              # innovation variance
Xt = gen_gts(n, RW(gamma2 = gamma2))
plot(Xt)

```



The random walk model is often used to explain phenomena in many different areas one of which is finance where stock prices follow these kind of processes.

2.4.3 First-Order Autoregressive Model

A first-order autoregressive model or AR(1) is a generalization of both the white noise and the random walk processes which are both special cases of an AR(1). A (Gaussian) AR(1) process can be defined as

$$X_t = \phi X_{t-1} + W_t,$$

where W_t is a Gaussian white noise. Clearly, an AR(1) with $\phi = 0$ is a Gaussian white noise and when $\phi = 1$ the process becomes a random walk.

Remark 2.1. An AR(1) is in fact a linear combination of past realisations of a white noise W_t process. Indeed, we have

$$\begin{aligned}
X_t &= \phi_t X_{t-1} + W_t = \phi(\phi X_{t-2} + W_{t-1}) + W_t \\
&= \phi^2 X_{t-2} + \phi W_{t-1} + W_t = \phi^t X_0 + \sum_{i=0}^{t-1} \phi^i W_{t-i}.
\end{aligned}$$

Under the assumption of infinite past (i.e. $t \in \mathbb{Z}$) and $|\phi| < 1$, we obtain

$$X_t = \sum_{i=0}^{\infty} \phi^i W_{t-i},$$

since $\lim_{i \rightarrow \infty} \phi^i X_{t-i} = 0$.

From the conclusion of the above the remark, you may have noticed how we assume that the considered time series have zero expectation. The following remark justifies this assumption.

Remark 2.2. We generally assume that an AR(1), as well as other time series models, have zero mean. The reason for this assumption is only to simplify the notation but it is easy to consider, for example, an AR(1) process around an arbitrary mean μ , i.e.

$$(X_t - \mu) = \phi(X_{t-1} - \mu) + W_t,$$

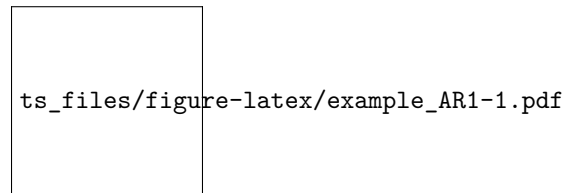
which is of course equivalent to

$$X_t = (1 - \phi)\mu + \phi X_{t-1} + W_t.$$

Thus, we will generally only work with zero mean processes since adding means is simple.

As for the previously presented models, we provide the code that gives an example of how an AR(1) can be simulated.

```
n = 1000                                # process length
phi = 0.5                               # phi parameter
sigma2 = 1                              # innovation variance
Xt = gen_gts(n, AR1(phi = phi, sigma2 = sigma2))
plot(Xt)
```



The AR(1) model is one of the most popular and commonly used models in many practical settings going from biology where it is used to explain the evolution of gene expressions to economics where it is used to model macroeconomic trends.

2.4.4 Moving Average Process of Order 1


As seen in the previous example, an AR(1) can be expressed as a linear combination of all past observations of the white noise process (W_t). In a similar manner we can (in some sense) describe the moving average process of order 1 or MA(1) as a “truncated” version of an AR(1). This model is defined as

$$X_t = \theta W_{t-1} + W_t, \tag{2.4}$$

where (again) W_t denotes a Gaussian white noise process. As we will see further on, as for the AR(1) model, this model can also be represented as a linear combination of past observations but it has different characteristics which can capture different types of dynamics in various practical cases.

An example on how to generate an MA(1) is given below:

```
n = 1000                                # process length
sigma2 = 1                              # innovation variance
theta = 0.5                             # theta parameter
Xt = gen_gts(n, MA1(theta = theta, sigma2 = sigma2))
plot(Xt)
```



The use of this model is widespread, especially combined with the AR(1) model, and can be found in fields such as engineering where it is often used for signal processing.

2.4.5 Linear Drift

A linear drift is a very simple deterministic time series model which can be expressed as

$$X_t = X_{t-1} + \omega,$$

where ω is a constant and with the initial condition $X_0 = c$, where c is an arbitrary constant (typically $c = 0$). This process can be expressed in a more familiar form as follows:

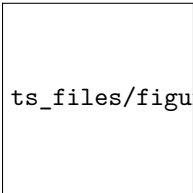
$$X_t = X_{t-1} + \omega = (X_{t-2} + \omega) + \omega = t\omega + c.$$

Therefore, a (linear) drift corresponds to a simple linear model with slope ω and intercept c .

Remark 2.3. You may argue that the definition of this model is not useful since it constitutes a simple linear model. However this model is often accompanied by other time series models (such as the ones presented earlier) and its estimation can be greatly improved when considered in conjunction with the other models.

Given its simple form, a linear drift can simply be generated using the code below:

```
n = 100                                # process length
omega = 0.5                            # slope parameter
Xt = gen_gts(n, DR(omega = omega))
plot(Xt)
```



This time series model is widely used in different areas of signal analysis where mechanical systems and measuring devices can be characterized by this type of behaviour.

2.5 Composite Stochastic Processes

In the previous paragraphs we defined and briefly discussed the basic time series models that can individually be used to describe and predict a wide range of phenomena in a variety of fields of application. However, their capability of capturing and explaining the different behaviours of phenomena through time increases considerably when they are combined to form so-called *composite models* (or composite processes). A composite (stochastic) process can be defined as the sum of underlying (or latent) time series models and in the rest of this book we will use the term *latent time series models* to refer to these kinds of models. A simple example of such a model is given by

$$\begin{aligned} Y_t &= Y_{t-1} + W_t + \delta \\ X_t &= Y_t + Z_t, \end{aligned}$$

where W_t and Z_t are two independent Gaussian white noise processes. This model is often used as a first basis to approximate the number of individuals in the context ecological population dynamics. For example, suppose we want to study the population of Chamois in the Swiss Alps. Let Y_t denote the “true” number of individuals in this population at time t . It is reasonable to assume that the number of individuals at time t (Y_t) is (approximately) the population at the previous time $t - 1$ (e.g. the previous year) plus a random variation and a drift. This random variation is due to the natural randomness in ecological population dynamics and reflects changes such as the number of predators, the abundance of food, or weather conditions. On the other hand, ecological *drift* is often of particular interest for ecologists as it can be used to determine the “long” term trends of the population (e.g. if the population is increasing, decreasing, or stable). Of course, Y_t (the number of individuals) is typically unknown and we observe a noisy version of it, denoted as X_t . This process corresponds to the true population plus a measurement error since some individuals may not be observed while others may have been counted several times. Interestingly, this process can clearly be expressed as a *latent time series model* (or composite stochastic process) as follows:

$$\begin{aligned} R_t &= R_{t-1} + W_t \\ S_t &= \delta t \\ X_t &= R_t + S_t + Z_t, \end{aligned}$$

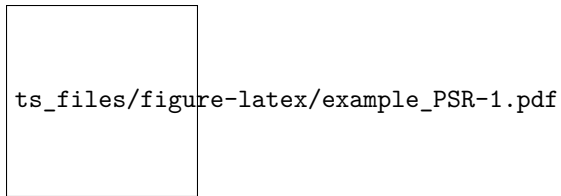
where R_t , S_t and Z_t denote, respectively, a random walk, a drift, and a white noise. The code below can be used to simulate such data:

```
n = 1000                                # process length
delta = 0.005                            # delta parameter (drift)
sigma2 = 10                             # variance parameter (white noise)
gamma2 = 0.1                             # innovation variance (random walk)
model = WN(sigma2 = sigma2) + RW(gamma2 = gamma2) + DR(omega = delta)
#Xt = gen_lts(n = n, model = model)
#plot(Xt)
```

In the above graph, the first three plots represent the latent (unobserved) processes (i.e. white noise, random walk, and drift) and the last one represents the sum of the three (i.e. (X_t)).

Let us consider a real example where these latent processes are useful to describe (and predict) the behavior of economic variables such as Personal Saving Rates (PSR). A process that is used for these settings is the “random-walk-plus-noise” model, meaning that the data can be explained by a random walk process in addition to which we observe some other process (e.g. a white noise model, an autoregressive model such as an AR(1), etc.). The PSR taken from the Federal Reserve of St. Louis from January 1, 1959, to May 1, 2015, is presented in the following plot:

```
# Load savingrt dataset
data("savingrt")
# Simulate based on data
savingrt = gts(as.vector(savingrt), start = 1959, freq = 12, unit_ts = "%",
              name_ts = "Saving Rates", data_name = "US Personal Saving Rates")
# Plot savingrt simulation
plot(savingrt)
```

It can be observed that the mean of this process seems to vary over time, suggesting that a random walk can indeed be considered as a possible model to explain this data. In addition, aside from some “spikes” and occasional sudden changes, the observations appear to gradually change from one time point to the other, suggesting that some other form of dependence between them could exist.

Chapter 3

Fundamental Properties of Time Series

“One of the first things taught in introductory statistics textbooks is that correlation is not causation. It is also one of the first things forgotten.” – Thomas Sowell

To make use of the R code within this chapter you will need to install (if not already done) and load the following libraries:

- quantmod;
- simts;
- astsa;
- robcor.

In this chapter we will discuss and formalize how knowledge about X_{t-1} (or more generally about all the information from the past, Ω_t) can provide us with some information about the properties of X_t . In particular, we will consider the correlation (or covariance) of X_t at different times such as $\text{corr}(X_t, X_{t+h})$. This “form” of correlation (covariance) is called the *autocorrelation* (*autocovariance*) and is a very useful tool in time series analysis. However, if we do not assume that a time series is characterized by a certain form of “stability”, it would be rather difficult to estimate $\text{corr}(X_t, X_{t+h})$ as this quantity would depend on both t and h leading to more parameters to estimate than observations available. Therefore, the concept of *stationarity* (which we will describe further on) is convenient in this context as it allows (among other things) to assume that

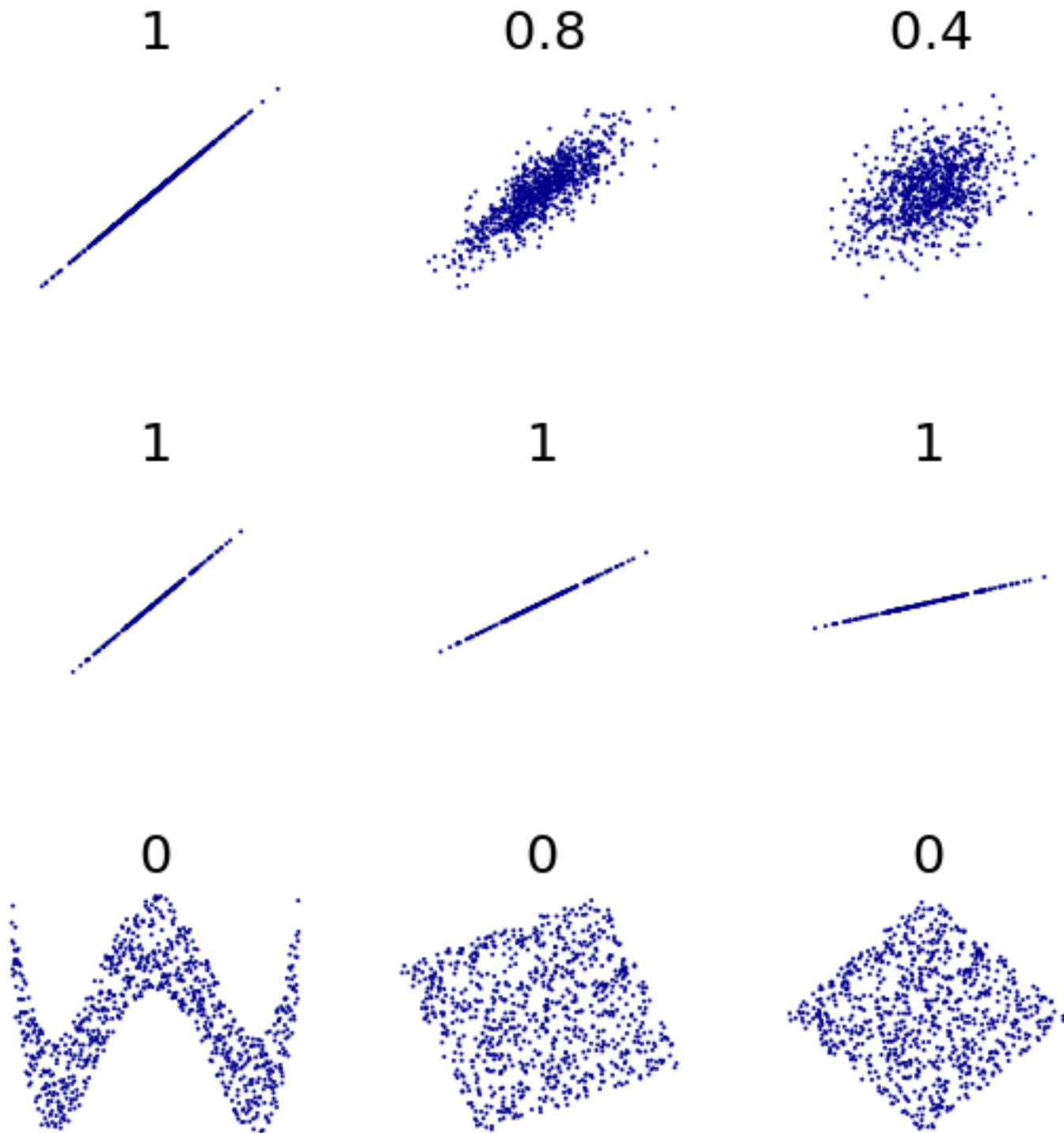
$$\text{corr}(X_t, X_{t+h}) = \text{corr}(X_{t+j}, X_{t+h+j}), \quad \text{for all } j,$$

implying that the autocorrelation (or autocovariance) is only a function of the lag between observations, rather than time itself. We will first discuss the concept of autocorrelation in time series, then we will discuss stationarity which will then allow us to adequately define and study estimators of the autocorrelation functions. Before moving on, it is helpful to remember that correlation (or autocorrelation) is only appropriate to measure a very specific kind of dependence, i.e. linear dependence. There are many other forms of dependence as illustrated in the bottom panels of the graph below, which all have a (true) zero correlation:

Several other metrics have been introduced in the literature to assess the degree of “dependence” of two random variables, however this goes beyond the material discussed in this chapter.

3.1 The Autocorrelation and Autocovariance Functions

We will introduce the autocorrelation function by first defining the **autocovariance function**.

Figure 3.1: Different forms of dependence and their Pearson's r values

Definition 3.1. The *autocovariance function* of a series (X_t) is defined as

$$\gamma_x(t, t+h) \equiv \text{cov}(X_t, X_{t+h}),$$

where the definition of covariance is given by:

$$\text{cov}(X_t, X_{t+h}) \equiv \mathbb{E}[X_t X_{t+h}] - \mathbb{E}[X_t] \mathbb{E}[X_{t+h}].$$

Similarly, the above expectations are defined as:

$$\begin{aligned} \mathbb{E}[X_t] &\equiv \int_{-\infty}^{\infty} x \cdot f_t(x) dx, \\ \mathbb{E}[X_t X_{t+h}] &\equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 \cdot f_{t,t+h}(x_1, x_2) dx_1 dx_2, \end{aligned}$$

where $f_t(x)$ and $f_{t,t+h}(x_1, x_2)$ denote, respectively, the density of X_t and the joint density of the pair (X_t, X_{t+h}) . Considering the notation used above, it should be clear that X_t is assumed to be a continuous random variable. Since we generally consider stochastic processes with constant zero mean, we often have

$$\gamma_x(t, t+h) = \mathbb{E}[X_t X_{t+h}].$$

In addition, in the context of this book we will normally drop the subscript referring to the time series (i.e. x in this case) if it is clear from the context which time series the autocovariance refers to. For example, we generally use $\gamma(t, t+h)$ instead of $\gamma_x(t, t+h)$. Moreover, the notation is even further simplified when the covariance of X_t and X_{t+h} is the same as that of X_{t+j} and X_{t+h+j} (for all j), i.e. the covariance depends only on the time between observations and not on the specific time t . This is consequence of an important property called *stationarity* that was mentioned earlier and will be discussed in the next section. In this case, we simply use the following notation:

$$\gamma(h) = \text{cov}(X_t, X_{t+h}).$$

This is the definition of autocovariance that will be used from this point onwards and therefore this notation will generally be used throughout the text thereby implying certain properties for the process (X_t) (i.e. stationarity). With this in mind, several remarks can be made on the autocovariance function:

1. The autocovariance function is *symmetric*. That is, $\gamma(h) = \gamma(-h)$ since $\text{cov}(X_t, X_{t+h}) = \text{cov}(X_{t+h}, X_t)$.
2. The autocovariance function “contains” the variance of the process as $\text{var}(X_t) = \gamma(0)$.
3. We have that $|\gamma(h)| \leq \gamma(0)$ for all h . The proof of this inequality is direct and follows from the Cauchy-Schwarz inequality, i.e.

$$\begin{aligned} (|\gamma(h)|)^2 &= \gamma(h)^2 = (\mathbb{E}[(X_t - \mathbb{E}[X_t])(X_{t+h} - \mathbb{E}[X_{t+h}])])^2 \\ &\leq \mathbb{E}[(X_t - \mathbb{E}[X_t])^2] \mathbb{E}[(X_{t+h} - \mathbb{E}[X_{t+h}])^2] = \gamma(0)^2. \end{aligned}$$

4. Just as any covariance, $\gamma(h)$ is “scale dependent” since $\gamma(h) \in \mathbb{R}$, or $-\infty \leq \gamma(h) \leq +\infty$. We therefore have:
 - if $|\gamma(h)|$ is “close” to zero, then X_t and X_{t+h} are “weakly” (linearly) dependent;
 - if $|\gamma(h)|$ is “far” from zero, then the two random variables present a “strong” (linear) dependence. However it is generally difficult to assess what “close” and “far” from zero means in this case.

5. $\gamma(h) = 0$ does not imply that X_t and X_{t+h} are independent but simply that they are uncorrelated. The independence is only implied by $\gamma(h) = 0$ in the jointly Gaussian case.

As hinted in the introduction, an important related statistic is the correlation of X_t with X_{t+h} or *autocorrelation*, which is defined as

$$\rho(h) = \text{corr}(X_t, X_{t+h}) = \frac{\text{cov}(X_t, X_{t+h})}{\sigma_{X_t} \sigma_{X_{t+h}}} = \frac{\gamma(h)}{\gamma(0)}.$$

Similarly to $\gamma(h)$, it is important to note that the above notation implies that the autocorrelation function is only a function of the lag h between observations. Thus, autocovariances and autocorrelations are one possible way to describe the joint distribution of a time series. Indeed, the correlation of X_t with X_{t+h} is an obvious measure of how *persistent* a time series is.

Remember that just as with any correlation:

1. $\rho(h)$ is “scale free” so it is much easier to interpret than $\gamma(h)$.
2. $|\rho(h)| \leq 1$ since $|\gamma(h)| \leq \gamma(0)$.
3. **Causation and correlation are two very different things!**

3.1.1 A Fundamental Representation

Autocovariances and autocorrelations also turn out to be very useful tools as they are one of the *fundamental representations* of time series. Indeed, if we consider a zero mean normally distributed process, it is clear that its joint distribution is fully characterized by the autocovariances $\mathbb{E}[X_t X_{t+h}]$ (since the joint probability density only depends of these covariances). Once we know the autocovariances we know *everything* there is to know about the process and therefore:

if two (Gaussian) processes have the same autocovariance function, then they are the same process.

3.1.2 Admissible Autocorrelation Functions

Since the autocorrelation function is one of the fundamental representations of time series, it implies that one might be able to define a stochastic process by picking a set of autocorrelation values (assuming for example that $\text{var}(X_t) = 1$). However, it turns out that not every collection of numbers, say $\{\rho_1, \rho_2, \dots\}$, can represent the autocorrelation of a process. Indeed, two conditions are required to ensure the validity of an autocorrelation sequence:

1. $\max_j |\rho_j| \leq 1$.
2. $\text{var} \left[\sum_{j=0}^{\infty} \alpha_j X_{t-j} \right] \geq 0$ for all $\{\alpha_0, \alpha_1, \dots\}$.

The first condition is obvious and simply reflects the fact that $|\rho(h)| \leq 1$ but the second is far more difficult to verify. To further our understanding of the latter we let $\alpha_j = 0$ for $j > 1$ and see that in this case the second condition implies that

$$\text{var} [\alpha_0 X_t + \alpha_1 X_{t-1}] = \gamma_0 \begin{bmatrix} \alpha_0 & \alpha_1 \end{bmatrix} \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \geq 0,$$

where γ_0 is a more compact notation for $\gamma(0)$. Thus, the matrix

$$\mathbf{A}_1 = \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix},$$

must be positive semi-definite. Taking the determinant we have

Figure 3.2: Admissible autocorrelation functions

$$\det(\mathbf{A}_1) = 1 - \rho_1^2,$$

implying that the condition $|\rho_1| \leq 1$ must be respected. Now, let $\alpha_j = 0$ for $j > 2$, then we must verify that:

$$\text{var}[\alpha_0 X_t + \alpha_1 X_{t-1} + \alpha_2 X_{t-2}] = \gamma_0 \begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \geq 0.$$

Again, this implies that the matrix

$$\mathbf{A}_2 = \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix},$$

must be positive semi-definite and it is easy to verify that

$$\det(\mathbf{A}_2) = (1 - \rho_2)(-2\rho_1^2 + \rho_2 + 1).$$

Thus, this implies that

$$\begin{aligned} -2\rho_1^2 + \rho_2 + 1 &\geq 0 \Rightarrow 1 \geq \rho_2 \geq 2\rho_1^2 - 1 \\ \Rightarrow 1 - \rho_1^2 &\geq \rho_2 - \rho_1^2 \geq -(1 - \rho_1^2) \\ \Rightarrow 1 &\geq \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2} \geq -1. \end{aligned}$$

Therefore, ρ_1 and ρ_2 must lie in a parabolic shaped region defined by the above inequalities as illustrated in Figure ??.

From our derivation, it is clear that the restrictions on the autocorrelation are very complicated, thereby justifying the need for other forms of fundamental representation which we will explore later in this text. Before moving on to the estimation of the autocorrelation and autocovariance functions, we must first discuss the stationarity of (X_t) , which will provide a convenient framework in which $\gamma(h)$ and $\rho(h)$ can be used (rather than $\gamma(t, t+h)$ for example) and (easily) estimated.

3.2 Stationarity

There are two kinds of stationarity that are commonly used. They are defined as follows:

Definition 3.2. A process (X_t) is *strongly stationary* or *strictly stationary* if the joint probability distribution of $(X_{t-h}, \dots, X_t, \dots, X_{t+h})$ is independent of t for all $t, h \in \mathbb{Z}$.

Definition 3.3. A process (X_t) is *weakly stationary*, *covariance stationary* or *second order stationary* if $\mathbb{E}[X_t]$ and $\mathbb{E}[X_t^2]$ are finite and $\mathbb{E}[X_t X_{t-h}]$ depends only on h and not on t for all $t, h \in \mathbb{Z}$.

These types of stationarity are *not equivalent* and the presence of one kind of stationarity does not imply the other. That is, a time series can be strongly stationary but not weakly stationary and vice versa. In some cases, a time series can be both strongly and weakly stationary and this occurs, for example, in the (jointly)

Gaussian case. Stationarity of (X_t) matters because *it provides the framework in which averaging dependent data makes sense*, thereby allowing us to easily obtain estimates for certain quantities such as autocorrelation.

Several remarks and comments can be made on these definitions:

- As mentioned earlier, strong stationarity *does not imply* weak stationarity. For example, an *iid* Cauchy process is strongly but not weakly stationary (why?).
- Weak stationarity *does not imply* strong stationarity. For example, consider the following weak white noise process:

$$X_t = \begin{cases} U_t & \text{if } t \in \{2k : k \in \mathbb{Z}\}, \\ V_t & \text{if } t \in \{2k+1 : k \in \mathbb{Z}\}, \end{cases}$$

where $U_t \stackrel{iid}{\sim} N(1, 1)$ and $V_t \stackrel{iid}{\sim} \mathcal{E}(1)$ is a weakly stationary process that is *not* strongly stationary.

- Strong stationarity combined with bounded values of $\mathbb{E}[X_t^2]$ *implies* weak stationarity.
- Weak stationarity combined with normally distributed processes *implies* strong stationarity.

Remark 3.1 (Existence of moments). It is important to note that, for a given value of r and a random variable X , the expected value $\mathbb{E}[X^r]$ may be infinite, or may not exist. If there exists an $r \in \mathbb{N}^+$ such that $\mathbb{E}[|X|^r]$ exists and is bounded, then we have that $\mathbb{E}[X^j] < \infty$ for all $j = 1, \dots, r$. This result is implied by Jensen's inequality. Indeed, the function $f(x) = x^k$ is convex for $x > 0$ and $k > 1$, so we have

$$\mathbb{E}[|X|^j]^{\frac{r}{j}} \leq \mathbb{E}\left[(|X|^j)^{\frac{r}{j}}\right] = \mathbb{E}[|X|^r] < \infty.$$

Therefore, we obtain $\mathbb{E}[X^j] \leq \mathbb{E}[|X|^j] < \infty$.

3.2.1 Assessing Weak Stationarity of Time Series Models

It is important to understand how to verify if a postulated model is (weakly) stationary. In order to do so, we must ensure that our model satisfies the following three properties:

1. $\mathbb{E}[X_t] = \mu_t = \mu < \infty$,
2. $\text{var}[X_t] = \sigma_t^2 = \sigma^2 < \infty$,
3. $\text{cov}(X_t, X_{t+h}) = \gamma(h)$ (i.e. the autocovariance only depends on h and not on t).

In the following examples, we evaluate the stationarity of the processes introduced in Section ??.

Example 3.1 (Gaussian White Noise). It is easy to verify that this process is stationary. Indeed, we have:

1. $\mathbb{E}[X_t] = 0$,
2. $\gamma(0) = \sigma^2 < \infty$,
3. $\gamma(h) = 0$ for $|h| > 0$.

Example 3.2 (Random Walk). To evaluate the stationarity of this process, we first derive its properties:

1. We begin by calculating the expectation of the process:

$$\mathbb{E}[X_t] = \mathbb{E}[X_{t-1} + W_t] = \mathbb{E}\left[\sum_{i=1}^t W_i + X_0\right] = \mathbb{E}\left[\sum_{i=1}^t W_i\right] + c = c.$$

Observe that the mean obtained is constant since it depends only on the value of the first term in the sequence.

2. Next, after finding the mean to be constant, we calculate the variance to check stationarity:

$$\begin{aligned}\text{var}(X_t) &= \text{var}\left(\sum_{i=1}^t W_i + X_0\right) = \text{var}\left(\sum_{i=1}^t W_i\right) + \underbrace{\text{var}(X_0)}_{=0} \\ &= \sum_{i=1}^t \text{var}(W_i) = t\sigma_w^2,\end{aligned}$$

where $\sigma_w^2 = \text{var}(W_t)$. Therefore, the variance depends on time t , contradicting our second property. Moreover, we have:

$$\lim_{t \rightarrow \infty} \text{var}(X_t) = \infty.$$

This process is therefore not weakly stationary.

3. Regarding the autocovariance of a random walk, we have:

$$\begin{aligned}\gamma(h) &= \text{cov}(X_t, X_{t+h}) = \text{cov}\left(\sum_{i=1}^t W_i, \sum_{j=1}^{t+h} W_j\right) \\ &= \min(t, t+h) \sigma_w^2 = (t + \min(0, h)) \sigma_w^2,\end{aligned}$$

which further illustrates the non-stationarity of this process.

Moreover, the autocorrelation of this process is given by

$$\rho(h) = \frac{t + \min(0, h)}{\sqrt{t}\sqrt{t+h}},$$

implying (for a fixed h) that

$$\lim_{t \rightarrow \infty} \rho(h) = 1.$$

Note that using $\gamma(h)$ and $\rho(h)$ in this context is actually an abuse of notation since both of these quantities are a function of h and t for a random walk process.

In the following simulated example, we illustrate the non-stationary feature of such a process:

