INTRODUCTION TO FINANCIAL ECONOMETRICS

SEBASTIANO MANZAN

Zicklin School of Business, Baruch College

This version: September 4, 2017

Contents

Chapter 1

Introduction

Financial Econometrics can be broadly defined as the area of statistics and econometrics devoted to the analysis of financial data. The goal of this book is to introduce you to the quantitative analysis of financial data in a *learning by doing* approach. Instrumental to achieve this is the use of the `R` programming language which is widely used in academia and the industry for data analysis. `R` is gaining popularity in economics and finance for its many advantages over alternative sofware packages. It is open source, available in all platforms, and it is supported by a wide community of users that contribute packages, discussions, and blogs. In addition, it represents one of the most popular software for data analytics which is becoming an increasingly important field in business as well as in finance.

Before we get started with the statistical methods and models, let's quickly consider a few examples of financial data. Financial markets around the world produce everyday billions of observations on, e.g., asset prices, shares transacted, and quotes. Every quarter thousands of companies in the U.S. and abroad provide accounting information that are used by investors to assess the profitability and the prospects of their investments. Exchange rates and commodities are also transacted in financial markets and their behavior determine the price of thousand of products that we consume every day. Let's consider a few example of financial data that we will use througout the book and start asking questions that we want the data to answer.

1.1 Financial Data

Figure 1.1 shows the a popular U.S. stock market index, the S&P 500 Index, divided by the smoothed earnings and dividends from 1871 until 2016. The annual data start in 1871 and it is provided by Professor Robert Shiller from Yale University. The Price-to-Earnings (PE) and Price-to-Dividends (PD) ratios are considered by investors as valuation ratios since they reflect the dollar amount that you pay for the asset for each dollar of earnings or dividends that the asset provides. A plot that shows a variable (e.g., PE or PD ratio) over time is called a *time series* graph and our goal is to understand the fluctuations over time of the variable.

The valuation ratios for the S&P 500 Index are considered measures of the valuation of the whole US equity market and are used to assess the opportunity to invest in the U.S market. The time series graphs

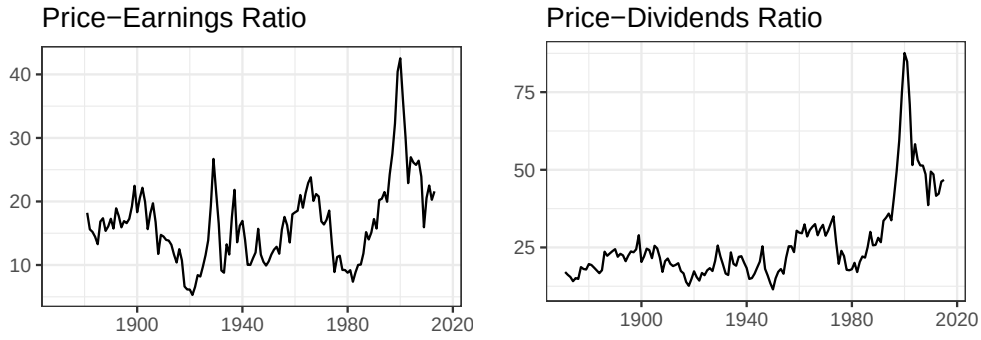


Figure 1.1: Annual Price-to-Earnings and Price-to-Dividends ratio for the Standard and Poors 500 Index starting in 1871.

show that the PE has historically fluctuated between 5 and 25 with the exception of the late 1990s when the Index skyrocketed to over 40 times smoothed earnings. The range of variation for the PD ratio up to 1990 has been between 11.5 and 30, but at the end of the 1990s it reached a record valuation of 87, only to subsequently fall back around 40 in the last part of the sample. The PE and PD ratios fluctuate a lot and there are several questions that we would like to find answers to:

- Why are market valuations fluctuating so much?
- Since these are ratios, is it the price in the numerator to make the ratio adjust to its historical values or earnings/dividends (denominator)?
- Are these fluctuations driven by the tendency of economies to have cycles of expansions and recessions?
- Why smart investors did not learn from such a long history to sell at (or close to) the valuation peak and buy at the bottom?
- What explains the extreme valuations in the late 1990s?
- The most recent valuations are still high relative to historical standards. Is that an indication that it should decline further?

These questions are not only relevant for financial institutions managing large portfolios, but also for small investors that are saving for retirement or for college.

The previous discussion considered the annual S&P 500 valuation ratios for over 100 years by sampling only the last price of the year. However, financial markets trade approximately 250 days a year and for every day we observe a closing price. The flow of news about the economy and company events contributes to the daily fluctuations of asset prices. Figure 1.2 shows the daily price of the S&P 500 Index from January 02, 1985 to September 01, 2017 on the left panel, while the right panel shows the daily return of the Index. The return represents the percentage change of the price in a day relative to the price of the previous day. The value of the Index at the beginning of 1985 was 165 and grew to 2477 in September 2017. There have been two severe downturns that are apparent in this graph: the correction in 2000 after the rapid growth of the 1990s and a second one that occured in 2008 during the great recession of 2008-2009. The graph on the right-hand side shows that the daily returns experience sometimes large changes, such as the one-day drop in October 19, 1987 of 22.9% and several instances

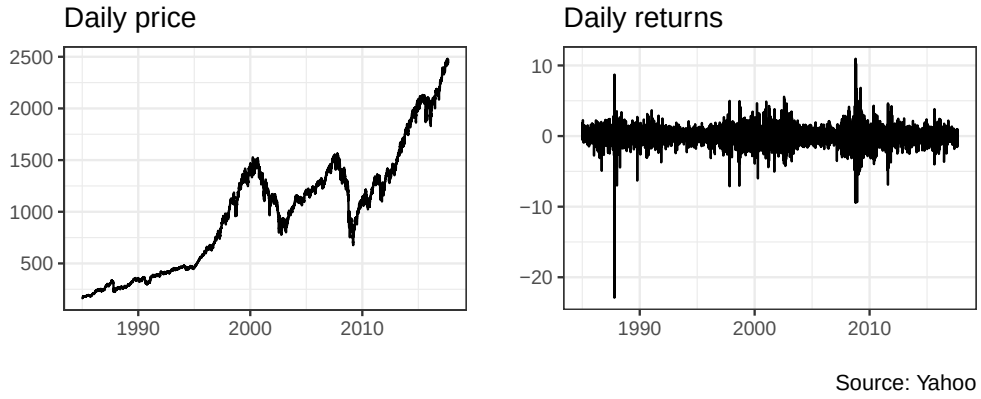


Figure 1.2: Daily prices and returns for the Standard and Poor 500 Index.

of changes as large as positive/negative 10% in one day. In addition, it seems that there are periods in which markets are relatively calm and fluctuate within a small range, while in other turbulent times the Index has large variation.

At the daily frequency, rather than actual dividends and earnings that are released at the quarterly frequency, it is news about these variables that make prices fluctuate. Several questions that we can try to answer using data at the daily frequency:

- What forces determine the boom-bust dynamics of asset prices?
- Why do we have these clusters of calm and turbulent times rather than having "normal" times with returns fluctuating on a constant range?
- What is the most likely value of the S&P 500 in 10 years from now?
- Are returns predictable?
- Is volatility, defined as the dispersion of returns around their mean, predictable?

Financial markets not only produce one closing price a day, but they produce also thousands of quotes and trades for each asset every day. Figure 1.3 shows the midpoint between the bid and ask price of the US Dollar (USD) and Japanese Yen (JPY) exchange rate in the last trading day of 2016. The picture shows the exchange rate at the 1 minute interval, but within each interval there are several quotes that are produced depending on the time of the day. Armed with these type of data, we can answer different type of questions:

- What factors determine the difference between the price at which you can buy or sell an asset, typically called the bid-ask spread?
- Is the spread constant over time or subject to fluctuations due to market events?
- Is it possible to use intra-day information to construct measures of volatility?
- Does the size of a trade have an impact on the price?
- Can we predict the direction of the next trade and the price change?

Working with high-frequency data is challenging due to large amount of quotes and transactions that are produced every day for thousands of assets. For example, the time series in Figure 1.3 represents a subsample of 1,320 1-minute quotes from the 31,076 available for the month of December 2016. The 1-minute quotes are obtained by taking the last quote in each 1 minute interval of weekdays of the month from a total sample of 14,237,744 quotes. The large number of observations makes tools like spreadsheets

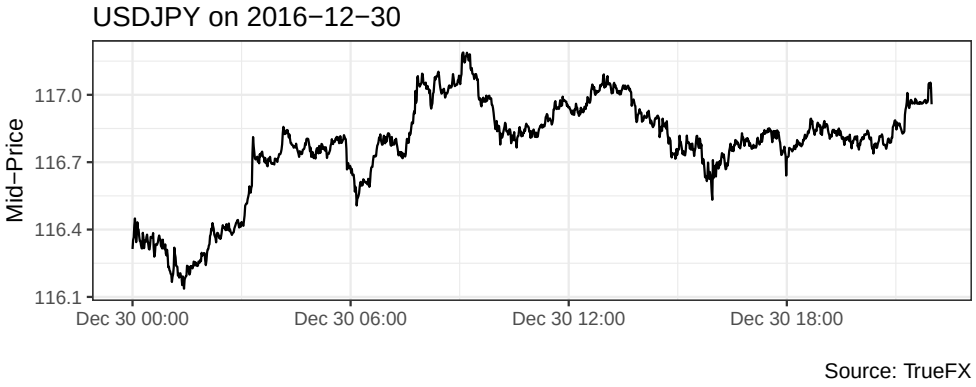


Figure 1.3: Intra-day mid-point between bid and ask price for the USD to JPY exchange rate sampled at the 1 minute frequency.

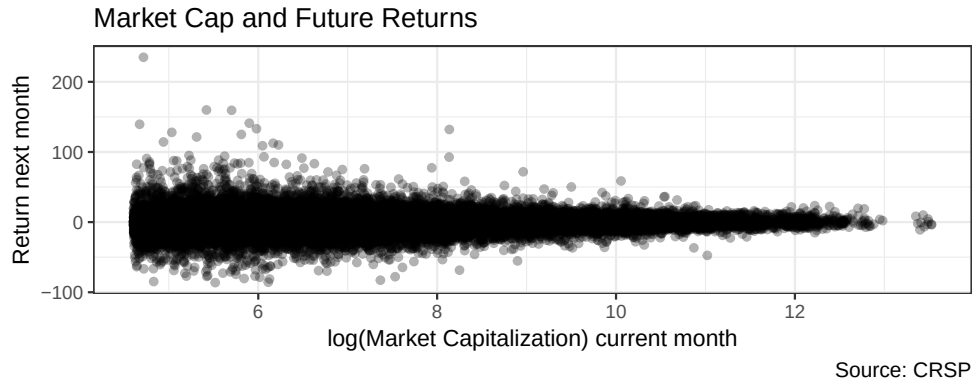


Figure 1.4: Scatter plot of the logarithm of the market capitalization for a stock listed in the NYSE, AMEX, or NASDAQ against the percentage return in the following month. Each point represents a stock-month pair in 2015. Stocks with market cap below 100 million dollars are dropped from the sample.

hardly useful to analyze high-frequency data. First, Microsoft Excel has a limit of rows of slighly more than one million. This means that we would not be able to load even one-month of quotes for the USDJPY exchange rate. Second, simple manipulations such as subsampling and aggregating the dataset to a lower frequency in a spreadsheet can be challenging. Thus, `R` offers a very interesting proposition by offering a flexible and powerful tool that is not afraid of (moderately) large dataset and allows to perform simple operation in few lines of code. In addition, for almost any task there is a dedicated package that provides ad-hoc functions.

These examples refer to *time series* that consists of the observations for a variable (e.g., the S&P 500 Index) over time. In the previous cases the time frequencies are one year, one day, and one minute and with each frequency we discussed different issues and different questions we want to find answers. When dealing with time series the objective of the analysis is to explain the factors driving the variation over time of the variable. Instead, there are other settings in which the goal of the analysis is to understand the relationship between different variables across different units (e.g., stocks) rather than over time. An

example is provided in Figure 1.4 that shows the scatter plot of the logarithm of the market capitalization of all the stocks listed in the NYSE, AMEX, and NASDAQ in the months of 2015 against the monthly return in the following month. Many questions arise here:

- Do stocks of large (small) capitalization companies outperform in the following month stocks of small (large) caps? Is size a predictor of future returns?
- In addition to size, what other company characteristics can be used to predict future stock performance?
- Are small caps "*riskier*" relative to large caps?
- Small caps stocks provide (on average) higher returns relative to large cap stocks: what are the factors explaining it? why?

In this example the data is called *cross-sectional* or *longitudinal* in the sense that the goal is to understand the connection between two (or more) variables (e.g., size and future returns) for the same stock.

1.2 Data sources

There are several sources of financial data that, in some cases, are publicly available, while in others are subscription-based. In this book we will use publicly available data when possible, and commercial datasets otherwise. A short-list of data providers is:

- http://finance.yahoo.com/:
  - The *Yahoo Finance* website allows to download daily/weekly/monthly data for individual stocks, indices, mutual funds, and ETFs. The data provided is the opening and closing daily price, the highest and lowest intra-day price, and the volume. In addition, the website provides also the adjusted closing price that is adjusted for dividend payments and stock splits. The data for Figure 1.2 above was obtained from this website.
  - The data can be downloaded as a `csv` file under the *historical prices* tab
  - There are several packages in `R` that allow to download the data directly without having to save the file. It requires to know the ticker of the asset.
- http://www.truefx.com:
  - *TrueFX* is a fintech startup that provides financial data and services to traders. Upon registration to their website, it is possible to download quotes data for many currency pairs in monthly files starting from 2009.
- https://fred.stlouisfed.org/:
  - the *Federal Reserve Economic Database* (*FRED*) is a very extensive database of economic and financial data for both the United States and other countries.
  - Similarly to Yahoo Finance, data can be downloaded in `csv` format or can be downloaded directly using `R`.
- http://www.quandl.com:
  - *QUANDL* is an aggregator of economic and financial datasets (from the Chicago Mercantile Exchange to FRED to many others)

The ones listed above are the most popular sources of economic and financial data that are publicly available. In addition, there are several subscription-based data providers that offer datasets in other These databases are typically available through the library:

- *CRSP*: the *Center for Research in Security Prices* (*CRSP*) at the University of Chicago provide stock prices starting from 1926
- *Compustat*: provides accounting information for stocks listed in the United States
- *TAQ*: *Trade And Quote* provides tick-by-tick data for all stocks listed in the NYSE, NASDAQ, and AMEX.

1.3 The plan

This book has two main objectives. The first is to introduce the reader to the most important methods and models that are used to analyze financial data. In particular, the discussion focuses on three main areas:

- *Regression Model*: the regression model has found extensive application in finance, in particular in investment analysis; we will review the basic aspects and assumptions of the model and we will apply it to measure the risk of asset portfolios.
- *Time Series Models*: the characteristic of time series models is that they exclusively rely on the past values of a variable to predict its future; they are very convenient tools when dealing with higher-frequency data since predictor variables might not be observable, hard to measure, or simply too noisy to be useful.
- *Volatility Models*: volatility models are time series models that are used to forecast the variance or the standard deviation of asset returns. The assumption underlying these models is that risk varies over time (see Figure 1.2) and this should be accounted for by measures of risk such as the standard deviation of returns. Since risk is an essential component of financial decisions, these models have found widespread application, in particular with the advent of risk management practice in financial institutions.

The goal of the book is to give you a hands-on understanding of these models and their relevance in the analysis of financial data. However, lots of things can go wrong when statistical tools are used "mechanically". Put it very simply, our goal is to extract useful information from a large amount of data and there are many issues that might distort our analysis. Analyzing data is not only concerned with producing numbers (a more elegant term would be *estimate*) but, more importantly, being aware of the potential pitfalls and possible remedies to make our analysis trustworthy.

The second objective of the book is to introduce the reader to the `R` programming language. `R` is becoming an increasingly useful tool for data analysis in many different fields and a fierce competitor of commercial statistical and econometric packages. The discussion of the techniques and models is closely integrated with the implementation in `R` so that all tables and graphs in the book can be reproduced by the reader. It is probably uncontroversial to say that in the era of "big data" `R` is a top-contender to replace the long-serving "spreadsheets" that is nearing the end of their usable life. `R` puts your capabilities to analyze data in a different level relative to what you can achieve using a spreadsheet or menu-driven software. Also, mastering `R` is a skill that can be useful if you decide that your destiny is to do marketing analytics rather than analyzing financial data.

Readings

Some readings that you might be interested if you want to find out more about the discussion in this chapter:

- Engle (1982)
- Shiller (2015)

Exercises

The goal of the empirical exercises in this Chapter is to review the important concepts from probability, statistics, finance, and economics and all involve data analysis. For this exercises you are allowed to use a spreadsheet, while from the next chapter we of `R`.

1. Go to Yahoo Finance and download historical data for tickers `SPY` (SPDR S&P 500 ETF) from January 1995 until the most recent at the daily frequency.
    - Sort the data by having the date from oldest to newest
    - Plot the time series of the price (x-axis is time and y-axis the price in $)
    - Plot the time series of the logarithm of the price. What are the advantages/disadvantages of plotting the logarithm of the price rather than the price?
    - Calculate the return as the percentage change of the price in day t relative to day $t-1$
    - Plot the return time series and discuss the evidence of clusters of high and low volatility
    - Calculate the average, standard deviation, skewness and kurtosis of the asset return and discuss the results
    - Calculate the fraction of observations in the sample that are smaller than -5% and larger than 5%
    - Assume that the daily returns are normally distributed with mean equal to the sample average and variance set at the sample variance; calculate the probability that the returns is smaller than -5% or larger than 5% and compare these values with the fractions calculated in the previous point
    - Calculate the sample quantile at 1% and 99% of the asset returns
    - Assuming normality, calculate the quantiles of the normal distribution with mean and variance equal to their sample estimates. Compare the values to the one obtained in the previous question
2. Download from Yahoo Finance historical data for `SPY` (S&P 500 Index), `WMT` (Walmart), and `AAPL` (Apple) at the daily frequency from January 1995 until the most recent available day.
    - Sort the data from the oldest date to the newest date; make sure that in each row the price of the three assets refers to the same date
    - Calculate the daily returns as the percentage change of the price in day t relative to the previous day $t-1$ for the three assets
    - Estimate a linear regression model in which the return of `WMT` and `AAPL` are the dependent variable and the return of `SPY` is the independent variable; provide an interpretation of the estimated coefficients

- Test the null hypothesis that the intercept and the coefficient of the market are equal to zero at 5% significance level
- Comparing the two stocks, which one is "riskier"?

3. Visit the page of Robert Shiller at Yale and download the file chapt26.xlsx
   - Do a time series plot of the nominal S&P 500 Index (column `P`)
   - Calculate the average annual growth rate of the Index
   - Plot the real price (column `RealP`) over time, where the real value is obtained by dividing the nominal by the CPI Index
   - Calculate the average annual real growth rate of the Index
   - The equity premium is defined as the difference between the average real return of investing in the equity market (we use the S&P 500 to proxy for this) and the real interest rate (column `RealR`). How large is the equity premium in this sample? Calculate the equity premium including and excluding data from 1990: is the magnitude of the premium significantly different? What explanation can be offered in case they are different?
   - Calculate the annual percentage change of the Index (`R`), the dividends (`D`), and earnings (`E`). Which of the three variables in more volatile?
   - The columns `P*`, `P*r` and `P*C` are measures of the fundamental value of the real S&P 500 Index that are obtained by discounting future cash flows. Plot the three measures of fundamental value and the real Index (`RealP`). Are the fundamental values tracking closely the real price? If not, what could explain the deviations of the Index price from its fundamental value?
4. Create a free account at TrueFX.com and go to the download area. TrueFX gives you access to tick-by-tick quote prices for a wide range of currencies and the data are organized by years (starting in 2009) and pairs. Choose a month, year and currency pair and download the file. Unzip it to a location in your hard drive and answer the following questions:
   - Open the `csv` file in a spreadsheet; your options are:
     a. Microsoft Excel has a limit of 1,048,576 rows which is most likely not enough for the file at hand; you will get an alert from Excel that the file was not loaded completely but still will load the first 1,048,576 rows
     b. OpenOffice Calc has a limit of approx 65,000 rows
     c. Google Docs spreadsheet has a limit of 200,000 cells; since the TrueFx files have 4 columns, you can only load up to 50,000 rows
   - The file contains 4 columns: the currency pair, date and time, bid price, ask price. Assign headers to the columns (you might have to drop one row to do that).
   - What is the first and last date in your sample (that has been truncated due to "row limitations")?
   - Create a mid-point column that is calculated as the average of the bid and ask price
   - Plot the time series of the mid-price (x-axis is the date and y-axis the mid-price)
   - Calculate the bid-ask spread which is defined as the difference between the bid and ask prices.
   - Calculate the mean, min, and max of the bid-ask spread. Plot the spread over time and evaluate whether it is varying over time.

Chapter 2

Getting Started with R

The aim of this chapter is to get you started with the basic tasks of data analysis using R. I will assume that you have read a *Getting started with R* chapter such as in Albert and Rizzo (2012) and Zuur et al. (2009) or completed an online *R 101* course at, for example, Big Data University or Datacamp. The starting point of this chapter is how to load data in R and we will discuss two ways of doing this: loading a data file stored in your computer and downloading the data from the internet. We will introduce R-terminology as needed in the discussion. Once we have loaded data in R it is time to explore the dataset by viewing the data, summarize it with statistical quantities (e.g., the mean), and plotting the variables and their distribution.

The way R works is as follows:

- **Object**: R stores information (i.e., number or string values) in objects
- **Structure**: these objects organize information in a different way:
  - *data frames* are tables with each column representing a variable and each row denoting an observational unit (e.g., a stock, a state, or a month); each column can be of a different type, for example a character string representing an address or a numerical value.
  - *matrices* are similar to data frames with the important difference that all variables must be of the same type (i.e., all numerical values or characters)
  - *lists* can be considered an object of objects in the sense that each element of a list could be a data frame or a matrix (or a list)
- **Function**: is a set of operations that are performed on an object; for example, the function `mean()` calculates the average of numerical variable by summing all the values and dividing by the number of elements.
- **Package**: is a set of functions that are used to perform certain tasks (e.g., load data, estimate models etc)

One of the most important functions that you will use in R is `help()`: by typing the name of a function within the brackets, R pulls detailed information about the function arguments and its expected outcome. With hundred of functions used in a typical R session and each having several arguments, even the most experienced R programmers ask for `help()`.

2.1 Reading a local data file

The file `List_SP500.csv` is a text file in *comma separated values* (csv) format that contains information about the 500 companies that are included in the S&P 500 Index[1]. The code below shows how to load a text file called `List_SP500.csv` using the `read.csv()` function. The output of reading the file is then assigned to the `splist` object that is stored in the `R` memory. Anytime we type `splist` in the command line, `R` will pull the data contained in the object. For example, the second line of code uses the `head(splist, 10)` function to print the first 10 rows of the `splist` object.

```
splist <- read.csv("List_SP500.csv")
head(splist,10)
```

```
   Ticker.symbol                  Security  Address.of.Headquarters Date.first.added
1            MMM                 3M Company      St. Paul, Minnesota
2            ABT        Abbott Laboratories  North Chicago, Illinois       1964-03-31
3           ABBV                AbbVie Inc.  North Chicago, Illinois       2012-12-31
4            ACN              Accenture plc          Dublin, Ireland       2011-07-06
5           ATVI        Activision Blizzard Santa Monica, California       2015-08-31
6            AYI          Acuity Brands Inc         Atlanta, Georgia       2016-05-03
7           ADBE          Adobe Systems Inc     San Jose, California       1997-05-05
8            AMD Advanced Micro Devices Inc    Sunnyvale, California       2017-03-20
9            AAP         Advance Auto Parts        Roanoke, Virginia       2015-07-09
10           AES                   AES Corp      Arlington, Virginia
```

The object `splist` represents a **data frame** in `R` terminology: a table in which each column represents a variable and each row is a different unit (in this case companies listed in the S&P 500 Index). The `splist` data frame has 4 columns and 505 rows[2]. The size of the data frame can be found using the `dim(splist)` command which provides the dimension of the frame, or using `ncol(splist)` and `nrow(splist)`. The commands `head()`, `dim()`, `ncol()` and `nrow()` are functions that execute a series of operations on a data objects. The function inputs are referred to as **arguments**, that, except for the data object, are typically set to default values (in the case of `head()` the default for argument `n` is 6). A useful command to obtain the properties of the columns that have been imported is `str()`:

```
str(splist)
```

```
'data.frame':   505 obs. of  4 variables:
 $ Ticker.symbol          : Factor w/ 505 levels "A","AAL","AAP",..: 314 7 5 8 51 57 9 33 3 17 ...
 $ Security               : Factor w/ 505 levels "3M Company","A.O. Smith Corp",..: 1 3 4 5 6 7 8 10 9 11 ...
 $ Address.of.Headquarters: Factor w/ 258 levels "Akron, Ohio",..: 224 161 161 66 212 8 206 228 197 5 ...
 $ Date.first.added       : Factor w/ 302 levels "","1964-03-31",..: 1 2 214 196 254 275 79 294 251 1 ...
```

The structure of the object provides the following information:

- the object is a `data.frame`
- number of observations (505) and variables (4)
- name of each variable included in the data frame (`Ticker.symbol, Security, Address.of.Headquarters, Date.first.added`)
- type of the variable (factor)

[1] The Table is obtained from this Wikipedia page. Accessed on September 04, 2017.
[2] Exercise: why do we have 505 rows if the Index is composed of 500 companies? Find out.

The **type** refers to the type of data represented by each variable and defines the operations that R can do on the variable. For example, if a numerical variable is mistakenly defined as a string of characters than R will not be able to perform mathematical operations on such variable and produce an error message. The types available in R are:

- `numeric`: (or `double`) is used for decimal values
- `integer`: for integer values
- `character`: for strings of characters
- `Date`: for dates
- `factor`: represents a type of variable (either numeric, integer, or character) that categorizes the values in a small (relative to the sample size) set of categories (or `levels`)

The structure of the dataset shows that all variables have been imported as `factor`, although this does not seem appropriate for these variables. The `factor` type should be reserved to variables that have a small number of categories. Instead, for this dataset we have that the `Ticker.symbol` variable has 505 levels out of 505 observations and `Security` has 505 levels. For the address and date variables the number of categories is smaller but still quite large to be considered a factor. In particular, the variable `Date.first.added` should be defined as a date[3] rather than `factor` or `character`.

The reason for this *missclassification* is that the base function `read.csv()` is set by default to classify as `factor` any variable that is considered a string. However, this can be prevented by setting the argument `stringsAsFactors = FALSE` this can be prevented and the result is reported below:

```
splist <- read.csv("List_SP500.csv", stringsAsFactors = FALSE)
str(splist)
```

```
'data.frame':   505 obs. of  4 variables:
 $ Ticker.symbol          : chr  "MMM" "ABT" "ABBV" "ACN" ...
 $ Security               : chr  "3M Company" "Abbott Laboratories" "AbbVie Inc." "Accenture plc" ...
 $ Address.of.Headquarters: chr  "St. Paul, Minnesota" "North Chicago, Illinois" "North Chicago, Illinois" "Dublin, Ireland" ...
 $ Date.first.added       : chr  "" "1964-03-31" "2012-12-31" "2011-07-06" ...
```

Adding the argument has the effect of reading all variables as `character`, including the date when the stock was first added to the Index. We can redefine the `type` of the variable by setting the variable `Date.first.added` in the `splist` object to be a date as done below:

```
splist$Date.first.added <- as.Date(splist$Date.first.added, format="%Y-%m-%d")
class(splist$Date.first.added)
```

```
[1] "Date"
```

Notice that:

- `$` sign is used to extract a variable/column in a data frame; `splist$Date.first.added` extract the `Date.first.added` from the data frame object `splist`
- `as.Date()` function converts the `splist$Date.first.added` from `character` to `Date`; the role of the argument `format="%Y-%m-%d"` is to specify the format of the date being defined[4]
- `class()` is a function used to obtain the class type of an object (or the variable in a data frame as in this case).

[3] One reason for defining dates is that it allows to define objects as time series and use many specialized functions available in R.

[4] See `help(as.Date)` for more details on this and later in the book.

Although the `csv` format is probably the most popular one for exchanging text files, there are other formats and dedicated functions in `R` to read these files. For example, `read.delim()` and `read.table()` are two functions that can be used for tab-delimited or space-delimited files.

An alternative to using the base function is to employ functions from other packages that presumably improve some aspects of the default function. `readr` is a package that provides smarter parsing of the variable and save the user some time in redefining variable. Below we apply the function `read_csv()` from the package `readr` to import the `List_SP500.csv` file:

```
library(readr)
splist <- read_csv("List_SP500.csv")
str(splist, max.level=1)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   505 obs. of  4 variables:
 $ Ticker symbol          : chr  "MMM" "ABT" "ABBV" "ACN" ...
 $ Security               : chr  "3M Company" "Abbott Laboratories" "AbbVie Inc." "Accenture plc" ...
 $ Address of Headquarters: chr  "St. Paul, Minnesota" "North Chicago, Illinois" "North Chicago, Illinois" "Dublin, Ireland" ...
 $ Date first added       : Date, format: NA "1964-03-31" "2012-12-31" "2011-07-06" ...
 - attr(*, "spec")=List of 2
  ..- attr(*, "class")= chr "col_spec"
```

Some remarks:

- before being able to use functions from a package, it needs to be installed in your local machine. `R` comes with a few base packages while the more specialized packages have to be installed by the user. This is done with the command `install.packages(readr)` that needs to be performed only once. After the installation, `library(readr)` or `require(readr)` are the commands used to pull the package that make all the functions in the packages available in the `R` environment. If you call a function from a package that is not loaded, `R` issues an error message that the function is not known.
- the function `read_csv()` correctly classifies the variable `Date.first.added` as a date.
- the other variables are defined as `character` instead of `factor`

The `readr` package produces an object that is a data frame of class `tbl_df`. Although the object is still a data frame, the class provides some additional capabilities that will be discussed in more detail in later chapters.

Let's consider another example. In the previous Chapter we discussed the S&P 500 Index with the data obtained from Yahoo Finance. The dataset was downloaded and saved with name `GSPC.csv` to a location in the hard drive. We can import the file in `R` using the `read.csv()` function:

```
index <- read.csv("GSPC.csv")
tail(index, 8)
```

```
          Date GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
8230 2017-08-23    2444.9    2448.9   2441.4     2444.0  2785290000        2444.0
8231 2017-08-24    2447.9    2450.4   2436.2     2439.0  2846590000        2439.0
8232 2017-08-25    2444.7    2454.0   2442.2     2443.1  2588780000        2443.1
8233 2017-08-28    2447.3    2449.1   2439.0     2444.2  2677700000        2444.2
8234 2017-08-29    2431.9    2449.2   2428.2     2446.3  2737580000        2446.3
8235 2017-08-30    2446.1    2460.3   2443.8     2457.6  2633660000        2457.6
8236 2017-08-31    2462.7    2475.0   2462.7     2471.6  3348110000        2471.6
8237 2017-09-01    2474.4    2480.4   2473.8     2476.6  2710730000        2476.6
```

where the `tail()` command is used to print the bottom 8 rows of the data frame. Files downloaded from Yahoo have, in addition to the date, 6 columns that represent the price at the open of the trading day, highest and lowest price of the day, the closing price, the volume transacted, and the adjusted closing price[5]. The structure of the object is provided here:

```
str(index)
```

```
'data.frame':   8237 obs. of  7 variables:
 $ Date         : Factor w/ 8237 levels "1985-01-02","1985-01-03",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ GSPC.Open    : num  167 165 165 164 164 ...
 $ GSPC.High    : num  167 166 165 165 165 ...
 $ GSPC.Low     : num  165 164 163 164 164 ...
 $ GSPC.Close   : num  165 165 164 164 164 ...
 $ GSPC.Volume  : num  67820000 88880000 77480000 86190000 92110000 ...
 $ GSPC.Adjusted: num  165 165 164 164 164 ...
```

In this case all variables are classified as numerical except for the `Date` variable that is read as a string and thus assigned the `factor` type. This can be solved by adding the argument `stringAsFactors = FALSE` and then define the variable using `as.Date`. Alternatively, the file can be read using `read_csv()` that is able to identify that the first column is a date.

2.2 Saving data files

In addition to reading files, we can also save (or write in `R` language) data files to the local drive. This is done with the `write.csv()` function that outputs a `csv` file of the data frame or matrix provided. The example below adds a column to the `index` data frame called `Range` which represents the intra-day percentage difference between the highest and lowest intra-day price. The frame is then saved with name `newdf.csv`.

```
index      <- read_csv("GSPC.csv")
index$Date <- as.Date(index$Date, format="%Y-%m-%d")
# create a new variable called range
index$Range <- 100 * (index$GSPC.High - index$GSPC.Low) / index$GSPC.Low
write.csv(index, file = "indexdf.csv", row.names = FALSE)
```

2.3 Time series objects

Time series data and models are widely used and there are several packages in `R` that provide an infrastructure to manage this type of data. The three most important packages are `ts`, `zoo`, and `xts` that provide functions to define time series objects, that is, objects in which variables are ordered in time. Since in this book we will mostly working with the `xts` package, we illustrate its main properties here and discuss further topics in later chapters.

The `index` object that was used above is currently a data frame and we would like to define it as a time series object starting in 1985-01-02 and ending in 2017-09-01. Once we define `index` to be a time series object, each column of the data frame will have implicitly defined the dates and properties related its

[5]The closing price adjusted for stock splits and dividends.

time series status. An example of when this becomes useful is when plotting the variables since the x-axis is automatically set to represent time.

The code below defines the time series object `index.xts` that is obtained by the `index` object as a `xts` object with dates equal to the `Date` column:

```
library(xts)
index.xts <- xts(subset(index, select=-Date), order.by=index$Date)
head(index.xts)
```

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted   Range
1985-01-02    167.20    167.20   165.19     165.37    67820000        165.37 1.21678
1985-01-03    165.37    166.11   164.38     164.57    88880000        164.57 1.05244
1985-01-04    164.55    164.55   163.36     163.68    77480000        163.68 0.72845
1985-01-07    163.68    164.71   163.68     164.24    86190000        164.24 0.62928
1985-01-08    164.24    164.59   163.91     163.99    92110000        163.99 0.41486
1985-01-09    163.99    165.57   163.99     165.18    99230000        165.18 0.96347
```

Some comments on the code above:

- the function `xts` from package `xts` requires two arguments: 1) a data object to convert to `xts`, 2) a sequence of dates to assign a time stamp to each observation.
- the first argument of `xts()` is `subset(index, select=-Date)` that takes the object `index` and eliminates the column `Date` (since there is a minus sign in front of the column name)
- the second argument is `order.by=` that provides the dates of each row in the data frame
- the reason for dropping the `Date` column is that the `index.xts` object will have the dates as one of its features so that we do not need anymore a dedicated column to keep track of the dates.
- The first 6 rows of the data frame show the dates associated with each row; however, the date is not a column (as it is in `index`) but the row names. The dates can be extracted from the time series object with the command `time(index.xts)`.

The `index.xts` is a `xts` object on which we can apply convenient functions to handle, analyze and plot the data. Some examples of functions that can be used to extract information form the object are:

```
start(index.xts)          # start date
end(index.xts)            # end date
periodicity(index.xts)    # periodicity/frequency (daily, weekly, monthly)
```

```
[1] "1985-01-02"
[1] "2017-09-01"
Daily periodicity from 1985-01-02 to 2017-09-01
```

In some situations we might be interested in changing the periodicity of our time series. For example, the dataset `index.xts` is at the daily frequency and we might want to create a new object that samples the data at the weekly or monthly frequency. The `xts` package provides the functions `to.weekly()` and `to.monthly()` and an example is given below. The default in subsampling the time series is to take the first value of the period, that is, the Monday value for weekly data and the first day of the month for monthly data.

```
index.weekly <- to.weekly(index.xts)
```

```
           index.xts.Open index.xts.High index.xts.Low index.xts.Close index.xts.Volume index.xts.Adjusted
1985-01-04         167.20         167.20        163.36          163.68        234180000             163.68
1985-01-11         163.68         168.72        163.68          167.91        509830000             167.91
```

```
1985-01-18         167.91         171.94         167.58         171.32         634000000              171.32
```

Other functions that allow to subsample the time series from daily to another frequency are `apply.weekly()` and `apply.monthly()` that apply a certain function to each week or month. For example:

```
index.weekly <- apply.weekly(index.xts, "first")
```

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted   Range
1985-01-04    167.20    167.20   165.19     165.37    67820000        165.37 1.21678
1985-01-11    163.68    164.71   163.68     164.24    86190000        164.24 0.62928
1985-01-18    167.91    170.55   167.58     170.51   124900000        170.51 1.77229
```

```
index.weekly <- apply.weekly(index.xts, "last")
```

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted   Range
1985-01-04    164.55    164.55   163.36     163.68    77480000        163.68 0.72845
1985-01-11    168.31    168.72   167.58     167.91   107600000        167.91 0.68027
1985-01-18    170.73    171.42   170.66     171.32   104700000        171.32 0.44533
```

Another task that is easy with `xts` objects is to subset a time series. The code below provides several examples of selecting only the observations for year 2007, between 2007 and 2009 (`'2007/2009'`), before 2007 (`'/2007'`), and after 2007 (`'2007/'`) or between two specific dates (`'2007-03-21/2008-02-12'`). the package allows also to use `::` instead of forward slash `/` in subsetting the object. This syntax is specific to `xts` objects and does not work for time series objects of other classes[6].

```
index.xts['2007']                     # only year 2007
index.xts['2007/2009']                # between 2007 and 2009
index.xts['/2007']                    # up to 2007
index.xts['2007/']                    # starting form 2007
index.xts['2007-03-21/2008-02-12'] # between March 21, 2007 and February 12, 2008
```

2.4 Reading an online data file

There are many websites that provide access to economic and financial data, such as Yahoo Finance and FRED, among others. `R` is able to access an url address and download a dataset in the `R` session, thus saving the user the time of visiting the website and downloading the file to the local drive. Even the base function `read.csv()` can do that as illustrated in the example below:

```
url <- 'https://fred.stlouisfed.org/graph/fredgraph.csv?chart_type=line&recession_bars=on&lg_scales=&bgcolor=%23e1e
data <- readr::read_csv(url)
head(data, 3)
```

```
# A tibble: 3 x 2
        DATE EXUSEU
      <date>  <dbl>
1 1999-01-01 1.1591
2 1999-02-01 1.1203
3 1999-03-01 1.0886
```

[6]The function `window()` is more general and works for all time series objects

```
tail(data, 3)
```

```
# A tibble: 3 x 2
        DATE EXUSEU
      <date>  <dbl>
1 2017-06-01 1.1233
2 2017-07-01 1.1530
3 2017-08-01 1.1783
```

The `url` above refers to the ticker `EXUSEU` (USD-EURO exchange rate) from January 1999 until August 2017 at the monthly frequency. However, it would be convenient to have a wrapper function that takes a ticker and does all these operations automatically. One such function is `getSymbols()` from package `quantmod` that can be used to download data from Yahoo and FRED and produces `xts` objects[7]. The next two sections discuss how to use `getSymbols()` to pull data from Yahoo and FRED and several operations to get the data ready for analysis.

2.4.1 Yahoo Finance

After loading the `quantmod` package, we can start using the function `getSymbols()`. We need to provide the function with the stock tickers for which we want to obtain historical data. Another argument that we need to provide is the source and in this case it is `src="yahoo"`. Actually this was not needed since `"yahoo"` is the default value. Finally, we can specify the period that we want to download data for with the `from=` and `to=` arguments. Below is an example:

```
library(quantmod)
data <- getSymbols(c("^GSPC","^DJI"), src="yahoo", from="1990-01-01")
periodicity(GSPC)
periodicity(DJI)
head(DJI, 2)
```

```
Daily periodicity from 1990-01-02 to 2017-09-01
Daily periodicity from 1990-01-02 to 2017-09-01
           DJI.Open DJI.High DJI.Low DJI.Close DJI.Volume DJI.Adjusted
1990-01-02   2748.7   2811.7  2732.5    2810.1   20680000       2810.1
1990-01-03   2814.2   2834.0  2786.3    2809.7   23620000       2809.7
```

By default, `getSymbols()` downloads data from Yahoo at the daily frequency and the object has 6 columns representing the open, high, low, close, volume and adjusted closing price for the stock. In this example we are using the function to retrieve historical data for the S&P 500 Index (ticker: `^GSPC`) and the Dow Jones Index (`^DJI`). `getSymbols()` can handle more than one ticker and creates a `xts` object for each ticker that is provided (excluding the `^` symbol)[8].

The `quantmod` package has also functions to extract the relevant variables when not all the information provided is needed for the analysis. In the example below, we extract the adjusted closing price with the

[7] There are other functions that can be used for this task. One is the `get.hist.quote()` from the package `tseries` and the package `fImport` includes the `yahooSeries()` and `fredSeries()` functions. `R` has so many packages that there are different ways of doing almost any data analaysis task. However, for the purpose of this book I decided to stick to a few ways of doing things which I hope it is easier to learn. Once you are a proficient `R` programmer, you can find your way in the package jungle!

[8] When downloading only one symbol, the additional argument `auto.assign=FALSE` assigns the data to the object `data`.

command `Ad()` from `GSPC` and `DJI` and then merge the two time series in a new `xts` data frame called `data.new`. The `merge()` command is useful in combining time series because it matches the dates of the rows so that each row represents observations for the same time period. In addition to `Ad()`, the package defines other functions:

- `Op()`, `Cl()`, `Hi()`, `Lo()` for the open, close, high, and low price, and `Vo()` for volume
- `OpCl()` for the open-to-close daily return and `ClCl()` for the close-to-close return
- `LoHi()` for the low-to-high difference (also called the intra-day range)

```
data.new <- merge(Ad(GSPC),Ad(DJI))
```

```
           GSPC.Adjusted DJI.Adjusted
1990-01-02        359.69       2810.1
1990-01-03        358.76       2809.7
1990-01-04        355.67       2796.1
```

A characteristic of the `getSymbols()` function is to allow the downloaded data to be stored in an *environment*. If you type `ls()` in your `R` console you will get a list of items that are currently stored in your global environment, that is, the place where the objects are stored. In the previous use of the function we did not specify the argument `env` and the default is tha the object will be assigned to the global environment. However, there are situations in which we might want to store the data in a separate environment and then use specialized functions to extract the data. Consider the new environment as a folder in the global environment where some files are stored. The example below takes the list of the S&P 500 companies and downloads the data in a new environment called `store.sp`. The function prints a message that `pausing 1 second between requests for more than 5 symbols` while downloading data. The command `ls(store.sp)` can be used to pull a list of the objects stored in the `store.sp` environment. This example shows the advantage of using a programming language such as `R`: with only three lines of code we are able to download data for 500 stocks at the daily frequency in a matter of minutes.

```
splist   <- read.csv("List_SP500.csv", stringsAsFactors = FALSE)
store.sp <- new.env()
getSymbols(splist$Ticker.symbol, env=store.sp, from="2010-01-01", src="yahoo")
```

2.4.2 FRED

The `getSymbols()` can be used to download macroeconomic data from the Federal Reserve Economic Data (FRED) database. Also in this case you need to know the ticker of the variable that you are interested to download. For example, `UNRATE` refers to the monthly civilian unemployment rate in percentage, `CPIAUCSL` is the ticker for the monthly Consumer Price Index (CPI) all urban consumers seasonally adjusted, and `GDPC1` points to the quarterly real Gross Domestic Product (GDP) seasonally adjusted. There are two differences with the code used earlier: the `src=FRED` instructs the function to download the data from FRED, and the second is that it is not possible to specify a start/end date for the series. The code below does the following:

- download the time series for the three tickers
- merge the three time series
- subsample the merged object to start in January 1950

```
library(quantmod)
macrodata <- getSymbols(c('UNRATE','CPIAUCSL','GDPC1'), src="FRED")
macrodata <- merge(UNRATE, CPIAUCSL, GDPC1)
macrodata <- window(macrodata, start="1950-01-02")
```

```
           UNRATE CPIAUCSL  GDPC1
1950-02-01    6.4    23.61     NA
1950-03-01    6.3    23.64     NA
1950-04-01    5.8    23.65 2147.6
1950-05-01    5.5    23.77     NA
```

```
           UNRATE CPIAUCSL GDPC1
2017-05-01    4.3   243.85    NA
2017-06-01    4.4   243.79    NA
2017-07-01    4.3   244.05    NA
2017-08-01    4.4       NA    NA
```

Merging the three variables produces `NA` since `GDPC1` is available at the quarterly frequency and `UNRATE` and `CPIAUCSL` at the monthly frequency.

2.4.3 Quandl

Quandl works as an aggregator of public databases, plus they offer access to subscription databases. Many financial and economic datasets can be accessed using Quandl and the `R` package `Quandl`. The tickers for FRED variables are the same we used earlier with the addition of `FRED/`. The output can be in many formats, the default being a data frame with a column `Date`. In the examples below I will use the `xts` type.

```
library(Quandl)
macrodata <- Quandl(c("FRED/UNRATE", 'FRED/CPIAUCSL', "FRED/GDPC1"),
                    start_date="1950-01-02", type="xts")
head(macrodata)
```

```
           FRED.UNRATE - Value FRED.CPIAUCSL - Value FRED.GDPC1 - Value
1950-02-01                 6.4                 23.61                 NA
1950-03-01                 6.3                 23.64                 NA
1950-04-01                 5.8                 23.65             2147.6
1950-05-01                 5.5                 23.77                 NA
1950-06-01                 5.4                 23.88                 NA
1950-07-01                 5.0                 24.07             2230.4
```

The `quandl` package produces an object `macrodata` that is already merged by date and ready for analysis.

2.4.4 Reading large files

The `read.csv()` file has some nuisances, but overall it works well and it is easy to use. However, it does not perform well when data files are large (in a sense to be defined). The fact that the base `read` functions are slow has lead to the development of alternative functions and packages that have two advantages: speed and better classification of the variables. The packages `readr` and `data.table` have become popular for fast importing and below I will make a comparison of the speed of importing the same file for the three functions.

The dataset that I will use in this comparison is obtained from the Center for Research in Security Prices (CRSP) at the University of Chicago and was used in Figure 1.4. The variables in the dataset are:

- `PERMNO`: identificative number for each company
- `date`: date in format `2015/12/31`
- `EXCHCD`: exchange code
- `TICKER`: company ticker
- `COMNAM`: company name
- `CUSIP`: another identification number for the security
- `DLRET`: delisting return
- `PRC`: price
- `RET`: return
- `SHROUT`: share oustanding
- `ALTPRC`: alternative price

The observations are all companies listed in the NYSE, NASDAQ, and AMEX from January 1985 until December 2016 at the monthly frequency for a total of 3,627,236 observations and 16 variable. The size of the file is 328Mb which, in the era of big data, is actually quite small. Remember that on a 64 bit machine the RAM memory determines the constraint to the size of the file that you can import in `R`.

First, we import the file using the base `read.csv()` function. To calculate the time that it took the function to import the dataset I will use the `Sys.time()` function that provides the current time, save it to `start.time`, and then take the difference between the ending time and `start.time`. Below is the code:

```
start.time <- Sys.time()
crsp       <- read.csv("crsp_eco4051_jan2017.csv", stringsAsFactors = FALSE)
end.csv    <- Sys.time() - start.time
```

```
Time difference of 37.843 secs
```

The `read.csv()` function took 37.84 seconds to load the file. The first alternative that we consider is the `read_csv()` function from the `readr` package that aims at improving speed and variable classification (including dates, as seen earlier). Below is the code:

```
library(readr)
start.time <- Sys.time()
crsp       <- read_csv("crsp_eco4051_jan2017.csv")
end_csv    <- Sys.time() - start.time
```

```
Time difference of 9.5125 secs
```

The `read_csv()` function reduces the reading time from 37.84 to 9.51, which is a reduction of 4 times. Finally, the function `fread()` from the `data.table` package[9]. Notice that in this case I am not loading the package with the `library(data.table)` but call the function with the notation `data.table::fread()`[10]:

[9] The arguments `verbose` and `showProgress` prevent the function from printing intermediate information about the reading progress. In a typical session you might want to have updates on the status of the reading, in particular when reading large files. I set the `data.table` argument to false because I prefer to have `fread` produce a data frame rather than a `data.table` object. The `data.table` package provides functionalities that are particularly useful when working with large datasets which we won't use in this book.

[10] One situation this might be useful is when different packages have functions with the same name.

```
start.time <- Sys.time()
crsp          <- data.table::fread("crsp_eco4051_jan2017.csv",
                                   data.table=FALSE,
                                   verbose=FALSE,
                                   showProgress = FALSE)
end.fread  <- Sys.time() - start.time
end.fread
```

```
Time difference of 6.5381 secs
```

Here the reduction is even larger since the function is 5.8 time faster relative to `read.csv()` and 1.5 times relative to `read_csv()`. Although the file was not very large, there is a remarkable difference in reading speed among these functions and suggest to use the latter two in case of larger data files.

2.5 Transforming the data

Most of the times the data that we import require creating new variables that are transformations of existing ones. In finance, a common case is creating returns of an asset as the percentage growth rate of the price. The same transformation is typically applied to macroeconomic variables, in particular to calculate the growth rate of real GDP or the inflation rate (that is the growth rate of a price index, such as `CPIAUCSL`). If we define GDP or the asset price in month t by P_t, the growth rate or return is calculated in two possible ways:

1. **Simple return**: $R_t = (P_t - P_{t-1})/P_{t-1}$
2. **Logarithmic return**: $r_t = log(P_t) - log(P_{t-1})$

An advantage of using logarithmic returns is that it simplifies the calculation of multiperiod returns. This is due to the fact that the (continuously compounded) return over k periods is given by $r_t^k = \log(P_t) - \log(P_{t-k})$ which can be expressed as the sum of one-period logarithmic returns, that is

$$r_t^k = \log(P_t) - \log(P_{t-k}) = \sum_{j=1}^{k} r_{t-j+1}$$

Instead, for simple returns the multi-period return would be calculated as $R_t^k = \prod_{j=1}^{k}(1 + R_{t-j+1}) - 1$. One reason to prefer logarithmic to simple returns is that it is easier to derive the properties of the sum of random variables, rather than their product. The disadavantage of using the continuously compounded return is that when calculating the return of a portfolio the weighted average of log returns of the individual assets is only an approximation of the log portfolio return. However, at the daily and monthly horizons returns are very small and thus the approximation error is relatively minor.

Transforming and creating variables in `R` is quite simple. In the example below, data for the S&P 500 Index is obtained from Yahoo using the `getSymbols()` function and the adjusted closing price is used to create two new variables/columns, `ret.simple` and `ret.log`. To do this we employ the `log()` function that represents the natural logarithm, and the `lag(, k)` function which lags a time series by k periods. An alternative way of calculating returns is using the `diff()` command that calculates the k-period difference of the variable, $P_t - P_{t-k}$. The operations are applied to all the elements of the vector `Ad(GSPC)` and the result is vector of the same dimension.

```
GSPC             <- getSymbols("^GSPC", from="1990-01-01", auto.assign = FALSE)
GSPC$ret.simple  <- 100 * (Ad(GSPC) - lag(Ad(GSPC), 1)) / lag(Ad(GSPC),1)
GSPC$ret.log     <- 100 * (log(Ad(GSPC)) - lag(log(Ad(GSPC)), 1))
GSPC$ret.simple  <- 100 * diff(Ad(GSPC)) / lag(Ad(GSPC), 1)
GSPC$ret.log     <- 100 * diff(log(Ad(GSPC)))
head(GSPC)
```

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted ret.simple  ret.log
1990-01-02    353.40    359.69   351.98     359.69   162070000        359.69         NA       NA
1990-01-03    359.69    360.59   357.89     358.76   192330000        358.76   -0.25855 -0.25889
1990-01-04    358.76    358.76   352.89     355.67   177000000        355.67   -0.86130 -0.86503
1990-01-05    355.67    355.67   351.35     352.20   158530000        352.20   -0.97562 -0.98041
1990-01-08    352.20    354.24   350.54     353.79   140110000        353.79    0.45145  0.45043
1990-01-09    353.83    354.17   349.61     349.62   155210000        349.62   -1.17867 -1.18567
```

Some comments on the new variables created:

- the values of `ret.simple` and `ret.log` are very close which confirms that using the approximated `ret.log` produces a very small error (at the daily frequency)
- the first value of the new variables is missing because we do not have P_{t-1} for the first observation

Transformations can also be applied to all the elements of a data frame, instead of a selected column. For example, we might have an `xts` object that contains the adjusted closing prices of several assets and we would like to calculate the daily returns for all assets. We can apply the same commands discussed above to the data frame as in the example below.

```
# "^GSPC" = S&P 500 Index, "^N225" = Nikkei 225, "^STOXX50E" = EURO STOXX 50
data  <- getSymbols(c("^GSPC", "^N225", "^STOXX50E"),  from="2000-01-01")
price <- merge(Ad(GSPC), Ad(N225), Ad(STOXX50E))
ret   <- 100 * diff(log(price))
tail(ret, 5)
```

```
           GSPC.Adjusted N225.Adjusted STOXX50E.Adjusted
2017-08-29      0.084247      -0.45011          -0.96370
2017-08-30      0.460453       0.74089           0.45613
2017-08-31      0.570467       0.71362           0.52043
2017-09-01      0.198058       0.22996           0.65284
2017-09-04            NA      -0.93481                NA
```

2.6 Plotting the data

Visualizing data is an essential task of data analysis. It helps capture trends and patterns in the data that can inspire further investigation and it is also very useful in communicating the results of an analysis. Plotting is one of the great features of `R` either for simple and quick plots or for more sophisticated data visualization tasks. The basic command for plotting in `R` is `plot()` that takes as arguments the x-variable, the y-variable, the type of plot (e.g., `"p"` for point and `"l"` for line) and additional arguments to customize the plot (see `help(plot)` for details). Figure 2.1 shows a time series plot of the S&P 500 Index starting in 1985 at the daily frequency. Notice that the object `index` is a `csv` file downloaded from Yahoo Finance that has the usual open/high/low/close/volume/adjusted close structure plus a `Date` column.

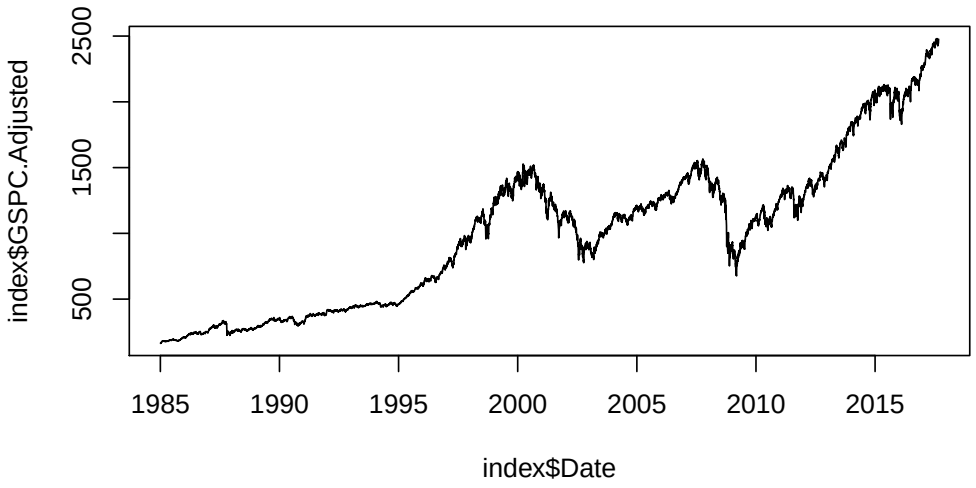


Figure 2.1: Plot of the S&P 500 Index over time starting in 1985.

After reading the file, it is necessary to define the `Date` column as a date and we then use that column as the x-axis while on the y-axis we plot the adjusted closing price.

```
setwd(location)
index      <- read_csv("GSPC.csv")
index$Date <- as.Date(index$Date, format="%Y-%m-%d")
plot(index$Date, index$GSPC.Adjusted, type="l")
```

The plots can be customized along many dimensions such as color and size of the labels, ticks, title, line and point type and much more. In Figure 2.2 the previous graph is customized by changing the color of the line to `"orange"`, changing the label's size (`cex=0.5`), the labels with `xlab` and `ylab`, and finally the title of the graph by setting `main`.

```
plot(index$Date, index$GSPC.Adjusted,
     type="l", xlab="", ylab="S&P 500 Index Value (log-scale)",
     main="S&P 500 Index", col="orange", cex.lab = 0.5)
```

An advantage of defining time series as `xts` objects is that there are specialized functions to plot. For example, when we use the base function `plot()` on a `xts` object it calls a `plot.xts()` functions that understands the nature of the data and, among other things, sets the x-axis to the time period of the variable without the user having to specify it (as we did in the previous graph). Figure 2.3 shows the time seris plot of the adjusted closing price of the S&P 500 Index.

```
plot(Ad(GSPC), main="S&P 500 Index")
```

For many economic and financial variables that display exponential growth over time, it is often convenient to plot the log of the variable rather than its level. This has the additional advantage that differences between the values at two points in time represent an approximate percentage change of the variable in that period of time. This can be achieved by plotting the natural logarithm of the variable as follows:

```
plot(log(Ad(GSPC)), xlab="Time",ylab="S&P 500 Index")
```

There are several plotting packages in `R` that can be used as an alternative to the base plotting functions.

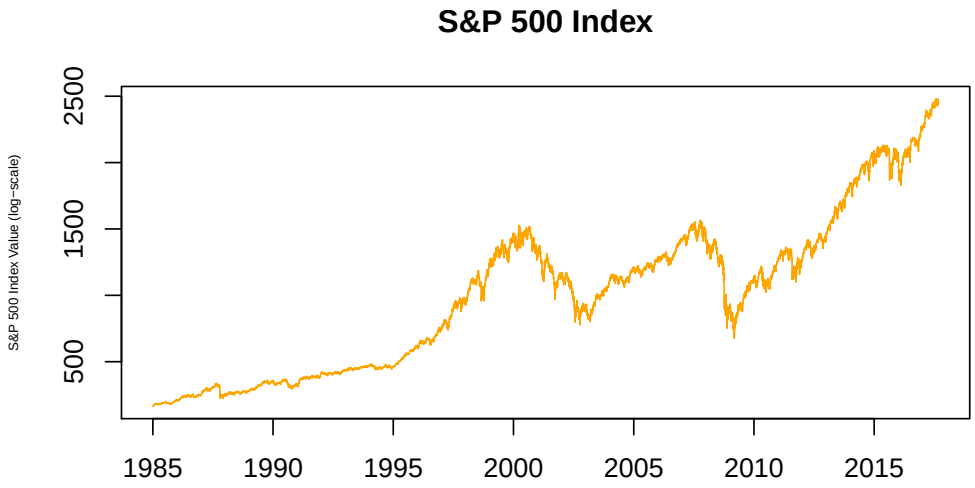


Figure 2.2: Logarithm of the S&P 500 Index at the daily frequency.

Figure 2.3: Different sub-sampling of the ‘index.xts‘ object.

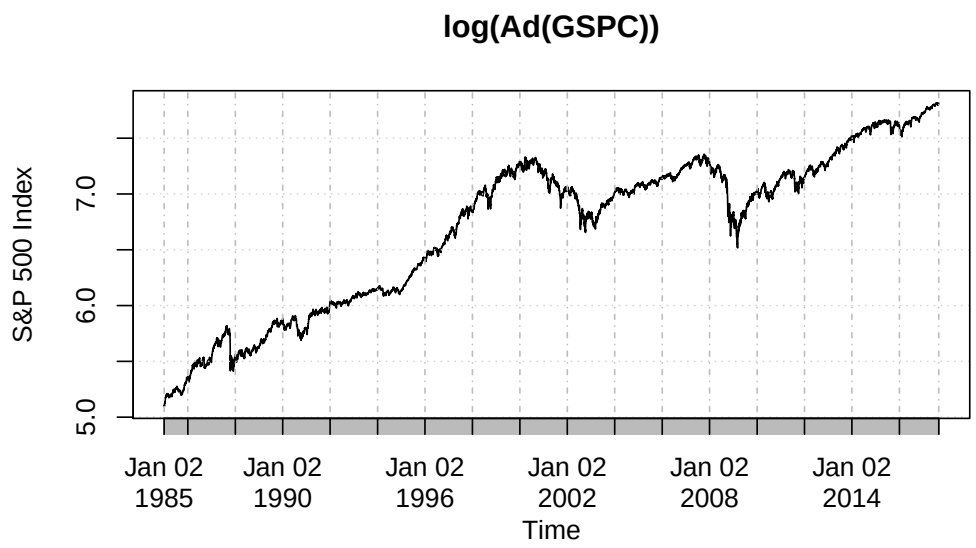


Figure 2.4: Time series plot of the logarithm of the S&P 500 Index.

I will discuss the `ggplot2` package[11] which is very flexible and makes it (relatively) easy to produce sophisticated graphics. This package has gained popularity among `R` users since it produces elegant graphs and greatly simplifies the production of advanced visualization. However, `ggplot2` does not interact with `xts` objects so that when plotting time series we need to create a `Date` variable and convert the object to a data frame. This is done in the code below, where the first line produces a data frame called `GSPC.df` that has a column `Date` and the remaining columns are from the `GSPC` object downloaded from Yahoo Finance. The function `coredata()` extracts the data frame from the `xts` object[12] The `ggplot2` has a `qplot()` function[13] that is similar to the base `plot()` function and requires:

- the x and y variables
- if `data=` is provided, then only the variable name is required
- the default plotting is points and if a different type is required it needs to be specified with `geom`

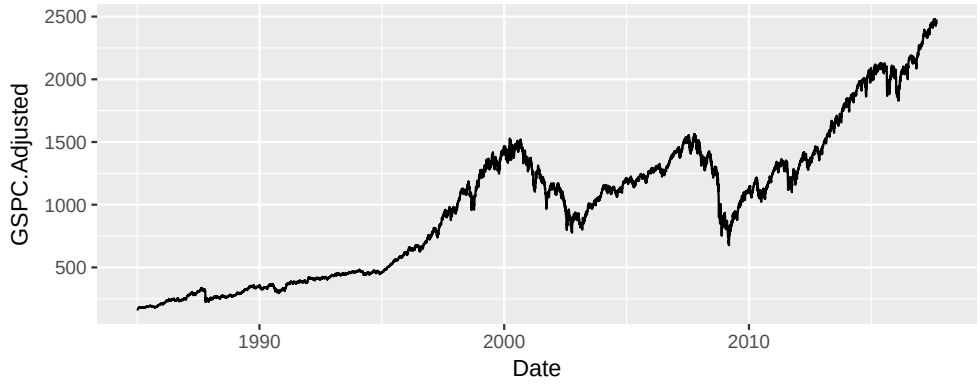
In the illustration below, the `GSPC` is converted to a data frame and then plotted using the `qplot()` against the date variable.

```
GSPC.df <- data.frame(Date = time(GSPC), coredata(GSPC))
library(ggplot2)
qplot(Date, GSPC.Adjusted, data=GSPC.df, geom="line")
```

[11] See Wickham (2009) for more details.

[12] If we did not include the `coredata()` part in this case the data frame would have been produced as well, but with row names represented by the dates (as in `xts` object). Not a big deal, though.

[13] see http://docs.ggplot2.org/dev/vignettes/qplot.html for more details.



In addition to the `qplot()` function, the `ggplot2` package provides a grammar to produce graphs. There are four building blocks to produce a graph:

- `ggplot()`: creates a new graph; can take as argument a data frame
- `aes()`: the *aes*thetics requires the specification of the x and y axis and the possible groups of variables
- `geom_`: the geometry represents the type of plot that the user would like to plot; some examples[14]:
  - `geom_point()`
  - `geom_line()`
  - `geom_histogram()`
  - `geom_bar()`
  - `geom_smooth()`
- `theme`: themes represents different styles of the graph

In Figure 2.5 there are several examples of time series plots for the S&P 500. The package `ggplot2` let us save the plot in an object (`plot1`) which we can later plot or modify by changing some of its features (as for `plot2`, `plot3`, and `plot4` below). Below I use also the function `grid.arrange()` from package `grid.Extra` to make a 2x2 grid of the 4 plots.

```
plot1 <- ggplot(GSPC.df, aes(Date, GSPC.Adjusted)) + geom_line(color="darkgreen")
plot2 <- plot1 + theme_bw()
plot3 <- plot2 + theme_classic() + labs(x="", y="Index", title="S&P 500")
plot4 <- plot3 + geom_line(color="darkorange") +  geom_smooth(method="lm") +
         theme_dark() + labs(subtitle="Period: 1985/2016", caption="Source: Yahoo")
library(gridExtra)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2)
```

A scatter plot between two variables is similarly produced by replacing the `Date` in the previous graphs with another variable. To produce Figure 2.6 I retrieve the returns for the S&P 500 and Nikkei 225 indices and calculate the monthly returns. The Figure shows the same scatter plot, but with different themes, labels, the `geom_vline()` and `geom_hline()` that produces vertical and horizontal lines, and the linear fit of regressing the US on the Japanese index.

```
data     <- getSymbols(c("^GSPC", "^N225"), from="1990-01-01")
price    <- merge(Ad(to.monthly(GSPC)), Ad(to.monthly(N225)))
ret      <- 100 * diff(log(price))
```

[14]The complete list of geoms is available at http://docs.ggplot2.org/current/

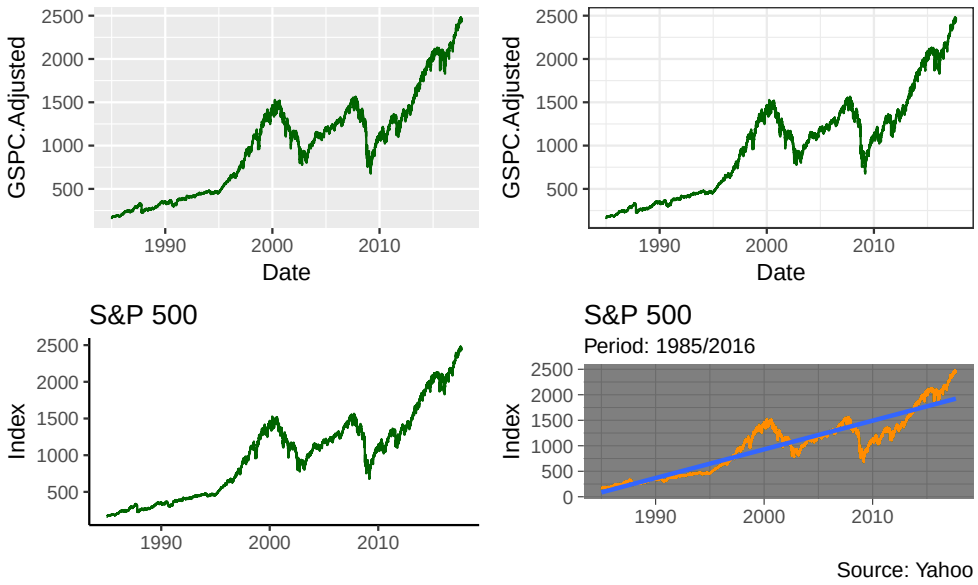


Figure 2.5: Put a caption here

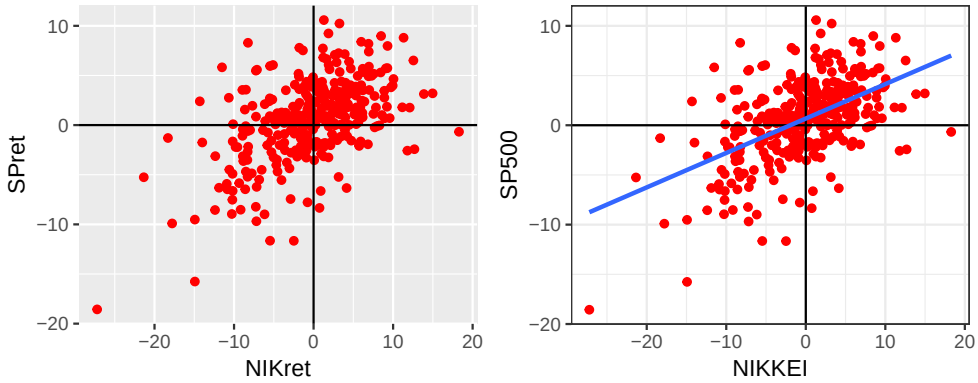


Figure 2.6: Put a caption here

```
GN.df    <- data.frame(Date=time(price), coredata(merge(price,ret)))
names(GN.df) <- c("Date","SP", "NIK", "SPret","NIKret")

plot1 <- ggplot(GN.df, aes(NIKret, SPret)) + geom_point(color="red") +
          geom_vline(xintercept = 0)  + geom_hline(yintercept = 0)
plot2 <- plot1 + geom_smooth(method="lm", se=FALSE) + theme_bw() +
          labs(x="NIKKEI", y="SP500")
grid.arrange(plot1, plot2, ncol=2)
```

2.7 Exploratory data analysis

A first step in data analysis is to calculate descriptive statistics that summarize the main statistical features of the distribution of the data, such as the average/median returns, the dispersion, the skewness and kurtosis. A function that provides a preliminary analysis of the data is `summary()` that has the following output for the simple and logarithmic returns:

```
summary(GSPC$ret.simple)
```

```
     Index              ret.simple
 Min.   :1990-01-02   Min.   :-9.0350
 1st Qu.:1996-11-20   1st Qu.:-0.4491
 Median :2003-10-27   Median : 0.0488
 Mean   :2003-10-28   Mean   : 0.0339
 3rd Qu.:2010-09-29   3rd Qu.: 0.5569
 Max.   :2017-09-01   Max.   :11.5800
                      NA's   :1
```

```
summary(GSPC$ret.log)
```

```
     Index              ret.log
 Min.   :1990-01-02   Min.   :-9.4695
 1st Qu.:1996-11-20   1st Qu.:-0.4501
 Median :2003-10-27   Median : 0.0488
 Mean   :2003-10-28   Mean   : 0.0277
 3rd Qu.:2010-09-29   3rd Qu.: 0.5554
 Max.   :2017-09-01   Max.   :10.9572
                      NA's   :1
```

Comparing the estimates of the mean, median, and 1st and 3rd quartile (25% and 75%) for the simple and log returns shows that the values are very close. However, when we compare the minimum and the maximum the values are quite different: the maximum drop is -9.035% for the simple return and -9.47% for the logarithmic return, while the maximum gain is 11.58% and 10.957%, respectively. The reason for the difference is that the logarithmic return is an approximation to the simple return that works well when the returns are small but becomes increasingly unreliable for large (positive or negative) returns.

Descriptive statistics can also be obtained by individual commands such as `mean()`, `sd()` (standard deviation), `median()`, and empirical quantiles (`quantile(, tau)` with tau a value between 0 and 1). If there are missing values in the series we need also to add the `na.rm=TRUE` argument to the function in order to eliminate these values. The package `fBasics` contains the functions `skewness()` and `kurtosis()` that are particularly relevant in the analysis of financial data. This package provides also the function `basicStats()` that provides a table with all of these descriptive statistics:

```
library(fBasics)
basicStats(GSPC$ret.log)
```

```
                ret.log
nobs        6974.000000
NAs            1.000000
Minimum       -9.469512
Maximum       10.957197
1. Quartile   -0.450125
3. Quartile    0.555389
Mean           0.027669
```

```
Median         0.048804
Sum          192.937921
SE Mean        0.013345
LCL Mean       0.001508
UCL Mean       0.053831
Variance       1.241906
Stdev          1.114409
Skewness      -0.248201
Kurtosis       8.875874
```

In addition, when the analysis involves several assets we want to measure their linear dependence through measures like the covariance and correlation. For example, the `my.df` object defined above is composed of the US and Japanese equity index and it is interesting to measure how the two index returns co-move. The functions to estimate the covariance is `cov()` and the correlation is `cor()`, with the additional argument of `use='complete.obs'` that tells `R` to estimate the quantity for all pairs on the set of dates that are common to all assets:

```
Ret <- subset(GN.df, select=c("SPret","NIKret"))
cov(Ret, use='complete.obs')
```

```
        SPret NIKret
SPret  17.053 13.470
NIKret 13.470 38.881
```

The elements in the diagonal are the variances of the index returns and the off-diagonal element represents the covariance between the two series. The sample covariance is equal to 13.47 which is difficult to interpret since it depends on the scale of the two variables. That is a reason for calculating the correlation that is scaled by the standard deviation of the two variable and is thus bounded between 0 and 1. The correlation matrix is calculated as:

```
cor(Ret, use='complete.obs')
```

```
         SPret  NIKret
SPret  1.00000 0.52309
NIKret 0.52309 1.00000
```

the diagonal elements are equal to 1 because they represent the correlation of the S&P 500 (N225) with the S&P 500 (N225), while the off-diagonal element is the sample correlation between the two indices. It is equal to 0.52 which indicates that the monthly returns of the two equity indices co-move, although their correlation is not extremely high.

Another useful exploratory tool in data analysis is the histogram that represents an estimator of the distribution of a variable. Histograms are obtained by dividing the range of a variable in small bins and then count the fraction of observations that fall in each bin. The histogram plot shows the distribution characteristics of the data. The function `hist()` in the `base` package and the `geom_histogram()` function in `ggplot2` are the commands to use in this task:

```
hist(GSPC$ret.log, breaks=50,  xlab="", main="")     # base function
qplot(ret.log, data=GSPC, geom="histogram", bins=50) # ggplot function
```

In these plots we divided the range of the variable into 50 bins and the number in the y-axis represents the number of observations. The histogram shows that daily returns are highly concentrated around 0 and with long tails, meaning that there are several observations that are far from 0 (which is approximately

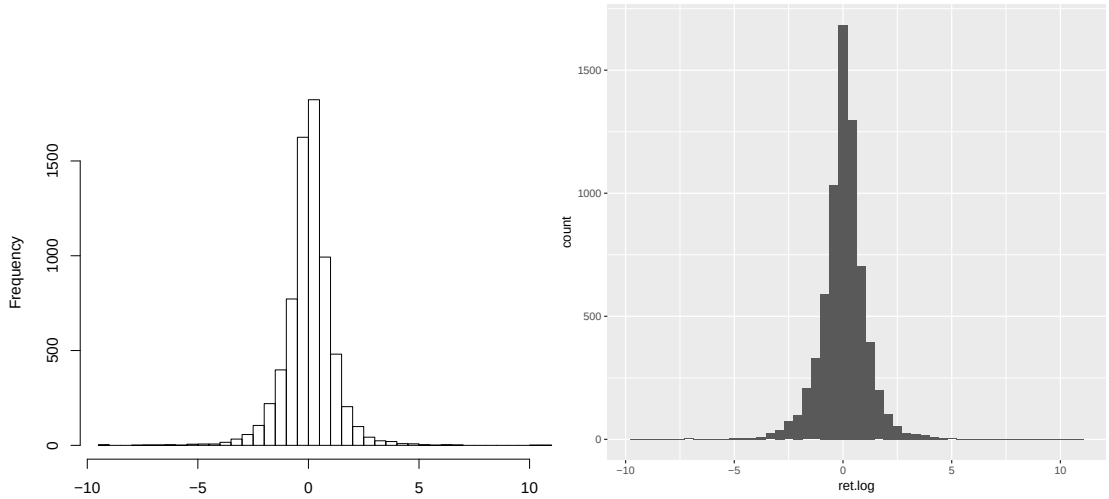


Figure 2.7: Histogram of the daily return of the SP500 produced with base graphics (left) and the 'ggplot2' package (right).

the mean/median of the series).

We can also add a nonparametric estimate of the frequency that smooths out the roughness of the histogram and makes the density estimates continuous[15]:

```
# base function
hist(GSPC$ret.log, breaks=50, main="", xlab="", ylab="",prob=TRUE)
lines(density(GSPC$ret.log,na.rm=TRUE),col=2,lwd=2)
box()
# ggplot function
ggplot(GSPC, aes(ret.log)) +
  geom_histogram(aes(y = ..density..), bins=50, color="black", fill="white") +
  geom_density(color="red", size=1.2) +
  theme_bw()
```

2.8 Dates and Times in R

When we read a time series dataset it is convenient to define a date variable that keeps track of the time ordering of the variables. We did this already when plotting the S&P 500 when we used the command `index$Date <- as.Date(index$Date, format="%Y-%m-%d")`. `as.Date` is a command that takes a string as input and defines it of the `Date` type. In this case we specified the format of the variable which, in this case, is composed of a 4-digit year (`%Y` otherwise `%y` for 2-digit), a hyphen, the 2 digit month (`%m`), a hyphen, and finally the two digit day (`%d`). Other possible formats of the day is the weekday (`%a` abbreviate or `%A` unabbreviated) and month name (`%b` abbreviate or `%B` unabbreviated). The default `R` output is `yyyy-mm-dd` as shown in the examples below for different date formats:

[15]The `prob=TRUE` argument makes the y-scale probabilities instead of frequencies

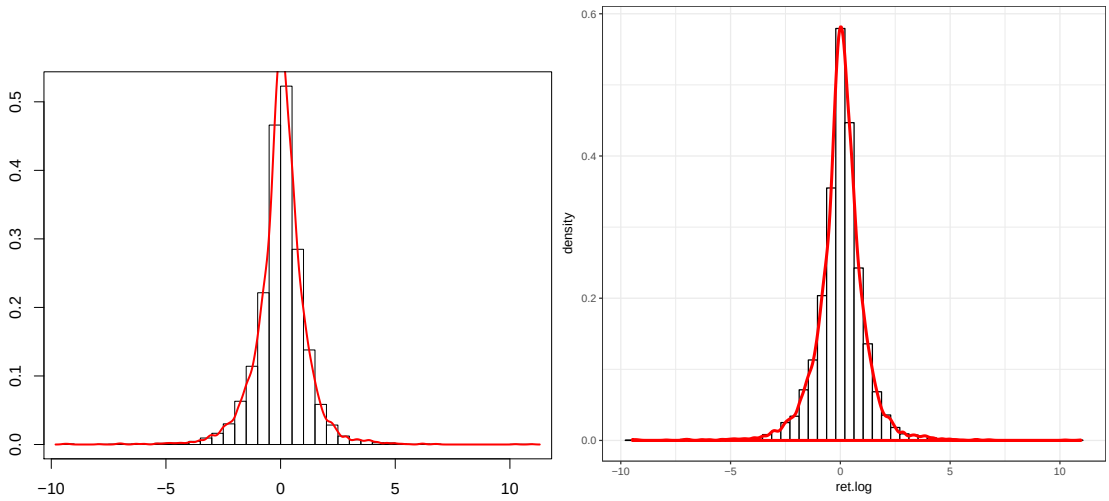


Figure 2.8: Histogram of the daily return of the SP500 produced with base graphics (left) and the 'ggplot2' package (right) together with a smoothed density.

```
as.Date("2011-07-17")  # no need to specify format
as.Date("July 17, 2011", format="%B %d,%Y")
as.Date("Monday July 17, 2011", format="%A %B %d,%Y")
as.Date("17072011", format="%d%m%Y")
as.Date("11@17#07", format="%y@%d#%m")
```

```
[1] "2011-07-17"
[1] "2011-07-17"
[1] "2011-07-17"
[1] "2011-07-17"
[1] "2011-07-17"
```

Once we have defined dates, we can also calculate the time passed between two dates by simply subtracting two dates or using the `difftime()` function that allows to specify the time unit of the result(e.g., `"secs"`, `"days"`, and `"weeks"`):

```
date1 <- as.Date("July 17, 2011", format="%B %d,%Y")
date2 <- Sys.Date()
date2 - date1
difftime(date2, date1, units="secs")
difftime(date2, date1, units="days")
difftime(date2, date1, units="weeks")
```

```
Time difference of 2241 days
Time difference of 193622400 secs
Time difference of 2241 days
Time difference of 320.14 weeks
```

In addition to the date, we might need to associate a time of the day to each observation as in the case of high-frequency data. In the previous Chapter we discussed the tick-by-tick quote data for the dollar-yen exchange rate obtained from TrueFX. The first 10 observations of the file for the month of December 2016 is shown below:

```
data.hf <- data.table::fread('USDJPY-2016-12.csv',
                             col.names=c("Pair","Date","Bid","Ask"),
                             colClasses=c("character","character","numeric","numeric"),
                             data.table=FALSE, verbose = FALSE, showProgress = FALSE)
```

```
      Pair                 Date    Bid    Ask
1  USD/JPY 20161201 00:00:00.041 114.68 114.69
2  USD/JPY 20161201 00:00:00.042 114.68 114.69
3  USD/JPY 20161201 00:00:00.186 114.68 114.69
4  USD/JPY 20161201 00:00:00.188 114.68 114.69
5  USD/JPY 20161201 00:00:00.189 114.69 114.70
6  USD/JPY 20161201 00:00:00.223 114.69 114.70
7  USD/JPY 20161201 00:00:00.343 114.69 114.70
8  USD/JPY 20161201 00:00:00.347 114.69 114.70
9  USD/JPY 20161201 00:00:00.403 114.69 114.69
10 USD/JPY 20161201 00:00:00.415 114.69 114.69
```

The first part of the date represents the day in the format `yyyymmdd` and we know how to handle that from the above discussion. The second part represents the time of the day the quote was issued and the format is hour:minute:second (`hh:mm:ss`). Notice that the seconds are decimal and `00.041` represents a fraction of a second. In this case the function `as.Date()` is not useful because it only takes care of the date part, but not the time part. Two other functions are available to convert date-time strings to date-time objects. The function `strptime()` and `as.POSIXlt()` can be used with similar functionality[16] Both functions require to specify the format in terms of hour (`%H`), minute (`%M`), second (`%S`), and decimal second (`%OS`). Notice that the date-time produced below are defaulted to the `EST` time zone, but this can be easily changed with argument `tz`.

```
strptime("20161201 01:00", format="%Y%m%d %H:%M")
strptime("20161201 00:00:01", format="%Y%m%d %H:%M:%S")
strptime("20161201 00:00:00.041", format="%Y%m%d %H:%M:%OS")
as.POSIXlt("20161201 00:00:00.041", format="%Y%m%d %H:%M:%OS")
```

```
[1] "2016-12-01 01:00:00 EST"
[1] "2016-12-01 00:00:01 EST"
[1] "2016-12-01 00:00:00.041 EST"
[1] "2016-12-01 00:00:00.041 EST"
```

The function `strptime()` can also be used to define dates with no time stamp and it will produce a `POSIXt` date.

```
date1 <- as.POSIXlt("20161201 00:00:00.041", format="%Y%m%d %H:%M:%OS")
date2 <- strptime("20161201 01:15:00.041", format="%Y%m%d %H:%M:%OS")
date2 - date1
difftime(date2, date1, unit="secs")
```

```
Time difference of 1.25 hours
Time difference of 4500 secs
```

We can now re-define the `Date` column in the `data.hf` frame using the `strptime()` function:

[16] While the `as.POSIXlt()` function produces a date-time in the format year, month, day, hour, minute, and second, the function `as.POSIXct()` outputs a value that is the difference relative to a base date (January 1, 1970). I will discuss only `as.POSIXlt()` since it is the one that will be used mostly in this book.

```
data.hf$Date <- strptime(data.hf$Date, format="%Y%m%d %H:%M:%OS")
str(data.hf)
```

```
'data.frame':   14237744 obs. of  4 variables:
 $ Pair: chr  "USD/JPY" "USD/JPY" "USD/JPY" "USD/JPY" ...
 $ Date: POSIXlt, format: "2016-12-01 00:00:00.041" "2016-12-01 00:00:00.042" "2016-12-01 00:00:00.186" "2016-12-01 00:00:00.188" ..
 $ Bid : num  115 115 115 115 115 ...
 $ Ask : num  115 115 115 115 115 ...
```

2.8.1 The lubridate package

There are several packages that provide functionalities to make it easier to work with dates and times. I will discuss the `lubridate` package that will be used in several chapters of this book since it has an easier syntax and provides several useful functions. The package has functions that can be used to define a string as a date:

- `ymd`: for dates in the format year, month, day
- `dmy`: dates with day, month, year format
- `mdy`: when the format is month, day, year
- `ymd_hm`: in addition to the date the time is provided in hour and minute (the date part can be changed to other formats)
- `ymd_hms`: the time format is hour, minute, and seconds

Below are some examples of dates that are parsed with these functions[17]:

```
library(lubridate)
ymd("20170717")
ymd("2017/07/17")
ymd_hm("20170717 01:00")
ydm_hms("20171707 00:00:00.041")
```

```
[1] "2017-07-17"
[1] "2017-07-17"
[1] "2017-07-17 01:00:00 UTC"
[1] "2017-07-17 00:00:00.040 UTC"
```

Notice that `ymd("17/07/2017")` would output a `NA` with a warning message that the parsing failed. This is because the string date is in the format `dmy()` instead of `ymd()`. The package `lubridate` provides also a series of functions that are particularly useful to extract parts of the date-time object, as shown below:

```
mydate <- ydm_hms("20171707 00:00:00.041")
year(mydate)
```

```
[1] 2017
```

```
month(mydate)
```

```
[1] 7
```

```
day(mydate)
```

```
[1] 17
```

[17] The default time zone in this case is `UTC` rather than `ETC` and there is a 7-hour difference between the two time zones.

```
minute(mydate)
```

```
[1] 0
```

```
second(mydate)
```

```
[1] 0.041
```

2.9 Manipulating data using dplyr

Time series data have a major role in financial analysis and we discussed two ways to manipulate them. The first is to define the data as a time series object (e..g, `xts`) that consists of embedding the time series properties in the object. The alternative approach is to maintain the data as a data frame object and define a `date` variable that keeps track of the temporal ordering of the data.

If the second route is taken, the `dplyr` package[18] is a useful tool that can be used to manipulate data frames. The package has the following properties:

- defines a new type of data frames, called `tibble`, that have some convenient features
- defines commands to manipulate data for the most frequent operations that make the task easier and more transparent
- these commands are executed faster relative to equivalent base `R` commands that is particularly useful when dealing with large datasets

To illustrate the use of the `dplyr` package I will use the `GSPC.df` data frame that was created earlier. It is composed of 7 columns and the last 3 rows are shown below:

```
          Date   Open   High    Low  Close     Volume Adjusted
8235 2017-08-30 2446.1 2460.3 2443.8 2457.6 2633660000   2457.6
8236 2017-08-31 2462.7 2475.0 2462.7 2471.6 3348110000   2471.6
8237 2017-09-01 2474.4 2480.4 2473.8 2476.6 2710730000   2476.6
```

The main functions (or verbs) of the `dplyr` package are:

- `mutate`: to create new variables
- `select`: to select columns of the data frame
- `filter`: to select rows based on a criterion
- `summarize`: uses a function to summarize columns/variables in one value
- `arrange`: to order a data frame based on one or more variables

In the code below, I create new variables using the `mutate` function. In particular, I create the percentage range calculated as 100 times the logarithm of the ratio of highest and lowest intra-day price, the open-close returns, the close-to-to close return, and a few variables that extract time information from the date using `lubridate` functions:

```
library(dplyr)
library(lubridate)

GSPC.df <- mutate(GSPC.df, range   = 100 * log(High/Low),
```

[18] See the page https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html for a more detailed introduction to the package.

```
                          ret.o2c = 100 * log(Close/Open),
                          ret.c2c = 100 * log(Adjusted / lag(Adjusted)),
                          year    = year(Date),
                          month   = month(Date),
                          wday    = wday(Date, label=T, abbr=F))
```

```
          Date   Open   High    Low  Close     Volume Adjusted   range   ret.o2c  ret.c2c year month      wday
8233 2017-08-28 2447.3 2449.1 2439.0 2444.2 2677700000   2444.2 0.41284 -0.127157 0.048695 2017     8    Monday
8234 2017-08-29 2431.9 2449.2 2428.2 2446.3 2737580000   2446.3 0.86071  0.588741 0.084247 2017     8   Tuesday
8235 2017-08-30 2446.1 2460.3 2443.8 2457.6 2633660000   2457.6 0.67454  0.470266 0.460453 2017     8 Wednesday
8236 2017-08-31 2462.7 2475.0 2462.7 2471.6 3348110000   2471.6 0.50064  0.364790 0.570467 2017     8  Thursday
8237 2017-09-01 2474.4 2480.4 2473.8 2476.6 2710730000   2476.6 0.26361  0.086046 0.198058 2017     9    Friday
```

Notice that the `dplyr` package has its own `lag()` and `lead()` functions, as well as additional functions that we will use in the following chapters. If we are interested in only a few variables of the data frame, we can use the `select` command with arguments the data frame and the list of variables to retain:

```
GSPC.df1 <- dplyr::select(GSPC.df, Date, year, month, wday, range, ret.c2c)
```

```
           Date year month      wday   range  ret.c2c
8232 2017-08-25 2017     8    Friday 0.47956 0.167147
8233 2017-08-28 2017     8    Monday 0.41284 0.048695
8234 2017-08-29 2017     8   Tuesday 0.86071 0.084247
8235 2017-08-30 2017     8 Wednesday 0.67454 0.460453
8236 2017-08-31 2017     8  Thursday 0.50064 0.570467
8237 2017-09-01 2017     9    Friday 0.26361 0.198058
```

The `filter()` command is used when the objective is to subset the data frame according to the values of some of the variables. For example, we might want to extract the data for the month of October 2008 (as done below), the days that are Wednesday, or that range is larger than 3. This is done by specifying the logical conditions as arguments of the `filter()` function:

```
GSPC.df2 <- dplyr::filter(GSPC.df1, year == 2008, month == 10)   # NB: the double equal sign
```

```
        Date year month      wday  range  ret.c2c
1 2008-10-01 2008    10 Wednesday 2.2759 -0.45543
2 2008-10-02 2008    10  Thursday 4.3324 -4.11249
3 2008-10-03 2008    10    Friday 4.9460 -1.35986
```

```
         Date year month      wday  range ret.c2c
21 2008-10-29 2008    10 Wednesday 5.0438 -1.1141
22 2008-10-30 2008    10  Thursday 3.6722  2.5477
23 2008-10-31 2008    10    Friday 4.1261  1.5249
```

Instead, the function `summarize()` is used to apply a certain function, such as `mean()` or `sd()`, to one or more variables in the data frame. Below, the average return and the average range is calculated for `GSPC.df2` that represents the subset of the data frame for October 2008.

```
summarize(GSPC.df2, av.ret = mean(ret.c2c, na.rm=T), av.range = mean(range, na.rm=T))
```

```
    av.ret av.range
1 -0.80712   6.6576
```

All the functions above can be combined to manipulate the data to answer the question of interest and they can also be used on groups of observations. `dplyr` has a function called `group_by()` that creates groups of observations that satisfy one or more criterion and we can then apply the `summarize()` function

on each group. Continuing with the daily S&P 500 example, we might be interested in calculating the average return and the average range in every year in our sample. The year is thus the grouping variable that we will use in `group_by()` that combined with the command `summarize()` used earlier produces the answer:

```
temp <- group_by(GSPC.df1, year)
summarize(temp, av.ret = mean(ret.c2c, na.rm=T), av.range = mean(range, na.rm=T))
```

```
# A tibble: 33 x 3
    year     av.ret av.range
   <dbl>      <dbl>    <dbl>
 1  1985  0.0976091  0.78993
 2  1986  0.0539351  1.12161
 3  1987  0.0079337  1.78254
 4  1988  0.0462060  1.22407
 5  1989  0.0956298  0.95458
 6  1990 -0.0268148  1.31328
 7  1991  0.0923095  1.10574
 8  1992  0.0171948  0.82298
 9  1993  0.0269462  0.70829
10  1994 -0.0061558  0.82103
# ... with 23 more rows
```

The operations performed above can be streamlined using the `piping` operator `%>%` (read as *then*). This is a construct that allows to write more compact and, hopefully, more readable code. The code is more compact because it is often the case that we are not interested in storing the intermediate results (e.g., `temp` in the previous code chunk), but only to produce the table with the yearly average return and range. The intermediate results are passed from the operation to the left of `%>%` to the operation on the right or it can be referenced with a dot(`.`). The advantage of piping is that the sequence of commands can be structured more transparently: first mutate the variables *then* select some variables *then* group by one or more variables *then* summarize. The code below uses the `%>%` operator to perform the previous analysis and then plots the average range by year using the `ggplot2` package:

```
GSPC.df %>% mutate(range   = 100 * log(High/Low),
                   ret.o2c = 100 * log(Close/Open),
                   ret.c2c = 100 * log(Adjusted / lag(Adjusted)),
                   year    = year(Date),
                   month   = month(Date),
                   day     = day(Date),
                   wday    = wday(Date, label=T, abbr=F)) %>%
           dplyr::select(year, range, ret.c2c) %>%
           group_by(year) %>%
           summarize(av.ret = mean(ret.c2c, na.rm=T), av.range = mean(range, na.rm=T)) %>%
           ggplot(., aes(year, av.range)) + geom_line(color="magenta") + theme_classic()
```

The previous discussion demonstrates that the `dplyr` package is a very powerful tool to quickly extract information from a dataset with benefits that reduces the loss of flexibility deriving from time series objects. For example, let's say that we are interested in answering the question: is volatility higher when the Index goes down relative to when it goes up? does the relationship change over time? To answer this question we need to break it down in the building blocks of the `dplyr` package: 1) first create a new variable that takes value, e.g., `up` or `down` (or 0 vs 1) if the GSPC return was positive or negative, 2) group the observations according to the newly created variable, 3) calculate the average range in these two groups. The `R` implementation is as follows:

```
GSPC.df1 %>% mutate(direction = ifelse(ret.c2c > 0, "up", "down")) %>%
             group_by(direction) %>%
             summarize(av.range = mean(range, na.rm=T))
```

```
# A tibble: 3 x 2
  direction av.range
      <chr>    <dbl>
1      down   1.3365
2        up   1.1730
3      <NA>   1.2094
```

The results indicate that volatility, measured by the intra-day range, is on average higher in days in which the market declines relative to days in which it increases. The value for `NA` is due to the fact that there is one missing value in the dataset. This can be easily eliminated by filtering out NA before creating the `direction` variable with the command `filter(GSPC.df1, !is.na(ret.c2c))`. The function `ifelse()` used above simply assigns value `up` to the variable `direction` if the condition `ret.c2c > 0` is satisfied, and otherwise it assigns the value `down`.

2.10 Creating functions in R

So far we discussed functions that are available in `R`, but one of the (many) advantages of using a programming language is that it is possible to create functions that are taylored to the analysis you are planning to conduct. We will illustrate this with a simple example. Earlier, we calculated the average monthly return of the S&P 500 Index using the command `mean(GSPC$ret.log, na.rm=T)` that is equal to 0.02767. We can write a function that calculates the mean and compares the results with those of the function `mean()`. Since the sample mean is obtained as $\bar{R} = \sum_{t=1}^{T} R_t/T$ we can write a function that

takes a time series as input and gives the sample mean as output. We can call this function `mymean` and the syntax of defining a function is as follows:

```
mymean <- function(Y)
{
  Ybar <- sum(Y, na.rm=T) / length(Y)
  return(Ybar)
}

mymean(GSPC$ret.log)
```

```
[1] 0.027665
```

Not surprisingly, the result is the same as the one obtained using the `mean` function. More generally, a function can take several arguments, but it has to return only one outcome, which could be a list of items. The function we defined above is quite simple and it has several limitations: 1) it does not take into account that the series might have `NA`s, and 2) it does not calculate the mean of each column in case there are several. As an exercise, modify the `mymean` function to accomodate for these issues.

2.11 Loops in R

A loop consists of a set of commands that we are interested to repeat a pre-specified number of times and to store the results for further analysis. There are several types of loops, with the `for` loop probably the most popular. The syntax in `R` to implement a `for` loop is as follows:

```
for (i in 1:N)
{
   ## write your commands here
}
```

where `i` is an indicator and `N` is the number of times the loop is repeated. As an example, we can write a function that contains a loop to calculate the sum of a variable and compare the results to the `sum()` function provided in `R`. This function could be written as follows:

```
mysum <- function(Y)
{
  N    = length(Y)   # define N as the number of elements of Y
  sumY = 0           # initialize the variable that will store the sum of Y

  for (i in 1:N)
  {
      if (!is.na(Y[i])) sumY = sumY + as.numeric(Y[i]) # current sum is equal to previous sum
  }                                  # plus the i-th value of Y
  return(sumY)                       # as.numeric(): makes sure to transform
}                                    # from other classes to a number

c(sum(GSPC$ret.log, na.rm=T), mysum(GSPC$ret.log))
```

```
[1] 192.94 192.94
```

Notice that to define the `mysum()` function we only use the basic `+` operator and the `for` loop. This is just a simple illustration of how the `for` loop can be used to produce functions that perform a certain operation on the data. Let's consider another example of the use of the `for` loop that demonstrates the validity of the Central Limit Theorem (CLT). We are going to do this by simulation, which means that we simulate data and calculate some statistic of interest and repeat these operations a large number of times. In particular, we want to demonstrate that, no matter how the data are distributed, the sample mean is normally distributed with mean the population mean and variance given by σ^2/N, where σ^2 is the population variance and N is the sample size. We assume that the population distribution is $N(0, 4)$ and we want to repeat a large number of times the following operations:

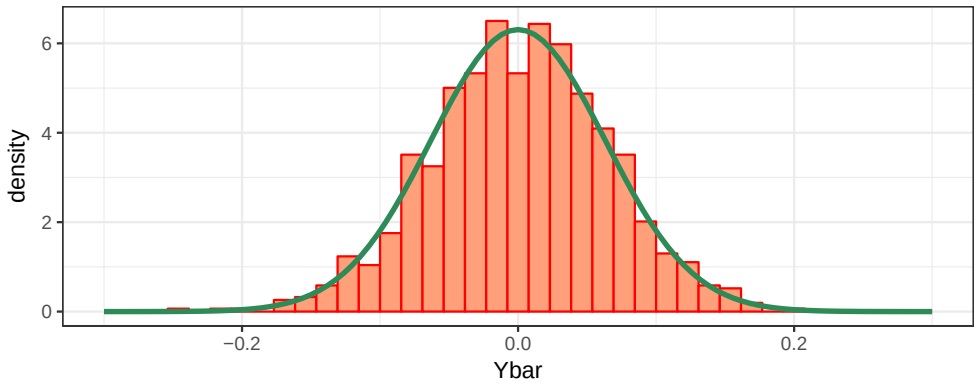
1. Generate a sample of length `N`
2. Calculate the sample mean
3. Repeat 1-2 `S` times

Every statistical package provides functions to simulate data from a certain distribution. The function `rnorm( N, mu, sigma)` simulate `N` observations from the normal distribution with mean `mu` and standard deviation `sigma` whilst `rt(N, df, ncp)` generates a sample of length `N` from the t distribution with `df` degrees-of-freedom and non-centrality parameter `ncp`. The code to perform this simulation is as follows:

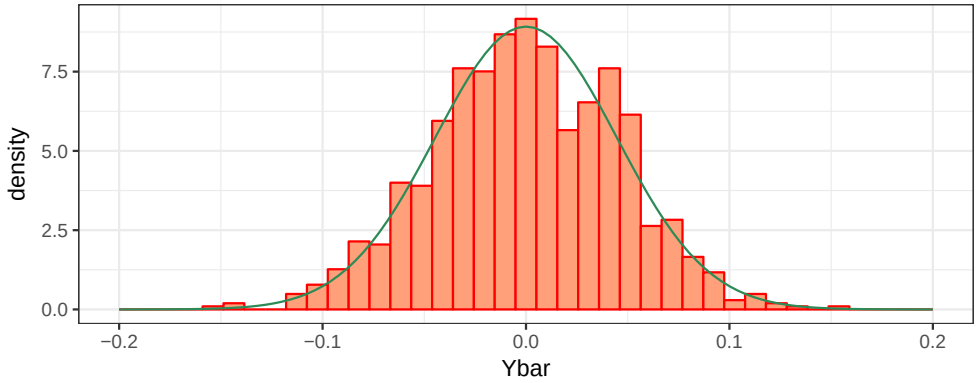
```
S     = 1000  # set the number of simulations
N     = 1000   # set the length of the sample
mu    = 0      # population mean
sigma = 2      # population standard deviation

Ybar = vector('numeric', S)  # create an empty vector of S elements
                             # to store the t-stat of each simulation
for (i in 1:S)
{
  Y        = rnorm(N, mu, sigma)  # Generate a sample of length N
  Ybar[i]  = mean(Y)              # store the t-stat
}
c(mean(Ybar), sd(Ybar))
```

```
[1] 0.0013602 0.0637967
```

The object `Ybar` contains 1000 elements each representing the sample mean of a random sample of length 1000 drawn from a certain distribution. We expect that these values are distributed as a normal distribution with mean equal to 0 (the population mean) and standard deviation $2/31.62278 = 0.06325$. We can assess this by plotting the histogram of `Ybar` and overlap it with the distribution of the sample mean. The graph below shows that the two distribution seem very close to each other. This is confirmed by the fact that the mean of `Ybar` and its standard deviation are both very close to their expected values. To evaluate the normality of the distribution, we can estimate the skewness and kurtosis of `store` which we expect to be close to zero to indicate normality. These values are -0.09 and 0.1 which can be considered close enough to zero to conclude that `Ybar` is normally distributed.



What would happen if we generate samples from a t instead of a normal distribution? For a small number of degrees-of-freedom the t distribution has fatter tails than the normal, but the CLT is still valid and we should expect results similar to the previous ones. We can run the same code as above, but replace the line `Y = rnorm(N, mu, sigma)` with `Y = rt(N, df)` with `df=4`. The plot of the histogram and normal distribution (with $\sigma^2 = df/(df-2)$) below shows that the empirical distribution of `Ybar` closely tracks the asymptotic distribution of the sample mean.



R commands

Table 2.1: R functions used in this Chapter.

Ad()	day()	getSymbols()	minute()	second()	theme_bw()
apply.monthly()	density()	ggplot()	month()	select()	theme_classic()
apply.weekly()	diff()	grid.arrange()	mutate()	start()	to.monthly()
as.Date()	difftime()	group_by()	names()	str()	to.weekly()
as.POSIXlt()	end()	head()	new.env()	strptime()	window()
basicStats()	filter()	hist()	periodicity()	subset()	write.csv()
box()	fread()	labs()	plot()	sum()	xts()
class()	geom_density()	lag()	qplot()	summarize()	ydm_hms()
cor()	geom_histogram()	length()	Quandl()	summary()	year()
cov()	geom_line()	lines()	read_csv()	Sys.time()	ymd_hm()
data.frame()	geom_smooth()	merge()	read.csv()	tail()	ymd()

Table 2.2: R packages used in this Chapter.

data.table	fBasics	gridExtra	quandl	readr
dplyr	ggplot2	lubridate	quantmod	NA

Exercises

Create a Rmarkdown file (`Rmd`) in Rstudio and answer each question in a separate section. To get started with Rmarkdown visit this page. The advantage of using Rmarkdown is that you can embed in the same document the `R` code, the output of your code, and your discussion and comments. This saves a significant amount of time relative to having to copy and paste tables and graphs from `R` to a word processor.

1. Perform the analysis in exercise 1 of Chapter 1 using `R` instead of Excel. In addition to the questions in the exercise, answer also the following question: which software between `Excel` and `R` makes the analysis easier, more transparent, more scalable, and more fun?

2. Download data from Yahoo Finance for `SPY` (SPDR S&P 500 ETF Trust) starting in `1995-01-01` at the daily frequency. Use the `dplyr` package to answer the following questions:

   - Is the daily average return and volatility (measured by the intra-day range), higher on Monday relative to Friday?
   - Which is the most volatile month of the year?
   - Is a large drop or a large increase in `SPY`, e.g. larger than 3% in absolute value, followed by a large return the following day? what about volatility?

3. Download data for `GLD` (SPDR Gold Trust ETF) and `SPY` (SPDR S&P 500 ETF) at the daily frequency starting in `2004-12-01`. Answer the following questions:

   - Create the daily percentage returns and use the `ggplot2` package to do a scatter plot of the two asset returns. Does the graph suggest that the two assets co-move?

- Add a regression line to the previous graph with command `geom_smooth(metod="lm")` and discuss if it confirms your previous discussion.
- Estimate the descriptive statistics for the two assets with command `basicStats()` from package `fBasics`; discuss the results for each asset and comparing the statistical properties of the two assets
- Calculate the correlation coefficient between the two assets; comment on the magnitude of the correlation coefficient.
- Use `dplyr` to estimate the correlation coefficient every year (instead of the full sample as in the previous question). Is the coefficient varying significantly from year to year? Use `ggplot2` to plot the average correlation over time.

4. Import the `TrueFX` file that you used in exercise 4 of the previous Chapter. Answer the following questions:
   - Import the file using the `fread()` from the `data.table` package and the `read_csv()` from the `readr` package:
     - calculate the time that it took to load the file and compare the results
     - print the structure of the file loaded with the two functions and compare the types of the variables (in particular, the date-time variable)
   - Calculate the dimension of the data frame
   - Use `dplyr` to calculate the following quantities:
     - total number of quotes in each day of the sample and plot it over time
     - the intra-day range determined by the log-difference of the highest and lowest price; do a time series plot
     - create a `minutes` variable using the `minute()` function of `lubridate`; use `group_by()` and `summarize()` to create a price at the 1 minute interval by taking the last observation within each minute. Use `ggplot2` to do a time series plot of the FX rate at the 1 minute interval.

Chapter 3

Linear Regression Model

Are financial returns predictable? This question has received considerable attention in academic research and in the finance profession. The mainstream theory in financial economics is that markets are efficient and that prices reflect all available information. If this theory is correct, current values of economic and financial variables should not be useful in predicting future returns. The theory can thus be empirically tested by evaluating if the data show evidence of a relationship between future returns and current value of predictive variables, e.g. the dividend-price ratio. In practice, the fact that many fund managers follow an "active" strategy of "outperforming the market" seems at odd with the theory that prices are unpredictable and that investors should only follow a "passive" strategy.

There is an extensive literature in financial economics trying to answer this question. A recent article by Welch and Goyal (2008) is a comprehensive evaluation of the many variables that were found to predict financial returns. The conclusion of the article is that there is some evidence, although not very strong, that returns are predictable. However, the predictability is hardly useful when trying to forecast returns in real-time[1]. The authors provide the data used in their paper and also an updated version to 2015 at this link. The dataset is provided at the monthly, quarterly, and annual frequency and contains several variables described in detail in the article. For now, we are only interested in using `D12` (the dividend), the `Rfree` (riskfree rate proxied by the 3-month T-bill rate), and the `CRSP_SPvw` that represents the return of the CRSP value-weighted Index. The code below downloads and reads the Excel file and then creates new variables using the `dplyr` package. The new variables created are `DP` (the percentage dividend-price ratio) and `ep_crsp` that is the equity premium defined as the difference between next year's percentage return on the CRSP value-weighted Index and the riskfree rate. Finally, a scatter plot of the equity premium against the dividend-price ratio at the annual frequency is produced in Figure 3.1. You should be able to reproduce the graph with the code below.

```
library(downloader)    # package to download files
library(readxl)          # package to read xlsx files
library(dplyr)
```

[1] We will spend more time on explaining why there could be a difference between *in-sample* results and *out-of-sample* (real-time). For now know this: *in-sample* means that we calculate the goodness of the predictive relationship on the same data that we use to estimate the parameters, while *out-of-sample* (or real-time if you prefer) means that we estimate the parameters of the relationship on past data but calculate the goodness of the relationship on future data.

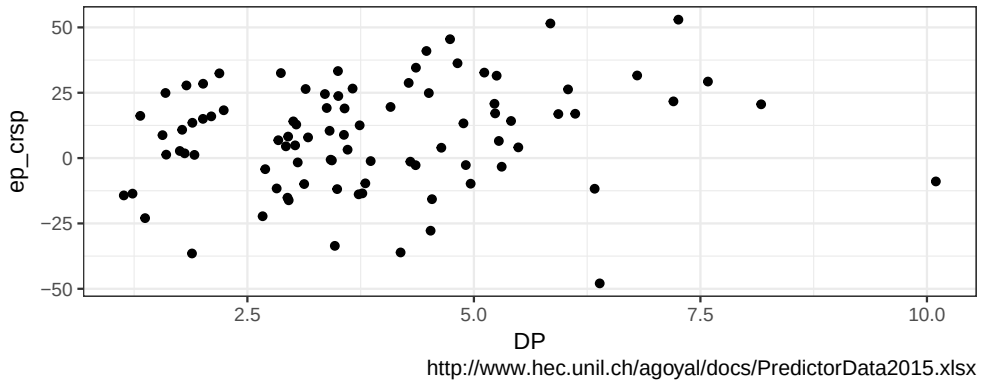


Figure 3.1: Annual observations of the dividend-price (DP) ratio and the equity premium for the CRSP value weighted Index (calculated as the difference between the next year return on the CRSP Value Weighted Index and the riskfree rate).

```
library(lubridate)

# link to the page of Amit Goyal with the data up to 2015; the file is an Excel xlsx file
url       <- "http://www.hec.unil.ch/agoyal/docs/PredictorData2015.xlsx"
file      <- download(url, destfile="goyal_welch.xlsx")
data      <- read_excel("goyal_welch.xlsx", sheet="Annual", na = "NaN")

data <- data %>% mutate(date    = ymd(paste(yyyy,"-01-01",sep="")),
                        DP      = 100 * D12 / Index,
                        ep_crsp = 100 * (lead(CRSP_SPvw - Rfree, 1, order_by=yyyy))) %>%
                dplyr::filter(yyyy >= 1926, yyyy < 2015)

library(ggplot2)
ep.plot <- ggplot(data, aes(DP,ep_crsp)) + geom_point() + theme_bw() + labs(caption=url)
ep.plot
```

So: are returns predictable? It seems that years when the DP ratio was high (e.g., higher than 5%) were followed by years with large positive returns. However, also low values of the DP ratio were followed by years with moderately positive returns, but also some negative returns. It seems that, on average, future returns are higher following years with higher DP ratio. The economic logic supporting this relationship is that a high value of the DP makes the asset attractive to investors that will buy more and put upward pressure on the price that is in denominator of the ratio. Sometimes, scatter plots are difficult to interpret and to extract a clear answer about the relationship between two variables. In these cases, it is useful to draw a regression line through the points. The regression line represents the average equity premium that is expected for a certain value of the dividend-price ratio and can be added to the previous graph with the `geom_smooth()` function as shown below:

```
ep.plot + geom_smooth(method="lm", se=FALSE, color="orange")
```

The regression line is upward sloping which means that increasing values of the dividend-price ratio predict higher returns in the following years. The regression line is equal to -0.04 + 2.16 * DP so that if

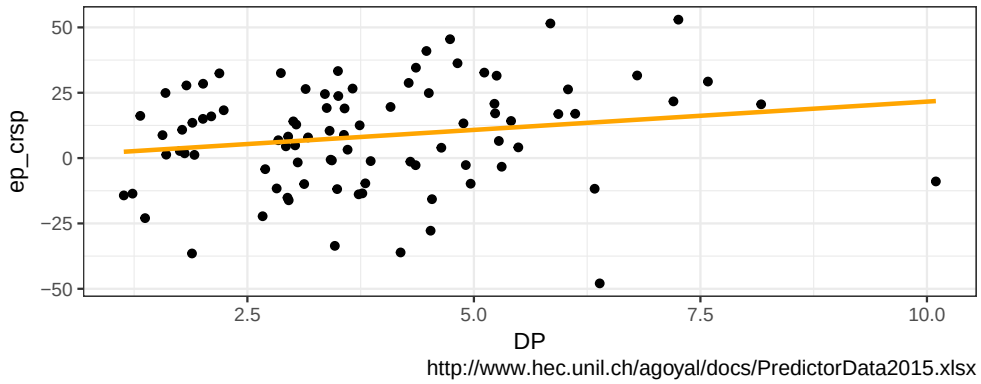


Figure 3.2: Scatter plot of the Dividend-to-Price ratio and the equity premium with the regression line added to the plot.

the DP ratio is equal to 2.5% the model predicts that next year the equity premium will be 5.37% and if the ratio is equal to 5% then the premium is expected to be equal to 10.78%[2]. The fact that the data points are quite distant from the regression line indicates that the DP has some explanatory power for the equity premium, but we do not expect the predictions to be very precise. For example, if you consider the years in which the DP ratio was close to 5%, there have been years in which the premium has been -10% and others as high as 38% despite the regression line predicted 10.78%. Hence, the regression line seems to account for some dependence between the dividend-price ratio and the equity premium, although there is still large uncertainty. Also, it seems a bit puzzling the behavior of the DP below 2.5%. In the years in which the DP was extremely low the equity premium was actually mostly positive between 0 and 25% while there are only a few years with a negative premium.

A scatter plot is a very useful way to visually investigate the relationship between two variables. However, it eliminates the time series characteristics of these variables that could reveal important features in the data. Figure 3.3 shows the dividend-price ratio and the equity premium over time (starting in 1926).

```
plot1 <- ggplot(data, aes(yyyy, DP)) + geom_line() + theme_bw() + labs(x="", y="DP") +
         geom_hline(yintercept = 5, color="violet") + geom_hline(yintercept = 2.5, color="violet")
plot2 <- ggplot(data, aes(yyyy, ep_crsp)) + geom_line() + theme_bw() +
         geom_hline(yintercept = 0, color="tomato3") + labs(x="", y="Equity Premium")
grid.arrange(plot1, plot2)
```

Let's consider first the DP ratio. The ratio was larger than 5% mostly between the mid-1930s and the mid-1950s, fluctuated in the range 2.5-5% until approximately 1995, and since then it has been below 2.5% except for 2008. Notice how values of the ratio below 2.5% never occured before 1995 and were associated, in most cases, with positive premiums and larger than expected based on the regression line. While we see a clear decline in the DP ratio in the latest part of the sample, such a trend is not observable in the equity premium that oscillated approximately between plus/minus 25% throughout the sample. This raises several questions about the relationship between the DP ratio and future returns:

- will the DP ratio ever go back to 5% or more?
- is the historical relationship between the DP ratio and the equity premium a good and reliable

[2] A rule-of-thumb emerging from this model is that next year returns are expected to be twice the end-of-year DP ratio.

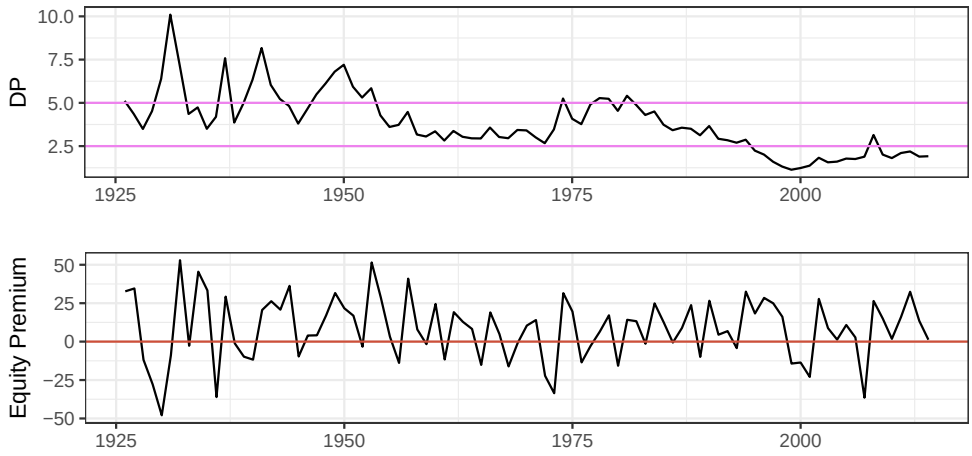


Figure 3.3: Time series of the DP ratio (top) and the equity premium (bottom) from 1926 to 2015.

guidance for the future?

- in other words, did the relationship change over time?

The goal of this Chapter is to review the *Linear Regression Model* (*LRM*) as the basic framework to investigate the relationship between variables. The goal is not to provide a comprehesive review, but rather to discuss the important concepts and discuss several applications to financial data. The most important task in data analysis is to be aware of the many problems that can arise in empirical work that can distort the analysis, such as the effect of outliers and of omitted variables. This Chapter is meant to provide an intentionally light review which will necessarily require that the reader consults more detailed and precise treatments of the LRM as in Stock and Watson (2010) and Wooldridge (2015).

3.1 LRM with one independent variable

The LRM assumes that there is a relationship between a variable Y observed at time t, denoted Y_t, and another variable X observed in the same time period, denoted X_t, and that the relationship is linear, that is,

$$Y_t = \beta_0 + \beta_1 * X_t + \epsilon_t$$

where Y_t is called the dependent variable, X_t is the independent variable, β_0 and β_1 are parameters, and ϵ_t is the error term. Typically, we use subscript t when the variables are observed over time (*time series data*) and subscript i when they vary across different individuals, firms, or countries at one point in time (*cross-sectional* or *longitudinal data*). The aim of the LRM is to explain the variation over time or across units of the dependent variable Y based on the variation of the independent variable X: high values of Y are explained by high (or low) values of X, depending on the parameter β_1. For example, in the case of the equity premium our goal is to understand why in some years the equity premium is positive and large while in other years it is small positive or negative. Another example is the cross-section of stocks in a certain period (month or quarter) and the aim in this case is to understand the characteristics of stocks that drive their future performance. We estimate the LRM by *Ordinary Least Squares* (OLS) which is an estimation method that determines the parameter values such that they minimize the sum of squared

residuals. For the LRM we have analytical formulas for the OLS coefficient estimates. The estimate of the slope coefficient, $\hat{\beta}_1$, is given by

$$\hat{\beta}_1 = \frac{\hat{\sigma}_{X,Y}}{\hat{\sigma}_X^2} = \hat{\rho}_{X,Y}\frac{\hat{\sigma}_Y}{\hat{\sigma}_X}$$

where $\hat{\sigma}_X^2$ and $\hat{\sigma}_Y^2$ represent the sample variances of the two variables, and $\hat{\sigma}_{X,Y}$ and $\hat{\rho}_{X,Y}$ are the sample covariance and the correlation of X and Y, respectively. The sample intercept, $\hat{\beta}_0$, is given by

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 * \bar{X}$$

where $\bar{X}$ and $\bar{Y}$ represent the sample mean of X_t and Y_t. Let's make the formulas operative using the dataset discussed in the introduction to this chapter. The dependent variable is actually Y_{t+1} in this application and represents the annual equity premium (`ep_crsp`) in the following year, while the independent variable X_t is the `DP`, the dividend-price ratio for the current year. To calculate the estimate of the slope coefficient β_1 we need to estimate the covariance of the premium and the ratio as well as the variance of `PD`. We discussed already how to estimate these statistical quantities in `R` in the previous chapter using the `cov()` and `var()` commands:

```
cov(data$ep_crsp, data$DP) / var(data$DP)
```

```
[1] 2.1637
```

The interpretation of the slope estimate is that if the `DP` changes by 1% then we expect the equity premium in the following year to change by 2.164%. The alternative way to calculate $\hat{\beta}_1$ is using the correlation coefficient and the ratio of the standard deviations of the two assets:

```
cor(data$ep_crsp, data$DP) * sd(data$ep_crsp) / sd(data$DP)
```

```
[1] 2.1637
```

This formula shows the relationship between the correlation and the slope coefficients when the LRM has only one independent variables. The correlation between `ep_crsp` and `DP` in the sample is 0.184 and it is scaled up by 11.76 that represents the ratio of the standard deviations of the dependent and independent variables. Notice that if we had standardized both X and Y to have standard deviation equal to 1 then the correlation coefficient would be equal to the slope coefficient. Once the slope parameter is estimated, we can then calculate the sample intercept as follows:

```
mean(data$ep_crsp) - beta1 * mean(data$DP)
```

```
[1] -0.035942
```

The OLS formulas for the estimators of the intercept and slope coefficients and their calculation are programmed in all statistical and econometric software, and even Microsoft Excel can provide you with the calculations. Still, it is important to understand what these numbers represent and how to relate to other quantities that we routinely use to measure dependence. The `R` function `lm()` (for *l*inear *m*odel) estimates the LRM automatically and provides all accessory information that is needed to conduct inference and evaluate the goodness of the model. The following example estimates the LRM with the equity premium as the dependent variable and the `DP` ratio as the independent:

```
# two equivalent ways of estimating a linear model
fit <- lm(data$ep_crsp ~ data$DP)
fit <- lm(ep_crsp ~ DP, data=data)
```

```
Call:
lm(formula = ep_crsp ~ DP, data = data)

Coefficients:
(Intercept)           DP
    -0.0359       2.1637
```

The `fit` object produces only the coefficient estimates and a more detailed report of the relevant statistics can be obtained using the function `summary(fit)` as shown below:

```
summary(fit)
```

```
Call:
lm(formula = ep_crsp ~ DP, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-61.72 -13.26   2.53  13.34  38.89

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.0359     5.2297   -0.01    0.995
DP            2.1637     1.2393    1.75    0.084 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20 on 87 degrees of freedom
Multiple R-squared:  0.0339,    Adjusted R-squared:  0.0227
F-statistic: 3.05 on 1 and 87 DF,  p-value: 0.0844
```

The `summary()` of the `lm` object provides the statistical quantities that are used in testing hypothesis and understand the relevance of the model in explaining the data. The information provided is[3]:

- *Estimate*: the OLS estimates of the coefficients in the LRM
- *Std. Error*: the standard errors represent the standard deviations of the estimates and measure the uncertainty in the sample about the coefficient estimates
- *t value*: the ratio of the estimate and the standard error of a coefficient; it represents the t-statistic for the null hypothesis that the coefficient is equal to zero, that is, $H_0 : \beta_i = 0$ (i=0,1). In large samples the t-statistic has a standard normal distribution and the two-sided critical values at 10, 5, and 1% are 1.64, 1.96, and 2.58, respectively. The null hypothesis that a coefficient is equal to zero against the alternative that is different from zero is rejected when the absolute value of the t-statistic is larger than critical values.
- *Pr(>|t|)*: the p-value represents the probability that the standard normal distribution takes a value larger (in absolute value) than the t-statistic. The null $H_0 : \beta_i = 0$ is rejected in favor of $H_1 : \beta_i \neq 0$ when the p-value is smaller than the significance level (1, 5, 10%) .
- *Residual standard error*: the variance of the residuals $\hat{\epsilon}_t = Y_t - \hat{\beta}_0 - \hat{\beta}_1 * X_t$.
- R^2: a measure of goodness-of-fit that represents the percentage of the variance of the dependent variable that is explained by the model, while $100 - R^2\%$ remains unexplained. The measure ranges

[3]The quantities discussed below should sound familiar and reflect concepts that you understand already. If they are not familiar, please review a introductory econometrics textbook for a more detailed treatment.

from 0 to 1 with 0 meaning that the model is totally irrelevant to explain the dependent variable and 1 meaning that it completely explain it without errors.

- *Adjusted* R^2: this quantity penalizes the R^2 measure for the number of parameters that are included in the regression. This is because R^2 does not decrease when additional independent variables are included in the model. Models with highest adjusted R^2 are preferred.
- *F-statistic*: tests the null hypothesis that all slope parameters are equal to zero against the alternative that at least one is different from zero. The critical value of the F-statistic depends on the number of observations and the number of variables included in the regression.

The regression results show that an increase of the `DP` ratio by 1% (% is the unit of the `DP` variable) leads to an increase of future returns by 2.16% and confirms that higher dividend-price ratio predict higher future returns. The coefficient of the `DP` ratio has a t-statistic of 1.75 and a p-value of 0.08 which means that it is statistically significant at 10% but not at 5%. Hence, we do find evidence indicating that the dividend-price ratio is a significant predictor of future returns, although only at 10% significance level. The goodness-of-fit measure R^2 is equal to 0.034 or 3.4% which is quite close to its lower bound of zero and indicates that the `DP` ratio has small predictive power for the equity premium. We thus find that returns are predictable, although the predictive content of the variable we used (the dividend-price ratio) is small. Maybe there could be other more powerful predictors of the equity premium, and the search continues.

Based on the coefficient estimates, we can then calculate the *fitted values* and the *residuals* of the regression model. The fitted values are calculated as $\hat{\beta}_0+\hat{\beta}_1*DP_t$ and represent the prediction of the equity premium in year $t+1$ based on the `DP` ratio of year t. We can denote the fitted values (or expected or predicted) as $E(EP_{t+1})$ where $E(\cdot)$ is the expectation and EP_{t+1} represents the equity premium in the following year. The residuals are the estimated errors and are obtained by subtracting the predicted equity premium to the value that actually realized, that is, $\hat{\epsilon}_{t+1} = EP_{t+1} - E(EP_{t+1})$. Figure 3.4 shows the time series of the equity premium (`orange` line) and the forecast from the model (`red` line), while the bottom graph shows the residuals that represent the distance between the realized and fitted equity premium in the top graph.

```
plot1 <- ggplot(data) + geom_line(aes(yyyy, ep_crsp), color="orange") +
                        geom_line(aes(yyyy, fit$fitted.values), color="red") +
                        theme_bw() + labs(x="", y="Premium & Fitted")
plot2 <- ggplot(data) + geom_line(aes(yyyy, fit$residuals), color="royalblue2") +
                        theme_bw() + labs(x="",y="Residuals") +
                        geom_hline(yintercept=0, color="slategray4")
grid.arrange(plot1, plot2)
```

Let's first look at the top graph. The `red` line (predicted equity premium) seems quite flat relative to the `orange` line (realized equity premium). The goal of the model is to produce predictions that are as close as possible to the realized values. Unfortunately, in this case it seems that the predictions of the model are not very informative about the following year equity premium. This makes visually clear the implications of a low R^2. If the R^2 was equal to zero the `red` line would have been flat and, at the other extreme, if R^2 was equal to one then the `red` line would overlap with the `orange` line[4]. In this case, the

[4] R^2 equal to 1 means that the sum of squared errors is equal to zero that is only possible when the fitted equity premium is equal to the realized equity premium.

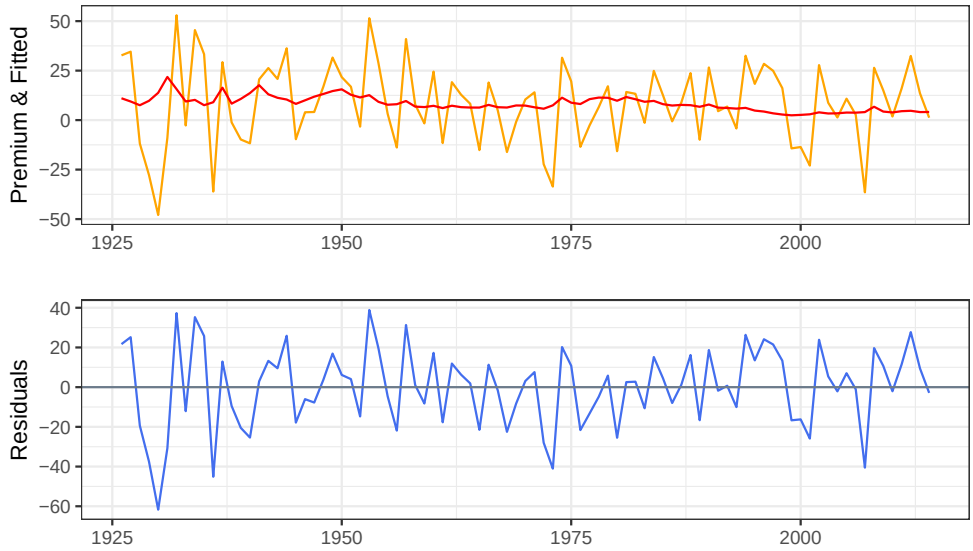


Figure 3.4: Time series of the realized and predicted equity premium (top) and the residuals obtained as the difference between the realized and predicted equity premium (bottom).

predicted equity premium varies over time in response to changes in the DP ratio, but hardly enough to keep track of the changes of the realized equity premium in any useful way.

3.2 Robust standard errors

The standard errors calculated by the `lm()` function are based on the assumption that the errors ϵ_t have constant variance and, for time series data, that they are independent over time. In practice, both assumptions might fail. The variance or standard deviation of the errors might vary over time or across individuals and errors might be dependent and correlated over time. For example, an indication of dependence in the error is when positive errors are more (less) likely to be followed by positive (negative) errors as opposed to be equally likely. Calculating the standard errors assuming homogeneity and independence when in fact the errors are heteroskedastic and/or dependent is that, typically, the standard errors are smaller than they should be to make correct inference. The solution is to *correct* the standard errors using Heteroskeasticity Corrected (*HC*) standard errors in cross-sectional regressions and using Heteroskedasticity and Autocorrelation Corrected (*HAC*) standard errors when dealing with time series data as proposed by Newey and West (1987). A practical rule is use corrected standard errors by default and in case the residuals are homoskedastic and independent over time there is only a small loss of precision in small samples. The code below shows how to calculate Newey-West HAC standard errors for a `lm` object.

```
library(sandwich)  # function NeweyWest() to calculates HAC standard errors
library(lmtest)    # function coeftest() produces a table of results with HAC s.e.
fit <- lm(ep_crsp ~ DP, data=data)
summary(fit)$coefficients
coeftest(fit, df = Inf, vcov = NeweyWest(fit, prewhite = FALSE))
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.036        5.2 -0.0069    0.995
DP             2.164        1.2  1.7459    0.084

z test of coefficients:

            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.0359     4.9993   -0.01     0.99
DP            2.1637     1.3173    1.64     0.10
```

Notice that:

- The coefficient estimates are the same since they are not biased by the presence of heteroskedasticity and auto-correlation in the errors
- The standard error for `DP` and the intercept increase slightly. This implies that the t-statistic for `DP` decreases from 1.75 to 1.64 and the p-value increases from 0.08 to 0.1. The `DP` is still significant at 10% level but it becomes a even more marginal case.

As mentioned earlier, when dealing with time series data it is good practice to estimate HAC standard errors by default which is what we will do in the rest of this book.

3.3 Functional forms

In the example above we considered the case of a linear relationship between the independent variable X and the dependent variable Y. However, there are situations in which the relationship between X and Y might not be well-explained by a linear model. This can happen, e.g., when the effect of changes of X on the dependent variable Y depends on the level of X. In this Section, we discuss two functional forms that are relevant in financial applications.

One functional form that can be used to account for nonlinearities in the relationship between X and Y is the quadratic model, which simply consists of adding the square of the independent variable as an additional regressor. The Quadratic Regression Model is given by

$$Y_t = \beta_0 + \beta_1 * X_t + \beta_2 * X_t^2 + \epsilon_t$$

that, relative to the linear model, adds some curvature to the relationship through the quadratic term. The model can still be estimated by OLS and the expected effect of a one unit increase in X is now given by $\beta_1 + 2\beta_2 X_t$. Hence, the effect on Y of changes in X is a function of the level of the independent variable X, while in the linear model the effect is β_1 no matter the value of X. Polynomials of higher order can also be used, but care needs to be taken since the additional powers of X_t are strongly correlated with X_t and X_t^2. The cubic regression model is thus given by:

$$Y_t = \beta_0 + \beta_1 * X_t + \beta_2 * X_t^2 + +\beta_3 * X_t^3 + \epsilon_t$$

The second type of nonlinearity that can be easily introduced in the LRM assumes that the slope coefficient of X is different below or above a certain threshold value of X. For example, if X_t represents the market return we might want to evaluate if there is a different (linear) relationship between the stock and the market return when the market return is, e.g., below or above the mean/median. We can define two new variables as $X_t * I(X_t \geq m)$ and $ X_t * I(X_t < m)$, where $I(A)$ is the indicator function which takes

value 1 if the event A is true and zero otherwise, and m is a threshold value to define the variable above and below. The model is given by:

$$Y_t = \beta_0 + \beta_1 X_t * I(X_t \geq m) + \beta_2 X_t * I(X_t < m) + \epsilon_t$$

where the coefficients β_1 and β_2 represent the effect on the dependent variable of a unit change in X_t when X_t is larger or smaller than the value m. Another way of specifying this model is by including the regressor X_t and only one of the two interaction effects:

$$Y_t = \gamma_0 + \gamma_1 X_t + \gamma_2 X_t * I(X_t \geq m) + \epsilon_t$$

In this case the coefficient γ_2 represents the differential effect of the variable X when it is below or above the value m. Testing the null hypothesis that $\gamma_2 = 0$ represents a way to evaluate whether there is a nonlinear relationship between the two variables.

3.3.1 Application: Are hedge fund returns nonlinear?

An interesting application of nonlinear functional forms is to model hedge fund returns. As opposed to mutual funds that mostly hold long positions and make limited use of financial derivatives and leverage, hedge funds make extensive use of these instruments to hedge downside risk and boost their returns. This potentially produces a nonlinear response to changes in market returns which cannot be captured by a linear model. As a simple example, assume that a hedge fund holds a portfolio composed of one call option on a certain stock. The performance/payoff will obviously have a nonlinear relationship to the price of the underlying asset and will be better approximated by either of the functional forms discussed above. We can empirically investigate this issue using the Credit Suisse Hedge Fund Indexes[5] that provide the overall and strategy-specific performance of a portfolio of hedge funds. Indexes are available for the following strategies: convertible arbitrage, dedicated short bias, equity market neutral, event driven, global macro and long-short equity among others. Since individual hedge fund returns data are proprietary, we will use these Indexes that could be interpreted as a sort of fund-of-funds of the overall universe or for specific strategies of hedge funds.