```
# Number of simulated processes
B = 200

# Length of random walks
n = 1000

# Output matrix
out = matrix(NA, B, n)

# Set seed for reproducibility
set.seed(6182)

# Simulate Data
for (i in seq_len(B)){
  # Simulate random walk
  Xt = gen_gts(n, RW(gamma = 1))
}
```

Figure 3.3: Two hundred simulated random walks.

```

# Store process
out[i,] = Xt
}

# Plot random walks
plot(NA, xlim = c(1,n), ylim = range(out), xlab = "Time", ylab = " ")
grid()
color = sample(topo.colors(B, alpha = 0.5))
grid()
for (i in seq_len(B)){
  lines(out[i,], col = color[i])
}

# Add 95% confidence region
lines(1:n, 1.96*sqrt(1:n), col = 2, lwd = 2, lty = 2)
lines(1:n, -1.96*sqrt(1:n), col = 2, lwd = 2, lty = 2)

```

In the plot above, two hundred simulated random walks are plotted along with theoretical 95% confidence intervals (red-dashed lines). The relationship between time and variance can clearly be observed (i.e. the variance of the process increases with the time).

Example 3.3 (Moving Average of Order 1). Similarly to our previous examples, we attempt to verify the stationary properties for the MA(1) model defined in the previous chapter:

1.

$$\mathbb{E}[X_t] = \mathbb{E}[\theta_1 W_{t-1} + W_t] = \theta_1 \mathbb{E}[W_{t-1}] + \mathbb{E}[W_t] = 0.$$

2.

$$\text{var}(X_t) = \theta_1^2 \text{var}(W_{t-1}) + \text{var}(W_t) = (1 + \theta^2) \sigma_w^2.$$

3. Regarding the autocovariance, we have

$$\begin{aligned} \text{cov}(X_t, X_{t+h}) &= \mathbb{E}[(X_t - \mathbb{E}[X_t])(X_{t+h} - \mathbb{E}[X_{t+h}])] = \mathbb{E}[X_t X_{t+h}] \\ &= \mathbb{E}[(\theta W_{t-1} + W_t)(\theta W_{t+h-1} + W_{t+h})] \\ &= \mathbb{E}[\theta^2 W_{t-1} W_{t+h-1} + \theta W_t W_{t+h} + \theta W_{t-1} W_{t+h} + W_t W_{t+h}]. \end{aligned}$$

It is easy to see that $\mathbb{E}[W_t W_{t+h}] = \mathbf{1}_{\{h=0\}} \sigma_w^2$ and therefore, we obtain

$$\text{cov}(X_t, X_{t+h}) = (\theta^2 \mathbf{1}_{\{h=0\}} + \theta \mathbf{1}_{\{h=1\}} + \theta \mathbf{1}_{\{h=-1\}} + \mathbf{1}_{\{h=0\}}) \sigma_w^2$$

implying the following autocovariance function:

$$\gamma(h) = \begin{cases} (\theta^2 + 1) \sigma_w^2 & h = 0 \\ \theta \sigma_w^2 & |h| = 1 \\ 0 & |h| > 1 \end{cases}.$$

Therefore, an MA(1) process is weakly stationary since both the mean and variance are constant over time and its covariance function is only a function of the lag (h). Finally, we can easily obtain the autocorrelation for this process, which is given by

$$\rho(h) = \begin{cases} 1 & h = 0 \\ \frac{\theta \sigma_w^2}{(\theta^2 + 1) \sigma_w^2} = \frac{\theta}{\theta^2 + 1} & |h| = 1 \\ 0 & |h| > 1 \end{cases}.$$

Interestingly, we can note that $|\rho(1)| \leq 0.5$.

Example 3.4 (Autoregressive of Order 1). As another example, we shall verify the stationary properties for the AR(1) model defined in the previous chapter.

Using the *backsubstitution* technique, we can rearrange an AR(1) process so that it is written in a more compact form, i.e.

$$\begin{aligned} X_t &= \phi X_{t-1} + W_t = \phi [\phi X_{t-2} + W_{t-1}] + W_t = \phi^2 X_{t-2} + \phi W_{t-1} + W_t \\ &\vdots \\ &= \phi^k X_{t-k} + \sum_{j=0}^{k-1} \phi^j W_{t-j}. \end{aligned}$$

By taking the limit in k (which is perfectly valid as we assume $t \in \mathbb{Z}$) and assuming $|\phi| < 1$, we obtain

$$X_t = \lim_{k \rightarrow \infty} X_t = \sum_{j=0}^{\infty} \phi^j W_{t-j}$$

and therefore such a process can be interpreted as a linear combination of white noise (W_t) and corresponds (as we will observe later on) to an MA(∞). In addition, the requirement $|\phi| < 1$ turns out to be extremely useful as the above formula is related to a **geometric series** which would diverge if $\phi \geq 1$ (for example when $\phi = 1$ we have a random walk). Indeed, remember that an infinite (converging) geometric series is given by

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}, \quad \text{if } |r| < 1.$$

The origin of this requirement comes from needing to ensure that the characteristic polynomial solution for an AR(1) lies outside the unit circle thereby ensuring that the process is stationary. Indeed, if $\phi \geq 1$, the process would not converge.

With this setup, we demonstrate how crucial this property is by calculating each of the requirements of a stationary process.

1. First, we will check if the mean is stationary. In this case, we choose to use limits in order to derive the expectation

$$\begin{aligned} \mathbb{E}[X_t] &= \lim_{k \rightarrow \infty} \mathbb{E} \left[\phi^k X_{t-k} + \sum_{j=0}^{k-1} \phi^j W_{t-j} \right] \\ &= \lim_{k \rightarrow \infty} \underbrace{\phi^k \mathbb{E}[X_{t-k}]}_{=0} + \lim_{k \rightarrow \infty} \sum_{j=0}^{k-1} \phi^j \underbrace{\mathbb{E}[W_{t-j}]}_{=0} = 0. \end{aligned}$$

As expected, the mean is zero and, hence, the first criterion for weak stationarity is satisfied.

2. Next, we determine the variance of the process

$$\begin{aligned} \text{var}(X_t) &= \lim_{k \rightarrow \infty} \text{var} \left(\phi^k X_{t-k} + \sum_{j=0}^{k-1} \phi^j W_{t-j} \right) = \lim_{k \rightarrow \infty} \sum_{j=0}^{k-1} \phi^{2j} \text{var}(W_{t-j}) \\ &= \lim_{k \rightarrow \infty} \sum_{j=0}^{k-1} \sigma_W^2 \phi^{2j} = \underbrace{\frac{\sigma_W^2}{1-\phi^2}}_{\text{Geom. Series}}. \end{aligned}$$

Once again, the above result only holds because we are able to use the convergence of the geometric series as a result of $|\phi| < 1$.

Figure 3.4: Comparison of theoretical ACF of AR(1) with different parameter values

3. Finally, we consider the autocovariance of an AR(1). For $h > 0$, we have

$$\gamma(h) = \text{cov}(X_t, X_{t+h}) = \phi \text{cov}(X_t, X_{t+h-1}) = \phi \gamma(h-1).$$

Therefore, using the symmetry of autocovariance, we find that

$$\gamma(h) = \phi^{|h|} \gamma(0).$$

Both the mean and variance do not depend on time. In addition, the autocovariance function can be viewed as a function that only depends on the time lag h and, thus, the AR(1) process is weakly stationary if $|\phi| < 1$. Lastly, we can obtain the autocorrelation for this process. Indeed, for $h > 0$, we have

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \frac{\phi \gamma(h-1)}{\gamma(0)} = \phi \rho(h-1).$$

After simplifying, we obtain

$$\rho(h) = \phi^{|h|}.$$

Thus, the autocorrelation function for an AR(1) exhibits a *geometric decay*, meaning that as $|\phi|$ gets smaller the autocorrelation reaches zero at a faster rate (on the contrary, if $|\phi|$ is close to 1 then the decay rate is slower). This is well illustrated in the following plots.

3.3 Estimation of Moments (Stationary Processes)

In this section, we discuss how moments and related quantities of stationary processes can be estimated. Informally speaking, the use of “averages” is meaningful for such processes suggesting that classical moments estimators can be employed. Indeed, suppose that one is interested in estimating $\alpha \equiv \mathbb{E}[m(X_t)]$, where $m(\cdot)$ is a known function of X_t . If X_t is a strongly stationary process, we have

$$\alpha = \mathbb{E}[m(X_t)] = \int m(x) f(x) dx$$

where $f(x)$ denotes the density of X_t , $\forall t$. Replacing $f(x)$ by $f_n(x)$, the empirical density, we obtain the following estimator

$$\hat{\alpha} = \hat{\mathbb{E}}[m(X_t)] = \frac{1}{T} \sum_{t=1}^T m(x_t).$$

In the next subsection, we examine how this simple idea can be used to estimate the mean, autocovariance and autocorrelation functions. Moreover, we discuss some of the properties of these estimators.

3.3.1 Estimation of the Mean Function

If a time series is stationary, the mean function is constant and a possible estimator of this quantity is, as discussed above, given by

$$\bar{X} = \frac{1}{T} \sum_{t=1}^T X_t.$$

Naturally, the k -th moment, say $\beta_k \equiv \mathbb{E}[X_t^k]$ can be estimated by

$$\hat{\beta}_k = \frac{1}{T} \sum_{t=1}^T X_t^k, \quad k \in \{x \in \mathbb{N} : 0 < x < \infty\}.$$

The variance of such an estimator can be derived as follows:

$$\begin{aligned} \text{var}(\hat{\beta}_k) &= \text{var}\left(\frac{1}{T} \sum_{t=1}^T X_t^k\right) \\ &= \frac{1}{T^2} \text{var}\left(\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}_{1 \times T} \begin{bmatrix} X_1^k \\ \vdots \\ X_n^k \end{bmatrix}_{T \times 1}\right) \\ &= \frac{1}{n^2} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}_{1 \times T} \mathbf{\Sigma}(k) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{T \times 1}, \end{aligned} \tag{3.1}$$

where $\mathbf{\Sigma}(k) \in \mathbb{R}^{T \times T}$ and its i -th, j -th element is given by

$$(\mathbf{\Sigma}(k))_{i,j} = \text{cov}(X_i^k, X_j^k).$$

In the case $k = 1$, (??) can easily be further simplified. Indeed, we have

$$\begin{aligned} \text{var}(\bar{X}) &= \text{var}\left(\frac{1}{n} \sum_{t=1}^T X_t\right) \\ &= \frac{1}{T^2} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}_{1 \times T} \begin{bmatrix} \gamma(0) & \gamma(1) & \cdots & \gamma(T-1) \\ \gamma(1) & \gamma(0) & & \vdots \\ \vdots & & \ddots & \vdots \\ \gamma(T-1) & \cdots & \cdots & \gamma(0) \end{bmatrix}_{T \times T} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{T \times 1} \\ &= \frac{1}{n^2} (T\gamma(0) + 2(T-1)\gamma(1) + 2(T-2)\gamma(2) + \cdots + 2\gamma(T-1)) \\ &= \frac{1}{T} \sum_{h=-T}^T \left(1 - \frac{|h|}{T}\right) \gamma(h). \end{aligned}$$

Obviously, when X_t is a white noise process, the above formula reduces to the usual $\text{var}(\bar{X}) = \sigma_w^2/T$. In the following example, we consider the case of an AR(1) process and discuss how $\text{var}(\bar{X})$ can be obtained or estimated.

Example 3.5. For an AR(1), we have $\gamma(h) = \phi^h \sigma_w^2 (1 - \phi^2)^{-1}$. Therefore, we obtain (after some computations):

$$\text{var}(\bar{X}) = \frac{\sigma_w^2 (T - 2\phi - T\phi^2 + 2\phi^{T+1})}{T^2 (1 - \phi^2) (1 - \phi)^2}. \quad (3.2)$$

Unfortunately, deriving such an exact formula is often difficult when considering more complex models. However, asymptotic approximations are often employed to simplify the calculation. For example, in our case we have

$$\lim_{T \rightarrow \infty} T \text{var}(\bar{X}) = \frac{\sigma_w^2}{(1 - \phi)^2},$$

providing the following approximate formula:

$$\text{var}(\bar{X}) \approx \frac{\sigma_w^2}{T(1 - \phi)^2}.$$

Alternatively, simulation methods can also be employed. For example, a possible strategy would be parametric bootstrap.

Example 3.6. Parametric bootstrap can be implemented in the following manner:

1. Simulate a new sample under the postulated model, i.e. $X_t^* \sim F_{\theta}$ (*note*: if θ is unknown it can be replaced by $\hat{\theta}$, a suitable estimator).
2. Compute the statistics of interest on the simulated sample (X_t^*).
3. Repeat Steps 1 and 2 B times where B is sufficiently “large” (typically $100 \leq B \leq 10000$).
4. Compute the empirical variance of the statistics of interest based on the B independent replications.

In our example, we would consider (X_t^*) to be \bar{X}^* and seek to obtain:

$$\hat{\sigma}_B^2 = \frac{1}{B-1} \sum_{i=1}^B (\bar{X}_i^* - \bar{X}^*)^2, \quad \text{where} \quad \bar{X}^* = \frac{1}{B} \sum_{i=1}^B \bar{X}_i^*,$$

where \bar{X}_i^* denotes the value of the mean estimated on the i -th simulated sample.

The figure below generated by the following code compares these three methods for $T = 10$, $B = 1000$, $\sigma^2 = 1$ and a grid of values for ϕ going from -0.95 to 0.95 :

```
# Define sample size
n = 10

# Number of Monte-Carlo replications
B = 5000

# Define grid of values for phi
phi = seq(from = 0.95, to = -0.95, length.out = 30)

# Define result matrix
result = matrix(NA, B, length(phi))

# Start simulation
for (i in seq_along(phi)){
```

```

# Define model
model = AR1(phi = phi[i], sigma2 = 1)

# Monte-Carlo
for (j in seq_len(B)){
  # Simulate AR(1)
  Xt = gen_gts(n, model)

  # Estimate Xbar
  result[j,i] = mean(Xt)
}
}

# Estimate variance of Xbar
var.Xbar = apply(result,2,var)

# Compute theoretical variance
var.theo = (n - 2*phi - n*phi^2 + 2*phi^(n+1))/(n^2*(1-phi^2)*(1-phi)^2)

# Compute (approximate) variance
var.approx = 1/(n*(1-phi)^2)

# Compare variance estimations
plot(NA, xlim = c(-1,1), ylim = range(var.approx), log = "y",
     ylab = expression(paste("var(", bar(X), ")")),
     xlab= expression(phi), cex.lab = 1)
grid()
lines(phi,var.theo, col = "deepskyblue4")
lines(phi, var.Xbar, col = "firebrick3")
lines(phi,var.approx, col = "springgreen4")
legend("topleft",c("Theoretical variance","Bootstrap variance","Approximate variance"),
     col = c("deepskyblue4","firebrick3","springgreen4"), lty = 1,
     bty = "n",bg = "white", box.col = "white", cex = 1.2)

```

ts_files/figure-latex/estimXbar-1.pdf

It can be observed that the variance of \bar{X} typically increases with ϕ . As expected when $\phi = 0$, we have $\text{var}(\bar{X}) = 1/n$ and in this case the process is a white noise. Moreover, the bootstrap approach appears to well approximate the curve of (??), while the asymptotic form provides a reasonable approximation when ϕ lies between -0.5 and 0.5. Naturally, the quality of this approximation would be far better for a larger sample size (here we consider $T = 10$, which is a little “extreme”).

3.3.2 Sample Autocovariance and Autocorrelation Functions

A natural estimator of the *autocovariance function* is given by:

$$\hat{\gamma}(h) = \frac{1}{T} \sum_{t=1}^{T-h} (X_t - \bar{X})(X_{t+h} - \bar{X})$$

leading to the following “plug-in” estimator of the *autocorrelation function*:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}.$$

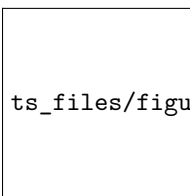
A graphical representation of the autocorrelation function is often the first step for any time series analysis (again assuming the process to be stationary). Consider the following simulated example:

```
# Set seed for reproducibility
set.seed(2241)

# Simulate 100 observation from a Gaussian white noise
Xt = gen_gts(100, WN(sigma2 = 1))

# Compute autocorrelation
acf_Xt = simts::auto_corr(Xt)

# Plot autocorrelation
plot(acf_Xt, show.ci = FALSE)
```



ts_files/figure-latex/basicACF-1.pdf

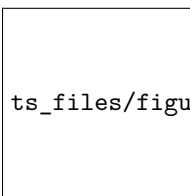
In this example, the true autocorrelation is equal to zero at any lag $h \neq 0$, but obviously the estimated autocorrelations are random variables and are not equal to their true values. It would therefore be useful to have some knowledge about the variability of the sample autocorrelations (under some conditions) to assess whether the data comes from a completely random series or presents some significant correlation at certain lags. The following result provides an asymptotic solution to this problem:

Theorem 3.1. *If X_t is a strong white noise with finite fourth moment, then for all $h \in \mathbb{Z} \setminus 0$, $\hat{\rho}(h)$ is approximately normally distributed with mean 0 and variance T^{-1} .*

The proof of this Theorem is given in Appendix ??.

Using this result, we now have an approximate method to assess whether peaks in the sample autocorrelation are significant by determining whether the observed peak lies outside the interval $\pm 2/\sqrt{T}$ (i.e. an approximate 95% confidence interval). Returning to our previous example and adding confidence bands to the previous graph, we obtain:

```
# Plot autocorrelation with confidence bands
plot(acf_Xt)
```



ts_files/figure-latex/basicACF2-1.pdf

It can now be observed that most peaks lie within the interval $\pm 2/\sqrt{T}$ suggesting that the true data generating process is uncorrelated.

Example 3.7. To illustrate how the autocorrelation function can be used to reveal some “features” of a time series, we download the level of the Standard & Poor’s 500 index, often abbreviated as the S&P 500.

This financial index is based on the market capitalization of 500 large companies having common stock listed on the New York Stock Exchange or the NASDAQ Stock Market. The graph below shows the index level and daily returns from 1990.

```
# Load package
library(quantmod)

# Download S&P index
getSymbols("^GSPC", from="1990-01-01", to = Sys.Date())

## [1] "GSPC"

# Extract index level and daily returns from the data
GSPC_index = gts(data = as.numeric(GSPC$GSPC.Close),
  start = 1990,
  freq = 252,
  Time = time(GSPC),
  unit_time = "year",
  name_ts = "Index Level",
  data_name = paste("S&P 500 (1990-01-01 - ", Sys.Date(), ")", sep = "")
)

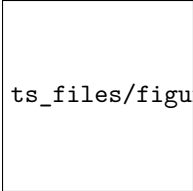
GSPC_returns = gts(data = as.numeric(ClC1(GSPC)),
  start = 1990,
  freq = 252,
  Time = time(GSPC),
  unit_time = "year",
  name_ts = "Daily Returns",
  data_name = paste("S&P 500 (1990-01-01 - ", Sys.Date(), ")", sep = "")
)

par(mfrow = c(1,2))
plot(GSPC_index)
plot(GSPC_returns)
```

ts_files/figure-latex/GSPC-1.pdf

From these graphs, it is clear that the returns are not identically distributed as the variance seems to change over time and clusters with either high or low volatility can be observed. These characteristics of financial time series are well known and further on in this book we will discuss how the variance of such processes can be approximated. Nevertheless, we compute the empirical autocorrelation function of the S&P 500 return to evaluate the degree of “linear” dependence between observations. The graph below presents the empirical autocorrelation.

```
sp500 = na.omit(GSPC_returns)
names(sp500) = paste("S&P 500 (1990-01-01 - ", Sys.Date(), ")", sep = "")
plot(simts::auto_corr(sp500))
```



ts_files/figure-latex/GSPCacf-1.pdf

As expected, the autocorrelation is small but it might be reasonable to believe that this sequence is not purely uncorrelated. Unfortunately, Theorem ?? is based on an asymptotic argument and since the confidence bands constructed are also asymptotic, there are no “exact” tools that can be used in this case. To study the validity of these results when T is “small” we can perform a simulation study. In the latter, we simulate processes from a Gaussian white noise and examine the empirical distribution of $\hat{\rho}(3)$ with different sample sizes (i.e. n is set to 5, 10, 30 and 300). Intuitively, the “quality” of the approximation provided by Theorem ?? should increase with the sample size T . The code below performs such a simulation and compares the empirical distribution of $\sqrt{T}\hat{\rho}(3)$ with a normal distribution with mean 0 and variance 1 (its asymptotic distribution), which is depicted using a red line.

```
# Number of Monte Carlo replications
B = 10000

# Define considered lag
h = 3

# Sample size considered
N = c(5, 10, 30, 300)

# Initialisation
result = matrix(NA,B,length(N))

# Set seed
set.seed(1)

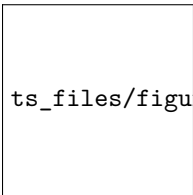
# Start Monte Carlo
for (i in seq_len(B)){
  for (j in seq_along(N)){
    # Simulate process
    Xt = rnorm(N[j])

    # Save autocorrelation at lag h
    result[i,j] = acf(Xt, plot = FALSE)$acf[h+1]
  }
}

# Plot results
par(mfrow = c(2,length(N)/2))
for (i in seq_along(N)){
  # Estimated empirical distribution
  hist(sqrt(N[i])*result[,i], col = "royalblue1",
        main = paste("Sample size n =",N[i]), probability = TRUE,
        xlim = c(-4,4), xlab = " ")

  # Asymptotic distribution
  xx = seq(from = -10, to = 10, length.out = 10^3)
  yy = dnorm(xx,0,1)
  lines(xx,yy, col = "red", lwd = 2)
```

}


 ts_files/figure-latex/simulationACF-1.pdf

As expected, it can clearly be observed that the asymptotic approximation is quite poor when $n = 5$ but as the sample size increases the approximation improves and is very close when, for example, $T = 300$. Therefore, this simulation would suggest that Theorem ?? provides a relatively “close” approximation of the distribution of $\hat{\rho}(h)$, especially when the sample size is large enough.

3.3.3 Robustness Issues

The data generating process delivers a theoretical autocorrelation (autocovariance) function that, as explained in the previous section, can then be estimated through the sample autocorrelation (autocovariance) functions. However, in practice, the sample is often issued from a data generating process that is “close” to the true one, meaning that the sample suffers from some form of small contamination. This contamination is typically represented by a small amount of extreme observations that are called “outliers” that come from a process that is different from the true data generating process.

The fact that the sample can suffer from outliers implies that the standard estimation of the autocorrelation (autocovariance) functions through the sample functions could be highly biased. The standard estimators presented in the previous section are therefore not “robust” and can behave badly when the sample suffers from contamination. To illustrate this limitation for a classical estimator, we consider the following two processes:

$$X_t = \phi X_{t-1} + W_t, \quad W_t \sim \mathcal{N}(0, \sigma_w^2),$$

$$Y_t = \begin{cases} X_t & \text{with probability } 1 - \epsilon \\ U_t & \text{with probability } \epsilon \end{cases}, \quad U_t \sim \mathcal{N}(0, \sigma_u^2),$$

where ϵ is “small” and $\sigma_u^2 \gg \sigma_w^2$. The process (Y_t) can be interpreted as a “contaminated” version of (X_t) and the figure below represents one realization of the process (Y_t) using the following setting: $T = 100$, $\sigma_u^2 = 10$, $\phi = 0.9$, $\sigma_w^2 = 1$ as well as $\alpha = 0.05$.

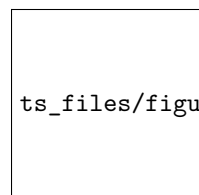
```
# Set seed for reproducibility
set.seed(2241)

# Define length of time series
N = 100

# Select observations from contamination distribution
epsilon = 0.05
index = sample(1:N, round(epsilon*N))
Ut = gen_gts(N, WN(sigma2 = 10))

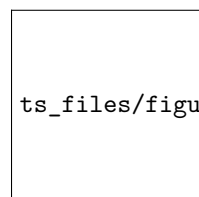
# Simulate observations from Xt and Yt
Xt = gen_gts(N, AR1(phi = 0.9, sigma2 = 1))
Yt = Xt
Yt[index] = Ut[index]
```

```
# Plot time series
par(mfrow = c(1,1))
plot(Yt, main = "Contaminated Time Series Yt")
```



The first question we can ask ourselves looking at this figure is: where are the outliers? You can probably spot a few but there are 5 outliers. If you're having difficulties detecting the outliers don't worry: it's a commonly known phenomenon in time series that detecting outliers is not always easy. Indeed, when looking at a time series for the first time it's not straightforward to understand if there's any form of contamination. Let's now compare (X_t) and (Y_t) in the following plots.

```
# Plot time series
par(mfrow = c(2,1))
plot(Xt, main = "Original Time Series Xt")
plot(Yt, main = "Contaminated Time Series Yt")
points(index-1, Yt[index], col = "red")
```



In this case it's more simple to detect the outliers (that are also highlighted with the red dots) since we can compare (Y_t) with the original uncontaminated time series (X_t) . Having highlighted how exploratory analysis can provide limited information on the presence of contamination in an observed time series, we now consider a simulated example to highlight how the performance of a “classical” autocorrelation estimator can deteriorate if the sample is contaminated (i.e. what is the impact of (Y_t) on the ACF estimator $\hat{\rho}(h)$). In this simulation, we will use the setting presented above and consider $B = 10^3$ bootstrap replications comparing the performance of the classical estimator when applied to an uncontaminated time series (X_t) and a contaminated time series (Y_t) .

```
B = 1000 # Number of simulations
N = 100 # Length of time series
epsilon = 0.05 # Amount of contamination
phi = 0.9 # AR(1) parameter value

# Store first 15 values of Empirical ACF (for Xt and Yt)
acf_xt = acf_yt = matrix(NA, B, 15)

for(i in 1:B) {

  # Set seed for reproducibility
  set.seed(i + 2241)

  # Select observations from contamination distribution
  index = sample(1:N, round(epsilon*N))
  Ut = gen_gts(N, WN(sigma2 = 10))
```

```

# Simulate observations from Xt and Yt
Xt = gen_gts(N, AR1(phi = 0.9, sigma2 = 1))
Yt = Xt
Yt[index] = Ut[index]

# Store ACF values
acf_xt[i, ] = as.numeric(auto_corr(Xt))[1:15]
acf_yt[i, ] = as.numeric(auto_corr(Yt))[1:15]
}

# Compute the Theoretical ACF of an AR(1) model (up to lag 15)
true_acf = phi^(1:15)

# Make boxplots of Empirical ACF for both settings and compare with true ACF
par(mfrow = c(1,2))
boxplot(acf_xt[, 3], col = "grey80", main = "ACF at lag 3 (uncontaminated)")
abline(h = true_acf[3], col = "red")
boxplot(acf_yt[, 3], col = "grey80", main = "ACF at lag 3 (contaminated)")
abline(h = true_acf[3], col = "red")

```

ts_files/figure-latex/simulationRobust-1.pdf

The boxplots represent the empirical distribution of the ACF estimator $\hat{\rho}(3)$: the left boxplot shows how the standard autocorrelation estimator is centered around the true value (red line) when the sample is not contaminated while the right boxplot shows how this estimate is considerably biased when the sample is contaminated. Indeed, it can be seen how the boxplot under contamination shows a lower value of autocorrelation indicating that it does not detect much dependence in the data although it should. The latter phenomenon is even more evident when analysing the empirical distributions at the larger lags. This is a known result in robustness, more specifically that outliers in the data can break the dependence structure and make it more difficult for the latter to be detected.

In order to limit this problem, different robust estimators exist for time series problems which are designed to reduce the impact of contamination on the estimation procedure. Among these estimators, there are a few that estimate the autocorrelation (autocovariance) functions in a robust manner. One of these estimators is provided in the `robacf()` function in the “robcor” package. The following simulated example shows how it limits the bias which is induced on the classic estimator $\hat{\rho}(h)$ from contamination. Unlike the previous simulation, we shall only consider data issued from the contaminated process (Y_t), and compare the performance of two estimators (i.e. classical and robust autocorrelation estimators):

```

B = 1000 # Number of simulations
N = 100 # Length of time series
epsilon = 0.05 # Amount of contamination
phi = 0.9 # AR(1) parameter value

# Store first 15 values of Empirical ACF (classic and robust)
cl_acf = rob_acf = matrix(NA, B, 15)

for(i in 1:B) {

```

```

# Set seed for reproducibility
set.seed(i + 2241)

# Select observations from contamination distribution
index = sample(1:N, round(epsilon*N))
Ut = gen_gts(N, WN(sigma2 = 10))

# Simulate observations from Yt
Yt = gen_gts(N, AR1(phi = 0.9, sigma2 = 1))
Yt[index] = Ut[index]

# Store Classic and Robust ACF values
cl_acf[i, ] = as.numeric(auto_corr(Yt))[1:15]
rob_acf[i, ] = as.numeric(auto_corr(Xt, robust = TRUE))[1:15]
}

# Compute the Theoretical ACF of an AR(1) model (up to lag 15)
true_acf = phi^(1:15)

# Make boxplots of both Classic and Robust Empirical ACF and compare with true ACF
par(mfrow = c(1,2))
boxplot(cl_acf[, 2], col = "grey80", main = "Classic ACF at lag 2")
abline(h = true_acf[2], col = "red")
boxplot(rob_acf[, 2], col = "grey80", main = "Robust ACF at lag 2")
abline(h = true_acf[2], col = "red")

```

ts_files/figure-latex/simulationRobust2-1.pdf

In this case we see the empirical distributions of the two estimators (classic and robust) for the ACF at time-lag $h = 2$. As you can see, the robust estimator remains close to the true value represented by the red line in the boxplots as opposed to the standard estimator. However, the price to pay in terms of bias reduction is a loss of efficiency of the robust estimator. Indeed it can often be observed that to reduce the bias induced by contamination in the sample, robust estimators pay a certain price in terms of efficiency.

To assess how much is “lost” by the robust estimator compared to the classical one in terms of efficiency, we consider one last simulation where we examine the performance of two estimators on data issued from the uncontaminated process, i.e. (X_t) . Therefore, the only difference between this simulation and the previous one is the value of ϵ set equal to 0 (the code shall thus be omitted). For this reason we simply show a plot that represents the ratio of the variances of the classic and robust ACF estimators respectively. If they happened to have the same efficiency, we would expect this ratio to be roughly equal to 1 over all the considered lags (we omit this ratio at lag 1 since it is numerically impossible to represent due to the infinitesimal size of the empirical variances).

ts_files/figure-latex/simulationRobust3-1.pdf

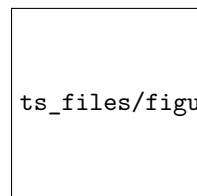
As can be seen on the plot, the black line representing the above described ratio decreases steadily as the lags increase and moves further away from the red dotted line representing equality of variances. Hence, since the variance of the classic ACF estimator is in the numerator of this ratio, we can conclude that the variance of the robust ACF estimator is always larger and increases over the lags. This can partly be explained by the fact that the classic ACF estimator $\hat{\rho}(h)$ makes use of all the observations while the robust estimator only uses part of the information coming from more “extreme” observations. Moreover, the number of observations available to estimate the greater lags is smaller and therefore the robust ACF estimator pays a larger price in terms of “loss” of information.

Let us finally investigate the importance of robust ACF estimation on some real data. We consider the data on monthly precipitation (`hydro`) presented in the previous chapter. This data is measured over 65 years (between 1907 and 1972) and is an example of data that is used to determine the behaviour of a water cycle. More specifically, precipitation is often considered the starting point for the analysis of a water cycle and, based on its behaviour, the rest of the water cycle is determined based on other variables. Therefore, a correct analysis of precipitation is extremely important to correctly define the behaviour of the water cycle passing through run-off and groundwater formation to evaporation and condensation. Given this, let us now take a look at the classic autocorrelation plot of this data.

```
# Load hydro dataset
data("hydro")

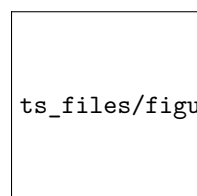
# Define the time series as a gts object
hydro = gts(as.vector(hydro), start = 1907, freq = 12, unit_ts = "mm", name_ts = "Precipitation", data_

# Plot the Empirical ACF
plot(auto_corr(hydro))
```



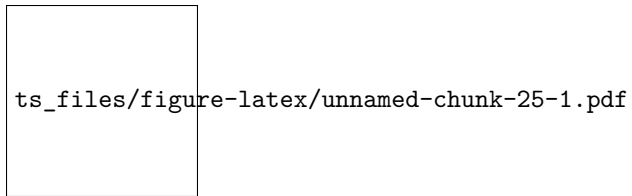
Based on this ACF plot, one would probably conclude that (counterintuitively) there does not appear to be any significant form of correlation between lagged observations in the data. From a hydrological point of view, one would therefore assume an uncorrelated model for precipitation (i.e. white noise) and, based on this, model the rest of the water cycle. However, let us take a look at the robust ACF plot.

```
# Plot the Robust ACF
plot(auto_corr(hydro, robust = TRUE))
```



If we analyse this output, the conclusion appears to be extremely different and, in some way, makes more sense from a hydrological point of view (i.e. the amount of precipitation between close months and over specific months is correlated). Indeed, we can see that there appears to be a seasonal correlation (“waves” in the ACF plot) and that close months appear to be correlated between them. To better highlight this difference (which can lead to different conclusions) let us finally compare the plots.