The file `hedgefunds.csv` contains the returns of 14 Credit Suisse Hedge Fund Indexes (the HF index and 13 strategies indexes) starting in January 1993 at the monthly frequency. Figure 3.5 shows the change in NAV for these indexes, with some strategies performing extremely well (e.g., global macro) and other performing rather poorly (e.g., dedicated short bias). All strategies seem to have experienced a decline during the 2008-2009 recession, although some strategies were affected to a lesser extent.

```
hfret      <- read_csv('hedgefunds.csv')
hfret$date <- as.yearmon(mdy(hfret$date))   # format the date to match the format of factor
sp500      <- dplyr::select(mydata, date, GSPC)  # merge with S&P 500
sp500$GSPC <- 100 * sp500$GSPC / sp500$GSPC[sp500$date == "Dec 1993"] # make Dec 1993 equal to 100
hfret      <- inner_join(hfret, sp500, by="date") # join the hfret and sp500 in one data frame
# reorganize the data in e columns: date, Strategy, NAV (makes it easier to use ggplot2)
hfret      <- arrange(hfret, date) %>% tidyr::gather(Strategy, NAV, -date)
```

[5] The data is available at the website http://www.hedgeindex.com upon registration.

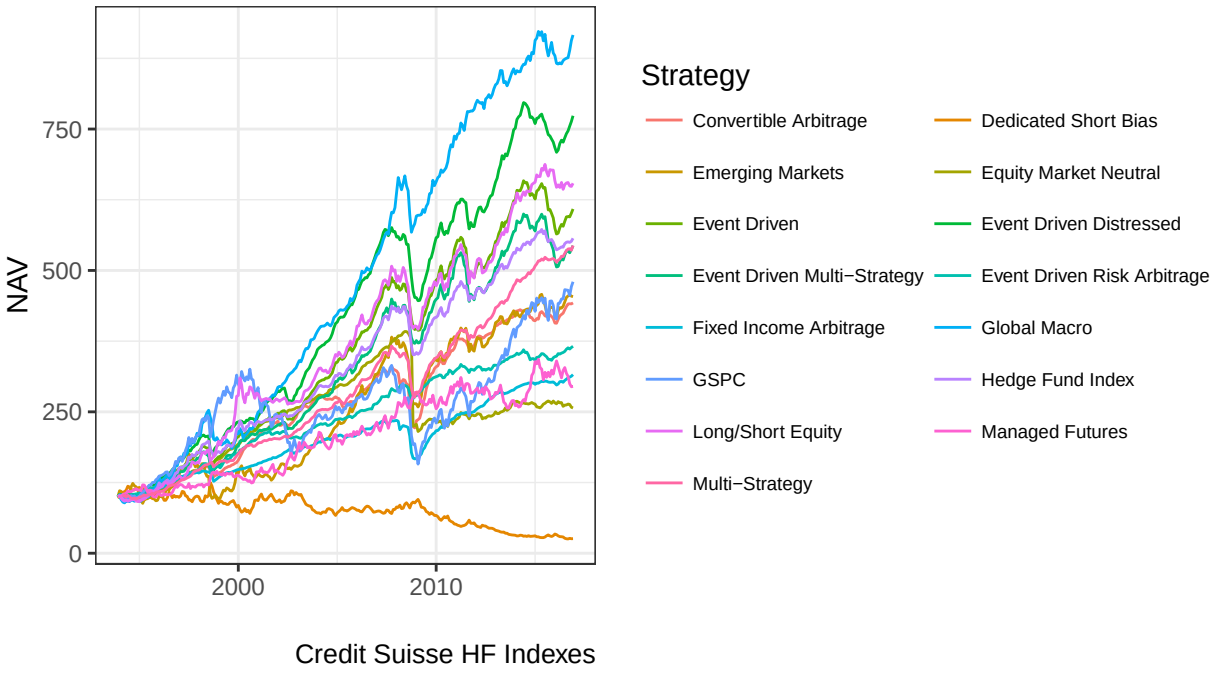


Figure 3.5: Net Asset Value (NAV) of the Credit Suisse HF Indexes starting in December 1993. In this graph the NAV of all strategies is standardized at 100 in December 1993.

```
ggplot(hfret, aes(date, NAV, color=Strategy)) +
  geom_line() + theme_bw() + labs(x="", caption="Credit Suisse HF Indexes") +
  theme(legend.text=element_text(size=7)) + guides(col = guide_legend(nrow = 8, byrow = TRUE))
```

Alternatively, we can evaluate and compare the descriptive statistics of the strategies by calculating the mean, standard deviation, skewness, and kurtosis for the monthly returns defined as the percentage change of the NAV. In the code below I use the `skewness` and `kurtosis` functions from the `e1071` package and format the results as a table using the `kable` function from the `knitr` package.

```
hfret <- hfret %>%
         group_by(Strategy) %>%
         mutate(RET = 100 * log(NAV / lag(NAV))) %>%
         dplyr::filter(!is.na(RET))
strategy.table <-hfret %>%
                 group_by(Strategy) %>%
                 summarize(AV = mean(RET),
                           SD = sd(RET),
                           SKEW = e1071::skewness(RET),
                           KURT = e1071::kurtosis(RET),
                           MIN  = min(RET), MAX = max(RET))
knitr::kable(strategy.table, digits=3, caption="Summary statistics for the HF strategy returns
             from the Credit Suisse Hedge Fund Indexes starting in December 1993.")
```

Some facts that emerge from this table:

Table 3.1: Summary statistics for the HF strategy returns from the Credit Suisse Hedge Fund Indexes starting in December 1993.

Strategy	AV	SD	SKEW	KURT	MIN	MAX
Convertible Arbitrage	0.54	1.9	-3.003	20.36	-13.5	5.6
Dedicated Short Bias	-0.49	4.6	0.521	0.81	-12.0	20.5
Emerging Markets	0.55	4.0	-1.255	8.39	-26.2	15.2
Equity Market Neutral	0.34	3.3	-13.735	210.97	-51.8	3.6
Event Driven	0.65	1.8	-2.216	11.16	-12.5	4.1
Event Driven Distressed	0.74	1.8	-2.281	13.15	-13.3	4.1
Event Driven Multi-Strategy	0.61	1.9	-1.768	7.78	-12.2	4.7
Event Driven Risk Arbitrage	0.47	1.1	-0.990	4.87	-6.4	3.7
Fixed Income Arbitrage	0.42	1.6	-5.057	40.80	-15.1	4.2
Global Macro	0.80	2.6	-0.126	4.82	-12.3	10.1
GSPC	0.57	4.3	-0.840	1.66	-18.6	10.2
Hedge Fund Index	0.62	2.0	-0.263	3.15	-7.8	8.2
Long/Short Equity	0.68	2.6	-0.201	3.83	-12.1	12.2
Managed Futures	0.39	3.3	-0.046	-0.15	-9.8	9.5
Multi-Strategy	0.62	1.4	-1.838	7.44	-7.6	4.2

- The best performing strategy is `Global Macro` with an average return of 0.8% monthly and the worst performing strategy is `Dedicated Short Bias` that realized an average return of -0.49%. As a comparison, the S&P 500 Index in the same period had a average return of 0.57% and the `Hedge Fund Index` of 0.62%
- In terms of volatility, the standard deviation of the S&P 500 in this period was 4.28% monthly and the `Hedge Fund Index` of 2. The least volatile HF strategy was `Event Driven Risk Arbitrage` (1.14%) and the most volatile was `Dedicated Short Bias` (4.65%)
- Overall, it seems that most strategies provide a better risk-return ratio relative to investing in the S&P 500 by providing higher returns per unit of volatility.
- Most strategies have negative skewness and positive excess kurtosis that indicate that the distribution of returns are left-skewed and fat tailed. This could be due to large negative returns due to significant declines in NAV. Some values of the kurtosis are quite extreme as in the case of the `Equity Market Neutral` strategy with a value of 210.97. This is an indication that some outliers might have occured in the sample and that further examination is required as discussed later in the Chapter. The column `MIN` and `MAX` show that `Equity Market Neutral` experienced a loss of 51.84% in one month while the largest gain was 3.59%. In analyzing this dataset we will have to consider carefully whether these extreme observations might be considered outliers and have an effect on our analysis.

Are the HF returns sensitive to movements in the U.S. equity market? To evaluate graphically this question we can do a scatter plot of the HF Index return against the S&P 500. Figure 3.6 shows that there seems to be positive correlation between these two variables, although the most striking feature of the plot is the difference in scale between the x- and y-axis: the HF returns range between $\pm 8\%$ while

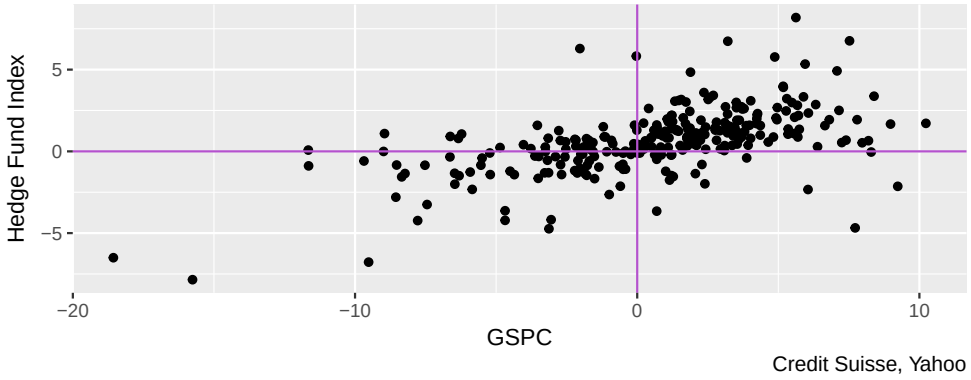


Figure 3.6: Scatter plot of the monthly returns of the SP 500 Index and the CSHedge Fund Index starting in December 1993.

the equity index between -20% and 12%. The standard deviation of the HF index is 2% compared to 4.28% for the S&P 500 Index which shows that hedge funds provide a hedge against large movements in markets. Notice that the code to make Figure 3.6 keeps the column names to be the strategy names given by Credit Suisse. This is mostly driven by the intent to make the code easier to read and more transparent. In practice, it is more convenient to rename the columns to some shorter and faster to type names (e.g., `HFI`). Below is the code to produce Figure 3.6.

```
hfret1 <- hfret %>% dplyr::select(-NAV) %>% tidyr::spread(Strategy, RET)

ggplot(hfret1, aes(GSPC, `Hedge Fund Index`)) + geom_point() +
  geom_vline(xintercept=0, color="mediumorchid3") +
  geom_hline(yintercept=0, color="mediumorchid3") +
  labs(caption = "Credit Suisse, Yahoo")
```

Before introducing non-linearities, let's estimate a linear model in which the HF index return is explained by the S&P 500 return. The results below show the existence of a statistically significant relationship between the two returns. The exposure of the HF return to S&P 500 fluctuations is 0.27 which means that if the market return changes by $\pm 1\%$ then we expect the fund return to change by $\pm\ 0.27\%$. The R^2 of the regression is 0.33, which is not very high for this type of regressions. A (relatively) *low* R^2 in this case is actually good news for hedge funds since most strategies promise to provide low (if none) exposure to market fluctuation. If this is case then a low goodness-of-fit statistic is good news. We can add the fitted linear relationship given by $0.47 + 0.27 R_t^{SP500}$ to the previous scatter plot to have a graphical understanding of the LRM:

```
fitlin     <- lm( `Hedge Fund Index` ~ GSPC, data=hfret1)
coeftest(fitlin, vcov=NeweyWest(fitlin, prewhite = FALSE))
```

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.4695     0.1162    4.04  7.0e-05 ***
GSPC          0.2687     0.0327    8.23  7.9e-15 ***
---
```

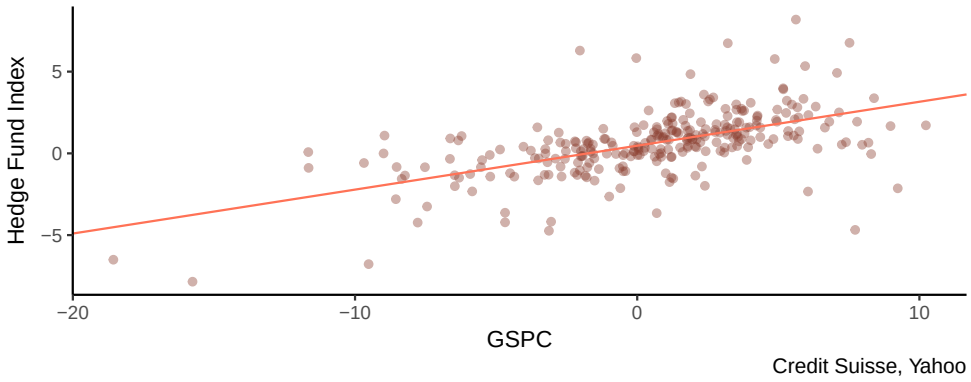


Figure 3.7: Scatter plot of the SP 500 and the HF Index together with the regression line obtained from the fitlin object.

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ggplot(hfret1, aes(GSPC, `Hedge Fund Index`)) +
  geom_point(color="coral4", alpha=0.4) +
  geom_abline(intercept = coef(fitlin)[1], slope = coef(fitlin)[2], color="coral1") +
  theme_classic() + labs(caption = "Credit Suisse, Yahoo")
```

To estimate a quadratic model we need to add the quadratic term to the linear regression. This can be done adding the term `I(GSPC^2)` where `I()` is used to add transformations of variables in the formula of the `lm()` function:

```
fitquad <- lm(`Hedge Fund Index` ~ GSPC + I(GSPC^2), data=hfret1)
coeftest(fitquad, vcov=NeweyWest(fitquad, prewhite = FALSE))
```

```
t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.61540    0.11323    5.44  1.2e-07 ***
GSPC         0.25064    0.03540    7.08  1.2e-11 ***
I(GSPC^2)   -0.00729    0.00423   -1.73    0.086 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the coefficient of the square term is statistically significant at 10% level we conclude that a nonlinear form is a better model to explain the relationship between the market and fund returns. In this case, we find that the p-value for the null hypothesis that the coefficient of the square market return is equal to zero is 0.09 (or 8.6%) which is smaller than 0.10 (or 10%) and we thus conclude that there is significant evidence of nonlinearity in the relationship between the HF and the market return. The effect of a $\pm$ 1% change in the S&P 500 is given by 0.25 + (-0.01)R_t^{SP500}.The fact that the coefficient of the quadratic term is negative implies that the parabola 0.62 + 0.25R_t^{SP500} + -0.01$(R_t^{SP500})^2$ will lay below the line 0.62 + 0.25(R_t^{SP500}). So, for large (either positive or negative) returns the parabola implies expectations of returns that are significantly lower relative to the line. Instead, if the coefficient of the quadratic term is positive then the parabola would lay above the line and imply that hedge fund returns become less sensitive (or even profit) from large movements in the market.

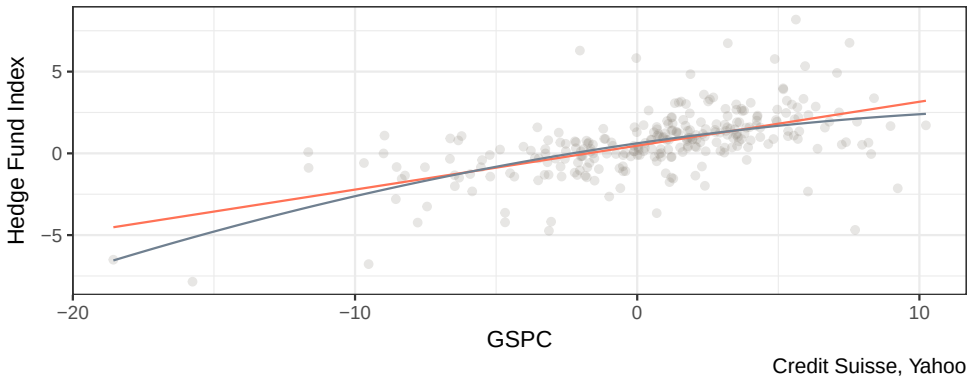


Figure 3.8: Scatter plot of the SP 500 and the HF index returns with linear and quadratic regression line.

In Figure 3.8 we see that the contribution of the quadratic term (dashed line) becomes apparent at the extremes (large absolute market returns), while for small returns it is close or overlaps with the fitted values of the linear model. In terms of goodness-of-fit, the adjusted R^2 of the quadratic regression is 0.34 which is a modest increase relative to 0.33 for the linear model.

```
ggplot(hfret1, aes(GSPC, `Hedge Fund Index`)) +
  geom_point(color="antiquewhite4", alpha=0.2) +
  stat_function(fun = function(x) coef(fitlin)[1] + coef(fitlin)[2]*x, color="coral1")  +
  stat_function(fun = function(x) coef(fitquad)[1] + coef(fitquad)[2]*x
                + coef(fitquad)[3]*x^2, color="slategrey")  +
  theme_bw() + labs(caption = "Credit Suisse, Yahoo")
```

The second type of nonlinearity that was discussed earlier assumes that the relationship between dependent and independent variables is linear but with different slopes below and above a certain value of the independent variable. In the example below we consider the median value of the independent variable as the threshold value (in this sample the median value of the S&P 500 is 1.11%). There are two equivalent ways to specify model[6] for estimation by including in the regression:

- the `GSPC` return and the interaction term `I(GSPC * (GSPC < m)`
- the `I(GSPC * (GSPC < m)` and `I(GSPC * (GSPC >= m)`

To avoid perfect multicollinearity we need to avoid to include the variable (e.g., `GSPC`) and both of its transformations in the regression since the model cannot be estimated. The `lm()` function in these cases drops one of the regressors and estimates the model on the remaining ones. One advantage of the first specification is that the coefficient of the term `I(GSPC * (GSPC < m))` represents the difference in the effect of a change of the independent variable on the dependent. If the null hypothesis that it is equal to zero is not rejected than the data indicate that the nonlinear model is not needed and we can continue the analysis with the linear model.

```
m             <- median(hfret1$GSPC, na.rm=T)
fit.piecewise <- lm( `Hedge Fund Index` ~ GSPC + I(GSPC * (GSPC < m)), data=hfret1)
coeftest(fit.piecewise, vcov=NeweyWest(fit.piecewise, prewhite = FALSE))
```

[6]This specification is called *piece-wise* linear in the sense that it is linear in a certain range of the independent variable.

```
t test of coefficients:

                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            0.6271     0.1380    4.54  8.3e-06 ***
GSPC                   0.2152     0.0624    3.45  0.00066 ***
I(GSPC * (GSPC < m))   0.0967     0.0915    1.06  0.29131
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fit.piecewise <- lm( `Hedge Fund Index` ~  I(GSPC *(GSPC < m)) + I(GSPC * (GSPC >= m)), data=hfret1)
coeftest(fit.piecewise, vcov=NeweyWest(fit.piecewise, prewhite = FALSE))
```

```
t test of coefficients:

                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             0.6271     0.1405    4.46  1.2e-05 ***
I(GSPC * (GSPC < m))    0.3119     0.0517    6.03  5.3e-09 ***
I(GSPC * (GSPC >= m))   0.2152     0.0635    3.39  0.00081 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The two specifications achieve the same result in term of R^2 (equal to 0.34) and the parameters of one model can be mapped into the parameters of the other model. The slope coefficient (or market exposure) of the HF Index below the median is 0.31 and above is 0.22, which suggests that the HF Index is more sensitive to downward movements of the market relative to upward movements. Instead, in the first regression the results show that a 1% change in the S&P 500 causes a 0.22 change, but the index return is below the median, that is `I(GSPC < m)=1`, then the effect is 0.31 which is obtained by summing 0.22 and 0.1. To evaluate statistically if the nonlinear specification is useful or not we can simply test whether the coefficient of `I(GSPC * (GSPC < m))` is equal to zero against the alternative that it is different from zero. The pvalue for this hypothesis is 0.29and even at 10% we do not reject the null hypothesis that it is equal to zero. Hence, this nonlinear specification does not seem to be supported by the data, as opposed to the quadratic model. The shape of the regression line for this model and for the quadratic model are very similar as shown in Figure 3.9, although the quadratic model is marginally superior to the linear model by having a significant coefficient in the square term, and by achieving higher adjusted R^2.

```
mat <- data.frame(X = (fit.piecewise$model[,2]+fit.piecewise$model[,3]), fitted = fitted(fit.piecewise))
mat <- arrange(mat, fitted)

ggplot(hfret1, aes(GSPC, `Hedge Fund Index`)) +
  geom_point(color="antiquewhite4", alpha=0.3) +
  geom_smooth(method="lm", se=FALSE, color="coral1") +
  geom_line(data=mat, aes(X,fitted),color="purple", size=1) +
  theme_bw() + labs(caption = "Credit Suisse, Yahoo")
```

3.4 The role of outliers

One of the strategies provided by Credit Suisse is the equity market neutral strategy that aims at providing positive expected return, with low volatility and no correlation with the equity market. Figure 3.10

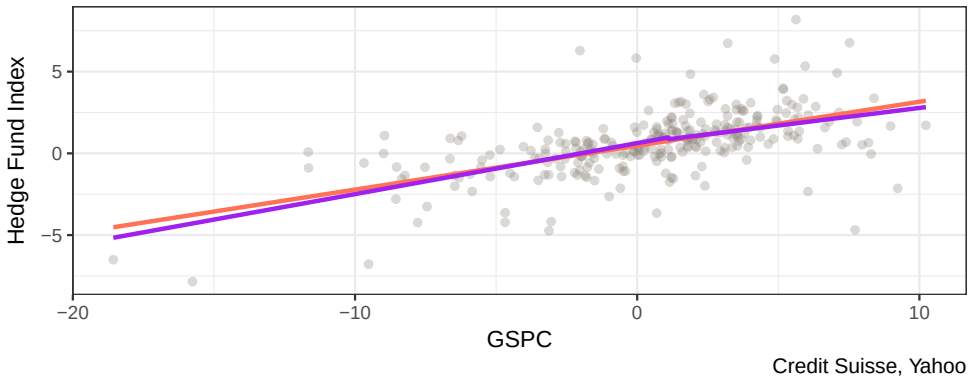


Figure 3.9: Scatter plot of the SP 500 return and the Hedge Fund Index together with the linear regression line and the piece-wise or threshold linear model.

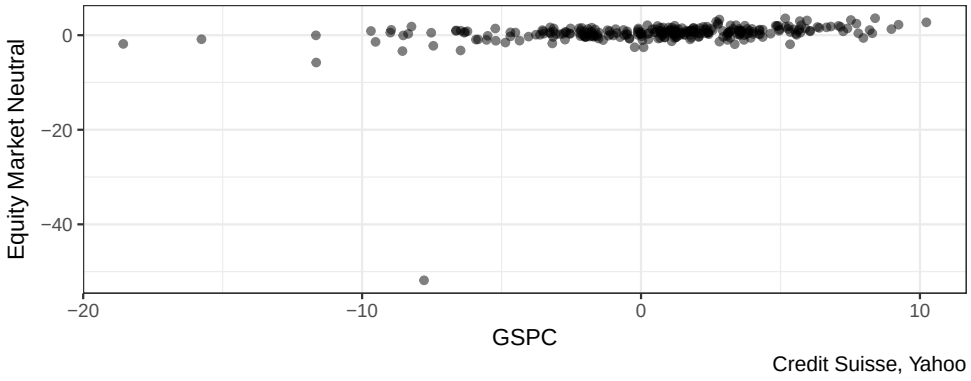


Figure 3.10: Scatter plot of the SP 500 returns and the HF Equity Market Neutral returns.

represents a scatter plot of the market return (proxied by the S&P 500) and the HF equity market neutral return.

```
ggplot(hfret1, aes(GSPC, `Equity Market Neutral`)) +
  geom_point(alpha=0.5) + theme_bw() + labs(caption = "Credit Suisse, Yahoo")
```

What is wrong with Figure 3.10? Did we do a mistake in plotting the variables? No, the only issue with the graph is the large negative return of -51.84% for the HF strategy relative to a loss for the S&P 500 of "only" 7.78%. To find out when the extreme observation occurred, we can use the command `which.min(hfret1$'Equity Market Neutral')` which indicates that it represents the 179th observation. What happened in November 2008 to create a loss of 51.84% to an aggregate index of market neutral strategy hedge funds? In a press release Credit Suisse discusses that they marked down to zero the assets of the Kingate Global Fund, which was a hedge fund based on the British Virgin Islands that acted as a feeder for the Madoff funds and was completely wiped out when the Ponzi scheme operated by Bernard Madoff was discovered. Does this extreme observation or outlier have an effect on our conclusions and our assessment of the equity market neutral strategy? Let's consider how the descriptive statistics in Table 3.1 would change if the outlier is excluded:

- the mean would increase from 0.34% to 0.53%

- the standard deviation would decline from 3.35% to 1.13%
- the skewness would change from -13.74 to 0.53
- the excess kurtosis would change from 210.97 to 3.97

These results make clear that there is a significant effect of the outlier in distorting the descriptive statistics. Statistically speaking, the estimates might be biased because the outlier has the effect of pushing away the sample estimates from their population values. This is an important issue in practice because we use these quantities to compare the risk-return tradeoff of different assets and also because we wish to predict the expected future return from investing in such a strategy. Should we include or exclude the outlier when calculating quantities that are the basis for investment decisions? This choice depends on our interpretation of the nature of the extreme observation: is it an intrinsic feature of the process to produce outliers occasionally or can it be attributed to a one-time event that is unlikely to happen again? In the current situation we need to assess whether another Ponzi scheme of the magnitude operated by Bernard Madoff can occur in the future and lead to the liquidation of a large equity market neutral hedge fund. It is probably fair to say that the circumstances that led to the 51.84% loss were so exceptional that it is warranted to simply drop that observation from the sample when estimating the model parameters.

In addition to creating bias in the descriptive statistics, outliers have also the potential to bias the coefficient estimates of the LRM. The code below shows the estimation results for the model $R_t^{EMN} = \beta_0 + \beta_1 R_t^{SP500} + \epsilon_t$, where R_t^{EMN} and R_t^{SP500} are the returns of the equity market neutral and S&P 500 returns. The first regression model includes all observations while the second drops the outlier:

```
fit0 <- lm(`Equity Market Neutral` ~ GSPC, data=hfret1)
coeftest(fit0, vcov=NeweyWest(fit0, prewhite = FALSE))
```

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.2261     0.2376    0.95    0.342
GSPC          0.2034     0.0821    2.48    0.014 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fit1 <- lm(`Equity Market Neutral` ~ GSPC, data = subset(hfret1, date != "Nov 2008"))
coeftest(fit1, vcov=NeweyWest(fit1, prewhite = FALSE))
```

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.4606     0.0937    4.91  1.5e-06 ***
GSPC          0.1184     0.0252    4.69  4.3e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

These results indicate that by dropping the November 2008 return the estimate of β_0 increases from 0.23 to 0.46 while the market exposure of the fund declines from 0.2 to 0.12. Hence, the effect of the large negative return is to depress the estimate of the interecept (interpreted as the risk-adjusted return) and to increase the slope (the HF exposure to the market return). However, even after removing the outlier the exposure β_1 is statistically different from zero at 1%, which indicates that the aggregate index has

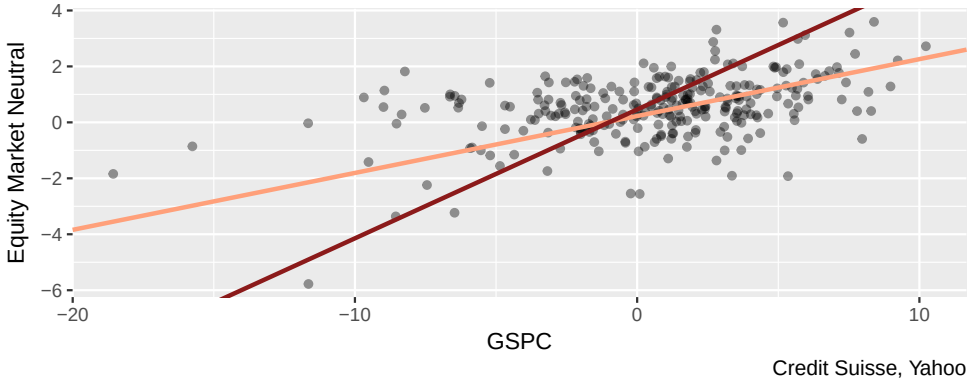


Figure 3.11: Scatter plot of the Equity Market Neutral Index and the SP 500 returns excluding the November 2008 observations. The regression lines are estimated with/without the extreme observation.

low but significant exposure to market fluctuations. The outlier has also an effect on the goodness-of-fit statistic since the R^2 increases from 0.07 to 0.2 when the extreme observation is excluded from the sample. Figure 3.11 shows the scatter plot of `GSPC` and `Equity Market Neutral` together with the regression lines estimated above. In this graph the observations for November 2008 is dropped and clearly the `lightsalmon1` line seems to be closer to the points relative to the `firebrick4` line that is estimated including the outlier.

```
c0 <- coef(fit0)
c1 <- coef(fit1)
dplyr::filter(hfret1, `Equity Market Neutral` > -20) %>%
  ggplot(., aes(GSPC, `Equity Market Neutral`)) + geom_point(alpha=0.4) +
  geom_abline(intercept = c0[1], slope = c0[2], color="lightsalmon1", size = 1) +
  geom_abline(intercept = c1[1], slope = c1[1],color="firebrick4", size = 1)  +
  labs(caption = "Credit Suisse, Yahoo")
```

3.5 LRM with multiple independent variables

In practice, we might be interested to include more than just one variable to explain the dependent variable Y. Denote the K independent variables that we are interested to include in the regression by $X_{k,t}$ for $k = 1, \cdots, K$. The linear regression with multiple regressors is defined as

$$Y_t = \beta_0 + \beta_1 * X_{1,t} + \cdots + \beta_K * X_{K,t} + \epsilon_t$$

Also in this case we can use OLS to estimate the parameters β_k (for $k = 1, \cdots, K$) by choosing the values that minimize the sum of the squared residuals. Care should be given to the correlation among the independent variables to avoid cases of extremely high dependence. The case of correlation among two independent variables equal to 1 is called *perfect collinearity* and the model cannot be estimated. This is because it is not possible to associate changes in Y_t with changes in $X_{1,t}$ or $X_{2,t}$ since the two independent variables have correlation one and move in the same direction and by a proportional amount. A similar situation arises when an independent variable has correlation 1 with a linear combination of the

independent variables[7]. The solution in this case is to exclude one of the variables from the regression. In practice, it is more likely to happen that the regressors have very high correlation although not equal to 1. In this case the model can be estimated but the coefficient estimates become unreliable. For example, equity markets move together in response to news that affect the economies worldwide. So, there are significant co-movements among these markets and thus high correlation which can become problematic in some situations. Before estimating the LRM, the correlation between the independent variables should be estimated to evaluate whether there are high correlations that might make impossible or unreliable to estimate the model. Correlations are high when they are larger than 0.85 and you should start assessing if these variables are both needed in the regression. A practical way to assess the effect of these correlations is to estimate the LRM with both variables, and then excluding one of them and including the other. By comparing the adjusted R^2 and the stability of the coefficient estimates and the standard errors should give an answer whether it is the case to include both or just one of the variables.

In `R` the LRM with multiple regressors is estimated using the `lm()` command discussed before. To illustrate the LRM with multiple regressors I will extend the earlier market model to a 3-factor model in which there are two more independent variables or factors to explain the variation over time of the returns of a portfolio. The factors are called the *Fama-French* factors after the two economists that first proposed these factors to model risk in portfolios. In addition to the market return (MKT), they construct two additional risk factors:

- *Small minus Big* (SMB) which is defined as the difference between the return of a portfolio of small capitalization stocks and a portfolio of large capitalization stocks. The aim of this factor is to capture the premium from investing in small cap stocks.
- *High minus Low* (HML) is obtained as the difference between a portfolio of high Book-to-Market (B-M) ratio stocks and a portfolio of low B-M stocks. The high B-M ratio stocks are also called *value* stocks while the low B-M ratio ones are referred to as *growth* stocks. The factor provides a proxy for the premium from investing in value relative to growth stocks.

More details about the construction of these factors are available at Ken French webpage where you can also download the data for the MKT, SMB, and HML factors and the risk-free rate (RF). The dataset starts in July 1926 and ends in June 2017 and Figure 3.12[8] shows the time series plot of the 3 factors and the risk-free rate.

```
plot.zoo(factors)
```

The data is in a `xts`-object called `factors` that has 1092 rows and 4 columns. To calculate the `mean()` and `sd()` of each column of the object we can use the `apply()` function that applies a function specified by the users to the rows or the columns of a data frame (specifying the argument `MARGIN` equal to 1 for

[7]One common situation in which this happens is when dummy variables are used in the regression. For example, assume that we are analyzing daily data and we create 7 dummy variables called `Monday` which is 1 every Monday and zero otherwise, and similarly for the other days of the week. If we include in the regression the intercept and the seven daily dummy variables for each day the sum of the dummy variables equals 1 which is the constant (think of the intercept as 1 times a constant value β_0). There are two solutions to this problem: 1) include the intercept β_0 and drop one of the sevel dummy variables, or 2) exclude the intercept and include all seven dummy variables. If you include the intercept and the seven dummy variables the command `lm()` does not provide an error message bu automatically drops one of the dummy variables and then estimate the model.

[8]The plot is made using the `plot.zoo()` since this function (contrary to `plot.xts`) produces a graph with a time series plot for each variable in the object (up to 10).

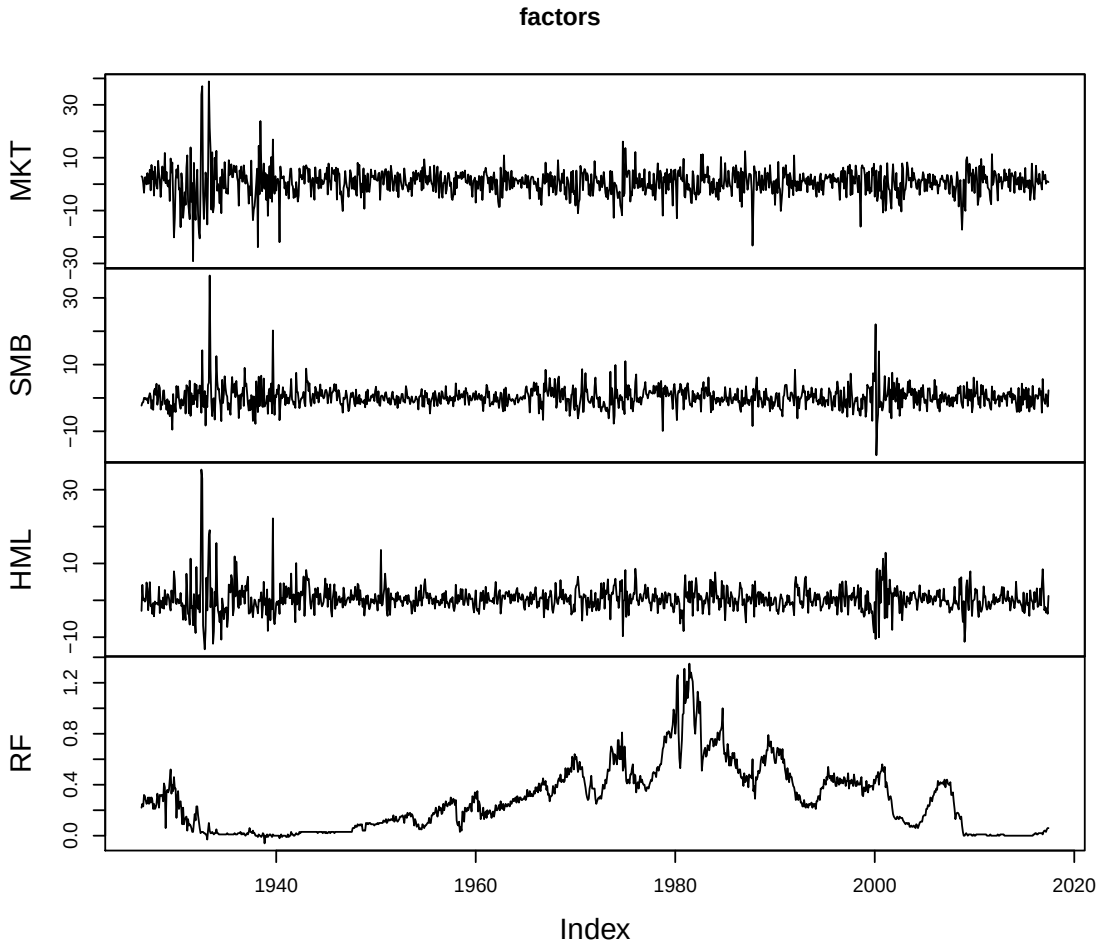


Figure 3.12: The time series of the Fama-French (FF) factors and the riskfree rate starting in 1926 at the monthly frequency. The data is obtained from Ken French website.

rows and 2 for columns). The example below shows how to calculate the mean and standard deviation of the variables and combines them in a data frame:

```
data.frame(AVERAGE = apply(factors, 2, mean), STDEV = apply(factors, 2, sd))
```

```
    AVERAGE STDEV
MKT    0.66  5.36
SMB    0.21  3.21
HML    0.39  3.50
RF     0.28  0.25
```

The average monthly MKT return from 1926 to 2017 has been 0.66% in excess of the risk-free rate. The average of SMB is 0.21% and represents the monthly premium deriving from investing in small capitalization relative to large capitalization stocks. The sample average of HML is 0.39% which measures the premium from investing in value stocks (high book-to-market ratio) relative to growth stocks (low book-to-market ratio). The standard deviation of MKT, SMB, and HML factors are 5.36%, 3.21%, and 3.5%, respectively. Another statistic that is useful to estimate is the correlation between the factors that allows us to measure their dependence. We can calculate the correlation matrix using the `cor()` function that was introduced earlier:

```
cor(factors)
```

```
       MKT    SMB   HML     RF
MKT  1.000  0.318 0.240 -0.065
SMB  0.318  1.000 0.123 -0.051
HML  0.240  0.123 1.000  0.021
RF  -0.065 -0.051 0.021  1.000
```

The correlation matrix is a table that shows the pair-wise correlation between the variable in the row and the one in the column. The numbers in the diagonal are all equal to 1 because they represent the correlation of a factor with itself. The correlation estimates show that SMB and HML are weakly correlated to the MKT returns (0.32 and 0.24, respectively) and also among each other (0.12). In this sense, it seems that the factors capture moderately correlated sources of risk which can be valuable from a diversification stand-point.

3.5.1 Size portfolios

Financial economics has devoted lots of energy to understand the factors driving asset prices and their expected returns. On the way, many *anomalies* have emerged in the sense of empirical facts that did not align with a theory. In this case the theory is the Capital Asset Pricing Model (CAPM) which states that the expected return of an asset should be proportial to the exposure to systematic risk measure by the excess market return relative to the risk-free rate of return. If we denote by R_t^i the excess return of asset i in period t and by R_t^{MKT} the excess market return in the same period, the CAPM predicts that

$$R_t^i = \alpha + \beta R_t^{MKT} + \epsilon_t$$

where the intercept α in the theory should be equal to zero and the slope β measures the exposure of the asset to market risk. Let's consider an application. The data library in Ken French webpage provides historical data for the returns of portfolio of stocks formed based on their market capitalization. The way these portfolio are constructed is by sorting once a year all the stocks listed in the NYSE, AMEX,

Table 3.2: Average return and standard deviation of the decile portfolios sorted by size.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
AV.RET	1.1	0.98	0.98	0.93	0.89	0.9	0.83	0.8	0.73	0.61
STD.DEV	10.0	8.73	7.97	7.44	7.06	6.8	6.43	6.1	5.80	5.05

and NASDAQ from the largest capitalization to the smallest and then creating portfolios that invest in a fraction of these stocks. The French dataset forms size portfolios as follows:

- 3 portfolios: the 30% of smallest capitalization, the 30% of largest, and the 40% of medium capitalization
- 5 portfolios: lowest 20%, largest 20%, and 20% interval between the smallest and largest (quintiles)
- 10 portfolios: divide the sample at intervals of 10% (deciles)

This process can be interpreted as a mutual fund that once a year (typically in June) forms a portfolio of stocks based on their market capitalization and keeps that allocation for one year. First, let's consider the 10 decile portfolios and calculate the average return over the period 1926-2017:

```
port10      <- c("Lo 10", "Dec 2","Dec 3","Dec 4","Dec 5",
                 "Dec 6","Dec 7","Dec 8", "Dec 9", "Hi 10")
size10      <- subset(port.size, select=port10)
table.size10 <- data.frame(`AV RET` = apply(size10, 2, mean),
                           `STD DEV` = apply(size10, 2, sd))
knitr::kable(t(table.size10), digits=3,
             caption = "Average return and standard deviation of the decile
                        portfolios sorted by size.")
```

The results in Table 3.2 show that, historically, portfolios of small caps have provided significantly higher returns relative to portfolios of large caps. A portfolio that consistently invested in the 10% of smaller caps earned an average monthly return of 1.12% relative to the portfolio of the largest companies that earned 0.61%. Why is the expected return of small caps portfolios higher relative to the larger caps? Is it because they are more risky? This is definitely the case since the standard deviation of the smallest cap portfolio is 9.96% relative to the 5.05% of the largest cap portfolio. Why is the standard deviation higher for small caps? do we want to look for an explanation to both? The CAPM model predicts that stocks or portfolios that are more exposed to systematic risk (high β) are riskier and receive compensation in the form of higher expected returns. Let's estimate the CAPM model in the previous Equation to the 10 size portfolios and, if the CAPM is correct, we should find that βs decline moving from the 1st decile portfolio to the 10th portfolio and that αs are close to zero. The code below estimates the CAPM model with the dependent variable being `size10` which is a `xts` object with 10 columns and 1092 rows. The `lm()` command will automatically estimate the regression on the independent variable `MKT` for each column of the object `size10`.

```
size.capm   <- lm(size10 ~ MKT, data=factors)
knitr::kable(coef(size.capm), digits=3,
             caption="OLS estimates of the intercept and slope coefficients in the CAPM
                      regression for 10 portfolio sorted by size (market capitalization).")
```

Table 3.3: OLS estimates of the intercept and slope coefficients in the CAPM regression for 10 portfolio sorted by size (market capitalization).

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
(Intercept)	0.19	0.075	0.11	0.11	0.082	0.11	0.071	0.063	0.034	-0.005
MKT	1.42	1.386	1.33	1.26	1.229	1.20	1.152	1.115	1.063	0.931

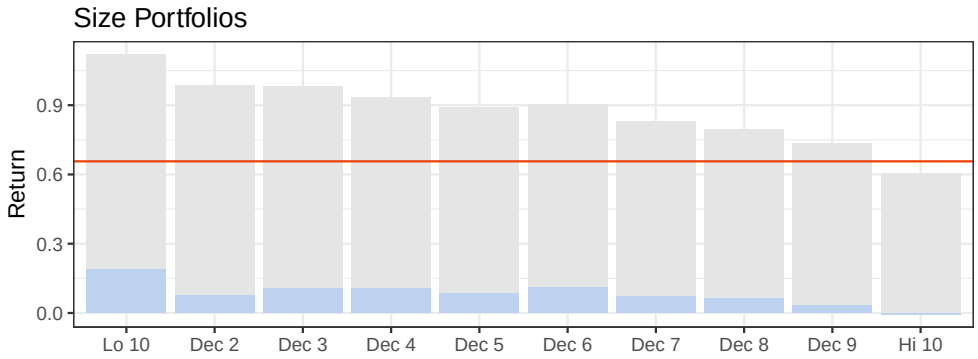


Figure 3.13: Bar plot of the average return of the decile portfolios sorted on size together with the estimated intercept (alpha). The horizontal line represents the average MKT return.

The estimates of the slope coefficients in Table 3.3 show, as expected, that smaller caps have higher *betas* relative to portfolios of large cap stocks (1.42 vs 0.93). This explains the fact that portfolios with more exposure to small caps are more volatile and provide higher expected returns relative to portfolios with predominantly larger stocks. What about the intercepts or *alpha*? The CAPM model predicted that these coefficients should be equal to zero, but a first assessment of the estimates does not seem to confirm this. The first portfolio has an estimate of α of 0.19 which, from the perspective of monthly returns, is a considerable number. However, a proper assessment of the hypothesis that $\alpha = 0$ should account for the standard errors of the estimates and perform a statistical test of the hypothesis against the alternative that the intercept is different from zero. The CAPM model also implies that the expected return of the portfolios, $E(R_t^i)$, is given by the sum of α and the compensation for exposure to market risk, βR_t^{MKT}. We can use the `ggplot2` package to do a bar-plot that helps visualizing the contribution of each component of the expected return.

```
size10.stat <- data.frame(PORT  = factor(names(size10), levels=names(size10)),
                          AVRET = apply(size10, 2, mean),
                          ALPHA = coef(size.capm)['(Intercept)',],
                          BETA  =  coef(size.capm)['MKT',])
ggplot(size10.stat) + geom_bar(aes(x = PORT, y = AVRET), fill="gray90",  stat = "identity") +
  geom_bar(aes(x = PORT, y = ALPHA),fill = "lightsteelblue2",  stat="identity") +
  theme_bw() + labs(x = "", y = "Return", title="Size Portfolios") +
  geom_hline(yintercept = mean(factors$MKT), color="orangered2")
```

Figure 3.13 shows that the CAPM model explains a significant component of the average return of the portfolios, but there is still between 10-20% of the return that is attributed to alpha. As we said before, it

could be that these intercepts are not estimated very precisely so that statistically they are not different from zero. To evaluate this hypothesis we need to obtain the standard errors or the t-statistic from the `lm()` object. Since the dependent variable `size10` in the previous regression is composed of 10 variable, the `lm` object for `summary(size.capm)` stores the regression results in a list with 10 elements and each element represents the results for one portfolio. For example:

```
summary(size.capm)[[1]]
```

```
Call:
lm(formula = `Lo 10` ~ MKT, data = factors)

Residuals:
        Lo 10
Min    -18.59
1Q      -2.85
Median  -0.40
3Q       1.98
Max     78.08

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1878     0.1963    0.96     0.34
MKT           1.4201     0.0364   39.02   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.4 on 1090 degrees of freedom
Multiple R-squared:  0.583, Adjusted R-squared:  0.582
F-statistic: 1.52e+03 on 1 and 1090 DF,  p-value: <2e-16
```

To extract the t-statistic of the intercept for each item in the list in one line we can use the `lapply()` or the `sapply()` function that apply a user-specified function to each item of a list. In this simple case, the function `sapply()` is preferable because it returns a vector rather than a list as the `lapply()` function does. The code to obtain this is shown below where we extract the value in row 1 and column 3 of `coefficients` for each item in the list:

```
capm.tstat <- sapply(summary(size.capm), function(x) tstat = x$coefficients[1,3])
```

```
Lo 10 Dec 2 Dec 3 Dec 4 Dec 5 Dec 6 Dec 7 Dec 8 Dec 9 Hi 10
 0.96  0.54  0.98  1.12  1.06  1.70  1.29  1.41  1.02 -0.21
```

Since all t-statistics are smaller than 1.96 we do not reject the null hypothesis that the $\alpha = 0$ for each of these 10 portfolios at 5% significance level, while at 10% significance only the intercept for the 6th decile is significant. Let's also extract from the regression results the adjusted R^2 of the regression and the standard error of regression. In the code below, I also bind the standard deviation of each portfolio return so that we can access the magnitude of the variables.

```
size.capm.stats <- sapply(summary(size.capm),
                          function(x) c(SER = x$sigma, AD.RSQ = x$adj.r.squared))
colnames(size.capm.stats) <- colnames(size10)
```

Table 3.4: Loadings of 3 Fama-French factors on 10 portfolio sorted by market capitalization.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
(Intercept)	-0.17	-0.17	-0.085	-0.055	-0.034	0.005	-0.001	0.01	-0.004	0.023
MKT	1.00	1.07	1.080	1.042	1.061	1.075	1.053	1.05	1.033	0.977
SMB	1.56	1.27	1.004	0.881	0.724	0.493	0.408	0.24	0.066	-0.215
HML	0.78	0.48	0.374	0.308	0.193	0.224	0.132	0.12	0.114	-0.034

The first portfolio is twice more volatile relative to the last portfolio and the CAPM model contributes to explain most of its variability (adjusted R^2 of 0.97), but only 0.58 for `Lo 10`[9]. This suggests that the small cap portfolios could benefit from adding additional risk factors that might explain some of the unexplained variation of the returns and so improve R^2.

The CAPM model predicts that the portfolio return should be explained by just one factor, the market return. This assumption might work for some portfolios, but it seems that it has some difficulties explaining the return of portfolios in which small cap stocks are predominant. Fama and French (1993) proposed a 3-factor model that add SMB and HML to the MKT factor given by

$$R_t^i = \alpha + \beta_{MKT} R_t^{MKT} + \beta_{SMB} SMB_t + \beta_{HML} HML_t + \epsilon_t$$

Let's see how the results would change if we estimate the 3-factor model on the size portfolios:

```
size.3fac <- lm(size10 ~ MKT + SMB + HML, data=factors)
knitr::kable(coef(size.3fac), digits=3,
             caption="Loadings of 3 Fama-French factors on 10
             portfolio sorted by market capitalization.")
```

The results in Table 3.4 show that:

- the estimated alpha are mostly negative
- the exposure to market risk (*market beta*) is very close to 1 for all portfolios and significantly smaller for the low decile portfolios relative to the CAPM regression (the beta for the first portfolio decreases from 1.42 to 1).
- The loading on the SMB for the low decile portfolios is large and positive and decreases until it becomes negative for the last decile portfolios. This is expected since the low portfolios have a larger exposure to small caps and thus benefit from the risk premium of small caps.
- Although we are not sorting stocks based on the book-to-market ratio but only on size, the loading on the HML factor is positive and large at low quantiles and decreases to approximately zero for the portfolio of largest cap stocks. This indicates the out-performance of the small caps portfolios is partly due also to a book-to-market effect, in the sense that small stocks are more likely to be value stocks and the regression is able to distinguish the component of the return that is due to the size effect and the part that is due to the value effect.

In addition, the comparison in Table 3.5 of the adjusted R^2 for the CAPM and 3-factor model indicates that the largest improvements in goodness-of-fit occur for the lowest decile portfolios, while for the top

[9]Remember that the R^2 is one minus the ratio of the variance of the residuals divided by the variance of the dependent variable; the values in the table are the standard deviations of the residuals and the quantities should be squared to calculate R^2 (also, the R^2 reported is adjusted).

Table 3.5: Adjusted R-square for the CAPM and the 3-factor models estimated on the size portfolios.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
CAPM	0.58	0.72	0.80	0.82	0.87	0.90	0.92	0.94	0.96	0.97
3-FACTOR	0.89	0.96	0.98	0.97	0.98	0.96	0.96	0.96	0.97	0.99

Table 3.6: Average return and standard deviation of the decile portfolios sorted by Book-to-Market ratio.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
AVRET	0.59	0.69	0.69	0.66	0.72	0.79	0.72	0.91	1.1	1.1
STDEV	5.70	5.30	5.40	5.92	5.70	6.01	6.43	6.72	7.7	9.2

decile portfolios the improvement is marginal.

```
size.capm.stats  <- sapply(summary(size.capm), function(x) AD.RSQ = x$adj.r.squared)
size.3fac.stats  <- sapply(summary(size.3fac), function(x) AD.RSQ = x$adj.r.squared)
size.stats       <- rbind(CAPM = size.capm.stats, `3-FACTOR` = size.3fac.stats)
colnames(size.stats) <- colnames(size10)
knitr::kable(size.stats, digits=3, caption="Adjusted R-square for the CAPM
             and the 3-factor models estimated on the size portfolios.")
```

3.5.2 Book-to-Market Ratio Portfolios

Another indicator that is often used to form portfolios is the book-to-market (BM) ratio, i.e., the ratio of the book value of a company to its market capitalization. Portfolios are formed by sorting stocks based on the BM ratio and decile portfolios are formed. Stocks with high BM ratio are called *value* and those with low BM ratio are called *growth*. Historically, value stocks have outperformed growth stocks which calls for an explanation similar to our earlier discussion of the outperformance of small relative to large capitalization stocks. Below is the code that calculates the average return and the standard deviation of the BM portfolios.

```
port10     <- c("Lo 10", "Dec 2","Dec 3","Dec 4","Dec 5",
                "Dec 6","Dec 7","Dec 8", "Dec 9", "Hi 10")
b2m10      <- subset(port.b2m, select=port10)

table.b2m10 <- data.frame(AVRET = apply(b2m10, 2, mean),
                          STDEV = apply(b2m10, 2, sd))
knitr::kable(t(table.b2m10), digits=3,
             caption = "Average return and standard deviation of the decile
                        portfolios sorted by Book-to-Market ratio.")
```

Table 3.6 shows that the first BM portfolio (growth stocks), has an average monthly return of 0.59% and a standard deviation of 5.7% as opposed to the last portfolio (value stocks) that has an average return of 1.07% and a standard deviation of 9.21%. Similarly to the size portfolios, there is a significant increase in the average return that comes also with an increase of its uncertainty. Similarly to the previous analysis,

Table 3.7: Coefficient estimates for the CAPM model on 10 portfolio sorted by book-to-market ratio.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
(Intercept)	-0.078	0.065	0.054	-0.025	0.065	0.11	0.002	0.16	0.22	0.11
MKT	1.011	0.947	0.970	1.049	1.002	1.03	1.101	1.14	1.28	1.46

Table 3.8: Coefficient estimates of the 3-factor model on 10 portfolio sorted by book-to-market ratio.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
(Intercept)	0.024	0.120	0.071	-0.064	-0.005	-0.003	-0.156	-0.036	-0.045	-0.25
MKT	1.077	0.978	0.986	1.032	0.975	0.977	1.011	1.015	1.103	1.19
SMB	-0.072	-0.004	-0.045	-0.036	-0.079	-0.058	0.021	0.118	0.223	0.55
HML	-0.336	-0.192	-0.047	0.151	0.270	0.427	0.546	0.665	0.854	1.09

we will proceed by estimating the CAPM model and the 3-factor model and evaluate the performance of each model in explaining the expected return of the BM portfolios.

```
b2m.capm <- lm(b2m10 ~ MKT, data=factors)
knitr::kable(coef(b2m.capm), digits=3,
             caption="Coefficient estimates for the CAPM model on 10 portfolio
             sorted by book-to-market ratio.")
```

As expected the exposure to market risk (beta) reported in Table 3.7 increases from 1.01 for the growth portfolio to 1.46 for the value portfolio. The beta for the BM portfolios is mostly close to 1 and increases significantly only for the top decile portfolios. In terms of the intercept α, the estimates are close to zero except for the top three deciles that are larger than zero and have t-statistics of 1.929, 2.052, 0.739 respectively.

```
b2m.3fac <- lm(b2m10 ~ MKT + SMB + HML, data=factors)
knitr::kable(coef(b2m.3fac), digits=3,
             caption="Coefficient estimates of the 3-factor model on 10 portfolio
             sorted by book-to-market ratio.")
```

Table 3.8 shows the results for the 3-factor model. The MKT beta for the value portfolio (top decile) has declined but it is still significantly larger than one. The SMB beta is mostly close to zero, except for the top three deciles where it is increasingly positive (equals 0.55 for the `Hi 10` portfolio). This confirms the findings for the size portfolios that value and small cap stocks intersect to a certain degree. Finally, the HML beta shows negative loadings in the growth portfolios and positive loadings in the value portfolios. In terms of goodness-of-fit, comparing the adjusted R^2 in Table 3.9 the 3-factor model is preferable for all portfolios to the CAPM model with the largest improvements occuring for the top decile portfolios.

```
b2m.capm.stats  <- sapply(summary(b2m.capm), function(x) AD.RSQ = x$adj.r.squared)
b2m.3fac.stats  <- sapply(summary(b2m.3fac), function(x) AD.RSQ = x$adj.r.squared)
b2m.stats       <- rbind(CAPM = b2m.capm.stats, `3-FACTOR` = b2m.3fac.stats)
colnames(b2m.stats) <- port10
knitr::kable(b2m.stats, digits=3, caption="Adjusted R-square for the CAPM
```

Table 3.9: Adjusted R-square for the CAPM and the 3-factor models estimate on the Book-to-Market portfolios.

	Lo 10	Dec 2	Dec 3	Dec 4	Dec 5	Dec 6	Dec 7	Dec 8	Dec 9	Hi 10
CAPM	0.90	0.91	0.93	0.90	0.89	0.85	0.84	0.83	0.80	0.72
3-FACTOR	0.94	0.93	0.93	0.91	0.91	0.91	0.93	0.95	0.95	0.93

```
and the 3-factor models estimate on the Book-to-Market portfolios. ")
```

3.5.3 CAPM and 3-factor models using `dplyr`

An alternative approach to estimate the asset pricing models on a set of portfolios or funds is to use the package `dplyr`. The advantage of using `dplyr` is that it allows to leverage its syntax and make the code more compact and transparent. In the application below, we estimate the CAPM and 3-factor model to the 10 size portfolios, but the code scales up to many more assets by only changing the input data frame. Some notes on the various steps and the new commands that are used below:

- first, the xts object `size10` is converted to a data frame and a column `date` is created
- the data frame is re-organized from 10 columns representing the returns over time of the strategy to a data frame with three columns representing:
  - `date`: the month of the observation
  - `Portfolio`: the portfolio (`Lo.10`, …, `Hi.10`)
  - `RET`: the value of the monthly return
- The function used to re-organize the data is `gather()` from the package `tidyr` that takes three arguments:
  - a data frame
  - the name of the new column to create based on the column names of the data frame
  - the name of the new column to create that will be filled wi the values of the returns of each strategy
- The `do()` command used to create `size.model` is used because there is no `dplyr` verb that does the operation we are interested in; in this case the `do()` command consists of the estimation of the CAPM and 3-factor model. The `size.model` will be a data frame with 10 rows and 3 columns representing the portfolio name, and the two additional columns containing the estimation output.
- To extract information from `size.model` we use the functions `tidy()` and `glance()` from package `broom`; the function `tidy()` produces a data frame with the portfolio names, the regressor (e.g., `(Intercept)`, `MKT`, `SMB`, `HML`) and additional columns for `estimate`, `std.error`, `statistic`, and `p.value`. Instead, the role of `glance()` is to extract the additional information about the regression such as the R^2, the adjusted R^2, the standard error of the regression (`sigma`), the F-statistic, its p-value, and a few more statistics.
- We then reorganize the information in a table format (`capm.tab` and `ff3fact.tab`) and print the tables using `kable()`

Table 3.10: Estimation results for the CAPM model on the 10 size portfolios.

Portfolio	alpha	MKT	r.squared	adj.r.squared	sigma	statistic	p.value
Lo.10	0.188	1.42	0.58	0.58	6.44	1523	0
Dec.2	0.075	1.39	0.72	0.72	4.61	2829	0
Dec.3	0.107	1.33	0.80	0.80	3.58	4318	0
Dec.4	0.108	1.26	0.82	0.82	3.16	4962	0
Dec.5	0.082	1.23	0.87	0.87	2.55	7281	0
Dec.6	0.111	1.20	0.90	0.90	2.13	9974	0
Dec.7	0.071	1.15	0.92	0.92	1.82	12561	0
Dec.8	0.063	1.11	0.94	0.94	1.46	18256	0
Dec.9	0.034	1.06	0.96	0.96	1.10	28976	0
Hi.10	-0.005	0.93	0.97	0.97	0.83	39492	0

```
size10.df  <- data.frame(date = ymd(time(size10)), coredata(size10))
factors.df <- data.frame(coredata(factors))
size10.df  <- tidyr::gather(size10.df, "Portfolio", "RET", 2:ncol(size10.df), factor_key = TRUE)
size.model <- size10.df %>%
              group_by(Portfolio) %>%
              mutate(MKT = factors.df$MKT, SMB = factors.df$SMB, HML = factors.df$HML) %>%
              do(fit.capm = lm(RET ~ MKT, data=.),
                 fit.3f   = lm(RET ~ MKT + SMB + HML, data=.))

capm.tab   <- broom::tidy(size.model, fit.capm) %>% dplyr::select(Portfolio, term, estimate) %>%
              tidyr::spread(term, estimate) %>% rename(alpha =  `(Intercept)`) %>%
              full_join(broom::glance(size.model, fit.capm), by = "Portfolio")

ff3fact.tab  <- broom::tidy(size.model, fit.3f) %>% dplyr::select(Portfolio, term, estimate) %>%
              tidyr::spread(term, estimate)  %>% rename(alpha = `(Intercept)`) %>%
              full_join(broom::glance(size.model, fit.capm), by = "Portfolio")
```

Table 3.10 and 3.11 show the regression results for the two asset pricing models estimated on the returns of the 10 size portfolios that includes the coefficient estimates of alpha and the exposures, in addition to the adjusted and unadjusted R^2 of the regression, the standard error of the regression (sigma), the F-statistic (statistic) and its p-value. Instead of presenting the regression results in a table format, we can plot the information in a graph which might be a more intuitive way of communicating the results. In Figure 3.14 the CAPM beta estimates for the 10 size portfolios are shown together with a line that represents the R^2 goodness-of-fit statistics for the portfolios. As we move from `Lo.10` to `Hi.10` (from small to large caps) the MKT beta decreases, the goodness-of-fit increases, and alpha decreases. These three indicators point in the same direction: the CAPM model is a good at explaining the top decile portfolios, but has some difficulties in *pricing* the low decile portfolios since R^2 is (relatively) low and alpha is large.

Table 3.11: Estimation results for the Fama French 3-factor model on the 10 size portfolios.

Portfolio	alpha	HML	MKT	SMB	r.squared	adj.r.squared	sigma	statistic	p.value
Lo.10	-0.167	0.780	1.00	1.561	0.58	0.58	6.44	1523	0
Dec.2	-0.170	0.481	1.07	1.269	0.72	0.72	4.61	2829	0
Dec.3	-0.085	0.374	1.08	1.004	0.80	0.80	3.58	4318	0
Dec.4	-0.055	0.308	1.04	0.881	0.82	0.82	3.16	4962	0
Dec.5	-0.034	0.193	1.06	0.724	0.87	0.87	2.55	7281	0
Dec.6	0.005	0.224	1.07	0.493	0.90	0.90	2.13	9974	0
Dec.7	-0.001	0.132	1.05	0.408	0.92	0.92	1.82	12561	0
Dec.8	0.010	0.115	1.05	0.236	0.94	0.94	1.46	18256	0
Dec.9	-0.004	0.114	1.03	0.066	0.96	0.96	1.10	28976	0
Hi.10	0.023	-0.034	0.98	-0.215	0.97	0.97	0.83	39492	0

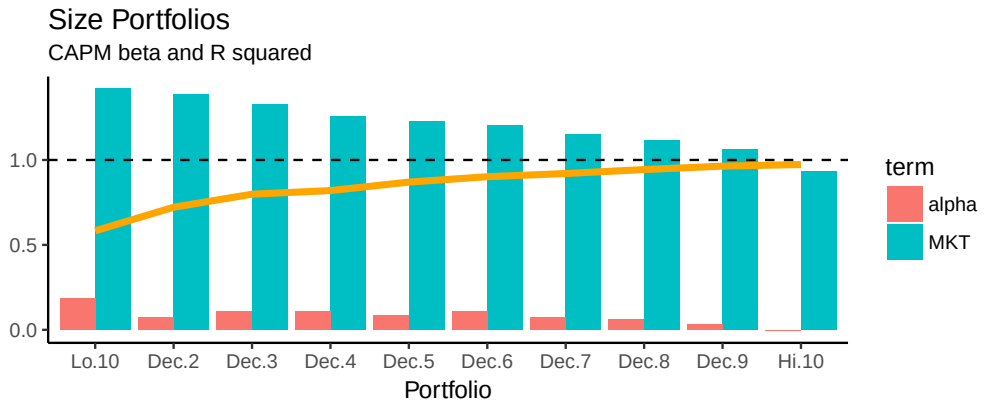


Figure 3.14: The alpha and beta estimates for the CAPM model for 10 portfolio sorted on size. The line represents the R^2 of the regression model.

```
capm.plot  <- broom::tidy(size.model, fit.capm) %>%
              mutate(term = plyr::mapvalues(term, from="(Intercept)", to="alpha"))

ggplot(capm.plot) +
   geom_bar(aes(x=Portfolio, y=estimate, fill=term), stat="identity", position="dodge") +
  geom_line(aes(x=Portfolio, y=r.squared,group=1), data=capm.tab, color="orange", size=1.5) +
  theme_classic() + geom_hline(yintercept = 1, color="black", linetype="dashed")+
  labs(x="Portfolio", y="", title="Size Portfolios", subtitle="CAPM beta and R squared")
```

When extending the analysis to the 3-factor model we find that the MKT exposure is quite similar across portfolios and close to 1[10]. The exposure to SMB risk is high for `Lo.10` and reduces as the component of small cap stocks decreases in higher decile portfolios. The contribution of the HML factor to the expected returns of the portfolios follows a similar pattern. Hence, low decile size portfolios benefit from both a

[10]Compare the MKT beta estimate for the CAPM model in Figure 3.14 and **??**: why did they become smaller when estimating the 3-factor model on the low decile portfolios?

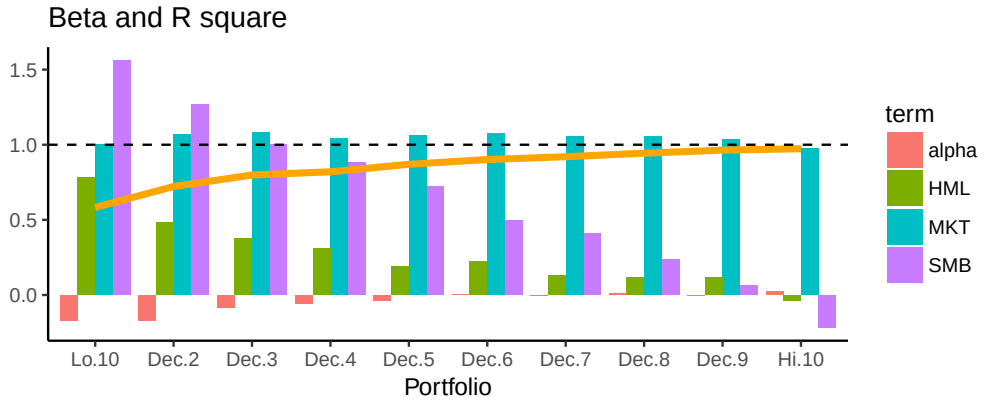


Figure 3.15: The alpha and beta estimates for the 3-factor model for 10 portfolio sorted on size. The line represents the R^2 of the regression model.

small cap and a value perspective. In addition, it turns out that the alpha estimates are negative, in particular for the low quantile portfolios which suggests that the model is actually *over-correcting* for the small size effect. A visualization of the estimation results is given in Figure 3.15.