```
# Compare classic and robust ACF
compare_acf(hydro)
```



Therefore, based on the choice of analysis (i.e. classic or robust), the entire water cycle analysis would change and could deliver very different conclusions.

A general overview of the concepts and advantages of robustness can be found in Appendix ??.

Chapter 4

The Family of Autoregressive Moving Average Models

In this chapter we introduce a class of time series models that is considerably flexible and among the most commonly used to describe stationary time series. This class is represented by the Seasonal AutoRegressive Integrated Moving Average (SARIMA) models which, among others, combine and include the autoregressive and moving average models seen in the previous chapter. To introduce this class of models, we start by describing a sub-class called AutoRegressive Moving Average (ARMA) models which represent the backbone on which the SARIMA class is built. The importance of ARMA models resides in their flexibility as well as their capacity of describing (or closely approximating) almost all the features of a stationary time series. The autoregressive parts of these models describe how consecutive observations in time influence each other while the moving average parts capture some possible unobserved shocks thereby allowing to model different phenomena which can be observed in various fields going from biology to finance.

With this premise, the first part of this chapter introduces and explains the class of ARMA models in the following manner. First of all we will discuss the class of linear processes, which ARMA models belong to, and we will then proceed to a detailed description of autoregressive models in which we review their definition, explain their properties, introduce the main estimation methods for their parameters and highlight the diagnostic tools which can help understand if the estimated models appear to be appropriate or sufficient to well describe the observed time series. Once this is done, we will then use most of the results given for the autoregressive models to further describe and discuss moving average models, for which we underline the property of invertibility, and finally the ARMA models. Indeed, the properties and estimation methods for the latter class are directly inherited from the discussions on the autoregressive and moving average models.

The second part of this chapter introduces the general class of SARIMA models, passing through the class of ARIMA models. These models allow to apply the ARMA modeling framework also to time series that have particular non-stationary components to them such as, for example, linear and/or seasonal trends. Extending ARMA modeling to these cases allows SARIMA models to be an extremely flexible class of models that can be used to describe a wide range of phenomena.

4.1 Linear Processes

In order to discuss the classes of models mentioned above, we first present the class of linear processes which underlie many of the most common time series models.

Definition 4.1 (Linear Process). A time series, (X_t) , is defined to be a linear process if it can be expressed as a linear combination of white noise as follows:

$$X_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j W_{t-j}$$

where $W_t \sim WN(0, \sigma^2)$ and $\sum_{j=-\infty}^{\infty} |\psi_j| < \infty$.

Note, the latter assumption is required to ensure that the series has a limit. Furthermore, the set of coefficients

$$(\psi_j)_{j=-\infty, \dots, \infty}$$

can be viewed as a linear filter. These coefficients do not have to be all equal nor symmetric as later examples will show. Generally, the properties of a linear process related to mean and variance are given by:

$$\begin{aligned} \mu_X &= \mu \\ \gamma_X(h) &= \sigma_W^2 \sum_{j=-\infty}^{\infty} \psi_j \psi_{h+j} \end{aligned}$$

The latter is derived from

$$\begin{aligned} \gamma(h) &= \text{Cov}(x_t, x_{t+h}) \\ &= \text{Cov}\left(\mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}, \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t+h-j}\right) \\ &= \text{Cov}\left(\sum_{j=-\infty}^{\infty} \psi_j w_{t-j}, \sum_{j=-\infty}^{\infty} \psi_j w_{t+h-j}\right) \\ &= \sum_{j=-\infty}^{\infty} \psi_j \psi_{j+h} \text{Cov}(w_{t-j}, w_{t-j}) \\ &= \sigma_w^2 \sum_{j=-\infty}^{\infty} \psi_j \psi_{j+h} \end{aligned}$$

Within the above derivation, the key is to realize that $\text{Cov}(w_{t-j}, w_{t+h-j}) = 0$ if $t-j \neq t+h-j$.

Lastly, another convenient way to formalize the definition of a linear process is through the use of the **backshift operator** (or lag operator) which is itself defined as follows:

$$B X_t = X_{t-1}.$$

The properties of the backshift operator allow us to create composite functions of the type

$$B^2 X_t = B(B X_t) = B X_{t-1} = X_{t-2}$$

which allows to generalize as follows

$$B^k X_t = X_{t-k}.$$

Moreover, we can apply the inverse operator to it (i.e. $B^{-1} B = 1$) thereby allowing us to have, for example:

$$X_t = B^{-1} B X_t = B^{-1} X_{t-1}$$

Example 4.1 (d-order Differences). We can re-express $X_t - X_{t-1}$ as

$$\delta X_t = (1 - B)X_t$$

or a second order difference as

$$\delta^2 X_t = (1 - B)^2 X_t$$

thereby generalizing to a d-order difference as follows:

$$\delta^d X_t = (1 - B)^d X_t.$$

Having defined the backshift operator, we can now provide an alternative definition of a linear process as follows:

$$X_t = \mu + \psi(B) W_t$$

where $\psi(B)$ is a polynomial function in B whose coefficients are given by the linear filters (ψ_j) (we'll describe these polynomials further on).

Example 4.2 (Linear Process of White Noise). The white noise process (X_t) , defined in ??, can be expressed as a linear process as follows:

$$\psi_j = \begin{cases} 1, & \text{if } j = 0 \\ 0, & \text{if } |j| \geq 1 \end{cases}.$$

and $\mu = 0$.

Therefore, $X_t = W_t$, where $W_t \sim WN(0, \sigma_W^2)$

Example 4.3 (Linear Process of Moving Average Order 1). Similarly, consider (X_t) to be a MA(1) process, given by ??. The process can be expressed linearly through the following filters:

$$\psi_j = \begin{cases} 1, & \text{if } j = 0 \\ \theta, & \text{if } j = 1 \\ 0, & \text{if } j \geq 2 \end{cases}$$

and $\mu = 0$.

Thus, we have: $X_t = W_t + \theta W_{t-1}$

Example 4.4 (Linear Process and Symmetric Moving Average). Consider a symmetric moving average given by:

$$X_t = \frac{1}{2q+1} \sum_{j=-q}^q W_{t+j}$$

Thus, (X_t) is defined for $q+1 \leq t \leq n-q$. The above process would be a linear process since:

$$\psi_j = \begin{cases} \frac{1}{2q+1}, & \text{if } -q \leq j \leq q \\ 0, & \text{if } |j| > q \end{cases}.$$

and $\mu = 0$.

In practice, if $q = 1$, we would have:

$$X_t = \frac{1}{3} (W_{t-1} + W_t + W_{t+1})$$

Example 4.5 (Autoregressive Process of Order 1). If $\{X_t\}$ follows an AR(1) model defined in ??, the linear filters are a function of the time lag:

$$\psi_j = \begin{cases} \phi^j, & \text{if } j \geq 0 \\ 0, & \text{if } j < 0 \end{cases}.$$

and $\mu = 0$. We would require the condition that $|\phi| < 1$ in order to respect the condition on the filters (i.e. $\sum_{j=-\infty}^{\infty} |\psi_j| < \infty$).

4.2 Autoregressive Models - AR(p)

The class of autoregressive models is based on the idea that previous values in the time series are needed to explain current values in the series. For this class of models, we assume that the p previous observations are needed for this purpose and we therefore denote this class as AR(p). In the previous chapter, the model we introduced was an AR(1) in which only the immediately previous observation is needed to explain the following one and therefore represents a particular model which is part of the more general class of AR(p) models.

Definition 4.2 (Autoregressive Models of Order p). The AR(p) models can be formally represented as follows

$$(X_t) = \phi_1 X_{t-1} + \dots + \phi_p X_{t-p} + W_t,$$

where $\phi_i \neq 0$ (for $i = 1, \dots, p$) and W_t is a (Gaussian) white noise process with variance σ^2 .

As earlier in this book, we will assume that the expectation of the process (X_t) , as well as that of the following ones in this chapter, is zero. The reason for this simplification is that if $\mathbb{E}[X_t] = \mu$, we can define an AR process *around* μ as follows:

$$X_t - \mu = \sum_{i=1}^p \phi_i (X_{t-i} - \mu) + W_t,$$

which is equivalent to

$$X_t = \mu^* + \sum_{i=1}^p \phi_i X_{t-i} + W_t,$$

where $\mu^* = \mu(1 - \sum_{i=1}^p \phi_i)$. Therefore, to simplify the notation we will generally consider only zero mean processes, since adding means (as well as other deterministic trends) is easy.

A useful way of representing AR(p) processes is through the backshift operator introduced in the previous section and is as follows

$$\begin{aligned} X_t &= \phi_1 X_{t-1} + \dots + \phi_p X_{t-p} + W_t \\ &= \phi_1 B X_t + \dots + \phi_p B^p X_t + W_t, \\ &= (\phi_1 B + \dots + \phi_p B^p) X_t + W_t \end{aligned}$$

which finally yields

$$(1 - \phi_1 B - \dots - \phi_p B^p) X_t = W_t,$$

which, in abbreviated form, can be expressed as

$$\phi(B)X_t = W_t.$$

We will see that $\phi(B)$ is important to establish the stationarity of these processes and is called the *autoregressive* operator. Moreover, this quantity is closely related to another important property of AR(p) processes called *causality*. Before formally defining this new property we consider the following example which provides an intuitive illustration of its importance.

Example: Consider a classical AR(1) model with $|\phi| > 1$. Such a model could be expressed as

$$X_t = \phi^{-1}X_{t+1} - \phi^{-1}W_t = \phi^{-k}X_{t+k} - \sum_{i=1}^{k-1} \phi^{-i}W_{t+i}.$$

Since $|\phi| > 1$, we obtain

$$X_t = - \sum_{j=1}^{\infty} \phi^{-j} W_{t+j},$$

which is a linear process and therefore is stationary. Unfortunately, such a model is useless because we need the future to predict the future. These processes are called non-causal.

4.2.1 Properties of AR(p) models

In this section we will describe the main property of the AR(p) model which has already been mentioned in the previous paragraphs and therefore let us now introduce the property of causality in a more formal manner.

Definition: An AR(p) model is *causal* if the time series $(X_t)_{-\infty}^{\infty}$ can be written as a one-sided linear process:

$$X_t = \sum_{j=0}^{\infty} \psi_j W_{t-j} = \frac{1}{\phi(B)} W_t = \psi(B) W_t, \quad (4.1)$$

where $\phi(B) = \sum_{j=0}^{\infty} \phi_j B^j$, and $\sum_{j=0}^{\infty} |\phi_j| < \infty$ and setting $\phi_0 = 1$.

As discussed earlier this condition implies that only the past values of the time series can explain the future values of it and not viceversa. Moreover, given the expression of the linear filters given by

$$\frac{1}{\phi(B)}$$

it is obvious that a solution exists only when $\phi(B) = \sum_{j=0}^{\infty} \phi_j B^j \neq 0$ (thereby implying causality). A condition for this to be respected is for the roots of $\phi(B) = 0$ to lie outside the unit circle.

Example 4.6 (Transform an AR(2) into a Linear Process). Consider an AR(2) process

$$X_t = 1.3X_{t-1} - 0.4X_{t-2} + W_t,$$

which we would like to transform into a linear process. This can be done using the following approach:

- Step 1: The autoregressive operator of this model can be expressed as

$$\phi(B) = 1 - 1.3B + 0.4B^2 = (1 - 0.5B)(1 - 0.8B),$$

and has roots 2 and 1.25, both > 1 . Thus, we should be able to convert it into a linear process.

- Step 2: We know that if an AR(p) process has all its roots outside the unit circle, then we can write $X_t = \frac{1}{\phi(B)} W_t$. By applying the partial fractions trick, we can inverse the autoregressive operator $\phi(B)$ as follows:

$$\begin{aligned}\phi^{-1}(B) &= \frac{1}{(1 - 0.5B)(1 - 0.8B)} = \frac{c_1}{(1 - 0.5B)} + \frac{c_2}{(1 - 0.8B)} \\ &= \frac{c_2(1 - 0.5B) + c_1(1 - 0.8B)}{(1 - 0.5B)(1 - 0.8B)} = \frac{(c_1 + c_2) - (0.8c_1 + 0.5c_2)B}{(1 - 0.5B)(1 - 0.8B)}.\end{aligned}$$

To solve for c_1 and c_2 :

$$\begin{cases} c_1 + c_2 &= 1 \\ 0.8c_1 + 0.5c_2 &= 0 \end{cases} \rightarrow \begin{cases} c_1 &= -5/3 \\ c_2 &= 8/3. \end{cases}$$

So we obtain

$$\phi^{-1}(B) = \frac{-5}{3(1 - 0.5B)} + \frac{8}{3(1 - 0.8B)}.$$

- Step 3: Using the Geometric series, i.e. $a \sum_{j=0}^{\infty} r^j = \frac{a}{1-r}$ if $|r| < 1$, we have

$$\begin{cases} \frac{-5}{3(1 - 0.5B)} = -\frac{5}{3} \sum_{j=0}^{\infty} 0.5^j B^j, & \text{if } |B| < 2 \\ \frac{8}{3(1 - 0.8B)} = \frac{8}{3} \sum_{j=0}^{\infty} 0.8^j B^j, & \text{if } |B| < 1.25. \end{cases}$$

So we can express $\phi^{-1}(B)$ as

$$\phi^{-1}(B) = \sum_{j=0}^{\infty} \left[-\frac{5}{3}(0.5)^j + \frac{8}{3}(0.8)^j \right] B^j, \quad \text{if } |B| < 1.25.$$

- Step 4: Finally, we obtain

$$\begin{aligned}X_t &= \phi(B)^{-1} W_t = \sum_{j=0}^{\infty} \left[-\frac{5}{3}(0.5)^j + \frac{8}{3}(0.8)^j \right] B^j W_t \\ &= \sum_{j=0}^{\infty} \left[-\frac{5}{3}(0.5)^j + \frac{8}{3}(0.8)^j \right] W_{t-j},\end{aligned}$$

which verifies that the AR(2) is causal, and therefore is stationary.

Example 4.7 (Causal Conditions for an AR(2) Process). We already know that an AR(1) is causal with the simple condition $|\phi_1| < 1$. It seems natural to believe that an AR(2) should be causal (and therefore stationary) with the condition that $|\phi_i| < 1$, $i = 1, 2$. However, this is actually not the case as we illustrate below.

We can express an AR(2) process as

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + W_t = \phi_1 B X_t + \phi_2 B^2 X_t + W_t,$$

thereby delivering the following autoregressive operator:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 = \left(1 - \frac{B}{\lambda_1}\right) \left(1 - \frac{B}{\lambda_2}\right)$$

where λ_1 and λ_2 are the roots of $\phi(B)$ such that

$$\begin{aligned}\phi_1 &= \frac{1}{\lambda_1} + \frac{1}{\lambda_2}, \\ \phi_2 &= -\frac{1}{\lambda_1} \frac{1}{\lambda_2}.\end{aligned}$$

That is,

$$\lambda_1 = \frac{\phi_1 + \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2},$$

$$\lambda_2 = \frac{\phi_1 - \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2}.$$

In order to ensure the causality of the model, we need the roots of $\phi(B)$, i.e. λ_1 and λ_2 , to lie outside the unit circle.

$$\begin{cases} |\lambda_1| > 1 \\ |\lambda_2| > 1, \end{cases}$$

if and only if

$$\begin{cases} \phi_1 + \phi_2 < 1 \\ \phi_2 - \phi_1 < 1 \\ |\phi_2| < 1. \end{cases}$$

We can show the *if* part of the statement as follows:

$$\begin{aligned} \phi_1 + \phi_2 &= \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 \lambda_2} = \frac{1}{\lambda_1} \left(1 - \frac{1}{\lambda_2}\right) + \frac{1}{\lambda_2} < 1 - \frac{1}{\lambda_2} + \frac{1}{\lambda_2} = 1 \text{ since } 1 - \frac{1}{\lambda_2} > 0, \\ \phi_2 - \phi_1 &= -\frac{1}{\lambda_1 \lambda_2} - \frac{1}{\lambda_1} - \frac{1}{\lambda_2} = -\frac{1}{\lambda_1} \left(\frac{1}{\lambda_2} + 1\right) - \frac{1}{\lambda_2} < \frac{1}{\lambda_2} + 1 - \frac{1}{\lambda_2} = 1 \text{ since } \frac{1}{\lambda_2} + 1 > 0, \\ |\phi_2| &= \frac{1}{|\lambda_1| |\lambda_2|} < 1. \end{aligned}$$

We can also show the *only if* part of the statement as follows:

Since $\lambda_1 = \frac{\phi_1 + \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2}$ and $\phi_2 - 1 < \phi_1 < 1 - \phi_2$, we have

$$\lambda_1^2 = \frac{(\phi_1 + \sqrt{\phi_1^2 + 4\phi_2})^2}{4\phi_2^2} < \frac{\left((1 - \phi_2) + \sqrt{(1 - \phi_2)^2 + 4\phi_2}\right)^2}{4\phi_2^2} = \frac{4}{4\phi_2^2} \leq 1.$$

Since $\lambda_2 = \frac{\phi_1 - \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2}$ and $\phi_2 - 1 < \phi_1 < 1 - \phi_2$, we have

$$\lambda_2^2 = \frac{(\phi_1 - \sqrt{\phi_1^2 + 4\phi_2})^2}{4\phi_2^2} < \frac{\left((\phi_2 - 1) + \sqrt{(\phi_2 - 1)^2 + 4\phi_2}\right)^2}{4\phi_2^2} = \frac{4\phi_2^2}{4\phi_2^2} = 1.$$

Finally, the causal region of an AR(2) is demonstrated as

4.2.2 Estimation of AR(p) models

Given the above defined properties of AR(p) models, we will now discuss how these models can be estimated, more specifically how the $p + 1$ parameters can be obtained from an observed time series. Indeed, a reliable estimation of these models is necessary in order to interpret and describe different natural phenomena and/or forecast possible future values of the time series.

A first approach builds upon the earlier definition of AR(p) models being a linear process. Recall that

$$X_t = \sum_{j=1}^p \phi_j X_{t-j} \quad (4.2)$$

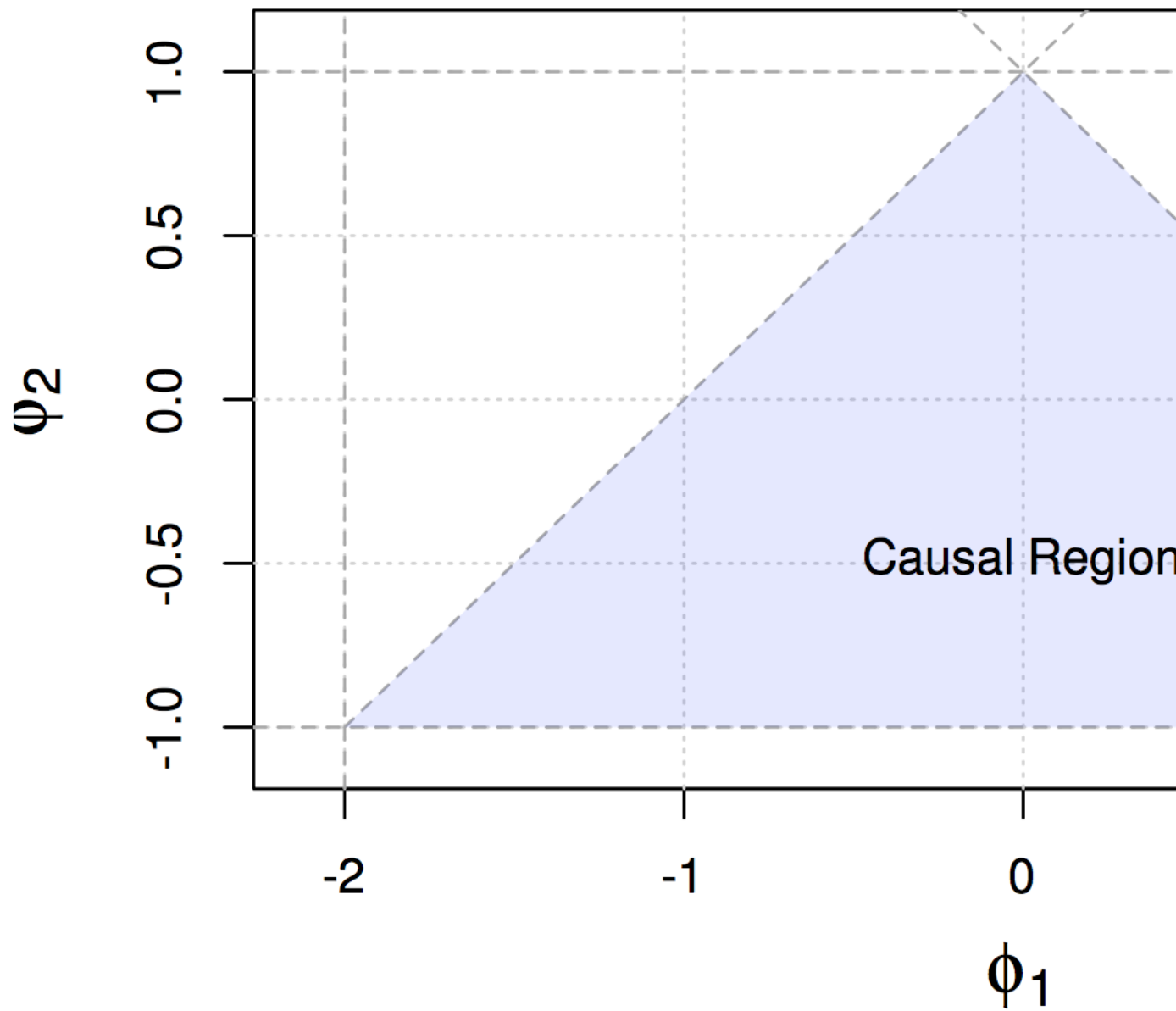


Figure 4.1: Causal Region for Parameters of an AR(2) Process

which delivers the following autocovariance function

$$\gamma(h) = \text{cov}(X_{t+h}, X_t) = \text{cov}\left(\sum_{j=1}^p \phi_j X_{t+h-j}, X_t\right) = \sum_{j=1}^p \phi_j \gamma(h-j), \quad h \geq 1. \quad (4.3)$$

Rearranging the above expressions we obtain the following general equations

$$\gamma(h) - \sum_{j=1}^p \phi_j \gamma(h-j) = 0, \quad h \geq 1 \quad (4.4)$$

and, recalling that $\gamma(h) = \gamma(-h)$,

$$\gamma(0) - \sum_{j=1}^p \phi_j \gamma(j) = \sigma_w^2. \quad (4.5)$$

We can now define the Yule-Walker equations.

Definition: The Yule-Walker equations are given by

$$\gamma(h) = \phi_1 \gamma(h-1) + \dots + \phi_p \gamma(h-p), \quad h = 1, \dots, p \quad (4.6)$$

and

$$\sigma_w^2 = \gamma(0) - \phi_1 \gamma(1) - \dots - \phi_p \gamma(p). \quad (4.7)$$

which in matrix notation can be defined as follows

$$\Gamma_p \phi = \gamma_p \text{ and } \sigma_w^2 = \gamma(0) - \phi' \gamma_p \quad (4.8)$$

where Γ_p is the $p \times p$ matrix containing the autocovariances $\gamma(k-j)$, where $j, k = 1, \dots, p$, while $\phi = (\phi_1, \dots, \phi_p)'$ and $\gamma_p = (\gamma(1), \dots, \gamma(p))'$ are $p \times 1$ vectors.

Considering the Yule-Walker equations, it is possible to use a method of moments approach and simply replace the theoretical quantities given in the previous definition with their empirical (estimated) counterparts that we saw in the previous chapter. This gives us the following Yule-Walker estimators

$$\hat{\phi} = \hat{\Gamma}_p^{-1} \hat{\gamma}_p \text{ and } \hat{\sigma}_w^2 = \hat{\gamma}(0) - \hat{\gamma}_p' \hat{\Gamma}_p^{-1} \hat{\gamma}_p. \quad (4.9)$$

These estimators have the following asymptotic properties.

Consistency and Asymptotic Normality of Yule-Walker estimators: The Yule-Walker estimators for a causal AR(p) model have the following asymptotic properties:

$$\sqrt{T}(\hat{\phi} - \phi) \xrightarrow{\mathcal{D}} \mathcal{N}(\mathbf{0}, \sigma_w^2 \Gamma_p^{-1}) \text{ and } \hat{\sigma}_w^2 \xrightarrow{\mathcal{P}} \sigma_w^2.$$

Therefore the Yule-Walker estimators have an asymptotically normal distribution and the estimator of the innovation variance is consistent. Moreover, these estimators are also optimal for AR(p) models, meaning that they are also efficient. However, there is also another method which allows to achieve this efficiency (also for general ARMA models that will be tackled further on) and this is the Maximum Likelihood Estimation (MLE) method. Considering an AR(1) model as an example, and assuming without loss of generality that its expectation is zero, we have the following representation of the AR(1) model

$$X_t = \phi X_{t-1} + W_t$$

where $|\phi| < 1$ and $W_t \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_w^2)$. Supposing we have observations $(x_t)_{t=1, \dots, T}$ issued from this model, then the likelihood function for this setting is given by

$$L(\phi, \sigma_w^2) = f(\phi, \sigma_w^2 | x_1, \dots, x_T)$$

which, for an AR(1) model, can be rewritten as follows

$$L(\phi, \sigma_w^2) = f(x_1)f(x_2|x_1) \cdots f(x_T|x_{T-1}).$$

If we define Ω_t^p as the information contained in the previous p observations (before time t), the above expression can be generalized for an AR(p) model as follows

$$L(\phi, \sigma_w^2) = f(x_1, \dots, x_p)f(x_{p+1}|\Omega_{p+1}^p) \cdots f(x_T|\Omega_T^p)$$

where $f(x_1, \dots, x_p)$ is the joint probability distribution of the first p observations. Going back to the AR(1) setting, based on our assumption on (W_t) we know that $x_t|x_{t-1} \sim \mathcal{N}(\phi x_{t-1}, \sigma_w^2)$ and therefore we have that

$$f(x_t|x_{t-1}) = f_w(x_t - \phi x_{t-1})$$

where $f_w(\cdot)$ is the distribution of W_t . This rearranges the likelihood function as follows

$$L(\phi, \sigma_w^2) = f(x_1) \prod_{t=2}^T f_w(x_t - \phi x_{t-1})$$

where $f(x_1)$ can be found through the causal representation

$$x_1 = \sum_{j=0}^{\infty} \phi^j w_{1-j}$$

which implies that x_1 follows a normal distribution with zero expectation and a variance given by $\frac{\sigma_w^2}{(1-\phi^2)}$. Based on this, the likelihood function of an AR(1) finally becomes

$$L(\phi, \sigma_w^2) = (2\pi\sigma_w^2)^{-\frac{T}{2}} (1-\phi^2)^{\frac{1}{2}} \exp\left(-\frac{S(\phi)}{2\sigma_w^2}\right)$$

with $S(\phi) = (1-\phi^2)x_1^2 + \sum_{t=2}^T (x_t - \phi x_{t-1})^2$. Once the derivative of the logarithm of the likelihood is taken, the minimization of the negative of this function is usually done numerically. However, if we condition on the initial values, the AR(p) models are linear and, for example, we can then define the conditional likelihood of an AR(1) as

$$L(\phi, \sigma_w^2|x_1) = (2\pi\sigma_w^2)^{-\frac{T-1}{2}} \exp\left(-\frac{S_c(\phi)}{2\sigma_w^2}\right)$$

where

$$S_c(\phi) = \sum_{t=2}^T (x_t - \phi x_{t-1})^2.$$

The latter is called the conditional sum of squares and ϕ can be estimated as a straightforward linear regression problem. Once an estimate $\hat{\phi}$ is obtained, this can be used to obtain the conditional maximum likelihood estimate of σ_w^2

$$\hat{\sigma}_w^2 = \frac{S_c(\hat{\phi})}{(T-1)}.$$

The estimation methods presented so far are standard for these kind of models. Nevertheless, if the data suffers from some form of contamination, these methods can become highly biased. For this reason, some robust estimators are available to limit this problematic if there are indeed outliers in the observed time series. One of these methods relies on the estimator proposed in Kunsch (1984) who underlines that the MLE score function of an AR(p) is given by

$$\kappa(\theta|x_j, \dots, x_{j+p}) = \frac{\partial}{\partial \theta} (x_{j+p} - \sum_{k=1}^p \phi_k x_{j+p-k})^2$$

where θ is the parameter vector containing, in the case of an AR(1) model, the two parameters ϕ and σ_w^2 (i.e. $\theta = [\phi \ \sigma_w^2]$). This delivers the estimating equation

$$\sum_{j=1}^{n-p} \kappa(\hat{\theta}|x_j, \dots, x_{j+p}) = 0.$$

The score function $\kappa(\cdot)$ is clearly not bounded, in the sense that if we arbitrarily move a value of (x_t) to infinity then the score function also goes to infinity thereby delivering a biased estimation procedure. To avoid that outlying observations bias the estimation excessively, a bounded score function can be used to deliver an M-estimator given by

$$\sum_{j=1}^{n-p} \psi(\hat{\theta}|x_j, \dots, x_{j+p}) = 0,$$

where $\psi(\cdot)$ is a function of bounded variation. When conditioning on the first p observations, this problem can be brought back to a linear regression problem which can be applied in a robust manner using the robust regression tools available in R such as `rlm` or `lmrob`. However, another available tool in R which does not require a strict specification of the distribution function (also for general ARMA models) is the `method = "rgmwm"` option within the `estimate()` function (in the `simts` package). This function makes use of a quantity called the wavelet variance (denoted as ν) which is estimated robustly and then used to retrieve the parameters θ of the time series model. The robust estimate is obtained by solving the following minimization problem

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} (\hat{\nu} - \nu(\theta))^T \Omega (\hat{\nu} - \nu(\theta)),$$

where $\hat{\nu}$ is the robustly estimated wavelet variance, $\nu(\theta)$ is the theoretical wavelet variance (implied by the model we want to estimate) and Ω is a positive definite weighting matrix. Below we show some simulation studies where we present the results of the above estimation procedures in absence and in presence of contamination in the data. As a reminder, so far we have mainly discussed three estimators for the parameters of AR(p) models (i.e. Yule-Walker, maximum likelihood, and RGMWM estimators).

Using the `simts` package, the first three estimators can be computed as follows:

```
mod = estimate(AR(p), Xt, method = select_method, demean = TRUE)
```

In the above sample code `Xt` denotes the time series (a vector of length T), `p` is the order of the AR(p) and `demean = TRUE` indicates that the mean of the process should be estimated (if this is not the case, then use `demean = FALSE`). The `select_method` input can be (among others) `"mle"` for the maximum likelihood and `"yule-walker"` for the Yule-Walker estimator or `"rgmwm"` for the RGMWM. For example, if you would like to estimate a zero mean AR(3) with the MLE you can use the code:

```
mod = estimate(AR(3), Xt, method = "mle", demean = FALSE)
```

On the other hand, if one wishes to estimate the parameters of this model through the RGMWM one can use the following syntax:

```
mod = estimate(AR(3), Xt, method = "rgmwm", demean = FALSE)
```

Removing the mean is not strictly necessary for the `rgmwm` (or `gmwm`) function since it won't estimate it and can consistently estimate the parameters of the time series model anyway. We now have the necessary R functions to deliver the above mentioned estimators and we can now proceed to the simulation study. In particular, we simulate three different processes X_t, Y_t, Z_t by using the first as an uncontaminated process defined as

$$X_t = 0.5X_{t-1} - 0.25X_{t-2} + W_t,$$

with $W_t \stackrel{iid}{\sim} N(0, 1)$. This first process (X_t) is uncontaminated while the other two processes are contaminated versions of the first that can often be observed in practice. The first type of contamination can be seen in (Y_t) and is delivered by replacing a portion of the original process with a process defined as

$$U_t = 0.90U_{t-1} - 0.40U_{t-2} + V_t,$$

where $V_t \stackrel{iid}{\sim} N(0, 9)$. The second form of contamination can be seen in (Z_t) and consists in the so-called point-wise contamination where randomly selected points from X_t are replaced with $N_t \stackrel{iid}{\sim} N(0, 9)$.

The code below performs the simulation study where it can be seen how the contaminated processes (Y_t) and (Z_t) are generated. Once this is done, for each simulation the code estimates the parameters of the AR(2) model using the three different estimation methods.