```
ff3fact.plot <- broom::tidy(size.model, fit.3f)  %>%
              mutate(term = plyr::mapvalues(term, from="(Intercept)", to="alpha"))

ggplot(ff3fact.plot) +
   geom_bar(aes(x=Portfolio, y=estimate, fill=term), stat="identity", position="dodge") +
  geom_line(aes(x=Portfolio, y=r.squared, group=1), data=ff3fact.tab, color="orange", size=1.5) +
  theme_classic() + geom_hline(yintercept = 1, color="black", linetype="dashed")+
  labs(x="Portfolio", y="", title="Beta and R square")
```

Figure 3.16 represents a scatter plot of the alpha of the regression and the MKT beta in the CAPM model. It is interesting to notice that larger alpha are also associated with higher beta. As we saw in Figure 3.14, the combination of large alpha and large beta describes mostly the lowest decile portfolios, while large cap have lower market exposure and no alpha.

```
ggplot(capm.tab, aes(x=alpha, y=MKT)) + geom_point() +
  theme_bw() + geom_smooth(method="lm", se=FALSE, color="darkgreen")
```

3.5.4 What a style!

The data on size and BM portfolio returns provided in French's data library are constructed for academic research and are not actually traded in financial markets. However, there are several products available in the form of mutual funds or ETF for those investors that are interested in getting exposure to these risks. As an example, let's consider the *DFA Small Cap Value* mutual fund (ticker: `DFSVX`) that invests in companies with small capitalization that are considered undervalued according to a valuation ratio (e.g., Book to Market ratio). In this case the name of the fund already provides a clear indication of the type of risks that are involved in investing in such as fund. By regressing the returns of `DFSVX` on the Fama-French factors we can assess whether the stated investment strategy (*small cap value*) is indeed

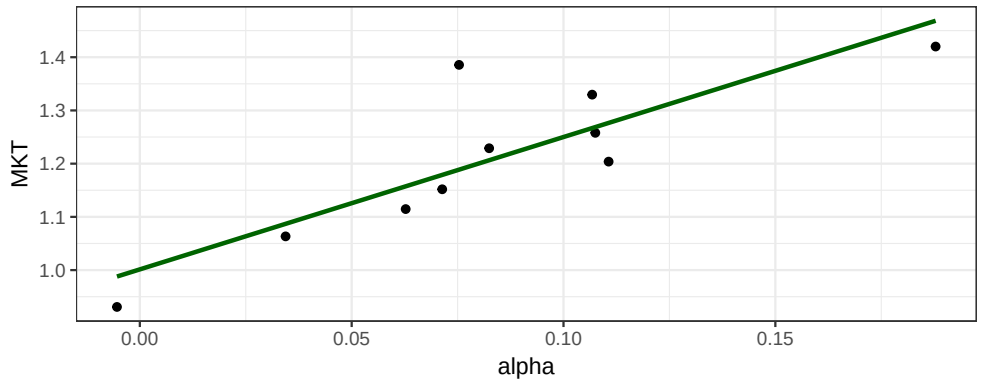


Figure 3.16: Scatter plot of the portfolio alpha and the MKT beta for the 10 portfolios sorted on size.

followed and whether the fund provides alpha, that is, the expected return that is not due to exposure to market, small cap, and value risks.

```
DFSVX  <- getSymbols("DFSVX", from = "1993-03-01", auto.assign=FALSE)
DFSVX  <- Ad(to.monthly(DFSVX))
DFSVX  <- 100 * diff(log(DFSVX))
names(DFSVX) <- 'DFSVX'

fit <- lm(DFSVX ~ MKT + SMB + HML, data=merge(factors, DFSVX))
fit.table <- coeftest(fit, vcov=NeweyWest(fit, prewhite = FALSE))
```

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.00581    0.06804    0.09     0.93
MKT          1.03885    0.02035   51.04   <2e-16 ***
SMB          0.78181    0.06477   12.07   <2e-16 ***
HML          0.66912    0.04094   16.34   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results show that `DFSVX` has statistically significant exposure to all three risk factors, in particular with large positive coefficients both in the SMB and in the HML factors. The R^2 of the regression is 0.96 that is very high and indicates that the factors explain most of the variation in the fund returns. In addition, the alpha is very close to zero and with a t-stat of 0.09 it is not statistically different from zero. The overall picture that emerges is that the `DFSVX` represents a tradable asset to obtain passive exposure to market, small cap, and value stocks. In addition, keep in mind that we might have ignored the investment strategy followed by the fund, and the simple interpretation of the regression results would have indicated that the manager invests in US small cap value stocks. Although the illustration is for mutual funds that are highly transparent about their investment strategy, the technique can be used also to investigate hedge fund returns or the returns of an investment strategy.

Let's practice using the Fama-French 3-factor model. Assume that you are assigned the task of classifying the investment style of 7 *mistery* funds that you are given to analyze. The only information you are provided is that these funds invest in U.S. equity only. This is called *style analysis* in the sense that you

are trying to understand the investment style or philosophy of a mutual or hedge fund. In the code below, I retrieve prices for 7 `mistery.ticks` from Yahoo Finance and create the monthly returns. Notice some new features in the code below:

- An environment called `mistery.env` is defined where the function `getSymbols()` stores the seven `xts` objects
- Function `eapply()` is an `apply` function that is used for environments and applies a function to each element of an environment
- The function that is used in `eapply()` to each of the seven elements is `function(x) 100 * ClCl(to.monthly(x))` which calculates the percentage return as follows:
  - takes element `x` of the environment
  - subsamples to monthly frequency using `to.monthly()`
  - calculates the close-to-close return with function `ClCL()`
  - multiplies by 100 to make the return percentage
- `mistery.list` is a list with each element representing a `xts` object with the return of a mistery fund
- `Reduce()` merges the elements in the list in a `xts`-object that has seven columns; keep in mind that each fund might have started at different points in time so that merging takes care of aligning the dates appropriately.
- The columns/variables of the `mistery.ret` are named `FUND1` to `FUND7` to keep their anonymity

The last three observations are shown below. Relative to the previous Chapter, we introduced the `eapply()` function that is not very often used by intermediate `R` programmers, but it is very useful in this case to extract the information needed in a few lines of code. Also, notice that this code could have been generalized to a larger number of tickers with no additional programming effort.

```
# mistery.ticks is a vector with 7 "mistery" tickers
mistery.env    <- new.env()
mistery.ticks  <- getSymbols(mistery.ticks, from = "1995-03-01", env = mistery.env)
mistery.list   <- eapply(mistery.env, function(x) 100*ClCl(to.monthly(x)))
mistery.ret    <- Reduce(merge, mistery.list[mistery.ticks])
mistery.names  <-  paste("FUND", 1:length(mistery.ticks), sep="")
colnames(mistery.ret) <- mistery.names
tail(mistery.ret, 3)
```

```
         FUND1 FUND2 FUND3 FUND4 FUND5 FUND6 FUND7
Jul 2017  2.07  0.79  0.47  1.68  0.77  0.87  1.41
Aug 2017  0.26 -2.31 -2.16 -0.94 -2.55 -2.02  0.81
Sep 2017  0.21  0.71  0.72  0.49  0.67  0.71  0.12
```

Once we create the returns for the seven funds, the next step is to estimate the 3-factor model. In this case we can follow the earlier approach of using `mistery.ret` in the `lm()` formula that will estimate the regression model on MKT, SMB, and HML for each column/variable. Can you tell the style of these funds based on the results on Table 3.12? Which fund invests in large, large value, large growth, small, small value, and small growth stocks?

```
mistery.data       <- merge(mistery.ret, factors)
fit                <- lm(mistery.data[,mistery.names] ~ MKT + SMB + HML, data=mistery.data)
mistery.rsq        <- sapply(summary(fit), function(x) AD.RSQ = x$adj.r.squared)
```

Table 3.12: Estimation results for seven mistery mutual fund returns regressed on the Fama-French 3 factors. The estimation is perfomed with the lm() command.

	FUND1	FUND2	FUND3	FUND4	FUND5	FUND6	FUND7
(Intercept)	-0.195	-0.36	-0.39	-0.36	-0.54	-0.13	-0.19
MKT	0.999	1.02	0.98	1.10	1.06	0.99	1.01
SMB	-0.156	0.76	0.88	0.00	0.80	0.69	-0.13
HML	-0.022	0.28	0.34	0.32	0.51	0.08	-0.12
R squared	0.984	0.92	0.88	0.92	0.91	0.95	0.96

When we apply the `lm()` function to a data frame as the dependent variable the function estimates the parameters on the same set of observations for all variables, that is, eliminating missing values. This can be an issue in this dataset since these funds started at different point in time[11]. For example, the earliest fund started in Apr 1995 and the latest in Jan 2013 so that the approach followed above drops many years of data for the longest existing funds. This is not necessarily a bad thing, since there are situation in which you might want to estimate the model parameters for all funds on the same time period. However, there are also cases in which you want to estimate the coefficient on the longest available sample for each fund. If this is the case, then we need to take a different coding strategy relative to the previous one which consists of estimating the 3-factor model having each column of `mistery.ret` as the dependent variable. This can be done using a loop over the columns of the object or using an `apply()`-type function that substitutes for the need of the loop. The example below first defines a function `myf()` that takes a variable `x` that is a `xts` object, merges the variables with the factors and then runs a regression of the variable on MKT, SMB, and HML. This function is then passed to `sapply()` which applies the function to each column of the object `mistery.ret`. The reason for using `sapply()` instead of `apply()` is that it provides a matrix of the results already formatted. Comparing Table 3.13 to 3.12 that are some minor differences on the coefficient estimates, but no major differences relative to the previous case.

```
myf <- function(x)
{
  data=merge(x,factors)
  fit <- lm(data[,1] ~ MKT + SMB + HML, data=data)
  output <- c(coef(fit), 'R squared' = summary(fit)$r.squared)
  return(output)
}
out <- sapply(mistery.ret,  myf)
knitr::kable(out, digits=3, caption="Estimation results for seven mistery mutual fund
             returns regressed on the Fama-French 3 factors. The estimation is performed
             individually on each fund")
```

Footnote[12] provides the solution to the mistery.

[11] However, this was not a problem earlier when estimating the size and BM portfolios because we had complete observations for all portfolios

[12] The *mistery* funds are: 1) DFUSX: DFA US Large Company, 2) DFSTX: DFA US Small Cap, 3) DFSCX: DFA US Micro Cap, 4) DFLVX: DFA US Large Cap Value, 5) DFSVX: DFA US Small Cap Value, 6) DSCGX: DFA US Small Cap Growth Instl, 7) DUSLX: DFA US Large Cap Growth Instl

Table 3.13: Estimation results for seven mistery mutual fund returns regressed on the Fama-French 3 factors. The estimation is performed individually on each fund

	FUND1	FUND2	FUND3	FUND4	FUND5	FUND6	FUND7
(Intercept)	-0.022	-0.27	-0.33	-0.219	-0.33	-0.13	-0.19
MKT	0.984	1.01	0.96	1.062	1.01	0.99	1.01
SMB	-0.175	0.78	0.93	-0.097	0.74	0.69	-0.13
HML	0.034	0.29	0.20	0.558	0.62	0.08	-0.12
R squared	0.988	0.90	0.81	0.844	0.85	0.95	0.96

3.6 Omitted variable bias

An important assumption that is introduced when deriving the properties of the OLS estimator is that the regressors included in the model represent all those that are relevant to explain the dependent variable. However, in practice it is difficult to make sure that this assumption is satisfied. In some situations we might not observe a variable that we believe relevant to explain the dependent variable, while in other situations we might not know which variables are actually relevant. What is the effect of omitting relevant variables on the OLS coefficient estimates? The answer depends on the correlation between the omitted variable and the included variable(s). If the omitted and included variables are correlated, then the estimate of the slope coefficients of the regressors will be biased, that is, they will be significantly different from their *true* value. However, if we omit a relevant variable that is not correlated with any of the included variables, then we do not expect any bias in the coefficient estimate. To illustrate this concept, we will discuss first an example on financial data and then perform a simulation exercise to illustrate the bias.

Assume that we are given to analyze a mutual fund return with the aim to understand the risk factors underlying the performance of the fund. Take this as another mistery fund that we want to classify in terms of investment style. We could start the empirical analysis as we did earlier, by assuming that the relevant factors to include in the regression are the Fama-French factors. The 3-factor model is thus given by $R_t^{fund} = \beta_0 + \beta_1 * MKT_t + \beta_2 * SMB_t + \beta_3 * HML_t + \epsilon_t$ and its estimation by OLS provides the following results:

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.0589     0.3838    0.15    0.878
MKT           1.0367     0.0753   13.77   <2e-16 ***
SMB           0.2746     0.1191    2.31    0.022 *
HML          -0.0280     0.1110   -0.25    0.801
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fund has an exposure of 1.04 against the US equity market that is highly significant. In addition, there seems to be significant (at 10%) exposure to SMB with a coefficient of 0.28 and a t-statistic of 2.31, but not to HML which has a t-statistic of -0.25. In addition, the R^2 of the regression is equal to 0.53, which indicates a reasonable fit, although for this type of regressions we typically expect larger

goodness-of-fit statistics (see the previous Section). Based on these results, we would conclude that the exposure to MKT makes the fund sensitive to the fluctuations of the overall US equity market, in addition to some exposure to the small cap premium deriving from increase exposure to small caps. However, it turns out that the fund is the *Oppenheimer Developing Markets* (ticker: ODMAX) which is a fund that invests exclusively in stocks from emerging markets and does not hold any US stock. How do we make sense of the regression results above and the stated investment strategy of the fund? How is it possible that the exposures to the MKT and SMB are large and significant in the regression above but the fund does not hold any US stock? It is possible because the MKT and SMB might be correlated with an omitted risk factor (e.g., emerging market risk) which causes bias in the estimates for MKT and SMB. The bias is due to the fact that omitting a relevant risk factor makes the MKT and SMB look significant since they co-move, to a certain extent, to that omitted factor although they would not be relevant if the omitted factor was included. The solution in this application is to include other factors that are more appropriate to proxy for the risk exposure of the fund. Given the investment objective of the fund, we could consider including as an additional risk factor the MSCI Emerging Markets (EM) Index. In terms of correlation between the EM factor and the Fama-French factors, the results below indicate that there is a strong positive correlation with the US-equity market and by smaller correlations with SMB (positive) and HML (negative).

```
  MKT   SMB   HML
 0.69  0.26 -0.14
```

It seems thus reasonable to include the EM factor as an additional risk factor to explain the performance of ODMAX. The Table below shows the estimation results of a regression of ODMAX excess monthly returns on 4 factors, the EM factor in addition to the FF factors.

```
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.3716     0.2326    1.60    0.112
EEM           0.7234     0.0642   11.26   <2e-16 ***
MKT           0.1938     0.0822    2.36    0.019 *
SMB           0.1554     0.0877    1.77    0.078 .
HML           0.0200     0.0632    0.32    0.752
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Not surprisingly, the estimated exposure to EM is 0.72 and highly significant, whilst the exposure to the FF factors has declined significantly. In particular, adding EM to the regression has the effect of reducing the coefficient of the MKT from 1.04 to 0.19. The drop in the MKT beta is due to the fact that in the first regression part of the coefficient of MKT was actually proxing for the effect of EM. The estimate from the first regression of 1.04 is biased because it captures both the effect of MKT on ODMAX, but also represents a good proxy for the omitted source of risk of the EM Index, given the large and positive correlation between MKT and EM. The effect of omitted variables and the resulting bias in the coefficient estimates is not only an econometric issue, but it has important practical implications. If we use the LRM for performance attribution, that is, disentangling the systematic component of the fund return (beta) from the risk-adjusted part (alpha), then omitting some relevant risk factors has the effect of producing bias in the remaining coefficients and thus changes our conclusion about the contribution of each component to the performance of the fund. Typically, for mutual funds we have a clear idea of the

investment strategy and the type of risks that the asset returns are exposed to. However, hedge funds are less transparent about their investment strategy and it might be difficult to pick the right factors and the coefficient estimates might be biased.

3.6.1 A simulation exercise

We can perform a simulation to illustrate the effect of omitted variables in producing biased coefficient estimates. The steps of the simulation are as follows:

1. We will assume that the dependent variable Y_t (for $t = 1, \cdots, T$) is generated by the following model:

$$Y_t = 0.5 * X_{1,t} + 0.5 * X_{2,t} + \epsilon_t$$

   where the factors $X_{1,t}$ and $X_{2,t}$ are simulated from the (multivariate) normal distribution with mean 0 and variance 1 for both variables, with their correlation set equal to ρ. The error term ϵ_t is also normally distributed with mean 0 and standard deviation 0.5. In the context of this simulation exercise, the model above for Y_t represents the *true* model with the *population* parameter values equal to $\beta_0 = 0$ and $\beta_1 = \beta_2 = 0.5$.

2. We then estimate by OLS the following model:

$$Y_t = \beta_0 + \beta_1 X_{1,t} + \eta_t$$

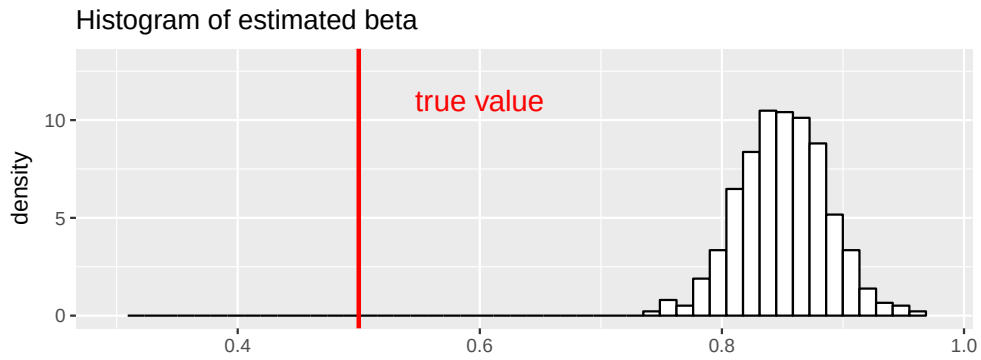
   where we intentionally omit the factor $X_{2,t}$ from the regression. Notice that $X_{2,t}$ is both a relevant variable to explain Y_t (since $\beta_2 = 0.5$) and it is correlated with $X_{1,t}$ if we set $\rho \neq 0$

3. We repeat step 1-2 S times and store the estimate of β_1

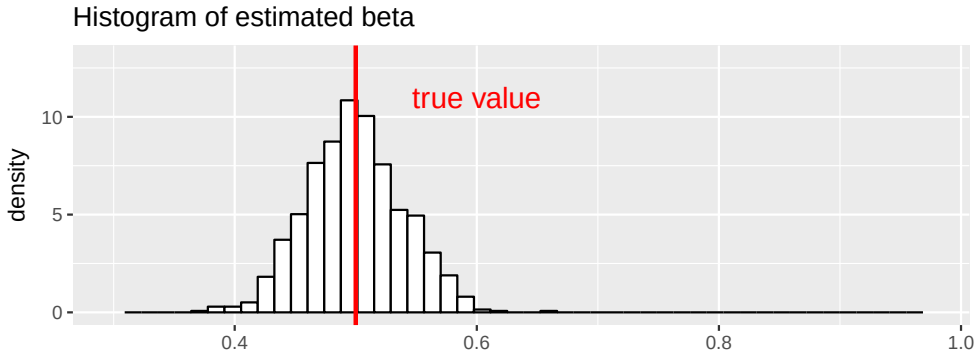
We can then analyze the properties of $\hat{\beta}_1$, the estimate of β_1, by, for example, plotting a histogram of the S values obtained in the simulation. If omitting $X_{2,t}$ does not introduce bias in the estimate of β_1, then we expect the histogram to be centered at the true value of the parameter 0.5. Instead, the histogram will be shifted away from the true value of the parameter if the omission introduces bias. The code below starts by setting the values of the parameters, such as the number of simulations, length of the time series, and the parameters of the distributions. Then the `for` loop iterates S times step 1 and 2 described above, while the bottom part of the program plots the histogram.

```
library(MASS) # this package is needed for function `mvrnorm()` to simulate from the
              # multivariate normal distribution

S     <- 1000                          # set the number of simulations
T     <- 300                           # set the number of periods
mu    <- c(0,0)                        # mean of variables X1 and X2
cor   <- 0.7                           # correlation coefficient between X1 & X2
Sigma <- matrix(c(1,cor,cor,1), 2, 2) # covariance matrix of X = [X1, X2]
beta  <- c(0.5, 0.5)                   # slope coefficient of X = [X1, X2]
eps   <- rnorm(T, 0, 0.5)              # errors

betahat <- matrix(NA, S, 1)            # vector to store the estimates of beta1
```

```
for (i in 1:S)                          # loop starts here
{                                               # simulate the indep. variable X = [X1, X2]
  X   <- mvrnorm(T, mu, Sigma)                  # `mvnorm` from package MASS simulate the
                                                # multivariate normal distribution (dim: Nx2)
  Y   <- beta[1]*X[,1] + beta[2] * X[,2] + eps# simulate the dep. variable Y (dim: Nx1)
  fit <- lm(Y ~ X[,1])                          # fit the linear model of Y on X1 but not X2
  betahat[i] <- coef(fit)[2]                    # store the estimated coefficient of X1
}                                       # loop ends here

# set the limits of the x-axis
xmin <- min(min(betahat), 0.3)
xmax <- max(betahat)
ymax <- 13

# plot the histogram of betahat together with a vertical line at the true value of beta
ggplot() + geom_histogram(aes(betahat, y = ..density..), bins=50, fill="white", color="black") +
           xlim(c(xmin, xmax)) + ylim(c(0,ymax)) + geom_vline(xintercept = beta[1], col="red", size=1) +
           annotate("text", label = "true value", x = beta[1]+0.1, y =11, size = 5, colour = "red") +
           labs(x="",title="Histogram of estimated beta")
```



In the simulation exercise we set the correlation between $X_{1,t}$ and $X_{2,t}$ equal to 0.7 (line `cor = 0.7`) and it is clear from the histogram above that the distribution of the OLS estimate is shifted away from the true value of 0.5. This illustrates quite well the problem of omitted variable bias: we expect the estimates of β_1 to be close to the true value of 0.5, but we find that these estimates range from 0.75 to 0.95. This bias does not disappear using longer samples since it is due to the fact that we omitted a relevant variable that is highly correlated with an included variable. If the omitted variable was relevant but uncorrelated with the included variable, then the histogram of the OLS estimates would look like the following plot that is produced with the same code used above, but by setting `cor = 0`.


Histogram of estimated beta
true value
density
10
5
0
0.4
0.6
0.8
1.0

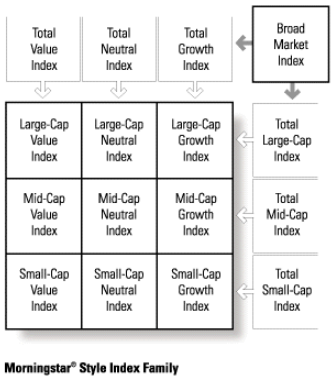


Figure 3.17: the Morningstar Investment Style Box

R commands

Table 3.14: R functions used in this Chapter.

annotate()	coeftest()	geom_abline()	mean()	rbind()	spread()
apply()	colnames()	inner_join()	median()	read_excel()	summary.lm()
arrange()	download()	kable()	mvrnorm()	rnorm()	which.min()
as.yearmon()	gather()	lm()	NeweyWest()	sapply()	NA

Table 3.15: R packages used in this Chapter.

downloader	lmtest	readxl	tidyr
knitr	MASS	sandwich	NA

Exercises

1. Use the dataset of Welch and Goyal (2008) to investigate the predictability of stock returns at the quarterly and monthly frequency:
   - Plot the scatter plot of the DP ratio and the equity premium at the quarterly and monthly frequency
   - Estimate the linear regression model of the future equity premium on the DP ratio using Newey-West standard errors. Is there evidence of predictability?
2. Morningstar is an investment advisory company that has developed a *style box* to classify US equity mutual funds as shown in Figure 3.17 and a detailed discussion is discussed in this factsheet. The style box is a popular tool to assess the characteristics of a mutual fund and it is provided in the profile page of a mutual fund in Yahoo Finance.

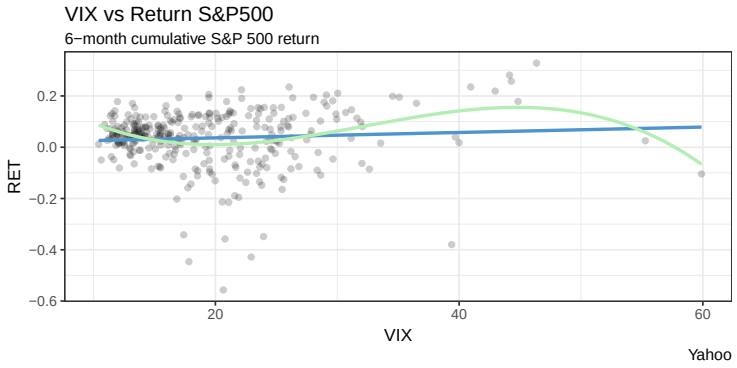


Figure 3.18: Relationship between the VIX and the cumulative SP 500 return in the following 6-month. The linear and cubic regressions are also shown.

Answer the following questions: - Use the Fama-French 3-factor model to classify three of the following mutal funds in the *style box* - Check the name and profile in Yahoo Finance or Morningstar to compare your analysis

RFV	RZG	RWJ	EZY	PFM	PXSV	EQL	SCHA	PXSG	MDYG
PFM	SPHQ	PWV	VB	RWJ	PWC	PDP	DES	VTI	PXSC
VIG	NDQ	MDY	RWK	DEF	GVT	FAD	VOE	SLY	EZM
EES	VIG	SPY	FAB	MGK	PKW	PWB	FVI	PXSG	SCHV
EZM	DWAQ	FVD	FTC	VYM	RWV	PEY	SCHA	FDV	SDY

3. Adrian et al. (2015) use a cubic polynomial to model the relationship between the VIX in month t and the cumulative return of the S&P500 Index in the following 6-months, that is,

$$R_{t:t+6} = \beta_0 + \beta_1 VIX_t + \beta_2 VIX_t^2 + \beta_3 VIX_t^3 + \epsilon_t$$

where $R_{t:t+6} = log(R_{t+6}) - log(R_t)$. Answer the following questions:

- Estimate the model using the tickers `^GSPC` and `^VIX` for the two assets starting from January 1990 at the monthly frequency
- What is the expect return in the 6-months following a VIX value of 55%?
- Test the null hypothesis that the model is linear
- Test the null hypothesis that the model is quadratic as opposed to cubic
- Produce a graph similar to Figure 3.18 and comment the results
- Remove the two monthly observations when the `VIX` index was higher than 50% and estimate the model on the remaining observations and answer the following questions:
  - is the shape of the cubic regression line different?
  - What is the expect return in the 6-months following a VIX value of 55%? Compare with the earlier result
  - Do you reject the null of linearity?
  - Is the cubic term signficant?
  - What is your conclusion about the robustness of the analysis on the full sample?

4. Consider the 10 size portfolios. Estimate the CAPM and the 3-factor model on sub-periods of 10 years

Chapter 4

Time Series Models

The goal of the LRM of the previous Chapter is to relate the variation (over time or in the cross-section) of variable Y with that of an explanatory variable X. Time series models assume that the independent variable X is represented by past values of the dependent variable Y. We typically denote by Y_t the value of a variable in time period t (e.g., days, weeks, and months) and Y_{t-1} refers to the value of the variable in the previous time period relative to today (i.e., t). More generally, Y_{t-k} (for $k \geq 1$) indicates the value of the variable k periods before t. The notation ΔY_t refers to the one-period change in the variable, that is, $\Delta Y_t = Y_t - Y_{t-1}$, while $\Delta^k Y_t = Y_t - Y_{t-k}$ represents the change over k periods of the variable. The aim of time series models can be stated as follows: is there useful information in the current and past values of a variable to predict its future?

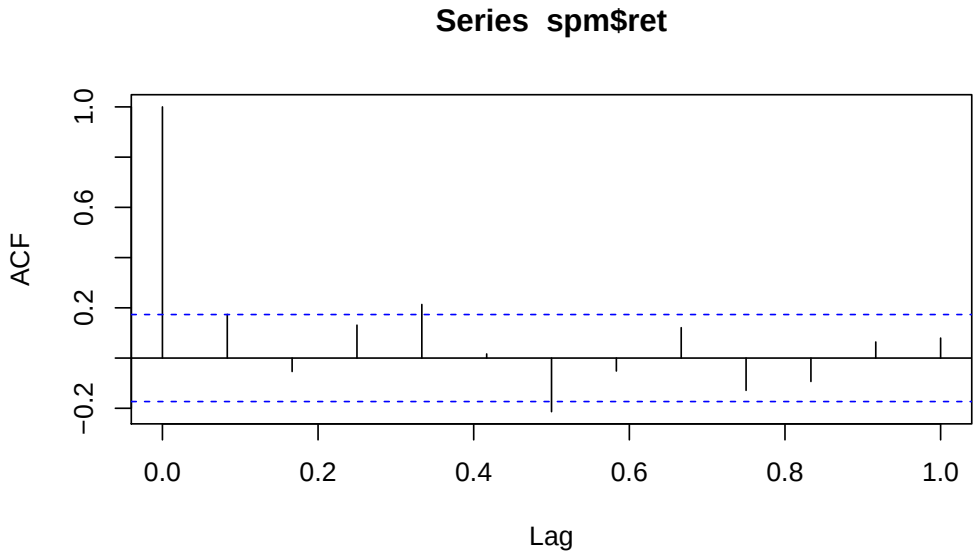
4.1 The Auto-Correlation Function (ACF)

A first exploratory tool that is used in time series analysis is the Auto-Correlation Function (ACF) that represents the correlation coefficient between the variable Y_t and its lagged value Y_{t-k} (hence the *Auto* part). We denote the sample correlation coefficient at lag k by $\hat{\rho}_k$ that is calculated as follows:

$$\hat{\rho}_k = \frac{\sum_{t=1}^{T}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})/T}{\hat{\sigma}_Y^2}$$

where $\bar{Y}$ and $\hat{\sigma}_Y^2$ denote the sample mean and variance of Y_t. The auto-correlation is bounded between $\pm$ 1 with positive values indicating that values of Y_{t-k} above/below the mean μ_Y are more likely to be associated with values of Y_t above/below the mean. In this sense, positive correlation means that the variable has some persistence in its fluctuations around the mean. On the other hand, negative values represent a situation in which the variable is more likely to reverse its past behavior. The ACF is called a function because $\hat{\rho}_k$ is typically calculated for many values of k. The `R` function to calculate the ACF is `acf()` and in the example below we apply it to the monthly returns of the S&P 500 Index from January 1990 until September 2017. The `acf()` function requires to specify the `lag.max` that represents the maximum value of k to use and the option `plot` that can be set to `TRUE` if the ACF should be plotted or `FALSE` if the estimated values should be returned.

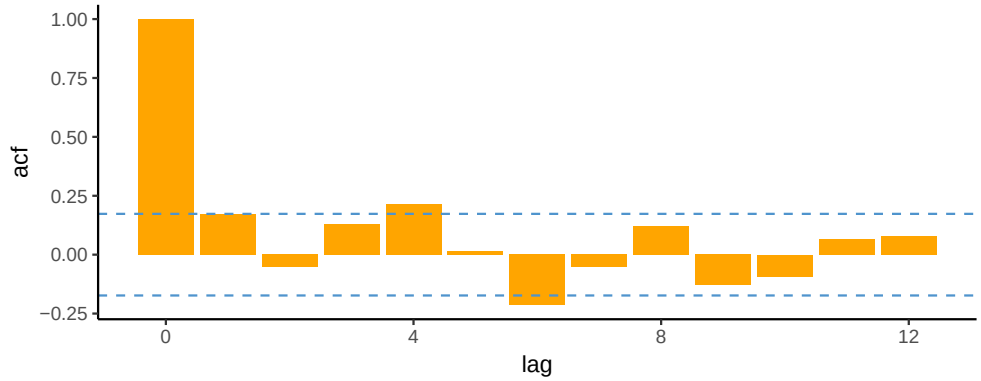
```
library(quantmod)
sp.data    <- getSymbols("^GSPC", start="1990-01-01", auto.assign=FALSE)
spm        <-  Ad(to.monthly(sp.data))
names(spm) <- "price"                    # rename the column "price" from "..Adjusted"
spm$return <- 100 * diff(log(spm$price)) # create the returns
spm        <- na.omit(spm)               # drops the first observation that is NA
# plot the ACF function
acf(spm$ret, lag.max=12, plot=TRUE)
```



The function provides a graph with the horizontal axis representing the lag k (starting at 0 and expressed as a fraction of a year) and the vertical axis representing the value of $\hat{\rho}_k$. The horizontal dashed lines are equal to $\pm 1.96/\sqrt{T}$ and represent the confidence interval for the null hypothesis that the autocorrelation is equal to 0. If the auto-correlation at lag k is within the interval we conclude that we fail to reject the null that the auto-correlation coefficient is equal to zero (at 5% level). The plot produced by `acf()` uses the default `R` plotting function, although we might prefer a more customized and elegant graphical output using the `ggplot2` package. This package does not provide a ready-made function to plot the ACF[1], but we can easily solve this problem by creating a data frame that contains the lag k and the estimate of the auto-correlation $\hat{\rho}_k$ and pass it to `ggplot()` function for plotting. The code below provides an example of how to produce a `ggplot` of the ACF.

```
Tm = length(spm$return)
spm.acf <- acf(spm$return, lag.max=12, plot=FALSE)
spm.acf <- data.frame(lag= 0:12, acf = spm.acf$acf)

library(ggplot2)
ggplot(spm.acf, aes(lag, acf)) +
  geom_bar(stat="identity", fill="orange") +
  geom_hline(yintercept =  1.96 * Tm^(-0.5), color="steelblue3", linetype="dashed") +
  geom_hline(yintercept = -1.96 * Tm^(-0.5), color="steelblue3", linetype="dashed") +
  theme_classic()
```

[1]However, the `autoplot()` function of package `ggfortify` builds on the `ggplot2` functionalities to provide an ACF plot.



The analysis of monthly returns for the S&P 500 Index shows that the auto-correlations are quite small and only few of them are statistically different from zero at 5% (such as lags 1, 4, 6). The ACF results indicate that there is very weak dependence in monthly returns so that lagged values might not be very useful in forecasting future returns. The conclusion is not very different when considering the daily frequency. The code below uses the dataset downloaded earlier to create a data frame `spd` with the adjusted closing price and the return at the daily frequency. In addition, we can exploit the flexibility of `ggplot2` to write a function to perform the task. In the example below, the function `ggacf` is created that takes the following arguments:

- *y* is a numerical vector that represents a time series
- *lag* represents the maximum lag to calculate the ACF; if not specified, the default is 12
- *plot.zero* takes values *yes*/*no* if the user wants to plot the autocorrelation at lag 0 (which is always equal to 1); the default is *no*
- *alpha* the significance level for the confidence bands in the graph; default is 0.05

```
spd        <-merge(Ad(sp.data), 100*ClCl(sp.data))
names(spd) <- c("price","return")
spd        <- na.omit(spd)
#  a function to plot the ACF using ggplot2
ggacf <- function(y, lag = 12, plot.zero="no", alpha = 0.05)
{
  T    <- length(y)

  y.acf <- acf(y, lag.max=lag, plot=FALSE)
  if (plot.zero == "yes") y.acf <- data.frame(lag= 0:lag, acf = y.acf$acf)
  if (plot.zero == "no")  y.acf <- data.frame(lag= 1:lag, acf = y.acf$acf[-1])

  library(ggplot2)
  ggplot(y.acf, aes(lag, acf)) +
  geom_bar(stat="identity", fill="orange") +
  geom_hline(yintercept =  qnorm(1-alpha/2) * T^(-0.5), color="steelblue3", linetype="dashed") +
  geom_hline(yintercept =  qnorm(alpha/2) * T^(-0.5), color="steelblue3", linetype="dashed") +
  geom_hline(yintercept = 0, color="steelblue3") +
  theme_classic() + ylab("") + ggtitle("ACF")
}
```

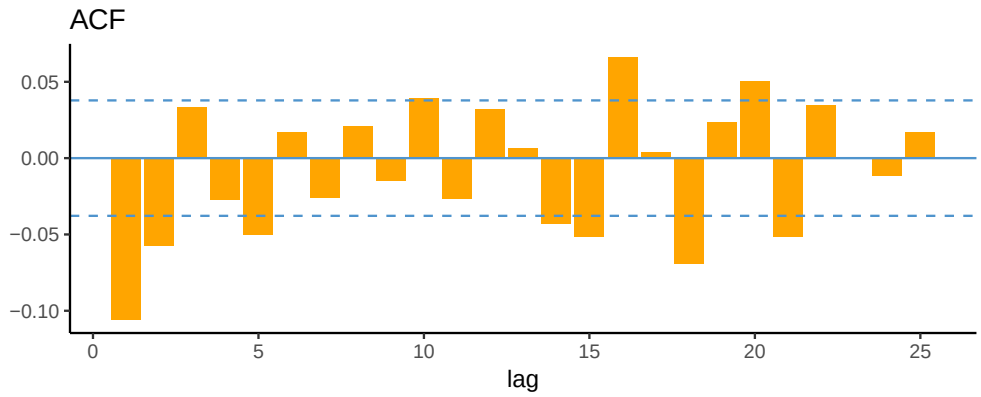


Figure 4.1: Auto-Correlation Function (ACF) for the daily returns of the SP 500.

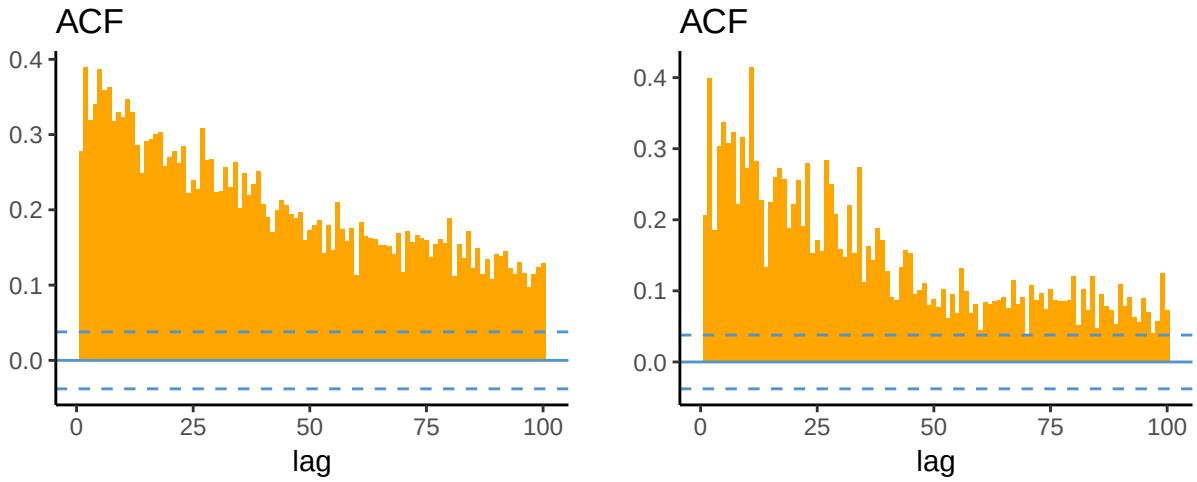


Figure 4.2: Auto-correlation function of absolute (left) and square (right) daily returns of the SP 500.

```
# use the function
ggacf(spd$return, lag=25)
```

There are 2686 daily returns in the sample so that the confidence band in Figure 4.1 is narrower around zero and several auto-correlations are significantly different from 0 (at 5% signficance level). Interestingly, the first order correlation is negative and significant suggesting that daily returns have a weak tendency to reverse the previous day change. A stylized fact is that daily returns are characterized by small auto-correlations, but large values of $\hat{\rho}_k$ when the returns are in absolute value or squared. Figure 4.2 shows the ACF for the absolute and squared daily returns up to lag 100 days:

```
p1  <- ggacf(abs(spd$return), 100)
p2  <- ggacf(spd$ret^2, 100)
```

These plots make clear that the absolute and square returns are significantly and positively correlated and that the auto-correlations decay very slowly as the lag increases. This shows that large (small) absolute returns are likely to be followed by large (small) absolute returns since the magnitude of the returns is

auto-correlated rather than their direction. This pattern is consistent with the evidence that returns display volatility clusters that represent periods of high volatility that can last several months, followed by long periods of lower volatility. This suggests that volatility (in the sense of absolute or squared returns) is persistent and strongly predictable, while returns are weakly predictable. We will discuss this topic and volatility models in the next Chapter.

4.2 The Auto-Regressive (AR) Model

The evidence in the ACF indicates that lagged values of the returns or absolute/square returns might be informative about the future value of the variable itself and we would like to incorporate this information in a regression model. The **Auto-Regressive** (AR) model is a regression model in which the independent variables are lagged values of the dependent variable Y_t. If we include only one lag of the dependent variable, we denote the model by `AR(1)` which is given by

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t$$

where β_0 and β_1 are coefficients to be estimated and ϵ_t is an error term with mean zero and variance σ^2_ϵ. More generally, the `AR(p)` model might include `p` lags of the dependent variable to account for the possibility that information relevant to predict Y_t is dispersed over several lags. The `AR(p)` model is given by

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t$$

The properties of the AR(p) model are:

- $E(Y_t|Y_{t-1}, \cdots, Y_{t-p}) = \beta_0 + \beta_1 Y_{t-1} \cdots + \beta_p Y_{t-p}$: the expected value of Y_t *conditional* on the recent p realizations of the variable can be interpreted as a forecast since only past information is used to produce an expectation of the variable today.
- $E(Y_t) = \beta_0/(1 - \beta_1 - \cdots - \beta_p)$: represents the *unconditional* expectation or the long-run mean of Y_t

$$\begin{aligned} E(Y_t) = & \quad \beta_0 + \beta_1 E(Y_{t-1}) + \cdots + \beta_2 E(Y_{t-p}) + E(\epsilon_t) \\ = & \quad \frac{\beta_0}{1 - \sum_{j=1}^{p} \beta_j} \end{aligned}$$

  assuming that $E(Y_t) = E(Y_{t-k})$ for all values of k. If the sum of the slopes β_i, for $i = 1, \cdots, p$, is equal to one the mean is not defined.
- $Var(Y_t) = \sigma^2_\epsilon/(1 - \beta^2_1 - \cdots - \beta^2_p)$ since

$$\begin{aligned} var(Y_t) = & \quad \beta_1^2 Var(Y_{t-1}) + \cdots + \beta_2^2 Var(Y_{t-p}) + Var(\epsilon_t) \\ = & \quad \left(\sum_{j=1}^{p} \beta_j^2\right) Var(Y_t) + \sigma^2_\epsilon \\ = & \quad \frac{\sigma^2_\epsilon}{1 - \sum_{j=1}^{p} \beta_j^2} \end{aligned}$$

  where $Var(Y_t) = Var(Y_{t-k})$ for all values of k.
- The sum of the slopes β_i is interpreted as a measure of *persistence* of the time series; persistence measures the extent that past values above/below the mean are likely to persist above/below the mean in the current period.

The concept of persistence is also related to that of *mean-reversion* which measures the speed at which a time series reverts back to its long-run mean. The higher the persistence of a time series the longer it will take for deviations from the long-run mean to be absorbed. Persistence is associated with positive values of the sum of the betas, while negative values of the coefficients imply that the series fluctuates closely around the mean. This is because a value above the mean in the previous period is expected to be below the mean in the following period, and viceversa for negative values. In economics and finance we typically experience positive coefficients due to the persistent nature of economic shocks, although the daily returns considered above are a case of first-order negative auto-correlation.

4.2.1 Estimation

The AR model can be estimated using the OLS estimation method, but there are also alternative estimation methods, such as Maximum Likelihood (ML). Although both OLS and ML are consistent in large sample, estimating an AR model with these methods might provide slightly different results (see Tsay (2005) for a more extensive discussion of this topic). There are various functions available in `R` to estimate time series models and we will discuss three of them below. While the first consist of using the `lm()` function and thus uses OLS in estimation, the functions `ar()` and `arima()` are specifically targeted for time series models and have ML as the default estimation method. Let's estimate an AR(12) model on the S&P 500 return at the monthly frequency.

lm()

The first approach is to use the `lm()` function discussed in the previous chapter together with the `dyn` package that gives `lm()` the capabilities to handle time series data and operations, such as the `lag()` and `diff()` operators. The estimation is implemented below[2]:

```
library(dyn)
fit <- dyn$lm(return ~ lag(return, 1:12), data = spm)
```

```
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.4012      0.409   0.982    0.328
lag(return, 1:12)1   0.1709      0.097   1.763    0.081
lag(return, 1:12)2  -0.0325      0.098  -0.330    0.742
lag(return, 1:12)3   0.1581      0.098   1.617    0.109
lag(return, 1:12)4   0.1493      0.098   1.527    0.130
lag(return, 1:12)5   0.0031      0.098   0.031    0.975
lag(return, 1:12)6  -0.2120      0.098  -2.167    0.033
lag(return, 1:12)7  -0.0707      0.098  -0.724    0.471
lag(return, 1:12)8   0.1213      0.098   1.238    0.219
lag(return, 1:12)9  -0.1157      0.097  -1.187    0.238
lag(return, 1:12)10  0.0475      0.097   0.488    0.627
lag(return, 1:12)11  0.0439      0.097   0.451    0.653
lag(return, 1:12)12  0.0073      0.096   0.076    0.939
```

[2]Since time series models explicitly account for time series dependence the errors of a well-specified model are likely to have no serial correlation. This suggests that the need to correct the standard errors is less compelling, although it is the case that heteroskedasticity might still play a role. In this Chapter the errors are obtained under the assumption of homoskedasticity which is the default output of `summary()`.

The Table with the results is in the familiar format from the previous chapter. The estimate of the sum of the coefficients of the lagged values is 0.27 which is not very high. This suggests that the monthly returns have weak persistence and that it is difficult to predict the return next month by only knowing the return for the current month. In terms of significance, at 5% level the only lag that is significant is 6 (with a negative coefficient), while the third lag is significant at 10%. The lack of persistence in financial returns at high-frequencies (intra-daily or daily), but also at lower frequencies (weekly, monthly, quarterly), is well documented and it is one of the stylized facts of returns common across asset classes.

ar()

Another function that can be used to estimate AR models is the `ar()` from the `tseries` package. The inputs of the function are the series Y_t, the maximum order or lag of the model, and the estimation method. An additional argument of the function is `aic` that can be `TRUE`/`FALSE`. This option refers to the Akaike Information Criterion (AIC) that is a method to select the optimal number of lags in the AR model and will be discussed in more detail below. The code below shows the estimation results for an AR(12) model when the option `aic` is set to `FALSE`[3]:

```
# estimation method "mle" for ML and "ols" or OLS
ar(spm$return, method="ols", aic=FALSE, order.max=12, demean = FALSE, intercept = TRUE)
```

```
Call:
ar(x = spm$return, aic = FALSE, order.max = 12, method = "ols",     demean = FALSE, intercept = TRUE)

Coefficients:
     1       2       3       4       5       6       7       8       9      10      11      12
 0.171  -0.032   0.158   0.149   0.003  -0.212  -0.071   0.121  -0.116   0.047   0.044   0.007

Intercept: 0.401 (0.385)

Order selected 12  sigma^2 estimated as  16.2
```

Since I opted for `method="ols"` the coefficient estimates are equal to the one obtained above with the `lm()` command. A problem with this function is that it does not provide the standard errors of the estimates. To evaluate their statistical significance we can obtain the standard errors of the estimates and calculate the t-statistics for the null hypothesis that the coefficients are equal to zero:

```
data.table <- data.frame(COEF = fit.ar$ar,
                         SE = fit.ar$asy.se.coef$ar,
                         TSTAT = fit.ar$ar /fit.ar$asy.se.coef$ar,
                         PVALUE = pnorm(-abs(fit.ar$ar /fit.ar$asy.se.coef$ar)))
```

```
     COEF    SE  TSTAT PVALUE
1   0.171 0.091  1.871  0.031
2  -0.032 0.093 -0.351  0.363
```

[3]The option `demean` means that by default the `ar()` function subtracts the mean from the series before estimating the regression. In this case, the `ar()` function estimates the model $Y_t - \bar{Y} = \beta_1 * (Y_{t-1} - \bar{Y}) + \epsilon_t$ without an intercept. If we denote the expected value of Y_t by μ, then using the previous equation we can express the intercept as $\beta_0 = (1 - \beta_1)\mu$. By replacing this value for β_0 in the AR(1) model we obtain $Y_t = (1 - \beta_1)\mu + \beta_1 Y_{t-1} + \epsilon_t$ that can be rearranged as $Y_t - \mu = \beta_1(Y_{t-1} - \mu) + \epsilon_t$. So, estimating the AR model or the AR model in deviation from the mean leads to the same estimate of β_1.

```
3   0.158 0.092  1.716  0.043
4   0.149 0.092  1.620  0.053
5   0.003 0.092  0.033  0.487
6  -0.212 0.092 -2.300  0.011
7  -0.071 0.092 -0.768  0.221
8   0.121 0.092  1.314  0.094
9  -0.116 0.092 -1.260  0.104
10  0.047 0.092  0.518  0.302
11  0.044 0.092  0.479  0.316
12  0.007 0.090  0.081  0.468
```

where we can see that only the third and sixth lags are significant at 5%. The standard errors are close but not equal to the values obtained for `lm()`.

arima()

The third function that can be used to estimate AR model is `arima()`. The function is more general relative to the previous one since it is able to estimate time series models called Auto-Regressive Integrated Moving Average (`ARIMA`) models. In this Chapter we will not discuss MA processes, and postpone the presentation of the *Integrated* part to a later Section of the Chapter. We can use the `arima()` to estimate the AR(p) model and the results shown below are similar in magnitude to the those obtained earlier:

```
# default estimation method is "ML", for OLS set method="CSS"
arima(spm$return, order=c(12, 0, 0))
```

```
Call:
arima(x = spm$return, order = c(12, 0, 0))

Coefficients:
        ar1     ar2   ar3    ar4    ar5     ar6     ar7    ar8     ar9   ar10   ar11   ar12  intercept
      0.182  -0.041  0.13  0.148  0.018  -0.190  -0.065  0.100  -0.123  0.039  0.053  0.007       0.43
s.e.  0.088   0.089  0.09  0.089  0.089   0.089   0.089  0.089   0.088  0.088  0.088  0.088       0.47

sigma^2 estimated as 15.7:  log likelihood = -358,  aic = 745
```

The coefficient estimates in this case are slightly different from the previous results since the default estimation method in this case is ML instead of OLS.

4.2.2 Lag selection

The most important modeling question when using AR(p) models is the choice of the lag order p. This choice is made by estimating the AR model for many values of p, e.g. from 0 to p_{max}, and then select the order that minimizes a criterion. A popular criterion is the Akaike Information Criterion (AIC) that is similar to the adjusted R^2 calculated for LRM, but with a different penalization term for the number of parameters included in the model. The Akaike Information Criterion (AIC) is calculated as

$$AIC(p) = \log(RSS/T) + 2 * (1 + p)/T$$

where RSS is the Residual Sum of Squares of the model, T is the sample size, and p is the lag order of the AR model. The AIC is calculated for models with different p and the lag selected is the one that

minimizes the criterion. This is done automatically by the function `ar()` when setting the argument `aic=TRUE` and the results are shown below for the monthly and daily returns:

```
ar(spm$return, aic=TRUE,  order.max=24, demean = FALSE, intercept = TRUE)
ar(spd$return, aic=TRUE,  order.max=24, demean = FALSE, intercept = TRUE)
```

```

Call:
ar(x = spm$return, aic = TRUE, order.max = 24, demean = FALSE,     intercept = TRUE)

Coefficients:
     1       2       3       4       5       6
 0.181  -0.053   0.158   0.152   0.010  -0.213

Order selected 6  sigma^2 estimated as  17.3

Call:
ar(x = spd$return, aic = TRUE, order.max = 24, demean = FALSE,     intercept = TRUE)

Coefficients:
     1      2      3      4      5      6      7      8      9     10     11     12     13     14     15
-0.106 -0.060  0.012 -0.037 -0.047  0.002 -0.029  0.020 -0.014  0.034 -0.016  0.033  0.002 -0.042 -0.047
    16     17     18     19     20     21
 0.046  0.011 -0.060  0.007  0.040 -0.030

Order selected 21  sigma^2 estimated as  1.59
```

The AIC has selected lag 6 for the monthly returns and 21 for the daily returns. However, a criticism against AIC is that it tends to select *large* models, that is, models with a large number of lags. In the case of the monthly returns, the ACF showed that they are hardly predictable based on the weak correlations and yet AIC suggests to use information up to 6 months earlier to predict next month returns. As for the daily returns, the order selected is 21 which seems large given the scarse predictability of the time series. An alternative criterion that can be used to select the optimal lag is the Bayesian Information Criterion (BIC) which is calculated as follows:

$$BIC(p) = \log(RSS/T) + \log(T) * (1 + p)/T$$

The only difference with AIC is that the number of lags in the model is multiplied by $\log(T)$ (instead of 2 for AIC). For time series with 8 or more observations the BIC penalization is larger relative to AIC and will lead to the selection of fewer lags. Unfortunately, the function `ar()` uses only the AIC and does not provide results for other selection criteria. The package `FitAR` provides the `SelectModel()` function that does the selection according to the criterion specified by the user, for example AIC or BIC. Below is the application to the monthly returns of the S&P 500 Index where the `spm$ret` is transformed to a `ts` object using `as.ts()` which is the time series type required by this function:

```
library(FitAR)
SelectModel(as.ts(spm$return), lag.max=24, Criterion="AIC")
SelectModel(as.ts(spm$return), lag.max=24, Criterion="BIC")
```

```
  p AIC-Exact AIC-Approx
1 6       371       -2.6
2 7       373       -1.8
3 8       374       -1.6
```

```
  p BIC-Exact BIC-Approx
1 0       380        5.8
2 1       381        9.6
3 2       385       11.1
```

BIC indicates the optimal lag is 0 while AIC selects 6. This is expected since BIC requires a larger improvement in fit to justify including additional lags. Also when considering the top 3 models, AIC chooses larger models relative to BIC.

```
SelectModel(as.ts(spd$return), lag.max=24, Criterion="AIC")
SelectModel(as.ts(spd$return), lag.max=24, Criterion="BIC")
```

```
   p AIC-Exact AIC-Approx
1 21      1269        -60
2 20      1270        -60
3 22      1270        -58
  p BIC-Exact BIC-Approx
1 2      1310      -12.9
2 1      1315      -19.7
3 3      1317       -6.8
```

At the daily frequency, the order selected by the two criteria is remarkably different: AIC suggests to include 21 lags of the dependent variable, while BIC only 2 days. In general, it is not always the case that more parsimonious models are preferable to larger models, although when forecasting time series it is preferable since they provide more stable forecasts. Adding the argument `Best=1` to the `SelectModel()` function returns the optimal order.

4.3 Forecasting with AR models

One of the advantages of AR models is that they are very suitable to produce forecasts about the future value of the series. Let's assume that the current time period is t and we observe the current observation. Assuming that we are using an AR(1) model, the forecast for the value of Y_{t+1} is given by

$$\hat{E}(Y_{t+1}|Y_t) = \hat{\beta}_0 + \hat{\beta}_1 * Y_t$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are the estimates of the parameters. If we want to forecast the variable two steps ahead, then the forecast is given by

$$\hat{E}(Y_{t+2}|Y_t) = \hat{\beta}_0 + \hat{\beta}_1 * \hat{E}(Y_{t+1}|Y_t) = \hat{\beta}_0(1 + \hat{\beta}_1) + \beta_1^2 * Y_t$$

The formula can be generalized to forecasting k steps ahead which is given by

$$\hat{E}(Y_{t+k}|Y_t) = \hat{\beta}_0 \left(\sum_{j=1}^{k} \hat{\beta}_1^{k-1} \right) + \hat{\beta}_1^k * Y_t$$

Let's forecast the daily and monthly returns of the S&P 500 Index. We already discussed how to select and estimate an AR model so that we need only to discuss how to produce the forecasts Y_{t+k} with k representing the forecast horizon. The functions `predict()` and `forecast()` (from package `forecast`) calculate the forecast based on a estimation object and the forecast horizon k. The code below shows

Table 4.1: Point and interval forecast for the monthly returns of the SP 500 Index starting in Sep 2017 and for 12 months ahead.

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Sep 11	0.42	-5.1	6	-8.1	8.9
Oct 11	0.42	-5.1	6	-8.1	8.9
Nov 11	0.42	-5.1	6	-8.1	8.9
Dec 11	0.42	-5.1	6	-8.1	8.9
Jan 12	0.42	-5.1	6	-8.1	8.9
Feb 12	0.42	-5.1	6	-8.1	8.9

Table 4.2: Point and interval forecast for the daily returns of the SP 500 Index starting in 2017-09-01 and for 12 days ahead.

	Point Forecast	Lo 50	Hi 50	Lo 90	Hi 90
2687	-0.028	-0.88	0.83	-2.1	2.1
2688	0.024	-0.84	0.89	-2.1	2.1
2689	0.034	-0.83	0.90	-2.1	2.1
2690	0.029	-0.83	0.89	-2.1	2.1
2691	0.029	-0.84	0.89	-2.1	2.1
2692	0.029	-0.83	0.89	-2.1	2.1

how to perform these three steps and produce 6-step ahead forecast, in addition to print a table with the results.

```
bic.monthly      <- SelectModel(as.ts(spm$return), lag.max = 24, Criterion = "BIC", Best=1)
fit.monthly      <- arima(spm$return, order = c(bic.monthly, 0, 0))
forecast.monthly <- forecast::forecast(fit.monthly, h = 6) # h is the forecast horizon
knitr::kable(as.data.frame(forecast.monthly), digits=4, caption=paste("Point and interval forecast for the monthly
             returns of the SP 500 Index starting in", end(spm)," and for 12 months ahead."))
```

```
bic.daily      <- SelectModel(as.ts(spd$return), lag.max = 24, Criterion = "BIC", Best=1)
fit.daily      <- arima(spd$return, order = c(bic.daily, 0, 0))
forecast.daily <- forecast::forecast(fit.daily, h = 6, level=c(50,90))
knitr::kable(as.data.frame(forecast.daily), digits=4, caption=paste("Point and interval forecast for the daily
             returns of the SP 500 Index starting in", end(spd)," and for 12 days ahead."))
```

The Table of the S&P 500 forecasts at the monthly frequency shows that they are equal at all horizons. This is due to the fact that BIC selected order 0 and in this case the forecast is given by $\hat{E}(Y_{t+k}|Y_t) = \hat{\beta}_0 = \hat{\mu}_Y$, where $\hat{\mu}_Y$ represents the sample average of the monthly returns. However, at the daily frequency the order selected is 2, and the forecasts start from a different value but rapidly converge toward the sample average of the daily returns. Visualizing the forecasts and the uncertainty around the forecast is a useful tool to understand the strength and the weakness of the forecasts. Figure 4.3 shows the forecast, the forecast intervals around the forecast, and the time series that is being forecast[4].

[4]The number of past observations that are plotted can be changed by setting the option `include=` to the number of

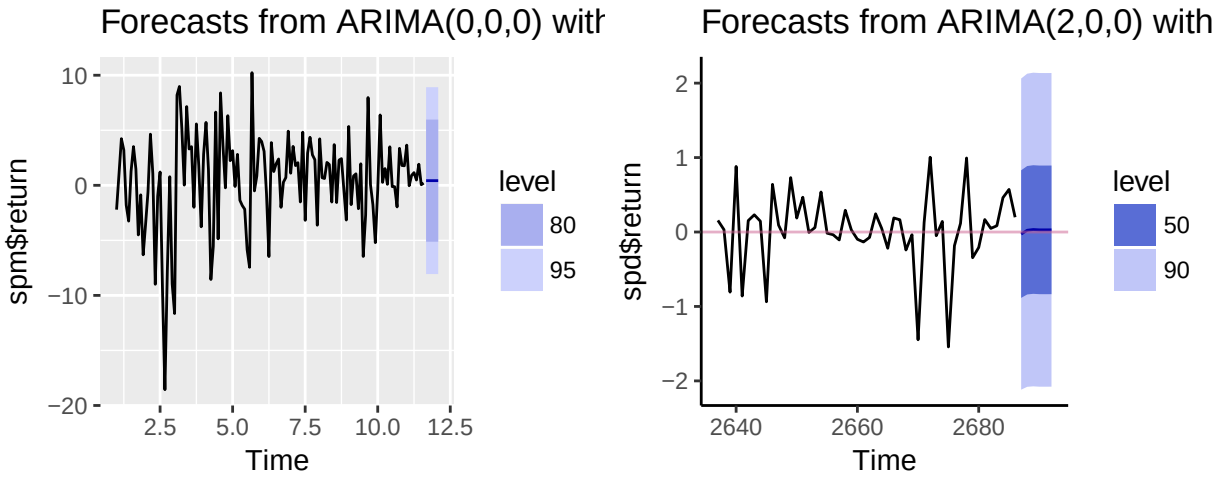


Figure 4.3: Point and interval forecasts for the monthly (left) and daily (right) returns of the SP 500 Index

```
library(ggfortify)
plot.monthly <- autoplot(forecast.monthly)
plot.daily <- autoplot(forecast.daily, include=50) + theme_classic() +
              geom_hline(yintercept = 0, color="hotpink3",alpha = 0.5)
grid.arrange(plot.monthly, plot.daily, ncol=2)
```

Let's consider a different example. Assume that we want to produce a forecast of the growth rate of real GDP for the coming quarters using time series models. This can be done as follows:

- download from `FRED` the time series of real GDP with ticker `GDPC1`
- transform the series to percentage growth rate
- select the order p^* using BIC
- estimate the AR(p^*) model
- produce the forecasts

As an example, assume that we are interested in forecasting the growth rate of GDP in the following quarter. Figure 4.4 shows the most recent 100 quarter of the real GDP growth rate until April 2017 together with 4 quarters of point and interval forecasts. The model forecasts that the GDP growth will moderately increase in the future quarters and that there is approximately 25% probability that output growth becomes negative.

```
gdp.level  <- getSymbols("GDPC1", src="FRED", auto.assign=FALSE)
gdp.growth <- 100 * diff(log(gdp.level)) %>% na.omit
gdp.p <- SelectModel(as.ts(gdp.growth), lag.max=12, Criterion="BIC", Best=1)
gdp.fit <- arima(gdp.growth, order=c(gdp.p, 0,0))
gdp.forecast <- forecast::forecast(gdp.fit, h=4, level=c(50,75,90))
autoplot(gdp.forecast, include=50) + theme_classic() + geom_hline(yintercept = 0, color="hotpink3")
```

observations that the user intends to show.

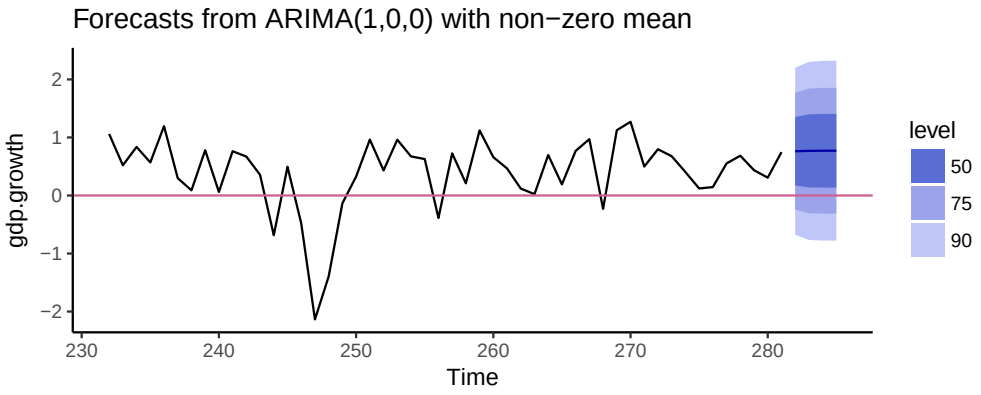


Figure 4.4: The growth rate of real GDP at the quarterly frequency.

4.4 Seasonality

Seasonality in a time series represents a regular pattern of higher/lower realizations of the variable in certain periods of the year. A seasonal pattern for daily data is represented by the 7 days of the week, for monthly data by the 12 months in a year, and for quarterly data by the 4 quarters in a year. For example, electricity consumption spikes during the summer months while being lower in the rest of the year. Of course there are many other factors that determine the consumption of electricity which might change over time, but seasonality captures the characteristic of systematically higher/lower values at certain times of the year. As an example, let's assume that we want to investigate if there is seasonality in the S&P 500 returns at the monthly frequency. To capture the seasonal pattern we use dummy variables that take value 1 in a certain month and zero in all other months. For example, we define the dummy variable JAN_t to be equal to 1 if month t is January and 0 otherwise, FEB_t takes value 1 every February and it is 0 otherwise, and similarly for the remaining months. We can then include the dummy variables in a regression model, for example,

$$Y_t = \beta_0 + \beta_1 X_t + \gamma_2 FEB_t + \gamma_3 MAR_t + \gamma_4 APR_t + \gamma_5 MAY_t +$$

$$\gamma_6 JUN_t + \gamma_7 JUL_t + \gamma_8 AUG_t + \gamma_9 SEP_t + \gamma_{10} OCT_t + \gamma_{11} NOV_t + \gamma_{12} DEC_t + \epsilon_t$$

where X_t can be lagged values of Y_t and/or some other relevant independent variable. Notice that in this regression we excluded one dummy variable (in this case JAN_t) to avoid perfect collinearity among the regressors (dummy variable trap). The coefficients of the other dummy variables should then be interpreted as the expected value of Y in a certain month relative to the expectation in January, once we control for the independent variable X_t. There is an alternative way to specify this regression which consists of including all 12 dummy variables and excluding the intercept:

$$Y_t = \beta_1 X_t + \gamma_1 JAN_t + \gamma_2 FEB_t + \gamma_3 MAR_t + \gamma_4 APR_t + \gamma_5 MAY_t +$$

$$\gamma_6 JUN_t + \gamma_7 JUL_t + \gamma_8 AUG_t + \gamma_9 SEP_t + \gamma_{10} OCT_t + \gamma_{11} NOV_t + \gamma_{12} DEC_t + \epsilon_t$$

The first regression is typically preferred because a test of the significance of the coefficients of the 11 monthly dummy variables provides evidence in favor or against seasonality. The same hypothesis could be evaluated in the second specification by testing that the 12 parameters of the dummy variables are equal to each other.