```
# Load simts
library(simts)

# Number of bootstrap iterations
B = 250

# Sample size
n = 500

# Proportion of contamination
eps = 0.05

# Number of contaminated observations
cont = round(eps*n)

# Simulation storage
res.Xt.MLE = res.Xt.YW = res.Xt.RGMWM = matrix(NA, B, 3)
res.Yt.MLE = res.Yt.YW = res.Yt.RGMWM = matrix(NA, B, 3)
res.Zt.MLE = res.Zt.YW = res.Zt.RGMWM = matrix(NA, B, 3)

# Begin bootstrap
for (i in seq_len(B)){
  # Set seed for reproducibility
  set.seed(1982 + i)

  # Generate processes
  Xt = gen_gts(n, AR(phi = c(0.5, 0.25), sigma2 = 1))
  Yt = Zt = Xt

  # Generate Ut contamination process that replaces a portion of original signal
  index_start = sample(1:(n-cont-1), 1)
  index_end = index_start + cont - 1
  Yt[index_start:index_end] = gen_gts(cont, AR(phi = c(0.9, -0.4), sigma2 = 9))

  # Generate Nt contamination that inject noise at random
  Zt[sample(n, cont, replace = FALSE)] = gen_gts(cont, WN(sigma2 = 9))

  # Fit Yule-Walker estimators on the three time series
  mod.Xt.YW = estimate(AR(2), Xt, method = "yule-walker",
                       demean = FALSE)
  mod.Yt.YW = estimate(AR(2), Yt, method = "yule-walker",
                       demean = FALSE)
  mod.Zt.YW = estimate(AR(2), Zt, method = "yule-walker",
                       demean = FALSE)

  # Store results
  res.Xt.YW[i, ] = c(mod.Xt.YW$mod$coef, mod.Xt.YW$mod$sigma2)
```

Figure 4.2: Boxplots of the empirical distribution functions of the Yule-Walker (YW), MLE and GMWM estimators for the parameters of the AR(2) model when using the X_t process (first row of boxplots), Y_t process (second row of boxplots) and Z_t (third row of boxplots).

```

res.Yt.YW[i, ] = c(mod.Yt.YW$mod$coef, mod.Yt.YW$mod$sigma2)
res.Zt.YW[i, ] = c(mod.Zt.YW$mod$coef, mod.Zt.YW$mod$sigma2)

# Fit MLE on the three time series
mod.Xt.MLE = estimate(AR(2), Xt, method = "mle",
                      demean = FALSE)
mod.Yt.MLE = estimate(AR(2), Yt, method = "mle",
                      demean = FALSE)
mod.Zt.MLE = estimate(AR(2), Zt, method = "mle",
                      demean = FALSE)

# Store results
res.Xt.MLE[i, ] = c(mod.Xt.MLE$mod$coef, mod.Xt.MLE$mod$sigma2)
res.Yt.MLE[i, ] = c(mod.Yt.MLE$mod$coef, mod.Yt.MLE$mod$sigma2)
res.Zt.MLE[i, ] = c(mod.Zt.MLE$mod$coef, mod.Zt.MLE$mod$sigma2)

# Fit RGMWM on the three time series
res.Xt.RGMWM[i, ] = estimate(AR(2), Xt, method = "rgmwm",
                             demean = FALSE)$mod$estimate
res.Yt.RGMWM[i, ] = estimate(AR(2), Yt, method = "rgmwm",
                             demean = FALSE)$mod$estimate
res.Zt.RGMWM[i, ] = estimate(AR(2), Zt, method = "rgmwm",
                             demean = FALSE)$mod$estimate
}

```

Having performed the estimation, we should now have 250 estimates for each AR(2) parameter and each estimation method. The code below takes the results of the simulation and shows them in the shape of boxplots along with the true values of the parameters. The estimation methods that are denoted as follows:

- **YW**: Yule-Walker estimator
- **MLE**: Maximum Likelihood Estimator
- **RGMWM**: the robust version of the GMWM estimator

It can be seen how all methods appear to properly estimate the true parameter values on average when they are applied to the simulated time series from the uncontaminated process (X_t). However, the MLE appears to be slightly more efficient (less variable) compared to the other methods and, in addition, the robust method (RGMWM) appears to be less efficient than the other two estimators. The latter is a known result since robust estimators usually pay a price in terms of efficiency (as an insurance against bias).

On the other hand, when checking the performance of the same methods when applied to the two contaminated processes (Y_t) and (Z_t) it can be seen that the standard estimators appear to be (highly) biased for most of the estimated parameters (with one exception) while the robust estimator remains close (on average) to the true parameter values that we are aiming to estimate. Therefore, when there's a suspicion that there could be some (small) contamination in the observed time series, it may be more appropriate to use a robust estimator.

To conclude this section on estimation, we now compare the above studied estimators in different applied settings where we can highlight how to assess which estimator is more appropriate according to the type of setting. For this purpose, let us start with an example we have already checked in the previous chapter when

Figure 4.3: Standard (left) and robust (left) estimates of the ACF function on the monthly precipitation data (hydro)

discussing standard and robust estimators of the ACF, more specifically the data on monthly precipitations. As mentioned before when discussing this example, the importance of modelling precipitation data lies in the fact that its usually used to successively model the entire water cycle. Common models for this purpose are either the white noise (WN) model or the AR(1) model. Let us compare the standard and robust ACF again to understand which of these two models seems more appropriate for the data at hand.

```
compare_acf(hydro)
```

As we had underlined in the previous chapter, the standard ACF estimates would suggest that there appears to be no correlation among lags and consequently, the WN model would be the most appropriate. However, the robust ACF estimates depict an entirely different picture where it can be seen that there appears to be a significant autocorrelation over different lags which exponentially decay. Although there appears to be some seasonality in the plot, we will assume that the correct model for this data is an AR(1) since that's what hydrology theory suggests. Let us therefore estimate the parameters of this model by using a standard estimator (MLE) and a robust estimator (RGMWM). The estimates for the MLE are the following:

```
mle_hydro = estimate(AR(1), as.vector(hydro), method = "mle", demean = TRUE)

# MLE Estimates
c(mle_hydro$mod$coef[1], mle_hydro$mod$sigma2)
```

```
##          ar1
## 0.06497727 0.22205713
```

From these estimates it would appear that the autocorrelation between lagged variables (i.e. lags of order 1) is extremely low and that (as suggested by the standard ACF plot) a WN model may be more appropriate. Considering the robust ACF however, it is possible that the MLE estimates may not be reliable in this setting. Hence, let us use the RGMWM to estimate the same parameters.

```
rgmwm_hydro = estimate(AR(1), hydro, method = "rgmwm")$mod$estimate

# RGMWM Estimates
t(rgmwm_hydro)
```

```
##          AR      SIGMA2
## Estimates 0.4048702 0.1065875
```

In this case, we see how the autocorrelation between lagged values is much higher (0.4 compared to 0.06) indicating that there is a stronger dependence in the data than what is suggested by standard estimators. Moreover, the innovation variance is smaller compared to that of the MLE. This is also a known phenomenon when there's contamination in the data since it leads to less dependence and more variability being detected by non-robust estimators. This estimate of the variance also has a considerable impact on forecast precision (as we'll see in the next section).

A final applied example that highlights the (potential) difference between estimators according to the type of setting is given by the “Recruitment” data set (in the `astsa` library). This data refers to the presence of new fish in the population of the Pacific Ocean and is often linked to the currents and temperatures passing through the ocean. As for the previous data set, let us take a look at the data itself and then analyse the standard and robust estimations of the ACF.

```
# Format data
fish = gts(rec, start = 1950, freq = 12, unit_time = 'month', name_ts = 'Recruitment')
```

Figure 4.4: Plot of the time series on fish recruitment monthly data in the Pacific Ocean from 1950 to 1987

Figure 4.5: Standard (left) and robust (left) estimates of the ACF function on the monthly fish recruitment data (rec)

```
# Plot data
plot(fish)

compare_acf(fish)
```

We can see that there appears to be a considerable dependence between the lagged variables which decays (in a similar way to the ACF of an AR(p)). Also in this case we see a seasonality in the data but we won't consider this for the purpose of this example. Given that there doesn't appear to be any significant contamination in the data, let us consider the Yule-Walker and MLE estimators. The MLE highly depends on the assumed parametric distribution of the time series (i.e. usually Gaussian) and, if this is not respected, the resulting estimations could be unreliable. Hence, a first difference of the time series can often give an idea of the marginal distribution of the time series.

```
# Take first differencing of the recruitment data
diff_fish = gts(diff(rec), start = 1950, freq = 12, unit_time = 'month', name_ts = 'Recruitment')

# Plot first differencing of the recruitment data
plot(diff_fish)
```

From the plot we can see that observations appear to be collected around a constant value and fewer appear to be further from this value (as would be the case for a normal distribution). However, various of these “more extreme” observations appear to be quite frequent suggesting that the underlying distribution may have a heavier tail compared to the normal distribution. Let us compare the standard and robust ACF estimates of this new time series.

```
compare_acf(diff_fish)
```

In this case we see that the patterns appear to be the same but the values between the standard and robust estimates are slightly (to moderately) different over different lags. This would suggest that there could be some contamination in the data or, in any case, that the normal assumption may not hold exactly. With this in mind, let us estimate an AR(2) model for this data using the Yule-Walker and MLE.

```
# MLE of Recruitment data
yw_fish = estimate(AR(2), rec, method = "yule-walker", demean = TRUE)

# MLE of Recruitment data
mle_fish = estimate(AR(2), rec, method = "mle", demean = TRUE)

# Compare estimates
# Yule-Walker Estimation
c(yw_fish$mod$coef[1], yw_fish$mod$sigma2)

##          ar1
## 1.354069 89.717052
```

Figure 4.6: Plot of the first difference of the time series on fish recruitment monthly data in the Pacific Ocean from 1950 to 1987

Figure 4.7: Standard (left) and robust (left) estimates of the ACF function on the first difference of the monthly fish recruitment data (rec)

```
# MLE Estimation
c(mle_fish$mod$coef[1], mle_fish$mod$sigma2)

##          ar1
## 1.351218 89.334361
```

It can be seen that, in this setting, the two estimators deliver very similar results (at least in terms of the ϕ_1 and ϕ_2 coefficients). Indeed, there doesn't appear to be a strong need for robust estimation and the choice of a standard estimator is justified by the will to obtain efficient estimations. The only slight difference between the two estimations is in the innovation variance parameter σ^2 and this could be (eventually) due to the normality assumption that the MLE estimator upholds in this case. If there is therefore a doubt on the fact that the Gaussian assumption does not hold for this data, then it is probably more convenient to use the Yule-Walker estimates.

Until now we have focussed on estimation based on an assumed model. However, how do we choose a model? How can we make inference on the models and their parameters? To perform all these tasks we will need to compute residuals (as, for example, in the linear regression framework). In order to obtain residuals, we need to be able to predict (forecast) values of the time series and, consequently, the next section focuses on forecasting time series.

4.2.3 Forecasting AR(p) Models

One of the most interesting aspects of time series analysis is to predict the future unobserved values based on the values that have been observed up to now. However, this is not possible if the underlying (parametric) model is unknown, thus in this section we assume, for purpose of illustration, that the time series (X_t) is **stationary** and its model is known. In particular, we denote forecasts by X_{t+j}^t , where t represents the time point from which we would like to make a forecast assuming we have an *observed* time series (e.g. $\mathbf{X} = (X_1, X_2, \dots, X_{t-1}, X_t)$) and j represents the j^{th} -ahead future value we wish to predict. So, X_{t+1}^t represents a one-step-ahead prediction of X_{t+1} given data $(X_1, X_2, \dots, X_{t-1}, X_t)$.

Let us now focus on defining a prediction operator and, for this purpose, let us define the Mean Squared Prediction Error (MSPE) as follows:

$$E[(X_{t+j} - X_{t+j}^t)^2].$$

Intuitively, the MPSE measures the square distance (i.e. always positive) between the actual future values and the corresponding predictions. Ideally, we would want this measure to be equal to zero (meaning that we don't make any prediction errors) but, if this is not possible, we would like to define a predictor that has the smallest MPSE among all possible predictors. The MPSE is not necessarily the best measure of prediction accuracy since, for example, it can be greatly affected by outliers or doesn't take into account importance of positive or negative misspredictions (e.g. for risks in finance or insurance). Nevertheless, it is a good overall measure of forecast accuracy and the next theorem states what the best predictor is for this measure.

Theorem 4.1 (Minimum Mean Squared Error Predictor). *Let us define*

$$X_{t+j}^t \equiv E[X_{t+j} | X_t, \dots, X_1], j > 0$$

Then

$$E[(X_{t+j} - m(X_1, \dots, X_t))^2] \geq E[(X_{t+j} - X_{t+j}^t)^2]$$

for any function $m(\cdot)$.

The proof of this theorem can be found in Appendix @ref(#appendixc). Although this theorem defines the best possible predictor, there can be many functional forms for this operator. We first restrict our attention to the set of linear predictors defined as

$$X_{t+j}^t = \sum_{i=1}^t \alpha_i X_i$$

where $\alpha_i \in \mathbb{R}$. It can be noticed, for example, that the α_i 's are not always the same based on the values of t and j (i.e. it depends from which time point you want to predict and how far into the future). Another aspect to notice is that, if the time series model underlying the observed time series can be expressed in the form of a linear operator (e.g. a linear process), then we can derive a linear predictor from this framework.

Let us consider some examples to understand how these linear predictors can be delivered based on the AR(p) models studied this far. For this purpose, let us start with an AR(1) process.

Example 4.8 (Forecasting with an AR(1) model). The AR(1) model is defined as follows:

$$X_t = \phi X_{t-1} + W_t$$

where $W_t \sim WN(0, \sigma^2)$.

From here, the conditional expected mean and variance are given by

$$\begin{aligned} E[X_{t+j}|\Omega_t] &= E[\phi X_{t+j-1} + W_{t+j} | \Omega_t] \\ &= \dots = \phi^j X_t \\ Var[X_{t+j}|\Omega_t] &= \left(1 + \phi^2 + \phi^4 + \dots + \phi^{2(j-1)}\right) \sigma^2 \end{aligned}$$

Within this derivation, it is important to remember that

$$\begin{aligned} \lim_{j \rightarrow \infty} E[X_{t+j}|\Omega_t] &= E[X_t] = 0 \\ \lim_{j \rightarrow \infty} Var_t[X_{t+j}|\Omega_t] &= var(X_t) = \frac{\sigma^2}{1 - \phi^2} \end{aligned}$$

From these last results we see that the forecast for an AR(1) process is “mean-reverting” since in general we have that the AR(1) can be written with respect to the mean μ as follows

$$(X_t - \mu) = \phi(X_{t-1} - \mu) + W_t,$$

which gives

$$X_t = \mu + \phi(X_{t-1} - \mu) + W_t,$$

and consequently

$$\begin{aligned} E[X_{t+j}|\Omega_t] &= \mu + E[(X_{t+j} - \mu)|\Omega_t] \\ &= \mu + \phi^j(X_t - \mu), \end{aligned}$$

which tends to μ as $j \rightarrow \infty$

Let us now consider a more complicated model, i.e. an AR(2) model.

Example 4.9 (Forecasting with an AR(2) model). Consider an AR(2) process defined as follows:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + W_t$$

where $W_t \sim WN(0, \sigma^2)$.

Based on this process we are able to find the predictor for each j -step ahead prediction using the following approach:

$$\begin{aligned} E[X_{t+1}|\Omega_t] &= \phi_1 X_t + \phi_2 X_{t-1} \\ E[X_{t+2}|\Omega_t] &= E[\phi_1 X_{t+1} + \phi_2 X_t|\Omega_t] = \phi_1 E[X_{t+1}|\Omega_t] + \phi_2 X_t \\ &= \phi_1 (\phi_1 X_t + \phi_2 X_{t-1}) + \phi_2 X_t \\ &= (\phi_1^2 + \phi_2) X_t + \phi_1 \phi_2 X_{t-1} \end{aligned}$$

Having seen how to build a forecast for an AR(1), let us now generalise this method to an AR(p) using matrix notation.

Example 4.10 (Forecasting with an AR(p) model). Consider AR(p) process defined as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + W_t$$

where $W_t \sim WN(0, \sigma_W^2)$.

The process can be rearranged into matrix form as follows:

$$\underbrace{\begin{bmatrix} X_t \\ \vdots \\ \vdots \\ X_{t-p+1} \end{bmatrix}}_{Y_t} = \underbrace{\begin{bmatrix} \phi_1 & \cdots & \phi_p \\ & & 0 \\ & I_{p-1} & \vdots \\ & & 0 \end{bmatrix}}_A \underbrace{\begin{bmatrix} X_{t-1} \\ \vdots \\ \vdots \\ X_{t-p} \end{bmatrix}}_{Y_{t-1}} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_C W_t$$

$$Y_t = AY_{t-1} + CW_t$$

From here, the conditional expectation and variance can be computed as follows:

$$\begin{aligned} E[Y_{t+j}|\Omega_t] &= E[AY_{t+j-1} + CW_{t+j}|\Omega_t] = E[AY_{t+j-1}|\Omega_t] + \underbrace{E[CW_{t+j}|\Omega_t]}_{=0} \\ &= E[A(AY_{t+j-2} + CW_{t+j-1})|\Omega_t] = E[A^2Y_{t+j-2}|\Omega_t] = A^j Y_t \\ \text{var}[Y_{t+j}|\Omega_t] &= \text{var}[AY_{t+j-1} + CW_{t+j}|\Omega_t] \\ &= \sigma^2 CC^T + \text{var}[AY_{t+j-1}|\Omega_t] = \sigma^2 A \text{var}[Y_{t+j-1}|\Omega_t] A^T \\ &= \sigma^2 CC^T + \sigma^2 ACC^T A + \sigma^2 A^2 \text{var}[Y_{t+j-2}|\Omega_t] (A^2)^T \\ &= \sigma^2 \sum_{k=0}^{j-1} A^k CC^T (A^k)^T \end{aligned}$$

Considering the recursive pattern coming from the expressions of the conditional expectation and variance, the predictions can be obtained via the following recursive formulation:

$$\begin{aligned} E[Y_{t+j}|\Omega_t] &= AE[Y_{t+j-1}|\Omega_t] \\ \text{var}[Y_{t+j}|\Omega_t] &= \sigma^2 CC^T + A \text{var}[Y_{t+j-1}|\Omega_t] A^T \end{aligned}$$

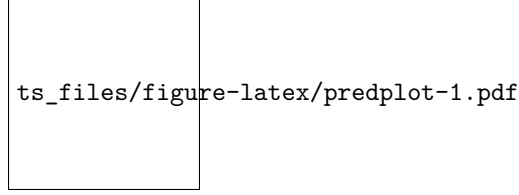


Figure 4.8: Values of the AR(2) predictions with the pink line being the median prediction, red and green lines are respectively 95% and 75% Confidence intervals.

With this recursive expression we can now compute the conditional expectation and variance of different AR(p) models. Let us therefore revisit the previous AR(2) example using this recursive formula.

Example 4.11 (Forecasting with an AR(2) in Matrix Form). We can rewrite our previous example of the predictions for an AR(2) process as follows

$$\underbrace{\begin{bmatrix} X_t \\ X_{t-1} \end{bmatrix}}_{Y_t} = \underbrace{\begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix}}_A \underbrace{\begin{bmatrix} X_{t-1} \\ X_{t-2} \end{bmatrix}}_{Y_{t-1}} + \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_C W_t$$

$$Y_t = AY_{t-1} + CW_t$$

Then, we are able to calculate the prediction as

$$\begin{aligned} E[Y_{t+2}|\Omega_t] &= A^2 Y_t = \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} X_t \\ X_{t-1} \end{bmatrix} \\ &= \begin{bmatrix} \phi_1^2 + \phi_2 & \phi_1 \phi_2 \\ \phi_1 & \phi_2 \end{bmatrix} \begin{bmatrix} X_t \\ X_{t-1} \end{bmatrix} = \begin{bmatrix} (\phi_1^2 + \phi_2) X_t + \phi_1 \phi_2 X_{t-1} \\ \phi_1 X_t + \phi_2 X_{t-1} \end{bmatrix} \end{aligned}$$

The above examples have given insight as to how to compute predictors for the AR(p) models that we have studied this far. Let us now observe the consequences of using such an approach to predict future values from these models. For this purpose, using the recursive formulation seen in the examples above we perform a simulation study where, for a fixed set of parameters, we make 5000 predictions from an observed time series and predict 50 values ahead into the future. The known parameters for the AR(2) process we use for the simulation study are $\phi_1 = 0.75$, $\phi_2 = 0.2$ and $\sigma^2 = 1$. The figure below shows the distribution of these predictions starting from the last observation $T = 200$.

It can be observed that, as hinted by the expressions for the variance of the predictions (in the examples above), the variability of the predictions increases as we try to predict further into the future.

Having now defined the basic concepts for forecasting of time series we can now start another topic of considerable importance for time series analysis. Indeed, the whole discussion on prediction is not only important to derive forecasts for future values of the phenomena one may be interested in, but also in understanding how well a model explains (and predicts) an observed time series. In fact, predictions allow to deliver residuals within the time series setting and, based on these residuals, we can obtain different inference and diagnostic tools.

Given the knowledge on predictions seen above, residuals can be computed as

$$r_{t+j} = X_{t+j} - \hat{X}_{t+j},$$

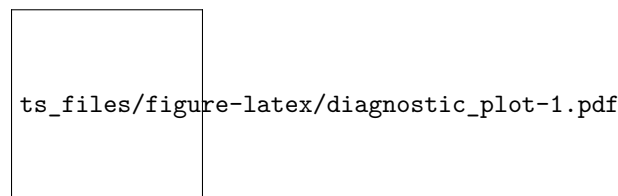
where \hat{X}_{t+j} represents an estimator of the conditional expectation $E[X_{t+j}|\Omega_t]$. The latter quantity will depend on the model underlying the time series and, assuming we know the true model, we could use the true parameters to obtain a value for this expectation. However, in an applied setting it is improbable that

we know the true parameters and therefore \hat{X}_{t+j} can be obtained by estimating the parameters (as seen in the previous sections) and then plugging them into the expression of $E[X_{t+j}|\Omega_t]$.

4.3 Diagnostic Tools for Time Series

The standard approach to understanding how well a model fits the data is analysing the residuals (e.g. linear regression). The same approach can be taken for time series analysis where, given our knowledge on forecasts from the previous section, we can now deliver residuals. The first step (and possibly the most important) is to use visual tools to check the residuals and also the original time series. Below is a figure that collects different diagnostic tools for time series analysis and is applied to a simulated AR(1) process of length $T = 100$.