We can create the seasonal dummy variables by first defining a variable `sp.month` using the `month()` function from package `lubridate` which identifies the month for each observation in a time series object either with a number from 1 to 12 or with the month name. For example, for the monthly S&P 500 returns we have:

```
sp.month <- lubridate::month(spm$return, label=TRUE)
head(sp.month, 12)
```

```
 [1] Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < Oct < Nov < Dec
```

We can then use the `model.matrix()` function that simply converts the qualitative variable `sp.month` to a matrix of dummy variables with one column for each month and value 1 when the date of the observation is in each of the months. The code below shows how to convert the `sp.month` variable to the matrix of dummy variables `sp.month.mat`:

```
sp.month.mat          <- model.matrix(~ -1 + sp.month)
colnames(sp.month.mat) <- levels(sp.month)
head(sp.month.mat, 8)
```

```
  Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1   0   1   0   0   0   0   0   0   0   0   0   0
2   0   0   1   0   0   0   0   0   0   0   0   0
3   0   0   0   1   0   0   0   0   0   0   0   0
4   0   0   0   0   1   0   0   0   0   0   0   0
5   0   0   0   0   0   1   0   0   0   0   0   0
6   0   0   0   0   0   0   1   0   0   0   0   0
7   0   0   0   0   0   0   0   1   0   0   0   0
8   0   0   0   0   0   0   0   0   1   0   0   0
```

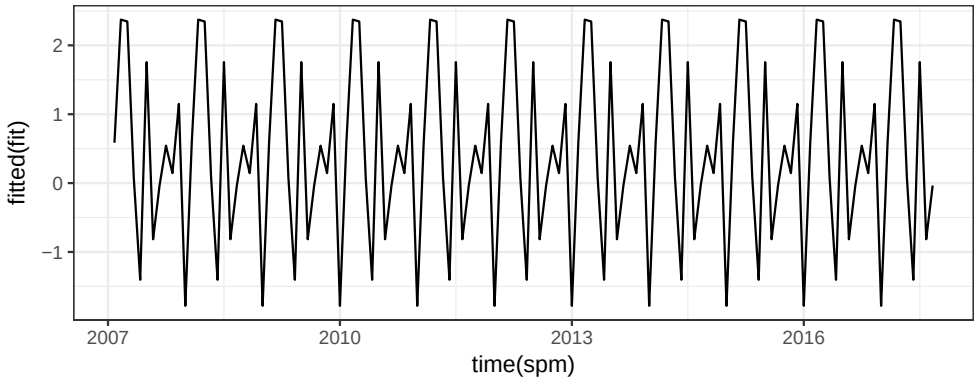
To illustrate the use of seasonal dummy variables we consider a simple example in which the monthly return of the S&P 500 is regressed on the 12 monthly dummy variables. As discussed before, to avoid the dummy variable trap the regression models does not include the intercept in favor of including the 12 dummy variables. The model is implemented as follows:

```
fit <- lm(spm$return ~ -1 + sp.month.mat)
```

```
                Estimate Std. Error t value Pr(>|t|)
sp.month.matJan   -1.782        1.4  -1.298    0.197
sp.month.matFeb    0.586        1.3   0.448    0.655
sp.month.matMar    2.373        1.3   1.814    0.072
sp.month.matApr    2.347        1.3   1.794    0.075
sp.month.matMay    0.082        1.3   0.063    0.950
sp.month.matJun   -1.404        1.3  -1.073    0.285
sp.month.matJul    1.757        1.3   1.342    0.182
sp.month.matAug   -0.816        1.3  -0.623    0.534
sp.month.matSep   -0.035        1.3  -0.027    0.979
sp.month.matOct    0.543        1.4   0.396    0.693
sp.month.matNov    0.144        1.4   0.105    0.916
sp.month.matDec    1.150        1.4   0.838    0.404
```

The results indicate that, in most months, the expected return is not significantly different from zero, except for March and April. In both cases the coefficient is positive which indicates that in those months it is expected that returns are higher relative to the other months. To develop a more intuitive

understanding of the role of the seasonal dummies, the graph below shows the fitted or predicted returns from the model above. In particular, the expected return of the S&P 500 in January is -1.78%, that is, $E(R_t|JAN_t = 1) = -1.78$, while in February the expected return is $E(R_t|FEB_t = 1) = 0.59\%$ and so on. These coefficients are plotted in the graph below and create a regular pattern that is expected to repeat every year:



Returns seems to be positive in the first part of the year and then go into negative territory during the summer months only to return positive toward the end of the year. However, keep in mind that only March and November are significant at 10%. We can also add the lag of the S&P 500 return to the model above to see if the significance of the monthly dummy variables changes and to evaluate if the goodness of the regression increases:

```
fit <- dyn$lm(return ~ -1 + lag(return, 1) + sp.month.mat, data = spm)
```

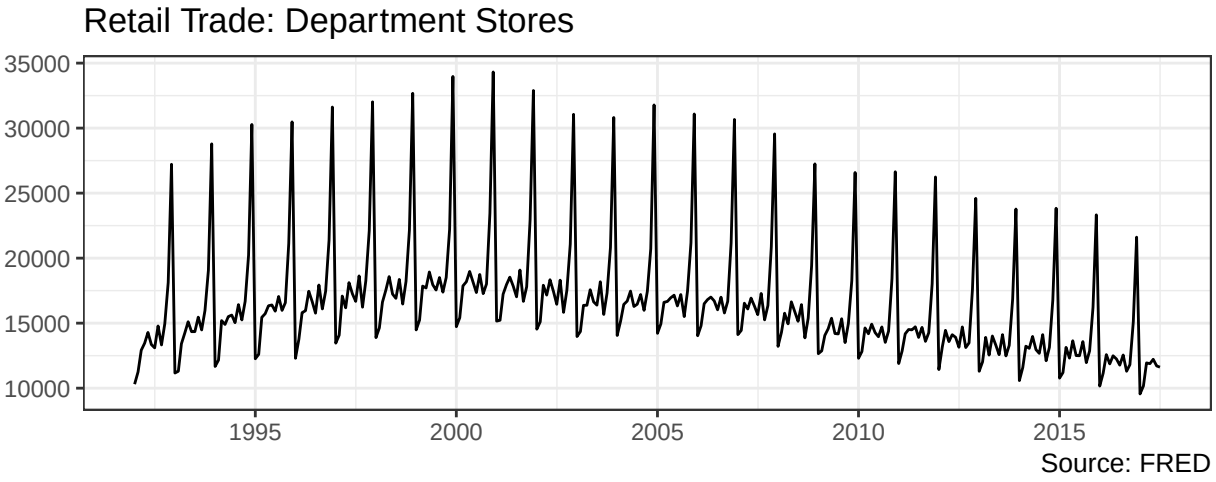
```
               Estimate Std. Error t value Pr(>|t|)
lag(return, 1)    0.201      0.092   2.201    0.030
sp.month.mat1    -2.014      1.357  -1.484    0.141
sp.month.mat2     1.225      1.363   0.899    0.371
sp.month.mat3     2.255      1.291   1.746    0.083
sp.month.mat4     1.869      1.308   1.428    0.156
sp.month.mat5    -0.390      1.308  -0.299    0.766
sp.month.mat6    -1.421      1.290  -1.101    0.273
sp.month.mat7     2.039      1.297   1.573    0.119
sp.month.mat8    -1.170      1.300  -0.900    0.370
sp.month.mat9     0.129      1.292   0.100    0.921
sp.month.mat10    0.555      1.353   0.410    0.683
sp.month.mat11    0.035      1.354   0.026    0.980
sp.month.mat12    1.121      1.353   0.828    0.409
```

The `-1` in the `lm()` formula is introduced when we want to exclude the intercept from the model.

Seasonality is a common characteristics of macroeconomic variables. Typically, we analyze these variables on a *seasonally-adjusted* basis, which means that the statistical agencies have already removed the seasonal component from the variable. However, they also provide the variables before the adjustment as in the case of the *Department Stores Retail Trade* series (FRED ticker `RSDSELDN`) at the monthly frequency. The time series graph for this variable from January 1992 is shown below.

```
library(quantmod)
deptstores <- getSymbols("RSDSELDN", src="FRED", auto.assign = FALSE)
qplot(time(deptstores), deptstores, geom="line") +
```

```
  labs(x=NULL, y=NULL, title="Retail Trade: Department Stores", caption="Source: FRED") + theme_bw()
```




```
 [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

The seasonal pattern is quite clear in the data and it seems to happen toward the end of the year. We can conjecture that the spike in sales is probably associate with the holiday season in December, but we can test this hypothesis by estimating a linear regression model in which we include monthly dummy variables as explanatory variables to account for this pattern. The regression model for the sales in $ of department stores, denoted by Y_t, is

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \gamma_2 FEB_t + \gamma_3 MAR_t + \gamma_4 APR_t + \gamma_5 MAY_t +$$

$$\gamma_6 JUN_t + \gamma_7 JUL_t + \gamma_8 AUG_t + \gamma_9 SEP_t + \gamma_{10} OCT_t + \gamma_{11} NOV_t + \gamma_{12} DEC_t + \epsilon_t$$

where, in addition to the monthly dummy variables, we have the first order lag of the variable. The regression results are shown below:

```
dept.month.mat         <- model.matrix(~ -1 + dept.month)
colnames(dept.month.mat) <- levels(dept.month)
fit <- arima(deptstores, order=c(1,0,0), xreg = dept.month.mat[,-1])
```

```
Call:
arima(x = deptstores, order = c(1, 0, 0), xreg = dept.month.mat[, -1])

Coefficients:
        ar1  intercept      Feb       Mar       Apr       May       Jun       Jul       Aug       Sep       Oct
      0.916  12633.434  717.578  2679.593  2586.464  3482.078  2830.702  2343.121  3696.211  2035.781  3233.145
s.e.  0.023    568.858  161.805   217.716   252.645   274.951   287.737   292.344   289.376   277.676   255.967
            Nov        Dec
       7099.968  16212.121
s.e.    221.148    164.707

sigma^2 estimated as 684981:  log likelihood = -2499.13,  aic = 5026.26
```

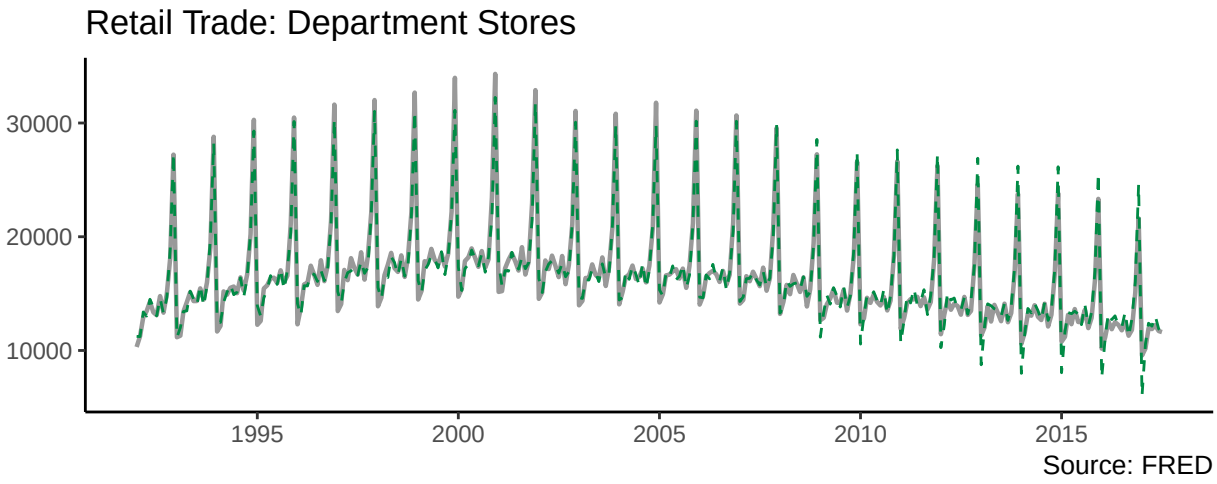


Figure 4.5: Retail trade and fitted value from the time series model.

The results indicate the retail sales at department stores are highly persistent with an AR(1) coefficient of 0.916. To interpret the estimate of the seasonal coefficients, notice that the dummy for the month of January was left out so that all other seasonal dummies should be interpreted as the difference in sales relative to the first month of the year. The results indicate that all coefficients are positive and thus retail sales are higher than January. From the low levels of January, sales seems to increase and remain relative stable during the summer months, and then increase significantly in November and December. Figure 4.5 of the variable and the model fit (`springgreen4` dashed line).

```
ggplot() + geom_line(aes(time(deptstores), deptstores),color="gray60", size=0.8) +
           geom_line(aes(time(deptstores), fitted(fit)), color="springgreen4", linetype=2) +
          theme_classic() + labs(x=NULL, y=NULL, title="Retail Trade: Department Stores", caption="Source: FRED")
```

So far we estimated the model and compared the realization of the variable with the prediction or fitted value of the model. In practice, it is useful to perform an *out-of-sample* exercise which consist of estimating the model up to a certain date and forecast the future even though we observe the realization for the out-of-sample period. This simulates the situation of a forecaster that in real-time observes only the past and is interested to forecast the future. Figure 4.6 shows the results of an out-of-sample exercise in which an AR(4) model with seasonal dummies is estimated with data up to December 2014 and forecasts are produced from that date onward. The blue line shows the forecast, the shaded areas the 80% and 95% forecast intervals, and the red line represents the realization of the variable. In the first part of the out-of-sample period, the forecasts underpredicted the realization of sales, but underpredicted the peak of sales at the end of the year. In the second year, the forecasts during the year were closer to the realization, but again the model overpredicted end-of-year sales.

```
library(forecast)
index = sum(time(deptstores) < "2014-12-31")
dept.insample   <- window(deptstores, end = "2014-12-31")
month.insample  <- dept.month.mat[1:index,]
dept.outsample  <- window(deptstores, start = "2014-12-31")
month.outsample <- dept.month.mat[(index+1):length(dept.month),]
```

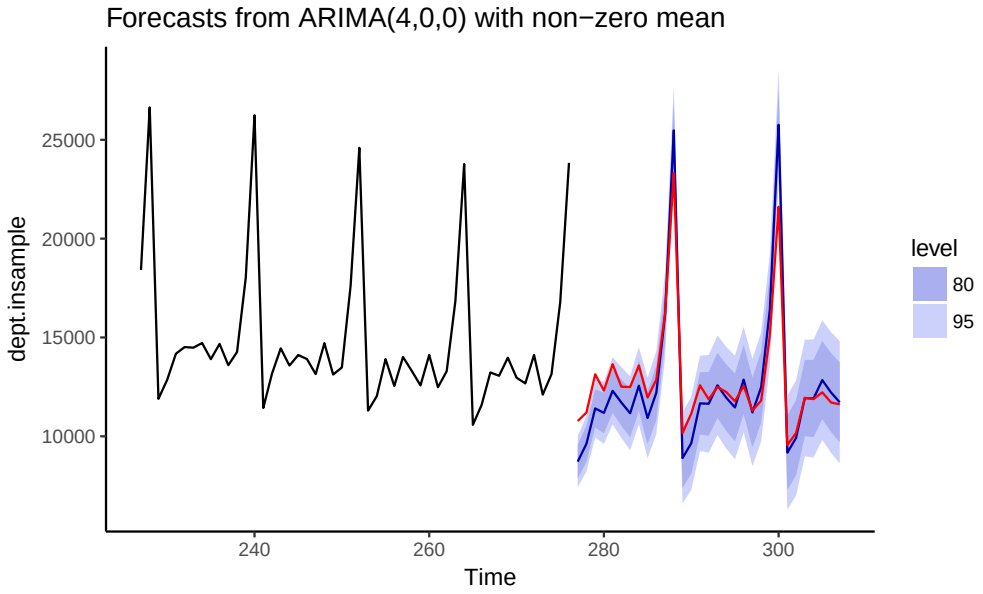


Figure 4.6: Out-of-sample forecast of Retail Trade starting in December 2014 and for forecast horizons 1 to 31 .

```
fit <- arima(dept.insample, order = c(4,0,0), xreg = month.insample[,-1])
autoplot(forecast(fit, h = length(dept.outsample), xreg = month.outsample[,-1]), include=50) +
  theme_classic() +
  geom_line(aes((index+1):(index+length(dept.outsample)), dept.outsample), color="red")
```

4.5 Trends in time series

A trend is defined as the tendency of an economic or financial time series to grow over time. Examples are provided in Figure 4.7 that shows the real Gross Domestic Product (FRED ticker: `GDPC1`) and the S&P 500 index (YAHOO ticker: `^GSPC`) at the quarterly frequency.

```
gdp.level        <- getSymbols("GDPC1", src="FRED", auto.assign=FALSE)
sp.level         <- getSymbols("^GSPC", src="yahoo", auto.assign=FALSE, from="1947-01-01")
macrodata        <- merge(gdp.level, Ad(to.monthly(sp.level)))
names(macrodata) <- c("GDP","SP500")
macrodata.q      <- to.quarterly(macrodata, OHLC = FALSE)
p1 <- ggplot(macrodata.q) + geom_line(aes(time(macrodata.q), GDP), alpha=0.4) + theme_classic() +
      labs(x=NULL, y=NULL, title="GDP", caption="Source: FRED")
p2 <- ggplot(macrodata.q) + geom_line(aes(time(macrodata.q), SP500), alpha=0.4) + theme_classic() +
      labs(x=NULL, y=NULL, title="SP 500", caption="Source: Yahoo")
```

The series have in common the feature of growing over time with no tendency to revert back to the mean. Actually, the trending behavior of the variables implies that the mean of the series is also increasing over time rather than being approximately constant. For example, if we were to estimate the mean

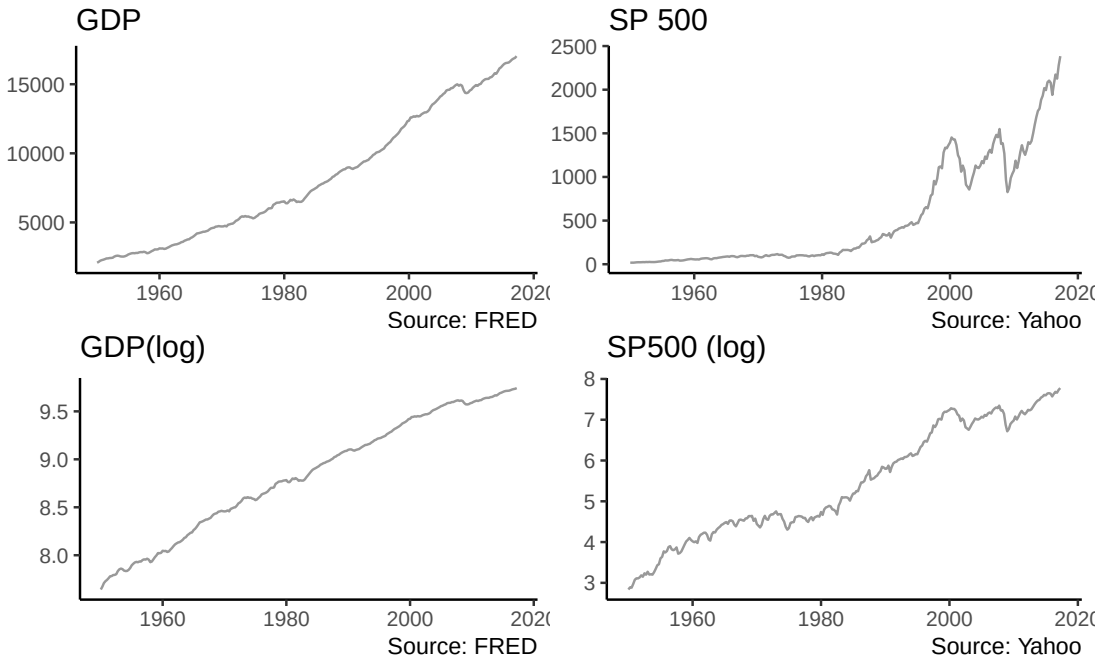


Figure 4.7: Real GDP (FRED: GDPC1) and the Standard and Poors 500 Index at the quarterly frequency starting in January 1947. Bottom graphs are in logarithmic scale.

of GDP and the S&P 500 in 1985 it would not have been a good predictor of the future value of the mean because the variables kept growing over time. This type of series are called *non-stationary* since some features of the distribution (e.g., the mean and/or the variance) change over time instead of being constant as it is the case for *stationary* processes. For the variables in the Figure, we can thus conclude that they are clearly non-stationary. In addition, very often in economics and finance we prefer to take the natural logarithm of the variables which makes exponential growth approximately linear. The same variables discussed above are plotted in natural logarithm in Figure 4.7. Taking the log of the variables is particularly relevant for the S&P 500 Index which shows a considerable exponential behavior, at least until the end of the 1990s.

4.5.1 Deterministic Trend

A simple approach to model the non-stationarity of these time series is to assume that they follow a deterministic trend, which we can assume that it is linear as a starting point. We thus assume that the series Y_t evolves according to the model

$$Y_t = \beta_0 + \beta_1 * t + d_t$$

where the deviation d_t is a zero mean stationary random variable (e.g., an AR(1) process). The independent variable, denoted t, represents the time trend and takes value 1 for the first observation, value 2 for the second observation and value T for the last observation[5]. This model decomposes the series Y_t in two components:

[5] Any sequence of numbers can constitute a trend as long as the difference between consecutive values is 1.

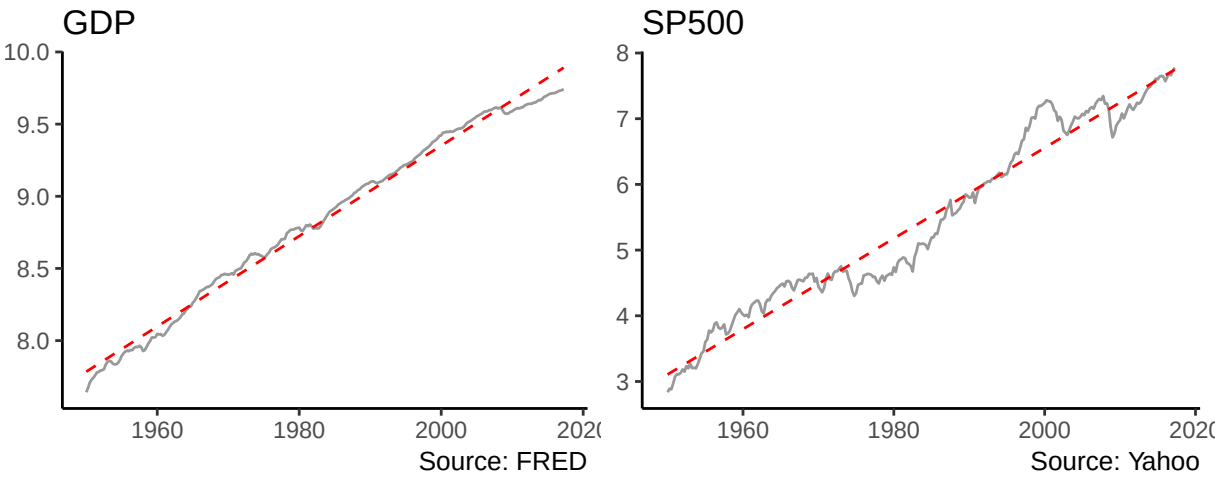


Figure 4.8: Linear trend model for the real GDP and the SP500.

- a *permanent* (non-stationary) component represented by $\beta_0 + \beta_1 * t$ that captures the long-run growth of the series
- a *transitory* (stationary) component d_t that captures the deviations from the long-run trend (e.g., business cycle fluctuations)

This type of trend is called *deterministic* since it makes the prediction that every time period the series Y_t is expected to increase by β_1 units. It is often referred as the *trend-stationary model* after the two components that constitute the model. The model can be estimated by OLS using the `lm()` function:

```
macrodata.q$trend <- 1:nrow(macrodata.q) # creates the trend variable
fit.gdp           <- lm(log(GDP) ~ trend, data=macrodata.q)
```

	(Intercept)	trend
GDP	7.77688	0.00783
SP500	3.08968	0.01725

where the estimate of β_1 is 0.0078 and indicates that real GDP is expected to grow 0.78% every quarter[6]. For the S&P 500 $\hat{\beta}_1 = 0.0173$ and represents a quarterly growth of 1.73%. The application of the linear trend model to the series shown above gives as a result the dashed trend line shown in Figure 4.8:

```
fit.gdp   <- lm(log(GDP) ~ trend, data=macrodata.q)
fit.sp500 <- dyn$lm(log(SP500) ~ trend, data=macrodata.q)

p1 <- ggplot(macrodata.q) + geom_line(aes(time(macrodata.q), log(GDP)), alpha=0.4) +
           geom_line(aes(time(macrodata.q), fitted(fit.gdp)), color="red", linetype="dashed")
p2 <- ggplot(macrodata.q) + geom_line(aes(time(macrodata.q), log(SP500)), alpha=0.4) +
           geom_line(aes(time(macrodata.q), fitted(fit.sp500)), color="red", linetype="dashed")
p1 <- p1 + theme_classic() + labs(x=NULL,y=NULL, title="GDP", caption="Source: FRED")
```

The deviation of the series from the fitted trend line are small for GDP, but for the S&P 500 index they are persistent and last for long periods of time (i.e., 10 years or even longer). Such persistent deviations

[6]We multiply the coefficient by 100 and give a percentage interpretation because the dependent variable is in logarithm but the independent is linear.

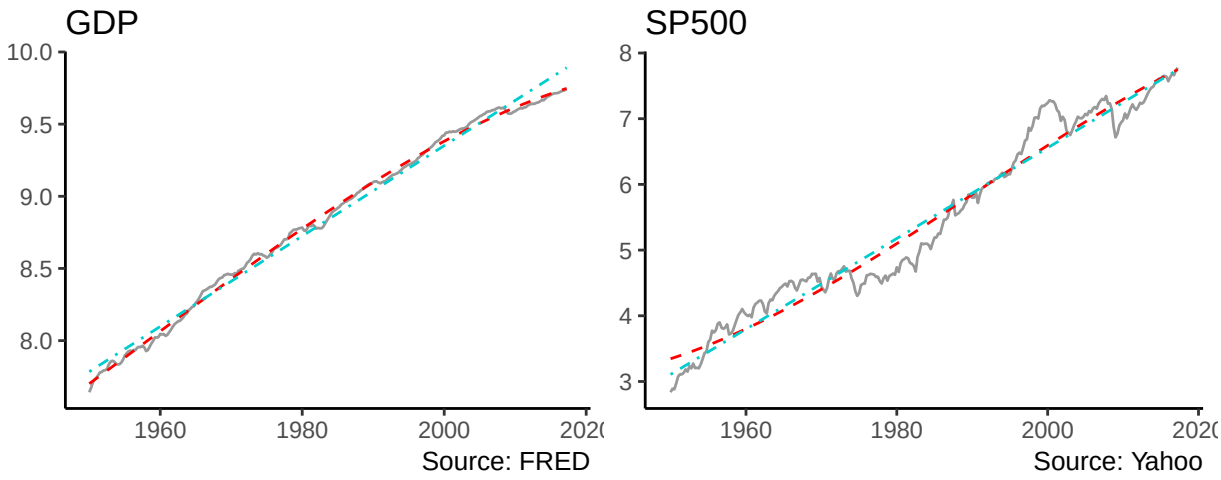


Figure 4.9: Linear and cubic trend model for the real GDP and the SP500.

might be due to the inadequacy of the linear trend model and the need to consider a nonlinear trend. This can be accommodated by adding a quadratic and/or cubic term to the linear trend model which becomes:

$$Y_t = \beta_0 + \beta_1 * t + \beta_2 * t^2 + \beta_3 * t^3 + d_t$$

where t^2 and t^3 represent the square and cube of the trend variable. The implementation in R requires the only additional step of including the square and cube of `trend` in the `lm()` formula:

```
fitsq   <- lm(log(GDP) ~ trend + I(trend^2), data=macrodata.q)
fitcb   <- lm(log(GDP) ~ trend + I(trend^2) + I(trend^3), data=macrodata.q)
```

```
            Estimate Std. Error   t value Pr(>|t|)
(Intercept)   7.6611     0.0064 1204.1189        0
trend         0.0104     0.0001   95.7755        0
I(trend^2)    0.0000     0.0000  -24.3167        0
            Estimate Std. Error  t value Pr(>|t|)
(Intercept)   7.6928     0.0080 956.2543   0.0000
trend         0.0090     0.0003  35.0520   0.0000
I(trend^2)    0.0000     0.0000   1.5317   0.1268
I(trend^3)    0.0000     0.0000  -5.8996   0.0000
```

The linear (dashed line) and cubic (dot-dash line) deterministic trends are shown in Figure 4.9. For the case of GDP the differences between the two lines is not visually large, although the quadratic and/or cubic regression coefficients might be statistically significant at conventional levels. In terms of fitness, the AIC of the linear model is -732.16 and -1045.44 and -1076.64 for the quadratic and cubic models, respectively. Hence, in this case we would select the cubic model which does slightly better relative to the quadratic, and significantly better relative to the linear trend model[7]. However, it could be argued that the deterministic trend model might not represent well the behavior of the S&P 500 index since there are significant departures of the series from the trend models for long periods of time.

Another way to visualize the ability of the trend-stationary model to explain these series is to plot the residuals or deviation from the cubic trend, d_t. Figure 4.10 shows the time series graph of the d_t for the

[7]The best model is the one that minimizes AIC and BIC.

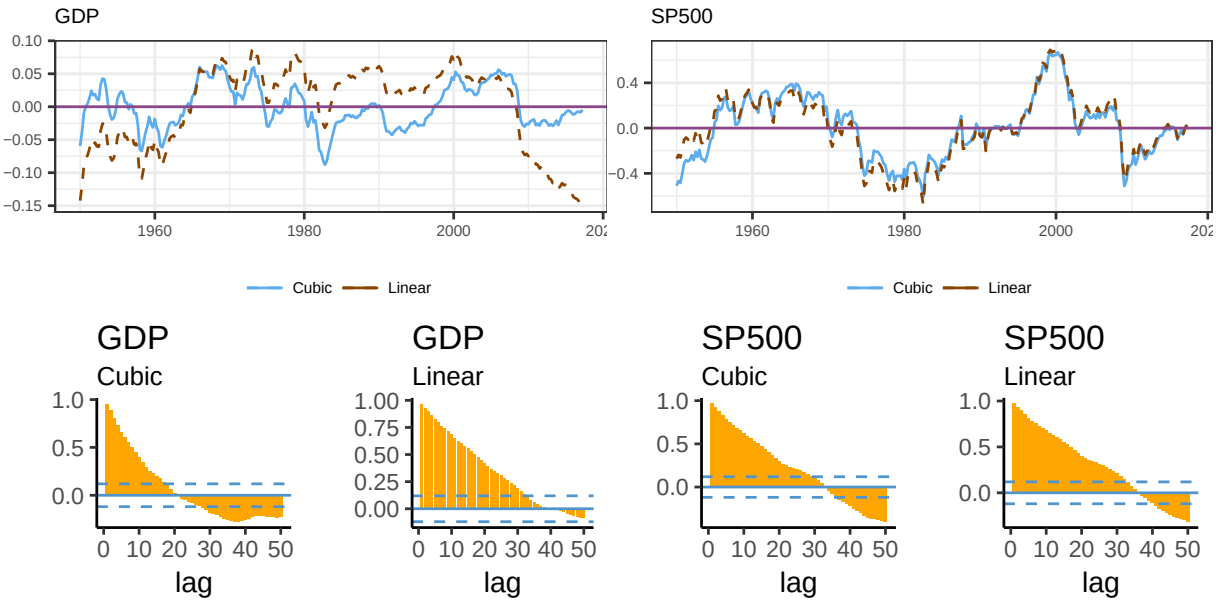


Figure 4.10: Deviation of the logarithm of real GDP and SP 500 from the linear and cubic trend model and its ACF up to lag 50.

linear and cubic models and the ACF functions with lag up to 50 (quarters for GDP and months for S&P 500):

```
p1 <- qplot(time(macrodata.q), residuals(fitcb.gdp), data=macrodata.q, geom="line", color="Cubic") +
          geom_line(aes(time(macrodata.q), residuals(fit.gdp), color="Linear"), linetype="dashed") +
          geom_hline(yintercept = 0, color="orchid4") + labs(title="GDP") +
          scale_colour_manual("", breaks=c("Cubic","Linear"), values=c("steelblue2","darkorange4"))
p2 <- qplot(time(macrodata.q), residuals(fitcb.sp500), data=macrodata.q, geom="line", color="Cubic") +
          geom_line(aes(time(macrodata.q), residuals(fit.sp500), color="Linear"), linetype="dashed") +
          geom_hline(yintercept = 0, color="orchid4") +  labs(title="SP500") +
          scale_colour_manual("", breaks=c("Cubic","Linear"), values=c("steelblue2","darkorange4"))
p3 <- ggacf(residuals(fitcb.gdp), 50) + labs(title="GDP", subtitle="Cubic")
p4 <- ggacf(residuals(fit.gdp), 50)+ labs(title="GDP", subtitle="Linear")
p5 <- ggacf(residuals(fitcb.sp500), 50) + labs(title="SP500", subtitle="Cubic")
p6 <- ggacf(residuals(fit.sp500), 50) + labs(title="SP500", subtitle="Linear")
mylayout <- rbind(c(1,1,2,2),
                  c(3,4,5,6))
grid.arrange(p1, p2, p3, p4,p5, p6, layout_matrix=mylayout)
```

The ACF shows clearly that the deviation of the log GDP from the cubic trend is persistent but with rapidly decaying values relative to the residuals of the linear trend model. On the other hand, for the S&P 500 it seems that the serial correlation decays at a slower rate, that might be an indication of a non-stationary time series. Even when we account for the trend in a variable, the deviation from the trend d_t might still be non-stationary and we will discuss later how to formally test this hypothesis.

4.5.2 Stochastic Trend

An alternative model that is often used in asset and option pricing is the *random walk with drift* model. The model takes the following form:

$$Y_t = \mu + Y_{t-1} + \epsilon_t$$

where μ is a constant and ϵ_t is an error term with mean zero and variance σ^2. The random walk model assumes that the expected value of Y_t is equal to the previous value of the series (Y_{t-1}) plus a constant term μ (which can be positive or negative), that is, $E(Y_t|Y_{t-1}) = \mu + Y_{t-1}$. The model can also be reformulated by substituting backward the value of Y_{t-1}, that is, $\mu + Y_{t-2} + \epsilon_{t-1}$ and we obtain $Y_t = 2 * \mu + Y_{t-2} + \epsilon_t + \epsilon_{t-1}$. We can then continue by substituting Y_{t-2}, Y_{t-3}, and so on until we reach the initial value Y_0 and the model can be written as

$$\begin{aligned} Y_t &= \mu + Y_{t-1} + \epsilon_t \\ &= \mu + (\mu + Y_{t-2} + \epsilon_{t-1}) + \epsilon_t \\ &= \mu + \mu + (\mu + Y_{t-3} + \epsilon_{t-2}) + \epsilon_{t-1} + \epsilon_t \\ &= \cdots \\ &= Y_0 + \mu t + \sum_{j=1}^{t} \epsilon_{t-j+1} \end{aligned}$$

This shows that a random walk with drift model can be expressed as the sum of two components (in addition to the starting value Y_0):

- deterministic trend (μt)
- the sum of all past errors/shocks to the series

In case the drift term μ is set equal to zero, the model reduces to $Y_t = Y_{t-1} + \epsilon_t = Y_0 + \sum_{j=1}^{t} \epsilon_{t-j+1}$ which is called the *random walk model* (without drift since μ is equal to zero). Hence, another way to think of the random walk model with drift is as the sum of a deterministic linear trend and a random walk process.

The relationship between the trend-stationary and the random walk with drift models becomes clear if we assume that the deviation from the trend d_t follow an AR(1) process, that is, $d_t = \phi d_{t-1} + \epsilon_t$, where ϕ is the coefficient of the first lag which is assumed to be smaller than 1 in absolute value and ϵ_t is a mean zero random variable. We can do backward substitution of the AR term in the trend-stationary model, that is,

$$\begin{aligned} Y_t = \beta_0 + \beta_1 t \quad &+ \quad d_t \\ &+ \quad \phi d_{t-1} + \epsilon_t \\ &+ \quad \phi^2 d_{t-2} + \phi * \epsilon_{t-1} + \epsilon_t \\ &+ \quad ... \\ &+ \quad \epsilon_t + \phi * \epsilon_{t-1} + \phi^2 * \epsilon_{t-2} + \cdots + \phi^{t-1}\epsilon_1 \\ &+ \quad \sum_{j=1}^{t} \phi^{j-1} \epsilon_{t-j+1} \end{aligned}$$

Comparing this equation with the one obtained for the random walk with drift model we find that the former is a special case of the latter for $\phi = 1$. Hence, the only difference between these models consists of the assumption on the persistence of the deviation from the trend. If the coefficient ϕ is less than 1 the deviation is stationary and thus the trend-stationary model can be used to de-trend[8] the series and then conduct the analysis on the deviation. However, when $\phi = 1$ the deviation from the trend as well is non-stationary (i.e., random walk) and the approach just described is not valid. We will discuss later what to do in this case. A more practical way to understand the issue of the (non-)stationarity of the deviation from the trend is to think in terms of the speed at which the series is likely to revert back to the trend-line. Series that oscillate around the trend are stationary while persistent deviations from the trend (slow reversion) are an indication of non-stationarity. How do we know if a series (e.g., the deviation from the trend) is stationary or not? In the following section we will discuss a test that evaluates this hypothesis and provides guidance as to what modeling approach to take.

The previous analysis of real GDP shows that the deviations of GDP from its trend seem to revert to the mean faster relative to the other two series: this can be seen both in the time series plot and also from the quickly decaying ACF. The estimate of the trend-stationary model shows that we expect GDP to grow around 0.774% per quarter (approximately 3.096% annualized), although GDP alternates periods above trend (expansions) and periods below trend (recessions). Alternating between expansions and recessions captures the mean-reverting nature of the GDP deviations from the long-run trend. We can evaluate the ability of the trend-stationary model to capture the expansion-recession feature of the business cycle by comparing the periods of positive and negative deviations with the peak and trough dates of the business cycle decided by the NBER dating committee. In the graph below we plot the deviation from the cubic trend estimated earlier together with the shaded areas that indicate the period of recessions.

```
# dates from the NBER business cycle dating committee
xleft  = c(1953.25, 1957.5, 1960.25, 1969.75, 1973.75, 1980,
           1981.5, 1990.5, 2001, 2007.917) # beginning
xright = c(1954.25, 1958.25, 1961, 1970.75, 1975, 1980.5, 1982.75,
           1991, 2001.75, 2009.417) # end

#fitgdp is the lm() object for the cubic trend model
ggplot() + geom_rect(aes(xmin=xleft, xmax=xright, ymin=rep(-0.10,10), ymax=rep(0.10, 10)),fill="lightcyan3") +
  geom_hline(yintercept = 0, color="grey35", linetype="dashed") +
  geom_line(aes(time(macrodata.q), residuals(fitcb.gdp))) +
  theme_classic() + labs(x=NULL, y=NULL)
```

Overall, there is a tendency for the deviation to sharply decline during recessions (shaded areas), and then increase during the recovery period and the expansion, which seems to have last longer since the mid-1980s.

Earlier we discussed that the distribution of a non-stationary variable changes over time. We can verify the stationarity properties of the trend-stationary model and the random walk with drift model by deriving the mean and variance of Y_t. Under the trend-stationary model the dynamics follows $Y_t = \beta_0 + \beta_1 t + d_t$ and we can make the simplifying assumption that $d_t = \phi d_{t-1} + \epsilon_t$ with ϵ_t a mean zero and variance σ^2_ϵ error term. Based on these assumptions, we obtain that $E(d_t) = 0$ and $Var(d_t) = \sigma^2_\epsilon/(1-\phi^2)$ so that:

- $E_t(Y_t) = \beta_0 + \beta_1 t$

[8] De-trending means running a regression of Y_t on the trend variable and analyze the residuals of this regression.

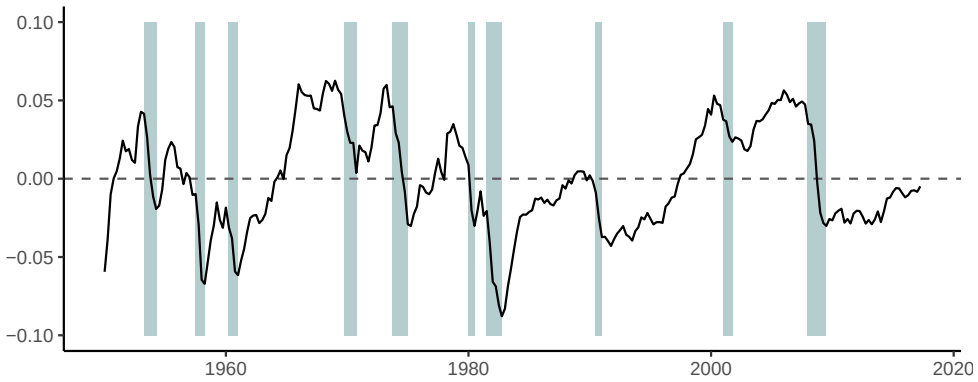


Figure 4.11: Deviation of the log real GDP from a cubic trend. The shaded areas represent the NBER recession periods.

- $Var_t(Y_t) = Var(d_t) = \sigma_\epsilon^2/(1-\phi^2)$

This demonstrates that the mean of a trend-stationary variable is a function of time and not constant, while the variance is constant. On the other hand for the random walk with drift model[9] we have that:

- $E_t(Y_t) = E_t\left(Y_0 + \mu t + \sum_{j=1}^{t} \epsilon_{t-j+1}\right) = Y_0 + \mu t$
- $Var_t(Y_t) = t\sigma_\epsilon^2$

From these results we see that for the random walk with drift model both the mean and the variance are time varying, while for the trend-stationary model only the mean varies with time.

The main difference between the trend-stationary and random walk with drift models thus consists in the non-stationarity properties of the deviations from the deterministic trend. For the trend-stationary model the deviations are considered stationary and they can be analysed using regression models estimated by OLS. However, for the random walk with drift the deviations are non-stationary and its time series cannot be considered in regression models because of several statistical issues that will be discussed in the next Section, followed by a discussion of an approach to statistically test if a series is non-stationary and non-stationary around a (deterministic) trend.

4.5.3 Why non-stationarity is a problem for OLS?

The estimation of time series models can be conducted by standard OLS techniques, as long as the series is stationary. If this is the case, the OLS estimator is consistent and t-statistics are distributed according to the Student t distribution. However, these properties fail to hold when the series is non-stationary and three problems arise:

1. the OLS estimate of the AR coefficients is biased in small samples
2. the t statistic is not normally distributed (even in large samples)
3. the regression of a non-stationary variables on a non-stationary variable leads to spurious results of dependence between the two series

We will discuss and illustrate these problems in the context of a simulation study using `R`. To show the first fact we follow the steps:

[9] In the derivation that follows we assume that Y_0 is constant similarly to β_0 for the trend-stationary model.

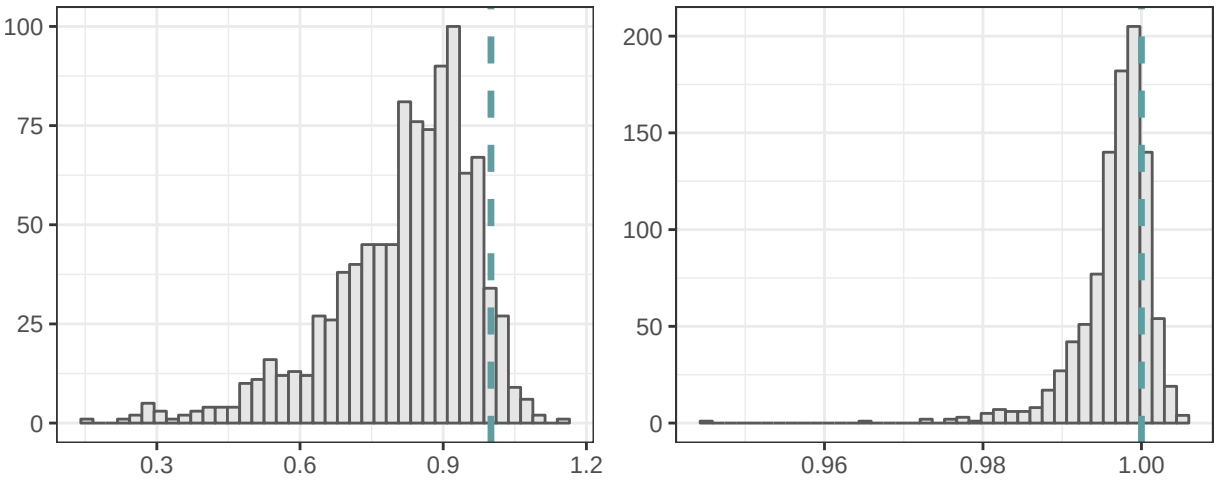


Figure 4.12: Simulated distribution of the OLS estimate of the AR(1) coefficient when the true generating process is a random walk with drift. Sample size is 25 for the left plot and 500 for the right plot.

1. simulate B time series from a random walk with drift model $Y_t = \mu + Y_{t-1} + \epsilon_t$ for $t = 1, \cdots, T$
2. estimate an AR(1) model $Y_t = \beta_0 + \beta_1 * Y_{t-1} + \epsilon_t$
3. store the B OLS estimates of β_1

We perform this simulation for a short and a long time series ($T = 25$ and $T = 500$, respectively) and compare the mean, median, and histogram of the $\hat{\beta}_1$ across the B simulations. Below is the code for $T = 25$ and the histogram of the simulated values for $T = 25$ and 500 is provided in Figure 4.12:

```
set.seed(1234)

T      = 25              # length of the time series
B      = 1000            # number of simulation
mu     = 0.1             # value of the drift term
sigma  = 1               # standard deviation of the error
beta25 = numeric(B)      # object to store the DF test stat

for (b in 1:B)
{
  Y        <- as.xts(arima.sim(n=T, list(order=c(0,1,0)), mean=mu, sd=sigma))
  fit      <- dyn$lm(Y ~ lag(Y,1))               # OLS estimation of DF regression
  beta25[b] <- summary(fit)$coef[2,1]                    # stores the results
}

# plotting
p1 <- ggplot() + geom_histogram(aes(beta25),color="grey35", fill="grey90", bins=40) + theme_bw() + labs(x=NULL,y=NU
  geom_vline(xintercept = 1, color="cadetblue", size=1.2, linetype="dashed")

T        = 500         # length of the time series
```

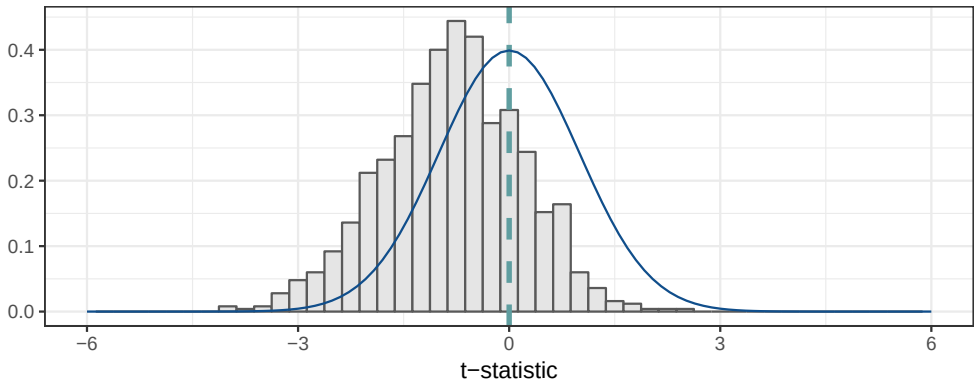


Figure 4.13: Simulated distribution of t-statistic of the OLS estimate of the AR(1) coefficient when the true generating process is a random walk with drift. Sample size is 500.

The histogram when the sample size is 25 shows that the $\hat{\beta}_1$ over 1000 simulations ranges between 0.5 and 0.5 with a mean of 0.5 and median of 0.5. Both the mean and the median are significantly smaller relative to the true value of 1. This demonstrates the bias in the OLS estimates in small samples when the variable is non-stationary. However, this bias has a tendency to decline for larger sample sizes as shown in the right histogram of Figure 4.12 where the sample size is set to 500. In this case the min/max estimates are 0.5 and 0.5 with a mean and median of 0.5 and 0.5, respectively. For the larger sample of 500 periods, even though the series is non-stationary, the coefficient estimates of β_1 are close to the theoretical value of 1 and thus there is no bias.

The second fact that arises when estimating AR by OLS when variables are non-stationary is that the t statistic does not follow the normal distribution even when samples are large. This can be seen clearly in Figure 4.13 for the statistic of the null hypothesis that $\beta_1 = 1$ and for T=500. The histogram below shows that the distribution of t statistic is shifted to the left relative to the standard normal distribution.

The third problem with non-stationary variables occurs when the interest is to model is the relationship between X and Y and both variables are non-stationary. This could lead to spurious results of statistical evidence of a relationship between the two series when indeed they are independent of each other. An intuitive explanation for this result can be provided when considering, e.g., two independent random walk with drift: estimating a LRM finds co-movement between the series due to the existence of a trend in both variables that makes the series move in the same or opposite direction. The simulation below shows more intuitively this results for two independent processes X and Y with the same drift parameter μ, but independent of each other (i.e., Y is not a function of X). The histogram of the t statistics for the significance of β_1 in $Y_t = \beta_0 + \beta_1 X_t + \epsilon_t$ is shown in the left plot of Figure 4.14, while the R^2 of the regression is shown on the right. The distribution of the t test statistic has a significant positive mean and would lead to an extremely large number of rejections of the hypothesis that $\beta_1 = 0$, when its true value is equal to zero. In addition, the distribution of the R^2 shows that in the vast majority of these 1000 simulations we would find a moderate to large goodness-of-fit measure which would suggest a significant relationship between the two variables, although the truth is that there is no relationship.

```
T     = 500          # length of the time series
B     = 1000          # number of simulation
mu    = 0.1          # value of the drift term
```

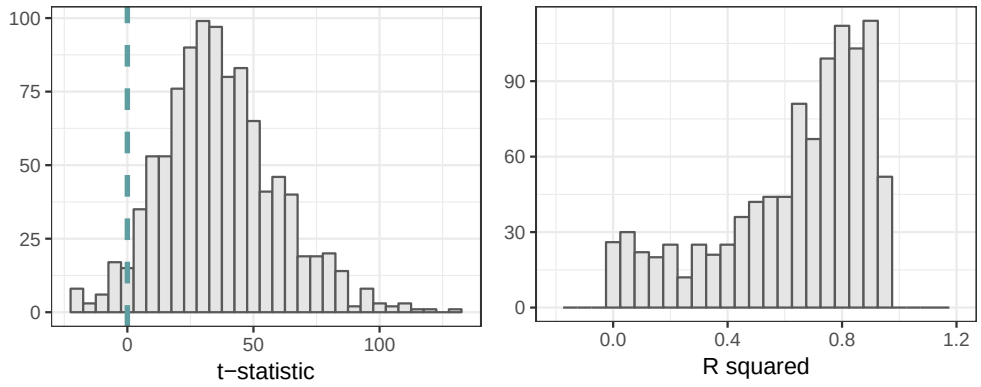


Figure 4.14: Simulated distribution of the t-statistic and R square for the regression of two independent non-stationary variables. Sample size is 500.

```
sigma = 1           # standard deviation of the error
tstat  = numeric(B) # object to store the DF test stat
R2     = numeric(B)
for (b in 1:B)
{
  Y       <- as.xts(arima.sim(n=T, list(order=c(0,1,0)), mean=mu, sd=sigma))   # simulates the Y series
  X       <- as.xts(arima.sim(n=T, list(order=c(0,1,0)), mean=mu, sd=sigma))   # simulates the X series

  fit      <- lm(Y ~ X)               # OLS estimation of DF regression
  fit.tab  <- summary(fit)$coefficients
  tstat[b] <- fit.tab[2,3]  # stores the results
  R2[b]    <- summary(fit)$r.square
}

p1 <- ggplot() + geom_histogram(aes(tstat),color="grey35", fill="grey90", binwidth=5) +
  theme_bw() + labs(x="t-statistic",y=NULL) +
  geom_vline(xintercept = 0, color="cadetblue", size=1.2, linetype="dashed")
p2 <- ggplot() + geom_histogram(aes(R2),color="grey35", fill="grey90", binwidth=0.05) +
  xlim(c(-0.2, 1.2)) + theme_bw() + labs(x="R squared",y=NULL)
grid.arrange(p1, p2, ncol=2)
```

4.5.4 Testing for non-stationarity

The first task when analyzing economic and financial time series is to evaluate if the variable can be considered stationary or not, since non-stationarity leads to several inferential problems that can be problematic for our analysis. Once we conclude that a time series is non-stationary we need to decide whether it is consistent with a trend-stationary model or rather with a random walk model (with or without drift). So, what we are really asking are two questions: is the time series stationary or non-stationary? and if it non-stationary, which of the two models discussed earlier is more likely to be consistent with the data?

The approach that we follow is to combine the trend-stationary and random walk models and test hypothesis about the coefficients of the combined model that provide answers to the previous questions. The trend-stationary model with an AR(1) process for the deviation is defined as

$$\begin{aligned} Y_t = &\quad \beta_0 + \beta_1 t + d_t \\ d_t = &\quad \phi d_{t-1} + \epsilon_t \end{aligned}$$

from the first Equation we obtain that $d_{t-1} = Y_{t-1} - \beta_0 - \beta_1(t-1)$ so that we can replace d_t in the first Equation and formulate the model in terms of t and Y_{t-1}, that is:

$$\begin{aligned} Y_t = &\quad \beta_0 + \beta_1 t + \phi d_{t-1} + \epsilon_t \\ = &\quad \beta_0 + \beta_1 t + \phi \left[Y_{t-1} - \beta_0 - \beta_1 (t-1) \right] + \epsilon_t \\ = &\quad \beta_0 (1-\phi) + \phi\beta_1 + \beta_1 (1-\phi) t + \phi Y_{t-1} + \epsilon_t \end{aligned}$$

if we set $\gamma_0 = \beta_0(1-\phi) + \phi\beta_1$ and $\gamma_1 = \beta_1(1-\phi)$ then the model becomes

$$Y_t = \gamma_0 + \gamma_1 t + \phi Y_{t-1} + \epsilon_t$$

and the null hypothesis that we want to test is that $\phi = 1$ against the alternative that $\phi < 1$. Notice that if $\phi_1 = 1$ then $\gamma_1 = 0$ and the model becomes $Y_t = \beta_1 + Y_{t-1} + \epsilon_t$, which is the random walk with drift model. Testing the hypothesis can lead to the two following outcomes:

- Failing to reject the null hypothesis that $\phi = 1$ (and $\gamma_1 = 0$) indicates that Y_t follows the random walk with drift model
- Rejecting the null hypothesis in favor of the alternative that $\phi < 1$ supports the trend-stationary model

If the time series Y_t does not show a clear pattern of growing over time, we can assume that $\beta_1 = 0$ which simplifies the model to $Y_t = \gamma_0 + \phi Y_{t-1} + \epsilon_t$ with $\gamma_0 = \beta_0(1-\phi)$. Testing the hypothesis that $\phi = 1$ in this case provides the following results:

- Failing to reject H_0 implies that the model is $Y_t = Y_{t-1} + \epsilon_t$ that represents the random walk without drift model
- Rejecting the null hypothesis suggests that Y_t should be modeled as a stationary AR process

The important choice to make is whether to include a trend (i.e., β_1) or not in the testing Equation. While for the real GDP and the S&P 500 in Figure 4.7 it is clear that a trend is necessary, in other situations (e.g., the unemployment rate or interest rates) a trend might not be warranted. To test $H_0 : \phi = 1$ (non-stationarity) against the alternative $H_1 : \phi < 1$ (stationarity) the model is reformulated by subtracting Y_{t-1} to both the left and right hand-side of the Equation:

$$\begin{aligned} Y_t - Y_{t-1} = &\quad \gamma_0 + \gamma_1 t + \phi Y_{t-1} - Y_{t-1} + \epsilon_t \\ \Delta Y_t = &\quad \gamma_0 + \gamma_1 t + \delta Y_{t-1} + \epsilon_t \end{aligned}$$

with $\delta = \phi - 1$. Testing the null that $\phi = 1$ is equivalent to testing $\delta = 0$ in this model against the alternative that $\delta < 0$. This model is estimated by OLS and the test statistic for what is called the **Dickey-Fuller** (**DF**) test is given by the t-statistic of $\hat{\delta}$, that is,

$$DF = \frac{\hat{\delta}}{\hat{\sigma}_\delta}$$

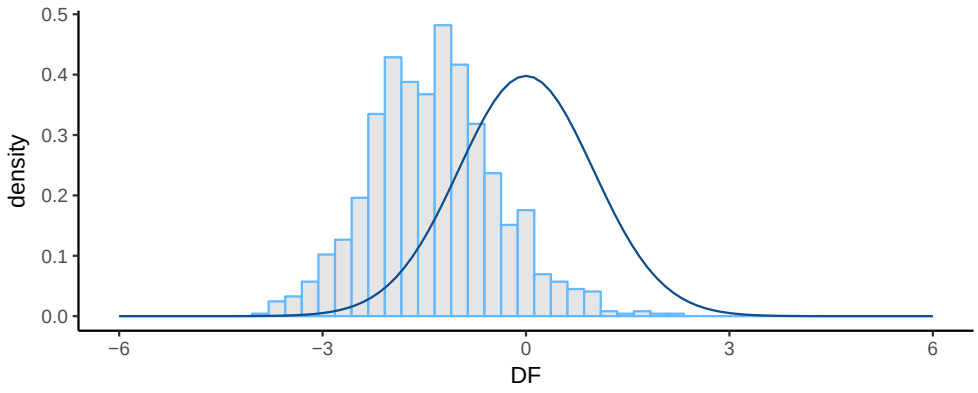


Figure 4.15: Simulated distribution of the DF test statistic and the t distribution. Sample size 100.

where $\hat{\sigma}_\delta$ represents the standard error of $\hat{\delta}$ assuming homoskedasticity. Even though subtracting Y_{t-1} makes the LHS of the Equation stationary, under the null hypothesis the regressor Y_{t-1} is non-stationary and leads to the distribution of the DF not being t distributed. Instead, it follows a different distribution with critical values that are tabulated and are provided below.

The non-standard distribution of the DF test statistic can be investigated via a simulation study using R. The code below performs a simulation in which:

- random walk time series (with drift) are generated
- the DF regression equation is estimated
- the t-statistic of Y_{t-1} is stored

These steps are repeated B times and Figure 4.15 shows the histogram of the DF statistic together with the Student t distribution with `T-1` degree-of-freedom (`T` represents the length of the time series set in the code).

```
T     = 100          # length of the time series
B     = 1000          # number of simulation
mu    = 0.1          # value of the drift term
sigma = 1            # standard deviation of the error

DF = numeric(B) # object to store the DF test stat
for (b in 1:B)
{
  Y     <- as.xts(arima.sim(n=T, list(order=c(0,1,0)), mean=mu, sd=sigma))
  fit   <- dyn$lm(diff(Y) ~ lag(Y,1))               # OLS estimation of DF regression
  DF[b] <-  summary(fit)$coef[2,3]                   # stores the results
}
# plotting
ggplot(data=data.frame(DF), aes(x=DF)) +
  geom_histogram(aes(y=..density..), bins=50, fill="grey90", color="steelblue1") +
  stat_function(fun = dt, colour = "dodgerblue4", args = list(df = (T-1))) +
  theme_classic()  + xlim(-6,6)
```

The graph shows clearly that the distribution of the DF statistic does not follow the t distribution: it has

a negative mean and median (instead of 0), it is slightly skewed to the right (positive skewness) rather than being symmetric, and its empirical 5% quantile is -2.851 instead of the theoretical value of -1.66. Since we are performing a one-sided test against the alternative hypothesis $H_1 : \delta < 0$, using the one-sided 5% critical value from the t distribution would lead to reject the null hypothesis of non-stationarity too often relative to the appropriate critical values of the DF statistic. For the simulation exercise above, the (asymptotic) critical value for the null of a random walk with drift is -2.86 at 5% significance level. The percentage of simulations for which we reject the null based on this critical value and that from the t distribution are

```
c(DF = sum(DF < -2.86) / B, T = sum(DF < -1.67) / B)
```

```
   DF     T
0.049 0.390
```

This shows that using the critical value from the t-distribution would lead to reject too often (39% of the times) relative to the expected level of 5%. Instead, using the correct critical value the null is rejected 4.9% which is quite close to the 5% significance level.

In practice, it is advisable to include lags of ΔY_t to control for serial correlation in the time series Y_t and its changes. This is called the **Augmented Dickey Fuller** (**ADF**) and requires to estimate the following regression model (for the case with a constant):

$$\Delta Y_t = \gamma_0 + \gamma_1 t + \delta Y_{t-1} + \sum_{j=1}^{p} \omega_j \Delta Y_{t-j} + \epsilon_t$$

which is simply adding lags of ΔY_t on the RHS of the previous Equation. The ADF test statistic is calculated as before by taking the t-statistic of the coefficient estimate of Y_{t-1}, that is, $ADF = \hat{\delta}/\hat{\sigma}_\delta$. The decision to reject the null hypothesis that $\delta = 0$ against the alternative that $\delta < 0$ relies on comparing the DF or ADF test statistic with the appropriate critical values. As discussed earlier, the DF statistic has a special distribution and critical values have been tabulated for the case with/without a constant and with/without a trend and for various sample sizes. Below you can find the critical values obtained for various sample sizes for a model with constant and with or without trend and the null is rejected if the test statistic is smaller relative to the critical value:

Table 4.3: Critical values for Augmented Dickey-Fuller (ADF) test with and without a trend

Sample Size	Without	Trend	With	Trend
	1%	5%	1%	5%
T = 25	-3.75	-3.00	-4.38	-3.60
T = 50	-3.58	-2.93	-4.15	-3.50
T = 100	-3.51	-2.89	-4.04	-3.45
T = 250	-3.46	-2.88	-3.99	-3.43
T = 500	-3.44	-2.87	-3.98	-3.42
T = ∞	-3.43	-2.86	-3.96	-3.41

There are several packages that provide functions that perform the ADF test on a time series. The

package `tseries` provides the `adf.test()` function that assumes a model with constant and trend and the user is only required to specify the time series and the lags of ΔY_t. The application to the logarithm of real GDP discussed above gives the following results[10]:

```
library(tseries)
adf.test(log(macrodata.q$GDP), k=4)
```

```
    Augmented Dickey-Fuller Test

data:  log(macrodata.q$GDP)
Dickey-Fuller = -1.039, Lag order = 4, p-value = 0.931
alternative hypothesis: stationary
```

The ADF test statistic is equal to -1.039 with a p-value of 0.931 which is larger than 5% and we thus conclude that we do not reject the hypothesis that $\delta = 0$ or, equilevantly, $\phi = 1$. Hence, the logarithm of GDP is not stationary even when including a trend in the analysis and a random walk with drift is the most appropriate modeling choice. The code below shows the application of the ADF test to the logarithm of the S&P 500 and the returns at the monthly and daily frequency:

```
adf.sp <- rbind(`Price monthly`  = adf.test(log(spm$price))[c("statistic", "p.value")],
                `Return monthly` = adf.test(spm$return)[c("statistic", "p.value")],
                `Price daily`    = adf.test(log(spd$price))[c("statistic", "p.value")],
                `Price daily`    = adf.test(spd$return)[c("statistic", "p.value")])
```

	statistic	p.value
Price monthly	-3.11727	0.112109
Return monthly	-4.75214	0.01
Price daily	-2.25721	0.469427
Price daily	-14.4059	0.01

The results show that at the 5% level the null of non-stationarity is not rejected for the monthly and daily price, but they are strongly rejected for the returns. This is expected since the returns are the growth rate of the price variable and they exhibit very small auto-correlation and a constant long-run mean. A problem with the `adf.test()` function is that it does not allow to change the type of test, in case we are interested to run the test without the trend variable. In this case we can use the `ur.df()` function from the `urca` package. Below is an application to the logarithm of the real GDP:

```
library(urca)
adf <- ur.df(log(macrodata.q$GDP), type="trend", lags=4) # type: "none", "drift", "trend"
summary(adf)
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend
```

[10] If the lag is not specified the function default is to use $k = trunc((length(x) - 1)^(1/3))$, where x denotes the time series object.

```
Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-0.03076 -0.00427  0.00045  0.00483  0.03373

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0779720  0.0682260    1.14     0.25
z.lag.1     -0.0091399  0.0087954   -1.04     0.30
tt           0.0000583  0.0000695    0.84     0.40
z.diff.lag1  0.3246143  0.0622132    5.22  3.7e-07 ***
z.diff.lag2  0.0992773  0.0653046    1.52     0.13
z.diff.lag3 -0.0423973  0.0645861   -0.66     0.51
z.diff.lag4 -0.0502209  0.0612288   -0.82     0.41
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.00832 on 258 degrees of freedom
Multiple R-squared:  0.155, Adjusted R-squared:  0.136
F-statistic: 7.91 on 6 and 258 DF,  p-value: 7.72e-08


Value of test-statistic is: -1.0392 13.0625 2.433

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -3.98 -3.42 -3.13
phi2  6.15  4.71  4.05
phi3  8.34  6.30  5.36
```

In this case the ADF test statistic is -1.039 which should be compared to the critical value at 5% -3.42 and we fail to reject the null hypothesis that the series is non-stationary so that we concluded that it follows a random walk with drift model.

4.5.5 What to do when a time series is non-stationary

If the ADF test leads to the conclusion that a series is non-stationary then we need to understand if the nature of the non-stationarity is a trend-stationary model or a random walk model (with or without a drift). In the first case, the non-stationarity is solved by *de-trending* the series, which means that the series is regressed on a time trend and the residuals of the regression are then interpreted as the deviation from the long-run trend and modeled using time series techniques (e.g., AR). This approach is often used for the logarithm of GDP and the deviation is interpreted as the *output gap*. However, in our results the log of GDP is non-stationary even when including a trend which indicates that the deviation might also be non-stationary. On the other hand, in case we find that the series follows a random walk model then the solution to the non-stationarity is *differencing* the series and modeling $\Delta Y_t = Y_t - Y_{t-1}$. The reason for this is that differencing the series removes the trend and this can be seen in the random walk with drift model $Y_t = \mu + Y_{t-1} + \epsilon_t$ where by taking Y_{t-1} to the left-hand side results in $\Delta Y_t = \mu + \epsilon_t$. When we consider returns of an asset prices we are differencing the price which, from a statistical perspective,

is justified in order to induce stationarity in the price series.

4.6 Structural breaks in time series

Another issue that we need to keep in mind when analyzing time series is that there could have been a shift (or structural change) in its mean or its persistence over time. The economy is an evolving system changing over time which makes the past less relevant, to some extent, to understand the future. It is important to be able to recognize structural change because it can be easily mistaken for non-stationarity when indeed it is not.

Let's assume that at time t^* the intercept of a time series shifts so that we can write the AR(1) model as

$$Y_t = \beta_{00} + \beta_{01} * I(t > t^*) + \beta_1 * Y_{t-1} + \epsilon_t$$

where β_{00} represents the intercept of the AR(1) model until time t^*, $\beta_{00} + \beta_{01}$ after time t^*, and $I(A)$ denotes the indicator function that takes value 1 if A is true and 0 otherwise. In the above example we assume that the dependence of the time series, measured by β_1, is constant, although it might as well change over time as we discuss later.

Let's simulate a time series from this model with a shift in the intercept at observation 200 on a total sample of length 400. To show the effect of an intercept shift we simulate a time series with a shift of the intercept from β_{00}=0 to $\beta_{0,1}$=1, while the AR(1) parameters is constant and equal to 0.8. The resulting time series is shown in Figure 4.16, with the horizontal line representing the unconditional means before/after the break at the values $\beta_{00}/(1-\beta_1)$ and $\beta_{01}/(1-\beta_1)$, respectively. For this choice of parameters it is clear that there was a shift at observation 200 when the unconditional mean of the time series shifts from 0 to 5.