```
set.seed(333)
true_model = AR(phi = 0.8, sigma2 = 1)
Xt = gen_gts(n = 100, model = true_model)
model = estimate(AR(1), Xt, demean = FALSE)
check(model = model)
```



All plots refer to the residuals of the model-fit and aim at visually assessing whether the fitted time series model captures the dependence structure in the data. The first row contains plots that can be interpreted in the same manner as the residuals from a linear regression. The first plot simply represents the residuals over time and should show if there is any presence of trends, seasonality or heteroskedasticity. The second plot gives a histogram (and smoothed histogram called a kernel) of the residuals which should be centered in zero and (possibly) have an approximate normal distribution. The latter hypothesis can also be checked using the third plot which consists in a Normal quantile-quantile plot and the residuals should lie on (or close to) the diagonal line representing correspondance between the distributions.

The first plot in the second row in the above figureshows the estimated ACF. It can be seen how the estimated autocorrelations all lie within the blue shaded area representing the confidence intervals. The second plot represents the Partial AutoCorrelation Function (PACF) and is another tool that will be investigated further on. In the meantime, suffice it to say that this tool can be interpreted as a different version of the ACF which measures autocorrelation conditionally on the previous lags (in some sense, it measures the direct impact of X_t on X_{t+h} removing the influence of all observations inbetween). In this case, we see that also the estimated PACF lies within the confidence intervals and, along with the interpretation of the ACF, we can state that the model appears to fit the time series reasonably well since the residuals can be considered as following a white noise process. Finally, the last plot visualises the “Ljung-Box Test” (which is a type of Portmanteau test) that tests whether the autocorrelation of the residuals at a set of lags is different from zero. The plot represents the p-value of this test at the different lags and also shows a dashed line representing the common significance level of $\alpha = 0.05$. If the modelling has done a good job, all p-values should be larger than this level. In this case, using the latter level, we can see that the autocorrelations for the residuals appear to be non-significant and therefore fitting an AR(1) model to the time series (which is the true model in this example) appears to do a reasonably good job.

4.3.1 The Partial AutoCorrelation Function (PACF)

Before further discussing these diagnostic tools, let us focus a little more on the PACF mentioned earlier. As briefly highlighted above, this function measures the degree of linear correlation between two lagged

observations by removing the effect of the observations in the intermediate lags. For example, let us assume we have three correlated variables X , Y and Z and we wish to find the direct correlation between X and Y : we can apply a linear regression of X on Z (to obtain \hat{X}) and Y on Z (to obtain \hat{Y}) and thereby compute $\text{corr}(\hat{X}, \hat{Y})$ which represents the partial autocorrelation. In this manner, the effect of Z on the correlation between X and Y is removed. Considering the variables as being indexed by time (i.e. a time series), we can therefore define the following quantities

$$\hat{X}_{t+h} = \beta_1 X_{t+h-1} + \beta_2 X_{t+h-2} + \dots + \beta_{h-1} X_{t+1},$$

and

$$\hat{X}_t = \beta_1 X_{t+1} + \beta_2 X_{t+2} + \dots + \beta_{h-1} X_{t+h-1},$$

which represent the regressions of X_{t+h} and X_t on all intermediate observations. With these definitions, let us more formally define the PACF.

Definition 4.3 (Partial AutoCorrelation Function). The partial autocorrelation function of a (weakly) stationary time series (X_t) is defined as

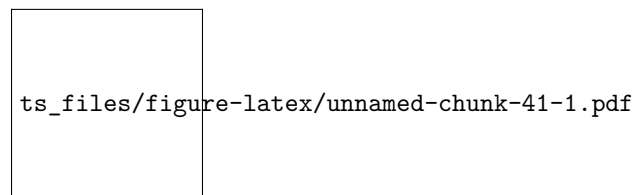
$$\rho_{1,1} = \text{corr}(X_{t+1}, X_t) = \rho(1)$$

and

$$\rho_{h,h} = \text{corr}((X_{t+h} - \hat{X}_{t+h}), (X_t - \hat{X}_t)), h > 1.$$

Therefore, the PACF is not defined at lag zero while it is the same as the ACF at the first lag since there are no intermediate observations. The definition of this function is important for different reasons. Firstly, as mentioned earlier, we have an asymptotic distribution for the estimator of the PACF which allows us to deliver confidence intervals as for the ACF. Indeed, representing the empirical PACF along with its confidence intervals can provide a further tool to make sure that the residuals of a model fit can be considered as being white noise. The latter is a reasonable hypothesis if both the empirical ACF and PACF lie within the confidence intervals indicating that there is no significant correlation between lagged variables. Let us focus on the empirical PACF of the residuals we saw earlier in the diagnostic plot.

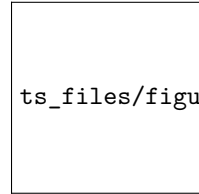
```
residuals = model$mod$residuals
plot(auto_corr(residuals, pacf = TRUE))
```



From the plot we see that the estimated PACF at all lags lies within the blue shaded area representing the 95% confidence intervals indicating that the residuals don't appear to have any form of direct autocorrelation over lags and, hence, that they can be considered as white noise. In addition to delivering a tool to analyse the residuals of a model fit, the PACF is especially useful for diagnostic purposes in order to detect the possible models that have generated an observed time series. With respect to the class of models studied this far (i.e. AR(p) models), the PACF can give important insight to the order of the AR(p) model, more specifically the value of p . Indeed, the ACF of an AR(p) model tends to decrease in a sinusoidal fashion and exponentially fast to zero as the lag h increases thereby giving a hint that the underlying model belongs to the AR(p) family. However, aside from hinting to the fact that the model belongs to the family of AR(p) models, the latter function tends to be less informative with respect to which AR(p) model (i.e. which value of p) is best suited for the observed time series. The PACF on the other hand is non-zero for the first p lags and then becomes zero for $h > p$, therefore it can give an important indication with respect to which order should be considered to best model the time series.

With the above discussion in mind, let us consider some examples where we study the theoretical ACF and PACF of different AR(p) models. The first is an AR(1) model with parameter $\phi = 0.95$ and its ACF and PACF are shown below.

```
par(mfrow = c(1, 2))
plot(theo_acf(ar = 0.95, ma = NULL))
plot(theo_pacf(ar = 0.95, ma = NULL))
```



We can see that the ACF decreases exponentially fast as we would expect for an AR(p) model while the PACF has the same values of the ACF at lag 1 (which is always the case) and is zero for $h > 1$. The latter therefore indicates the order p of the AR(p) model which in this case is $p = 1$. Let us now consider the following models as further examples:

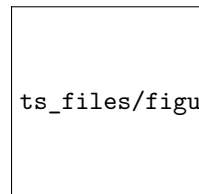
$$X_t = 0.5X_{t-1} + 0.25X_{t-2} + 0.125X_{t-3} + W_t$$

and

$$X_t = -0.1X_{t-2} - 0.1X_{t-4} + 0.6X_{t-5} + W_t.$$

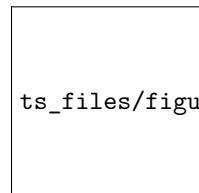
The first model is an AR(3) model with positive coefficients while the second is an AR(5) model where $\phi_1 = \phi_3 = 0$. The ACF and PACF of the first AR(3) model is shown below.

```
par(mfrow = c(1, 2))
plot(theo_acf(ar = c(0.5, 0.25, 0.125), ma = NULL))
plot(theo_pacf(ar = c(0.5, 0.25, 0.125), ma = NULL))
```



Again, we see that the ACF decreases exponentially and also the PACF plot shows a steady decrease until it becomes zero for $h > 3$. This is because the values of ϕ_i are all positive and decreasing as well. Let us now consider the second model (the AR(5) model) whose ACF and PACF plots are represented below.

```
par(mfrow = c(1, 2))
plot(theo_acf(ar = c(0, -0.1, 0, -0.1, 0.6), ma = NULL))
plot(theo_pacf(ar = c(0, -0.1, 0, -0.1, 0.6), ma = NULL))
```

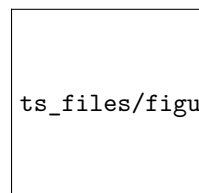


In this case we see that the ACF has a sinusoidal-like behaviour where the values of $\phi_1 = \phi_3 = 0$ and the negative $\phi_2 = \phi_4 = -0.1$ values deliver this alternating ACF form which nevertheless decreases as h increases.

These values deliver the PACF plot on the right which is negative for the first lags and is greatly positive at $h = 5$ only to become zero for $h > 5$.

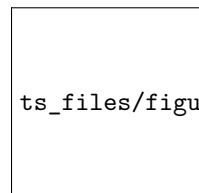
These examples therefore give us further insight as to how to interpret the empirical (or estimated) versions of these functions. For this reason, let us study the empirical ACF and PACF of some real time series, the first of which is the data representing the annual number of Lynx trappings in Canada between 1821 and 1934. The time series is represented below.

```
lynx_gts = gts(lynx, start = 1821, data_name = "Lynx Trappings", unit_time = "year", name_ts = "Trapping",
plot(lynx_gts)
```



We can see that there appears to be a seasonal trend within the data but let us ignore this for the moment and check the ACF and PACF plots below.

```
par(mfrow = c(1, 2))
plot(auto_corr(lynx_gts))
plot(auto_corr(lynx_gts, pacf = TRUE))
```



The seasonal behaviour also appears in the ACF plot but we see that it decreases in a sinusoidal fashion as the lag increases hinting that an AR(p) model could be a potential candidate for the time series. Looking at the PACF plot on the right we can see that a few partial autocorrelations appear to be significant up to lag $h = 8$. Therefore an AR(8) model could be a possibly good candidate to explain (and predict) this time series.

4.3.2 Portmanteau Tests

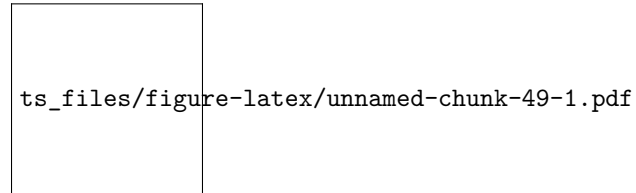
In a similar manner to using the ACF and PACF confidence intervals to understand if the residuals can be considered as white noise, other statistical tests exist to determine if the autocorrelations can be considered as being significant or not. Indeed, the 95% confidence intervals can be extremely useful in detecting significant (partial) autocorrelations but they cannot be considered as an overall test for white noise since (aside from being asymptotic and therefore approximate) they do not consider the multiple testing hypothesis implied by them (i.e. if we're testing all lags, the actual level of significance should be modified in order to bound type I errors). For this reason, Portmanteau tests have been proposed in order to deliver an overall test that jointly considers a set of lags and determines whether their autocorrelation is significantly different from zero (i.e. whether the residuals can be considered white noise or not).

One of the most popular Portmanteau tests is the Ljung-Box test whose statistic is defined as follows:

$$Q_h = T(T+2) \sum_{j=1}^h \frac{\hat{\rho}_j^2}{T-j},$$

where h represents the maximum lag of the set of lags for which we wish to test for serial autocorrelation (i.e. from lag 1 to lag h). Under the null hypothesis $Q_h \sim \chi_h^2$ since asymptotically $\hat{\rho}_j \sim \mathcal{N}(0, \frac{1}{T})$ under the null and therefore the statistic is proportional to the sum of squared standard normal variables. The last plot of Figure @ref(fig:diagnostic_plot) can therefore be better interpreted under this framework and is reproduced below.

```
lb_test = diag_ljungbox(as.numeric(residuals), order = 0, stop_lag = 20, stdres = FALSE, plot = TRUE)
```



Each p-value represented in the plot above is therefore the p-value for the considered maximum lag. Therefore the first p-value (at lag 1) is the p-value for the null hypothesis that all lags up to lag 1 are equal to zero, the second is for the null hypothesis that all lags up to lag 2 are equal to zero and so on. Hence, each p-value represents a global test on the autocorrelations up to the considered maximum lag h and delivers additional information regarding the dependence structure within the residuals. In this case, as interpreted earlier, all p-values are larger than the $\alpha = 0.05$ level and is an additional indication that the residuals follow a white noise process (and that the fitted model appears to well describe the observed time series).

There are other Portmanteau tests with different advantages (and disadvantages) over the Ljung-Box test but these are beyond the scope of this textbook.

4.4 Inference for AR(p) Models

For all the above methods, it would be necessary to understand how “precise” their estimates are. To do so we would need to obtain confidence intervals for these estimates and this can be done mainly in two manners:

- using the asymptotic distribution of the parameter estimates;
- using parametric bootstrap.

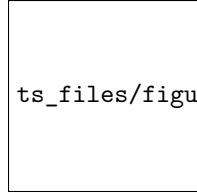
The first approach consists in using the asymptotic distribution of the estimators presented earlier to deliver approximations of the confidence intervals which get better as the length of the observed time series increases. Hence, if for example we wanted to find a 95% confidence interval for the parameter ϕ , we would use the quantiles of the normal distribution (given that all methods presented earlier present this asymptotic distribution). However, this approach can present some drawbacks, one of which its behaviour when the parameters are close to the boundaries of the parameter space. Suppose we consider a realization of length $T = 100$ of the following AR(1) model:

$$X_t = 0.96X_{t-1} + W_t, \quad W_t \sim \mathcal{N}(0, 1),$$

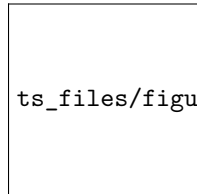
which is represented in Figure ??

```
set.seed(8)
x = gen_gts(n = 100, AR1(0.96, 1))
plot(x)
```

It can be seen that the parameter $\phi = 0.96$ respects the condition for stationarity (i.e. $|\phi| < 1$) but is very close to its boundary. Using the MLE and the GMWM estimators, we first estimate the parameters and then compute confidence intervals for ϕ using the asymptotic normal distribution.



ts_files/figure-latex/simAR1ci-1.pdf

Figure 4.9: AR(1) with ϕ close to parameter bound


ts_files/figure-latex/asymIC-1.pdf

Figure 4.10: Estimated asymptotic distribution of $\hat{\phi}$ for MLE and GMWM parameter estimates. The dashed vertical line represents the true value of ϕ , the solid line denotes the upper bound of the parameter space for ϕ and the vertical ticks represent the limits of the 95% confidence intervals for both methods.

```
# Compute the parameter estimates using MLE
fit.ML = estimate(AR(1), x, demean = FALSE)
c("phi" = fit.ML$mod$coef[1], "se" = sqrt(fit.ML$mod$sigma2))

## phi.ar1      se
## 0.898898 1.067223

# Construct asymptotic confidence interval for phi
fit.ML$mod$coef[1] + c(-1,1)*1.96*as.numeric(sqrt(fit.ML$mod$var.coef))

## [1] 0.8171176 0.9806783

# Compute the parameter estimates with inference using GMWM
fit.gmwm = gmwm(AR1(), x)
res.gmwm = summary(fit.gmwm, inference = TRUE)$estimate
res.gmwm

##      Estimates    CI Low  CI High      SE
## AR1    0.7291624 0.5628965 0.8389802 0.08392348
## SIGMA2 1.0193074 0.7423558 1.2160671 0.14399803
```

From the estimation summary, we can notice that both confidence intervals contain values that would make the AR(1) non-stationary (i.e. values of ϕ larger than 1). However, these confidence intervals are based on the (asymptotic) distributions of $\hat{\phi}$ which are shown in Figure ??.

Therefore, if we estimate a stationary AR(1) model, it would be convenient to have more “realistic” confidence intervals that give limits for a stationary AR(1) model. A viable solution for this purpose is to use parametric bootstrap. Indeed, parametric bootstrap takes the estimated parameter values and uses them in order to simulate from the assumed model (an AR(1) process in this case). For each simulation, the parameters are estimated and stored in order to obtain an empirical (finite sample) distribution of the estimators. Based on this distribution it is consequently possible to find the $\alpha/2$ and $1 - \alpha/2$ empirical quantiles thereby delivering confidence intervals that should not suffer from boundary problems (since the estimation procedure looks for solutions within the admissible regions). The code below gives an example of how this confidence interval is built based on the same estimation procedure but using parametric bootstrap (using $B = 10000$ bootstrap replicates).

```

# Number of Iterations
B = 50

# Set up storage for results
est.phi.gmwm = rep(NA,B)
est.phi.ML = rep(NA,B)

# Model generation statements
model.gmwm = AR(phi = c(fit.gmwm$estimate[1]), sigma2 = c(fit.gmwm$estimate[2]))
model.mle = AR(phi = c(fit.ML$mod$coef), sigma2 = c(fit.ML$mod$sigma2))

# Begin bootstrap
for(i in seq_len(B)){

  # Set seed for reproducibility
  set.seed(B + i)

  # Generate process under MLE parameter estimate
  x.star = gen_gts(100, model.mle)

  # Attempt to estimate phi by employing a try
  est.phi.ML[i] = tryCatch(estimate(AR(1), x.star, demean = FALSE)$mod$coef,
                           error = function(e) NA)

  # Generate process under GMWM parameter estimate
  x.star = gen_gts(100, model.gmwm)
  est.phi.gmwm[i] = estimate(AR(1), x.star, method = "gmwm")$mod$estimate[1]
}

```

In Figure ??, we compare the estimated densities for $\hat{\phi}$ using asymptotic results and bootstrap techniques for the MLE and the GMWM estimators. It can be observed that the issue that arose previously with “non-stationary” confidence intervals does not occur with the parametric bootstrap approach since the interval regions of the latter lie entirely within the boundaries of the admissible parameter space.

To further emphasize the advantages of parametric bootstrap, let us consider one additional example of an AR(2) process. For this example, discussion will focus solely on the behaviour of the MLE. Having said this, the considered AR(2) process is defined as follows:

$$X_t = 1.98X_{t-1} - 0.99X_{t-2} + W_t \quad (4.10)$$

where $W_t \sim WN(0, 1)$. The generated process is displayed in Figure ??.

```

set.seed(432)
Xt = gen_gts(500, AR(phi = c(1.98, -0.99), sigma2 = 1))
plot(Xt)

fit = estimate(AR(2), Xt, demean = FALSE)
fit

## Fitted model: AR(2)
##
## Estimated parameters:
##
## Call:
## arima(x = as.numeric(Xt), order = c(p, integrated, q), seasonal = list(order = c(P,

```

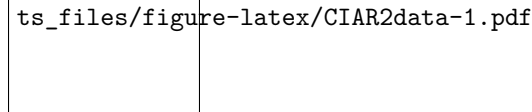


Figure 4.11: Generated AR(2) Process

```
##      seasonal_intergrated, Q), period = s), include.mean = demean, method = meth)
##
## Coefficients:
##          ar1      ar2
##      1.9787 -0.9892
## s.e.  0.0053  0.0052
##
## sigma^2 estimated as 0.8419:  log likelihood = -672.58,  aic = 1351.15
```

With the model's coefficients readily available, the parametric bootstrap is able to be constructed. As before, the bootstrap implementation mirrors prior versions with one caveat regarding the storage of ϕ being two dimensional and, thus, requiring a *matrix* to store the result instead of a traditional atomic vector in *R*.