```
set.seed(1234)
T         = 400;   Tbreak    <- round(T/2)
beta00    = 0;     beta01    = 1
beta1     = 0.8;   sigma.eps = 0.8
eps       = sigma.eps * rnorm(T)
Y         = as.xts(ts(numeric(T)))
for (t in 2:T) Y[t] = beta00 + beta01 * (t > Tbreak) + beta1 * Y[t-1] + eps[t]
```

A time series with structural change can be easily mistaken for non-stationarity. As it is clear from Figure 4.17, from the perspective of a linear trend model the simulate time series seems a good candidate for such non-stationary behavior. Obviously, the trend model is misspecified and particularly irrelevant if the interest is to build a model for forecasting since the trend model predicts that Y should increase by 0.02104 units per time period. However, there is no underlying drift in the model and in the future the series will keep oscillating around its mean of 5 if not other break occurs.

4.6.1 Break at a know date

If the visual analysis of the time series shows the possibility of a break we can use that information and incorporate it in the model. An example is the break in the simulated series in Figure 4.16 that shows

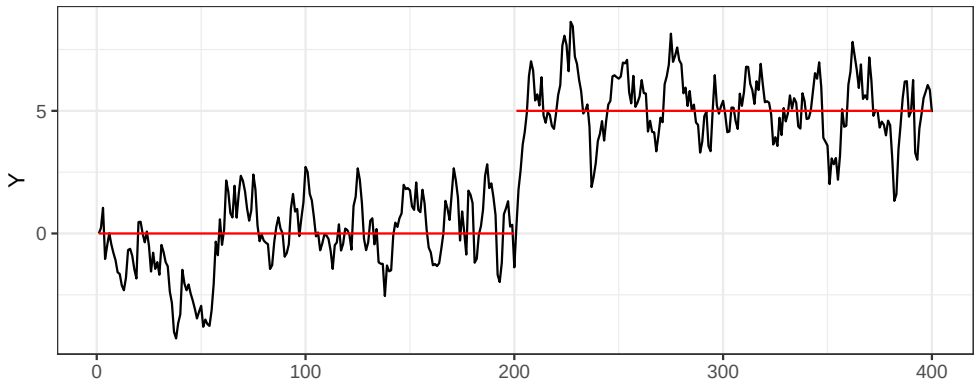


Figure 4.16: Simulated AR(1) time series with a structural break in the intercept.

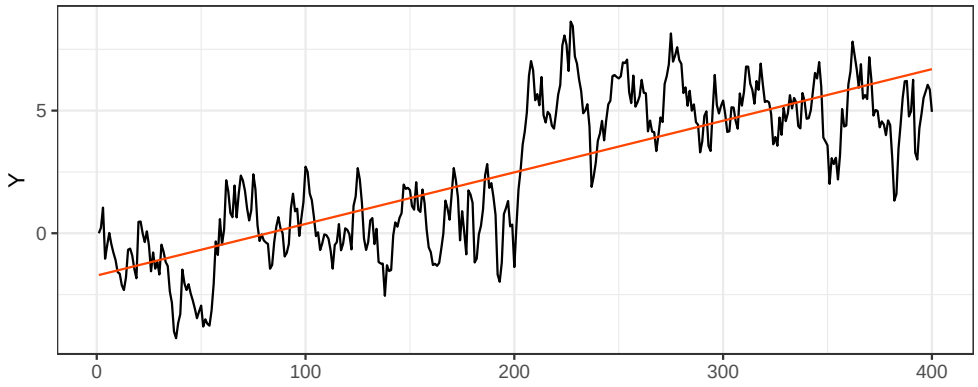


Figure 4.17: A linear trend model estimated on the simulated AR(1) time series with a structural break in the intercept.

a level shift half-way through the sample. We can estimate an AR(1) model with an intercept shift at observation 200 as follows:

```
fit    <- dyn$lm(Y ~  I(trend > Tbreak) + lag(Y, 1), data=merge(Y,trend))
```

```
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            -0.039      0.058  -0.681    0.496
I(trend > Tbreak)TRUE   1.030      0.166   6.220    0.000
lag(Y, 1)               0.813      0.027  29.928    0.000
```

The regression results show that the OLS estimate of $\hat{\beta}_{00}$ is equal to -0.039 which is not statistically different from the true value 0, while $\hat{\beta}_{0,1} = 1.03$ that is also close to its true value of 1. Hence, by incorporating in the model the occurrence of a break we are able to recover parameter estimates that are close to the true value. In addition, if we did include a time trend in this regression it would turn out to be insignificant at standard levels as shown in the code below:

```
fit <- dyn$lm(Y ~ I(trend > Tbreak) + lag(Y, 1) + trend, data=merge(Y,trend))
```

```
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            -0.068      0.093  -0.730    0.466
I(trend > Tbreak)TRUE   0.984      0.203   4.847    0.000
lag(Y, 1)               0.811      0.028  29.353    0.000
trend                   0.000      0.001   0.393    0.694
```

4.6.2 Break at an unknown date

In most cases, we might suspect that there was a break in the series but a simple visual analysis might not be enough to decide when it happened and if it was a break in the intercept, slope, or both. In this case, an approach is to scan through all possible break dates and compare the goodness of fit of the model with and without break. We then perform and F test of the null hypothesis that the model parameters did not change. If we fix the AR order to equal $p = 1$, the approach is as follows:

1. Estimate the AR(p) model on the full sample:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t$$

   for $t = 1, \cdots, T$ and denote by RSS_R its Residual Sum of Squares
2. For each t^* between t_{min} and t_{max} (e.g., $t_{min} = 0.15 * T$ and $t_{max} = 0.85 * T$), we estimate the model:

$$Y_t = \beta_{0,0} * I(t \leq t^*) + \beta_{01} * I(t > t^*) + \beta_{1,0} * I(t \leq t^*) * Y_{t-1} + \beta_{1,1} * I(t > t^*) * Y_{t-1} + u_t$$

   where RSS_u represents the Residual Sum of Squares of the model in 2 (unrestricted) and RSS_ϵ for the model in 1 (restricted)
3. Calculate the F statistic for the null joint hypothesis that $\beta_{00} = \beta_{01}$ and $\beta_{10} = \beta_{11}$ at each τ which is given by

$$F(\tau) = \frac{RSS_u - RSS_R}{RSS_u/(T - 2 * (p + 1))}$$

4. The $supF = \sup_\tau F(\tau)$ test statistic is the largest value of $F(\tau)$

While the F statistic follows the F distribution with $p+1$ degrees-of-freedom in the numerator and $T-2*(p+1)$ in the denominator, the $supF$ statistic has a non-standard distribution with the critical values provided in the Table below.

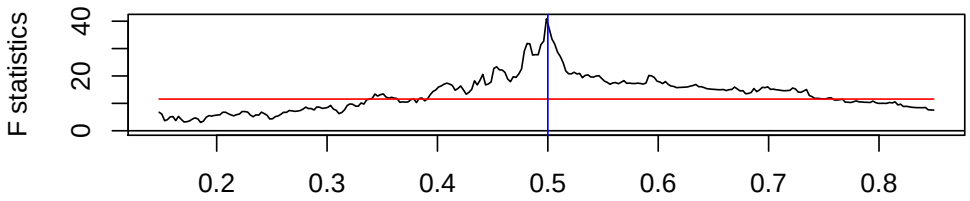


Figure 4.18: F-statistic for structural change at an unkown date.

Table 4.4: Critical values for supF test with 15% trimming on each side

p	10%	5%	1%
1	7.12	8.68	12.16
2	5.00	5.86	7.78
3	4.09	4.71	6.02
4	3.59	4.09	5.12
5	3.26	3.66	4.53
6	3.02	3.37	4.12
7	2.84	3.15	3.82
8	2.69	2.98	3.57
9	2.58	2.84	3.38
10	2.48	2.71	3.23

The test for structural change can be easily implemented using the `Fstats` function in the `strucchange` package. The application to the simulated series is as follows:

```
library(strucchange)
fit <- dyn$lm(Y ~ lag(Y, 1))
test <- Fstats(fit, from=0.15, to=0.85, data=as.ts(Y))
plot(test, xlab="")
abline(v=0.5, col="blue", ltype=2)
```

The time series plot of $F(\tau)$ shows a clear peak which corresponds to 0.5 times the sample size that indeed corresponds to the break date. Once we find the break date, we can proceed as discussed earlier for the case of a known break dates, that is, estimate the model separately on the sub-periods. In practice, it is not necessarily the case that a time series experiences only one break but there could be several. In case there are two intercept shifts corresponding to one third (intercept shift from 0 to 1) and two third (from 1 to 2) of the sample, the time series and the $F(\tau)$ test are as follows.

The $F(\tau)$ plot shows clearly two peaks that are statistically significant and are very close to the break dates set in simulating the data. So far we only discussed shift of the intercept, but there could be as well changes in the persistence of the time series. Below we show a simulated time series from the following model:

$$Y_t = \beta_0 + \beta_{10} Y_{t-1} I(t \leq t^*) + \beta_{11} Y_{t-1} I(t > t^*) + \epsilon_t$$

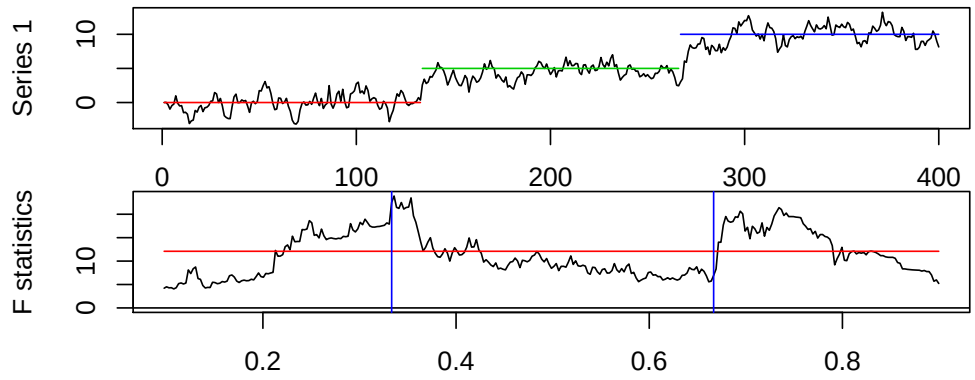


Figure 4.19: Simulated time series with three breaks in the mean and the F-statistic for the test of structural change at an unkown date.

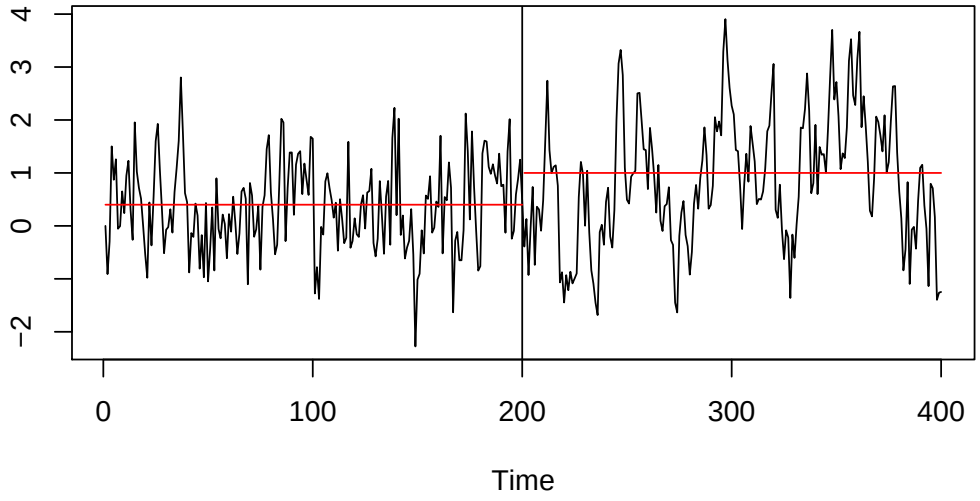


Figure 4.20: Simulated AR(1) time series with three structural breaks in the mean.

where t^* denotes the break date. In the simulation we set the parameters to β_0=0.2, β_{10}=0.5, β_{11}=0.8, and the break date t^* is set equal to 200.

The $F(\tau)$ statistic for this time series is shown below. Also in this case the maximum value of $F(\tau)$ occurs close to 0.5 that represents halfway through the sample. Once the break date has been identified we can then proceed by estimating the model that accounts for the change in parameters and use this model for further analysis or forecasting.

4.7 Forecasting Revenue and Earnings-Per-Share (EPS)

`... this section is still work in progress ...`

An interesting application of time series methods is forecasting revenue and EPS of listed companies. Analysts at investment banks provide their forecast about the future prospects of companies which are then used by their clients as inputs in the investment process. Although analysts use a much wider

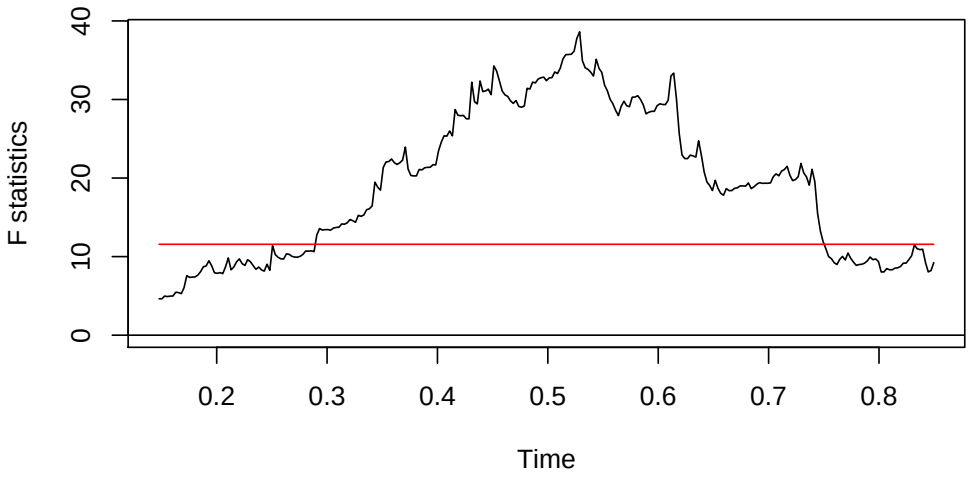


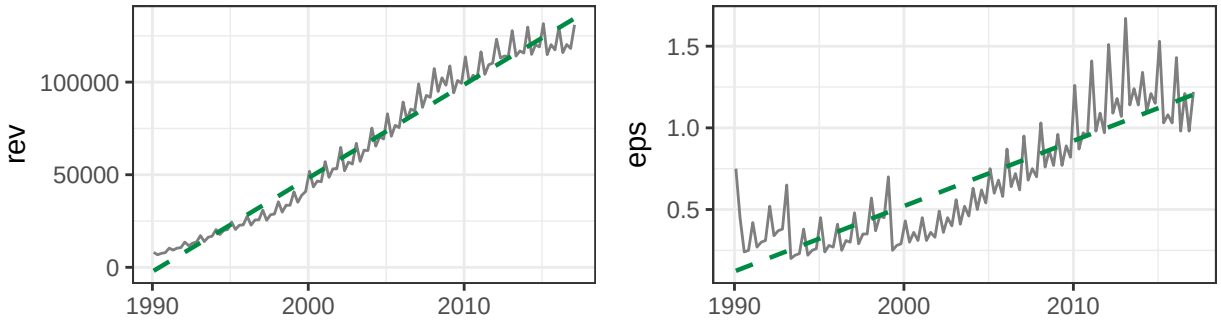
Figure 4.21: F statistic for the structural break test.

information set than just the history of revenue and EPS of a company, it is still interesting to calculate a purely time series forecast of these variables as a benchmark forecast.

Below we show the revenue and EPS time series for Walmart (ticker: WMT) from 1990 Q1 until the `as.yearqtr(max(data$datadate))`. The dashed line represents a linear deterministic trend estimated over the full sample. These series share some common features:

1. they grow over time, although a linear trend over the full sample might not be the best model to explain the series
2. an evident seasonal component in both series
3. possible structural breaks in the level and/or dynamics of the series

For the EPS series it is evidence that there were two breaks in the 1990s in which the variable dropped and continued growing afterwards. On the other hand, the Revenue series seems to have a slightly exponential behavior which could be taken into account by modeling the logarithm of the variable. These series seem good to apply the test for non-stationarity and structural break that we discussed earlier. We will start by modeling the time series of Revenue.



4.7.1 Modeling Revenue

Let's first test for non-stationarity of WMT revenue. Unfortunately, we cannot use the `ur.df()` function from the `urca` package since there is a clear seasonal pattern in the time series which cannot be added in that function. Hence, we estimate the DF regression as follows:

```
## dyn$lm(drev ~ lag(rev, 1) + lag(drev, 1:4) + trend + Q2 + Q3 + Q4, data=mydata)
## dyn$lm(drev ~ lag(rev, 1) + lag(drev, 1:4) + trend + trendsq + Q2 + Q3 + Q4, data=mydata)
adf.tab
```

```
        Rev + trend     Rev + quad trend      Log Rev + trend Log Rev + quad trend
             -1.250               -0.752               -1.349               -0.752
```

where `drev` denotes the first difference of the revenue, `trend` is the trend variable and `trendsq` its square, and `Q2`, `Q3`, and `Q4` represent the quarterly dummy variables that take value 1 in quarter 2, 3, and 4 and are zero otherwise. The ADF test statistic is the t-statistic of the lagged value of the revenue or the log revenue and the critical value at 5% when $T = 93$ is -3.45. The ADF test statistic is -1.25 for the linear trend and -0.752 for the quadratic trend. In both cases we do not reject the null hypothesis which suggests that the most appropriate model is a random walk with drift rather than a linear or quadratic trend model with seasonal dummy variables. When considering the logarithm of revenue, the ADF statistic is -1.349 and -4.542, respectively, so that for the quadratic trend model we reject the null hypothesis and thus conclude it is a valid model for log revenue since the deviations from the trend are stationary. Hence, in this case we find that the modeling implications of the test for non-stationarity rely on the decision to take the logarithm of the variable or not. For the purpose of illustrating the modeling process and the decisions involved, we continue by discussing the specification of the linear and quadratic trend models and later discuss the modeling of the changes of revenue and log revenue.

The first issue that we need to resolve is whether we should use Revenue in million of $ or transform the variable to logarithm. Below is a graph of the two series together with a deterministic trend. In the graph below the time series of WMT revenue is shown in million of $ or logarithm and for linear or quadratic trend.

The graph for Revenue shows that the series seems to be increasingly volatile over time, in the sense that revenue are more (less) variable when their level is high (low). This type of heteroskedasticity can be solved by making the changes in revenue a percentage of the level, which is achieved by taking the logarithm of the variable. The graphs show that the variability of the series appears more homogeneous over time when revenue is log-transformed. The linear trend doesn't seem useful here because the time series is off the trend line in particular at the beginning and at the end of the sample. On the other hand, the quadratic trend appears as a better fit in both cases. Overall, the quadratic model for log Revenue provides a good starting point to model the series and use it as a building block to which we sequentially add features such as seasonality and AR terms. The estimation results for the quadratic model are provided below:

```
Call:
arima(x = mydata$lrev, order = c(0, 0, 0), xreg = mydata[, c("trend", "trendsq")])

Coefficients:
      intercept  trend  trendsq
          8.898  0.054        0
```

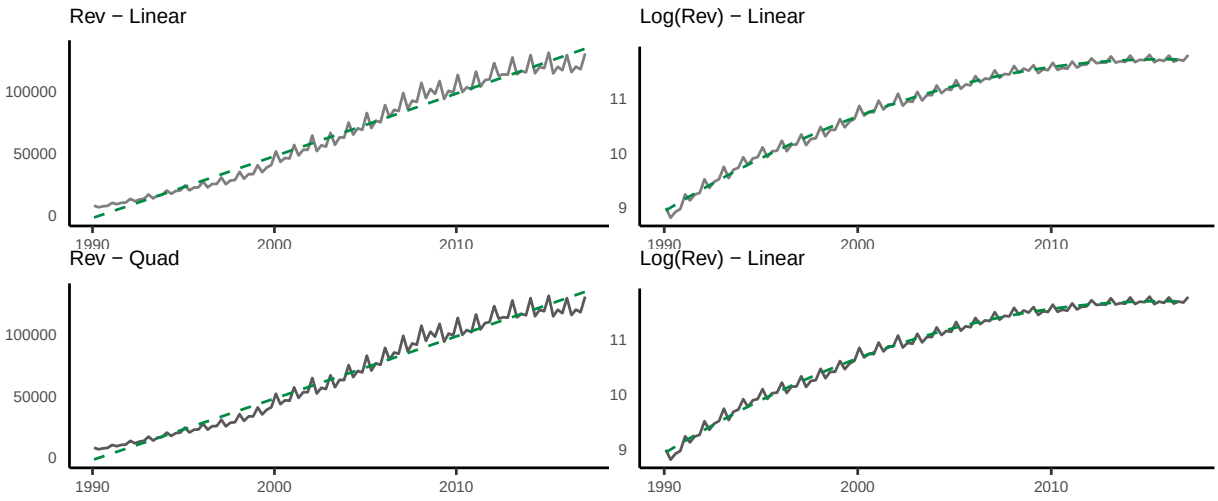


Figure 4.22: Time series plot of the revenue and logarithm of revenue together with the predicted value from the linear and quadratic trend models.

```
s.e.      0.023  0.001        0

sigma^2 estimated as 0.00598:  log likelihood = 124.36,  aic = -240.72
```

The interpretation of the results for the quadratic trend model is that revenue is expected to grow by $100 * (\beta_1 + 2 * \beta_2 * trend) * \Delta trend\%$ every quarter. In 2017 Q1 the value of the trend is 109[11] and the forecast for revenue growth in 2017 Q2 based on the coefficient estimates is 0.361%.

An additional characteristic of the time series is its seasonal pattern which is quite clear from the plots. To account for this pattern we can include quarterly dummy variables in the regression as discussed earlier. The dummy variables for the quarters can be created using the `quarter()` function from package `lubridate` which provides value 1,2,3, and 4 based on the date. To avoid the dummy variable trap we can include in the regression the intercept and drop one of the seasonal variables, for example `Q1`. The estimation results for the quadratic trend model with seasonal dummy variables are as follows:

```
Call:
arima(x = mydata$lrev, order = c(0, 0, 0), xreg = mydata[, c("Q2", "Q3", "Q4",
    "trend", "trendsq")])

Coefficients:
      intercept      Q2      Q3      Q4  trend  trendsq
          9.001  -0.174  -0.119  -0.142  0.054        0
s.e.      0.013   0.011   0.011   0.011  0.000        0

sigma^2 estimated as 0.00158:  log likelihood = 196.87,  aic = -379.74
```

where the seasonal pattern is clear: WMT's revenue are -17.4% higher in `Q2` relative to `Q1`, -11.9% higher in `Q3` relative to `Q1`, and -14.2% in `Q4`. This pattern is not surprising for a retailer since the last quarter of the year is associated with the holiday shopping. The fitted line in Figure 4.22 is very close to the realized time series and it is difficult to visually investigate the success of the model in modeling revenues.

[11] The sample period starts in 1990 Q1 and ends in 2017 Q1 for a total of 109 observations.

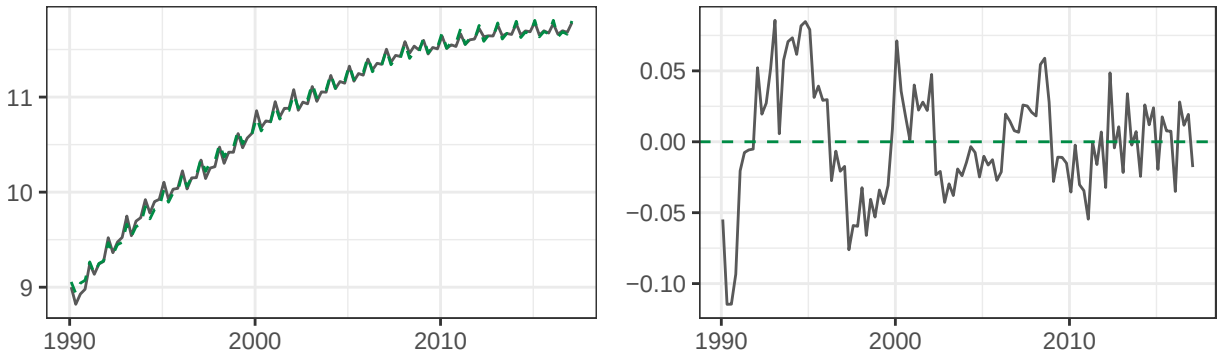
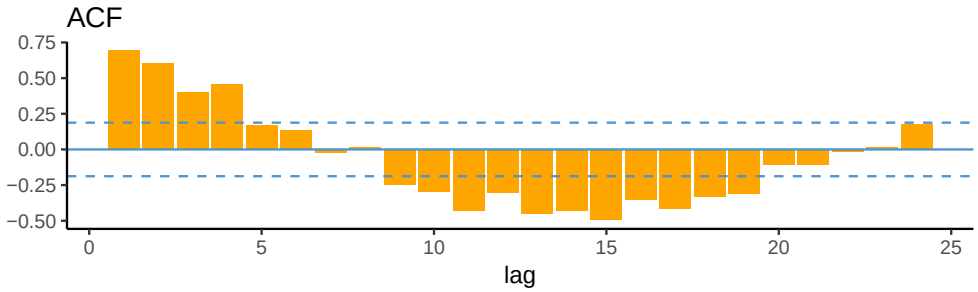


Figure 4.23: The fitted value from the quadratic trend model for the logarithm of revenue (left) and the residuals of the model (right).

However, it becomes clearer when plotting the residuals of the regression since most of the residuals are within a $\pm$ 5% range and alternating periods of positive residuals (estimate underpredicted the realization) and negative residuals (estimate overpredicted the realization). It is clear that the residuals are stationary since they mean-revert relatively quickly, and this can be better assessed by looking at the ACF of the residuals.

The ACF of the residuals shows that there is significant correlation in the residuals that starts around 0.7 at lag 1 and then at lag 5 and then again at lag 9 to 12. The existence of such dependence means that the deviations of revenue from the quadratic trend model have a systematic component which should be included in the model to improve its goodness-of-fit and its forecasting power. This could be done by adding an AR component to the model to account for these systematic deviations of revenue from the quadratic trend model with quarterly dummy variables.

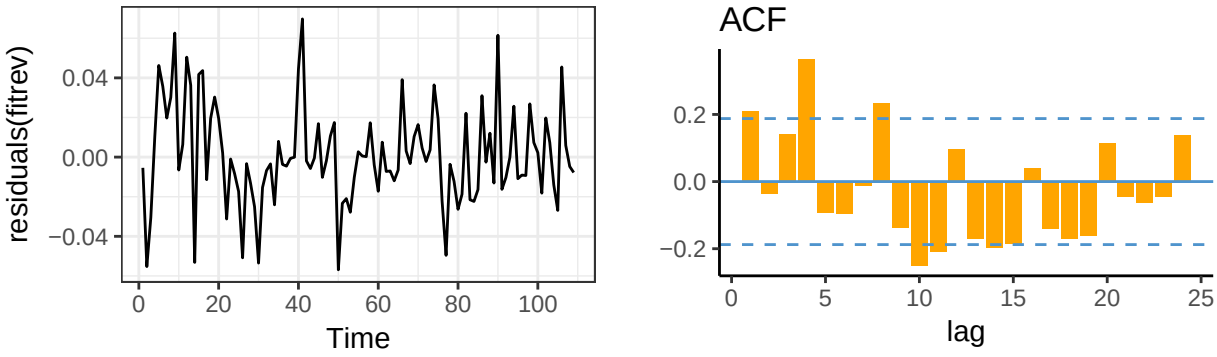


We thus add 12 lags of log(Revenue) to the regression model and find that only lags 1 to 4 are significant after we sequentially eliminate the irrelevant lags. However, adding these lags makes the `Q4` dummy variable irrelevant and we thus drop it from the regression and the final model is:

```
Call:
arima(x = mydata$lrev, order = c(4, 0, 0), xreg = mydata[, c("Q2", "Q3", "Q4",
    "trend", "trendsq")])

Coefficients:
       ar1    ar2     ar3    ar4  intercept      Q2      Q3      Q4  trend  trendsq
      0.58  0.228  -0.309  0.374      8.954  -0.175  -0.120  -0.144  0.056        0
s.e.  0.09  0.102   0.103  0.090      0.039   0.008   0.006   0.008  0.001        0
```

```
sigma^2 estimated as 0.000623:  log likelihood = 246.79,  aic = -471.57
```

The coefficient of `Q4` has significantly reduced from the earlier model without lags due to the effect of lag 4 and 5 in (partly) capturing the seasonal pattern in the data. Checking the residuals of this model shows that there are still some lags that are statistically significant at 5%, such as at the 1 year horizon and for lags over two years, but the model seems to be overall well-specified.



Based on this model, the 1 to h steps ahead forecasts based on the information available in 2017 Q1 are shown in Figure 4.24. However, we are modeling the logarithm of revenue so that the forecasted value is also in logarithm and the scale of the plot in Figure 4.24 is thus logarithmic. The intuitive approach of taking the exponential of the forecasts from the log-revenue model provides however an approximate solution, since the $exp(E(ln(X)) \neq E(X)$. In practice, the error in doing this is quite small that can be considered as a good approximation.

```
h = 8 # forecast horizon
forecast.dates   <- seq(max(time(mydata)), by = "quarter", length.out = h)
forecast.quarter <- lubridate::quarter(forecast.dates)
forecast.xreg    <- cbind(Q2 = forecast.quarter == 2,
                          Q3 = forecast.quarter == 3,
                          Q4 = forecast.quarter == 4,
                          trend   = max(mydata$trend) + 1:h,
                          trendsq = (max(mydata$trend) + 1:h)^2)
library(forecast)
p1 <- forecast(fitrev, h, xreg=forecast.xreg, level=c(50,90))
autoplot(p1, include=24, is.date=TRUE) + theme_bw() +
  scale_fill_gradientn(colours = c("plum1","plum3"), breaks=p1$level, guide="legend")
knitr::kable(exp(data.frame(p1)), caption="The point and interval forecasts from the previous model; taking the exp
```

The application of the ADF test to revenue shows that the series can be considered non-stationary, although in the case of the logarithm we reject the null hypothesis when the quadratic trend is included in the model. We thus consider also the case of taking differences of revenue and log revenue by creating the variable `drev = diff(rev)` and `dlrev = diff(log(rev))`. Below we show a time series plot of these two variables.

Table 4.5: The point and interval forecasts from the previous model; taking the exponential of the forecasts of the logarithm of the variable provides an approximate forecast of the level of the variable.

	Point.Forecast	Lo.50	Hi.50	Lo.90	Hi.90
110	132631	130417	134883	127296	138190
111	110002	107882	112164	104903	115349
112	117339	114826	119908	111303	123703
113	112513	110086	114993	106687	118658
114	129676	126744	132677	122641	137115
115	107295	104805	109844	101324	113617
116	113231	110477	116054	106632	120238
117	109228	106531	111993	102768	116094

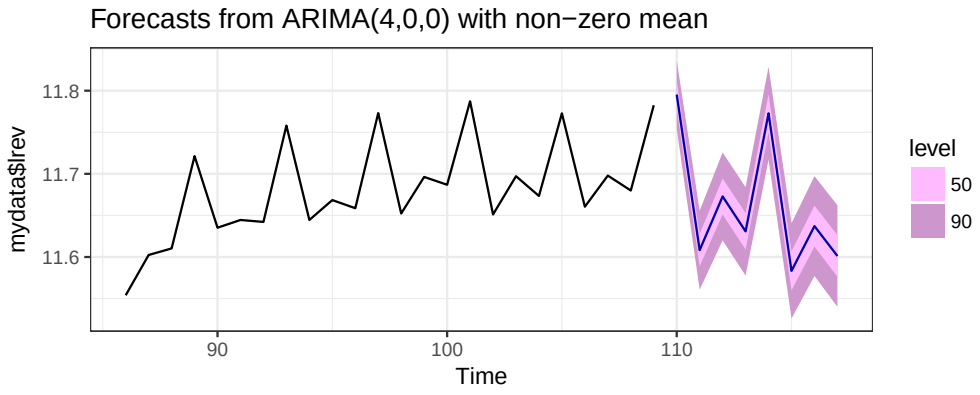
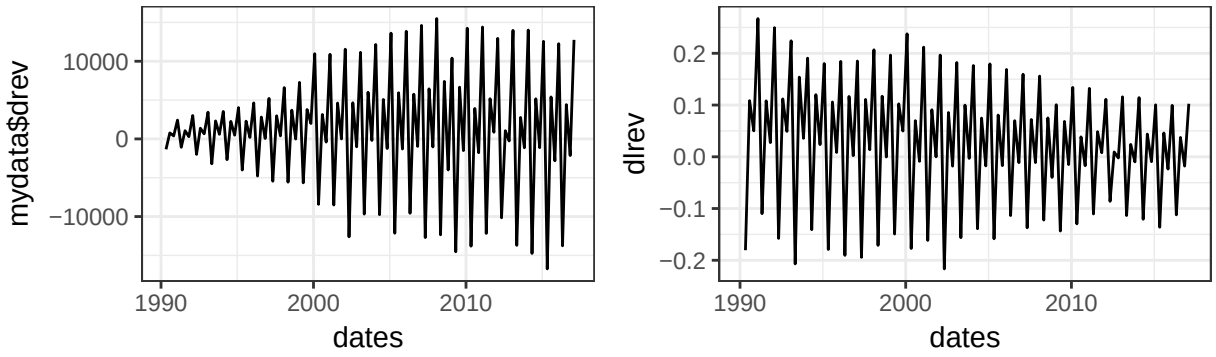


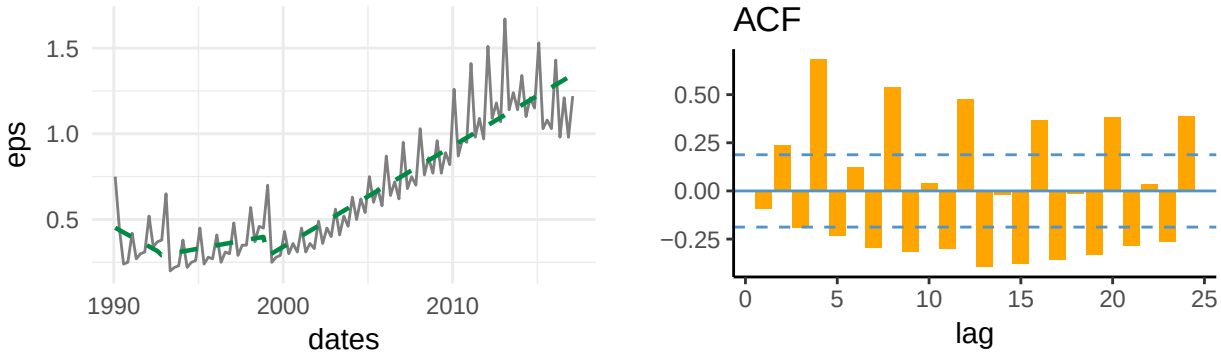
Figure 4.24: h step ahead forecasts for the logarithm of revenue based on the AR(4) model with seasonal dummies and quadratic trend.



The time series plot of `drev` on the left shows that the variable seems to have increasing dispersion as time progresses. This might be problematic in estimating the parameters of the model since the recent observations will have a larger weight in estimation relative to older observations. This problem can be easily solved, as discussed earlier, by taking the logarithm of revenue as shown on the right plot. Changes become relative to the level in the previous period and thus more homogeneous (technically, homoskedastic), although it seems that in the last years of the sample the dispersion has somewhat decreased.

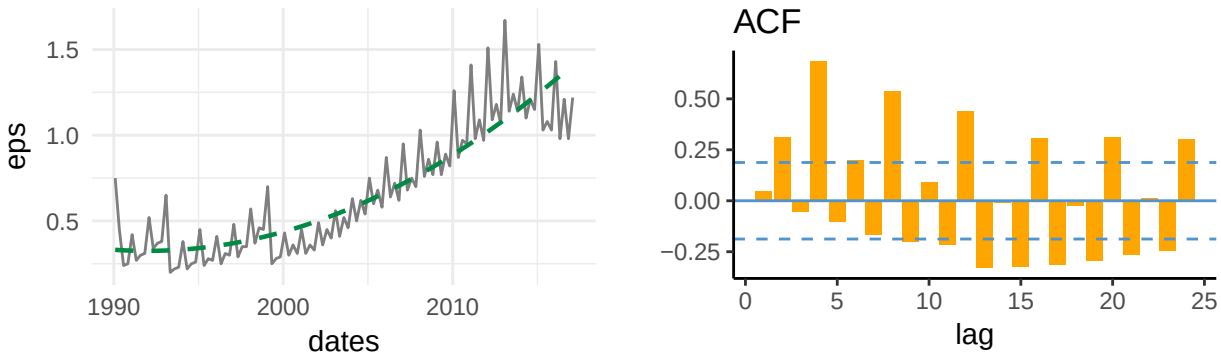
4.7.2 Earnings-Per-Share (EPS)

As we saw earlier, the EPS series is not well-explained by a linear trend, and one of the reasons is the occurence of two breaks in the 1990s. The first step in modeling the EPS series is to identify the break dates to try to account for these events. The first break occured in 1993Q1 when EPS jumped to \$0.20 from \$0.65 for 1992Q4, while the second break occured in 1999Q1 when EPS dropped to \$0.20 from \$0.70 in the previous quarter. We can account for these breaks by fitting different trend lines to the time period before 1993Q1, between 1993Q1 and 1999Q1, and after 1999Q1. This can be done by creating a constant and trend variables for each of these three subperiods. For example, the constant for the period between 1993Q1 and 1998Q1 can be created as `(time >= 1993) * (time <= 1998.75)`, where `time = time(trend)`. The trend can be similarly created by `trend * ((time >= 1993) * (time <= 1998.75))`. The fitted line for the piece-wise linear trend model is shown below, and it is clear that the model improves significantly over a linear trend model with no break (earlier graph). The graph on the right shows the ACF of the residuals that shows a clear seasonal pattern which we did not account yet in the model.

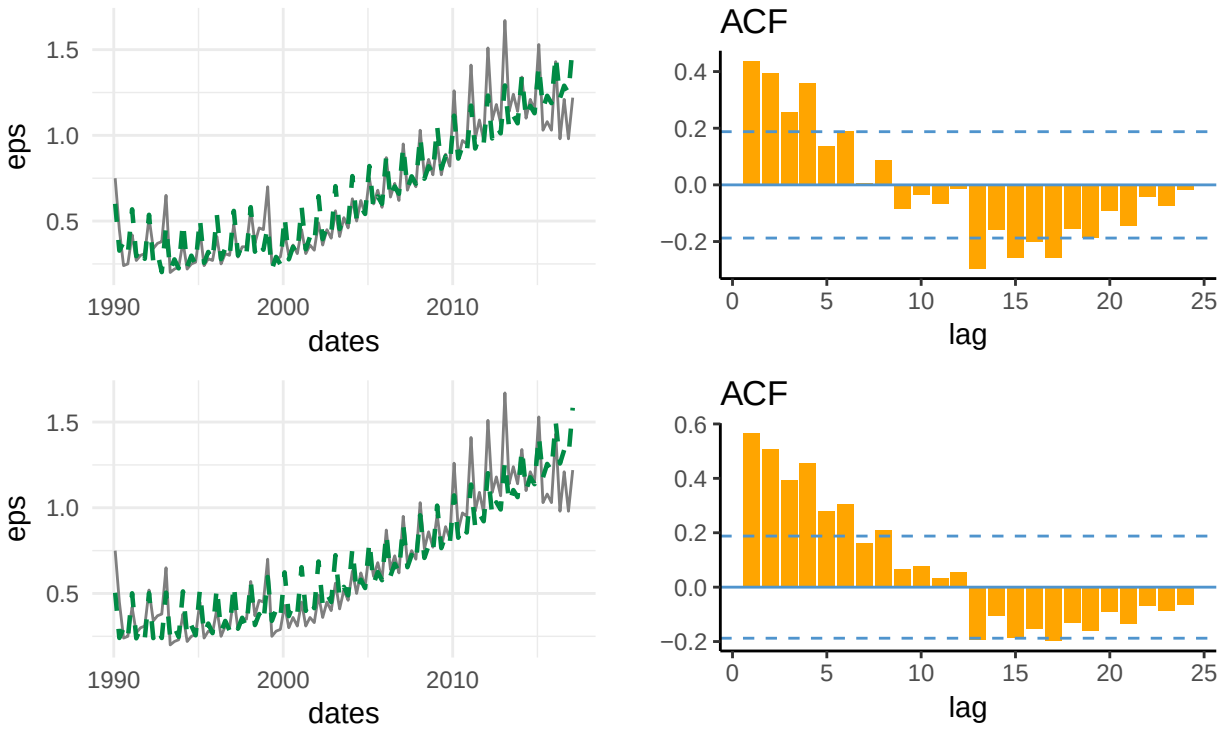


Splitting the full sample in sub-periods results in the same coefficient estimates that would be obtained by estimating the regression in each sub-period. In other words, the observations in each sub-period contribute to estimate the parameters in that period but not in others. This is important if our goal is to model or forecast the time series: in case our only goal is actually to forecast EPS, then we might as well consider EPS since the first quarter of 1999 and neglect the rest of the sample since it is uninformative about the current `regime`. However, it might also be the case that, once accounted for the shift in the trend line, we might want to estimate the seasonality and AR parameters on the full sample of observations, instead of the sub-period.

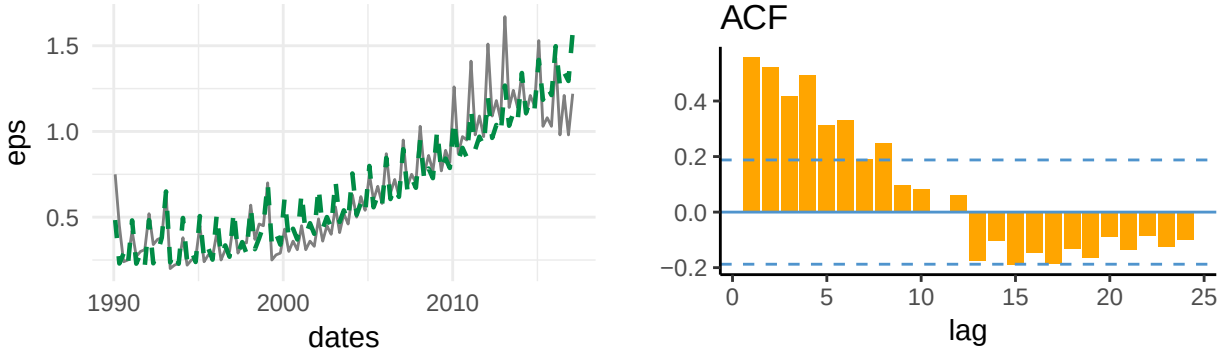
For the case of Revenue we considered a quadratic trend model and it is instructive to see how it would perform for EPS. In this case we might not consider the two jumps as breaks, but fluctuations around a quadratic trend. The fitted line shown below starts relatively flat until approximately 2000 when it starts an upward movement. In terms of goodness-of-fit, the previous model dominates the quadratic trend model even after accounting for the different number of parameters of the two models. In what follows, we examine the two models to evaluate the effect of adding features, such as seasonality and AR terms, on the model performance.



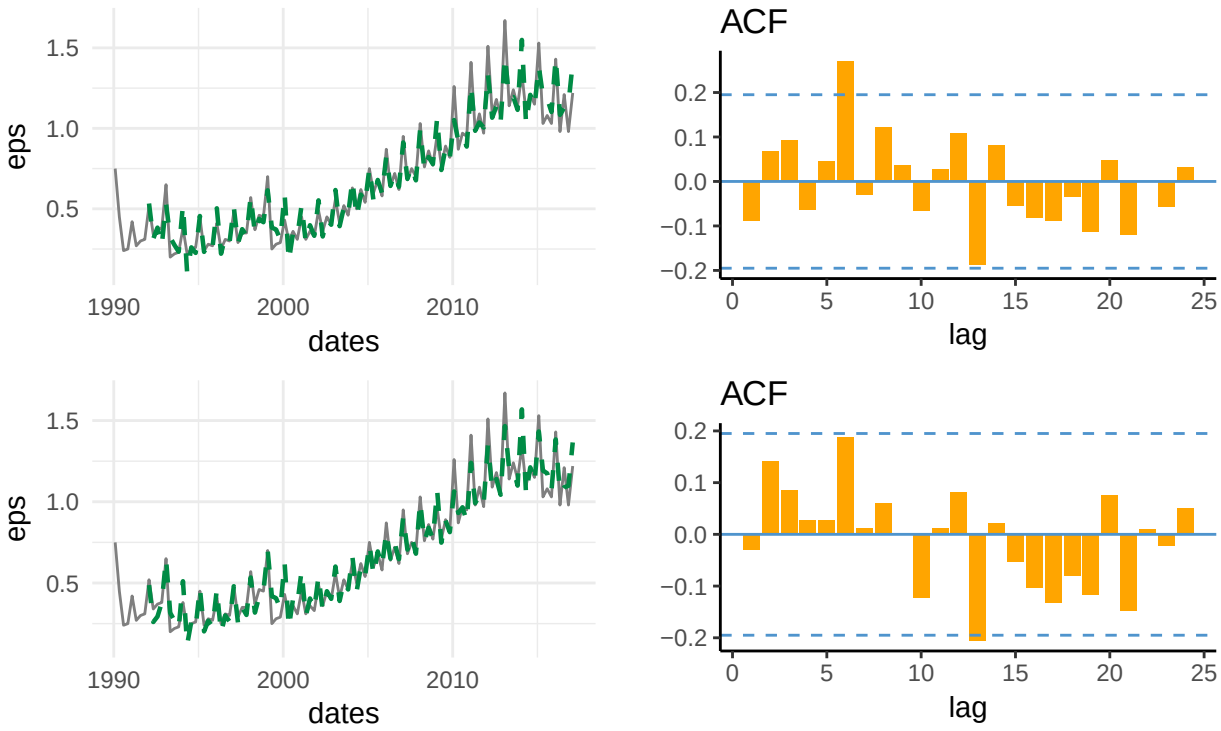
The next step in modeling the EPS series is to account for the seasonality in the EPS series by adding quarterly dummy variables to the regression. The results are shown below for the piece-wise linear trend and the quadratic trend models. It is clear that the fit improves in both cases since the fitted line is closer to the EPS time series.



The two breaks that are unaccounted by the quadratic model imply large errors which could be `dummied-out` by adding in the regression two dummy variables that are equal to one in the quarters in which the break occurred. Neutralizing the effect of these large observations has the benefit of providing unbiased estimates of the parameters which helps also in producing more accurate forecasts of the future.



Finally, the residuals still display some auto-correlation which suggests including AR terms in the regression. We add 12 lags and found that lag 1, 4, 5 and 8 are significant and remain so even after we exclude the insignificant lags. The fitted lines for the two models when including the AR component and the ACF of the residuals are shown below. In both cases the overall fit of the model is quite good and the residuals ACF is mostly insignificant.



To compare the goodness-of-fit of these models we report below the Adjusted R^2 of the different specifications considered so far. The best model is specification (E) which consists of the linear trend model with breaks, in addition to seasonality and AR terms. Specification (F) has a performance which is very close and is characterized of the quadratic trend model with dummy variables for the break dates, as well as seasonality and AR terms. Both the seasonal dummy variables and the AR terms are essential in improving the performance of the models. However, the difference in adjusted R^2 are very small and a more powerful test might be to evaluate the performance of the models out-of-sample.

Table 4.6: Adjusted R^2 for different specifications of the EPS model

Model	Adj. R^2
(A) Linear Trend & Break	0.9592
(B) Quadratic Trend + Dummies	0.805
(C) = (A) + Seasonality	0.9184
(D) = (B) + Seasonality	0.8942
(E) = (C) + AR	0.9892
(F) = (E) + AR	0.9898

4.8 Automatic time series modeling

Coming soon … or later …

```
#library(forecast)
#library(prophet)
```

R commands

Table 4.7: R functions used in this Chapter.

acf()	ar()	autoplot()	model.matrix()	ur.df()	NA
adf.test()	arima()	Fstats()	SelectModel()	NA	NA

Table 4.8: R packages used in this Chapter.

dyn	forecast	strucchange	urca
FitAR	ggfortify	tseries	NA

Exercises

1. Exercise 1
2. Exercise 2

Chapter 5

Volatility Models

A stylized fact across many asset classes is that the standard deviation of returns, also referred to as volatility, varies significantly over time. Figure 5.1 shows the time series of the daily percentage returns of the S&P 500 and of the Japanese Yen to US Dollar exchange rate from January 1980 until September 2017. The horizontal dashed lines represent a 95% confidence interval obtained as the sample average return plus/minus 1.96 times the sample standard deviation of the returns. A features that emerges from the Figure is that financial returns stay within these bands for long periods of time followed by periods of large positive and negative returns that last for several months or years. Hence, the dispersion of the distribution of returns in any given day seems to be different with some periods of high volatility and others prolonged periods of low volatility. So far we have discussed models for the expected return, that is $E(R_{t+1})$, but now we are arguing that we should also consider models for the expected variance of returns, $Var(R_{t+1})$. Modeling volatility is thus a important area of financial econometrics since it provides a measure of risk which is an important input to many financial decisions (e.g., from option pricing to risk management).

A measure of market volatility exist already and is represented by the *CBOE Volatility Index* (or `VIX`). The `VIX` is obtained from the implied volatilities of S&P 500 Index option prices and it is interpreted as a measure of market risk or uncertainty contained in option prices. Figure 5.2 shows the daily time

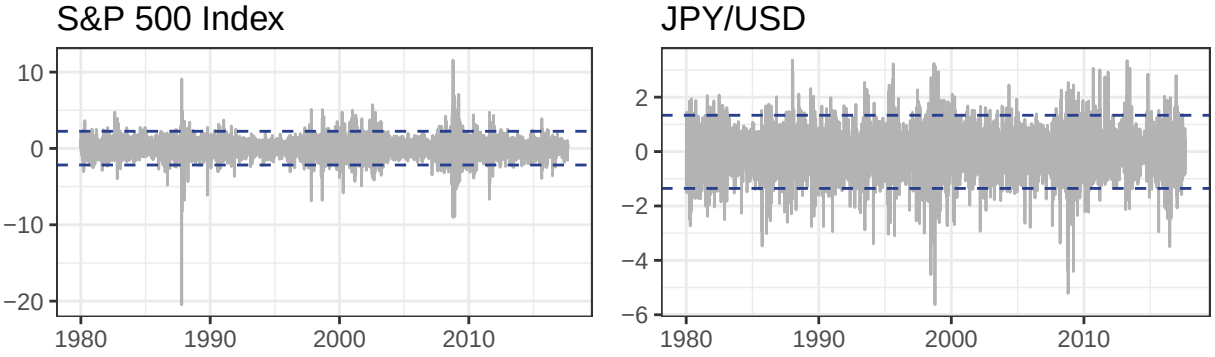


Figure 5.1: Daily percentage returns of the SP 500 Index and the Japanese Yen to US Dollar exchange rates starting in January 1980.

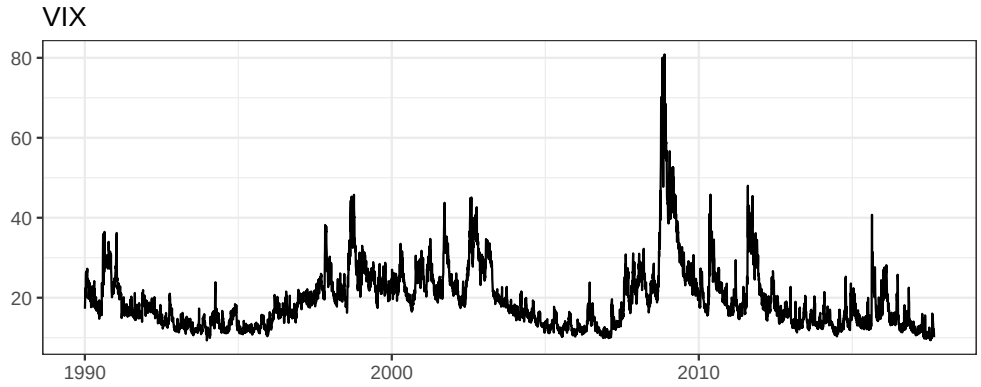


Figure 5.2: The CBOE Volatility Index (VIX) at the daily frequency since January 1990.

series of the `VIX` since January 1990 on an annualized scale. Although a measure of volatility such as `VIX` is available for the S&P 500, for many other assets such a measure is not available and we can either construct it based on option prices or based on historical returns.

This Chapter discusses several approaches that can be used to model and forecast volatility based on historical returns. The basic setup that we consider is a model that assumes that returns at time $t+1$, denoted R_{t+1}, are defined as:

$$R_{t+1} = \mu_{t+1} + \eta_{t+1}$$

This model decomposes the return R_{t+1} in two components:

- *Expected return*: μ_{t+1} represents the predictable part or the expected return of the asset. At the daily and intra-daily frequency, this component is typically assumed equal to zero. Alternatively, it could be assumed to follow an AR(p) process $\mu_{t+1} = \phi_0 + \phi_1 * R_t + \cdots + \phi_p * R_{t-p+1}$ or being a function of other contemporaneous or lagged variables, that is, $\mu_{t+1} = \phi_0 + \phi_1 * X_t + \phi_2 X_{t-1}$.
- *Shock*: η_{t+1} is the unpredictable component of the return; it can assumed to be equal to $\eta_{t+1} = \sigma_{t+1}\epsilon_{t+1}$ where:
  - *Volatility*: σ_{t+1} is the standard deviation of the shock conditional on information available at time t; in a time series model volatility is a function of past returns, e.g., $\sigma^2_{t+1} = \omega + \alpha R_t$.
  - *Standardized shock*: ϵ_{t+1} represents the shock which is assumed to have mean 0 and variance 1. A typical assumption is that it is normally distributed, but alternative distributional assumption can be introduced such as a t distribution that allows for fatter tails.

The aim of this Chapter is to discuss different models to estimate and forecast σ_{t+1} using simple techniques such as Moving Average (MA) and Exponential Moving Average (EMA), followed by a discussion of a time series model such as the Auto-Regressive Conditional Heteroskedasticy (ARCH) model. ARCH models have been extended in several directions, but for the purpose of this Chapter we will consider the two most important generalizations: GARCH (Generalized ARCH) and GJR-GARCH (Glosten, Jagganathan and Runkle ARCH) that includes an asymmetric relationship between the shocks and the conditional variance.

5.1 Moving Average (MA) and Exponential Moving Average (EMA)

A simple approach to estimate the conditional variance is to average the square returns over a recent window of observations, for example the last M observations. The Moving Average (MA) estimate of the variance in day t, σ^2_{t+1}, is given by

$$\sigma^2_{t+1} = \frac{1}{M}\left(R^2_t + R^2_{t-1} + \cdots + R^2_{t-M+1}\right) = \frac{1}{M}\sum_{j=1}^{M} R^2_{t-j+1}$$

and the standard deviation is calculated as $\sqrt{\sigma^2_{t+1}}$. The two extreme values of the window are $M = 1$, which implies $\sigma^2_{t+1} = R^2_t$, and $M = t$, that leads to $\sigma^2_{t+1} = \sigma^2$, where σ^2 represents the unconditional variance estimated on the full sample. Small values of M imply that the volatility estimate is very responsive to the most recent square returns, whilst for large values of M the estimate responds very little to the latest returns. Another way to look at the role of the window size on the smoothness of the volatility is to interpret it as an average of the last M days each carrying a weight of $100/M\%$ (i.e., for $M = 25$ the weight is 4%). When M increases, the weight given to each observation in the window becomes smaller so that each daily square return (even when extreme) has a smaller impact on changing the volatility estimate. The discussion so far represents the typical implementation of the MA approach which relies on the assumption that $\mu_{t+1} = 0$ so that $R_{t+1} = \eta_{t+1}$. More generally, we might want to include an intercept or assume that the expected return follows a AR(1). In this case, the MA is calculated by averaging the squares of η_{t+1} calculated as $\eta_{t+1} = R_{t+1} - \mu_{t+1}$.

The MA approach can be implemented in `R` using the `rollmean()` function provided in package `zoo` which requires to specify the window size (M in the notation above). An example for the S&P 500 daily returns is provided in Figure 5.3:

```
GSPC              <- getSymbols("^GSPC", from="1980-01-01", auto.assign=FALSE)
sp500daily        <- 100 * ClCl(GSPC) %>% na.omit
names(sp500daily) <- "RET"
sigma25           <- zoo::rollmean(sp500daily^2, 25, align="right")
names(sigma25)    <- "MA25"
ggplot(merge(sigma25, sp500daily)) + geom_line(aes(time(sp500daily), abs(RET)),color="gray80") +
  theme_bw() + geom_line(aes(index(sp500daily), MA25^0.5), color="orangered3")  +
  labs(x=NULL, y=NULL)
```

The effect of increasing the window size from $M = 25$ (continuous line) to 100 (dashed line) is shown in Figure 5.4: the longer window smooths out the fluctuation of the MA(25) since each observation is given a smaller weight. This implies that large (negative or positive) returns increase volatility less when using longer windows relative to shorter ones.

One drawback of the MA approach is that large daily returns are able to increase/decrease significantly the volatility estimate when they enter/exit the window of M days. An extension of the MA approach is to have a smoothly decreasing weight assigned to older returns instead of the discrete jump of the weight from $1/M$ to 0 at the $M+1$th observation. This approach is called Exponential Moving Average (EMA) and gained popularity in finance since the investment bank J.P. Morgan proposed it as a model to predict volatility for Value-at-Risk (VaR) calculations. The EMA with parameter λ is calculated as follows:

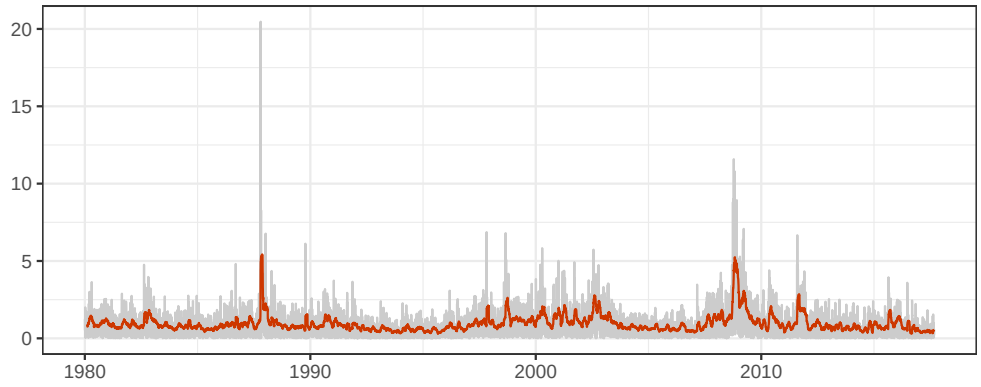


Figure 5.3: Absolute returns of the SP 500 Index and the Moving Average (MA) with parameter M set to 25.

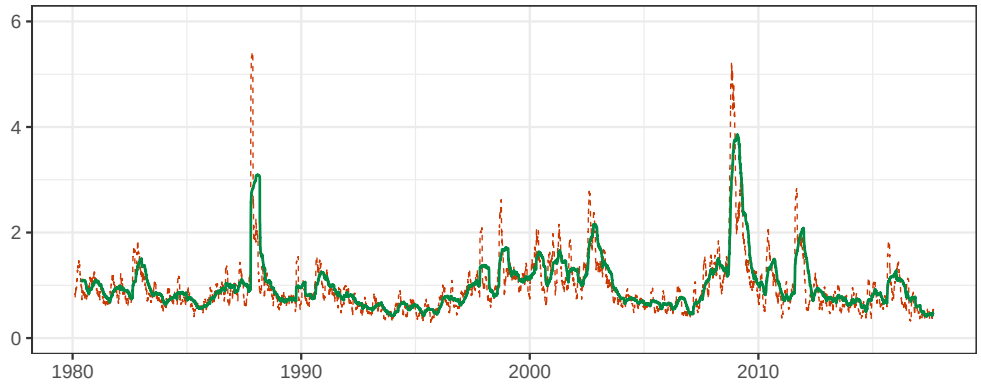


Figure 5.4: The Moving Average (MA) with parameter M set to 25 and 100. Can you guess if the dashed line corresponds to M equal to 25 or 100 days?

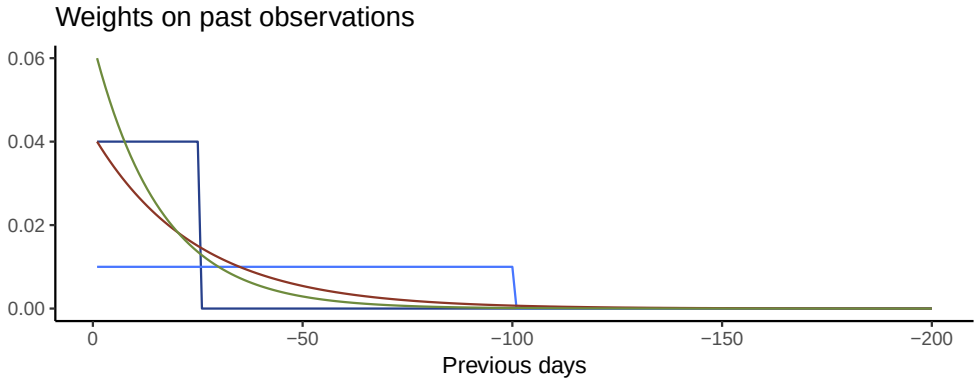


Figure 5.5: Weight on past values for MA with M equal to 25 and 100 and EMA equal to 0.04 and 0.06.

$$\sigma^2_{t+1} = \lambda \sum_{j=1}^{\infty} (1-\lambda)^{j-1} R^2_{t-j+1}$$

where λ is a smoothing parameter between zero and one. After some algebra, the expression above can be rewritten as follows:

$$\sigma^2_{t+1} = (1-\lambda) * \sigma^2_t + \lambda R^2_t$$

which shows that the conditional variance estimate in day $t+1$ is given by a weighted average of the previous day estimate and the square return in day t, with the weights equal to $1-\lambda$ and λ, respectively. A typical value of λ is 0.06 which means that day t has a 6% weight in calculating the volatility forecast, day $t-1$ has weight $(0.94 * 6) = 5.64\%$, day $t-2$ has weight $(0.94^2 * 6) = 5.3016\%$, day $t-k$ has weight $(0.94^k * 6)\%$ and so on. Figure 5.5 shows the MA and EMA weight on past observations. The MA method assigns a positive and constant weight to the last M observations as opposed to the EMA approach that spreads the weight over a longer period and weights differently more recent observation relative to older ones.

The typical value for M is 25 (one trading month) and for λ it is 0.06. The graph below compares the volatility estimates from the MA(25) and EMA(0.06) methods. In this example we use the package `TTR` that provides functions to calculate moving averages, both simple and of the exponential type:

```
library(TTR)
# SMA() function for Simple Moving Average; n = number of days
ma25  <- SMA(sp500daily^2, n=25)
names(ma25) <- "MA25"
# EMA() function for Exponential Moving Average; ratio = lambda
ema06 <- EMA(sp500daily^2, ratio=0.06)
names(ema06) <- "EMA06"
autoplot(merge(ma25, ema06)^0.5, ncol=2) + theme_bw()
```

Figure 5.6 shows the MA and EMA daily volatility estimate for over 23 years and, at this scale, it is difficult to see large differences between the two methods. However, by plotting a sub-period of time, some differences become evident. Figure 5.7 shows the period between beginning 2008 and end of 2009.

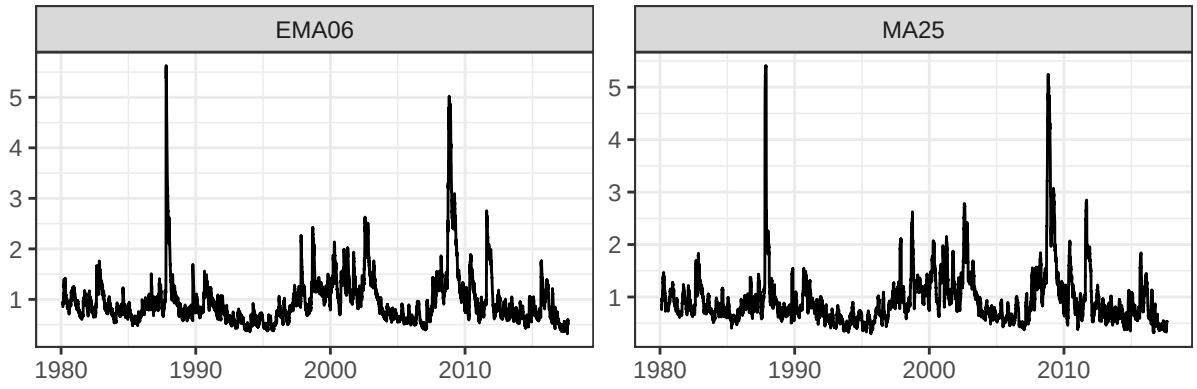


Figure 5.6: Time series of the MA(25) and EMA(0.06) volatility estimates.

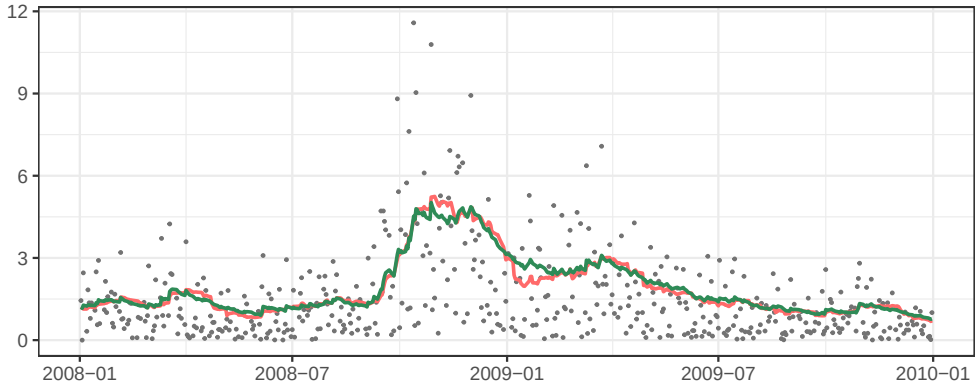


Figure 5.7: Volatility estimates of MA(25) and EMA(0.06) from January 2008 to December 2009. The dots represents the absolute daily returns.

The biggest difference between the two methods is that in periods of rapid change (from small/large to large/small absolute returns) the EMA line captures the change in volatility regime more smoothly relative to the simple MA. This is in particular clear in the second part of 2008 and the beginning of 2009, with some differences between the two lines.

The parameters of MA and EMA are, usually, chosen a priori rather than being estimated from the data. There are possible ways to estimate these parameters although they are not popular in the financial literature and typically do not provide great benefit for practical purposes. In the following Section we discuss a more general volatility model which generalizes the EMA model and it is typically estimated from the data.

5.2 Auto-Regressive Conditional Heteroskedasticity (ARCH) models

The ARCH model was proposed by Engle (1982) to model the time-varying variance of the inflation rate. Since then it has gained popularity in finance as a way to model the changes in the volatility of asset

returns. Assuming that $\mu_{t+1} = 0$ as in the previous Sections, the model simplifies to $R_{t+1} = \eta_{t+1} = \sigma_{t+1}\epsilon_{t+1}$, that is, the return next period is given by an exogenous shock, ϵ_{t+1}, amplified by the time-varying volatility, σ_{t+1}[1]. The conditional variance σ^2_{t+1} can be assumed to be a function of the previous period square return, that is,

$$\sigma^2_{t+1} = \omega + \alpha * R^2_t$$

where ω and α are parameters to be estimated. The AR part in the ARCH name relates to the fact that the variance of returns is a function of the lagged (square) returns. This specification is called the ARCH(1) model and can be generalized to include p lags which results in the ARCH(p) model:

$$\sigma^2_{t+1} = \omega + \alpha_1 * R^2_t + \cdots + \alpha_p * R^2_{t-p+1}$$

The empirical evidence indicates that financial returns typically require p to be large. A parsimonious alternative to the ARCH model is the Generalized ARCH (GARCH) model which is characterized by the following Equation for the conditional variance:

$$\sigma^2_{t+1} = \omega + \alpha * R^2_t + \beta * \sigma^2_t$$

In this case the variance is affected by R^2_t as well as by last period conditional variance σ^2_t. This specification is parsimonious, relative to an ARCH model with large p, since all past square returns are included in σ^2_{t-1}, but using only 3 parameters rather than of $p+1$. The specification above represents a GARCH(1,1) model since it includes one lag of the square return and the conditional variance. However, it can be easily generalized to the GARCH(p,q) case in which p lags of the square return and q lags of the conditional variance are included. The empirical evidence suggests that the GARCH(1,1) is typically the best model for several asset classes and it is only in rare instances outperformed by p and q different from 1.

We can derive the mean of the conditional variance in the case of the GARCH(1,1) which is given by:

$$\sigma^2 = \omega/(1 - (\alpha + \beta))$$

where $\sigma^2 = E(\sigma^2_{t+1})$ and represents the *unconditional* variance as opposed to σ^2_t that denotes the *conditional* variance of the returns. The difference between these quantities is that σ^2_{t+1} is a forecast of volatility based on the information available at time t, while σ^2 represents the time-average of the conditional forecasts. The Equation for σ^2 indicates that a condition for the variance to be finite is that $\alpha + \beta < 1$. If this condition is not satisfied, the variance of the returns is infinite and volatility behaves like a random walk model, rather than being mean-reverting. This situation is similar to the case of the AR(1) in which a coefficient (in absolute value) less than 1 ensures that the time series is stationary, and thus mean-reverting. We can think similarly about the case of the variance, where the condition that $\alpha + \beta < 1$ guarantees that volatility is stationary and mean reverting. What does it mean that variance (or volatility) is mean reverting? It means that volatility might be highly persistent, but oscillates between periods in which it is higher and lower than its unconditional level σ. This interpretation is consistent with the empirical observation that volatility switches between periods in which it is high and others in which it is low. On the other hand, non-stationary volatility implies that periods of high volatility (i.e., higher than average) are expected to persist in the long run rather than reverting back to

[1] More generally, if $\mu_{t+1} \neq 0$ then the conditional variance is a function of the current and past values of η^2_t rather than R^2_t.

the unconditional variance. The same holds for periods of low volatility which, in case of non-stationarity, is expected to last in the long run. This distinction is practically relevant, in particular when the interest is to forecast future volatility and at long horizons.

The distinction between mean-reverting (i.e., stationary) and non-stationary volatility is an important property that differentiates volatility models. In particular, the EMA model can be considered a special case of the GARCH model when the parameters are set equal to $\omega = 0$, $\alpha = \lambda$ and $\beta = 1 - \lambda$. This shows that EMA imposes the assumption that $\alpha + \beta = 1$, and thus that volatility is non-stationary. Empirically, this hypothesis can be tested by assuming the null hypothesis $\alpha + \beta = 1$ versus the one sided hypothesis that $\alpha + \beta < 1$.

Another popular GARCH specification was proposed by Glosten, Jagganathan and Runkle (hence, GJR-GARCH) which assumes that the square return has a different effect on volatility depending on its sign. The conditional variance Equation of this model is:

$$\sigma^2_{t+1} = \omega + \alpha_1 * R^2_t + \gamma_1 R^2_t * I(R_t \leq 0) + \beta * \sigma^2_t$$

In this specification, when the return is positive its effect on the conditional variance is α_1 and when it is negative the effect is $\alpha_1 + \gamma_1$. Testing the hypothesis that $\gamma_1 = 0$ thus provides a test of the symmetry of the effect of shocks on volatility. The estimation of the model on asset returns from many asset classes shows that α_1 is estimated close to zero and insignificant, while γ_1 is found positive and significant. The evidence thus suggests that negative shocks lead to more uncertainty and an increase in the volatility of asset returns, while positive shocks do not have a relevant effect.

5.2.1 Estimation of GARCH models

ARCH/GARCH models cannot be estimated using OLS because the model is nonlinear in parameters[2] The estimation of GARCH models is thus performed using an alternative estimation technique called **Maximum Likelihoood** (**ML**). The ML estimation method represents a general estimation principle that can be applied to a large set of models, not only to volatility models.

It might be useful to first discuss the ML approach to estimation in comparison to the familiar OLS approach. The OLS approach is to choose the parameter values that minimize the sum of the square residuals which measures the *unfitness* of the model to explain the data for a certain set of parameter values. Instead, the approach of ML is to choose the parameter values that maximize the likelihood or probability that the data were generated by the model. For the case of a simple AR model with homoskedastic errors it can be shown that the OLS and ML estimators are equivalent. A difference between OLS and ML is that the first provides analytical formula for the estimation of linear models, whilst the ML estimator is the result of numerical optimization[3].

We assume that volatility models are of the general form discussed earlier, i.e., $R_{t+1} = \mu_{t+1} + \sigma_{t+1}\epsilon_t$. Based on the distributional assumption introduced earlier, we can say that the standardized residuals $(R_{t+1} - \mu_{t+1})/\sigma_{t+1}$ should be normally distributed with mean 0 and variance 1. Bear in mind that

[2]Furthermore, the volatility is not directly observable which does not allow the OLS estimation of the conditional variance model as it would be the case if the dependent variable is observable. However, recent advances in financial econometrics that will be discussed in the following Chapter have partly overcome this difficulty.

[3]Numerical optimization consists of algorithms that are used to find the minimum or maximum of an objective function.

both μ_{t+1} and σ_t^2 depend on parameters that are denoted by θ_μ and θ_σ. For example, for an AR(1)-GARCH(1,1) model the parameter of the mean is $\theta_\mu = (\phi_0, \phi_1)$ and the parameters of the conditional variance is $\theta_\sigma = (\omega, \alpha, \beta)'$. Given the assumption of normality, the density or likelihood function of R_{t+1} is

$$f(R_{t+1}|\theta_\mu, \theta_\sigma) = \frac{1}{\sqrt{2\pi\sigma_{t+1}^2(\theta_\sigma)}} \exp\left[-\frac{1}{2}\left(\frac{R_{t+1} - \mu_{t+1}(\theta_\mu)}{\sigma_{t+1}(\theta_\sigma)}\right)^2\right]$$

which represents the normal density for the return in day $t+1$. We wrote the conditional mean and variance as $\mu_{t+1}(\theta_\mu)$ and $\sigma_t^2(\theta_\sigma)$ to make explicit their dependence on parameters over which the likelihood function will be maximized. Since we have T returns, we are interested in the joint likelihood of the observed returns and denoting by p the largest lag of R_{t+1} used in the conditional mean and variance, we can define the (conditional) likelihood function $\mathcal{L}(\theta_\mu, \theta_\sigma)$ $(= f(R_{p+1}, \cdots, R_{t+1}|\theta_\mu, \theta_\sigma, R_1, \cdots, R_p))$ as

$$\mathcal{L}(\theta_\mu, \theta_\sigma) = \prod_{t=p+1}^{T} f(R_{t+1}|\theta_\mu, \theta_\sigma) = \prod_{t=p+1}^{T} \frac{1}{\sqrt{2\pi\sigma_{t+1}^2(\theta_\sigma)}} \exp\left[-\frac{1}{2}\left(\frac{R_{t+1} - \mu_{t+1}(\theta_\mu)}{\sigma_{t+1}(\theta_\sigma)}\right)^2\right]$$

The ML estimates $\hat{\theta}_\mu$ and $\hat{\theta}_\sigma$ are thus obtained by maximizing the likelihood function $\mathcal{L}(\theta_\mu, \theta_\sigma)$. It is convenient to log-transform the likelihood function to simplify the task of maximizing the function. We denote the log-likelihood by $l(\theta_\mu, \theta_\sigma)$ and it is given by

$$l(\theta_\mu, \theta_\sigma) = \ln \mathcal{L}(\theta_\mu, \theta_\sigma) = -\frac{1}{2}\sum_{t=p+1}^{T}\left[\ln(2\pi) + \ln\sigma_{t+1}^2(\theta_\sigma) + \left(\frac{R_{t+1} - \mu_{t+1}(\theta_\mu)}{\sigma_{t+1}(\theta_\sigma)}\right)^2\right]$$

since the first term $\ln(2\pi)$ does not depend on any parameter, it can be dropped from the function. The estimates $\hat{\theta}_\mu$ and $\hat{\theta}_\sigma$ are then obtained by maximizing

$$l(\theta_\mu, \theta_\sigma) = -\frac{1}{2}\sum_{t=p+1}^{T}\left[\ln\sigma_{t+1}^2(\theta_\sigma) + \left(\frac{R_{t+1} - \mu_{t+1}(\theta_\mu)}{\sigma_{t+1}(\theta_\sigma)}\right)^2\right]$$

The maximization of the likelihood or log-likelihood is performed numerically, which means that we use algorithms to find the maximum of this function. The problem with this approach is that, in some situations, the likelihood function is not well-behaved and characterized by local maxima. The numerical search has to be started at some initial values and, in the difficult cases just mentioned, the choice of these values is extremely important to achieve the global maximum of the function. The choice of valid starting values for the parameters can be achieved by a small-scale grid search over the space of possible values of the parameters.

In the case of volatility models the likelihood function is usually well-behaved and achieves a maximum quite rapidly. In the following Section we discuss some `R` packages that implement GARCH estimation and forecasting.

5.2.2 Inference for GARCH models

... coming soon (or later) ...

5.2.3 GARCH in R

There are several packages that provide functions to estimate models from the GARCH family. One of the earliest is the `garch()` function in the `tseries` package, which is however quite limited in the type of models it can estimate. More flexible functions for GARCH estimation are provided by the package `fGarch`, that allows to specify the conditional mean μ_{t+1} and the conditional variance σ^2_{t+1}. The function to perform the estimation is called `garchFit`. The example below shows the application of the `garchFit` function to the daily returns of the S&P 500 index. The first example estimates a GARCH(1,1) with only the intercept in the conditional mean (i.e., $\mu_{t+1} = \mu$) and then consider an AR(1) for the conditional mean (i.e., $\mu_{t+1} = \mu + \beta_1 R_t$) which we denote as the AR(1)-GARCH(1,1) model. Notice that to specify the AR(1) for the conditional mean we use the function `arma(p,q)` which is a more general function than those used to estimate AR(p) models.

```
library(fGarch)
fit <- garchFit(~garch(1,1), data=sp500daily, trace=FALSE)
round(fit@fit$matcoef, 3)
```

```
        Estimate  Std. Error  t value Pr(>|t|)
mu         0.059       0.008    7.079        0
omega      0.015       0.002    7.182        0
alpha1     0.081       0.006   14.032        0
beta1      0.906       0.007  133.890        0
```

The output provides standard errors for the parameter estimates as well as t-stats and p-values for the null hypothesis that the coefficients are equal to zero. The results show that the mean μ is estimated equal to 0.059% and it is statistically significant even at 1%. Hence, for the daily S&P 500 returns the assumption of a zero expected return is rejected. The estimate of α is 0.081 and of β is 0.906, with their sum equal to 0.987, which is quite close enough to 1 to conclude that volatility is very persistent and close to being non-stationary. It might also be interesting to evaluate the need to introduce some dependence in the conditional mean, for example, by assuming an AR(1) model. The command `arma(1,0) + garch(1,1)` in the `garchFit()` function estimates an AR(1) model with GARCH(1,1) conditional variance:

```
fit <- garchFit(~ arma(1,0) + garch(1,1), data=sp500daily, trace=FALSE)
round(fit@fit$matcoef, 3)
```

```
        Estimate  Std. Error  t value Pr(>|t|)
mu         0.058       0.008    7.045    0.000
ar1        0.003       0.011    0.265    0.791
omega      0.015       0.002    7.182    0.000
alpha1     0.081       0.006   14.031    0.000
beta1      0.906       0.007  133.850    0.000
```

The estimate of the AR(1) coefficient is 0.003 and it is not statistically significant at 10% level, which shows the irrelevance of including dependence in the conditional mean of daily financial returns.

Based on the GARCH model estimation, we can then obtain the conditional variance $\hat{\sigma}^2_t$ and the conditional standard deviation $\hat{\sigma}_t$. The conditional standard deviation is extracted by appending `@sigma.t` to the `garchFit` object:

```
sigma <- fit@sigma.t  # class is numeric
qplot(time(sp500daily), sigma, geom="line", xlab=NULL, ylab="") +
  theme_bw() + labs(title="Conditional standard deviation")
```

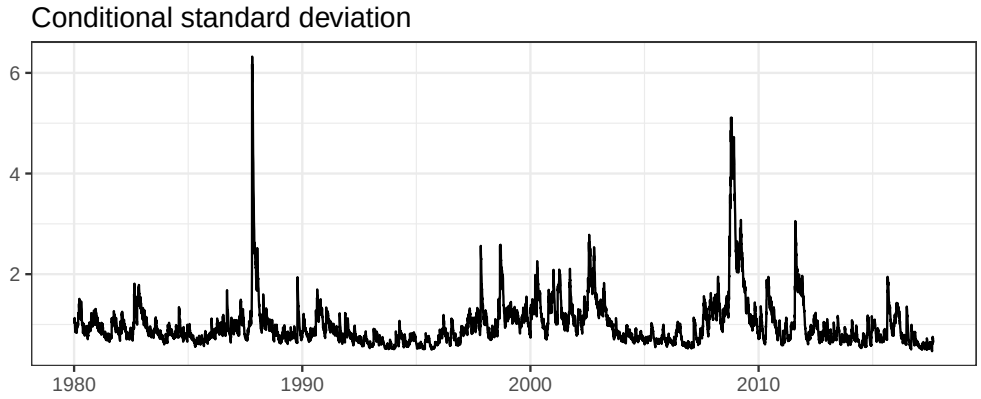


Figure 5.8: Conditional standard deviation estimated by the AR(1)-GARCH(1,1) model for the SP500 returns.

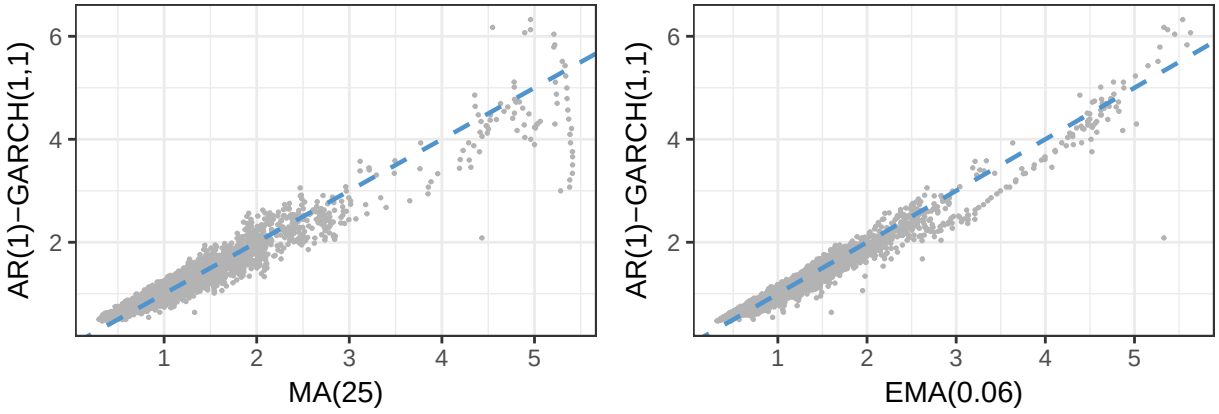


Figure 5.9: Comparison of the estimated standard deviation from the GARCH(1,1) model and from the EMA with parameter equal to 0.06.

Figure 5.8 shows the significant variation over time of the standard deviation that alternates between periods of low and high volatility, in addition to sudden increases in volatility due to the occurrence of large returns. It is also interesting to compare the fitted standard deviation from the GARCH model with the ones obtained from the MA and EMA methods. The correlation between the MA and GARCH conditional standard deviation is 0.967 and between EMA and GARCH is 0.981 and, to a certain extent, they can be considered very good substitutes for each other (in particular at short horizons). Figure 5.9 shows the scatter plot of the conditional standard deviation for the GARCH model vs the MA and EMA estimates. Points above/below the diagonal represent days in which the GARCH estimate is larger/smaller relative to the MA/EMA estimates. The largest differences occur for the volatility estimates obtained with the MA method for values larger than 5%. This is due to the different reaction of σ^2_{t+1} to large (absolute) returns in the MA models relative to the GARCH and EMA models.

The residuals of the GARCH model can also provide valuable information about the goodness of the model in explaining the returns. Appending `@residuals` to the `garchFit` estimation object we can extract the residuals of the GARCH model that represent an estimate of $\sigma_{t+1}\epsilon_{t+1}$. Figure 5.10 shows that the residuals maintain most of the characteristics of the raw returns, in particular the clusters of

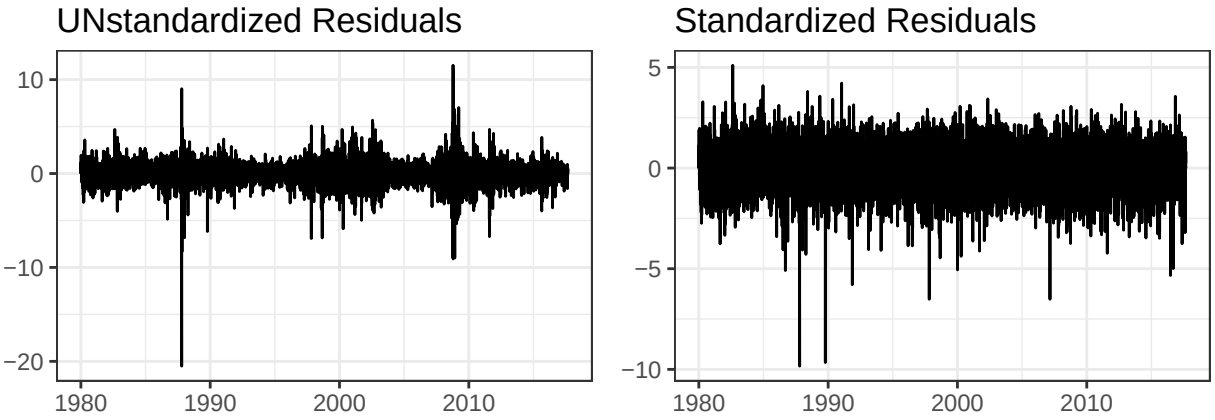


Figure 5.10: Standardized and unstandardized residuals for the SP500 daily returns using the AR(1)-GARCH(1,1) model.

volatility. This is because these residuals have been obtained from the last fitted model in which $\hat{\mu}_{t+1} = 0.058 + (0.003) * R_{t-1}$ and the residuals are thus given by $R_{t+1} - \hat{\mu}_{t+1}$. The contribution of the intercept is to demean the return series while the small coefficient on the lagged return leads to returns that are very close to the residuals.

```
res <- fit@residuals  # class is numeric
p1  <- qplot(time(sp500daily), res, geom="line", main="UNstandardized Residuals") +
  labs(x=NULL, y=NULL) + theme_bw()
p2  <- qplot(time(sp500daily), res/sigma, geom="line", main="Standardized Residuals") +
  labs(x=NULL, y=NULL) + theme_bw()
gridExtra::grid.arrange(p1, p2, ncol=2)
```

Figure 5.10 shows the time series of the residuals $\sigma_{t+1}\epsilon_{t+1}$ together with the standardized residuals ϵ_{t+1} which should satisfy the properties of being independent over time (i.e., no auto-correlation) and normally distributed (given our earlier assumptions). The time series of the standardized residuals seem to be exempt from heteroskedasticity (e.g., volatility clustering) which has been taken into account by σ_{t+1}. However, we notice a few large negative standardized residuals that are quite unlikely to happen assuming a normal distribution[4].

The first property of the standardized residuals is that they should be approximately normally distributed. To assess this assumption, Figure 5.11 compares the histogram of the standardized residuals with the standard normal distribution. The histogram of the standardized residuals has a large peak at the center of the distribution and fatter left tail relative to the normal, although this is difficult to see in the graph. However, calculating the skewness of the standardized residuals, equal to -0.463, and the excess kurtosis, equal to -0.463, indicate that the large negative returns in the time series plot point to the deviation of the standardized returns distribution from normality. We can conclude that the normality assumption we made earlier seems not to be supported in the data in favor of a distribution with fatter tails and possibly skewed. However, the skewness could also be due to misspecification of the conditional variance in the sense that the GARCH(1,1) model might be missing some important features of the volatility dynamics.

[4]The probability of an event of minus 6 or smaller is very small and on a sample length of 9501 it is expected to occur 9 times every 1 million days.

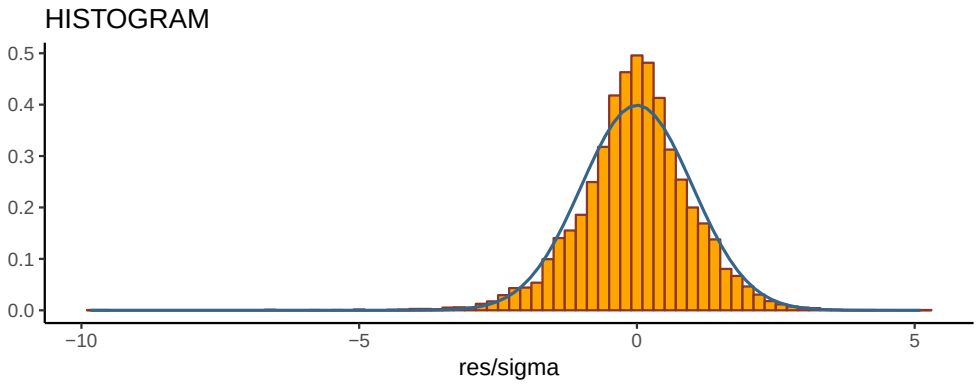


Figure 5.11: Histogram of the standardized residuals and the standard normal distribution.

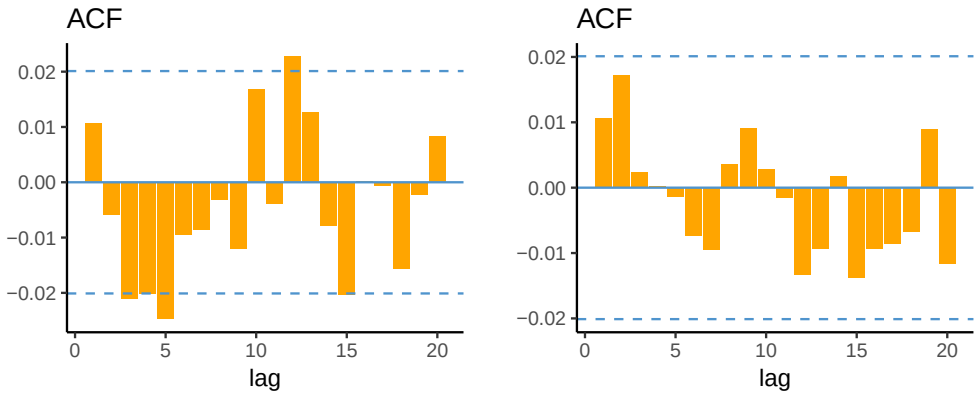


Figure 5.12: ACF of the standardized residuals of the AR(1)-GARCH(1,1) model and their square values.

We can also consider the auto-correlation of residuals and square residuals to assess if there is neglected dependence in the conditional mean and variance and the ACF are shown in Figure 5.12. Overall, there is weak evidence of auto-correlation in the standardized residuals and in their squares, so that, from this standpoint, the GARCH(1,1) model seems to be well specified to model the daily returns of the S&P 500.

```
p1 <- ggacf((res/sigma), lag=20)
p2 <- ggacf((res/sigma)^2, lag=20)
gridExtra::grid.arrange(p1, p2, ncol=2)
```

Another package that provides functions to estimate a wide range of GARCH models is the `rugarch` package. This packages requires first to specify the functional form of the conditional mean and variance using the function `ugarchspec()` and then proceed with the estimation using the function `ugarchfit()`. Below is an example for an AR(1)-GARCH(1,1) model estimated on the daily S&P 500 returns:

```
library(rugarch)
spec = ugarchspec(variance.model=list(model="sGARCH",garchOrder=c(1,1)),
                  mean.model=list(armaOrder=c(1,0)))
fitgarch = ugarchfit(spec = spec, data = sp500daily)
```

```
       Estimate      SE
mu        0.059   7.059
```

```
ar1       0.003   0.265
omega     0.015   6.999
alpha1    0.081  13.775
beta1     0.906 130.497
```

The estimation results are the same as those obtained for the `fGarch` package and more information about the estimation results can be obtained using command `show(fitgarch)`.

The estimation of the GJR-GARCH model is quite straightforward in this package since it requires only to specify the option `model='gjrGARCH'` in `ugarchspec()`, in addition to selecting the orders for the conditional mean and variance as shown below:

```
spec = ugarchspec(variance.model=list(model="gjrGARCH",garchOrder=c(1,1)),
                  mean.model=list(armaOrder=c(1,0)))
fitgjr = ugarchfit(spec = spec, data = sp500daily)
```

	Estimate	SE
mu	0.035	4.203
ar1	0.008	0.686
omega	0.019	8.343
alpha1	0.018	4.225
beta1	0.904	131.711
gamma1	0.119	11.881

The results for the S&P 500 confirm the earlier discussion that positive returns have a negligible effect in increasing volatility ($\hat{\alpha}_1$ =0.018) while negative returns have a very large and significant effect ($\hat{\gamma}_1$ = 0.119). Figure 5.13 compares the time series of the volatility estimate $\hat{\sigma}_t$ for the GARCH and GJR models. Although the time series look pretty similar, the scatter plot shows that during turbulent times the volatility estimate of GJR is larger relative to GARCH(1,1) due to the more pronounced reaction to negative returns.

```
p1 <- autoplot(merge(GARCH = sigma(fitgarch), GJR = sigma(fitgjr)), scales="fixed") + theme_bw()
p2 <- ggplot(data=merge(GARCH = sigma(fitgarch), GJR = sigma(fitgjr))) +
  geom_point(aes(x = GARCH, y=GJR), color="gray70", size=0.3) +
  geom_abline(intercept=0, slope=1, color="steelblue3", linetype="dashed", size=0.8) + theme_bw()
gridExtra::grid.arrange(p1, p2, ncol=2)
```

Is this asymmetry a general feature of financial data or is it more limited to some asset classes? We can answer this question by estimating the GJR-GARCH model on the daily returns of the JPY/USD exchange rate. The estimation results for γ in this case provide a t-stat of 1 that is not significant even at 10% significance level thus indicating that exchange rate returns do not show the asymmetric behavior of volatility that is a features of stock returns.

	Estimate	SE
mu	-0.004	-0.651
ar1	0.003	0.283
omega	0.008	2.531
alpha1	0.040	4.072
beta1	0.938	54.315
gamma1	0.007	1.289

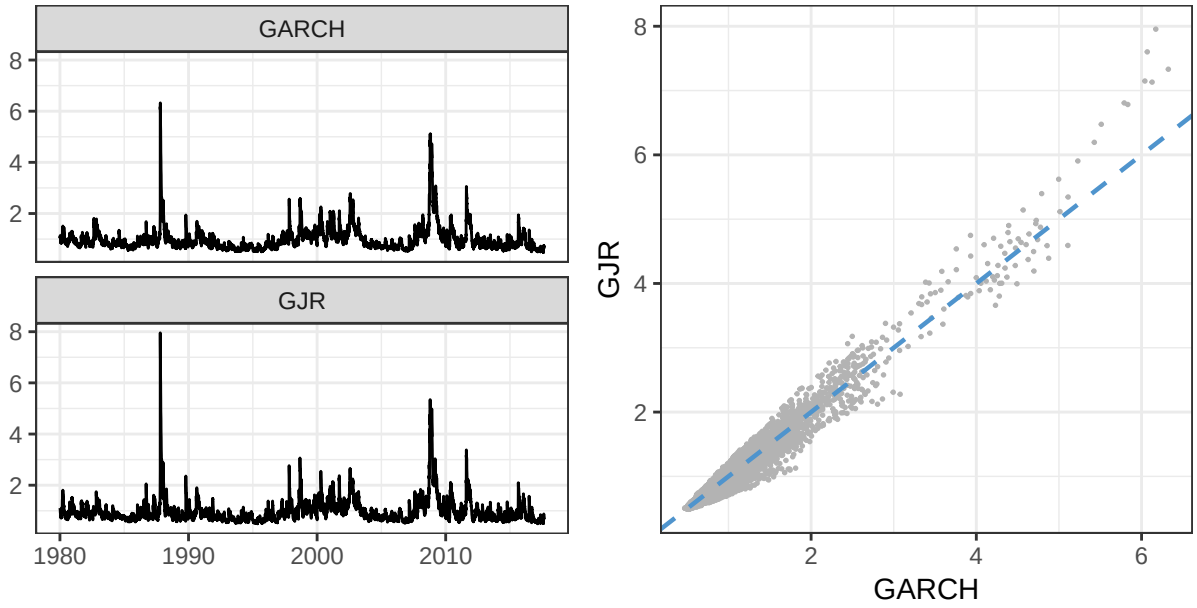


Figure 5.13: Time series of the GARCH and GJR volatility estimate (left) and their scatter plot (right).

The selection of the best performing volatility model can be done using the AIC selection criterion, similarly to the selection of the optimal order p for AR(p) models. The package `rugarch` provides the function `infocriteria()` that calculates AIC and several other selection criteria. These criteria are different in the amount of penalization that they involve for adding more parameters (AR(1)-GJR has one parameter more than AR(1)-GARCH). For all criteria, the best model is the one that provides the smallest value. In this case the GJR specification clearly outperforms the basic GARCH(1,1) model for all criteria.

```
fit.ic <- cbind(infocriteria(fitgarch), infocriteria(fitgjr))
colnames(fit.ic) <- c("GARCH","GJR")
```

```
                GARCH     GJR
Akaike        2.66663 2.64352
Bayes         2.67040 2.64804
Shibata       2.66663 2.64352
Hannan-Quinn 2.66791 2.64505
```

The function `ugarchforecast()` allows to compute the out-of-sample forecasts for a model `n.ahead` periods. The plot below shows the forecasts made in 2017-09-01 when the volatility estimate σ_{t+1} was 0.629 for GARCH and 0.665. Both models forecast an increase in volatility in the future since the volatility is mean-reverting in these models (and at the moment the forecast was made volatility was below its long-run level σ).

```
nforecast = 250
garchforecast <- ugarchforecast(fitgarch, n.ahead = nforecast)

ggplot(temp) + geom_line(aes(Date, GARCH), color="tomato3") +
  geom_line(aes(Date, GJR), color="steelblue2") +
  geom_hline(yintercept = sd(sp500daily), color="seagreen4", linetype="dashed") +
```

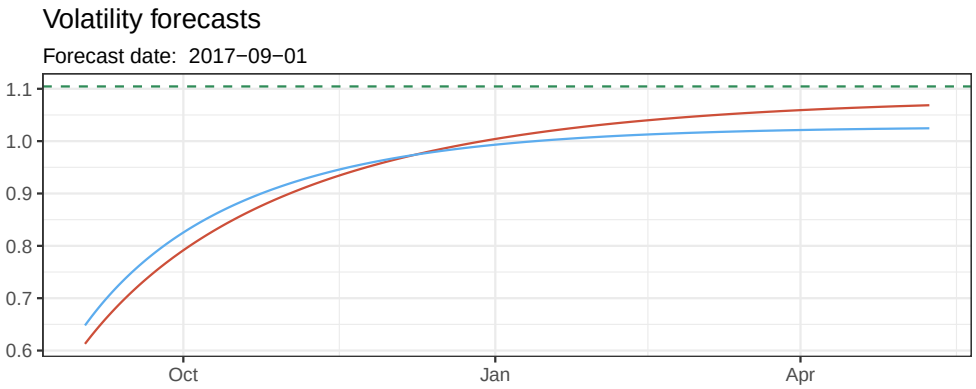


Figure 5.14: Volatility forecasts from 1 to 250 days ahead. The horizontal line represents the unconditional standard deviation of the returns.

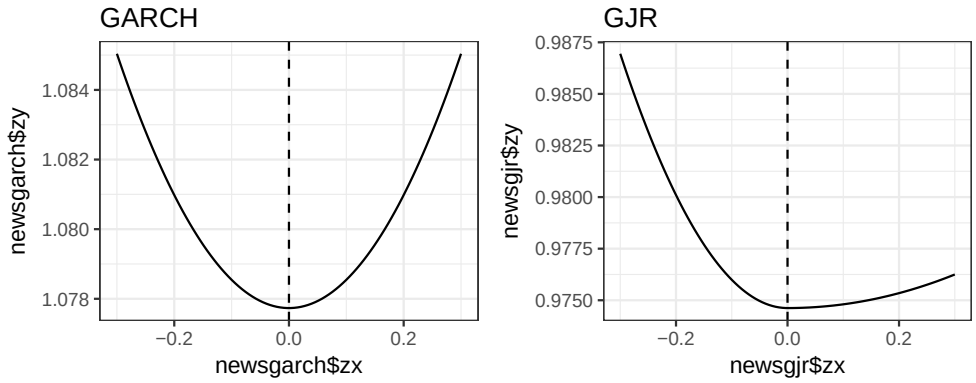


Figure 5.15: News Impact Curve (NIC) for the GARCH and GJR models.

```
  labs(x=NULL, y=NULL, title="Volatility forecasts",
       subtitle=paste("Forecast date: ", end(sp500daily))) + theme_bw()
```

Finally, we can compare the GARCH and GJR specifications based on the effect of a shock (ϵ_t) on the conditional variance (σ_t^2). The left plot refers to the GARCH(1,1) model and clearly show that positive and negative shocks (of the same magnitude) increase the conditional variance by the same amount. However, the news impact curve for the GJR model clearly shows the asymmetric effect of shocks, since there is no effect when ϵ_{t-1} is positive but a large effect when the shock is negative.

```
newsgjr   <- newsimpact(fitgjr)
p1 <- qplot(newsgarch$zx, newsgarch$zy, geom="line", main="GARCH") +
  geom_vline(xintercept = 0, linetype="dashed") + theme_bw()
p2 <- qplot(newsgjr$zx, newsgjr$zy, geom="line", main="GJR") +
  geom_vline(xintercept = 0, linetype="dashed") + theme_bw()
gridExtra::grid.arrange(p1, p2, ncol=2)
```

R commands

Table 5.1: R functions used in this Chapter.

rollmean()	garch()	ugarchfit()
SMA()	garchFit()	ugarchforecast()
EMA()	ugarchspec()	newsimpact()

Table 5.2: R packages used in this Chapter.

zoo	tseries	rugarch
TTR	fGarch	NA

Exercises

1. Exercise 1
2. Exercise 2

Chapter 6

High-Frequency Data

Since the 1990s the availability of intra-day transaction and quote data has sparkled research on existing questions and new ones have emerged from the analysis of the micro-structure of these markets. One of the old questions in financial econometrics is how to model and predict volatility as we discussed in the previous Chapter using several GARCH-type models. As we mentioned earlier, there has been a fair amount of research on **realized volatility** measures which represent an observable quantity for volatility based on intra-day returns. This approach is not only interesting because it provided an observable measure of volatility, but also because it contributed to the development of a new set of models that takes this measure as the dependent variable and try to explain it with time series and multivariate models. The scope of this Chapter is not so much to overview all of the interesting issues that arise with high-frequency data, but rather to get students started with the analysis of these data and constructing measures of realized volatility in `R`. To this goal, we will use the `highfrequency` package which provides several useful functions for data handling and construction of realized volatility measures.

The two main financial markets that provide high-frequency data are the FOREX and the US equity markets. The latter is contained in the `TAQ` (Trade And Quote) database which contains all trades (including quantity traded) and quotes for all listed stocks since the early 1990s. Contrary to this centralized market, the FOREX market is decentralized and there are only observations for quote of many currency pairs but no transaction data. An interesting issue of the FOREX market is that it is always open and geographically disperse since the trading day starts in Tokyo, which is followed by the opening of the London market, and finally by New York. In this Chapter we will consider exchange rates high-frequency data from `TrueFX`, a data provider which allows free access to high-frequency quote data from many years upon registration.

One big hurdle that we will encounter in this Chapter is that financial markets produce lots of data which require better data-handling tools and more computer power that has been needed so far. So, when you load the dataset, hold your breath and hope that your computer doesn't crash!

6.1 Data management

Register for free to TrueFX and you will be able to download high-frequency data for a wide set of currencies. Due to the large amount of quotes, the data are provided in monthly files. For example, to load R the file for December 2013 of the USD/JPY (U.S. Dollar vs Japanese Yen) with the command `data <- read.csv('USDJPY-2013-12.csv', header=FALSE)` which took 14 seconds on a Intel Core i5 2.6 GHz and 8GB of RAM machine. Let's first have a look at the data:

```
       V1                    V2      V3      V4
1 USD/JPY 20131202 00:00:00.320 102.488 102.495
2 USD/JPY 20131202 00:00:03.172 102.489 102.496
3 USD/JPY 20131202 00:00:03.732 102.490 102.496
4 USD/JPY 20131202 00:00:04.413 102.490 102.497
5 USD/JPY 20131202 00:00:09.104 102.490 102.497
```

The first column `V1` represents the currency pair, the second column provides the date and time of the quote, and the third and forth columns represent the bid and ask quotes. The file consists of a total of 2276644 quotes that were produced in that month. Before starting to work on the data, we need to define the dataset as a time series object and it is thus important that we undestand the time format of the data provider and how to implement it in R. The first observation of the file is dated 20131202 00:00:00.320, that provides the day of the quote in the format YYYYMMDD followed by the time of the day in the format hh:mm:ss, with the seconds going from 00.000 to 61.000 that includes three decimals. The need to fraction the seconds is due to the high speed at which quotes and trades are produced in financial markets. So, the next step is to define the object `data` as a time series which requires to convert the second column of the file using the following command `datetime <- strptime(data[,2], format="%Y%m%d %H:%M:%OS")` which produces the following object[1]:

```
[1] "2013-12-02 00:00:00.320 EST" "2013-12-02 00:00:03.172 EST" "2013-12-02 00:00:03.732 EST"
[4] "2013-12-02 00:00:04.413 EST" "2013-12-02 00:00:09.104 EST" "2013-12-02 00:00:15.895 EST"
```

We are now ready to define the file as a time series object in R and we are going to use the `xts` package which is particularly suitable to handle time and dates for high-frequency data. We discard the first and second columns, and just focus on the bid/ask quotes as shown below:

```
library(xts)
rate <- as.xts(data[,3:4], order.by=datetime)
colnames(rate) <- c("bid","ask")
head(rate)
```

```
                          bid     ask
2013-12-02 00:00:00.319 102.488 102.495
2013-12-02 00:00:03.171 102.489 102.496
2013-12-02 00:00:03.732 102.490 102.496
2013-12-02 00:00:04.413 102.490 102.497
2013-12-02 00:00:09.104 102.490 102.497
2013-12-02 00:00:15.894 102.491 102.497
```

[1] To make the seconds appear with three decimals you need also to set 'op <- options(digits.secs =3)'

6.2 Aggregating frequency

We have now created the `xts` object `rate` which contains the bid and ask quotes for the USD/JPY exchange rate in the month of December 2013. One of the first tasks in analyzing high-frequency data is to sub-sample the quotes at higher frequencies, such as 30 seconds, 1 or 5 minutes. One reason for sub-sampling is that at the second and micro-second level the quote and price changes are very small and contaminated by market microstructre noise, that is, erratic movements due to behavior of market participants interacting in the market rather than, for example, informational issues. A practical benefit of subsampling is that we discard many of the 2276644 observations and are able to work with smaller datasets which are easier to plot and analyze.

As an example, let's say that we want to sub-sample the dataset at the 1 minute frequency. We achieve this by dividing the trading day in intervals of 1 minute and then pick the trade or quote that are closer in time to the 1 minute interval. The `highfrequency` package provides several functions to manage and analyze high-frequency data. The function `aggregatets()` has the purpose of aggregating the high-frequency quotes and trades to the desired frequency in seconds or minutes. In the example below, we sub-sample the mid-point between bid and ask USD/JPY exchange rate to the 1 minute frequency:

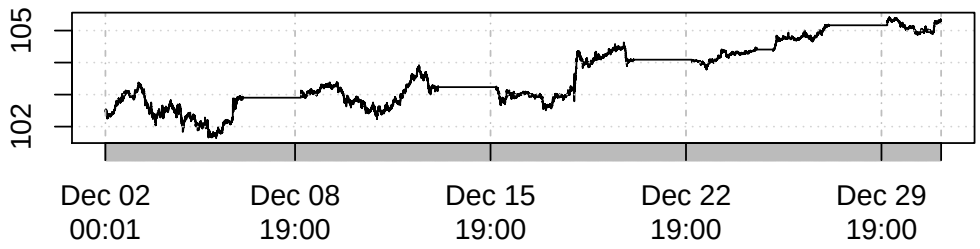
```
library(highfrequency)

midrate        <- (rate[,1]+rate[,2])/2
names(midrate) <- "midpoint"

midrate1m      <- aggregatets(midrate, on="minutes", k=1)
head(midrate1m)
```

```
                    midpoint
2013-12-02 00:01:00  102.525
2013-12-02 00:02:00  102.497
2013-12-02 00:03:00  102.505
2013-12-02 00:04:00  102.495
2013-12-02 00:05:00  102.495
2013-12-02 00:06:00  102.493
```

The first observation corresponds to the last quote before 0, 1, 0, 2, 11, 113, 1, 335, 0, EST, -18000 as we can verify from analyzing the full dataset in the neighborhood of the first minute and all other quotes are discarded:

```
                        midpoint
2013-12-02 00:00:55.415  102.525
2013-12-02 00:00:55.592  102.525
2013-12-02 00:01:01.940  102.525
2013-12-02 00:01:06.908  102.526
```

The result of the aggregation is a significant reduction of the sample size from 2276644 to 43096. We can now plot the time series of the exchange rate in December 2013 and the graph is shown below. It appears that there are periods in which the exchange rate is constant, which corresponds to weekends in which there is not much activity.

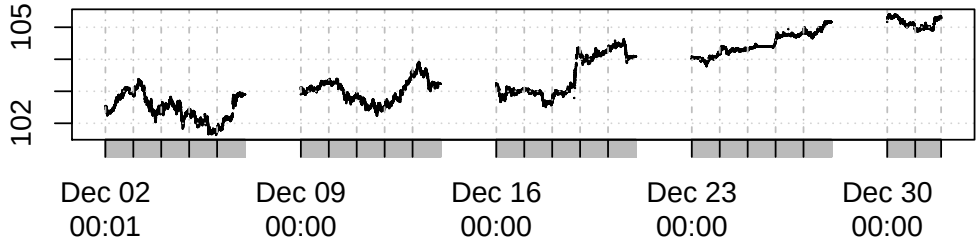


Since there is no price dynamics during weekends, we could decide to eliminate them from the sample and this can be achieved by eliminating the observations during the weekend as in the code below:

```
index <- .indexwday(midrate1m)
unique(index)
```

```
[1] 1 2 3 4 5 6 0
```

```
# keep only Monday to Friday (day 1 to 5)
midrate1m <- midrate1m[index %in% 1:5]
plot(midrate1m, main="", type="p", pch=1, cex=0.05)
```



The plot looks similar to the earlier one because the line plot connects the By eliminating the week-ends the time series has reduced from 43096 to 31576.

The next thing is to calculate the returns that represent the percentage changes of the exchange rate at the 1 minute frequency. Below we calculate the returns and some summary statistics:

```
ret1m <- 100 * diff(log(midrate1m))
summary(ret1m)
```

```
     Index                          midpoint
 Min.   :2013-12-02 00:01:00.00   Min.   :-0.385477
 1st Qu.:2013-12-09 11:34:45.00   1st Qu.:-0.005270
 Median :2013-12-16 23:08:30.00   Median : 0.000000
 Mean   :2013-12-16 09:41:51.10   Mean   : 0.000085
 3rd Qu.:2013-12-24 10:42:15.00   3rd Qu.: 0.005359
 Max.   :2013-12-31 22:16:00.00   Max.   : 0.491053
                                  NA's   :1
```

The mean and median are very close to zero with a minimum return of -0.385477% and maximum of 0.491053%. The standard deviation is 0.013521% which is expressed in terms of the 1-minute return.

6.3 Realized Volatility

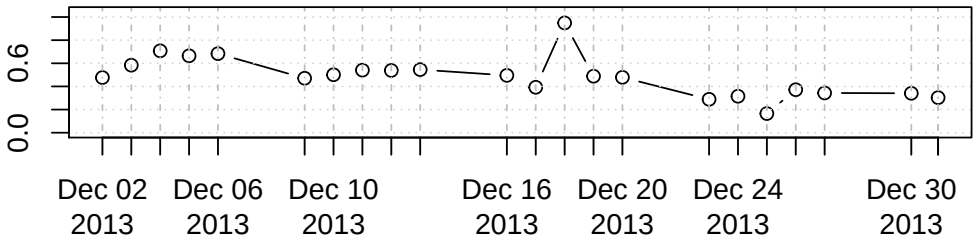
One of the important applications of high-frequency data is to calculate non-parametric measures of volatility which are alternative to the GARCH modeling approach. These measures are then used to

build a model to explain the volatility dynamics as well as for forecasting volatility. The simplest measure of **realized volatility** in day t is given by the sum of the square intra-day returns at frequency m (for a total of M returns in day t). In formula:

$$RV_t = \sum_{m=1}^{M} R_{t,m}^2$$

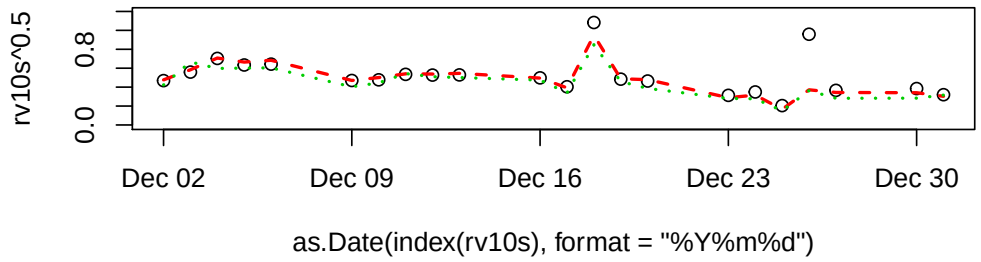
where $R_{t,m}$ represents the m-th interval at a certain frequency of day t. This can be implemented quite easily in `R` by selecting the intra-day return of each day, squaring them, and summing them up to produce the volatility measure in that day. However, we can avoid the programming effort since the `highfrequency` package provides functions to calculate the realized volatility measure at the chosen frequency. The `rCov()` function allows to calculate the realized volatility for the specificied return time series and at the appropriate frequency:

```
rv1m   <- rCov(ret1m, align.by="minutes", align.period=1)
plot(rv1m^0.5, ylim=c(0, 1.1*max(rv1m^0.5)), main="", type="b")
```



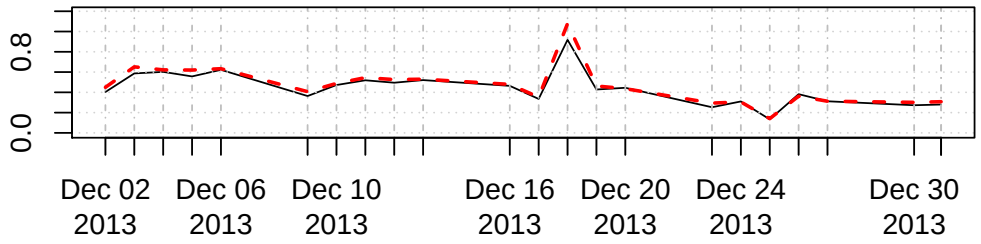
The plot shows the time series of realized volatility for the daily return of the USD/JPY in December 2013 obtained from the 1 minute returns. An important choice that has to be made concerns the time interval to use for the intra-day returns. Ideally, we would like to calculate returns over short intervals which would allow to increase the sample size of square returns of each day that the realized volatility is calculated. On the other hand, very high-frequency returns introduce microstructure noise that might bias the volatility measure and increase the variability of the volatility measure. In the example below, we compare realized volatility measures from returns at the 10 seconds, 1 minute, and 10 minutes frequency. The results show that the measure obtained from the 10-seconds return overall tracks the measures obtained with the higher frequencies, except for December 26th, 2013, when the volatility calculated on the 10-sec returns spikes up to almost 1% whilst the other two measures stay below 0.4% on that day. Overall, there is no optimal way to choose the interval to use, but for a wide class of assets many researchers have converged on using the 5-minute frequency to calculate realized volatility measures.

```
rv10s       <- rCov(ret10s, align.by="seconds", align.period=10)
rv10m       <- rCov(ret10m, align.by="minutes", align.period=10)
plot(as.Date(index(rv10s), format="%Y%m%d"),rv10s^0.5, ylim=c(0, maxy), main="", lty=1)  # black continuous
lines(as.Date(index(rv1m), format="%Y%m%d"), rv1m^0.5,  col=2, lty=2, lwd=2)             # red dashed
lines(as.Date(index(rv10m), format="%Y%m%d"), rv10m^0.5, col=3, lty=3, lwd=2)             # green dots
```



The realized volatility measure RV_t is criticized because it is not robust to microstructure noise and to outliers of jumps, as it is shown in the previous graph. An alternative to the RV_t estimator is the median RV estimator which consists of the taking the median of the intra-day returns

```
medrv5m  <- medRV(ret5m, align.by="minutes", align.period=5)
plot(medrv5m^0.5, ylim=c(0,maxy), main="",lty=1)
lines(rv5m^0.5, col=2, lty=2, lwd=2)
```



6.4 Modeling realized volatility

AR(1) model for realized volatility:

$$RV_t = \beta_0 + \beta_1 * RV_{t-1} + \epsilon_t$$

Heterogeneous AR HAR(1) for realized volatlity:

$$RV_t = \beta_0 + \beta_1 * RV_{t-1} + \beta_2 * \hat{RV}^5_{t-1} + \beta_3 * \hat{RV}^{22}_{t-1} + \epsilon_t$$

where the 5 and 22 day moving averages are defined as $\hat{RV}^5_{t-1} = \sum_{j=1}^{5} R_{t-j}/5$ and $\hat{RV}^{22}_{t-1} = \sum_{j=1}^{22} R_{t-j}/22$.

To be completed.

Chapter 7

Measuring Financial Risk

The emergence of derivatives in the 1980s and several bankruptcies due to their misuse led the financial industry and regulators to develop rules to measure the potential portfolio losses in adverse market conditions. One of the goals of these efforts was to provide banks with a framework to determine the capital needed to survive during periods of economic and financial distress. *Measuring* risk is a necessary condition for *managing* risk: institutions need to be able to quantify the amount of risk they face to effectively elaborate a strategy to mitigate the consequences of extreme negative events. The aim of this Chapter is to discuss the application of some methods that have been proposed to measure financial risk, in particular market risk which represents the potential losses deriving from adverse movements of equity or currency markets, interest rates etc.

7.1 Value-at-Risk (VaR)

The `VaR` methodology was introduced in the early 1990s by the investment bank J.P. Morgan to measure the minimum portfolio loss that an institution might face if an unlikely adverse event occurred at a certain time horizon. Let's define the profit/loss of a financial institution in day $t+1$ by $R_{t+1} = 100 * ln(W_{t+1}/W_t)$, where W_{t+1} is the portfolio value in day $t+1$. Then *Value-at-Risk* (`VaR`) at $100(1-\alpha)\%$ is defined as

$$P(R_{t+1} \leq VaR_{t+1}^{1-\alpha}) = \alpha$$

where the typical values of α are 0.01 and 0.05. In practice, $VaR_{t+1}^{1-\alpha}$ is calculated every day and for an horizon of 10 days (2 trading weeks). If $VaR_{t+1}^{1-\alpha}$ is expressed in percentage return it can be easily transformed into dollars by multiplying the portfolio value in day t (denoted by W_t) with the expected loss, that is, $VaR_{t+1}^{1-\alpha} = W_t * (exp(VaR_{t+1}^{1-\alpha}/100) - 1)$. From a statistical point view, 99% `VaR` represents the 1% quantile, that is, the value of R_{t+1} such that there is only 1% probability that the random variable takes a value smaller or equal to that value. The graphs below shows the Probability Density Function (`PDF`) and the Cumulative Distribution Function (`CDF`). Risk calculation is devoted to the left tail of the return distribution since it represents large and negative outcomes for the portfolio of financial institutions. This is the reason why the profession has devoted a lot of energy to make sure that the left tail, rather than the complete distribution, is appropriately specified since a *poor* model for the left tail

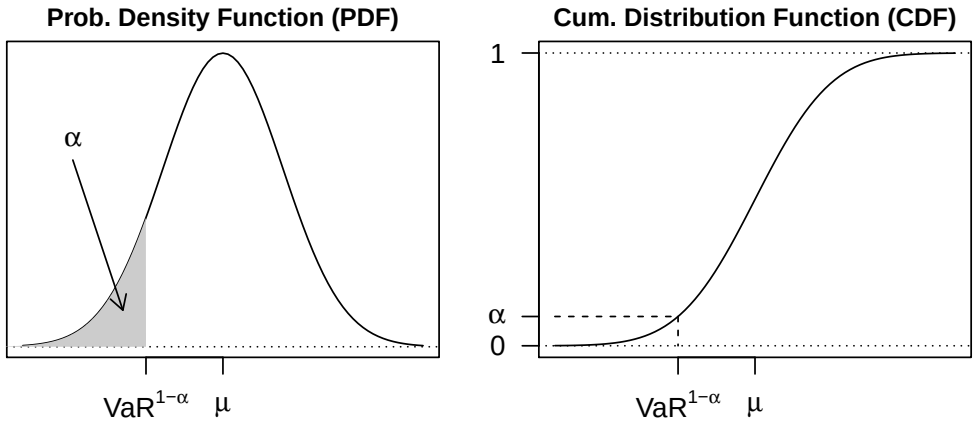


Figure 7.1: The Probability Density Function (PDF) and Cumulative Distribution Function (CDF) for a normal distribution with mean μ and variance σ^2. Value at Risk ($VaR^{1-\alpha}$) at $1-\alpha$ level represents the α quantile of the distribution.

implies *poor* risk estimates (poor in a sense that will become clear in the backtesting section).

7.1.1 `VaR` assuming normality

Calculating `VaR` requires making an assumption about the distribution of the profit/loss of the institution. The simplest and most familiar assumption we can introduce is that $R_{t+1} \sim N(\mu, \sigma^2)$, where μ represents the mean of the distribution and σ^2 its variance. The assumption is equivalent to assume that the profit/loss follows the model $R_{t+1} = \mu + \sigma\epsilon_{t+1}$, with $\epsilon_{t+1} \sim N(0,1)$. The quantiles for this model are given by $\mu + z_\alpha\sigma$, where α is the probability level and z_α represents the α quantile of the standard normal distribution. The typical levels of α for `VaR` calculations are 1 and 5% and the corresponding z_α are -1.64 and -2.33, respectively. Hence, 99% `VaR` under normality is obtained as follows:

$$VaR_{t+1}^{0.99} = \mu - 2.33 * \sigma$$

As an example, assume that an institution is holding a portfolio that replicates the S&P 500 Index and wants to calculate the 99% `VaR` for this position. If we assume that returns are normally distributed, then to calculate $VaR_{t+1}^{0.99}$ we need to estimate the expected daily return of the portfolio (i.e., μ) and its expected volatility (i.e., σ). Let's assume that we believe that the distribution is approximately constant over time so that we can estimate the mean and standard deviation of the returns over a long period of time. In the illustration below we use the S&P 500 time series that was used in the volatility chapter that consists of daily returns from 1990 to 2017 (6973 observations).