```
B = 50
est.phi.ML = matrix(NA, B, 2)
model = AR(phi = fit$mod$coef, sigma2 = fit$mod$sigma2)

for(i in seq_len(B)) {

  set.seed(B + i)                # Set Seed for reproducibility

  x.star = gen_gts(500, model) # Simulate the process

  # Obtain parameter estimate with protection
  # that defaults to NA if unable to estimate
  est.phi.ML[i,] = tryCatch(estimate(AR(2), x.star, demean = FALSE)$mod$coef,
    error = function(e) NA)

}
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:simts':
##
##      select
```

As for any other estimation and inference procedure, these confidence intervals rely on the assumption that the chosen model (AR(2) in this case) is the true model underlying the observed time series. However, if this assumption were not correct, then the derived confidence intervals would not be correct and this could possibly lead to bad conclusions. For this reason, one could decide to use a non-parametric approach to computing these confidence intervals, such as the traditional bootstrap that consists in a simple random sampling with replacement from the original data. However, this approach would not be appropriate in the presence of dependence between observations since this procedure would break the dependence structure and therefore deliver unreliable conclusions. Therefore, to preserve the dependence structure of the original data one option would be to use *block bootstrapping*, which is a particular form of non-parametric bootstrap. There are many different kinds of block-bootstrap procedures for time series which are all related to the

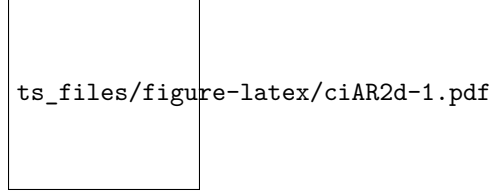


Figure 4.12: Estimated distributions of $\hat{\phi}_1$ and $\hat{\phi}_2$ based on the MLE using asymptotic and parametric bootstrap techniques. The colored contours represent the density of the distributions and the dark grey lines represent the boundary constraints of $|\phi_2| < 1$ and $\phi_2 = 1 - \phi_1$.

Moving Block Bootstrap (MBB) which is defined as follows.

Definition 4.4. Suppose that (X_t) is a weakly stationary time series with T observations. The procedure is as follows:

1. Divide the time series into overlapping blocks $\{S_1, \dots, S_M\}$ of length ℓ , $1 \leq \ell < N$, resulting in $M = N - \ell + 1$ blocks structured as:

$$\begin{aligned} S_1 &= (X_1, X_2, \dots, X_\ell) \\ S_2 &= (X_2, X_3, \dots, X_{\ell+1}) \\ &\dots \\ S_M &= (X_{N-\ell+1}, X_{N-\ell+2}, \dots, X_N) \end{aligned}$$

2. Draw $M = \lfloor \frac{N}{\ell} \rfloor$ blocks with replacement from these $\{S_1, \dots, S_M\}$ blocks and place them in order to form (X_t^*) .
3. Compute the statistic of interest on the simulated sample (X_t^*) .
4. Repeat Steps 2 and 3 B times where B is sufficiently “large” (typically $100 \leq B \leq 10000$).
5. Compute the empirical mean and variance on the statistic of interest based on the B independent replications.

The approach taken by MBB ensures that within each block the dependency between observations is preserved. However, one particular issue that now arises is that some inaccuracy is introduced as a result of successive blocks potentially being independent from each other. In reality, this is one of the trade-offs of the MBB approach that can be mitigated by selecting an optimal ℓ (to improve the accuracy of the procedure one should choose $\ell = T^{1/3}$ as $T \rightarrow \infty$ as being the optimal choice. An earlier variant of MBB is called the nonoverlapping block bootstrap (NBB) which doesn’t allow the blocks to share common data points and is presented below.

Definition 4.5. Suppose that (X_t) is weakly stationary time series with T observations.

1. Divide time series into nonoverlapping blocks $\{S_1, \dots, S_M\}$ of length ℓ , $1 \leq \ell < N$, resulting in $M = \lfloor \frac{N}{\ell} \rfloor$ blocks structured as:

$$\begin{aligned} S_1 &= (X_1, X_2, \dots, X_\ell) \\ S_2 &= (X_{\ell+1}, X_{\ell+2}, \dots, X_{2\ell}) \\ &\dots \\ S_K &= (X_{(K-1)\ell+1}, X_{(K-1)\ell+2}, \dots, X_N) \end{aligned}$$

2. Draw M blocks with replacement from these $\{S_1, \dots, S_M\}$ blocks and place them in order to form (X_t^*) .
3. Compute the statistic of interest on the simulated sample (X_t^*) .
4. Repeat Steps 2 and 3 B times where B is sufficiently “large” (typically $100 \leq B \leq 10000$).

5. Compute the empirical mean and variance on the statistic of interest based on the B independent replications.

Alternatively, depending on the case one can also use modifications of the MBB that seeks to change how the beginning and end of the time series is weighted such as a circular block-bootstrap (CBB) or a stationary bootstrap (SB). Regardless, the outcomes from using MBB on time series data are considerably better than simple resampling. The code below implements the MMB to obtain a bootstrap distribution for the estimated parameters of an AR(1) model.

```
ar1_blockboot = function(Xt, block_len = 10, B = 500) {

  n = length(Xt)           # Length of Time Series
  res = rep(NA, B)         # Bootstrapped Statistics
  m = floor(n/block_len)   # Amount of Blocks

  for (i in seq_len(B)) {  # Begin MMB

    set.seed(i + 1199)     # Set seed for reproducibility
    x_star = rep(NA, n)    # Setup storage for new TS

    for (j in seq_len(m)) { # Simulate new time series

      index = sample(m, 1) # Randomize block starting points

      # Place block into time series
      x_star[(block_len*(j - 1) + 1):(block_len*j)] =
        Xt[(block_len*(index - 1) + 1):(block_len*index)]
    }

    # Calculate parameters with protection
    res[i] = tryCatch(estimate(AR(1), x_star, demean = FALSE)$mod$coef,
      error = function(e) NA)

  }

  na.omit(res)             # Deliver results

}
```

Having defined the function above, let us consider a scenario where the model's assumption that the residuals are Gaussian is violated. Consider two AR(1) processes with the same coefficients but with different innovation noise generation procedures:

$$\begin{aligned}\mathcal{M}_1: & X_t = 0.5X_{t-1} + W_t, \quad W_t \sim \mathcal{N}(0, 1) \\ \mathcal{M}_2: & X_t = 0.5X_{t-1} + V_t, \quad V_t \sim t_4\end{aligned}$$

where t_4 represents the Student- t distribution with 4 degrees of freedom (heavy tails). The generation procedure for \mathcal{M}_1 is straightforward, use: `gen_gts()`. For the first model (\mathcal{M}_1) we will use the `gen_gts()` function while for the second model \mathcal{M}_2 we will make use the `arima.sim()` function in which we can assign a vector of innovations from a t distribution.

```
set.seed(1)                                     # Set seed for reproducibility
xt_m1 = gen_gts(500, AR1(phi = 0.5, sigma2 = 1)) # Gaussian noise only
xt_m2 = gts(arima.sim(n = 500, list(ar = 0.5, ma = 0), # Student-t noise
  rand.gen = function(n, ...) rt(n, df = 4)))
```

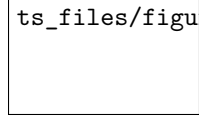

 ts_files/figure-latex/mbbdatavis-1.pdf

Figure 4.13: AR(1) processes generated under different noise conditions.

From Figure ??, the time series appear to be considerably different as a result of the noise process being altered even though the ϕ coefficient was held constant. Among others, the difference is also related to the variance of the t distribution defined as $\frac{\nu}{\nu-2}$ for $\nu > 2$ (and ∞ for $\nu \leq 2$). Therefore, the innovation variance of the i.i.d white noise process is given by $\sigma_{\mathcal{M}_1}^2 = 1$ while the noise generated from the t distribution has innovation variance $\sigma_{\mathcal{M}_2}^2 = 2$. To compare the performance of the bootstrap procedures we now implement the corresponding function for the parametric bootstrap:

```
ar1_paraboot = function(model, B = 10000) {

  est.phi = rep(NA,B)    # Define a storage vector

  for(i in seq_len(B)) { # Perform bootstrap

    set.seed(B + i)      # Set seed for reproducibility

    # Simulate time series underneath the estimated model
    x.star = gen_gts(500, AR(phi = model$mod$coef, sigma2 = model$mod$sigma2))

    # Attempt to estimate parameters with recovery
    est.phi[i] = tryCatch(estimate(AR(1), x.star, demean = FALSE)$mod$coef,
      error = function(e) NA)

  }

  na.omit(est.phi)      # Return estimated phis

}
```

Now that we have two functions implementing the block and parametric bootstraps, we can now apply these procedures to obtain the confidence intervals for the AR(1) parameters for the two time series generated respectively by model \mathcal{M}_1 and \mathcal{M}_2 .

```
B = 50

# Model 1
fit_m1_mle = estimate(AR(1), xt_m1, demean = FALSE)
para_m1_phi = ar1_paraboot(fit_m1_mle, B = B)
block_m1_phi = ar1_blockboot(xt_m1, block_len = 25, B = B)

# Model 2
fit_m2_mle = estimate(AR(1), xt_m2, demean = FALSE)
para_m2_phi = ar1_paraboot(fit_m2_mle, B = B)
block_m2_phi = ar1_blockboot(xt_m2, block_len = 25, B = B)
```

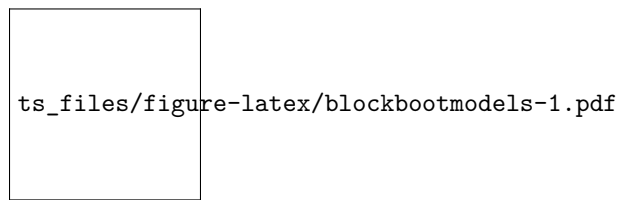


Figure 4.14: Estimated parametric and non-parametric blockbootstrap distributions of $\hat{\phi}$ for the MLE parameter estimates. The histogram bars represent the empirical results from the bootstraps with the green representing parametric bootstrap and the red representing the block bootstrap approach. The dashed vertical line represents the true value of ϕ and the vertical ticks correspond to the limits of the 95% confidence intervals for both estimation techniques.

Appendix A

Proofs

A.1 Proof of Theorem 1

We let $X_t = W_t + \mu$, where $\mu < \infty$ and (W_t) is a strong white noise process with variance σ^2 and finite fourth moment (i.e. $\mathbb{E}[W_t^4] < \infty$).

Next, we consider the sample autocovariance function computed on (X_t) , i.e.

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (X_t - \bar{X}) (X_{t+h} - \bar{X}).$$

For this equation, it is clear that $\hat{\gamma}(0)$ and $\hat{\gamma}(h)$ (with $h > 0$) are two statistics involving sums of different lengths. As we will see, this prevents us from using directly the multivariate central limit theorem on the vector $[\hat{\gamma}(h) \quad \hat{\gamma}(h)]^T$. However, the lag h is fixed and therefore the difference in the number of elements of both sums is asymptotically negligible. Therefore, we define a new statistic

$$\tilde{\gamma}(h) = \frac{1}{n} \sum_{t=1}^n (X_t - \mu) (X_{t+h} - \mu),$$

which, as we will see, is easier to use and show that $\hat{\gamma}(h)$ and $\tilde{\gamma}(h)$ are asymptotically equivalent in the sense that:

$$n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)] = o_p(1).$$

Therefore, assuming this result to be true, $\tilde{\gamma}(h)$ and $\hat{\gamma}(h)$ would have the same asymptotic distribution, it is sufficient to show the asymptotic distribution of $\tilde{\gamma}(h)$. So that before continuing the proof of Theorem 1 we first state and prove the following lemma:

Lemma A1: Let

$$X_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j W_{t-j},$$

where (W_t) is a strong white process with variance σ^2 , and the coefficients satisfying $\sum |\psi_j| < \infty$. Then, we have

$$n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)] = o_p(1).$$

Proof: By Markov inequality, we have

$$\mathbb{P}\left(|n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)]| \geq \epsilon\right) \leq \frac{\mathbb{E}|n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)]|}{\epsilon},$$

for any $\epsilon > 0$. Thus, it is enough to show that

$$\lim_{n \rightarrow \infty} \mathbb{E}\left[|n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)]|\right] = 0$$

to prove Lemma A1. By the definitions of $\tilde{\gamma}(h)$ and $\hat{\gamma}(h)$, we have

$$\begin{aligned} n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)] &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) \\ &\quad + \frac{1}{\sqrt{n}} \sum_{t=1}^{n-h} [(X_t - \mu)(X_{t+h} - \mu) - (X_t - \bar{X})(X_{t+h} - \bar{X})] \\ &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) + \frac{1}{\sqrt{n}} \sum_{t=1}^{n-h} [(\bar{X} - \mu)(X_t + X_{t+h} - \mu - \bar{X})] \\ &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) + \frac{1}{\sqrt{n}} (\bar{X} - \mu) \sum_{t=1}^{n-h} (X_t + X_{t+h} - \mu - \bar{X}) \\ &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) + \frac{1}{\sqrt{n}} (\bar{X} - \mu) \left[\sum_{t=1+h}^{n-h} X_t - (n-h)\mu + h\bar{X} \right] \\ &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) + \frac{1}{\sqrt{n}} (\bar{X} - \mu) \left[\sum_{t=1+h}^{n-h} (X_t - \mu) - h(\mu - \bar{X}) \right] \\ &= \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n (X_t - \mu)(X_{t+h} - \mu) + \frac{1}{\sqrt{n}} (\bar{X} - \mu) \sum_{t=1+h}^{n-h} (X_t - \mu) + \frac{h}{\sqrt{n}} (\bar{X} - \mu)^2, \end{aligned}$$

where $\bar{X} = \frac{1}{n} \sum_{t=1}^n X_t = \mu + \frac{1}{n} \sum_{t=1}^n \sum_{j=-\infty}^{\infty} \psi_j W_{t-j} = \mu + \frac{1}{n} \sum_{j=-\infty}^{\infty} \sum_{t=1}^n \psi_j W_{t-j}$.

Then, we have

$$\begin{aligned} \mathbb{E}\left[|n^{\frac{1}{2}}[\tilde{\gamma}(h) - \hat{\gamma}(h)]|\right] &\leq \frac{1}{\sqrt{n}} \sum_{t=n-h+1}^n \mathbb{E}[|(X_t - \mu)(X_{t+h} - \mu)|] \\ &\quad + \frac{1}{\sqrt{n}} \mathbb{E}\left[\left|(\bar{X} - \mu) \sum_{t=1+h}^{n-h} (X_t - \mu)\right|\right] + \frac{h}{\sqrt{n}} \mathbb{E}[(\bar{X} - \mu)^2]. \end{aligned}$$

Next, we consider each term of the above equation. For the first term, since $(X_t - \mu)^2 = \left(\sum_{j=-\infty}^{\infty} \psi_j W_{t-j}\right)^2$, and $\mathbb{E}[W_i W_j] \neq 0$ only if $i = j$. By Cauchy-Schwarz inequality we have

$$\mathbb{E}[|(X_t - \mu)(X_{t+h} - \mu)|] \leq \sqrt{\mathbb{E}[|(X_t - \mu)|^2] \mathbb{E}[|(X_{t+h} - \mu)|^2]} = \sigma^2 \sum_{i=-\infty}^{\infty} \psi_i^2.$$

Then, we consider the third term, since it will be used in the second term

$$\mathbb{E}[(\bar{X} - \mu)^2] = \frac{1}{n^2} \sum_{t=1}^n \sum_{i=-\infty}^{\infty} \psi_i^2 \mathbb{E}[W_{t-i}^2] = \frac{\sigma^2}{n} \sum_{i=-\infty}^{\infty} \psi_i^2.$$

Similarly, for the second term we have

$$\begin{aligned} \mathbb{E} \left[\left| (\bar{X} - \mu) \sum_{t=1+h}^{n-h} (X_t - \mu) \right| \right] &\leq \sqrt{\mathbb{E} [|(\bar{X} - \mu)|^2] \mathbb{E} \left[\left| \sum_{t=1+h}^{n-h} (X_t - \mu) \right|^2 \right]} \\ &= \sqrt{\mathbb{E} [(\bar{X} - \mu)^2] \mathbb{E} \left[\sum_{t=1+h}^{n-h} (X_t - \mu)^2 + \sum_{t_1 \neq t_2} (X_{t_1} - \mu)(X_{t_2} - \mu) \right]} \\ &\leq \sqrt{\frac{\sigma^2}{n} \sum_{i=-\infty}^{\infty} \psi_i^2 \cdot (n-2h)\sigma^2 \left(\sum_{j=-\infty}^{\infty} |\psi_j| \right)^2} \\ &\leq \sqrt{\frac{n-2h}{n}} \sigma^2 \left(\sum_{i=-\infty}^{\infty} |\psi_i| \right)^2. \end{aligned}$$

Combining the above results we obtain

$$\begin{aligned} \mathbb{E} |n^{\frac{1}{2}} [\tilde{\gamma}(h) - \hat{\gamma}(h)]| &\leq \frac{1}{\sqrt{n}} h \sigma^2 \sum_{i=-\infty}^{\infty} \psi_i^2 + \sqrt{\frac{n-2h}{n^2}} \sigma^2 \left(\sum_{i=-\infty}^{\infty} |\psi_i| \right)^2 + \frac{h}{n\sqrt{n}} \sigma^2 \sum_{i=-\infty}^{\infty} \psi_i^2 \\ &\leq \frac{1}{n\sqrt{n}} (nh + \sqrt{n-2h} + h) \sigma^2 \left(\sum_{i=-\infty}^{\infty} |\psi_i| \right)^2, \end{aligned}$$

By the taking the limit in n we have

$$\lim_{n \rightarrow \infty} \mathbb{E} \left[|n^{\frac{1}{2}} [\tilde{\gamma}(h) - \hat{\gamma}(h)]| \right] \leq \sigma^2 \left(\sum_{i=-\infty}^{\infty} |\psi_i| \right)^2 \lim_{n \rightarrow \infty} \frac{nh + \sqrt{n-2h} + h}{n\sqrt{n}} = 0.$$

We can therefore conclude that

$$\sqrt{n}[\tilde{\gamma}(h) - \hat{\gamma}(h)] = o_p(1),$$

which concludes the proof of Lemma A1. \blacksquare

Returning to the proof of Theorem 1, since the process (Y_t) , where $Y_t = (X_t - \mu)(X_{t+h} - \mu)$, is iid, we can apply multivariate central limit theorem to the vector $[\tilde{\gamma}(h) \quad \hat{\gamma}(h)]^T$, and we obtain

$$\begin{aligned} \sqrt{n} \left\{ \begin{bmatrix} \tilde{\gamma}(0) \\ \tilde{\gamma}(h) \end{bmatrix} - \mathbb{E} \begin{bmatrix} \tilde{\gamma}(0) \\ \tilde{\gamma}(h) \end{bmatrix} \right\} &= \frac{1}{\sqrt{n}} \begin{bmatrix} \sum_{t=1}^n (X_t - \mu)^2 - n\mathbb{E}[\tilde{\gamma}(0)] \\ \sum_{t=1}^n (X_t - \mu)(X_{t+h} - \mu) - n\mathbb{E}[\tilde{\gamma}(h)] \end{bmatrix} \\ &\xrightarrow{\mathcal{D}} \mathcal{N} \left(0, n \operatorname{var} \begin{pmatrix} \tilde{\gamma}(0) \\ \tilde{\gamma}(h) \end{pmatrix} \right) \end{aligned}$$

Moreover, by Cauchy–Schwarz inequality and since $\operatorname{var}(X_t) = \sigma^2$, we have

$$\sum_{t=1}^n (X_t - \mu)(X_{t+h} - \mu) \leq \sqrt{\sum_{t=1}^n (X_t - \mu)^2 \sum_{t=1}^n (X_{t+h} - \mu)^2} < \infty.$$

Therefore, by bounded convergence theorem and (W_t) is iid, we have

$$\begin{aligned} \mathbb{E}[\tilde{\gamma}(h)] &= \frac{1}{n} \mathbb{E} \left[\sum_{t=1}^n (X_t - \mu)(X_{t+h} - \mu) \right] \\ &= \frac{1}{n} \left[\sum_{t=1}^n \mathbb{E}(X_t - \mu) \mathbb{E}(X_{t+h} - \mu) \right] = \begin{cases} \sigma^2, & \text{for } h = 0 \\ 0, & \text{for } h \neq 0 \end{cases}. \end{aligned}$$

Next, we consider the variance of $\tilde{\gamma}(h)$ when $h \neq 0$,

$$\begin{aligned} \operatorname{var}[\tilde{\gamma}(h)] &= \frac{1}{n^2} \mathbb{E} \left\{ \left[\sum_{t=1}^n (X_t - \mu)(X_{t+h} - \mu) \right]^2 \right\} \\ &= \frac{1}{n^2} \mathbb{E} \left\{ \left[\sum_{i=1}^n (X_i - \mu)(X_{i+h} - \mu) \right] \left[\sum_{j=1}^n (X_j - \mu)(X_{j+h} - \mu) \right] \right\} \\ &= \frac{1}{n^2} \mathbb{E} \left[\sum_{i=1}^n \sum_{j=1}^n (X_i - \mu)(X_{i+h} - \mu)(X_j - \mu)(X_{j+h} - \mu) \right]. \end{aligned}$$

Also by Cauchy–Schwarz inequality and the finite fourth moment assumption, we can use the bounded convergence theorem. Once again since (W_t) is white noise process, we have

$$\mathbb{E}[(X_i - \mu)(X_{i+h} - \mu)(X_j - \mu)(X_{j+h} - \mu)] \neq 0$$

only when $i = j$.

Therefore, we obtain

$$\begin{aligned} \operatorname{var}[\tilde{\gamma}(h)] &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} \left[(X_i - \mu)^2 (X_{i+h} - \mu)^2 \right] \\ &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}(X_i - \mu)^2 \mathbb{E}(X_{i+h} - \mu)^2 = \frac{1}{n} \sigma^4. \end{aligned}$$

Similarly, for $h = 0$, we have

$$\text{var}[\tilde{\gamma}(0)] = \frac{1}{n^2} \mathbb{E} \left\{ \left[\sum_{t=1}^n (X_t - \mu)^2 \right]^2 \right\} - \frac{1}{n^2} \left[\mathbb{E} \sum_{t=1}^n (X_t - \mu)^2 \right]^2 = \frac{2}{n} \sigma^4.$$

Next, we consider the covariance between $\tilde{\gamma}(0)$ and $\tilde{\gamma}(h)$, for $h \neq 0$, and we obtain

$$\begin{aligned} \text{cov}[\tilde{\gamma}(0), \tilde{\gamma}(h)] &= \mathbb{E}[\tilde{\gamma}(0) \tilde{\gamma}(h)] - \mathbb{E}[\tilde{\gamma}(0)] \mathbb{E}[\tilde{\gamma}(h)] = \mathbb{E}[\tilde{\gamma}(0) \tilde{\gamma}(h)] \\ &= \mathbb{E} \left[\left[\sum_{t=1}^n (X_t - \mu)^2 \right] \left[\sum_{t=1}^n (X_t - \mu) (X_{t+h} - \mu) \right] \right] = 0. \end{aligned}$$

Therefore by Slutsky's Theorem we have,

$$\begin{aligned} \sqrt{n} \left\{ \begin{bmatrix} \hat{\gamma}(0) \\ \hat{\gamma}(h) \end{bmatrix} - \begin{bmatrix} \sigma^2 \\ 0 \end{bmatrix} \right\} &= \sqrt{n} \left\{ \begin{bmatrix} \tilde{\gamma}(0) \\ \tilde{\gamma}(h) \end{bmatrix} - \begin{bmatrix} \sigma^2 \\ 0 \end{bmatrix} \right\} + \underbrace{\sqrt{n} \left\{ \begin{bmatrix} \hat{\gamma}(0) \\ \hat{\gamma}(h) \end{bmatrix} - \begin{bmatrix} \tilde{\gamma}(0) \\ \tilde{\gamma}(h) \end{bmatrix} \right\}}_{\xrightarrow{P} 0} \\ &\xrightarrow{D} \mathcal{N} \left(0, \begin{bmatrix} 2\sigma^4 & 0 \\ 0 & \sigma^4 \end{bmatrix} \right). \end{aligned}$$

Next, we define the function $g \left(\begin{bmatrix} a \\ b \end{bmatrix} \right) = b/a$, where $a \neq 0$. For this function it is clear that

$$\nabla g \left(\begin{bmatrix} a \\ b \end{bmatrix} \right) = \begin{bmatrix} -\frac{b}{a^2} \\ \frac{1}{a} \end{bmatrix}^T,$$

and thus using the Delta method, we have for $h \neq 0$

$$\sqrt{n} \hat{\rho}(h) = \sqrt{n} \left\{ g \left(\begin{bmatrix} \hat{\gamma}(0) \\ \hat{\gamma}(h) \end{bmatrix} \right) - \mu \right\} \xrightarrow{D} \mathcal{N}(0, \sigma_r^2),$$

where

$$\begin{aligned} \mu &= g \left(\begin{bmatrix} \sigma^2 \\ 0 \end{bmatrix} \right) = 0, \\ \sigma_r^2 &= \nabla g \left(\begin{bmatrix} \sigma^2 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 2\sigma^4 & 0 \\ 0 & \sigma^4 \end{bmatrix} \nabla g \left(\begin{bmatrix} \sigma^2 \\ 0 \end{bmatrix} \right)^T = \begin{bmatrix} 0 & \sigma^{-2} \end{bmatrix} \begin{bmatrix} 2\sigma^4 & 0 \\ 0 & \sigma^4 \end{bmatrix} \begin{bmatrix} 0 \\ \sigma^{-2} \end{bmatrix} = 1. \end{aligned}$$

Thus, we have

$$\sqrt{n} \hat{\rho}(h) \xrightarrow{D} \mathcal{N}(0, 1),$$

which concludes the proof the Theorem 1. ■

Appendix B

Robust Regression Methods

This appendix is largely based on the introduction to linear robust regression presented in ? and ?. In these references it is stated that the vast majority of the statistical models employed in different fields going from finance to biology and engineering, for example, are parametric models. Based on these models, assumptions are made concerning the properties of the variables of interest (and the models themselves) and optimal procedures are derived under these assumptions. Among these procedures, the least squares and maximum likelihood estimators are well known examples that, however, are only optimal when the underlying statistical assumptions are exactly satisfied. If the latter case does not hold, then these procedures can become considerably biased and/or inefficient when there exist small deviations from the model. The results obtained by classical procedures can therefore be misleading when applied to real data (see e.g. ? and ?).

In order to address the problems arising from violated parametric assumptions, robust statistics can be seen as an extension to classical parametric statistics by directly considering the deviations from the models. Indeed, while parametric models may be a good approximation of the true underlying situation, robust statistics does not assume that the model is exactly correct. A robust procedure as stated in ? therefore should have the following features:

- It should efficiently estimate the assumed model.
- It should be reliable and reasonably efficient under small deviations from the model (e.g. when the distribution lies in a neighborhood of the assumed model).
- Larger deviations from the model should not affect the estimation procedure excessively.

A robust estimation method is a compromise with respect to these three features. This compromise is illustrated by ? using an insurance metaphor: “sacrifice some efficiency at the model in order to insure against accidents caused by deviations from the model”.

It is often believed that robust procedures may be avoided by using the following two-step procedure:

- Clean the data using some rule for outlier rejection.
- Apply classical optimal procedures on the “clean” data.

Unfortunately such procedures cannot replace robust methods as discussed in ? for the following reasons:

- It is rarely possible to separate the two steps. For example, in a parametric regression setting, outliers are difficult to recognize without reliable (i.e. robust) estimates of the model’s parameters.
- The cleaned data will not correspond to the assumed model since there will be statistical errors of both kinds (false acceptance and false rejection). Therefore in general the classical theory is not applicable to the cleaned sample.
- Empirically, the best rejection procedures do not reach the performance of the best robust procedures. The latter are apparently superior because they can make a smoother transition between the full acceptance and full rejection of an observation using weighting procedures ?.

- Empirical studies have also shown that many of the classical rejection methods are unable to deal with multiple outliers. Indeed, it is possible that a second outlier “masks” the effect of the first so that neither are rejected.

Unfortunately the least squares estimator suffers from a dramatic lack of robustness. A single outlier can have an arbitrarily large effect on the estimated parameters. In order to assess the robustness of an estimator we first need to introduce an important concept, namely the influence function. This concept was introduced in ? and it formalizes the bias caused by one outlier. The influence function of an estimator represents the effect of an infinitesimal contamination at the point x or (\mathbf{x}, y) in the regression setting) on the estimate, standardized by the mass of contamination. Mathematically, the influence function of the estimator T for the model F is given by:

$$\text{IF}(x|T, F) = \lim_{\varepsilon \rightarrow 0} \frac{T((1 - \varepsilon)F + \varepsilon\Delta_x) - T(F)}{\varepsilon}$$

where Δ_x is a probability measure which puts mass 1 at the point x .

B.1 The Classical Least-Squares Estimator

The standard definition of the linear model is derived as follows. Let $(\mathbf{x}_i, y_i) : i = 1, \dots, n$ be a sequence of independent identically distributed random variables such that:

$$y_i = \mathbf{x}_i^T \boldsymbol{\beta} + u_i$$

where $y_i \in \mathbb{R}$ is the i -th observation, $\mathbf{x}_i \in \mathbb{R}^p$ is the i -th row of the design matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{\beta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}$ is a p -vector of unknown parameters, $u_i \in \mathbb{R}$ is the i -th error.

The least-squares estimator $\hat{\boldsymbol{\beta}}_{LS}$ of $\boldsymbol{\beta}$ can be expressed as an M -estimator¹ Least-squares estimators are an example of the larger class of M -estimators. The definition of M -estimators was motivated by robust statistics which delivered new types of M -estimators. defined by the estimating equation:

$$\sum_{i=1}^n (y_i - \mathbf{x}_i^T \boldsymbol{\beta}) \mathbf{x}_i = 0. \quad (\text{B.1})$$

This estimator is optimal under the following assumptions:

- u_i are normally distributed.
- $\mathbb{E}[u_i] = 0$ for $i = 1, \dots, n$.
- $\text{Cov}(u_1, \dots, u_n) = \sigma^2 \mathbf{I}_n$ where \mathbf{I}_n denotes the identity matrix of size n .

In other words, least-squares estimation is only optimal when the errors are normally distributed. Small departures from the normality assumption for the errors results in considerable loss of efficiency of the least-squares estimator (see ?, ? and ?).

B.2 Robust Estimators for Linear Regression Models

The “Huber estimator” introduced in ? was one of the first robust estimation methods applied to linear models. Basically, this estimator is a weighted version of the least-squares estimate with weights of the form:

¹ M -estimators are obtained as the minima of sums of functions of the data.

$$w_i = \min \left(1, \frac{c}{|r_i|} \right)$$

where r_i is the i -th residual and c is a positive constant which controls the trade-off between robustness and efficiency.

Huber proposed an M -estimator $\hat{\beta}_H$ of β defined by the estimating equation:

$$\sum_{i=1}^n \psi_c(y_i - \mathbf{x}_i^T \beta) \mathbf{x}_i = 0$$

where $\psi_c(\cdot)$ corresponds to Huber's weight function

$$w(x) = \begin{cases} 1, & \text{if } |x| \leq k \\ \frac{k}{|x|}, & \text{if } |x| > k \end{cases} \quad (\text{B.2})$$

and, thus, is defined as:

$$\psi_c(r) = \begin{cases} r \\ c \cdot \text{sign}(r). \end{cases}$$

However, the Huber estimator cannot cope with problems caused by outlying points in the design (or covariate) matrix X . An estimator which was developed to address this particular issue is the one proposed by Mallows which has the important property that the influence function is bounded also for the matrix X (see ? for more details).

B.3 Applications of Robust Estimation

Having rapidly highlighted the theory of robustness, we now focus on the application of robust techniques in practical settings. Over the next three examples, estimation between classical or traditional methods will be compared with robust methods to illustrate the usefulness of using the latter techniques can have in specific scenarios.

Example B.1. Consider a simple linear model with Gaussian errors such as

$$y_i = \alpha + \beta x_i + \varepsilon_i, \quad \varepsilon_i \stackrel{iid}{\sim} \mathcal{N}(0, \sigma^2) \quad (\text{B.3})$$

for $i = 1, \dots, n$. Next, we set the parameter values $\alpha = 5$, $\beta = 0.5$ and $\sigma^2 = 1$ in order to simulate 20 observations from the above simple linear model where we define $x_i = i$ for $i = 1, \dots, 20$. In the left panel of Figure ??, we present the simulated observations together with the fitted regression lines obtained by least-squares and a robust estimation method. It can be observed that both lines are very similar and “close” to the true model given by $y_i = 5 + 0.5i$. Indeed, although the robust estimator pays a small price in terms of efficiency compared to the the least-squares estimator, the two methods generally deliver very “similar” results when the model assumption holds. On the other hand, the robust estimators provide (in general) far more reliable results when outliers are present in a data-set. To illustrate this behavior we modify the last observation by setting $\varepsilon_{20} = -10$ (which is “extreme” under the assumption that $\varepsilon_i \stackrel{iid}{\sim} \mathcal{N}(0, 1)$). The modified observations are presented in the right panel of Figure ?? together with the fitted regression lines. In this case, the least-squares is strongly influenced by the outlier we introduced while the robust estimator remains stable and “close” to the true model.