```
mu    = mean(sp500daily)
sigma = sd(sp500daily)
var   = mu + qnorm(0.01) * sigma
```

```
[1] -2.557
```

The 99% `VaR` is -2.557% and represents the minimum loss of holding the S&P500 for the following day with 1% (or smaller) probability. If we use a shorter estimation window of one year (252 observations),

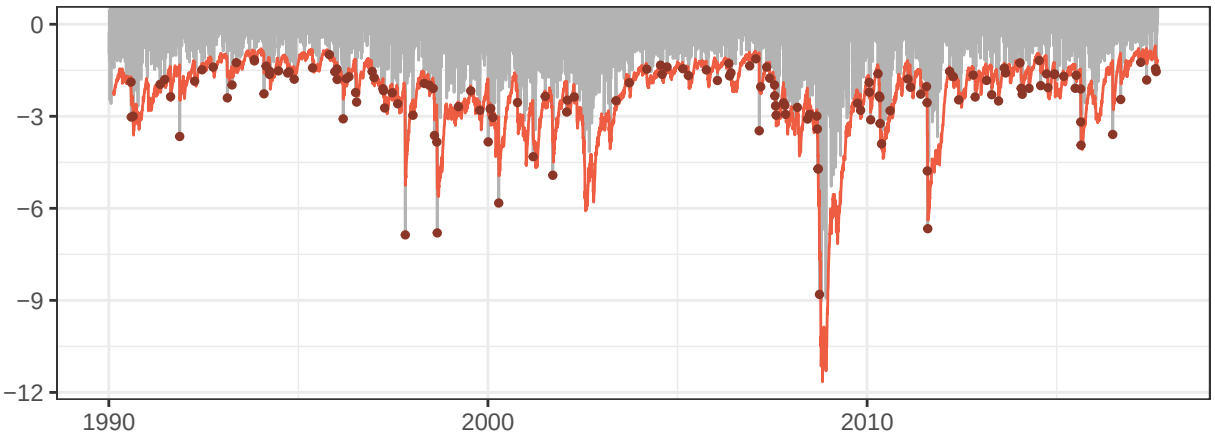


Figure 7.2: 99% VaR for the daily returns of the SP 500 Index. The dots represent the days in which the returns were smaller than VaR, also called violations or hits.

the VaR estimation would be -1.74%. The difference between the two `VaR` estimates is quite remarkable given that we only changed the size of the estimation window. The standard deviation declines from 1.114% in the full sample to 0.77% in the shorter sample, whilst the mean changes from 0.034% to 0.05%. As discussed in the volatility modeling Chapter, it is extremely important to account for time variation in the distribution of financial returns if the interest is to estimate `VaR` at short horizons (e.g., a few days ahead).

7.1.2 Time-varying `VaR`

So far we assumed that the mean and standard deviation of the return distribution are constant and represent the long-run distribution of the variable. However, this might not be the best way to predict the distribution of the profit/loss at very short horizons (e.g., 1 to 10 days ahead) if the return volatility changes over time. In particular, in the volatility chapter we discussed the evidence that the volatility of financial returns changes over time and introduced models to account for this behavior. We can model the conditional distribution of the returns in day $t+1$ by assuming that both μ_{t+1} and σ_{t+1} are time-varying conditional on the information available in day t. Another decision that we need to make is the distribution of the errors. We can assume, as above, that the errors are normally distributed so that 99% `VaR` is calculated as:

$$VaR_{t+1}^{0.99} = \mu_{t+1} - 2.33 * \sigma_{t+1}$$

where the expected return μ_{t+1} can be either constant (i.e. $\mu_{t+1} = \mu$), or an AR(1) process ($\mu_{t+1} = \phi_0 + \phi_1 * R_t$), and the conditional variance can be modeled as MA, EMA, or with a GARCH-type model. In the example below, I assume that the conditional mean is constant (and equal to the sample mean) and model the conditional variance of the demeaned returns as an EMA with parameter $\lambda = 0.06$:

```
library(TTR)
mu       <- mean(sp500daily)
sigmaEMA <- EMA((sp500daily-mu)^2, ratio=0.06)^0.5
var      <- mu + qnorm(0.01) * sigmaEMA
```

The VaR time series shows in Figure 7.2 inherits the characteristic of the volatility of alternating between calm periods of low volatility and risk, and other periods of increased uncertainty and potentially large losses. For this example, we find that in 2.108% of the 6973 days the return was smaller relative to VaR. Since we calculated VaR at 99% we expected to experience only 1% of days with violations and the difference between the sample and population value might indicate that the risk model might be misspecified. We will discuss the properties of the violations for a risk model in the section devoted to backtesting.

7.1.3 Expected Shortfall (ES)

VaR represents the maximum (minimum) loss that is expected with, e.g., 99% (1%) probability. However, it can be criticized on the ground that it does not convey the information of the potential loss that is expected if indeed an extreme event (only likely 1% of less) occurs. For example, a VaR of -5.52% provides no information on how large the portfolio loss is expected to be if the portfolio return will happen to be smaller than VaR. That is, how large do we expect the loss be in case VaR is violated? A risk measure that quantifies this potential loss is Expected Shortfall (ES) which is defined as

$$ES_{t+1}^{1-\alpha} = E(R_{t+1}|R_{t+1} \leq VaR_{t+1}^{1-\alpha})$$

that is, the expected portfolio return conditional on being on a day in which the return is smaller than VaR. This risk measure focuses its attention on the left tail of the distribution and it is highly dependent on the shape of the distribution in that area, while it neglects all other parts of the distribution.

An analytical formula for ES is available if we assume that returns are normally distributed. In particular, if $R_{t+1} = \sigma_{t+1}\epsilon_{t+1}$ with $\epsilon_{t+1} \sim N(0,1)$, then VaR is calculated as $VaR_{t+1}^{1-\alpha} = z_\alpha \sigma_{t+1}$. The conditioning event is that the return in the following day is smaller than VaR and the probability of this event happening is α, e.g. 0.01. We then need to calculate the expected value of R_{t+1} over the interval from minus infinity to R_{t+1} which corresponds to a truncated normal distribution with density function given by $f(R_{t+1}|R_{t+1} \leq VaR_{t+1}^{1-\alpha}) = \phi(R_{t+1})/\Phi(VaR_{t+1}^{1-\alpha}/\sigma_{t+1})$, where $\phi(\cdot)$ and $\Phi(\cdot)$ represent the PDF and the CDF of the normal distribution (i.e., $\Phi(VaR_{t+1}^{1-\alpha}/\sigma_{t+1}) = \Phi(z_\alpha) = \alpha$). We can thus express ES as

$$ES_{t+1}^{1-\alpha} = -\sigma_{t+1}\frac{\phi(z_\alpha)}{\alpha}$$

where z_α is equal to -2.33 and -1.64 for α equal to 0.01 and 0.05, respectively. If we are calculating VaR at 99% so that α is equal to 0.01 then ES is equal to

$$ES_{t+1}^{0.99} = -\sigma_{t+1}\frac{\phi(-2.33)}{0.01} = -2.64\sigma_{t+1}$$

where the value -2.64 can be obtained in R typing the command `(2*pi)^(-0.5) * exp(-(2.33^2)/2) / 0.01` or using the function `dnorm(-2.33) / 0.01`. If $\alpha = 0.05$ then the constant to calculate ES is -2.08 instead of -1.64 for VaR. Hence, ES leads to more conservative risk estimates since the expected loss in a day in which VaR is exceeded is always larger than VaR. This is shown in Figure 7.3 where the difference between VaR and ES is plotted as a function of $1 - \alpha$:

```
sigma = 1
alpha = seq(0.001, 0.05, by=0.001)
ES    =  - dnorm(qnorm(alpha)) / alpha * sigma
VaR   = qnorm(alpha) * sigma
```

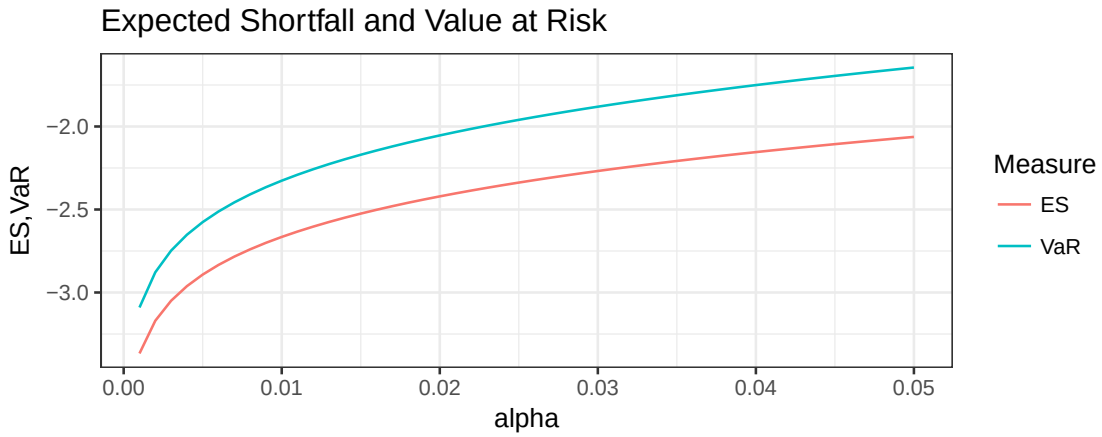


Figure 7.3: Comparison of the ES and VaR risk measures under the assumption of normally distributed profit/loss at different values of α.

7.1.4 $\sqrt{K}$ rule

The Basel Accords require `VaR` to be calculated at a horizon of 10 days and for a risk level of 99%. In addition, the Accords allow financial institution to *scale up* the 1-day `VaR` to the 10 day horizon by multiplying it by $\sqrt{10}$. Why $\sqrt{10}$? Under what conditions is the `VaR` for the cumulative returns over 10 days, denoted by $VaR_{t+1:t+10}$, equal to $\sqrt{10} * VaR_{t+1}$?

In day t we are interested in calculating the risk of holding the portfolio over a horizon of K days, that is, assuming that we can liquidate the portfolio only on the Kth day. Regulators require banks to use $K = 10$ that corresponds to two trading weeks. To calculate risk of holding the portfolio in the next K days we need to obtain the distribution of the sum of K daily returns or cumulative return, denoted by $R_{t+1:t+k}$, which is given by $\sum_{k=1}^{K} R_{t+k} = R_{t+1} + \cdots + R_{t+K}$, where R_{t+k} is the return in day $t + k$. If we assume that these daily returns are *independent and identically distributed* (*i.i.d.*) with mean μ and variance σ^2, then the expected value of the cumulative return is

$$E\left(\sum_{k=1}^{K} R_{t+k}\right) = \sum_{k=1}^{K} \mu = K\mu$$

and its variance is

$$Var\left(\sum_{k=1}^{K} R_{t+k}\right) = \sum_{k=1}^{K} \sigma^2 = K\sigma^2$$

so that the standard deviation of the cumulative return is equal to $\sqrt{K}\sigma$. If we maintain the normality assumption that we introduced earlier, than the 99% VaR of $R_{t+1:t+K}$ is given by

$$VaR^{1-\alpha}_{t+1:t+K} = K\mu - 2.33\sqrt{K}\sigma$$

In this formula, the mean and standard deviation are estimated on daily returns and they are then scaled up to horizon K.

This result relies on the assumption that returns are serially independent which allows us to set all covariances between returns in different days equal to zero in the formula for the variance. The empirical

evidence from the ACF of daily returns indicates that this assumption is likely to be accurate most of the time, although in times of market booms or busts returns could be, temporarily, positively correlated. What would be the effect of positive correlation in returns on `VaR`? The first-order covariance can be expressed in terms of correlation as $\rho\sigma^2$, with ρ representing the first order serial correlation. To keep things simple, assume that we are interested in calculating `VaR` for the two-day return, that is, $K = 2$. The variance of the cumulative return is $Var(R_{t+1} + R_{t+2})$ which is equal to $Var(R_{t+1}) + Var(R_{t+2}) + 2Cov(R_{t+1}, R_{t+2}) = \sigma^2 + \sigma^2 + 2\rho\sigma^2$. This can be re-written as $2\sigma^2(1+\rho)$ which shows that in the presence of positive correlation ρ the cumulative return becomes riskier, relative to the independent case, since $2\sigma^2(1+\rho) > 2\sigma^2$. The Value-at-Risk for the two-day return is then $VaR^{0.99}_{t+1:t+2} = 2\mu - 2.33 * \sigma * \sqrt{2}\sqrt{1+\rho}$, which is smaller relative to the `VaR` assuming independence that is given by $2\mu - 2.33 * \sigma * \sqrt{2}$. Hence, neglecting positive correlation in returns leads to underestimating risk and the potential portfolio loss deriving from an extreme (negative) market movement.

7.1.5 `VaR` assuming non-normality

One of the stylized facts of financial returns is that the empirical frequency of large positive/negative returns is higher relative to the frequency we would expect if returns were normally distributed. This finding of *fat tails* in the return distribution can be partly explained by time-varying volatility: the extreme returns are the outcome of a regime of high volatility that occurs occasionally, although most of the time returns are in a calm period with low volatility. The effect of mixing periods of high and low volatility is that the (unconditional) volatility estimate based on the full sample overestimates uncertainty when volatility is low, and underestimates it in turbulent times. This can be illustrated with a simple example: assume that in 15% of days volatility is high and equal to 5% while in the remaining 85% of the days it is low and equal to 0.5%. Average volatility based on all observations is then 0.15 * 5 + 0.85 * 0.5 = 1.175%. This implies that in days of high volatility, the returns appear *large* relative to a standard deviation of 1.175% although they are *normal* considering that they belong to the high volatility regimes with a standard deviation of 5. On the other hand, the bulk of returns belong to the low volatility regime which looks like a concentration of small returns relative to the average volatility of 1.175%.

To evaluate the evidence of deviation from normality, we can calculate the sample skewness and excess kurtosis for the standardized residuals and compare them to the their theoretical value of 0. The skewness of the returns standardized by the EMA volatility estimate is -0.177 and the excess kurtosis is estimated equal to 0.125. Both statistics indicate that the standardized residuals are not close to normality due to the presence of large residuals (kurtosis) in the left tail of the distribution (skewness). This evidence is very important for risk measurement which is mostly concerned with the far left of the distribution: time-variation in volatility is the biggest factor driving non-normality in returns so that when we calculate risk measures we are confident that a combination of the normal distribution and a dynamic method to forecast volatility should provide reasonably accurate `VaR` estimates. However, we still might want to account for the non-normality in the standardized returns ϵ_t and we will consider two possible approaches in this Section.

One approach to relax the assumption of normally distributed errors is the *Cornish-Fisher approximation* which consists of performing a Taylor expansion of the normal distribution around its mean. This has the effect of producing a distribution which is a function of skewness and kurtosis. We skip the mathematical

details of the derivation and focus on the VaR calculation when this approximation is adopted. If we assume the mean μ is equal to zero, the 99% VaR for normally distributed returns is calculated as $-2.33 * \sigma_{t+1}$ or, more generally, by $z_\alpha \sigma_{t+1}$ for $100(1-\alpha)\%$ VaR, where z_α represents the $1-\alpha$-quantile of the standard normal distribution. With the Cornish-Fisher (CF) approximation VaR is calculated in a similar manner, that is, $VaR_{t+1}^{1-\alpha} = z_\alpha^{CF} \sigma_{t+1}$, with the quantile z_α^{CF} calculated as follows:

$$z_\alpha^{CF} = z_\alpha + \frac{SK}{6}\left[z_\alpha^2 - 1\right] + \frac{EK}{24}\left[z_\alpha^3 - 3z_\alpha\right] + \frac{SK^2}{36}\left[2z_\alpha^5 - 5z_\alpha\right]$$

where SK and EK represent the skewness and excess kurtosis, respectively. If the data are normally distributed then $SK = EK = 0$ so that $z_\alpha^{CF} = z_\alpha$. However, in case the distribution is asymmetric and/or with fat tails the effect is that $z_\alpha^{CF} \leq z_\alpha$. In practice, we estimate the skewness and the excess kurtosis from the sample and use those values to calculate the quantile for VaR calculations. Figure 7.4 we show the quantile $z_{0.01}^{CF}$ and its relationship to $z_{0.01}$ as a function of the skewness and excess kurtosis parameters. The left plot shows the effect of the skewness parameter on the quantile, while holding the excess kurtosis equal to zero. Instead, the plot on the right shows the effect of increasing values of excess kurtosis, while the skewness parameter is kept constant and equal to zero. As expected, the $z_{0.01}^{CF}$ is smaller than $z_{0.01} = -2.33$ and it is interesting to notice that negative skewness increases the (absolute) value of $z_{0.01}^{CF}$ more than positive skewness of the same magnitude. This is due to the fact that negative skewness implies a higher probability of large negative returns compared to large positive returns. The effect on VAR of accounting for asymmetry and fat tails in the data is thus to provide more conservative risk measures.

```
alpha = 0.01
# case 1: skewness holding excess kurtosis equal to 0
EK = 0; SK = seq(-1, 1, by=0.05)
z = qnorm(alpha)
zCF = z + (SK/6) * (z^2 - 1) + (EK/24) * (z^3 - 3 * z) +
  (SK^2/36) * (2*z^5 - 5*z)
q1 <- ggplot() + geom_line(aes(SK,zCF), color="seagreen4") +
      labs(x="Skewness", y=expression(z[alpha])) +
      geom_hline(yintercept=rep(z, length(SK)), linetype="dashed", size=1.2) + theme_bw() +
      coord_cartesian(y=c(-7.1, -1.9))
# case 2: kurtosis holding skewness equal to 0
EK1 = seq(0, 10,by=0.1); SK1 = 0
```

An alternative approach to allow for non-normality is to make a different distributional assumption for ϵ_{t+1} that captures the fat-tailness in the data. A distribution that is often considered is the t distribution with a small number of degrees of freedom. Since the t distribution assigns more probability to events in the tail of the distribution, it will provide more conservative risk estimates relative to the normal distribution[1]. Figure 7.5 shows the t distribution for 4, 10, and ∞ degree-of-freedom, while the plot on the right zooms on the shape of the left tail of these distributions. It is clear that the smaller the d.o.f used the more likely are extreme events relative to the standard normal distribution ($d.o.f. = \infty$). So, also this approach delivers more conservative risk measures relative to the normal distribution since it assigns higher probability to extreme events.

[1]The standard normal distribution is a t distribution with ∞ degrees of freedom. In practice, for 100 or more degrees of freedom the quantiles of the t and standard normal are almost identical.

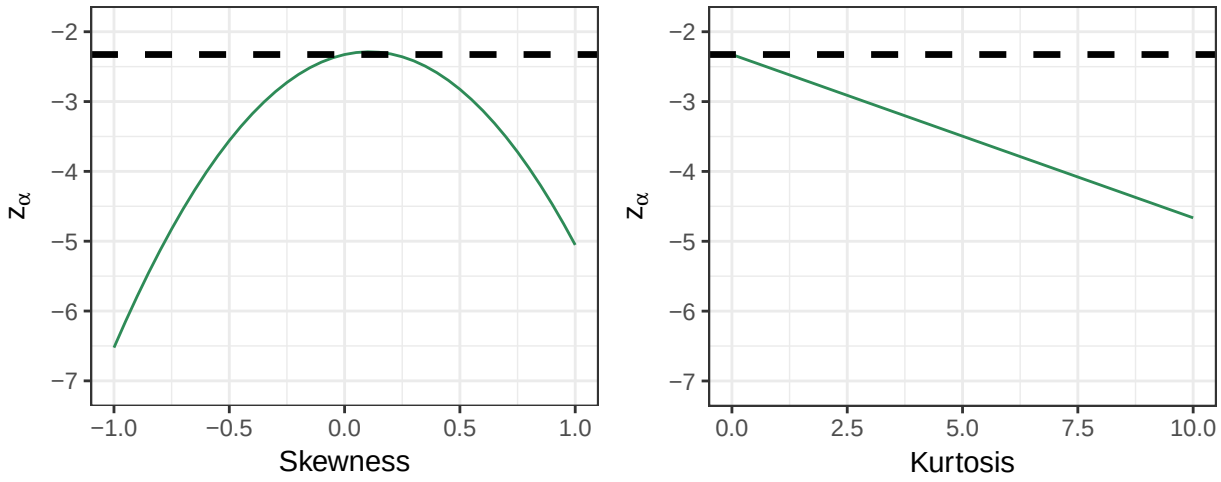


Figure 7.4: The effect of skewness and kurtosis on the critical value used in the Cornish-Fisher approximation approach.

```
q1 <- ggplot(data.frame(x = c(-8, 8)), aes(x = x)) +
        stat_function(fun = dnorm, aes(colour = "N(0,1)")) +
        stat_function(fun = dt, args = list(df=4),  aes(colour = "t(4)")) +
        stat_function(fun = dt, args = list(df=10),  aes(colour = "t(10)")) +
        theme_bw() + labs(x = NULL, y= NULL) + coord_cartesian(x=c(-3,3)) +
        scale_colour_manual("Distribution", values = c("seagreen3", "steelblue3", "tomato3")) +
        theme(legend.position="bottom")
q2 <- q1 + coord_cartesian(x=c(-8, -3), y=c(0,0.03))
```

To be able to use the t distribution for risk calculation we need to estimate the degree-of-freedom parameter, denoted by d. In the context of a GARCH volatility model this can be easily done by considering d as an additional parameter to be estimated by maximizing the likelihood function based on the assumption that the ϵ_{t+1} shocks follow a t_d distribution. A simple alternative approach to estimate the degree-of-freedom parameter d exploits the fact that the excess kurtosis of a t_d distribution is equal to $EK = 6/(d-4)$ (for $d > 4$) which is only a function of the parameter d. Thus, based on the sample excess kurtosis we can then back out an estimate of d. The steps are as follows:

1. estimate by ML the GARCH model assuming that the errors are normally distributed (or using EMA)
2. calculate the standardized residuals as $\epsilon_{t+1} = R_{t+1}/\sigma_{t+1}$
3. estimate the excess kurtosis of the standardized residuals and obtain d as $d = 6/EK + 4$

For the standardized returns of the S&P 500 the sample excess kurtosis is equal to 2.337 so that the estimate of d is equal to approximately 7 which indicates the need for a fat-tailed distribution. In practice, it would be advisable to estimate the parameter d jointly with the remaining parameters of the volatility model, rather than separately. Still, this simple approach provides a starting point to evaluate the usefulness of fat tailed distributions in risk measurement.

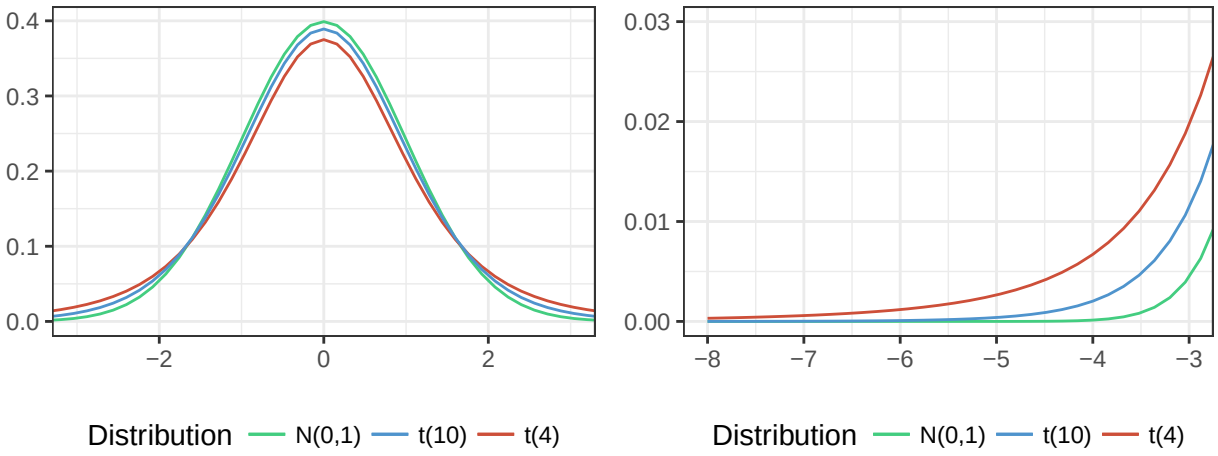


Figure 7.5: Comparison of the standard normal distribution and t distribution with 4 and 10 degree-of-freedom. The right plot zooms on the left tail of the three distributions.

7.2 Historical Simulation (HS)

So far we discussed methods to calculate `VaR` that rely on choosing a probability distribution and a volatility model. A distributional assumption we made is that returns are normal, which we later relaxed by introducing the Cornish-Fisher approximation and the t distribution. In terms of the volatility model, we considered several possible specifications such as EMA, GARCH, and GJR-GARCH, which we can compare using selection criteria.

An alternative approach that avoids making any assumption and *let the data speak for itself* is represented by **Historical Simulation** (`HS`). `HS` consists of calculating the empirical quantile of returns at $100 * \alpha\%$ level based on the most recent `M` days. This approach is called *non-parametric* because it estimates 99% `VaR` to be the value such that 1% of the more recent `M` returns are smaller than, without introducing assumptions on the return distribution and the volatility structure. In addition to its flexibility, `HS` is very easy to compute since it does not require the estimation of any parameter for the volatility model and the distribution. However, there are also a few difficulties in using `HS` to calculate risk measures. One issue is the choice of the estimation window size `M`. Practitioners often use values between `M=250` and `1000`, but, similarly to the choice of smoothing in MA and EMA, this is an ad hoc value that has been validated by experience rather than being optimally selected based on a criterion. Another complication is that `HS` applied to daily returns provides a `VaR` measure at the one day horizon which, for regulatory purposes, should then be converted to a 10-day horizon. What is typically done is to apply the $\sqrt{10}$ rule discussed before, although it does not have much theoretical justification in the context of `HS` since we are not actually making any assumption about the return distribution. An alternative would be to calculate `VaR` as the 1% quantile of the (non-overlapping) cumulative return instead of the daily return. However, this would imply a much smaller sample size, in particular for small `M`.

The implementation in `R` is quite straightforward and shown below. The function `quantile()` calculates the $1 - \alpha$ quantile for a return series which we can combine with the function `rollapply()` from package `zoo` to apply it recursively to a rolling window of size `M`. For example, the command `rollapply(sp500daily, 250, quantile, probs=alpha, align="right")` calculates the function

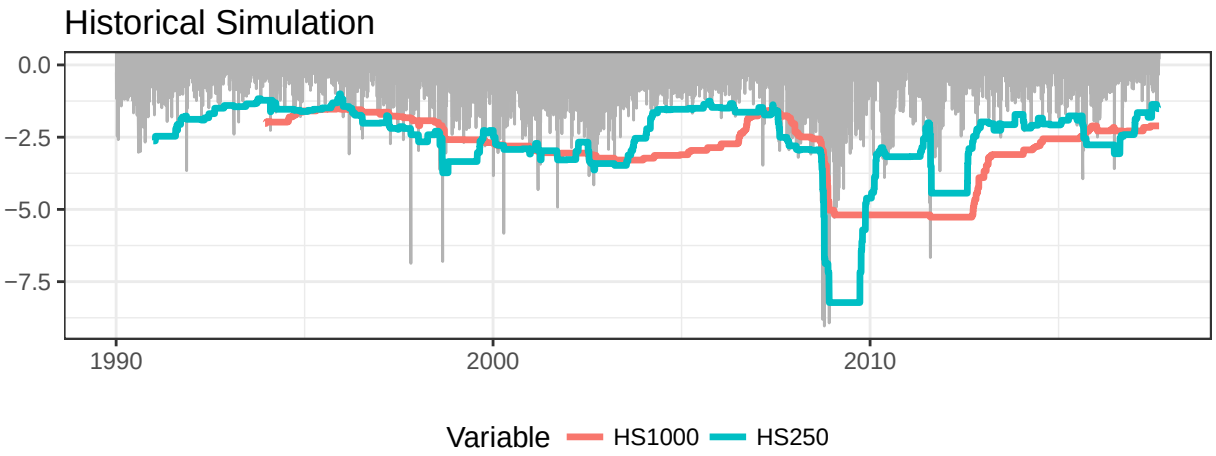


Figure 7.6: Returns and 99% VaR for the Historical Simulation (HS) method with parameter M set to 250 and 1000 days.

`quantile` (for `probs=alpha` and `alpha=0.01`) for the `sp500daily` time series with the first `VaR` forecast for day `251` until the end of the sample. Figure 7.6 compares `VaR` calculated with the `HS` method for an estimation window of `250` and `1000` days. The shorter estimation window makes the `VaR` more sensitive to market events as opposed to `M=1000` that changes very slowly. A characteristic of `HS` for `M=1000` is that it is constant for long periods of time, even though volatility might have significantly decreased for several months.

```
M1 = 250
M2 = 1000
alpha = 0.01
hs1 <- rollapply(sp500daily, M1, quantile, probs=alpha, align="right")
hs2 <- rollapply(sp500daily, M2, quantile, probs=alpha, align="right")

mydata <- merge(hs1, hs2) %>% fortify %>%
  tidyr::gather(Variable, Value, -Index)

ggplot() + geom_line(aes(index(sp500daily), sp500daily), color="gray70") +
  geom_line(data=mydata, aes(Index, Value, color=Variable), size=1.2) +
  coord_cartesian(y=c(min(sp500daily, na.rm=T), 0)) + theme_bw() +
  labs(x=NULL, y=NULL, title="Historical Simulation") + theme(legend.position="bottom")
```

How good is the risk model? One metric that is used as an indicator of goodness of the risk model is the frequency of returns that are smaller than `VaR`, that is, $\sum_{t=1}^{T} I(R_{t+1} \leq VaR_{t+1})/T$, where T represents the total number of days and $I(\cdot)$ denotes the indicator function. A good risk model should have the frequency of *violations* or *hits* close to the level $1-\alpha$ for which `VaR` is calculated. If we are calculating `VaR` at 99% then we would expect that the model shows (approximately) 1% violations. For `HS` above, we have that for `VaR` calculated on the 250 window there are 1.334% of days that represent violations. For the longer window of M=1000 the violations of `VaR` represents 1.276% of the sample. For `HS`, we thus find that they are larger than expected and in the backtesting Section we will evaluate if the difference

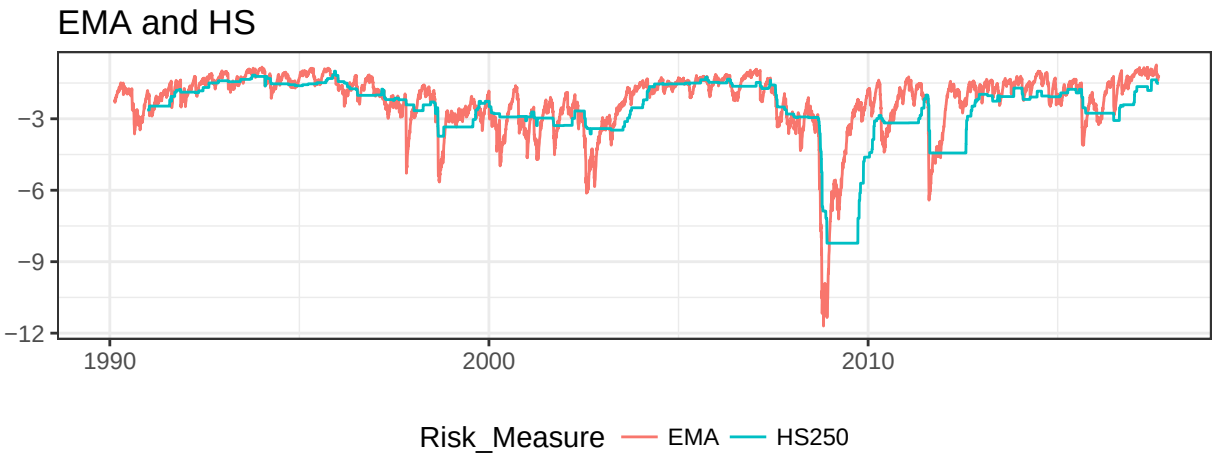


Figure 7.7: 99% VaR for the SP 500 based on the HS(250) and EMA(0.06) methods.

from the theoretical value of 1% is statistically significant.

In Figure 7.7, the `VaR` calculated from `HS` is compared to the EMA forecast. Although the `VaR` level and the dynamics is quite similar, `HS` changes less rapidly and remains constant for long periods of time, in particular in periods of rapid changes in market volatility.

```
library(TTR)
ema <- - 2.33 * EMA(sp500daily^2, ratio=0.06)^0.5
```

7.3 Simulation Methods

We discussed several approaches that can be used to calculate `VaR` based on parametric assumptions or that are purely non-parametric in nature. In most cases we demonstrated the technique by forecasting `Value-at-Risk` at the 1-day horizon. However, for regulatory purposes the risk measure should be calculated at a horizon of 10 days. An approach that is widely used by practitioners and that is also allowed by regulators is to scale up the 1-day `VaR` to 10-day by multiplying it by $\sqrt{10}$.

An alternative is to use simulation methods that generate artificial future returns based on the risk model. The model makes assumptions about the volatility model, and the distribution of the error term. Using simulations we are able to produce a large number of future possible paths for the returns that are conditional on the current day. In addition, it becomes very easy to obtain the distribution of cumulative returns by summing daily simulated returns along a path. We will consider two popular approaches that differ in the way simulated shocks are generated: **Monte Carlo Simulation** (`MC`) consists of iterating the volatility model based on shocks that are simulated from a certain distribution (normal, t or something else), and **Filtered Historical Simulation** (`FHS`) that assumes the shocks are equal to the standardized returns and takes random sample of those values. The difference is that `FHS` does not make a parametric assumption for the ϵ_{t+1} (similarly to `HS`), while `MC` does rely on such assumption.

7.3.1 Monte Carlo Simulation

At the closing of January 22, 2007 and September 29, 2008 the risk manager needs to forecast the distribution of returns from 1 up to K days ahead[2], including the cumulative or multi-period returns over these horizons. The model for returns is $R_{t+1} = \sigma_{t+1}\epsilon_{t+1}$, where σ_{t+1} is the volatility forecast based on the information available up to that day. The ϵ_{t+1} are the random shocks that the risk manager assumes are normally distributed with mean 0 and variance 1. A Monte Carlo simulation of the future distribution of the portfolio returns requires the following steps:

1. *Estimation of the volatility model*: we assume that σ_{t+1} follows a GARCH(1,1) model and estimate the model by ML using all observations available up to and including that day. Below is the code to estimate the model using the `rugarch` package. The volatility forecast for the following day is 0.483% which is significantly lower relative to the sample standard deviation from January 1990 to January 2007 of 0.993%. To use some terminology introduced earlier, the conditional forecast (0.483%) is lower relative to the unconditional forecast (0.993%).

```
lowvol   = as.Date("2007-01-22")
spec     = ugarchspec(variance.model=list(model="sGARCH",garchOrder=c(1,1)),
                mean.model=list(armaOrder=c(0,0), include.mean=FALSE))
fitgarch = ugarchfit(spec = spec, data = window(sp500daily, end=lowvol))
gcoef <- coef(fitgarch)
sigma <- ugarchforecast(fitgarch, n.ahead=1)@forecast$sigmaFor
```

```
 omega alpha1  beta1
 0.005  0.054  0.942
    2007-01-22
T+1      0.483
```

2. The next step consists of simulating a large number of return paths, say S, that are consistent with the model assumption that $R_{t+1} = \sigma_{t+1}\epsilon_{t+1}$. This is done by generating values for the error term ϵ_{t+1} from a certain distribution and multiply these values by the forecast $\sigma_{t+1} = 0.483$. Generating random variables can be easily done in `R` using the command `rnorm()` which returns random values from the standard normal distribution (and `rt()` does the same for the t distribution). Denote by $\epsilon_{s,t+1}$ the s-th simulated value of the shock (for $s = 1, \cdots, S$), then the s-th simulated value of the return is produced by multiplying σ_{t+1} and $\epsilon_{s,t+1}$, that is, $R_{s,t+1} = \sigma_{t+1}\epsilon_{s,t+1}$.

3. The next step is to use the simulate returns $R_{s,t+1}$ to predict volatility next period, denoted by $\sigma_{s,t+2}$. Since we have assumed a GARCH specification the volatility forecast is obtained by $(\sigma_{s,t+2})^2 = \omega + \alpha R^2_{s,t+1} + \beta\sigma^2_{t+1}$ and the (simulated) returns at time $t+2$ are obtained as $R_{s,t+2} = \sigma_{s,t+2}\epsilon_{s,t+2}$, where $\epsilon_{s,t+2}$ represent a new set of simulated values for the shocks in day $t+2$.

4. Continue the iteration to calculate $\sigma_{s,t+k}$ and $R_{s,t+k}$ for $k = 1, \cdots, K$. The cumulative or multi-period return between $t+1$ and $t+k$ is then obtained as $R_{s,t+1:t+K} = \sum_{j=1}^{k} R_{s,t+j}$.

The code below shows how a MC simulation can be performed and Figure 7.8 shows the quantiles of $R_{s,t+k}$ as a function of k on the left plot and the quantiles of $\sigma_{s,t+k}$ on the right plot (from 0.10 to 0.90 at intervals of 0.10 in addition to 0.05 and 0.95). The volatility is branching out from $\sigma_{t+1} = 0.483$ and its uncertainty increases with the forecast horizon. The green dashed line represents the average (over the

[2]The regulator requires $K = 10$ but in the following analysis we consider $K = 250$ to evaluate the distributional characteristics of `VaR` at longer horizons.

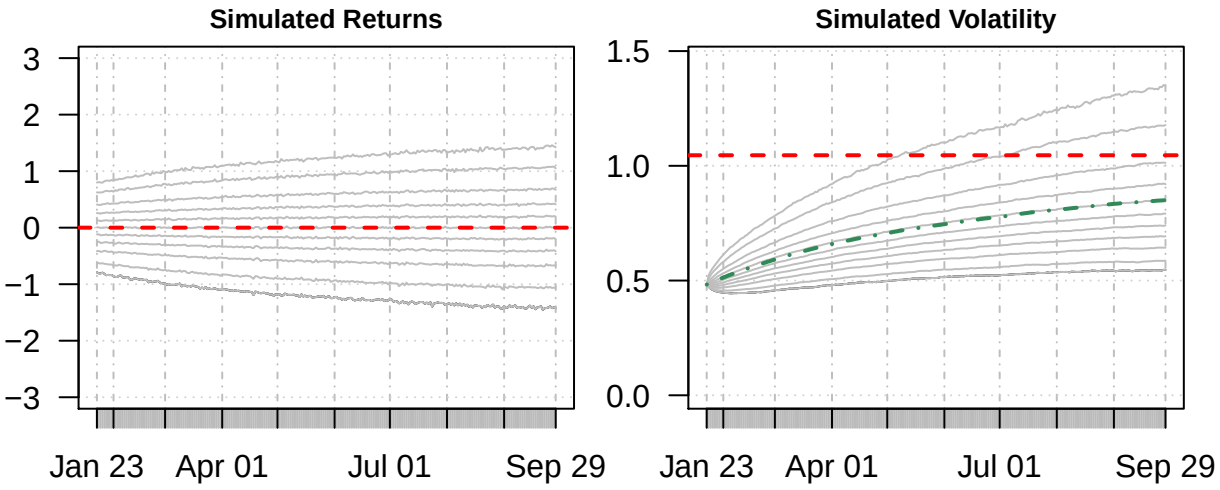


Figure 7.8: Quantiles of the simulated return and volatility distributions forecast in January 22, 2007

simulations) volatility at horizon k that, as expected for GARCH models, tends toward the unconditional mean of volatility. Since January 22, 2007 was a period of low volatility, the graph shows that we should expect volatility to increase at longer horizons. This is also clear from the left plot where the return distribution becomes wider as the forecast horizon progresses.

```
set.seed(9874)

S = 10000  # number of MC simulations
K = 250    # forecast horizon

# create the matrices to store the simulated return and volatility
R     = xts(matrix(sigma*rnorm(S), K, S, byrow=TRUE), order.by=futdates)
Sigma = xts(matrix(sigma, K, S), order.by=futdates)

# iteration to calculate R and Sigma based on the previous day
for (i in 2:K)
{
  Sigma[i,]  = (gcoef['omega'] + gcoef['alpha1'] * R[i-1,]^2 +
                gcoef['beta1'] * Sigma[i-1,]^2)^0.5
  R[i,]  = rnorm(S) * Sigma[i,]
}
```

How would the forecasts of the return and volatility distribution look like if they were made in a period of high volatility? To illustrate this scenario we consider September 29, 2008 as the forecast base. The GARCH(1,1) forecast of volatility for the following day is 3.085% and the distribution of simulated returns and volatilities are shown in Figure 7.9. Since the day was in a period of high volatility, the assumption of stationarity of volatility made by the GARCH model implies that volatility will reduce in the future. This is clear from the return quantiles converging in the left plot, as well as from the declining average volatility in the right plot.

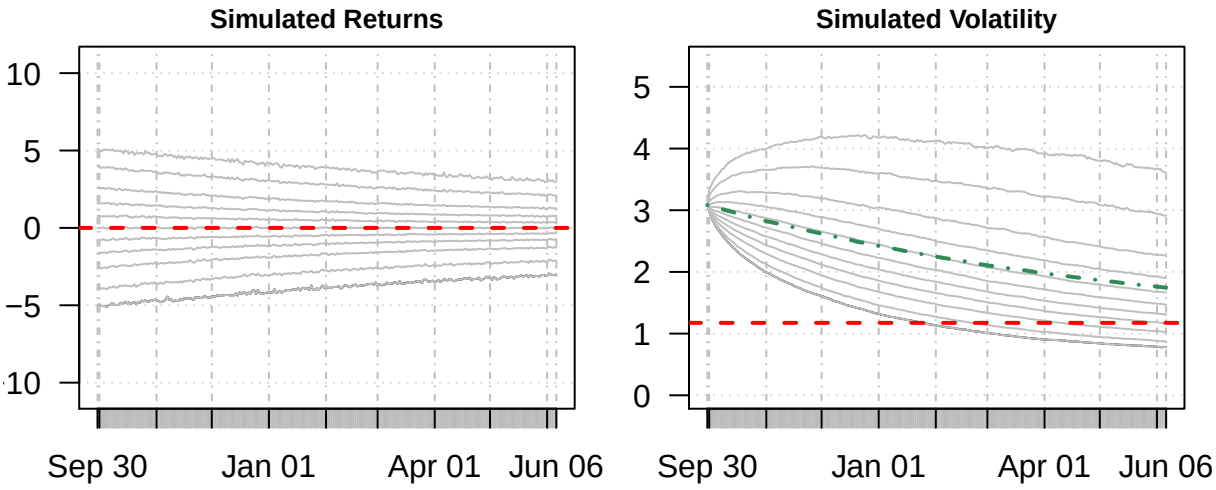


Figure 7.9: Quantiles of the simulated return and volatility distributions forecast in September 29, 2008

The cumulative or multi-period return can be easily obtained by the `R` command `cumsum(Ret)` where `Ret` represents the K by S matrix of simulated one-day returns that the function cumulatively sums over the columns. The outcome is also a K by S matrix with each column representing a possible path of the cumulative return from 1 to K steps ahead. Figure 7.10 shows the quantiles at each horizon k calculated across the S simulated cumulative returns. The quantile at probability 0.01 represents the 99% `VaR` that financial institutions are required to report for regulatory purposes (at the 10 day horizon). The left plot represents the distribution of expected cumulative returns conditional on being on January 22, 2007 while the right plot is conditional on September 29, 2008. The same scale of the y-axis for both plots highlights the striking difference in the dispersion of the distribution of future cumulative returns. Although we saw above that the volatility of daily returns is expected to increase after January 22, 2007 and decrease following September 29, 2008, the levels of volatility in these two days are so different that when accumulated over a long period of time they lead to very different distributions for the cumulative returns.

MC simulations make also easy to calculate `ES`. For each horizon k, `ES` can be calculated as the average of those simulated returns that are smaller than `VaR`. The code below shows these steps for the two dates considered and Figure 7.11 compare the two risk measures conditional on being on January 22, 2007 and on September 29, 2008. Similarly to the earlier discussion, `ES` provides a larger (in absolute value) potential loss relative to `VaR`, a difference that increases with the horizon k.

```
VaR    = xts(apply(Rcum, 1, quantile, probs=0.01), order.by=futdates)
VaRmat = matrix(VaR, K, S, byrow=FALSE)
Rviol  = Rcum * (Rcum < VaRmat)
ES     = xts(rowSums(Rviol) / rowSums(Rviol!=0), order.by=futdates)
```

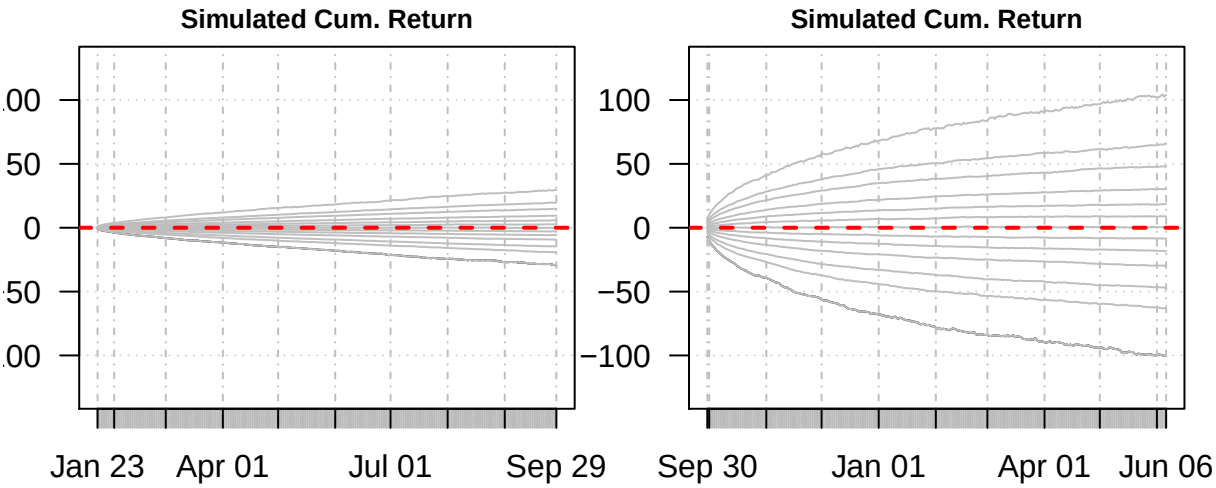


Figure 7.10: Quantiles of the simulated distribution of the cumulative return produced in January 22, 2007 (left) and in September 29, 2008 (right).

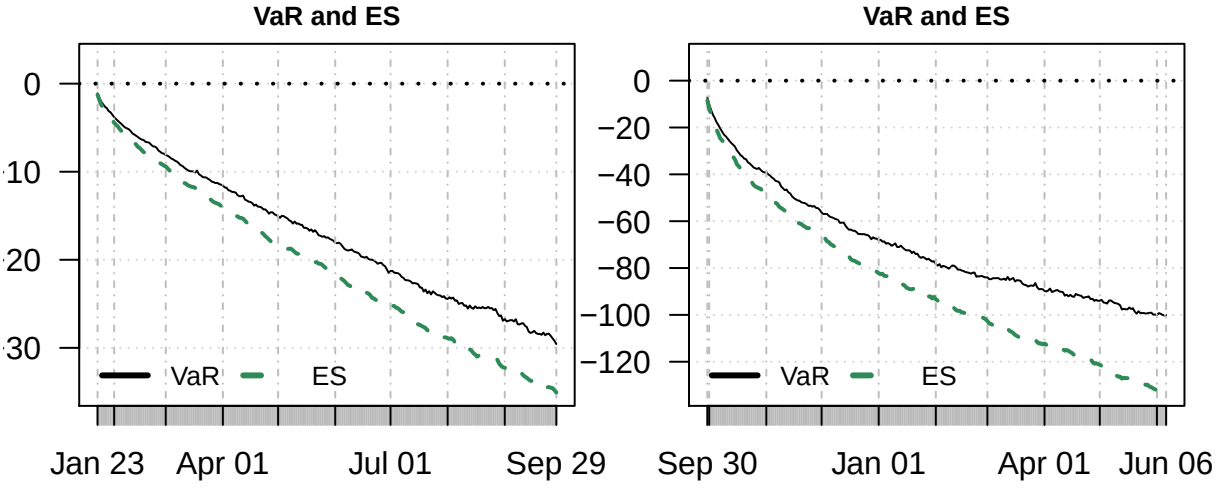


Figure 7.11: VaR and ES for holding periods from 1 to 250 days produced in January 22, 2007 (left) and in September 29, 2008 (right).

7.3.2 Filtered Historical Simulation (FHS)

Filtered Historical Simulation (FHS) combines aspects of the parametric and the HS approaches to risk calculation. This is done by assuming that the volatility of the portfolio return can be modeled with, e.g., EMA or a GARCH specification, while the non-parametric HS approach is used to model the standardized returns $\epsilon_{t+1} = R_{t+1}/\sigma_{t+1}$. It can be considered a simulation method since the only difference with the MC method discussed above is that the random draws from the standardized returns are used to generate simulated returns instead of draws from a parametric distribution (e.g., normal or t). This method might be preferred when the risk manager is uncomfortable making assumptions about the shape of the distribution, either in terms of the thickness of the tails or the symmetry of the distribution. In R this method is implemented by replacing the command `rnorm(S)` that generates a random normal sequence of length S with `sample(std.resid, S, replace=TRUE)`, where `std.resid` represents the standardized residuals. This command produces a sample of length `S` of values randomly taken from the `std.resid` series. Hence, we are not generating *new* data, but we are simply taking different samples of the same standardized residuals so that to preserve their distributional properties.

Figure 7.12 compares the MC and FHS approaches in terms of the expected volatility of the daily returns (i.e., average volatility at each horizon k across the S simulations) and 99% VaR (i.e., 0.01 quantile of the simulated cumulative returns) for the two forecasting dates that we are considering. The expected future volatility by FHS converges at a slightly lower speed relative to MC during periods of low volatility, while the opposite is true when the forecasting point occurs during a period of high volatility. This is because FHS does not restrict the shape of the distribution on the left tail (as MC does given the assumption of normality) so that large negative standardized returns contribute to determine future returns and volatility. Of course, this result is specific to the S&P 500 daily returns that we are considering as an illustrative portfolio and it might be different when analyzing other portfolio returns.

In terms of VaR calculated on cumulative returns we find that FHS predicts lower risk relative to MC at long horizon, with the difference becoming larger and larger as the horizon K progresses. Hence, while at short horizons the VaR forecasts are quite similar, they become increasingly different at longer horizon.

```
# standardized residuals
std.resid  = as.numeric(residuals(fitgarch, standardize=TRUE))
std.resid1 = as.numeric(residuals(fitgarch1, standardize=TRUE))

for (i in 2:K)
{
  Sigma.fhs[i,]  = (gcoef['omega'] + gcoef['alpha1'] * R.fhs[i-1,]^2 +
                    gcoef['beta1'] * Sigma.fhs[i-1,]^2)^0.5
  R.fhs[i,]  = sample(std.resid, S, replace=TRUE) * Sigma.fhs[i,]
  Sigma.fhs1[i,]  = (gcoef1['omega'] + gcoef1['alpha1'] * R.fhs1[i-1,]^2 +
                    gcoef1['beta1'] * Sigma.fhs1[i-1,]^2)^0.5
  R.fhs1[i,]  = sample(std.resid1, S, replace=TRUE) * Sigma.fhs1[i,]
}
```

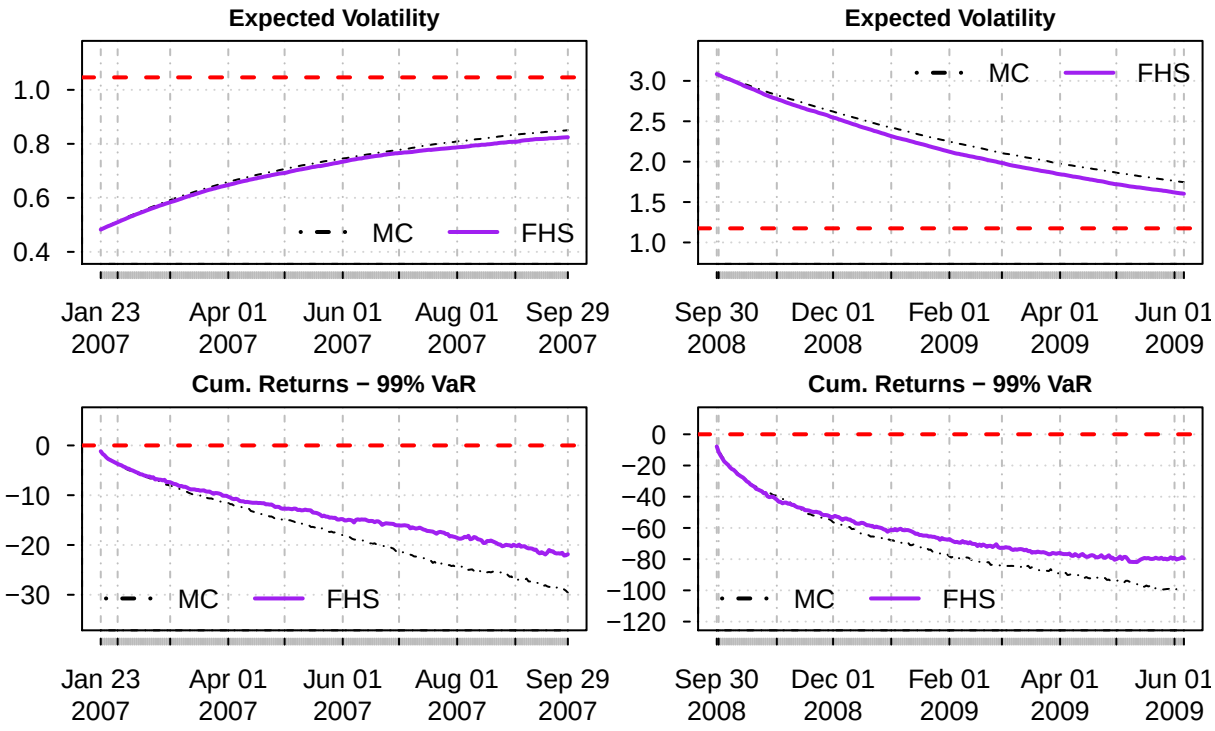


Figure 7.12: Comparison of MC and FHS volatility and 99% VaR produced in January 22, 2007 (left) and in September 29, 2008 (right).

7.4 VaR for portfolios

So far we discussed the simple case of a portfolio composed of only one asset and calculated the Value-at-Risk for such position. However, financial institutions hold complex portfolios that include many assets and expose them to several types of risks. How do we calculate VaR for such diversified portfolios?

Let's first characterize the portfolio return as a weighted average of the individual asset returns, that is,

$$R^p_{t+1} = \sum_{j=1}^{J} w_j R_{t+1,j}$$

where R^p_t represents the portfolio return in day t, and w_j and $R_{j,t}$ are the weight and return of asset j in day t (and there is a total of J assets). To calculate `VaR` for this portfolio we need to model the distribution of R^p_t. If we assume that the underlying assets are normally distributed, then also the portfolio return is normally distributed and we only need to estimate its mean and standard deviation. In the case of 2 assets (i.e., $J = 2$) the expected return of the portfolio return is given by

$$E(R^p_{t+1}) = \mu_{t+1,p} = w_1\mu_{t+1,1} + w_2\mu_{t+1,2}$$

which is the weighted average of the expected return of the individual assets. The portfolio variance is equal to

$$Var(R^p_{t+1}) = \sigma^2_{t+1,p} = w_1^2\sigma^2_{t+1,1} + w_2^2\sigma^2_{t+1,2} + 2w_1w_2\rho_{t+1,12}\sigma_{t+1,1}\sigma_{t+1,2}$$

which is a function of the individual (weighted) variances and the correlation between the two assets, $\rho_{1,2}$. The portfolio *Value-at-Risk* is then given by

$$\begin{aligned} VaR^{0.99}_{t+1,p} &= \mu_{t+1,p} - 2.33\sigma_{t+1,p} \\ &= w_1\mu_{t+1,1} + w_2\mu_{t+1,2} - 2.33\sqrt{w_1^2\sigma^2_{t+1,1} + w_2^2\sigma^2_{t+1,2} + 2w_1w_2\rho_{t+1,12}\sigma_{t+1,1}\sigma_{t+1,2}} \end{aligned}$$

In case it is reasonable to assume that $\mu_1 = \mu_2 = 0$, the $VaR^{0.99}_{t,p}$ formula can be expressed as follows:

$$\begin{aligned} VaR^{0.99}_{p,t} &= -2.33\sqrt{w_1^2\sigma^2_{t+1,1} + w_2^2\sigma^2_{t+1,2} + 2w_1w_2\rho_{t+1,12}\sigma_{t+1,1}\sigma_{t+1,2}} \\ &= -\sqrt{2.33^2w_1^2\sigma^2_{t+1,1} + 2.33^2w_2^2\sigma^2_{t+1,2} + 2 * 2.33^2w_1w_2\rho_{t+1,12}\sigma_{t+1,1}\sigma_{t+1,2}} \\ &= -\sqrt{(VaR^{0.99}_{t+1,1})^2 + (VaR^{0.99}_{t+1,2})^2 + 2 * \rho_{t+1,12}VaR^{0.99}_{t+1,1}VaR^{0.99}_{t+1,2}} \end{aligned}$$

which shows that the portfolio `VaR` in day t can be expressed in terms of the individual `VaR`s of the assets and the correlation between the two asset returns. Since the correlation coefficient ranges between $\pm$ 1, the two extreme cases of correlation implies the following `VaR`:

- $\rho_{t+1,12} = 1$: $VaR^{0.99}_{t+1,p} = -\left(|VaR^{0.99}_{t+1,1}| + |VaR^{0.99}_{t+1,2}|\right)$ the two assets are perfectly correlated and the total portfolio `VaR` is the sum of the individual `VaR`s
- $\rho_{t+1,12} = -1$: $VaR^{0.99}_{t+1,p} = -|VaR^{0.99}_{t+1,1} - VaR^{0.99}_{t+1,2}|$ the two assets have perfect negative correlation then the total risk of the portfolio is given by the difference between the two `VaR`s since the risk in one asset is offset by the other asset, and vice versa.

In the Equations above we added a t subscript also to the correlation coefficient, that is, $\rho_{t+1,12}$ represents the correlation between the two asset conditional on the information available up to that day. There is evidence supporting the fact that correlations between assets might be changing over time in response to market events or macroeconomic shocks (e.g., a recession). In the following Section we discuss some methods that can be used to model and predict correlations.

7.4.1 Modeling correlations

A simple approach to modeling correlations consists of using MA and EMA smoothing similarly to the case of forecasting volatility. However, in this case the object to be smoothed is not the square return, but the product of the returns of asset 1 and 2 (N.B.: we are implicitly assuming that the mean of both assets can be set equal to zero). Denote the return of asset 1 by $R_{t,1}$, of asset 2 by $R_{t,2}$, and by $\sigma_{t,12}$ the covariance between the two assets in day t. We can estimate $\sigma_{12,t}$ by a MA of M days:

$$\sigma_{t+1,12} = \frac{1}{M}\sum_{m=1}^{M} R_{t-m+1,1}R_{t-m+1,2}$$

and the correlation is then obtained by dividing the covariance estimate with the standard deviation of the two assets, that is:

$$\rho_{t+1,12} = \sigma_{t+1,12} / \left(\sigma_{t+1,1} * \sigma_{t+1,2}\right)$$

In case the portfolio is composed of J assets then there are $J(J-1)/2$ asset pairs for which we need to calculate correlations. An alternative approach is to use EMA smoothing which can be implemented using the recursive formula discussed earlier:

$$\sigma_{t+1,12} = (1-\lambda)\sigma_{t,12} + \lambda R_{t,1}R_{t,2}$$

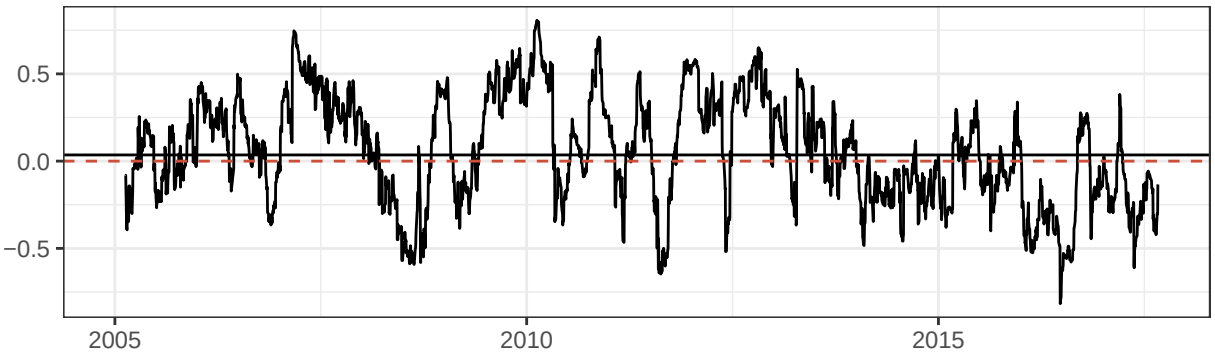


Figure 7.13: Time-varying correlation between GLD and SPY estimated using EMA(0.06).

and the correlation is obtained by dividing the covariance by the forecasts of the standard deviations for the two assets.

To illustrate the implementation in `R` we assume that the firm is holding a portfolio that invests a fraction w_1 in a gold ETF (ticker: `GLD`) and the remaining fraction $1 - w_1$ in the S&P 500 ETF (ticker: `SPY`). The closing prices are downloaded from Yahoo Finance starting in Jan 03, 2005 and ending on Sep 01, 2017 and the goal is to forecast portfolio `VaR` for the following day. We will assume that the expected daily returns for both assets are equal to zero and forecast volatilities and the correlation between the assets using EMA with $\lambda = 0.06$. In the code below `R` represents a matrix with 3190 rows and two columns representing the `GLD` and `SPY` daily returns.

```
data     <- getSymbols(c("GLD", "SPY"), from="2005-01-01")
R        <- 100 * merge(ClCl(GLD), ClCl(SPY))
names(R) <- c("GLD","SPY")
prod     <- R[,1] * R[,2]
cov      <- EMA(prod, ratio=0.06)  # EMA for the product of returns
sigma    <- do.call(merge, lapply(R^2, FUN = function(x) EMA(x, ratio=0.06)))^0.5
names(sigma) <- names(R)
corr     <- cov / (sigma[,1] * sigma[,2])
```

The time series plot of the EMA correlation in Figure 7.13 shows that the dependence between the gold and S&P 500 returns oscillates significantly around the long-run correlation of NA. In certain periods gold and the S&P 500 have positive correlation as high as 0.81 and in other periods as low as -0.82. During 2008 the correlation between the two assets became large and negative since investors fled the equity market toward gold that was perceived as a safe haven during turbulent times. Based on these forecasts of volatilities and correlation, portfolio `VaR` can be calculated in `R` as follows:

```
w1  = 0.5     # weight of asset 1
w2  = 1 - w1  # weight of asset 2
VaR = -2.33 * ( (w1*sigma[,1])^2 + (w2*sigma[,2])^2 +
                  2*w1*w2*corr*sigma[,1]*sigma[,2] )^0.5
```

The one-day portfolio Value-at-Risk fluctuates substantially between -0.64% and -8.01%, which occurred during the 2008-09 financial crises. It is also interesting to compare the portfolio `VaR` with the risk measure if the portfolio is fully invested in either asset. The time series graph below shows the `VaR` for

Figure 7.14: 99% VaR for a portfolio of GLD and SPY.

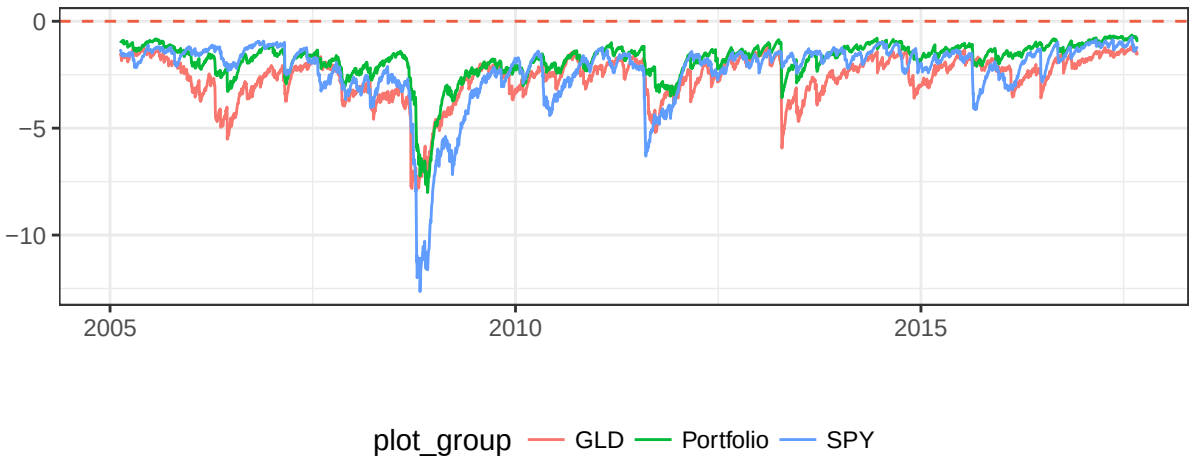


Figure 7.15: Comparison of 99% VaR for a portfolio invested in GLD and SPY, and for a position that is fully invested in GLD or SPY.

three scenarios: a portfolio invested 50% in gold and equity, 100% gold, and 100% S&P 500. Portfolio `VaR` is between the `VaR`s for the individual assets when correlation is positive. However, in those periods in which the two assets have negative correlation (e.g., 2008) the portfolio `VaR` is higher than both individual `VaR`s since the two assets are moving in opposite directions and the risk exposures partly offset each other.

```
VaRGLD = -2.33 * sigma[,1]
VaRSPY = -2.33 * sigma[,2]
```

7.5 Backtesting VaR

In the risk management literature *backtesting* refers to the evaluation/testing of the properties of a risk model based on past data. The approach consists of using the risk model to calculate `VaR` based on the information (e.g., past returns) that was available to the risk manager at that point in time. The `VaR` forecasts are then compared with the actual realization of the portfolio return to evaluate if they satisfy the properties that we expect should hold for a *good* risk model. One such characteristic is that the **coverage** of the risk model, defined as the percentage of returns smaller than `VaR`, should be close to

100*(1-α)%. In other words, if we are testing VaR with α equal to 0.01 we should expect, approximately, 1% of days with violations. If we find that returns were smaller than VaR significantly more/less often than 1%, then we conclude that the model has inappropriate coverage. For the S&P 500 return and VaR calculated using the EMA method the coverage is equal to

```
V = na.omit(lag(sp500daily,1) <= var)
T1  = sum(V)
TT   = length(V)
alphahat = mean(T1/TT)
```

```
[1] 0.011
```

In this case, we find that the 99% VaR is violated as often (0.011) then expected (0.01). We can test the hypothesis that $\alpha = 0.01$ (or 1%) by comparing the likelihood that the sample has been generated by α as opposed to its estimate $\hat{\alpha}$, which in this example is equal to 0.011. One way to test this hypothesis is to define the event of a violation of VaR, that is $R_{t+1} \leq VaR_{t+1}$, as a binomial random variable with probability α that the event occurs. Since we have a total of T days and introducing the assumption that violations are independent of each other, then the joint probability of having T_1 violations (out of T days) is

$$\mathcal{L}(\alpha, T_1, T) = \alpha^{T_1}(1-\alpha)_0^T$$

where $T_0 = T - T_1$. The hypothesis $\alpha = 0.01$ can be tested by comparing this likelihood above at the estimated α and at the theoretical value of 0.01 (more generally, if VaR is calculated at 95% then α is 0.05). This type of tests are called likelihood ratio tests and can be interpreted as the distance between the theoretical value (i.e., using $\alpha = 0.01$) of the likelihood of obtaining T_1 violation in T days and the likelihood based on the sample estimate $\hat{\alpha}$. The statistic and distribution of the test for **Unconditional Coverage** (**UC**) are

$$UC = -2\ln\left(\frac{\mathcal{L}(0.01, T_1, T)}{\mathcal{L}(\hat{\alpha}, T_1, T)}\right) \sim \chi_1^2$$

where $\hat{\alpha} = T_1/T$ and χ_1^2 denotes the chi-square distribution with 1 degree-of-freedom. The critical values at 1, 5, and 10% are 6.63, 3.84, and 2.71, respectively, and the null hypothesis $\alpha = 0.01$ is rejected if LR^{UC} is larger than the critical value. In practice, the test statistic can be calculated as follows:

$$-2\left[T_1\ln\left(\frac{0.01}{\hat{\alpha}}\right) + T_0\ln\left(\frac{0.99}{1-\hat{\alpha}}\right)\right]$$

In the example discussed above we have $\hat{\alpha} = 0.011$, $T_1 = 77$, and T is 6942. The test statistic is thus

```
UC  = -2 * ( T1 * (log(0.01/alphahat))
             + (TT - T1) * log(0.99/(1-alphahat)))
```

Since 0.802 is smaller than 3.84 we do not reject the null hypothesis at 5% significance level that $\alpha = 0.01$ and conclude that the risk model provides appropriate coverage.

While testing for coverage, we introduced the assumption that violations are independent of each other. If this assumption fails to hold, then a violation in day t has the effect of increasing/decreasing the probability of experiencing a violation in day $t+1$, relative to its unconditional level. A situation in which this might happen is during financial crises when markets enter a downward spiral which is likely to lead to several consecutive days of violations and thus to the possibility of underestimating risk. The empirical evaluation of this assumption requires the calculation of two probabilities: 1) the probability

of having a violation in day t given that a violation occurred in day $t-1$, and 2) the probability of having a violation in day t given that no violation occurred the previous day. We denote the estimates of these conditional probabilities as $\hat{\alpha}_{1,1}$ and $\hat{\alpha}_{0,1}$, respectively. They can be estimated from the data by calculating $T_{1,1}$ and $T_{0,1}$ that represent the number of days in which a violation was preceded by a violation and a no violation, respectively. In R we can determine these quantities as follows:

```
T11 = sum((lag(V,1)==1) & (V==1))
T01 = sum((lag(V,1)==0) & (V==1))
```

where we obtain that $T_{0,1} = 77$ and $T_{1,1} = 0$, with their sum equal to T_1. Similarly, we can calculate $T_{1,0}$ and $T_{0,0}$ and the estimates are NA and NA, respectively, that sum to T_0. Since we look at violations in two consecutive days we lose one observation and our total sample size is now $T - 1 = 400$. We can then calculate the conditional probabilities of a violation in a day given that the previous day there was no violation as $\hat{\alpha}_{0,1} = T_{0,1}/(T_{0,1} + T_{1,1})$ while the probability of a violation in two consecutive days, that is, $\hat{\alpha}_{1,1} = 1 - \hat{\alpha}_{0,1} = T_{1,1}/(T_{0,1} + T_{1,1})$. Similarly, we can calculate $\hat{\alpha}_{1,0}$ and $\hat{\alpha}_{0,0}$. For the daily S&P 500 introduced earlier, the estimates are $\hat{\alpha}_{1,1} = 0$ and $\hat{\alpha}_{0,1} = 1$, and $\hat{\alpha}_{1,0} =$ NA and $\hat{\alpha}_{0,0} =$ NA. While we have an overall estimated probability of a violation of 1.1%, this probability is equal to 0% after a day in which the risk model was violated and NA% following a day without a violation. Since these probabilities are significantly different from each other, we should conclude that the violations are not independent over time. To make this conclusion in statistical terms, we can statistically test the hypothesis of independence of the violations by stating the null hypothesis as $\alpha_{0,1} = \alpha_{1,1} = \alpha$ which can be tested using the same likelihood ratio approach discussed earlier. In this case, the statistic is calculated as the ratio of the likelihood under independence, $\mathcal{L}(\hat{\alpha}, T_1, T)$, relative to the likelihood under dependence, which we denote by $\mathcal{L}(\hat{\alpha}_{0,1}, \hat{\alpha}_{1,1}, T_{0,1}, T_{1,1}, T)$ which is given by

$$\mathcal{L}(\hat{\alpha}_{0,1}, \hat{\alpha}_{1,1}, T_{0,1}, T_{1,1}, T) = \hat{\alpha}_{1,0}^{T_{1,0}}(1-\hat{\alpha}_{1,0})^{T_{1,1}}\hat{\alpha}_{0,1}^{T_{0,1}}(1-\hat{\alpha}_{0,1})^{T_{0,0}}$$

the test statistic and distribution for the hypothesis of **Independence** (**IND**) in this case is

$$IND = -2\ln\left(\frac{\mathcal{L}(\hat{\alpha}, T_0, T)}{\mathcal{L}(\hat{\alpha}_{0,1}, \hat{\alpha}_{1,1}, T_{0,1}, T_{1,1}, T)}\right) \sim \chi_1^2$$

and the critical values are the same as for the UC test. The numerator of the IND test is the likelihood in the denominator of the UC test and it is based on the empirical estimate of α rather than the theoretical value α. The value of the IND test statistic is NA which is smaller relative to the critical value at 5% so that we do not reject the null hypothesis that the violations occur independently.

Finally, we might be interested in testing the hypothesis $\alpha_{0,1} = \alpha_{1,1} = 0.01$ which tests jointly the independence of the violations as well as the coverage which should be equal to 1%. This test is referred to as **Conditional Coverage** and the test statistic is given by the sum of the previous two test statistics, that is, $CC = IND + UC$ and it is distributed as a χ_2^2. The critical values at 1, 5, and 10% in this case are 9.21, 5.99, and 4.61, respectively.

`R` commands

Exercises

1. Exercise 1
2. Exercise 2

Bibliography

Adrian, T., Crump, R. K., and Vogt, E. (2015). Nonlinearity and flight to safety in the risk-return trade-off for stocks and bonds. *Federal Reserve of New York working paper*.

Albert, J. and Rizzo, M. (2012). *R by Example*. Springer Science & Business Media.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, pages 987–1007.

Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56.

Newey, W. K. and West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55:703–708.

Shiller, R. J. (2015). *Irrational exuberance*. Princeton University Press.

Stock, J. H. and Watson, M. W. (2010). *Introduction to Econometrics*. Addison Wesley.

Tsay, R. S. (2005). *Analysis of financial time series*, volume 543. John Wiley & Sons.

Welch, I. and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*, 21(4):1455–1508.

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer Science & Business Media.

Wooldridge, J. M. (2015). *Introductory Econometrics: A Modern Approach*.

Zuur, A., Ieno, E. N., and Meesters, E. (2009). *A Beginner's Guide to R*. Springer Science & Business Media.