```

# Load robust library
library("robustbase")

# Set seed for reproducibility
set.seed(867)

# Sample size
n = 20

# Model's parameters
alpha = 5      # Intercept
beta = 0.5     # Slope
sig2 = 1       # Residual variance

# Construct response variable y
x = 1:n
y = alpha + beta*x + rnorm(n,0,sqrt(sig2))

# Construct "perturbed" version of y
y.perturbed = y
y.perturbed[20] = alpha + beta*x[20] - 10

# Compute LS estimates
LS.y = lm(y ~ x)
LS.y.fit = coef(LS.y)[1] + coef(LS.y)[2]*x
LS.y.pert = lm(y.perturbed ~ x)
LS.y.pert.fit = coef(LS.y.pert)[1] + coef(LS.y.pert)[2]*x

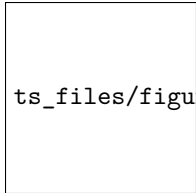
# Compute robust estimates
RR.y = lmrob(y ~ x)
RR.y.fit = coef(RR.y)[1] + coef(RR.y)[2]*x
RR.y.pert = lmrob(y.perturbed ~ x)
RR.y.pert.fit = coef(RR.y.pert)[1] + coef(RR.y.pert)[2]*x

# Define colors
gg_color_hue <- function(n, alpha = 1) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100, alpha = alpha)[1:n]
}
couleurs = gg_color_hue(6)

# Compare results based on y and y.perturbed
par(mfrow = c(1,2))
plot(NA, ylim = range(cbind(y,y.perturbed)), xlim = range(x),
     main = "Uncontaminated setting", xlab = "x", ylab = "y")
grid()
points(x,y, pch = 16, col = couleurs[4])
lines(x, alpha + beta*x, col = couleurs[3], lty = 2)
lines(x, LS.y.fit, col = couleurs[5])
lines(x, RR.y.fit, col = couleurs[1])

legend("topleft",c("Observations","Estimated model (least-squares)",
                  "Estimated model (robust)", "True model"),

```



ts_files/figure-latex/slmrobex-1.pdf

Figure B.1: Simulation Study Comparing Robust and Classical Regression Methodologies

```

lwd = c(NA,1,1,1), col = couleurs[c(4,5,1,3)],
lty = c(NA,1,1,2), bty = "n", pch = c(16,NA,NA,NA))

plot(NA, ylim = range(cbind(y,y.perturbed)), xlim = range(x),
     main = "Contaminated setting", xlab = "x", ylab = "y")
grid()
points(x[1:19],y.perturbed[1:19], pch = 16, col = couleurs[4])
points(x[20], y.perturbed[20], pch = 15, col = couleurs[6])
lines(x, alpha + beta*x, col = couleurs[3], lty = 2)
lines(x, LS.y.pert.fit, col = couleurs[5])
lines(x, RR.y.pert.fit, col = couleurs[1])

legend("topleft",c("Uncontaminated observations", "Contaminated observation",
                  "Estimated model (least-squares)", "Estimated model (robust)",
                  "True model"), lwd = c(NA,NA,1,1,1), col = couleurs[c(4,6,5,1,3)],
      lty = c(NA,NA,1,1,2), bty = "n", pch = c(16,15,NA,NA,NA))

```

Example B.2 (Robust v. Classical Simulation Study.). The next example presents a simulation study where the robust and classical techniques will be compared in order to show that the previous example was not due to a “lucky” sample favorable to the robust approach. To do so, the simulation study will generate 100 realizations from the model in the previous example and use the same robust and classical estimators to retrieve the parameters for each of the 100 iterations.

```

# Number of bootstrap replications
B = 10^3

# Initialisation
coef.LS.cont = coef.rob.cont = matrix(NA,B,2)
coef.LS.uncont = coef.rob.uncont = matrix(NA,B,2)

# Start Monte-carlo
for (j in 1:2){
  for (i in seq_len(B)) {
    # Control seed
    set.seed(2*j*B + i)

    # Uncontaminated case
    if (j == 1){
      y = alpha + beta*x + rnorm(n,0,sqrt(sig2))
      coef.LS.uncont[i,] = as.numeric(lm(y ~ x)$coef)
      coef.rob.uncont[i,] = as.numeric(lmrob(y ~ x)$coef)
    }

    # Contaminated case

```

```

if (j == 2){
  y = alpha + beta*x + rnorm(n,0,sqrt(sig2))
  y[20] = alpha + beta*x[20] - 10
  coef.LS.cont[i,] = as.numeric(lm(y ~ x)$coef)
  coef.rob.cont[i,] = as.numeric(lmrob(y ~ x)$coef)
}
}
}

# Make graph
colors = gg_color_hue(6, alpha = 0.5)
names = c("LS - uncont", "Rob - uncont", "LS - cont", "Rob - cont")

par(mfrow = c(1,2))
boxplot(coef.LS.uncont[,1], coef.rob.uncont[,1], coef.LS.cont[,1],
        coef.rob.cont[,1], main = expression(alpha),
        col = colors[c(5,1,5,1)],
        cex.main = 1.5, xaxt = "n")
axis(1, labels = FALSE)
text(x = seq_along(names), y = par("usr")[3] - 0.15, srt = 45, adj = 1,
     labels = names, xpd = TRUE)

abline(h = alpha, lwd = 2)

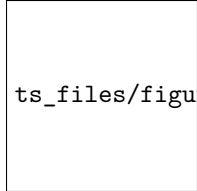
boxplot(coef.LS.uncont[,2], coef.rob.uncont[,2], coef.LS.cont[,2],
        coef.rob.cont[,2], main = expression(beta),
        col = colors[c(5,1,5,1)],
        cex.main = 1.5, xaxt = "n")
axis(1, labels = FALSE)
text(x = seq_along(names), y = par("usr")[3] - 0.015, srt = 45, adj = 1,
     labels = names, xpd = TRUE)
abline(h = beta, lwd = 2)

```

ts_files/figure-latex/lmbsrob-1.pdf

It can be seen that, as underlined earlier, that the estimations resulting from the two methods appear to be quite similar, with the robust estimator performing slightly less efficiently than the classical estimator. However, under the contaminated setting it is evident how the robust estimation procedure is not affected much by the outliers in the simulated data while the classical techniques appear to be highly biased. Therefore, if there appear to be outliers in the data, a robust estimation procedure may be considered preferable. In fact, the results of the two estimations could be compared to understand if there appears to be some deviation from the model assumptions.

Example B.3. A practical example which underlines the usefulness of robust estimation procedures is given by the dataset that contains the properties of 47 stars from the Hertzsprung-Russell diagram of the star cluster CYG OB1 in the direction of Cygnus. From the plot it can be observed that there appears to be a cluster of four stars on the upper left hand-side of the plot. The rest of the data however appears to have a reasonably good linear behavior.



ts_files/figure-latex/robstarcluster-1.pdf

Figure B.2: Comparison of Estimation Methodologies on 47 observations from the Hertzsprung-Russell diagram of the star cluster CYG OB1 in the direction of Cygnus.

```

colors = gg_color_hue(6)
data(starsCYG, package = "robustbase")
par(mfrow = c(1,1))
plot(NA, xlim = range(starsCYG$log.Te) + c(-0.1,0.1),
     ylim = range(starsCYG$log.light),
     xlab = "Temperature at the surface of the star (log)",
     ylab = "Light intensity (log)")
grid()
points(starsCYG, col = colors[4], pch = 16)

LS = lm(starsCYG$log.light ~ starsCYG$log.Te)
rob = lmrob(starsCYG$log.light ~ starsCYG$log.Te)
x = seq(from = min(starsCYG$log.Te)-0.3,
        to = max(starsCYG$log.Te)+0.3, length.out = 4)
fit.LS = coef(LS)[1] + x*coef(LS)[2]
fit.rob = coef(rob)[1] + x*coef(rob)[2]

lines(x, fit.LS, col = colors[5])
lines(x, fit.rob, col = colors[1])

legend("topright",c("Observations","Estimated model (least-squares)",
                    "Estimated model (robust)"), lwd = c(NA,1,1),
       col = colors[c(4,5,1)], lty = c(NA,1,1),
       bty = "n", pch = c(16,NA,NA))

```

From Figure ?? it can be seen that classical linear regression is considerably affected by the cloud of apparent outliers on the left hand side of the plot. As a result, the regression line has a negative slope when the majority of the data appears to have a clear positive linear trend. The latter is however the case when using a robust approach which indeed detects a positive linear trend. Having said this, there are in fact two distinct populations in the data consisting in “giants” (upper left corner) and “main sequencers” (right handside). Thus, when observing data in general it is always necessary to understand if we are dealing with real outliers (unexplained outlying data) or simply with data that can be explained by other factors (e.g. a different population as in this example).

```

summary(LS)

##
## Call:
## lm(formula = starsCYG$log.light ~ starsCYG$log.Te)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.1052 -0.5067  0.1327  0.4423  0.9390

```

```
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.7935      1.2365   5.494 1.75e-06 ***
## starsCYG$log.Te -0.4133      0.2863  -1.444   0.156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5646 on 45 degrees of freedom
## Multiple R-squared:  0.04427,    Adjusted R-squared:  0.02304
## F-statistic: 2.085 on 1 and 45 DF,  p-value: 0.1557

summary(rob)

##
## Call:
## lmrob(formula = starsCYG$log.light ~ starsCYG$log.Te)
## \--> method = "MM"
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.80959 -0.28838  0.00282  0.36668  3.39585
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -4.9694      3.4100  -1.457  0.15198
## starsCYG$log.Te  2.2532      0.7691   2.930  0.00531 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Robust residual standard error: 0.4715
## Multiple R-squared:  0.3737, Adjusted R-squared:  0.3598
## Convergence in 15 IRWLS iterations
##
## Robustness weights:
## 4 observations c(11,20,30,34) are outliers with |weight| = 0 ( < 0.0021);
## 4 weights are ~= 1. The remaining 39 ones are summarized as
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.6533 0.9171 0.9593 0.9318 0.9848 0.9986
## Algorithmic parameters:
##      tuning.chi          bb      tuning.psi      refine.tol
##      1.548e+00      5.000e-01      4.685e+00      1.000e-07
##      rel.tol      scale.tol      solve.tol      eps.outlier
##      1.000e-07      1.000e-10      1.000e-07      2.128e-03
##      eps.x warn.limit.reject warn.limit.meanrw
##      8.404e-12      5.000e-01      5.000e-01
##      nResample      max.it      best.r.s      k.fast.s      k.max
##      500          50          2          1          200
##      maxit.scale      trace.lev      mts      compute.rd fast.s.large.n
##      200          0          1000          0          2000
##      psi      subsampling      cov
##      "bisquare"      "nonsingular"      ".vcov.avar1"
## compute.outlier.stats
##      "SM"
## seed : int(0)
```

Appendix C

Proofs

C.1 Proof of Theorem ??

Consider:

$$(X_t - m)^2 = [(X_t - E[X_t|\Omega_t]) + (E[X_t|\Omega_t] - m)]^2$$

where $m = m(X_1, \dots, X_t)$ and $\Omega_t = (X_1, \dots, X_t)$.

Therefore we can write

$$(X_t - m)^2 = (X_t - E[X_t|\Omega_t])^2 + (E[X_t|\Omega_t] - m)^2 + 2(X_t - E[X_t|\Omega_t])(E[X_t|\Omega_t] - m)$$

Focusing on only the last term (and dropping the constant 2), we have that

$$\underbrace{(X_t - E[X_t|\Omega_t])}_{=\varepsilon_t} (E[X_t|\Omega_t] - m).$$

At this point, let us study the value of $E[\varepsilon_t|\Omega_t]$ (the reason for this will become apparent in the next steps of the proof):

$$E[\varepsilon_t|\Omega_t] = E[X_t - E[X_t|\Omega_t]|\Omega_t] = E[X_t|\Omega_t] - E[X_t|\Omega_t] = 0$$

Given this, we now consider the law of total expectation (i.e. $E[X] = E[E[X|Y]]$) which allows us to rewrite the expectation of the last term as follows

$$E[\varepsilon_t(E[X_t|\Omega_t] - m)] = E[E[\varepsilon_t(E[X_t|\Omega_t] - m)|\Omega_t]] = E\left[\underbrace{E[\varepsilon_t|\Omega_t]}_{=0}(E[X_t|\Omega_t] - m)\right] = 0$$

Since we have shown that the expectation of the last term is zero, we have that

$$(X_t - m)^2 = (X_t - E[X_t|\Omega_t])^2 + (E[X_t|\Omega_t] - m)^2.$$

Now the first term is positive and doesn't depend on m , so we focus on the second term which is minimized for $m = E[X_t|\Omega_t]$ thereby minimizing the entire expression in terms of m . ■